Computer Science for All Ages

Michael Gannotti

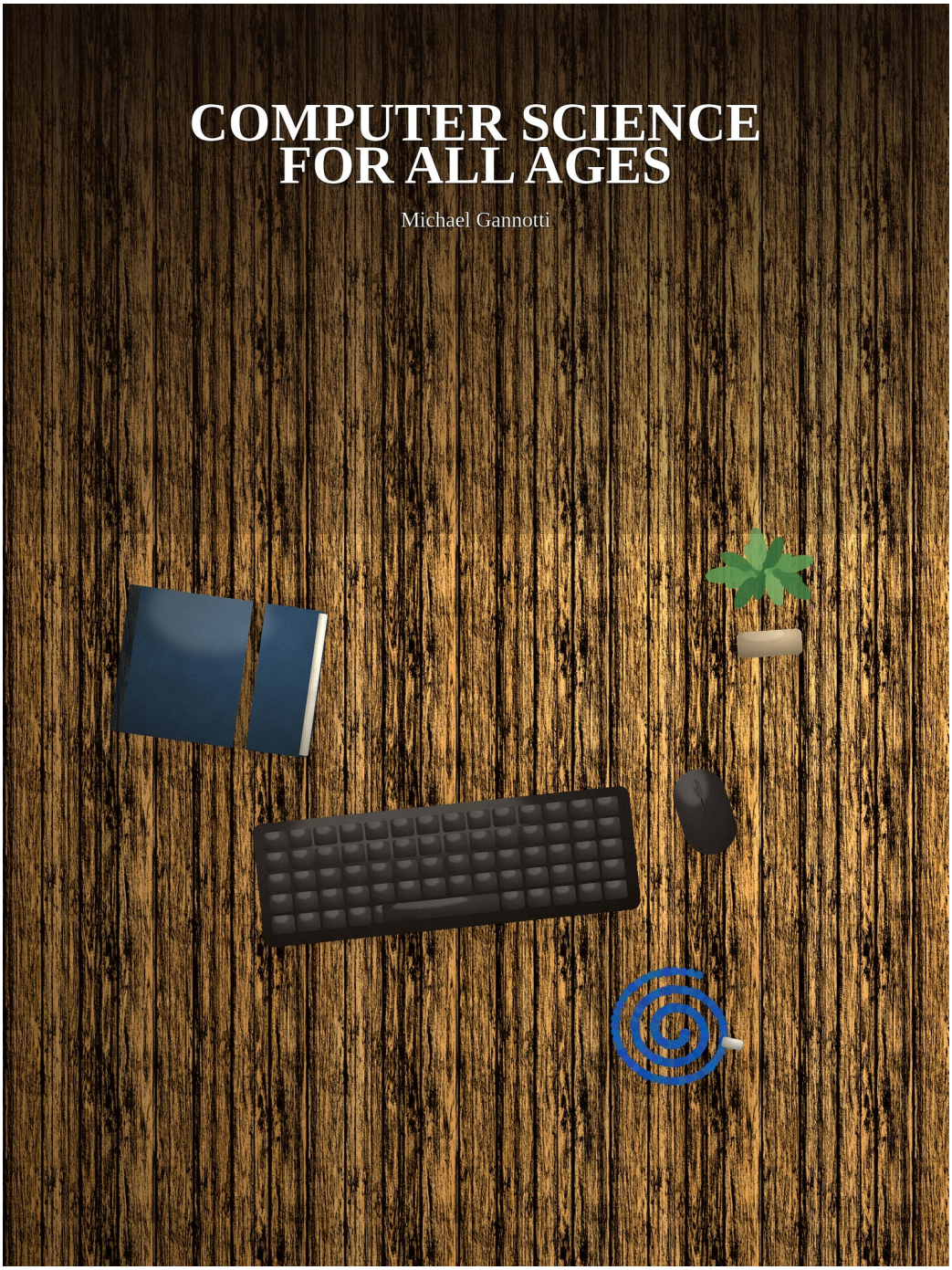

Figure 1: Cover

Computer Science for All Ages

Michael Gannotti

This illustrated edition is the teaching-manual of 1 September 2026. Endnotes run in one series. Chapter-opening images are original still-lifes, not portraits of persons or children. Documents still constrain the facts.

Contents

Welcome

This book exists because the computing hour at your table is the whole program. There is no department down the hall and no specialist waiting after lunch. There is you, a student, and today's idea — a device to look at, a file to save, a try that might not run the first time, a question worth waiting for. That is enough, if you know what to do with the hour.

I wrote this for a capable, busy, willing parent. You may be teaching two ages at once. You may be fitting computing between a job, a toddler, and a grocery run. You may have loved computers, or you may remember them as a fog of menus and error messages. Many adults feel rusty around computers when they sit down to teach. That feeling is common. It is not a verdict. We will move on from it. This book will make you fluent enough in this week's idea to notice a wrong explanation and ask a good question — without taking over the keyboard.

You do not need to be a computer scientist. You do need to understand this week's idea well enough to hear "the computer is thinking" as a prior idea doing work, not as a cute slip. You need a session shape you can run on a Tuesday. You need a few sentences that actually help. That is the job. A path through computing — from first-grade looking-at-devices through a college-ready sequence that includes both computer use and computer science — is possible at a kitchen table. The path is not a personality trait and it is not a percentile. It is a sequence of ideas, practiced until they hold, with you in the chair.

About 3.4 percent of U.S. students ages 5–17 were homeschooled in 2022–23 — roughly 1.765 million children.[1] That figure is context, not a ranking. You are

[1] Ellena Sempeles and Jiashan Cui, *Parent and Family Involvement in Education: 2023*, NCES 2024-113 (Washington, DC: National Center for Education Statistics, September 2024), Table A-6: 3.4 percent homeschooled, approximately 1,765,000 students, ages 5–17 with a K–12 grade equivalent, 2022–23. https://nces.ed.gov/pubs2024/2024113.pdf. Access date for URLs in these notes:

one of those tables. Homeschooling is legal in all fifty states and the District of Columbia. The paperwork is not one load: Texas asks almost nothing of the state agency; New York asks for a written plan and regular reports.[2] There is no national homeschool diploma and no national computer-science credit a registrar already files.[3] In the last federal tables that published the row, computer science at home was listed under Science as "computer science (e.g., computer programming)": taught that year in 16 percent of households overall, and ever in grades 9–12 in 37 percent.[4] Those numbers describe a minority subject. They do not describe you. The titles that travel — Computer Science, Digital Literacy, Technology Applications, Keyboarding, AP Computer Science Principles, AP Computer Science A — are names a stranger can read. The hour in front of you is the work.

Computer science is why and how computers work. Typing, office software, and chatting with a model are computer *use*. Both belong at the table. They are not the

---

1 September 2026. Latest federal count this book uses; not a 2026 national total. A neighboring 5.2 percent received instruction at home (homeschooled or full-time virtual) and is a different bucket.

[2]Texas Education Agency, "Home Schooling," https://tea.texas.gov/families-and-students/finding-school-your-child/home-schooling. New York State Education Department, "Home Instruction Questions and Answers," https://www.nysed.gov/nonpublic-schools/home-instruction-questions-and-answers. Homeschooling is legal in all fifty states and D.C.; these two pages illustrate a light-process and a heavier-process pattern, not a national template. This book is not legal advice.

[3]Texas Education Agency, "Home Schooling": "The State of Texas does not award a diploma to students that are home schooled." New York State Education Department, "Home Instruction Questions and Answers": a high school diploma "may only be awarded to a student enrolled in a registered secondary school." North Carolina Division of Non-Public Education, "Home School Records Retention & Diplomas FAQs," https://www.doa.nc.gov/divisions/non-public-education/home-schools/faqs/records-retention-diplomas: the State does not issue a diploma or a transcript; the home-school chief administrator does. NCAA Eligibility Center, *Home School Toolkit 2025–26*, http://fs.ncaa.org/Docs/eligibility_center/Student_Resources/Home_School_Toolkit.pdf, treats proof of graduation as a family- or umbrella-issued artifact. None of Texas, North Carolina, Pennsylvania, Virginia, or New York requires computer science or technology as a homeschool subject.

[4]Jiashan Cui and Rachel Hanson, *Homeschooling in the United States: Results from the 2012 and 2016 Parent and Family Involvement Survey*, NCES 2020-001 (Washington, DC: National Center for Education Statistics, 2019), Table 9: among K–12-equivalent homeschoolers taught *that year*, computer science (e.g., computer programming) 16 percent overall (K–2 11 percent, flagged by NCES for interpret-with-caution; grades 3–5 18 percent; 6–8 17 percent; 9–12 17 percent). Table 8: grades 9–12 *ever* taught during home instruction, computer science 37 percent. The questionnaire listed CS under Science. There is no PFI item for keyboarding or digital literacy. The 2023 First Look did not republish the subject-taught tables. https://nces.ed.gov/pubs2020/2020001.pdf.

same thing.[5] A year of slides is a year of use. A year of programs the child can trace is computer science. This book owns both columns and labels them honestly.

The path through these pages is hardware, then operating systems, then networks, then the internet, then productivity as *use*, then first programs, then AI as a topic — how a pattern-in, pattern-out system works, not a year of chatting. Age bands 5–10, 11–14, and 15–18 live inside each chapter, not as twelve thin grade books. You place by skill, not by birthday.

What a good computing hour looks like

You sit down already knowing today's idea. Something is on the table: a keyboard, a cable, a notebook, a short try that could come out otherwise. The child looks. Then they try — a key press, a save, a small program, a packet of cards across a hallway. They record what they expected and what happened, in their own hand. You ask one good question — not "what is the vocabulary word?" but "what did you expect it to do, and what did it do instead?" Then you wait. Three seconds feels long. It is the work. Debugging can take longer still. The hour ends with one short task they do alone: a spoken sentence, a named file they can find tomorrow, a trace with the screen off. You stop talking sooner than feels polite. The child tries. You hold the key. That is the hour. The next long piece of this front matter, *The Computing Hour*, will teach it in full. Later chapters will not reinvent it.

[5] K–12 Computer Science Framework (2016), 12–13: computer science is "the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society"; computer literacy, educational technology, digital citizenship, and information technology "are distinguished from computer science because they are focused on using computer technologies rather than understanding why they work and how to create those technologies." https://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf. Computer Science Teachers Association, *CSTA K–12 Computer Science Standards* (2026), DOI 10.1145/3820482: computer science is not "using technology tools like word processors, slide decks, or generative AI tools"; programming is essential and is not the whole of CS. California Department of Education, *Computer Science Standards for California Public Schools* (adopted 6 September 2018), 15–16: computer science is not learning how to type, use office software, or build or repair computers.

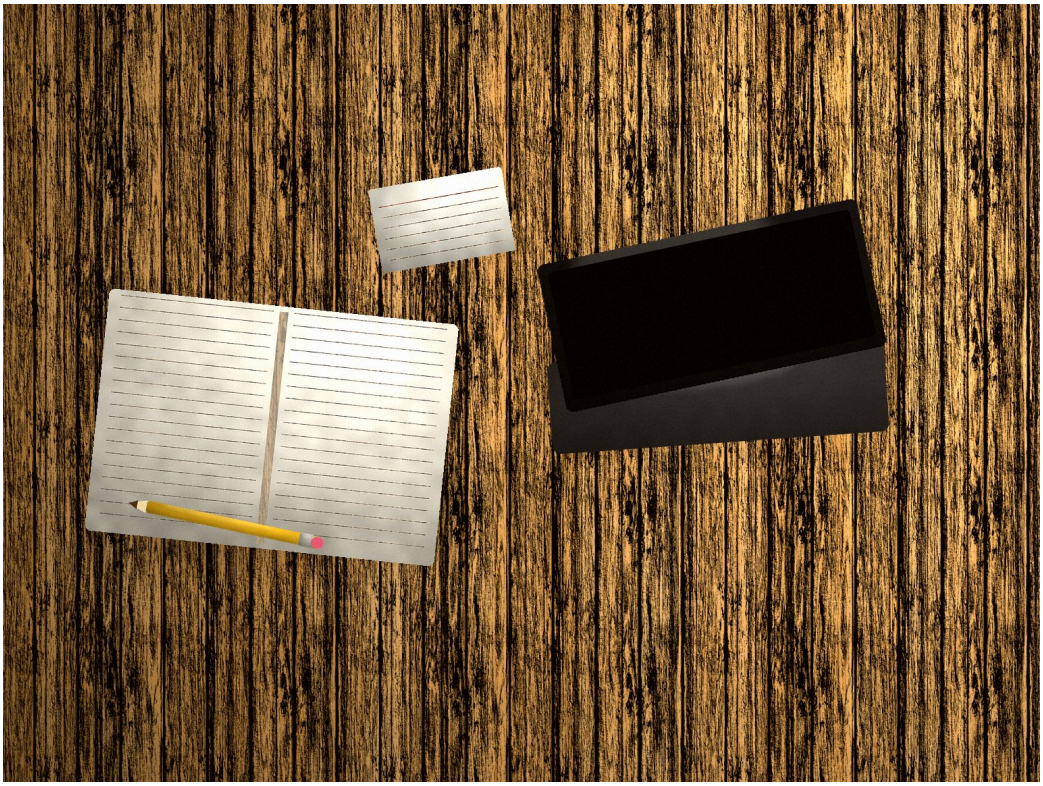

Figure 2: A kitchen-table computing hour

Look first, then the word. A vocabulary list is not a year. An hour that is only clicking, or only a word list, is not the hour. The chapters will give you wording. The hour will stay this shape.

What you will actually get

Each teaching chapter does eight jobs, always in the same order, so you are never hunting for the move.

You will learn why this week's idea is worth the struggle — what it unlocks later. You will understand the idea yourself, in plain language, with one everyday picture and one precise picture, and with the wrong explanations you should be able to hear. You will get a session you can run this week, including exact wording, named try-its, and a debug box with this week's questions. You will get practice that builds

learning, not a slog of coverage. Your student will get a short section of their own. If it isn't clicking, you will get three diagnostics and a next move for each, with no shame in the room. Tools, including AI, stay optional and adult-side. And you will get a plain checklist for "done enough," so you can place by skill rather than by birthday.

The teaching chapters follow the work, not twelve thin grade labels: hardware; operating systems; networking; the internet; productivity software as use; first programs; AI as a topic; then how to choose and sequence. Records and resources come last.

This week you can learn the session shape and today's idea well enough to hear a wrong explanation. Today the child can look at one device, try one small thing, and say one true thing about it without you filling the silence.

What this book will not do

This book will not hand you 180 days of worksheets. A year of photocopies is not a teaching method, and I will not pretend it is. Coverage is a map. Depth is the hour.

It will not sell you a curriculum. Later, a short resources chapter names common programs by fit — parent load, style, how they place a student — so you can choose. Scratch, CS Unplugged, a Code.org course, a university intro, a free office suite: tools. None of them is this book, and this book is not a catalog in disguise. A family may combine. Combining does not turn a brand into a course name. The transcript still says Computer Science or Digital Literacy.

It will not promise a score. There is no guaranteed percentile, no guaranteed AP number, and no guaranteed college letter inside these pages. What this book can promise is a path: the ideas in order, at the skill the child actually has, until they can do the next idea unaided. There is no national homeschool computer-science score in these pages to wave.[6]

[6]National Center for Education Statistics, Technology and Engineering Literacy (TEL) 2018: grade 8 mean 152; 46 percent at or above NAEP Proficient. NCES states that "the NAEP Proficient achievement level does not represent grade-level proficiency as determined by other assessment standards." TEL is not a computer-science test; the National Assessment Governing Board voted to sunset it on 4 August 2023. There is no current NAEP computer-science assessment and no nationally representative homeschool CS score.

It is not a Code.org tract, for or against. The *K–12 Computer Science Framework* and the CSTA standards are maps of outcomes. They are not a curriculum, not a federal law, and not a homeschool mandate.[7] Code.org helped write a map. That is authorship. It is not a statute. A publisher's "grade 6" book is a scope, not a legal grade. AP Computer Science Principles and AP Computer Science A sit beyond the floor, as named courses a family may choose — not as the kitchen-table default.

And this is not a sequel. *Autonomous AI and Education*, *Science for Homeschooling*, *Critical Thinking for Life*, and *Mathematics for Homeschooling* are other books. AI lives here as a topic in one teaching chapter, and as one computing-hour box in this front matter, then a short tools box in each chapter. Using a chatbot is use. Understanding how a model works is computer science. Those two jobs stay distinct.

This is not a book that lectures your child about you, or you about your character. The student is a person, not a percentile. When this book speaks to them, it speaks with respect. No baby-talk. No research sermon.

The promise

If you only remember one sentence, remember this: a path through computing is the promise.

The work is to take this child through computing — looking at devices, files and accounts, networks as a model, the internet as a system, productivity as honest use, a first program they can trace, AI as a topic a registrar can read — without skipping the looking because they are "good with phones," and without parking them in a typing year because a catalog printed Computer Science on the cover. You will sometimes slow down. You will sometimes skip ahead. Both are teaching. Birthday is not placement.

Three things have to hold. You understand this week's idea well enough to hear

[7] K–12 Computer Science Framework (2016), 14: the concepts and practices "are not specific, measurable performance expectations in the form of standards, nor are they detailed lesson plans and activities in the form of curriculum." FAQ: "The framework statements are not standards"; "States and districts should make the final decision on the documents they use." https://k12cs.org/faq/. CSTA 2026: the Standards describe what students should know and be able to do and do not prescribe curriculum, materials, or assessments. Code.org sat on the Framework steering committee; that is authorship, not a statute.

a wrong explanation. The child does the looking, the trying, and the debugging before the tool. Any helper — including an AI agent — stays on your side of the table: a supplement you host and constrain, not a substitute for the child's program or the talk after the try, and not a school-safe product by default.

You can do this. You do not have to know next year's idea today. You have to know this week's idea well enough to sit still while they struggle, then ask one good question. Start here. Read *How to Use This Book This Week*, then the one-page list of five things, then *The Computing Hour*. After that, open the chapter that matches the skill in front of you. You will know more after one chapter than you know this morning. Your child will have something to try today. We can do this.

How to Use This Book This Week

Start at the skill in front of you, not on page one of Grade 1 because a catalog, a birthday, or a well-meant relative said so.

This book is a handbook you open to this week's topic, not a novel you read cover to cover. The teaching chapters run: hardware; operating systems; networking; the internet; productivity software; first programs; AI as a topic; then how to choose and sequence. Records and resources come last. Age bands 5–10, 11–14, and 15–18 live *inside* each topic chapter. They tell you how an idea usually sits at that age. They do not tell you where *this* child sits.

This book does not replace the program already on the shelf. If Scratch, CS Unplugged, a Code.org course, a high-school text path, or an office suite is already working, keep it. Use this book to hear a wrong explanation, to run the hour, and to name the credit a stranger can read. Combining a program honestly is ordinary. Combining does not turn a brand into a course name. The transcript still says Computer Science or Digital Literacy.

Pick the chapter by skill, not by birthday

Open the chapter you think is right. Skip to **What "done enough" looks like** at the end of the *previous* chapter, or to **If it isn't clicking** in the one you opened. If the child can already do those checks unaided, you are too early. If the checks from two chapters back are still failing, drop back.

If they still live in naming parts and saving one file — if a word list is noise and a keyboard, a cable, and a named folder are the lesson — they are still in the 5–10 band of hardware and operating systems, whatever their age. Young children already notice a great deal. What they can do depends on what they have already

been invited to look at, not on a birthday.[8] A short sit with a device is in range. A network topology diagram is not, yet.

If they cannot yet say what they expected a program to do — and what it did instead — they are still in a first-programs stage, whatever the transcript year. Middle-grade students can debug when someone teaches the looking.[9] They are not computer scientists. A fifteen-year-old who still treats a green run as understanding is not "behind in AP." They are still learning to trace.

If they can name what they expected, point at a record, and find a file tomorrow, and they are still in a "grade 4" reader only because the cover says so, skip ahead. Instruction for a six-year-old need not be "first grade"; it should match what the student actually needs and already knows.[10] A publisher's "grade 6" book is a scope, not a legal grade. Use this book's checklists, or a placement test from the program you own, and then teach.

Age is time-on-task and prior knowledge, not a native-coder window. A fifteen-year-old who never had the early hour is behind in practice, not locked out.[11] Plan the band. Then sit down.

---

[8]National Academies of Sciences, Engineering, and Medicine, *Cultivating Interest and Competencies in Computing: Authentic Experiences and Design Factors* (Washington, DC: National Academies Press, 2021), policymakers' brief: without early opportunities, few students will be positioned to thrive in high-school computer science — a pipeline and opportunity claim, not a critical-period claim. Access date for URLs in these notes: 1 September 2026. What a child can do at a given age depends on prior invitations to look, try, and debug, not on a birthday.

[9]David Weintrop and Uri Wilensky, "Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms," *ACM Transactions on Computing Education* 18, no. 1 (2017), Article 3: high-school novices in either modality learned in a five-week introductory unit. The paper is high school, not ages 5–10, and not a homeschool trial. It is evidence that a late start is time-on-task, not a closed window.

[10]New York State Education Department, "Home Instruction Questions and Answers," https://www.nysed.gov/nonpublic-schools/home-instruction-questions-and-answers: instruction for a six-year-old need not be "first grade"; "instruction should be geared to the level appropriate to the student's needs and previous level of achievement."

[11]Weintrop and Wilensky, "Comparing Block-Based and Text-Based Programming"; Caitlin Kelleher, Randy Pausch, and Sara Kiesler, "Storytelling Alice Motivates Middle School Girls to Learn Computer Programming," CHI 2007: middle-school students who had not been programmers learned basic constructs in a short intervention. Age in this book is a planning band, not a native-coder window.

Two ages at one table

Two children may need two chapters. That is ordinary. The session shape in *The Computing Hour* still holds; the idea on the table changes. You may run a twenty-minute name-the-parts sit with one child and a forty-five-minute first-program talk with the other. You do not need two personalities. You need two first problems.

You can share a warm-up. You cannot share a "done enough" checklist. The younger child saves one named file. The older one traces a loop with the screen off. Same kitchen. Two skills.

Maps are maps

The *K–12 Computer Science Framework* and the CSTA standards are maps of outcomes. They are not a curriculum, not a federal requirement, and not a homeschool mandate.[12] California, Texas, and Virginia print their own maps, and they do not agree about course titles or year. That is not a problem for you to solve. It is a reason not to treat any one of them as national law.

Computer science is why and how computers work. Typing, office software, and using a chatbot are use. Productivity software is in this book because families need it. Its transcript title is never Computer Science. A stranger who reads "Computer Science" and finds only slides has been misled. Name the year what it was.

How you use the parent half

Most of each teaching chapter is for you. Read **For the parent: understand it yourself** *before* the lesson, not over the child's shoulder.

Why this matters tells you what this idea unlocks. **For the parent: understand it yourself** gives one everyday picture, one precise picture, and three to five wrong explanations you should be able to hear. Sit with those. The five-minute warm-up at the end of that section is how you stay out of their working memory.

[12] K–12 Computer Science Framework (2016), 14 and FAQ: framework statements "are not standards" and "do not address individual grade-level granularity"; states and districts decide which documents to use. https://k12cs.org/faq/. Computer Science Teachers Association, *CSTA K–12 Computer Science Standards* (2026), DOI 10.1145/3820482: the Standards do not prescribe curriculum, instructional materials, or assessments. Neither document is homeschool law.

How to teach it this week assumes the session shape from *The Computing Hour*. It will not rebuild the hour. It will give you this week's named try-its, the wording for *this* idea, and a debug box: an opening question about what they expected, a few follow-ups, how to wait, and what a stuck silence usually means. Later chapters supply this week's questions. They do not reinvent the shape.

Practice that actually builds learning names the try-its: time, materials, safety and accounts, the fun, the skill. A kitchen object can motivate. It does not replace the try or the unaided sentence. **Tools, including AI** is optional, short, and for the adult. The rules live once in *The Computing Hour*. **What "done enough" looks like** is how you leave: unaided work, not a perfect Tuesday. Placement is by skill, not birthday.

You do not have to read the whole chapter tonight. You do have to read the parent half of *this week's* idea before you sit down with the child.

The five-minute parent warm-up

Make this a habit.

Five minutes. Child not yet in the chair. Phone face down.

1. Read today's idea until you can say it in one sentence.
2. Look at one object yourself — a cable, a save dialog, a short script. Ask, out loud, what would count as a result. Stay off the keyboard until you have looked.
3. Glance at the "wrong explanations you should be able to hear." Name the one you would have given at fourteen.
4. Write one sentence you will actually say. Not a speech. Example: "What did you expect it to do?" Or: "Where did you put it, and how will you find it tomorrow?"
5. Close the book to the student page. You are ready.

That five minutes is how you stay out of their working memory. If you are learning the idea *while* they are stuck, you will talk too much. Prepare first. Then sit still.

How the student uses “For the student”

Every teaching chapter includes a short section written to the student, not about them. One or two pages. Warmer. Direct. What the idea is, a tiny worked example, two tries, an “explain it back” prompt, and one challenge. Hand it over after your short model, not instead of it. You stay in the room. The challenge is optional that day.

The student page is not something to send off with an unsupervised chatbot. The attempt is still theirs. You still hold the key.

When to skip ahead

Skip ahead when this chapter’s “done enough” checklist is already true *unaided*. Slow down when the same wrong explanation repeats after a clear looking and a real attempt. **If it isn’t clicking** will give you three likely causes and a next move. Missing pictures or missing talk are skill problems, not character problems.

Diagnose the looking before you buy the high-school survey. If you need a human tutor, that is a normal high-school plan, not a failure of the kitchen.

For this week: pick the chapter by skill. Do the five-minute warm-up. Run the hour as *The Computing Hour* describes it. Let the student page be theirs. Stop talking sooner than you want to. That is how you use the book.

If You Only Remember Five Things

Keep this page. The chapters will add wording, try-its, and this week's questions. They will not replace these.

1. Computer science is why and how computers work. Typing and chatting with a model are use. Teach both. Label them honestly. A year of slides is Digital Literacy. A year of programs the child can trace is Computer Science. Hardware, files, networks, and a first program sit in the science column. Keyboarding and office software sit in the use column. The two columns sit at the same table. They are not the same credit.[13]

2. The child tries before the tool. You ask one good question and wait. Three seconds. Maybe longer when they are staring at a script or a save dialog. The silence is the work.[14] "What did you expect it to do?" is closer than "what is the

[13]K–12 Computer Science Framework (2016), 12–13: computer literacy, digital citizenship, educational technology, and information technology "are distinguished from computer science because they are focused on using computer technologies rather than understanding why they work and how to create those technologies." Access date for URLs in these notes: 1 September 2026. https://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf. Computer Science Teachers Association, *CSTA K–12 Computer Science Standards* (2026), DOI 10.1145/3820482: computer science is not "using technology tools like word processors, slide decks, or generative AI tools." California Department of Education, *Computer Science Standards for California Public Schools* (2018), 15–16: computer science is not typing or office software. NCAA High School Review Committee, *Policies and Procedures 2026–27*, Appendix A: keyboarding, office applications, and usage of generative AI do not contribute to computer-science course approval; operating systems, networks, programming, and AI development can.

[14]Robert J. Stahl, "Using Think-Time and Wait-Time Skillfully in the Classroom," ERIC Digest ED370885 (May 1994), https://files.eric.ed.gov/fulltext/ED370885.pdf: think-time is "a distinct period of uninterrupted silence"; a slow three seconds is a useful classroom minimum, not a magic cliff. Student task-completion work-time — debugging, hunting a file — can honestly run longer. Stahl is general classroom, not a computer-science trial. Karen Brennan and Mitchel Resnick,

vocabulary word?" Struggle before rescue: ask, wait, hint, then model. You do not finish the function. You do not grab the keyboard.

3. A green run is not understanding. A program that runs is performance. A program the child can trace with the screen off is the skill. A fluent paste from a model can look like mastery at 9 p.m. It often is not still there in the morning.[15] If they cannot say what they expected and what happened, they copied a step. The unaided exit is the check this hour trusts.

4. One path. An honest transcript. Coverage is a map. Depth is the hour. You cannot do every standard. Combine a program if it already works; the transcript still says Computer Science or Digital Literacy, never a brand.[16] You will sometimes slow down. You will sometimes skip ahead. Both are teaching. Birthday is not placement. A path through computing is the promise. A percentile is not.

5. You host the account. Ages 5–10 work from scripts and talk with you, not from a live open chat. The child looks, tries, and debugs. Any helper stays on your side of the table. Under thirteen, you hold the login. One family computer is enough.[17] A second machine is optional convenience, not a moral upgrade. The kitchen is not a hacking lab.

The child tries before the tool. You hold the key. A helper may explain this week's idea *to you* from a named page. It does not sit in the chair during the attempt.

"New frameworks for studying and assessing the development of computational thinking," AERA 2012: testing and debugging is a practice, developed through trial and error and the support of others.

[15] James Prather et al., "The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers," ICER 2024, arXiv:2405.17739: among 21 CS1 students given Copilot and ChatGPT on one problem, 20 finished a working program; struggling students often left with an illusion of competence. This is a useful study, not a promise that every home will see the same result. College Board, AP Computer Science Principles 2026–27 AI guidance: the student must be able to explain the code on the exam.

[16] K–12 Computer Science Framework (2016) FAQ: the framework statements are not standards and not curriculum. CSTA 2026: the Standards do not prescribe curriculum. Transcript titles a stranger can read: Computer Science; Digital Literacy / Technology Applications; Keyboarding (a skill, not CS); AP Computer Science Principles; AP Computer Science A. Never a brand.

[17] Children's Online Privacy Protection Rule, 16 CFR 312, 90 Fed. Reg. 16918 (22 April 2025): operators collecting personal information from children under 13; general compliance date 22 April 2026, which has passed. Federal Trade Commission COPPA FAQ: COPPA "was not designed to protect children from viewing particular types of content." A parent holds the household login. Nothing in the opened homeschool statutes requires a dedicated lab or a one-child-one-laptop policy.

If this week needs a compass, this is it. Five things. Then sit down and teach.

The Computing Hour

The hour has a shape. Learn it once. Later chapters will give you today's idea, today's try-it, and today's questions. They will not rebuild this hour. When a chapter says "run the session," it means this.

You do not need a school bell. You need a looking start, a short model, a real attempt, one good question then a wait, a look together at what they expected versus what happened, and an unaided check. Younger children may finish in twenty-five minutes. Older students may need forty-five. The shape does not change. The idea on the table does.

Sit down having already done the five-minute parent warm-up from *How to Use This Book This Week*. You know today's idea and what would count as a result. The child looks, tries, and debugs. You hold the key.

1. Warm-up

Three to five minutes. A device, a file, or a last-week try they already know. Mixed, not a quiz of names they cannot yet attach. This is retrieval, not a test of character.

Say: "We're going to start with something you already know." Or: "Look at last week's file. Can you still find it?"

Keep this short. The warm-up is not the lesson. When you feel yourself teaching a new idea here, stop.

2. Short model

Five to eight minutes. One new move, visible. An object on the table. Then you stop.

First save, first packet of cards, first loop that has to stop: show the move, then fade.

Say: “I’m going to show this one short. Then you’ll try.” Then: “Watch how I save it with a real name in a real folder.” On the second pass, leave a hole: “Your turn to tell me the name.”

If you are still talking at minute twelve, you are giving a lecture. Close the model. Hand them the device, the notebook, or the student page.

3. Student attempt

This is the center of the hour. Eight to twelve minutes. The first try is theirs.

Hand them “For the student” if the chapter has it, or the first try-it from **How to teach it this week**. Then you talk less than you want to.

Say: “This one is yours. I’ll be quiet.” Then be quiet.

If they stall, use this order: ask, wait, hint, then model. Not the reverse. Ask: “Show me what you tried.” “What did you expect it to do?” Wait. Count a slow three in your head — longer if they are still on the task. The silence is the work. If you fill it, you took the problem back.

Hint, one hint: “Look at the name you gave it.” Or: “You already know it has to stop. Point to the part that should stop it.” Then, if they are still stuck after a real try: “I’m going to show you this one step. Then you take it from here.” Show the step. Return the keyboard.

If they have no idea to hang the observation on, that is a hole, not useful struggle. Fill it with a short retell, then return to the task. Keep your hands off their keyboard.

4. One good question, then wait

Two or three minutes for the question itself; the talk around it can run longer, using the debug shape below. Not “did you get it?” They will say yes.

An authentic question is one for which you have not already written the answer. “What is the vocabulary word?” is recitation. “What did you expect it to do, and what did it do instead?” is closer — if you do not already have the sentence you want to hear.

Say: “What did you expect it to do?” Or: “Where did you put it, and how will you find it tomorrow?”

Then wait. Robert Stahl, writing for ordinary classrooms, calls this think-time: a distinct stretch of uninterrupted silence. Teachers often pause less than a second and a half. A slow three is a useful minimum, not a magic cliff. When the child is debugging or hunting a file, the wait can honestly be longer — fifteen seconds, a minute, sometimes two.[18] Stahl is a classroom habit, not a computer-science trial.

One question. Maybe a follow-up. Then stop. If they cannot explain it back, they copied a step. Return to the looking and give one more attempt.

5. Look together: expected versus happened

Five to ten minutes. This is the heart of debugging.

Put the record on the table. Ask what they thought would happen. Ask what did. Look at the gap together. You do not snatch the keyboard at the first red mark.

Say: “You thought it would _____. It did _____. What’s different?” If two things changed, you cannot tell which one mattered. Name that. Then try once

[18] Robert J. Stahl, “Using Think-Time and Wait-Time Skillfully in the Classroom,” ERIC Digest ED370885 (May 1994), https://files.eric.ed.gov/fulltext/ED370885.pdf. Access date for URLs in these notes: 1 September 2026. Stahl, writing for a social-studies clearinghouse, prefers *think-time*: “a distinct period of uninterrupted silence.” Typical teacher pause after a question: “between 0.7 and 1.4 seconds.” A three-second silence is “a minimal amount of time unless the teacher has sound reasons to reduce this time”; 2.9 versus 3.0 is not a cliff. Among eight categories he names student task-completion work-time, which may be 3–5 seconds, several (15, 20, 30, or 90) seconds, or 2 or more minutes. Stahl is general classroom, not a computer-science randomized trial. No CS-specific wait-time study was opened for this book.

more with one change. A program that runs is performance. A program they can trace is the skill.

6. Unaided exit

Three to five minutes. One short task. Book closed. No hints. No chatbot. No looking at the worked example.

Tonight's helped practice is not this check. Say: "One task. I won't help. That's the point." Early years: a spoken sentence, or a saved file they can find tomorrow. Middle years: a paragraph. High school: a program they can talk through with the screen off. If it is wrong, note tomorrow's first move. Fade until the check does not need you.

When to stop talking

After you ask, wait. If you fill the silence, you took the problem back. During the attempt, your job is almost nothing. If you hear yourself explaining while they try, you have moved back into the model. Stop.

Instead of "I was never good with computers," say today's sentence: "What did you expect it to do?" Praise a clear look or a named file — not speed. Hands in your lap. Count three. For a stuck debug, count longer.

The try-it shape, once

Later chapters will name this week's try. They will not reprint this spine. When a chapter says "run the try-it," it means this.

Time. A short sit, then stop. Twenty minutes can be a whole hour for a seven-year-old. There is no national table of computing minutes for the kitchen.[19]

[19] No NCES, IES, or NCAA page opened on 1 September 2026 prescribed a national homeschool computing-minutes table. North Carolina's five-hour day is a recommendation across the whole program, not a CS block. Pennsylvania names 180 days or 900/990 hours for the whole program. New York's unit is 6,480 minutes. None of these is a national computing hour.

Materials. Household objects. One family computer. Cards and string for unplugged work. Unplugged can start the idea. A machine, if it is available and wanted, should then run it, so the child sees the same idea happen.[20]

Safety and accounts. Who owns the login. Under thirteen, you hold the key. Ages 5–10 default: scripts and parent-child talk, not live open chat. Looking at the family router as an object is enough. Reconfiguring it is not a first hour. Packets are a model of how a message can be cut into pieces, travel, and be put back. They are not a lab to break the household Wi-Fi.

The fun. Pointing at what is actually there. Sending a packet of cards to the other room. Making a sprite move. Hiding a file and finding it tomorrow.

The skill. Name the part. Save the file. Say what you expected. Trace one input. The fun is the motive. The skill is why you sat down.

If they click forever with no chance to say what they expected, they get activity without a mind.

The debug box, once

Later chapters will fill this week's questions. They will not reprint this shape. When a chapter prints a debug box, it means this.

Opening question. Authentic. About what they expected or what they see. No answer you have already written. Examples of the *shape*, not required topics: "What did you expect it to do, and what did it do instead?" "Where did you put it, and how will you find it tomorrow?" "If we cut the message into pieces, how does the other room know how to put it back?"

Three to five follow-ups. Pick; you do not need all five every Tuesday. Observation: What do you actually *see*? Expectation: What did you think would happen? Evidence: What in the notebook or on the screen supports that? Tell for sure: Can we tell it was the thing we changed? Revision: What would you change in what you said, now that you've looked again?

[20]Tim Bell, "CS Unplugged or Coding Classes?," *Communications of the ACM* 64, no. 5 (2021), https://cacm.acm.org/opinion/cs-unplugged-or-coding-classes/: Unplugged "never sought to replace computer programming"; used alone it can "justify poor decisions"; combined with programming it can act as a catalyst. CS Unplugged is a University of Canterbury project, not a curriculum and not a homeschool statute.

How to wait. After you ask, a slow three. After they stop, a slow three again. If they are still working — debugging, hunting a save dialog — the wait can be a minute or more. If the silence is hard, look at the notebook, the cable, or the script, not at the child. Say, "Take your time. I'll wait."

What a stuck silence usually means. Wait time has been too short for years — count. The question was recitation in disguise. The child does not have enough of the idea — tell a shorter one, then ask again. They are searching for a next step: "Describe the problem, then we'll get help." Or they think a green run is understanding. A finished program in a few minutes can leave a false sense of being on track.[21] Pose, wait, and point back at the record.

Accounts and safety, once

Later chapters will point here in one sentence. They will not reprint this box.

> **You hold the account. The child tries on a machine you can see. One family computer is enough.**
>
> Under thirteen, federal children's-privacy rules cover companies that collect personal information from a child. The general compliance date, 22 April 2026, has already passed. Those rules are about collection. They do not, by themselves, make a site's content safe.[22] You choose the service. You hold the login. You sit with the child.

[21] James Prather, Brent N. Reeves, Juho Leinonen, Stephen MacNeil, Arisoa S. Randrianasolo, Brett A. Becker, Bailey Kimmel, Jared Wright, and Thezyrie Briggs, "The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers," ICER 2024, arXiv:2405.17739: n = 21 CS1 students, one C++ problem, Copilot and ChatGPT, 35 minutes; 20 of 21 finished a working program. New difficulties named: Progression, Interruption, Mislead. Struggling students often finished with an illusion of competence. Limits the authors name: small sample, one site, extra-credit volunteers, not a delayed unaided-skill trial. This is a useful study, not a promise that every home will see the same result, and not a study of any particular home agent.

[22] Children's Online Privacy Protection Rule, 16 CFR 312, 90 Fed. Reg. 16918 (22 April 2025), document 2025-05904: operators of commercial sites and services directed to children under 13, and operators with actual knowledge they collect personal information from children under 13. Effective date 23 June 2025; general compliance date 22 April 2026 (passed as of the access date). Federal Trade Commission, COPPA FAQ: COPPA "was not designed to protect children from viewing particular types of content." COPPA does not apply to teenagers. A homeschool is not, by that fact, a school authorization. Photos, video, and audio of a child are personal information when collected by a covered operator.

Ages 5–10 work from scripts and talk with you, not from a live open chat. Photos of a child are personal information when a company collects them; keep faces in a local file unless you chose a share. Location stays off unless you turned it on. Thirteen to eighteen are still a household-account hour: the student may hold more of the login; you still know where the work lives.

A second machine or a small board computer is optional convenience, not a moral upgrade. Home network settings are a parent job. Look at the router as an object. This hour is not a hacking lab. Packets stay a model.

Stop the activity if you cannot make the account or the setup safe. That is teaching.

The AI rules, once

Later chapters will point here in one sentence. They will not reprint this box.

Child first. You hold the key. AI is optional for the adult.

The child looks, tries, traces, and writes before any tool is in the room. You keep the answer key. A chatbot does not sit beside them during the attempt. A fluent paste is performance, not the child's knowledge of the algorithm.

You may, on your account, after the child has tried:

- Explain this week's idea *to you* from a named page or manual — a textbook page, a language's documentation, a sentence from a standards map. If the model and the page disagree, the page wins. Named page, not "explain computers."
- Make extra practice of the same skill with new names and shapes, answers hidden. You hold the key. The child never sees it. Tonight's assigned sheet stays off the camera.
- Draft a labelled SCRIPT after the child tried — marked generated, dated, named, checked against a page. You vet it. Then you and the child *perform* it. Ages 5–10 especially: the child hears a person, not a tab.
- Give a hint after an attempt. One next step, not the function. Never "here is the program."

- Diagnose work already done. “Here is the child’s program and the error. What should I *ask*?” The child still owns the edit.
- Draft talk-box questions you vet. You delete coverage slog. You ask; the child talks to you, not to the tab.
- For first programs: point at a named language doc or textbook page, not “write my function.”

You may not:

- Park an unsupervised chatbot as the child’s only partner, especially ages 5–10.
- Ask the model to write the child’s program or homework.
- Invent a native-expert identity you cannot check — “you are a university professor; write this.”
- Grade code or writing from a detector you have not opened and cannot trust. Detectors flag human work often enough that they are not a verdict on a child’s page.[23]
- Issue a certificate of competence or fluency.
- Let the model invent data, network traces, or benchmark numbers. The oracle is the child’s run on this machine, or a named page.

Ages 5–10: scripts and parent-child talk, not live open chat. AI does not write the program or invent output.

In a high-school *math* study, an unguarded chatbot raised assisted practice and then cut the unaided exam. The same crutch is available whenever a fluent program can replace the child’s attempt.[24] In a small first-year programming lab, almost every student finished a

[23]Weixin Liang, Mert Yuksekgonul, Yining Mao, Eric Wu, and James Zou, “GPT detectors are biased against non-native English writers,” *Patterns* 4, no. 7 (2023): 100779; preprint arXiv:2304.02819 reports an average false-positive rate of 61.22 percent on human TOEFL essays.

[24]Hamsa Bastani, Osbert Bastani, Alp Sungu, Haosen Ge, Özge Kabakcı, and Rei Mariman, “Generative AI without guardrails can harm learning: Evidence from high school mathematics,” *Proceedings of the National Academy of Sciences* 122, no. 26 (2025): e2422633122. Field experiment, nearly 1,000 Turkish high-school *math* students: a ChatGPT-like tool raised assisted practice grades 48 percent relative to control, then cut unaided exam grades 17 percent; a hint tutor that withheld full solutions raised practice without that exam drop. This is a mathematics trial, not a coding trial. The transferable caution is the crutch: if the model produces the program, the child is a spectator.

working program with the tools on; struggling students often left with an illusion of competence. This is a useful study, not a promise that every home will see the same result.[25]

Language models invent citations. Treat every model-supplied quotation, "typical result," and paper title as untrusted until you find it in a named book or the child's notebook.[26]

Hermes 4 is a chat model. Hermes Agent is a different runtime that lives on a machine. They are not the same tool. Grok the assistant is the conversation in a tab. Grok Bot is a cloud computer that keeps working — not a school product, and not a homeschool tutor. A school partnership in El Salvador uses the assistant, not Grok Bot, and is not a U.S. kitchen product. Picture-making jobs make pictures, not a computer-science course. Copilot and ChatGPT are tools some families already have. They are not required for this book. If you use any of these, they live on *your* machine or *your* account. They are not the child's programming partner, and they are not a companion.

You can teach this entire book with no AI. Many families will. The hour still has the same shape.

---

[25]James Prather, Brent N. Reeves, Juho Leinonen, Stephen MacNeil, Arisoa S. Randrianasolo, Brett A. Becker, Bailey Kimmel, Jared Wright, and Thezyrie Briggs, "The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers," ICER 2024, arXiv:2405.17739: n = 21 CS1 students, one C++ problem, Copilot and ChatGPT, 35 minutes; 20 of 21 finished a working program. New difficulties named: Progression, Interruption, Mislead. Struggling students often finished with an illusion of competence. Limits the authors name: small sample, one site, extra-credit volunteers, not a delayed unaided-skill trial. This is a useful study, not a promise that every home will see the same result, and not a study of any particular home agent.

[26]William H. Walters and Esther Isabelle Wilder, "Fabrication and errors in the bibliographic citations generated by ChatGPT," *Scientific Reports* 13 (2023): 14045: 55 percent of GPT-3.5 citations and 18 percent of GPT-4 citations were fabricated; of the real citations, 43 percent and 24 percent had substantive errors. https://www.nature.com/articles/s41598-023-41032-5. Dated to 2023 models; this book does not update those rates to 2026. Hermes Agent: https://hermes-agent.org/ (open-source agent runtime, February 2026, Nous Research). Hermes 4 is a separate chat and reasoning model family; Nous Portal documentation states it is not recommended inside Hermes Agent. Grok Bot: xAI, "Introducing Grok Bot," 11 August 2026, https://x.ai/news/introducing-grok-bot, expanded 26 August 2026; FAQ https://docs.x.ai/grok-bot/faq — cloud computer; no education product on the pages opened 1 September 2026. Grok the assistant, Grok Bot, and the El Salvador ministry partnership for Grok the assistant (11 December 2025) are three different objects. A "Game artist" job on Grok Bot makes pictures, not a computer-science course.

A Tuesday, said plainly

Here is an illustration, not a reported family. A parent has spent five minutes looking at one object and asking what would count as a result. The child warms up on last week's named file. The parent shows a short model — one save, one find — then puts the try on the table. The child tries. The parent asks, "What did you expect it to do?" and waits. They look together at the gap. Exit: "One task. Find it tomorrow. I won't help." Right or wrong, the hour had a shape.

On a high-school day the same shape holds: warm-up, short model, their try, the expected-versus-happened question, an unaided trace with the screen off.

You can run that hour. You need today's idea, a keyboard you refuse to take, and the willingness to stop talking. Later chapters will say: run the session as in *The Computing Hour*. That sentence is this page.

Chapter 1

Hardware

Figure 3: A kitchen-table still-life from a high three-quarter view: a keyboard, a coiled cable, a closed retired case, a notebook with the words “it won’t turn on” written as an object on the page. No people. No logos. No readable screen.

Why this matters

Hardware is the thing you can point at. A keyboard. A screen. A case. A cable. Maybe a tablet, a phone, a printer. Before a child can talk about files, networks, or programs, they need a body in the room that is a computer — something that takes an input, does work, and shows an output. That is this week's idea. It is worth the struggle because later weeks sit on it. An operating system is software that notices a key press. A network is how devices talk. A program is a process a machine can run. If "the computer" remains a magic box, those later sentences have nowhere to land.

The sentence that surprises people is this one: looking at a device, and naming what you pressed and what you saw, is computer science. Typing a letter, searching the web, and chatting with a model are computer *use*. Both belong at a kitchen table. They are not the same thing. The *K–12 Computer Science Framework* calls computer science the study of computers and algorithmic processes — their principles, their hardware and software designs, their applications, and their impact on society. The same pages warn that computer science is often confused with the everyday use of computers, such as learning how to use the Internet and create digital presentations. A survey those pages cite found that a majority of students believed creating documents and presentations, and searching the Internet, were computer science activities. That is the Framework's own cited survey, not a new poll. The confusion is old. Your job this week is to give the child a looking move that cuts it.[27]

Computer literacy, digital citizenship, educational technology, and information technology are about *using* the machine. Computer science is about why it works and how people make it work. The Framework says instruction in all three areas — computer science, computer literacy, and digital citizenship — is important. It also says they are not the same thing.[28] A later chapter of this book will teach a real

[27] *K–12 Computer Science Framework* (2016), PDF p. 12: computer science is "the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society." Same page: computer science "is often confused with the everyday use of computers, such as learning how to use the Internet and create digital presentations." A Google & Gallup 2015 survey cited there found that a majority of students believed creating documents and presentations (78 percent) and searching the Internet (57 percent) were computer science activities. Report as the Framework's own cited survey, not as a 2026 poll. https://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf. Access date for URLs in these notes: 1 September 2026.

[28] *K–12 Computer Science Framework* (2016), PDF pp. 12–13, distinguishing computer literacy,

document, honestly titled Digital Literacy or Keyboarding. This chapter teaches the device as a system. The transcript word a stranger can read is **Computer Science**. It is never "PC Repair." California's public-school standards say computer science is not learning how to build or repair computers. An athletic-eligibility rule set draws the same line for a different reason: basic computer usage and repair do not contribute to a computer-science core course.[29] You are not running a certification shop. You are teaching names, a hardware-plus-software story, and a few accurate words for "it will not turn on."

The maps you may hear named — the Framework, the Computer Science Teachers Association standards, a large state's public-school document — are maps. They are not a homeschool statute, not a curriculum, and not a promise of minutes. CSTA's 2017 document turns the Framework into student expectations. In the early years those expectations are ordinary: identify common physical parts; describe a basic hardware or software problem using accurate words. A little later: inside and outside work as a system; hardware and software work together; common troubleshooting. At eleven to fourteen a child can combine a sensor or a camera with a program to collect something, and can troubleshoot as a list. At fifteen to eighteen they can compare layers: application software, system software, hardware. Everyday objects contain computers. Repair is still not the definition.[30]

---

educational technology, digital citizenship, and information technology from computer science because they focus on using computer technologies "rather than understanding why they work and how to create those technologies." FAQ: the Framework "is meant to complement the instruction of computer literacy and digital citizenship. Instruction in all three areas is important for all students." Framework statements "are not standards." https://k12cs.org/faq/.

[29]California Department of Education, *Computer Science Standards for California Public Schools, Kindergarten Through Grade Twelve*, adopted by the State Board of Education 6 September 2018, pp. 15–16: computer science is not "learning how to build or repair computers," nor typing or office software. Public-school standards, not a homeschool statute. https://www.cde.ca.gov/be/st/ss/documents/csstandards.pdf. NCAA High School Review Committee, *Policies and Procedures 2026–27*, Appendix A: content that does not contribute to computer-science approval includes "Basic computer usage/repair (e.g., keyboarding, typing)"; content that contributes includes operating systems, networks and hardware design. Athletic eligibility, not a standards body. https://ncaaorg.s3.amazonaws.com/committees/ncaa/hsreview/HSRC_PoliciesAndProcedures_Staff.pdf.

[30]Computer Science Teachers Association, *CSTA K–12 Computer Science Standards, Revised 2017*: 1A-CS-02 identify common physical components; 1A-CS-03 describe basic hardware and software problems using accurate terminology; 1B-CS-01 internal and external parts function as a system; 1B-CS-02 hardware and software work together; 1B-CS-03 common troubleshooting; 2-CS-02 design projects that combine hardware and software to collect and exchange data; 3A-CS-02 compare levels of abstraction among application software, system software,

What this week unlocks is a child who can point, press, and say what happened. Skip the looking, and later "the operating system" arrives as a vocabulary card. Skip the talk, and later troubleshooting is "it's broken" forever. Skip the accurate words, and later a parent who is not a computer scientist cannot hear a wrong picture. You do not need to be a computer scientist. You do need to understand today's idea well enough to hear "the computer is thinking" as a wrong picture, not a cute slip.

This week you can learn how a key press that makes a letter appear is a hardware-plus-software story. Today the student can name two parts you can touch, press a key, and say what they saw, looking at the machine.

For the parent: understand it yourself

You do not need to open a case. You do need to understand today's idea well enough to hear "the computer is thinking," "it's broken," and "we did computer science — we used the tablet" not as cute slips.

Many adults feel rusty around the inside of a machine they have used for years. That feeling is common. It is not a verdict. Five minutes of this section, then the warm-up at the end, is enough for tomorrow. The child still does the looking.

Everyday picture. A keyboard sits on the table. You press the letter *b*. A *b* appears in a document. You are not teaching a parts catalogue. You are attaching a small, true story to a thing the child can see: something you can touch (the key) sent a signal; software noticed; software told the screen what to show. Then you ask, "What did you press, and what did you see happen?" and you wait. The work is the pointing and the sentence. The word *CPU* can wait.

Precise picture. Hardware is the physical parts of a computing system. Software is the instructions those parts follow. They work together. A keyboard is hardware. The program that draws the letter is software. Between them, on many machines, sits a layer of software that notices the press and hands it to the application — California's grades 3–5 example names that layer an operating system, as the software that notices a key press, not as a fifteen-year-old's sysadmin course. You will teach that layer in the next chapter. This week you only need to hear that a letter

---

and hardware. Levels 1A–3A for all students; 3B specialty. Professional-association standards, not national law. https://csteachers.org/wp-content/uploads/2025/03/csta-k-12-computer-science-standards-revised.pdf.

appearing is not "the computer thinking." It is hardware plus software doing a job you can describe.[31]

The Framework's Computing Systems strand is the household picture of the band. In early grades, students learn features and applications of common computing devices, and they learn how systems use both hardware and software to represent and process information. Troubleshooting at about grade 2, on that map, is describing the problem and getting help — not diagnosing a system. Around grade 5: power, connections, restart. Around grade 8: a systematic process, and tradeoffs such as functionality, cost, size, speed, accessibility, and aesthetics. Those are grade-band endpoints on a map. They are not a law of your Tuesday.[32]

A professional-association map (CSTA 2017) says similar things in student language: name common physical components; describe basic hardware and software problems using accurate terminology; later, internal and external parts function as a system; hardware and software work together; common troubleshooting. At eleven to fourteen, design a project that combines hardware and software to collect and exchange data. At the high-school floor, compare levels of abstraction among application software, system software, and hardware. A 2026 revision of those standards still says computer science is not using word processors, slide decks, or generative AI tools, and is not limited to coding. Hardware as a system stays in the book. Using the tablet to type a caption is use.[33]

When the child misses this week's idea, you have two adult moves, not a slogan. A recast is you saying their observation again, more accurately, without stopping

[31] California Department of Education, *Computer Science Standards* (2018), 3-5.CS.2 descriptive example: a keyboard, desktop computer, monitor, operating system, and word-processing application as interacting parts; "The keyboard (hardware) detects a key press, which the operating system and word processing application…" Examples are not compulsory. The operating system appears here as the software layer that notices a key press.

[32] *K–12 Computer Science Framework* (2016), Computing Systems overview: "In early grades, students learn features and applications of common computing devices" and "learn how systems use both hardware and software to represent and process information." Grade-band troubleshooting: grade 2, describing the problem and getting help; grade 5, power, connections, restart; grade 8, a systematic process and tradeoffs (functionality, cost, size, speed, accessibility, aesthetics). Grade-band endpoints, not age labels the Framework wrote.

[33] Computer Science Teachers Association, *2026 CSTA PK–12 Computer Science Standards*, DOI 10.1145/3820482, PDF p. 2: computer science is not "using technology tools like word processors, slide decks, or generative AI tools," and is not "limited to coding or programming." PDF p. 4: the standards "do not prescribe specific curriculum, instructional materials, or assessments." Not yet the California, Texas, or Virginia map. https://csteachers.org/pk12standards/.

the looking: “You pressed *b*. A *b* appeared.” A prompt is you making them try the accurate words — two choices, a start — when this week’s idea is the one that slipped: “Was that the keyboard, or the screen?” If the miss does not block the looking and is not this week’s idea, recast once and go on. If it is this week’s idea, or meaning broke, prompt; then, if they are still stuck, give the words and have them say them looking at the machine. You will not interrupt every utterance. You will not pretend correction never helps.

After you ask, wait longer than feels polite. Robert Stahl, writing for a general classroom, not as a computing trial, called that pause *think-time*: a distinct period of uninterrupted silence. Typical teacher pauses after a question run under a second and a half. A slow three is a useful minimum, not a magic cliff. When the child is staring at a cable or a dark screen, the right category is task-completion work-time, which can honestly be fifteen, twenty, thirty, or ninety seconds, or two or more minutes. Look at the device if the silence is hard. This book does not treat that wait as a computer-science discovery. It is a useful classroom habit you can steal.[34]

A generated picture of “the inside of a computer” is a picture to look at together, labelled generated. It is not a teardown. The child looks at the machine in the room.

Wrong answers you should be able to hear

1. “The computer is thinking.” A letter appeared, or a spinner spun, and the child gave the box a mind.

What it usually means. Everyday talk treats delay as thought. The hardware-plus-software story is not yet attached to the press.

What to say. “What did you press, and what did you see happen?” Wait. If they are still stuck: “You pressed a key. Software noticed. A letter appeared.” They say it, looking at the keyboard. You go on. You do not open a lecture on consciousness.

[34] Robert J. Stahl, “Using Think-Time and Wait-Time Skillfully in the Classroom,” ERIC Digest ED370885, May 1994. Think-time as “a distinct period of uninterrupted silence.” Typical teacher pause after a question “between 0.7 and 1.4 seconds.” A three-second period is “a minimal amount of time unless the teacher has sound reasons to reduce this time”; 2.9 versus 3.0 is not a cliff. Student task-completion work-time may be several (15, 20, 30, or 90) seconds or two or more minutes. General classroom, not a computer-science trial. https://files.eric.ed.gov/fulltext/ED370885.pdf.

2. A parts list with no finger on a part. "CPU, RAM, GPU," recited from a heading, while the eyes never land on the keyboard.

What it usually means. Coverage won. The hour became a catalogue.

What to say. Cover the list. Put a finger on the key. "What part were you touching?" Two names you can point at are enough this week. Inner names can wait until a looking hour that has an inside.

3. "It's broken" as the whole story. The tablet will not turn on, or the app will not open, and the child has no more words.

What it usually means. Troubleshooting has not yet become a description. On the early map, the skill is saying what is wrong and getting help, not a stack dump.

What to say. "What words would you use to tell me?" Offer two: "It will not turn on" or "The app will not open." Then: "What did you already try?" Power, a cable, a restart — later. Accurate words first.

4. A fluent teardown nobody in the house did. A generated motherboard. A video of someone else's case. A certificate of "hardware fluency."

What it usually means. The work was outsourced. A picture can help you look. It cannot replace the machine on the table.

What to say. "Show me the keyboard. Press a key. Say what you saw, with the generated picture off." If they cannot, the picture comes off. They look again.

5. Repair as the definition. Swapping a stick of memory as this week's "computer science," or a year titled PC Repair on a record a stranger will read as Computer Science.

What it usually means. A useful hobby collided with a credit name. Opening a retired case, if you have one, is a looking hour. It is not the course.

What to say. Name two parts. Press a key. Describe one problem. If a curious older student wants to watch you open a machine that is already retired and unplugged, that is extra looking. The transcript still says Computer Science.

A sixth: the eyes never land on the actual cable, only on a cartoon of one. Cover the caption. Finger on the thing. "What do you actually *see*?" Then wait.

Five-minute parent warm-up

Before the lesson, with the family computer or tablet, a cable, and a scrap of paper, no student in the room.

1. Press one key in a document. Say out loud: “I am not going to ask what hardware is. I am going to ask what I pressed and what I saw.”
2. Write today’s opening on the scrap: “What did you press, and what did you see happen?” Say the question once. Count a slow three. That silence is the lesson.
3. Write the wrong turn you must be able to hear: “The computer is thinking.” Next to it, the repair you will actually say: “What did you press, and what did you see happen?”
4. Unplug the cable from the wall — you, not a child — and look at the plug. Notice that looking behind a live machine is not this week’s try. Safety is the adult in the room.
5. Put the pencil down. You are ready. The child still looks.

If those five minutes are fluent for you, you can hear a mind in the box, a catalogue with no finger, and a repair year in disguise. That is the job.

How to teach it this week

Run the computing hour from the front of this book. This chapter fills it with hardware. Warm-up. Short model. Student attempt. One good question, then wait. Mixed practice. Unaided exit. The curriculum on the shelf is the map. The hour is triage.

Warm-up (2–3 minutes, unaided). Yesterday’s object, if you have one: “What part did you touch last time?” No device required if a drawing of the keyboard is still on the table. No definition.

Short model (5 minutes). You put the keyboard or the tablet on the table. Think aloud only as much as you must: “I am going to press a key and watch.” Press. Pause. “A letter appeared.” Then the question: “What did you press, and what did you see happen?” You are showing looking, not delivering a lecture on motherboards.

Fade as soon as you can. Second sitting: they press while you watch. Third: they say what they saw. Fourth: they name two parts unaided, looking at the machine.

Include one incorrect example you say on purpose: “The computer thought of a *b*” — then you wrinkle your face and repair it: “I pressed *b*. A *b* appeared.” The child hears the mismatch. You do not turn that into a worksheet.

Exact wording you can steal.

On a key: “What did you press, and what did you see happen?” Then wait.

On a part: “What part were you touching?” Then wait.

On a dark screen: “What words would you use to tell me?” Then wait.

On a mind in the box: “Was that thinking, or was that a key plus software?” Then wait.

On a generated picture: “If we cannot point at it in this room, it is not tonight’s looking. The picture stays closed until you have tried.”

When you are about to take over: “Your eyes. I’ll wait.”

Then wait. A slow three after you ask is a useful minimum. After they stop, wait again. That second pause is where a revision can still happen. Stahl’s think-time is a general-classroom habit. Use it here as a habit. A follow-up that arrives in half a second trains the child to wait *you* out, not to look.

First try for the student, after the model. Two parts still in view. Product goal on a sticky note: name each one, then press a key and say what you saw. Book closed for the attempt. You wait. Struggle before rescue. You do not grab the keyboard.

One good question. After the attempt: “What did you press, and what did you see happen?” — now without pointing first. The student generates the sentence.

Student attempt, then mixed practice. Two or three short namings of the new type, unaided, after the fade. Then mix: yesterday’s part if you have one, last week’s “it’s broken” sentence if you have one, one “the computer is thinking” repair, one part they have not named today.

Exit ticket (3–5 minutes, unaided). One question, device still in view, one sentence they say alone. Tool closed. You grade the unaided sentence. A fluent generated teardown is performance. The sentence they can still say is the test.

How to fade help. You press and say → they press while you say → they say while you point → they name unaided → they ask *you* the question. When they

can attach a press to a letter without your prompt, put the scaffold away. Bring it back when a new kind of looking arrives — a problem to describe, a tradeoff, a layer.

When to stop talking. After one model and one question. You hold the key: is there a device, a sentence said for real, and an unaided try? You are not the catalogue.

Safety this week is the adult in the room the whole try. Unplug from the wall before anyone looks behind a machine. No one opens a live power supply. No one forces a stuck button. A retired case, if you use one, is already unplugged; the adult opens it; capacitors are not toys; you do not plug it in open. One family computer is enough. A second machine is optional convenience, not a moral upgrade.

Age-band moves

These are sequence anchors, not twelve mini-books. Place by skill, not by birthday. A fourteen-year-old who cannot yet name a keyboard and a screen is still in the first band of this chapter for the *looking*. They may still put Computer Science on a high-school transcript if the year's work is systems, programs, data, and impacts — not if the year is only this looking hour. A nine-year-old who can already describe a problem in accurate words is not "ahead" because a catalogue printed grade 6 on a cover. They are ready for the next try-it.

Ages 5–10. Names you can point at. Keyboard, screen, computer or tablet, maybe a printer or a cable. Hardware and software work together: a press, a letter. Troubleshooting is a sentence: "The tablet will not turn on." "The app will not open." Then they get help. Inner workings are not required. A looking hour at a retired case is still allowed if you have one and an adult opens it. Short sits. Then stop. The child points. The child says one line. You wait. You do not start a parts catalogue. You do not start registry talk. The word *operating system* may be heard as the software that notices the press. It is not a course.

Ages 11–14. Combine. A camera or a simple sensor plus a program that collects something — a photo they take, a list of temperatures from a cheap outdoor thermometer they type, a sound they record — is enough. No opened map requires a particular board. Troubleshooting becomes a list: power, connections, restart. Tradeoffs become a conversation: this device is lighter; that one lasts longer; this one is easier to read. Application versus the thing that launches it can begin here; the next chapter owns it. You still do not run a repair shop.

Ages 15–18. Layers. Application software, system software, hardware. Everyday objects contain computers — a thermostat, a car's dash, a watch. The child can compare what you can see (the screen, the key) with what you cannot (the software that noticed the press). Specialty talk — categorizing operating-system roles — waits for the next chapter and sits *beyond* the floor. Repair is still not the credit. If the student later sits an introductory college-shaped computing course, systems and networks are one piece of it, not the whole year, and not this week's floor.

Named try-it: Name-the-Parts Still-Life

Time. 10–15 minutes.

Materials. The family computer or tablet. A cable. A notebook.

Safety / accounts. Adult present. Unplug from the wall before anyone looks behind. No opening a live power supply. Parent-held login if the device needs one. The child does not create a new account to look at a keyboard.

The fun. Pointing at what is actually there. The cable is not a diagram of a cable.

The skill. Accurate names: keyboard, screen, computer, maybe printer. Not a parts catalogue. Not A+ certification.

How to run it. Put the still-life on the table. Ask what they actually *see*. Wait. If they sort by color of the case, the looking missed the job. Ask what you *touch* to make a letter. Grades 5–10: two names and stop. Ages 11–14: add the cable and "what it is for." Ages 15–18: after the names, one sentence about a part you cannot see that still has a job.

Figure 4: A close still-life: keyboard, coiled cable, closed case, notebook. Adult-demonstration quality. No children. No logos. No readable UI.

Named try-it: Key-Press Story

Time. 10 minutes.

Materials. A keyboard and a document, or a tablet's on-screen keyboard and a notes app.

Safety / accounts. Ordinary use. Parent-held account. No public share of the child's sentence.

The fun. "I pressed, a letter appeared."

The skill. Hardware and software work together. That is the California 3–5 picture in kitchen language. It is not operating-systems-as-a-course.

How to run it. Model one press. Ask the opening question. Wait. If they say the

computer thought, recast: “You pressed. A letter appeared.” If they name a CPU they cannot point at, cover the word. Finger on the key.

Named try-it: Describe-the-Problem

Time. 10 minutes.

Materials. A device that is off, or a pretend-off, or a real “it won’t open.” A notebook.

Safety / accounts. Do not force a stuck button. Adult present. No one opens a live machine to “fix” it.

The fun. Being the person who can *say* what is wrong.

The skill. Early troubleshooting: describe the problem and get help. “The tablet will not turn on.” “The app will not open.” Not a stack dump.

How to run it. Show an off device, or close an app and ask them to open it with a cable unplugged (you unplugged it). “What words would you use to tell me?” If they say “it’s broken,” prompt two choices. Write their sentence only after they say it. Ages 5–10 stop at the sentence. Ages 11–14 add one thing they would try next (power, cable, restart) as a list, not as a repair class. Ages 15–18 write a three-step list someone else could follow.

Debug / talk box

This is an illustration, not a reported family. A key has been pressed. A letter is on the screen, or is not. The parent has not announced a definition of hardware.

Opening question

“What did you press, and what did you see happen?”

Not: “Define hardware.”

Follow-ups (pick three to five; you will not use all of them every sitting)

1. What part were you touching?
2. What do you think the software was doing?
3. If it will not turn on, what words would you use to tell me?
4. Can we tell it was the cable — or did something else change too?

5. What would make you change your mind?

How to wait

Ask. Count a slow three in your head. Look at the device, not at the child's face, if the silence is hard. After they stop talking, wait again before you speak. Stahl called that pause think-time in a general classroom. A slow three is a useful minimum, not a cliff. When they are hunting a cable or a dark screen, task-completion work-time can honestly be a minute or two. A follow-up that arrives in half a second trains the child to wait *you* out, not to look.

What a stuck silence usually means

The question was recitation in disguise — you already had "CPU" in mind. Or the wait has been too short for years, and they learned that you will fill it. Or there was not enough of the idea on the table: a word with no press in the body. Or they think "it's broken" is the whole story. Mapping what they saw into a sentence is hard work. Next move: do one short press again, then ask the opening question again. Done enough this week: the child names two parts and describes one problem in accurate words, unaided.

Practice that actually builds learning

Blocked, for a new move. The day you introduce names, give a short set that is only that: the still-life, two parts, point. The day you introduce the key-press story, only the press. The day you introduce "it will not turn on," only the off device. Mixing too early makes the child hunt for the "right" school move instead of looking.

Mixed, for when to use it. Later the same week, the keyboard reappears next to the off tablet and the cable. Mixing is the practice of choosing a looking move: is this a name, a press-and-see, or a problem to describe? You impose that mix on the week you already have. A purchased reader that is all "parts of a computer" can still be mixed by you closing the book and putting a finger on the actual key.

Brief retrieval of tries already done. Two minutes, mixed, after the child can talk those tries untimed. Cover any vocabulary boxes. You listen. Overlearning a word list until tonight's score is perfect is the wrong retrieval for this band. Retrieval is a second look at a part they already named, not a race.

One incorrect example to diagnose. After the child has attempted the key-press story, show a card that guessed "the computer is thinking" from a spinner on the screen. Ask: "What did this person use to decide?" After "it's broken," show the off tablet and ask what words they would use. The child names the miss. You do not narrate it as a verdict on talent.

Talk every sitting. Oral language is the product at 5–10. A spoken sentence is done enough. A caption is a sentence you write only after they say it. Complex talk — a reason tied to a part — is already the thing later writing will need. You are not assigning a paragraph. You are teaching a sentence the child can hold.

Use this week's talk box. Same opening: "What did you press, and what did you see happen?" Same wait. Same stuck-silence meanings. Do not turn practice into a second lecture.

Named try-it: Retired-Case Looking (optional)

Time. 15–20 minutes. Ages 11–14, or a curious 5–10 with an adult.

Materials. A retired machine you already own. A screwdriver the adult holds.

Safety / accounts. Unplugged. Adult opens. No capacitors as toys. You do not plug it in open. No live power supply. If you do not own a retired machine, skip this try-it. It is not required.

The fun. Seeing that there is an inside.

The skill. Devices have insides that sense and act. Internal and external parts function as a system. Not A+ certification. Not a national requirement that every second-grader open a case.

How to run it. You open. They look. "What do you actually *see*?" Wait. Name only what you can point at. A generated motherboard photo, if you use one later, is labelled generated and is not this hour. Stop while they still have a look left.

Named try-it: Tradeoff Cards

Time. 15 minutes. Ages 11–14 and 15–18.

Materials. Three device pictures you printed without brand marks, or three real devices already in the house (phone, laptop, tablet). Paper. A pencil.

Safety / accounts. Ordinary looking. No shopping login. No child's photo uploaded anywhere.

The fun. "This one is lighter; that one lasts longer."

The skill. Tradeoffs: functionality, cost, size, speed, accessibility, aesthetics. That is the middle-grade shape of device design on the Framework map. It is not a shopping list.

How to run it. Lay out three. "Which would you pick for a long walk, and why?" Wait. If they pick by color only, ask what would happen if the battery died. Ages 15–18 add a layer sentence: which parts are hardware you can point at, and which jobs are software you cannot see?

A drawing of the still-life is making in the service of looking. It does not replace the key. Clicking forever without a chance to say what they pressed is activity without a mind. The hour is looking and talk, and you hold the key.

For the student

You are learning how a computer is a thing you can point at. A keyboard is a part. A screen is a part. You press a key. Something you can see happens — a letter, a light, a new picture. Hardware is the part you can touch. Software is the instructions the parts follow. They work together. A long word is a name. You do not need every long word yet. The looking is the work.

A keyboard on the table. You press *b*. A *b* appears. You say what you pressed and what you saw. That is computer science you can see. It is not the same as typing a whole letter to Grandma. Typing is useful. This looking is why the letter can appear at all.

Tiny worked example

Look at a keyboard, with no parts-list showing.

What do you actually *see*? Keys. A cable, maybe. A screen nearby.

What did you press, and what did you see happen? I pressed *b*. A *b* appeared.

That is looking. The caption, if you want one: "I pressed. A letter appeared."

Two tries

1. Name two parts you can touch. Point while you name them. Then press a key and say what you saw, looking at the machine.
2. Look at a device that will not turn on, or that is pretending to be off. Say what is wrong in ordinary words. “It will not turn on” is a good sentence. “The app will not open” is a good sentence. “It’s broken” is a start. Make it more accurate.

For the second one, you are the person who can *tell* someone what happened. You are not the person who has to fix it today.

Write nothing if writing fights the sitting. Talk the sentence. Your grown-up can remember it.

Explain it back

Tell someone at the table, in your own words, how a key press makes a letter appear. Then point to two parts and say what you *see*, not the word from a list. Then say one problem in accurate words.

Challenge

Some people look at a spinner on the screen and say the computer is thinking. Some people say “CPU” because the heading said so, without pointing at anything. Some people say a year of looking at devices is “PC Repair.” What would those three moves mean? How do you look at the *thing* on the table instead? How do you ask what you actually pressed and what you actually saw?

You are allowed to struggle. You may use a keyboard, a cable, a notebook. You look. You talk. If you get stuck, ask for a hint — not the vocabulary word. Then try again.

A tablet is a computer you can hold. A picture a program made of “the inside of a computer” is a scene. It is interesting. It is not the keyboard in the room.

When you talk, a sentence about what you pressed is enough for today. You do not have to write a paragraph. That is later. Today you look, you press, you say one true thing.

If it isn’t clicking

Three diagnostics. Each one has a next move. None of them is a verdict on talent, and none of them is a reason to wait for a birthday.

1. The child cannot name two parts, or recites a list, or says the computer is thinking whenever anything happens.

Looking is not yet attached to the *things* on the table. Next move: fewer objects, more pressing. Two parts you can talk about in the body — keyboard, screen. Cover words. Name the press out loud, then ask what they saw. “The computer is thinking” is ordinary. Stay with a press you can see. If this is still the bottleneck after several weeks of short daily looking, stay here. An operating-system week on top of a child who cannot point at a key will not hold. A human who will sit with a keyboard and wait, not a tablet that narrates a teardown, is a reasonable next step if you have tried the small-set work and the look still does not come.

2. The child recites a parts video, or a “inside your computer” script, and cannot point to the actual cable.

The cue is still a performance. Next move: objects first, press second, question third. Cover the caption. Finger on the key. The prompt is always through the looking: “What do you actually *see*?” After an attempt, one incorrect example to explain (“this person said CPU because the heading said CPU — what did they use?”). Slow down the “interesting” video until a sentence can be tied to a part; a child who can recite a cartoon and cannot point to the keyboard is not ready to skip this. Go ahead once they can name two parts, press, and say what they saw with their eyes on the machine. A tutor is useful if recitation remains the default after a couple of weeks of daily object-first work *and* the hour has become a fight.

3. The child will not talk, or every sitting ends in “it’s broken,” and the silence after your question is a wall.

Talk is the product, and a shrug is the mind at rest. Next move: shorter sits, one object, the opening question from the talk box, wait a slow three twice. If silence is hard, you look at the device, not at them. An authentic question is one you do not already have the sentence for. If you hear yourself answering, you have started doing the work. For “it’s broken,” ask what words they would use, then offer two. One new part a week is enough. If the hour has become a fight after a stretch of daily short sits, a human who will wait, not a chatbot that fills the silence, is the release valve.

When to slow down. Parts still unnamed. Recitation still the default. Sits so long that looking never starts. A generated motherboard doing the work of the keyboard. Those are brakes. “Not ready” because of age is the brake this book will not use. “Not ready” because the objects have not yet become a looking tool is a real brake. The early map will not require inner workings this year; that is not a reason to skip the key.

When to go ahead. Two parts are an easy point. The child presses, then talks. A problem has words they can use. A key-press story has a letter they can show. Short sits are ordinary. Then the next chapter’s home screen, one file, and log off have somewhere to sit. Being “good with computers” because a child loves games is not a reason to skip the looking.

When to get a human tutor. You have run the two-part try, or the covered-caption looking, or the wait after the question, for a stretch of daily sittings, and the same diagnosis is still the one in the room, and the hour has become a fight. A tutor is a release valve, not a failure of the sitting. Keep yourself as the person who can still hear a vocabulary quiz and a generated-teardown-as-observation. Outsourcing the hearing is the thing to avoid, not asking for help.

A child who can press and will not talk is missing one strand. A child who can talk a stream about processors and cannot point to a key is missing another. Hearing which is thin is enough. If the child is missing the look, model a *different* pair — a tablet’s volume button, a printer’s power — then guided pressing, then the keyboard. If the child has the look and the task is a first problem-sentence, one off device is the second sequence.

A fifteen-year-old who never had the early looking is behind in time-on-task, not locked out. Plan the band.

Tools, including AI

Optional helpers for you, the adult. The full rules live in the computing-hour box in the front of this book. This chapter is a pointer, not a second policy paper.

The short version: the child looks and talks first. You hold the device and the key. A tool may explain today’s idea *to you* from a named page — a Framework sentence, a CSTA line, a page in a book you already own. If the model and the page disagree, the page wins. A tool may make extra isomorphic practice (new names

of parts, same pointing job) with the answers on a separate page you keep. It may draft a labelled SCRIPT after the child tried, which you vet and then perform. It may give you a hint to ask after an attempt, or help you diagnose work the child has already done, or draft talk-box questions you vet. The child looks at the machine. The child talks in the room.

Leave these out of the hour: a chatbot as the looker; a model pretending to narrate a teardown as today's observation; a generated motherboard treated as the family computer; "what's inside a CPU" as the whole sitting; an unsupervised chatbot during the try, especially ages 5–10; a tool that writes the caption so the child does not have to look. Ages 5–10: scripts and parent-child talk, not live open chat. A generated picture can be a scene to look at together for two minutes — "what did this picture get wrong?" — labelled generated, and then it comes off the table. A real keyboard comes on. A generated picture has no press in the room.

A language model will happily invent a "lab note" from a first-grader, or a list of parts the child never touched. Treat every model-supplied observation, caption, and "as we saw" as untrusted until it lives in the child's pointing or in a named book page. Fluent talk from a tool is performance, not the child's knowledge of the press. The unaided sentence at the keyboard tomorrow morning is the test.[35] A small study of novice programmers found that almost everyone finished a working program with a coding helper, and that struggling students often left with an illusion of competence. That paper is a first programming course, not a hardware hour. The sitting condition is still the crutch: if the tool produces the looking, the child is a spectator.[36] A high-school math study found that an unguarded chatbot raised assisted practice and then cut the unaided exam. That paper is mathematics, not a computing trial. The crutch is still the sitting condition.[37]

[35] Nicholas C. Soderstrom and Robert A. Bjork, "Learning Versus Performance: An Integrative Review," *Perspectives on Psychological Science* 10, no. 2 (2015): 176–199. Current performance is often an unreliable index of learning.

[36] James Prather et al., "The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers," ICER 2024 / arXiv 2405.17739. n = 21 CS1 students, one C++ problem, Copilot plus ChatGPT, 35 minutes; 20 of 21 finished a working program; struggling students often left with an illusion of competence. Not a homeschool trial and not a hardware study. The sitting condition — a tool that produces the response — is the mechanism used here.

[37] Hamsa Bastani et al., "Generative AI without Guardrails Can Harm Learning: Evidence from High School Mathematics," *Proceedings of the National Academy of Sciences* 122, no. 26 (2025): e2422633122. GPT Base raised assisted practice grades 48 percent relative to control, then cut unaided exam grades 17 percent, in a high-school mathematics randomized trial. Not a computing trial.

Do not police the child's sentence with a detector. Detectors flag human writing as generated at high rates in the study this book uses; they are the wrong tool for a caption about a key.[38] Do not ask a model to invent a network trace or a benchmark. Do not park an agent as the child's only partner. Hermes Agent is not Hermes 4. Grok is not Grok Bot. Copilot and ChatGPT are not required for this hour. A keyboard, a cable, and a notebook are tools too. Use them, then fade them.

What "done enough" looks like

Placement is by skill, not birthday. A "grade 2 computer" book is a publisher's scope, not a legal grade. A parts poster, a well-used older workbook, and a library picture book can place the same child in three different rooms, because they cut the grain differently. Choose by fit. This book names programs as options in the resources chapter. It does not rank them. The transcript, when you need one, still says **Computer Science**. It does not say a brand. It does not say PC Repair.

NCAA may treat networks and hardware *design* as contributing content if a later high-school year is submitted as math or science and is academic. Basic usage and repair will not contribute. That is eligibility, not a reason to skip this looking hour, and not a reason to title the year PC Repair.[39] There is no national computer-science credit. None of the five state homeschool statutes this book opened requires computer science. A path through computing is the promise. A percentile is not.

Checklist before moving on

- The child can name two parts they can touch, unaided, looking at the machine, not at a list.
- When asked what they pressed and what they saw, the child points and says a sentence. Reciting "the computer is thinking" has receded as the way to *do* the hour.

[38] Weixin Liang et al., "GPT Detectors Are Biased Against Non-Native English Writers," *Patterns* 4, no. 7 (2023): 100779. Average false-positive rate 61.22 percent on human TOEFL essays in that study. One sentence: do not police the child's caption with a detector.

[39] NCAA High School Review Committee, *Policies and Procedures 2026–27*, Appendix A, as in note 28. NCAA Eligibility Center, Division I Academic Standards fact sheet (September 2023, still hosted 1 September 2026): 16 core courses; computer science is not named as an additional discipline. A student can meet Division I without a computer-science year. http://fs.ncaa.org/Docs/eligibility_center/Student_Resources/DI_ReqsFactSheet.pdf.

- One problem has been described in accurate words: will not turn on, app will not open, or a cousin of those.
- Hardware and software have been told as a story that includes a press and a letter, not as two vocabulary cards.
- You can hear a catalogue with no finger, a generated teardown, and a repair year in disguise, and you can ask a good question, without taking the keyboard.
- Sits can be short. A spoken sentence is enough at 5–10. You did not require a paragraph or a certification checklist.
- Ages 11–14, if you are there: one systematic list (power, connections, restart) or one tradeoff sentence, plus the names and the press.
- Ages 15–18, if you are there: one comparison of application / system software / hardware in ordinary language, still without a repair year.

If most of that list is true, go on to the home screen, one named file, and log off, even if the birthday says otherwise. If the birthday says “ninth grade” and the words are still doing the work of the objects, stay. The next chapter is the software layer that notices the key, with the keyboard still in the room.

A path through computing is the promise. A percentile is not.

Chapter 2

Operating systems

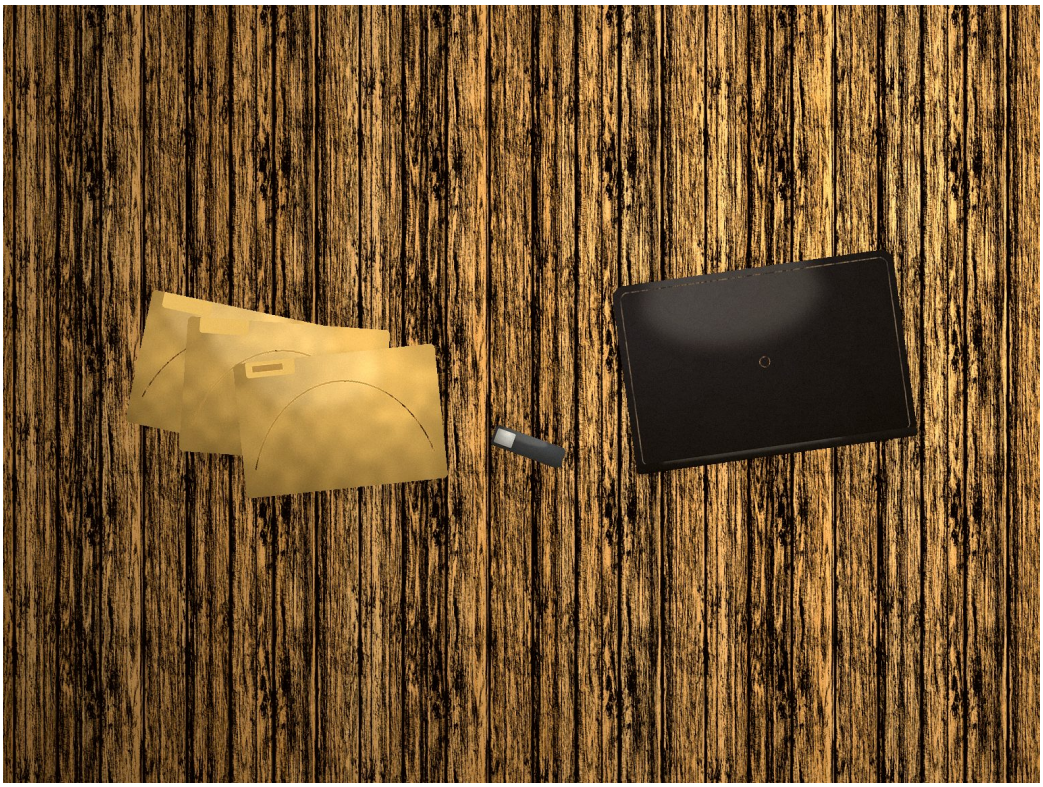

Figure 5: A kitchen-table still-life from a high three-quarter view: a notebook with a filename written on the cover, a closed laptop, a paper folder tab standing like a file. No people. No logos. No readable screen.

Why this matters

An operating system is the software that helps hardware and applications talk. You already taught a key press and a letter. This week you name the layer in between: something noticed the press and handed it to the program that drew the letter. At five to ten that layer may be named at all as the thing that shows a home screen of apps. At eleven to fourteen the child can tell an application from the thing that launches it. At fifteen to eighteen the floor is the system-software layer — application, system software, hardware — and categorizing operating-system roles sits *beyond* that floor. Files and accounts sit next to this idea as literacy plus the computer-science word *data*: a folder, a filename, a save, a password the child does not share, log off.

That is worth the struggle because a year of tapping icons with no layer underneath is computer *use* without a system. A child who can save one named file in one named folder and find it tomorrow has more of this week's idea than a child who can chant *operating system* with nothing on the table. A child who can still find the file after log-off has more than a child who says "I saved it" with no name and no place. The first is the start of a system you can talk about. The second is a performance. Both have a place. Only the first is the promise.

The maps are maps. CSTA's 2017 document does not print "operating system" in its early identifiers. California's grades 3–5 example does: a keyboard detects a key press, which the operating system and a word-processing application then handle. CSTA 2017 puts "Categorize the roles of operating system software" at Level 3B — ages sixteen to eighteen, specialty, not the floor for every student. So: five-to-ten may hear the word; eleven-to-fourteen can tell application from launcher; fifteen-to-eighteen floor owns the system-software layer; beyond the floor owns categorization. A standalone "Operating Systems" on a transcript is specialty, not the floor. The credit name a stranger can read is still **Computer Science**. It is never a brand of system software.[40]

[40] California Department of Education, *Computer Science Standards for California Public Schools* (adopted 6 September 2018), 3-5.CS.2 descriptive example: keyboard, desktop, monitor, operating system, and word-processing application as interacting parts. Examples are not compulsory. Public-school standards, not a homeschool statute. https://www.cde.ca.gov/be/st/ss/documents/csstandards.pdf. Computer Science Teachers Association, *CSTA K–12 Computer Science Standards, Revised 2017*: no "operating system" identifier at Levels 1A or 1B; 3A-CS-02 compare levels of abstraction among application software, system software, and hardware; 3B-CS-01 "Categorize the roles of operating system software" (ages

Files are data that live somewhere. CSTA's early data line is ordinary: store, copy, search, retrieve, modify, and delete information using a computing device, and define the stored information as data. Keep login information private, and log off. ISTE's student profile asks learners to protect digital privacy on devices and manage personal data. That is files and accounts as habits plus a CS word. It is not a corporate directory. It is not registry editing because the chapter title contains "operating systems."[41]

A 1999 National Research Council report on fluency with information technology still names the split this book uses: contemporary skills (using today's applications), foundational concepts (principles of computers, networks, information), and intellectual capabilities (applying IT to solve problems). Organizing and finding information sits among the capabilities. The report is undergraduate-focused. The *idea* survives on a kitchen table: a file you can find is a skill; knowing that the saved thing is data is a concept.[42]

This week you can learn how the home screen is a software layer, and how to hear "I saved it" with no filename and no folder as a wrong turn, not a cute slip. Today the student can save one named file in one named folder, log off, and find it tomorrow.

For the parent: understand it yourself

You do not need to administer a network. You do need to understand today's idea well enough to hear "I saved it," "it's in the computer," and "the home screen *is* the app" not as cute slips.

Many adults feel rusty around folders they have not thought about since a first

16–18, specialty). Levels 1A–3A for all students; 3B specialty. Not national law. Access date for URLs in these notes: 1 September 2026.

[41]CSTA 2017, 1A-DA-05: store, copy, search, retrieve, modify, and delete information using a computing device and define the information stored as data. 1A-IC-18: keep login information private, and log off of devices appropriately. International Society for Technology in Education, *ISTE Standards* © 2024 4.02, 1.2.d: take action to protect digital privacy on devices and manage personal data and security while online. ISTE Students are technology-for-learning standards, not CS standards, and have no grade bands. https://iste.org/standards/students.

[42]National Research Council, *Being Fluent with Information Technology* (Washington, DC: National Academies Press, 1999). Three strands: contemporary skills, foundational concepts, intellectual capabilities. Undergraduate-focused; not a K–12 standard. Organizing and finding information among capabilities/skills. ERIC ED440625.

office job, or around a home screen they tap without naming. That feeling is common. It is not a verdict. Five minutes of this section, then the warm-up at the end, is enough for tomorrow. The child still does the saving.

Everyday picture. A home screen of pictures. You say, while it is true: "These pictures are apps. Something underneath is showing them." You open one app. You type one sentence. You save it with a real name in a real folder you both can see. You log off. Tomorrow you come back and find it. Then you ask, "Where did you put it, and how will you find it tomorrow?" and you wait. The work is the place and the name. The word *kernel* can wait.

Precise picture. An operating system is system software. It is not the game. It is not the document app. It is the layer that starts the machine, notices input, manages files, and launches applications. California's example is the cleanest kitchen sentence this book has: the keyboard (hardware) detects a key press, which the operating system and the word-processing application then use. At the high-school floor, CSTA asks students to compare levels of abstraction among application software, system software, and hardware. Categorizing the *roles* of operating-system software — memory, files, devices, accounts as a list of jobs — is specialty. Teach the layer this week. Save the catalogue of roles for a student who has the layer and wants more.[43]

A file is stored information. The CS word is *data*. A folder is a named place. Search is a tool, not a substitute for knowing where you put something. This book's planning picture, labelled as planning rather than as a measured law of age, is: ages 5–10, save *one* named file in *one* named folder and find it tomorrow; ages 11–14, nested folders plus "search is a tool, not a substitute"; ages 15–18, paths, extensions, and "Downloads is not a filing system." There is no opened trial that children miss a critical period for folder brains. Journalism about a "search-not-folders generation" is not a finding you need. Teach the file. Find it tomorrow.[44]

A password is a key you do not share. Log off is how you put the key away. Permissions — who can read, who can change — are a system idea at fifteen to

[43]CSTA 2017, 3A-CS-02 and 3B-CS-01, as in note 39. *K–12 Computer Science Framework* (2016): framework statements are not standards, not curriculum, not national law. https://k12cs.org/faq/.

[44]Age-band file picture in this chapter is a planning inference for the teaching manual (one named file at 5–10; nested folders and search-as-tool at 11–14; paths and Downloads at 15–18), not a measured critical period. No opened child-sample trial of folder versus search models was used as a finding.

eighteen, with the parent holding administrator. A locked box and two keys on paper will teach the idea without opening the family's account panel as a first hour.

When the child misses this week's idea, recast or prompt. Recast: "You saved it. The name is *Tuesday note*. The folder is *School*." Prompt: "Where did you put it — on the home screen, or in a folder with a name?" If they are still stuck, give the name and the folder, have them find it, and stop. You will not interrupt every save. You will not pretend a missing filename is a cute slip.

After you ask, wait. Stahl's think-time is a general-classroom habit: a distinct period of uninterrupted silence. A slow three is a useful minimum. Hunting a save dialog is task-completion work-time. It can honestly be a minute. Look at the notebook with the filename, not at the child's face, if the silence is hard.[45]

A generated "file system diagram" is a picture to look at together, labelled generated. It is not the file in the folder.

Wrong answers you should be able to hear

1. "I saved it" with no name and no folder. The document vanished into a default place the child cannot name.

What it usually means. Saving has been a button, not a place. The layer is still invisible.

What to say. "Where did you put it, and how will you find it tomorrow?" Wait. If they are still stuck: "Let's give it a name. Let's pick a folder. Tomorrow we will look there first." They do it. You go on.

2. The home screen *is* the app. Quitting the game shut the whole device, or they cannot say what is showing the icons.

What it usually means. Application and launcher are one blob. Ordinary at five to ten. The 11–14 move is to pull them apart.

What to say. "Is that an app, or the thing that runs apps?" Quit the app without shutting the device. Then the reverse, once, with you watching.

[45] Robert J. Stahl, "Using Think-Time and Wait-Time Skillfully in the Classroom," ERIC Digest ED370885, May 1994. Think-time as uninterrupted silence; typical teacher pause 0.7–1.4 seconds; three seconds a useful minimum, not a cliff; task-completion work-time may run to minutes. General classroom, not a computer-science trial. https://files.eric.ed.gov/fulltext/ED370885.pdf.

3. "It's in the computer." Search found something. They cannot say whether it is the file they made.

What it usually means. Search did the finding. Knowing where you put it never had to work.

What to say. "Search found something — is that the same as knowing where you put it?" Close search. Open the named folder. If the file is not there, the save did not land where they thought. That is the lesson, not a scold.

4. A fluent sysadmin paragraph nobody in the house ran. A chatbot essay on process scheduling. A "native expert" voice. A certificate of OS fluency.

What it usually means. The work was outsourced. A layer you can point at on the home screen is this week. A catalogue of roles is later, and still the child's words.

What to say. "Show me the home screen. Save one file. Find it with the machine's extra windows closed." If they cannot, the paragraph comes off.

5. Rebuilding the family's accounts as the hour. A nine-year-old granted administrator. Registry talk. A brand of system software as the transcript title.

What it usually means. The chapter title ate the try-it. Parent holds admin. The child saves a file. The transcript says Computer Science, not a brand.

What to say. Log off as a habit. Who is allowed, on paper or in a dialog you already understand. Then stop.

Five-minute parent warm-up

Before the lesson, with the family device at the home screen, a scrap of paper, no student in the room.

1. Look at the home screen. Say out loud: "These icons are apps. Something underneath is showing them." Open a document app. Type one line. Save it as `Warmup-note` in a folder named `School` (or a name you already use).
2. Write today's opening on the scrap: "Where did you put it, and how will you find it tomorrow?" Say the question once. Count a slow three.
3. Write the wrong turn: "I saved it." Next to it, the repair: "What is the name? What is the folder?"
4. Log off. Notice that you still know the folder. That is the skill.
5. Put the pencil down. You are ready. The child still saves.

If those five minutes are fluent for you, you can hear a nameless save and a home-screen blob. That is the job.

How to teach it this week

Run the computing hour from the front of this book. This chapter fills it with the software layer, a file, and log off. Warm-up. Short model. Student attempt. One good question, then wait. Mixed practice. Unaided exit.

Warm-up (3–5 minutes, unaided). Yesterday's key press, if you have chapter 1 behind you: "What did you press, and what did you see happen?" Then: "What showed you the app?" No new account. No definition.

Short model (5–8 minutes). Home screen. You: "These pictures are apps. Something underneath is showing them." Open one. Type one sentence the child cares about. Save with a name you say out loud. Point at the folder. Then the question: "Where did you put it, and how will you find it tomorrow?" You are showing a layer and a place, not delivering a lecture on kernels.

Fade as soon as you can. Second sitting: they type while you name the folder. Third: they name the folder while you watch. Fourth: they save unaided, book closed. Include one incorrect example you say on purpose: "I saved it" with a shrug — then you wrinkle your face and repair it: "The name is ____. The folder is ____." The child hears the mismatch.

Exact wording you can steal.

On the file: "Where did you put it, and how will you find it tomorrow?" Then wait.

On the name: "What is the name you gave it?" Then wait.

On the layer: "Is that an app, or the thing that runs apps?" Then wait.

On a miss: "If you cannot find it, what words would you use?" Then wait.

On search: "Search found something — is that the same as knowing where you put it?" Then wait.

On leaving: "What would happen if we logged off?" Then wait.

On the chatbot: “If we cannot find the file with the extra windows closed, it is not tonight’s save. The model stays closed until you have tried.”

First try for the student, after the model. One sentence still to type. Product goal on a sticky note: save it with a real name in a real folder. Book closed for the attempt. You wait. Struggle before rescue. You do not grab the mouse.

One good question. After the attempt: “Where did you put it, and how will you find it tomorrow?”

Student attempt, then mixed practice. Two short saves of the new type, unaided, after the fade. Then mix: yesterday’s part name if you have chapter 1, last week’s “I saved it” repair, one log-off, one file they have not opened today.

Exit ticket (3–5 minutes, unaided). One question, folder still named on the sticky, one find they do alone. Tool closed. Tomorrow’s five-minute find is the real ticket. A fluent generated diagram is performance. The file they can still open is the test.

How to fade help. You save and say the path → they point while you save → they save while you point → they save unaided → they ask *you* where you put yours. When they can attach a name and a folder without your prompt, put the scaffold away. Bring it back when a nested folder or a permission arrives.

When to stop talking. After one model and one question. You hold the key: is there a named file, a named folder, a log-off, and an unaided try? You are not the control panel.

If typing is slow, that is not a reason to skip the file. You may type what they say. They still choose the name and the folder. Keyboarding as technique lives in a later literacy chapter. There is no opened standard that requires a speed gate before a child may save data. A child who can dictate a sentence, name it, and find it tomorrow has this week’s skill. A child who can type quickly and dump everything into Downloads does not.

If two ages share the machine, the younger child’s hour is the home screen and one file. The older child’s extra is app versus launcher, or who is allowed, on a second sitting. You do not owe both bands the same ten minutes. You owe each band one real try.

Safety this week is accounts. Parent-held login. The child does not create a new account as the hour. Save locally or in a parent-held cloud. No public share. Under thirteen, the parent holds the key; scripts and parent-child talk, not live open

chat as the default. COPPA covers operators collecting personal information from children under thirteen. The parent is usually not the operator of a classroom app. The parent's job is to choose services, hold logins, and sit with the child. COPPA does not keep a child from every kind of content. Teenagers are still a household-account hour. Do not reconfigure family accounts as a first five-to-ten sitting.[46]

Age-band moves

Place by skill, not by birthday. A sixteen-year-old who cannot find yesterday's file is still in the first band of this chapter for the *file*. They may still put Computer Science on a transcript if the year's work is systems plus programs they can trace. A ten-year-old who can quit an app without shutting the device is ready for the launcher try-it, not for a specialty course titled Operating Systems.

Ages 5–10. Hear the word if it helps: the software that notices a key press and shows a home screen of apps. Save *one* named file in *one* named folder. Find it tomorrow. Log off. Password as a key you do not share. Short sits. Then stop. You do not teach the registry. You do not grant administrator. Nested folders can wait.

Ages 11–14. Application versus the thing that launches it. Quit the app without shutting the device, then the reverse. Restart and update as a systematic list, next to chapter 1's power-cable-restart. Nested folders. Search is a tool, not a substitute. Permissions can begin as "who is allowed to see this," on paper.

Ages 15–18. System-software layer as the floor: application software, system software, hardware. Paths. Extensions. Downloads is not a filing system. Who is allowed: a sharing or permission dialog the parent already understands, or a locked box and two keys. Categorize OS roles — memory, files, devices, accounts as jobs the layer does — only if the floor is already easy. That categorization is specialty. A transcript line "Operating Systems" as a standalone course is specialty, not the 3A floor. The stranger-readable credit is still Computer Science.

[46] Federal Trade Commission, Children's Online Privacy Protection Rule, 16 CFR 312, 90 FR 16918 (22 April 2025); general compliance 22 April 2026 (passed as of access date). COPPA covers operators collecting personal information from children under 13. FTC FAQ: COPPA "was not designed to protect children from viewing particular types of content." Homeschool is not school authorization under the 2024 NPRM's unfinalized ed-tech amendments.

Named try-it: Home-Screen Layer

Time. 8–10 minutes.

Materials. The family device at the home screen.

Safety / accounts. Parent-held login. The child does not create a new account.

The fun. "These icons are apps; something underneath is showing them."

The skill. OS as the software that notices a key press and shows apps. Not sysadmin. Not a brand name as the credit.

How to run it. Sit at the home screen. Ask what the pictures are. Wait. Then: "What is showing them?" If they name an app as the whole machine, quit one app and notice the home screen is still there. Ages 5–10 stop at "something underneath." Ages 11–14 name application versus launcher. Ages 15–18 add the word *system software* as the floor between app and hardware.

Named try-it: One File, One Folder, Tomorrow

Time. 15 minutes today, 5 minutes tomorrow.

Materials. A word processor or notes app. A named folder. A notebook for the filename.

Safety / accounts. Save locally or in a parent-held cloud. No public share.

The fun. Hiding something and finding it.

The skill. Files as data that live somewhere. Store, retrieve, name. Not "the cloud is magic."

How to run it. Child types one true sentence. Child (or you, if typing is the bottleneck) saves with a spoken name in a spoken folder. Write the name on the notebook. Log off. Tomorrow: "Where did you put it, and how will you find it tomorrow?" Search stays closed on the first try. If they cannot find it, the save is the lesson. Ages 11–14 nest one folder inside another. Ages 15–18 say the path out loud and notice what Downloads collected without being asked.

Figure 6: A notebook with a dummy filename, a paper folder tab, a closed laptop. No readable UI. No brand. No children.

Named try-it: Log Off as a Habit

Time. 5 minutes.

Materials. The device. The parent-held account.

Safety / accounts. Parent holds the password. Child does not share it, write it on a sticky in a public room, or type it into a chat.

The fun. The click of being done.

The skill. Keep login information private; log off. Password as a key you do not share. Not a lecture on hashing.

How to run it. Save first, so log-off is not a loss. Then log off. "What would happen if we walked away without this click?" Wait. Tomorrow's find still works

if the file was in a named folder.

Debug / talk box

This is an illustration, not a reported family. A sentence has been typed. A save dialog is somewhere. The parent has not announced a definition of a file system.

Opening question

“Where did you put it, and how will you find it tomorrow?”

Not: “Define a file system.”

Follow-ups (pick three to five)

1. What is the name you gave it?
2. Is that an app, or the thing that runs apps?
3. If you cannot find it, what words would you use?
4. Search found something — is that the same as knowing where you put it?
5. What would happen if we logged off?

How to wait

Ask. Count a slow three. Look at the notebook with the filename if the silence is hard. Hunting a save dialog is task-completion work-time; Stahl named minutes as honest. You do not fill the silence with the path.

What a stuck silence usually means

They have only ever used Search. Or the question was a vocabulary quiz (*What is an OS?*). Or wait-time has been too short for years. Or they think “it’s in the computer” is enough. Next move: close search, open the named folder, ask the opening question again. Done enough: one named file in one named folder, found the next day, plus log off.

Practice that actually builds learning

Blocked, for a new move. The day you introduce the home screen as a layer, only the home screen. The day you introduce the file, only one save. The day you introduce log off, save first, then the click. Mixing too early makes the child hunt for the “right” school move instead of a place.

Mixed, for when to use it. Later the same week, the file reappears next to chapter 1's key press and this week's home screen. Mixing is choosing: is this a part you touch, a layer that shows apps, or a file you must find tomorrow?

Brief retrieval. Two minutes the next day: find the file, log off once, name one icon as an app. Cover vocabulary boxes. Retrieval is a second find, not a race.

One incorrect example to diagnose. After the child has saved, show a card that says "I saved it" with no name. Ask: "What did this person skip?" After a search-only find, close search and ask whether knowing the folder is the same skill. The child names the miss.

Use this week's talk box. Same opening. Same wait. Same stuck-silence meanings.

Named try-it: App vs Launcher

Time. 10 minutes. Ages 11–14, and 15–18 if the blob is still there.

Materials. One game or one document app.

Safety / accounts. Ordinary use. Parent-held login. No new account.

The fun. Quitting the app without shutting the whole device, then the reverse.

The skill. Application versus the thing that launches it. Not 3B roles. Not a brand war.

How to run it. Open the app. Use it for one minute. Quit the app. Notice the home screen or desktop is still there. Then, once, shut the device and start it again — with you watching — and notice how much longer that takes, and that every app went away. "Which move closed the game? Which move closed the layer?"

Named try-it: Who Is Allowed

Time. 15 minutes. Ages 15–18; a paper version can sit at 11–14.

Materials. A file's sharing or permission dialog the parent already understands, *or* a paper analogue: a locked box, two keys, two paper people.

Safety / accounts. Parent holds admin. Nobody "hacks the admin." Nobody changes family accounts as a first try. The paper analogue is enough.

The fun. “This person can read; that person cannot change.”

The skill. Permissions as a system idea. Not a corporate directory. Not a break-in.

How to run it. On paper: one key opens the box to look; the other would let someone change the note inside. Who gets which key? If you use a real dialog, you click; the child talks. “Who is allowed, and how would we check tomorrow?”

Talk every sitting. At 5–10 a spoken name and folder are done enough. You write them on the notebook only after they say them. At 11–14 a sentence that pulls app from launcher is the product. At 15–18 a path said out loud, or a who-is-allowed choice with a reason, is enough. You are not assigning a kernel essay.

If the family already lives in Search, do not scorn Search. Use it *after* the named folder has had a chance. Search is a tool. It is a poor only-method, because it cannot teach the idea that data lives in a place you chose. Tomorrow’s five minutes with Search closed is the practice that builds the idea. Then Search can come back as a helper.

A weekly shape you impose: one new move, blocked; then mixed retrieval of the file; then tomorrow’s find. The year is a map. This week is one layer, one file, one log-off.

For the student

You are learning that something underneath the pictures on the home screen is showing those pictures. That something is software. People call it an operating system when they want a name. You do not need every job it does yet. You need to see that an app is not the whole machine, and that a file you make has a name and a place.

You type one sentence. You give it a name. You put it in a folder. You log off. Tomorrow you find it. That is this week’s work. “I saved it” is not enough if you cannot say the name or the folder.

Tiny worked example

A sentence: *The cable was behind the plant.*

Name: `Cable-note`.

Folder: `School`.

Tomorrow: open `School`, open `Cable-note`. The words are still there.

That is a file. The saved thing is data. Data is information a computer is storing.

Two tries

1. Save one named file in one named folder. Write the name on paper. Log off. Tomorrow, find it without Search on the first try.
2. Look at the home screen. Point at two pictures. Say which are apps. Say what you think is showing them. Then quit one app without shutting the device.

Write nothing if writing fights the sitting. Talk the name and the folder. Your grown-up can remember them — and you still have to find the file.

Explain it back

Tell someone at the table where you put the file and how you will find it tomorrow. Then say whether the home screen is an app or the thing that runs apps. Then say why you log off.

Challenge

Some people tap Search for every file and cannot name a folder. Some people shut the whole device to leave a game. Some people say “I saved it” with no name. Some people want the transcript to say a brand of system software. What would those moves mean? How do you look at the *place* instead? How do you ask where you put it and how you will find it tomorrow?

You are allowed to struggle. You may use one folder, one name, one log-off. You save. You talk. If you get stuck, ask for a hint — not the path typed for you. Then try again.

A picture a program made of “how files work” is a scene. It is interesting. It is not your file in your folder.

Passwords are keys. You do not share yours, not with a sibling as a joke, not with a site that asks for it in a chat, not by writing it on a sticky the whole kitchen can read. Log off is how you put the key away when you leave the table. If someone else uses this computer, log off is how your file stays yours.

Some days the save dialog will feel like a wall of words. Ask for a hint: “Which of these two names is the folder we picked?” Two choices. Then you pick. You

still click. Your grown-up does not grab the mouse if you can still click.

If you are older, try this extra: say the path out loud. Notice what landed in Downloads without you choosing a folder. Pick one of those files and put it in a named place. Then, on paper, say who is allowed to read a note and who is allowed to change it. You still do not need administrator. Your grown-up holds that key.

If it isn't clicking

Three diagnostics. Each one has a next move. None of them is a verdict on talent.

1. The child cannot find yesterday's file, or says "I saved it" with no name, or thinks Search is the same as knowing the folder.

The place is not yet attached to the save. Next move: one folder, one name, written on paper *before* they click Save. Tomorrow, Search stays closed on the first try. "It's in the computer" is ordinary. Stay with one file. If this is still the bottleneck after several weeks of short daily saves, stay here. A permissions week on top of a child who cannot find a file will not hold. A human who will sit with a folder and wait, not a tool that "just finds it," is a reasonable next step if you have tried the one-file work and the find still does not come.

2. The child recites "operating system," or a video about kernels, and cannot quit an app without shutting the device.

The cue is still a performance. Next move: home screen first, quit second, question third. Cover the caption. Finger on an icon. "Is that an app, or the thing that runs apps?" After an attempt, one incorrect example ("this person shut the laptop to leave the game — what did they close?"). Slow down the "interesting" video until a sentence can be tied to the home screen. Go ahead once they can save, find, and say what the home screen is. A tutor is useful if recitation remains the default after a couple of weeks of daily file-first work *and* the hour has become a fight.

3. The child will not talk, or every sitting ends in a lost file and tears, and the silence after your question is a wall.

Talk is the product, and a shrug is the mind at rest. Next move: shorter sits, one sentence to type (or dictate), the opening question, wait a slow three twice. If silence is hard, look at the notebook with the filename. If typing is the whole bottleneck, you type what they say; they still choose the name and the folder. Keyboarding is

a later literacy hour. It is not a gate that blocks this chapter. If the hour has become a fight after a stretch of daily short sits, a human who will wait, not a chatbot that fills the silence, is the release valve.

When to slow down. Files still unnamed. Recitation still the default. Sits so long that saving never starts. A generated diagram doing the work of the folder. Rebuilding accounts as the first hour. Those are brakes. "Not ready" because of age is the brake this book will not use.

When to go ahead. The file is an easy find the next day. The child logs off without a reminder most days. An app can close without the device closing. Then chapter 3's packets have somewhere to sit. Being "good with computers" because a child can open apps is not a reason to skip the named folder.

When to get a human tutor. You have run the one-file try, or the covered-caption looking, or the wait after the question, for a stretch of daily sittings, and the same diagnosis is still in the room, and the hour has become a fight. A tutor is a release valve. Keep yourself as the person who can still hear "I saved it" and a brand as a credit name.

Tools, including AI

Optional helpers for you, the adult. The full rules live in the computing-hour box in the front of this book. This chapter is a pointer, not a second policy paper.

The child saves and finds first. You hold the account and the key. A tool may explain today's idea *to you* from a named page. If the model and the page disagree, the page wins. Extra isomorphic practice (new filenames, same find-it-tomorrow job) with the answers hidden is allowed. A labelled SCRIPT after the child tried, a hint after an attempt, a diagnosis of a save that already went wrong, talk-box questions you vet — those are allowed. Ages 5–10: scripts and parent-child talk, not live open chat.

Leave these out of the hour: a chatbot as the saver; "write my filename"; an unsupervised tab during the try; a model that invents a file tree the child never made; a certificate of OS fluency; a detector grading the child's sentence. A fluent paragraph about kernels is performance, not the child's knowledge of the folder. The unaided find tomorrow morning is the test.[47] If the tool produces the path, the

[47] Nicholas C. Soderstrom and Robert A. Bjork, "Learning Versus Performance: An Integrative

child is a spectator. The same crutch that can raise assisted practice and then fail a closed exam in mathematics is available whenever a fluent answer replaces the child's attempt.[48]

Do not let the model invent a network trace or a list of "what our computer is running." The oracle is the file on this machine, or a named page. Hermes Agent is not Hermes 4. Grok is not Grok Bot. Copilot and ChatGPT are not required for a save dialog. A notebook with a filename, a paper folder tab, and a log-off are tools too. Use them, then fade them.

What "done enough" looks like

Placement is by skill, not birthday. A publisher's "grade 6 computers" book is a scope, not a legal grade. The transcript, when you need one, says **Computer Science**. It does not say a brand of system software. It does not say Windows, or any other product name, as the credit. NCAA may treat operating systems as contributing content in a later high-school year submitted as math or science. Using office apps will not make that year. This chapter's file-and-log-off work is still the floor the later year sits on.[49]

Checklist before moving on

- One named file in one named folder, found the next day, unaided, Search closed on the first try.
- The child can say, in ordinary language, that the home screen of apps is being shown by software underneath — the operating system, if you used the word.
- Log off is a habit the child can do without a speech.
- "I saved it" with no name has receded as the way to *do* the hour.

Review," *Perspectives on Psychological Science* 10, no. 2 (2015): 176–199. James Prather et al., "The Widening Gap," ICER 2024 / arXiv 2405.17739: n = 21 CS1 students; 20 of 21 finished a working program with Copilot plus ChatGPT; struggling students often left with an illusion of competence. Not a homeschool OS trial. The sitting condition is the mechanism used here.

[48] Hamsa Bastani et al., "Generative AI without Guardrails Can Harm Learning," *PNAS* 122, no. 26 (2025): e2422633122. High-school mathematics: GPT Base +48 percent on assisted practice, −17 percent on an unaided exam. Not a computing RCT.

[49] NCAA High School Review Committee, *Policies and Procedures 2026–27*, Appendix A: operating systems contribute to computer-science approval; usage of word processing and related programs does not. Athletic eligibility, not a diploma rule. Transcript titles in this book: Computer Science, not a product name.

- You can hear a Search-only find, a home-screen blob, and a brand as a credit name, and you can ask a good question without taking the mouse.
- Ages 11–14, if you are there: quit an app without shutting the device; nested folder optional.
- Ages 15–18, if you are there: system software named as a layer; who-is-allowed on paper or in a dialog you already understand. Categorizing OS roles is extra, not the gate.

If most of that list is true, go on to packets, protocols, and the router as an object, even if the birthday says otherwise. If the birthday says “tenth grade” and the file still cannot be found tomorrow, stay. The next chapter is how devices talk, with the named folder still in the room.

A path through computing is the promise. A percentile is not.

Chapter 3

Networking

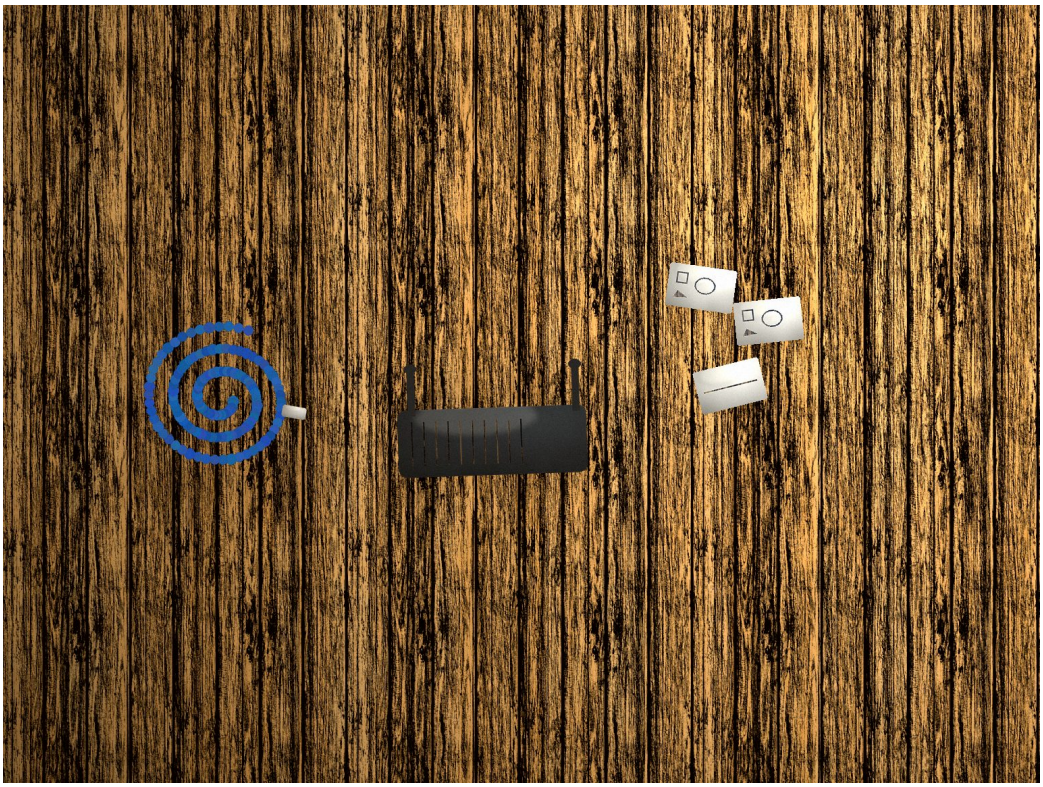

Figure 7: A kitchen-table still-life from a high three-quarter view: word cards in a line, a coiled unbranded cable, a closed router as a plain box. No people. No logos. No readable admin screen.

Why this matters

A network is how devices talk. Last week a file lived in a folder. This week a message leaves one room and arrives in another — in pieces, by agreement, sometimes scrambled so a sibling cannot read it without a key. At five to ten the idea is human: computers connect you to other people, places, and things, and a password protects a device. At eight to eleven, still inside that band's older half, information is broken into smaller pieces, sent as packets through more than one device, and put back together. At eleven to fourteen the child can model a protocol as agreed rules, and can scramble a message with a key as a *model* of secure transmission. At fifteen to eighteen the floor names devices with jobs: a router, a switch, a server, an address. Malware, and the tradeoff between ease and safety, wait for the next chapter's older band and for ordinary language, not for a break-in.

That is worth the struggle because "the Wi-Fi" is not a cloud with a mood. It is a system. A child who can cut a sentence into cards, send them down a hall, and reassemble them has more of this week's idea than a child who can chant *packet* with nothing on the table. A child who can say one rule both rooms had to follow has more than a child who says "it just goes." The first is the start of a network you can talk about. The second is a shrug. Both happen at kitchen tables. Only the first is the promise.

The maps are maps. The Framework's Networks overview says that in early grades students learn that computers connect them to other people, places, and things around the world, and that they learn how to protect their personal information. Details of towers and routing are not expected at that level. CSTA's early line is a password: what it is, why we use it. A little later: information broken into packets. California's grades 3–5 follow the same spine. At eleven to fourteen: model the role of protocols; physical and digital security measures; apply more than one method of encryption to *model* secure transmission. At the high-school floor: routers, switches, servers, topology, addressing. None of that is a home-school statute. None of it is a hacking lab.[50]

[50] *K–12 Computer Science Framework* (2016), Networks and the Internet overview: in early grades, students learn that computers connect them to other people, places, and things around the world, and how to protect their personal information; details of towers and routing are not expected at this level. Computer Science Teachers Association, *CSTA K–12 Computer Science Standards, Revised 2017*: 1A-NI-04 explain what passwords are and why we use them; 1B-NI-04 information broken into packets; 2-NI-04 model the role of protocols in transmitting data across

An athletic-eligibility rule set draws a useful cut for a later transcript. Networks and hardware design can contribute to a computer-science course submitted as math or science. Basic internet navigation — using a browser, clicking a link, downloading a file — does not. This chapter teaches the system. The next chapter will teach reaching other people, and will keep browsing in Digital Literacy when that is all the hour was. The stranger-readable title for *this* week's system work is **Computer Science**. It is never "Wi-Fi."[51]

Unplugged work can show what a packet is. CS Unplugged, a University of Canterbury project, has routing and network activities as enrichment. It is not a complete curriculum. The project's later voice is blunt: Unplugged never sought to replace programming, and treating it as a whole course can justify a poor decision. Unplugged can act as a catalyst when combined with a machine hour. If a computer is available and wanted, see a download that can pause after the cards have had their say. Cards are not a networking credit by themselves.[52]

This week you can learn how a protocol is an agreed set of rules so two machines can talk, and how to hear "the Wi-Fi is thinking" as a wrong picture. Today the student can send a cut-up sentence to another room, reassemble it, and say one rule you both had to follow.

---

networks and the Internet; 2-NI-05 physical and digital security measures; 2-NI-06 apply multiple methods of encryption to model secure transmission; 3A-NI-04 routers, switches, servers, topology, addressing. California Department of Education, *Computer Science Standards for California Public Schools* (2018), 3-5.NI.4 follows the packet spine. Framework and CSTA are maps, not homeschool statutes. Access date for URLs in these notes: 1 September 2026.

[51] NCAA High School Review Committee, *Policies and Procedures 2026–27*, Appendix A: networks and hardware design contribute to computer-science approval; "Basic internet navigation skills (e.g., using a web browser, hyperlinks, downloading files, etc.)" do not. Athletic eligibility, not a standards body.

[52] Tim Bell, Ian H. Witten, and Mike Fellows, *Computer Science Unplugged* (2015 revision): activities introduce building blocks of how computers work without using a computer. Tim Bell, "CS Unplugged or Coding Classes?," *Communications of the ACM* 64, no. 5 (2021): Unplugged can be misused as a complete curriculum; "Unplugged never sought to replace computer programming"; unplugged-first work Bell reports (Hermans and Aivaloglou, that study's PDF not opened here) showed no difference in understanding of programming concepts, with more self-efficacy and a wider vocabulary of commands in the unplugged-first group; Bell's gloss is catalyst when combined with programming. "CS Unplugged is not a curriculum."

For the parent: understand it yourself

You do not need to administer a network. You do need to understand today's idea well enough to hear "the Wi-Fi is thinking," "it just goes," and "let's hack the router" not as cute slips.

Many adults feel rusty around packets they have never had to name. That feeling is common. It is not a verdict. Five minutes of this section, then the warm-up at the end, is enough for tomorrow. The child still carries the cards.

Everyday picture. A sentence on a strip of paper: *The plant is on the sill.* You cut it into words. A child walks cards to the other room, not necessarily in order. The other room has a rule you both agreed on before you started — maybe a number on the back of each card. They put the sentence back. Then you ask, "If we cut the message into pieces, how does the other room know how to put it back?" and you wait. The work is the pieces and the rule. The word *TCP* can wait.

Precise picture. A packet is a piece of a message with enough extra information that the other end can put the message back. A protocol is the agreed set of rules so both ends do that the same way. Encryption, at eleven to fourteen on this map, is a *model*: scramble with a key, unscramble with the key, notice that a sibling without the key cannot read it. It is not a career. It is not a promise about the family's wireless network. A router, in this house, is a box that is a computer. It has a job: it helps messages find a path. Looking at the box is this week. Opening its settings is a parent job. Reconfiguring it is not a first five-to-ten hour. This book will not tell you how to break into it, test it as an attacker, or write a procedure that treats your home network as a lab to exploit. There is nothing to find down that road in these pages.[53]

Passwords at five to ten are citizen habits *and* the start of a CS idea. What is a password, and why do we use it? Because a device and the information on it can be reached by someone who is not allowed. The Framework's later intersection example is useful when the child is older: a student who understands how a program can try a list of words in a split second is a student who will probably not use a dictionary word for a password. That sentence is why-and-how helping use. This week, at the early band, the habit is enough: a key you do not share. The list-of-words picture can wait until the child has a protocol and a model of

[53] CSTA 2017, 2-NI-06, as in note 49: encryption to *model* secure transmission, not as a career and not as a test of a home network. Home network settings are a parent job.

scrambling.[54]

When the child misses this week's idea, recast or prompt. Recast: "The cards travelled. The numbers on the back were the rule." Prompt: "What travelled? What stayed here?" If they want to "hack" because the word is in the air, redirect to the model: a toy scramble on paper, a password as a key, a router as a box you look at. You will not interrupt every card. You will not pretend "the Wi-Fi is thinking" is a cute slip.

After you ask, wait. Stahl's think-time is a general-classroom habit. A slow three is a useful minimum. Reassembling cards is task-completion work-time. It can honestly be a full minute. Look at the cards, not at the child's face, if the silence is hard.[55]

A generated "map of the Internet" is a picture to look at together, labelled generated. It is not a packet the child carried.

Wrong answers you should be able to hear

1. "The Wi-Fi is thinking." A page loaded slowly, or a call froze, and the child gave the air a mind.

What it usually means. Everyday talk treats delay as thought. Packets and paths are not yet attached to a thing.

What to say. "If we cut the message into pieces, how does the other room know how to put it back?" Wait. If they are still stuck: "Pieces travelled. A rule put them back. The box in the hall is part of the path." They say a version of that. You go on.

2. "It just goes." No pieces, no rule, no other room.

What it usually means. The message is still magic. The cards have not yet had to work.

---

[54] *K–12 Computer Science Framework* (2016), PDF p. 13: a student who knows how to program a computer to iterate over all of the words in a list in a split second is a student who will probably not use a dictionary word for a password; understanding why and how computers work helps students make good decisions about use.

[55] Robert J. Stahl, "Using Think-Time and Wait-Time Skillfully in the Classroom," ERIC Digest ED370885, May 1994. Think-time as uninterrupted silence; three seconds a useful minimum, not a cliff; task-completion work-time may run to a minute or more. General classroom, not a computer-science trial. https://files.eric.ed.gov/fulltext/ED370885.pdf.

What to say. Cut the sentence. Send it. Ask what travelled and what stayed. If a piece is missing, can we tell?

3. Protocol as a vocabulary word. They can say *protocol* and cannot name one rule you both agreed on.

What it usually means. Coverage won. The hour became a glossary.

What to say. Invent a silly handshake: knock twice, then speak. Skip a step on purpose. "What rule did we agree on before we started?" The word can come after the miss.

4. "Let's hack it." The router, a neighbor's network, a toy "password cracker," a video of someone else's break-in, treated as this week's computer science.

What it usually means. A word in the air ate the model. Encryption at this band is scramble-and-key on paper. It is not an attack.

What to say. "We are going to scramble a sentence so a sibling cannot read it without the key. That is the model. We are not going to treat the family box as a lock to pick." Then the cards. Then stop.

5. The router as a mystery blob, or as a sysadmin hour. Either no one has ever pointed at the box, or the first sitting is the parent's settings page with a child clicking.

What it usually means. Object and admin got mashed. The child looks at the box. The parent holds the settings. Looking is CS. Reconfiguring is a parent job.

What to say. Five minutes, parent-led: "This is a thing in our house that is a computer." Unplug nothing unless you, the adult, chose that for a later, careful try. Child does not open the admin page.

Five-minute parent warm-up

Before the lesson, with index cards, a pencil, and a view of the family router as a closed box, no student in the room.

1. Write a short sentence. Cut it into words. Number the backs. Walk the cards to another room yourself. Put them back. Say out loud: "I am not going to ask what a packet is. I am going to ask how the other room knew the order."

2. Write today's opening on the scrap: "If we cut the message into pieces, how does the other room know how to put it back?" Say it once. Count a slow three.
3. Write the wrong turn: "The Wi-Fi is thinking." Next to it, the repair: "What travelled? What rule did we agree on?"
4. Look at the router. Do not open its page. Notice it is a box. That is the object.
5. Put the pencil down. You are ready. The child still carries the cards.

If those five minutes are fluent for you, you can hear a mind in the air, a glossary with no rule, and a break-in in disguise. That is the job.

How to teach it this week

Run the computing hour from the front of this book. This chapter fills it with people, packets, a rule, a toy scramble, and a box. Warm-up. Short model. Student attempt. One good question, then wait. Mixed practice. Unaided exit.

Warm-up (3–5 minutes, unaided). Yesterday's file, if you have chapter 2 behind you: "Where did you put it, and how will you find it tomorrow?" Then: "If we wanted that sentence to reach the other room, what would have to travel?" No new account. No router settings.

Short model (5–8 minutes). A sentence, cut. You number the backs. You agree the rule out loud: "Lowest number first." You send. They reassemble — or you reassemble once while they watch, then they do it. Then the question: "If we cut the message into pieces, how does the other room know how to put it back?" You are showing pieces and a rule, not delivering a lecture on routing tables.

Fade as soon as you can. Second sitting: they cut. Third: they invent the numbering. Fourth: they send without you in the hall. Include one incorrect example you say on purpose: send the cards with *no* numbers — then notice the other room cannot be sure. Repair: put the numbers back. The child hears the miss.

Exact wording you can steal.

On packets: "If we cut the message into pieces, how does the other room know how to put it back?" Then wait.

On what moved: "What travelled? What stayed here?" Then wait.

On the rule: “What rule did we agree on before we started?” Then wait.

On a missing piece: “If a piece is missing, can we tell?” Then wait.

On a password: “Who is allowed to know it, and why?” Then wait.

On a mind in the air: “Was that thinking, or were pieces taking a path?” Then wait.

On a break-in request: “The model is a scramble on paper. The box stays a box we look at.” Then the cards.

First try for the student, after the model. One sentence still on a strip. Product goal on a sticky: cut, send, reassemble, say the rule. You wait in the first room. Struggle before rescue. You do not carry the cards for them.

One good question. After the attempt: the opening question, now without pointing at the numbers first.

Student attempt, then mixed practice. Two short sends of the new type, unaided, after the fade. Then mix: yesterday’s filename if you have it, last week’s “I saved it” repair, one handshake protocol, one send with a piece held back on purpose.

Exit ticket (3–5 minutes, unaided). One sentence, one send, one rule said alone. Tool closed. Cards on the table. A fluent generated “how the Internet works” paragraph is performance. The reassembly they can still do is the test.

How to fade help. You number and send → they send while you number → they number while you watch → they invent the rule unaided → they ask *you* to be the other room. When they can attach pieces to a rule without your prompt, put the scaffold away. Bring it back when encryption as a model or a named device arrives.

When to stop talking. After one model and one question. You hold the key: is there a message in pieces, a rule, a reassembly, and an unaided try? You are not the router’s settings page.

Safety this week is ordinary movement and ordinary accounts. No running into furniture. Adult watches the hall. No child account for a network try. Parent only on any admin page. Child looks at the box. Do not reconfigure the family router as a first K–5 hour. Do not write, search for, or follow exploit steps, payloads, or attack procedures for the family router, the wireless network, a neighbor’s network, or any other system. A toy cipher on paper is a model. It is not a break-in.

If two ages share the hour, the younger child's sitting is people-places-things and packet cards. The older child's extra is the handshake, the scramble, or the router as object, on a second sitting. You do not owe both bands the same fifteen minutes.

Age-band moves

Place by skill, not by birthday. A seventeen-year-old who cannot reassemble a cut sentence is still in the first band of this chapter for the *packet*. They may still put Computer Science on a transcript if the year's work is systems plus programs they can trace. A ten-year-old who can already say the numbering rule is ready for a handshake, not for a specialty course in network engineering.

Ages 5–10. Computers connect you to other people, places, and things. A password protects a device and information. Older half of the band (about eight to eleven): packets — cut, send, reassemble. Details of towers and routing are not expected. Short sits. Then stop. The child carries cards. The child says one rule. You wait. You do not start a protocol dump. You do not start malware. You do not start a break-in.

Ages 11–14. Model the role of protocols: a handshake the family invents, then a miss when someone skips a step. Physical and digital security in ordinary language: a lock on a door, a password on a file. Encryption as a model: scramble-and-key on paper. Not a certification. Not a career. A machine hour, if available: a download that can pause, so the cards' idea *runs*.

Ages 15–18. Named devices with jobs: router, switch, server, address, in ordinary language. The family router as an object you can point at. Topology as "who is connected to whom," on paper, not as a labelled picture you cannot check. Bandwidth, load, and delay as specialty talk if the floor is already easy. Malware in ordinary language belongs mainly in the next chapter. Usability versus security can begin as a tradeoff sentence here: a long password is harder to use and harder for a stranger; a sticky on the monitor is easier and worse. You still do not run an attack lab.

Named try-it: People, Places, Things

Time. 10 minutes. Ages 5–10; a quick reminder at any age.

Materials. Two devices in the house, or a drawing of "here" and "Grandma." Optional: a parent-held hello later, in chapter 4.

Safety / accounts. No child account. No public profile. Parent holds any login.

The fun. "This can reach that."

The skill. Devices connect people. Not a transport-layer lecture.

How to run it. Point at two things that can talk: a phone and a tablet, or "our table" and "Grandma's table." "What people, places, or things can this one reach?" Wait. If they name a brand of browser, come back to the people.

Named try-it: Packet Cards

Time. 15 minutes.

Materials. A sentence cut into word cards. Two rooms. A hallway. A pencil for numbers.

Safety / accounts. No running into furniture. Adult watches. No child network account.

The fun. The message arrives in pieces and has to be put back.

The skill. Information broken into packets, sent through more than one device (here: a person in a hall), reassembled. Unplugged as catalyst. Then, if a machine is available and wanted, see a download that can pause. Not a complete curriculum. Not a networking credit by itself.

How to run it. Agree the rule. Cut. Send. Reassemble. Ask the opening question. On a second round, hold one card back. "If a piece is missing, can we tell?" Ages 5–10 may stop at order-numbers. Ages 11–14 add a second rule (a knock before you enter). Ages 15–18 name the hall as a path with a job, like a router.

Figure 8: Word cards in a line, a coiled unbranded cable, a closed router as a box. No readable UI. No children. No logos.

Named try-it: Protocol as Agreed Rules

Time. 15 minutes. Ages 11–14, and 15–18 if the rule is still invisible.

Materials. A silly handshake the family invents (knock twice, then speak). Optional: the packet cards again.

Safety / accounts. Ordinary play. No network settings.

The fun. When someone skips a step, the message fails.

The skill. Model the role of protocols. Not a dump of public rule documents.

How to run it. Invent three steps. Practice once. Then you skip step two on purpose. “What rule did we agree on before we started?” If they cannot name it,

the protocol was only a dance. Name it. Try again. Then send one packet-card sentence that is allowed only after the handshake.

Debug / talk box

This is an illustration, not a reported family. Cards are on the far-room table. One may be missing. The parent has not announced a definition of a protocol.

Opening question

"If we cut the message into pieces, how does the other room know how to put it back?"

Not: "Define a protocol."

Follow-ups (pick three to five)

1. What travelled? What stayed here?
2. What rule did we agree on before we started?
3. If a piece is missing, can we tell?
4. Who is allowed to know the password, and why?
5. Was that the Wi-Fi thinking — or pieces taking a path?

How to wait

Ask. Count a slow three. Look at the cards if the silence is hard. Reassembly can honestly take a minute of work-time. You do not fill the silence with *packet*.

What a stuck silence usually means

They think "Wi-Fi" is a cloud, not a system. Or the question was a vocabulary quiz. Or wait-time has been too short for years. Or they want to "hack" because the word is in the air — redirect to the model. Next move: one short sentence, numbers on the back, opening question again. Done enough: the child can say, in ordinary language, that a message can be cut into pieces, travel, and be reassembled, and that a password is a key.

Practice that actually builds learning

Blocked, for a new move. The day you introduce people-places-things, only two devices and a pointing. The day you introduce packets, only the cards. The day

you introduce a handshake, only the knocks. Mixing too early makes the child hunt for the glossary word.

Mixed, for when to use it. Later the same week, the cards reappear next to the home-screen layer and the named file. Mixing is choosing: is this a file in a folder, a layer that shows apps, or a message in pieces?

Brief retrieval. Two minutes: reassemble a three-word sentence; name one rule; point at the router as a box. Cover vocabulary cards.

One incorrect example to diagnose. After a good send, send a pack with no numbers. Ask: "What did this person skip?" After "the Wi-Fi is thinking," point at the cards and ask what actually travelled. The child names the miss.

Use this week's talk box. Same opening. Same wait. Same stuck-silence meanings.

Named try-it: Scramble-and-Key

Time. 15 minutes. Ages 11–14; 15–18 as a quick model if needed.

Materials. Paper. A pencil. A Caesar-shift you both can do by hand (shift each letter one or two places in the alphabet), or Unplugged-style cryptography cards if you already own them.

Safety / accounts. This is a *model*, not a promise that the family's wireless network is "hacked," tested, or weak. No one applies the toy to a real login, a real router, or anyone else's system. No exploit steps. No payloads. No attack procedures.

The fun. A secret that a sibling cannot read without the key.

The skill. Encryption as a model of secure transmission. Not a security-industry course. Not a break-in.

How to run it. Pick a three-word sentence. Agree the key (for example: each letter moves one place). You scramble one word as a model. They scramble the rest. A sibling without the key tries to read it and cannot. Then they unscramble with the key. "What travelled? What stayed here — the key?" If they ask how to "do this to Wi-Fi," the answer is: we do not. The paper is the model. The box in the hall stays a box we look at.

Named try-it: Router as Object

Time. 5 minutes, parent-led.

Materials. The family router, looked at, not opened.

Safety / accounts. Parent only on any admin page. Child looks at the box. No reconfigure-the-network hour. No default-password folklore as a lab. No exploit steps.

The fun. "This is a thing in our house that is a computer."

The skill. A network has devices with jobs. Not a settings tutorial. Not a hacking lab.

How to run it. Walk to the box. "What do you actually *see*?" Lights, cables, a closed case. "This helps messages find a path." If a light is blinking, you may say that something is happening; you do not guess a hidden log. Ages 15–18 may add: this box has an address on the network, the way a house has a number on a street — ordinary language, not a configuration task. Then you leave the hall.

Talk every sitting. At 5–10 a spoken rule is done enough. At 11–14 a handshake plus a scramble. At 15–18 a named job for the box. You are not assigning a routing-table essay.

If a machine is available and wanted, plug the idea in: start a download you already trust, pause it, notice that pieces can stop arriving. Bell's point stands: unplugged work is a catalyst, not a year without a computer. If no machine is wanted today, the cards still taught the idea. Tomorrow you can plug it in.

A weekly shape: one new move, blocked; mixed retrieval of the cards; then the box as an object. The year is a map. This week is pieces, a rule, a key on paper.

For the student

You are learning how devices talk. A message can be cut into pieces. The pieces can travel. The other side can put them back if you both agreed on a rule first. People call those pieces packets when they want a name. People call the rule a protocol. You do not need every long word yet. You need to carry the cards and say the rule.

A password is a key. You do not share it. A box in the hall may be a computer with a job: helping messages find a path. You can look at it. You do not have to open it.

Tiny worked example

Sentence: *The plant is on the sill.*

Cards: The / plant / is / on / the / sill.

Rule: numbers on the back, lowest first.

Other room: puts 1, 2, 3, 4, 5, 6 in order. The sentence is back.

If card 4 stays in the hall, the other room can tell something is missing.

Two tries

1. Cut a sentence. Send it to another room. Reassemble it. Say one rule you both had to follow.
2. Invent a tiny handshake (knock twice, then speak). Skip a step on purpose. Say what went wrong. Then try a scramble: move each letter one place, and give the key only to the person who is allowed to read it.

Write nothing if writing fights the sitting. Talk the rule. Your grown-up can remember it — and you still have to put the cards back.

Explain it back

Tell someone at the table how the other room knew the order. Then say what travelled and what stayed. Then say who is allowed to know a password, and why.

Challenge

Some people say the Wi-Fi is thinking when a page is slow. Some people say *protocol* from a list and cannot name a rule. Some people want to “hack the router” because a video made that look like computer science. What would those three moves mean? How do you look at the *cards* instead? How do you ask how the other room puts the message back?

You are allowed to struggle. You may use cards, a hallway, a pencil, a closed box. You send. You talk. If you get stuck, ask for a hint — not the numbered answer key already written on every card. Then try again.

A picture a program made of “the Internet” is a scene. It is interesting. It is not the sentence you carried.

If you are older, try this extra: point at the family router and say one job it has, in ordinary words. On paper, draw three boxes — you, the hall box, the other room — and arrows for a message in pieces. You still do not open the settings. Your grown-up holds that key. A toy scramble on paper is a model of hiding a message. It is not a way to test anyone’s real lock.

If it isn’t clicking

Three diagnostics. Each one has a next move. None of them is a verdict on talent.

1. The child cannot reassemble the cards, or says “it just goes,” or gives the Wi-Fi a mind whenever anything is slow.

The pieces are not yet attached to a rule. Next move: a three-word sentence, numbers already on the backs, one hall. Cover the word *packet*. Ask what travelled. “The Wi-Fi is thinking” is ordinary. Stay with cards. If this is still the bottleneck after several weeks of short daily sends, stay here. A protocol week on top of a child who cannot put cards in order will not hold. A human who will sit with cards and wait, not a video that narrates the Internet, is a reasonable next step if you have tried the small-set work and the reassembly still does not come.

2. The child recites a networking video, or a list of device names, and cannot name one rule you both agreed on.

The cue is still a performance. Next move: handshake first, miss second, question third. Cover the caption. “What rule did we agree on before we started?” After an attempt, one incorrect example (cards with no numbers). Slow down the “interesting” video until a sentence can be tied to the hall. Go ahead once they can send, reassemble, and say the rule. A tutor is useful if recitation remains the default after a couple of weeks of daily card-first work *and* the hour has become a fight.

3. The child will not talk, or every sitting becomes a request to “hack something,” and the silence after your question is a wall.

Talk is the product, and a thrill-word is the mind at rest. Next move: shorter sits, one sentence, the opening question, wait a slow three twice. If silence is hard, look at the cards. For “let’s hack it,” redirect to scramble-and-key on paper and

the router as a closed box. One new rule a week is enough. If the hour has become a fight after a stretch of daily short sits, a human who will wait, not a chatbot that fills the silence with an adventure, is the release valve.

When to slow down. Cards still a jumble. Recitation still the default. Sits so long that sending never starts. A generated Internet map doing the work of the hall. A settings page as the first five-to-ten hour. Those are brakes. "Not ready" because of age is the brake this book will not use.

When to go ahead. A cut sentence is an easy reassembly. The child can name the rule. A password is a key they do not share. Then chapter 4's far-away hello and public-or-private have somewhere to sit. Being "good at Wi-Fi" because a child can join a network is not a reason to skip the cards.

When to get a human tutor. You have run the three-word send, or the covered-caption looking, or the wait after the question, for a stretch of daily sittings, and the same diagnosis is still in the room, and the hour has become a fight. A tutor is a release valve. Keep yourself as the person who can still hear a mind in the air and a break-in in disguise.

Tools, including AI

Optional helpers for you, the adult. The full rules live in the computing-hour box in the front of this book. This chapter is a pointer, not a second policy paper.

The child carries the cards first. You hold the account and the key. A tool may explain today's idea *to you* from a named page. If the model and the page disagree, the page wins. Extra isomorphic practice (new sentences, same cut-and-reassemble job) with the answers hidden is allowed. A labelled SCRIPT after the child tried, a hint after an attempt, a diagnosis of a send that already went wrong, talk-box questions you vet — those are allowed. Ages 5–10: scripts and parent-child talk, not live open chat.

Leave these out of the hour: a chatbot as the other room; "write my protocol"; an unsupervised tab during the try; a model that invents a network trace or a packet capture the child never made; a certificate of networking fluency; any request that would produce exploit steps, payloads, or attack procedures for the family router, the wireless network, or any other system. If a tool offers that road, you close it. The paper cards are the lab.

A fluent paragraph about routing is performance, not the child's knowledge of the hall. The unaided reassembly is the test.[56] If the tool produces the numbered cards, the child is a spectator. The same crutch that can raise assisted practice and then fail a closed exam in mathematics is available whenever a fluent answer replaces the child's attempt.[57] Do not let the model invent benchmark numbers or "what our router is doing." The oracle is the sentence on the far-room table, or a named page.

Hermes Agent is not Hermes 4. Grok is not Grok Bot. Copilot and ChatGPT are not required for a hallway. Cards, a cable you can point at, and a closed box are tools too. Use them, then fade them.

What "done enough" looks like

Placement is by skill, not birthday. A publisher's "grade 7 networks" book is a scope, not a legal grade. The transcript, when you need one, says **Computer Science**. It does not say Wi-Fi. It does not say a product name. NCAA may treat networks and hardware design as contributing content in a later high-school year submitted as math or science. Basic internet navigation will not make that year. This chapter's packets-and-rules work is the system floor the later year sits on.[58]

Checklist before moving on

- The child can say, in ordinary language, that a message can be cut into pieces, travel, and be reassembled.
- One rule both rooms followed can be named unaided.
- A password is a key they do not share, in a sentence they can say.
- "The Wi-Fi is thinking" has receded as the way to *do* the hour.

[56] Nicholas C. Soderstrom and Robert A. Bjork, "Learning Versus Performance: An Integrative Review," *Perspectives on Psychological Science* 10, no. 2 (2015): 176–199. James Prather et al., "The Widening Gap," ICER 2024 / arXiv 2405.17739: n = 21 CS1; 20 of 21 finished a working program; illusion of competence among struggling students. Not a networking trial. The sitting condition is the mechanism used here.

[57] Hamsa Bastani et al., "Generative AI without Guardrails Can Harm Learning," *PNAS* 122, no. 26 (2025): e2422633122. High-school mathematics: GPT Base +48 percent on assisted practice, −17 percent on an unaided exam. Not a computing RCT.

[58] NCAA HSRC 2026–27, as in note 50. Transcript titles in this book: Computer Science for the system; never a product name; never "Wi-Fi" as the credit.

- You can hear a glossary with no rule, a generated Internet map, and a break-in in disguise, and you can ask a good question without carrying the cards.
- Ages 11–14, if you are there: a handshake that fails when a step is skipped; a scramble-and-key on paper as a model.
- Ages 15–18, if you are there: the family router pointed at as a computer with a job; one ordinary-language address or path sentence. No settings hour required.

If most of that list is true, go on to reaching other people, public and private, and search-then-leave-the-page, even if the birthday says otherwise. If the birthday says “eleventh grade” and the cards still cannot be put back, stay. The next chapter is the internet as many computers talking, with the hallway still in the room.

A path through computing is the promise. A percentile is not.

Chapter 4

The internet

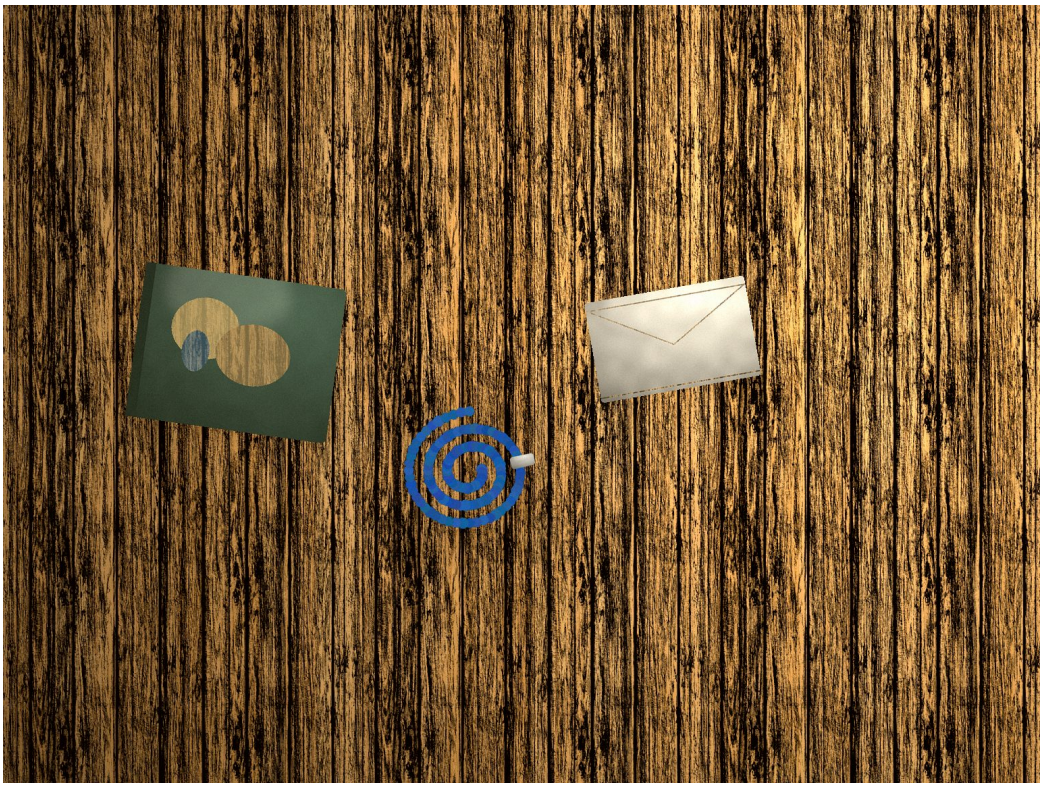

Figure 9: A kitchen-table still-life from a high three-quarter view: an unbranded map edge, a paper envelope as a message, a closed laptop. No people. No logos. No readable browser.

Why this matters

The internet is a system of networks, not a browser icon. Last week a message travelled down a hall in pieces. This week the same idea stretches: many computers, far away, talking. At five to ten the child learns that we can reach other people, that a password is a key, and — in the older half of the band — that packets still apply. At eleven to fourteen, protocols across networks and the Internet; public and private as a tradeoff; search, share, leave a footprint. At fifteen to eighteen, the floor adds scalability and reliability in ordinary language, malware as "software meant to harm or sneak," and privacy when collection is automatic. An introductory college-shaped computing course, if you later choose one, includes computer systems and networks as one piece of an exam, not as this week's floor.

That is worth the struggle because "I found it on the internet" is a start, not a source. A child who can send a parent-held hello and say whether it was private or public has more of this week's idea than a child who can chant *URL* with nothing on the table. A child who can leave a page and check who made it has more than a child who stays on the first hit until it feels true. The first is the start of a system you can talk about, plus a citizen move. The second is a shrug with a search box. Both happen at kitchen tables. Only the first is the promise.

Keep two columns honest. Column B is the system: many computers, packets, protocols across the Internet, public and private as a computing tradeoff. That work is **Computer Science**. Column A is the tool: opening a browser, clicking a link, downloading a file, searching. That work is **Digital Literacy**. An athletic-eligibility rule set puts the same cut in another language: using a web browser, hyperlinks, and downloading files do not contribute to a computer-science core course. Teach both this week if the family needs both. Label them honestly. A year of fluent browsing titled Computer Science has misled a stranger.[59]

The Framework says the Internet enables people to connect with others worldwide through many different points of connection. CSTA at eleven to fourteen asks students to model protocols across networks and the Internet, and to weigh public versus private. At the high-school floor: malware; privacy concerns related to au-

[59] NCAA High School Review Committee, *Policies and Procedures 2026–27*, Appendix A: content that does not contribute to computer-science approval includes "Basic internet navigation skills (e.g., using a web browser, hyperlinks, downloading files, etc.)." Content that contributes includes networks and hardware design. Athletic eligibility, not a standards body. *K–12 Computer Science Framework* (2016), pp. 12–13: computer literacy includes performing an Internet search; that use is distinguished from computer science. Access date for URLs in these notes: 1 September 2026.

tomated data collection. ISTE's student profile — a digital-literacy overlay, not a CS standard — asks learners to protect privacy, manage time online, and evaluate digital content for accuracy, validity, bias, origin, and relevance. A 1999 fluency report still names skills (use a browser, evaluate information) versus concepts (what a network is). This chapter owns the concept as CS and one looking move as literacy. It does not become a second book on evaluating claims. One move if a website is the source: who made this, and can we leave the page and check.[60]

A language model is not a search engine and not a witness to a page it did not open. A fluent citation can be fake. A 2023 study of short literature reviews found that more than half of one model's citations were fabricated, and about one in five of a later model's were. Of the real ones, many had substantive errors. That study is dated to those models. It is a demonstration you can run, not a 2026 rate you should quote as current. The named page on the table still wins.[61]

This week you can learn how the browser is a tool on a system, and how to hear "I found it on the internet" as a start, not a source. Today the student can send or receive one parent-held hello, then say whether that hello was private or public.

For the parent: understand it yourself

You do not need to be a network engineer. You do need to understand today's idea well enough to hear "I found it on the internet," "the browser *is* the internet," and

[60] *K–12 Computer Science Framework* (2016): the Internet enables people to connect with others worldwide through many different points of connection. Computer Science Teachers Association, *CSTA K–12 Computer Science Standards, Revised 2017*: 2-NI-04 model the role of protocols across networks and the Internet; 2-IC-23 public versus private; 3A-NI-05 malware; 3A-IC-29 privacy concerns related to automated data collection. International Society for Technology in Education, *ISTE Standards* © 2024 4.02: 1.2 Digital Citizen; 1.3 Knowledge Constructor, including evaluating accuracy, validity, bias, origin, and relevance of digital content. ISTE Students are not CS standards and have no grade bands. National Research Council, *Being Fluent with Information Technology* (1999): skills (use a browser, evaluate information) versus concepts (what a network is). Framework FAQ: complement computer literacy and digital citizenship; instruction in all three is important.

[61] William H. Walters and Esther Isabelle Wilder, "Fabrication and Errors in the Bibliographic Citations Generated by ChatGPT," *Scientific Reports* 13 (2023): 14045. GPT-3.5: 55 percent of citations fabricated; GPT-4: 18 percent fabricated. Of the real citations, 43 percent (GPT-3.5) and 24 percent (GPT-4) had substantive errors. Do not update those rates to 2026 models without a new opened paper. A model is not a search engine and not a witness to a page it did not open.

"the chatbot cited it, so it exists" not as cute slips.

Many adults feel rusty around a web they use every day without naming the system. That feeling is common. It is not a verdict. Five minutes of this section, then the warm-up at the end, is enough for tomorrow. The child still talks after the hello.

Everyday picture. A paper envelope. "Here" on one side of the table. "There" on the other — a relative, a friend, a library. You send a parent-held hello, or you walk the envelope. Then you ask: was that hello for one person, or for anyone who walked by? Then, if you open a page: "Who made this page, and how would we check without staying on it?" You wait. The work is the people and the check. The word *DNS* can wait.

Precise picture. The internet is many networks connected. A browser is software that asks for a page and shows it. Closing the browser does not shut the internet. A page is made by someone. Checking often means leaving that page: another source, a book already on the table, a known organization, a reverse search for a claim — not staying until the design feels official. Public versus private is a tradeoff, not a scare: a post Grandma can see is one choice; a classroom note is another; a message that can be copied is already less private than it feels. Malware, at fifteen to eighteen, is software meant to harm, sneak, or steal, spoken in ordinary language. A catalogue of attack families is not a five-to-ten hour. You do not treat a generated screenshot of a website as a visit.[62]

Digital citizenship — a password kept private, time online as a household choice, respectful talk — is complementary to computer science and distinct from it. The Framework says instruction in computer science, computer literacy, and digital citizenship all matter. Teach the habit here. If the hour was only browsing, the transcript word is Digital Literacy, not Computer Science. A five-to-ten-year-old does not need their own email. Under thirteen, the parent holds the key. Teenagers are still a household-account hour.[63]

[62] CSTA 2017, 3A-NI-05 and 3A-IC-29, as in note 59. College Board, AP Computer Science Principles course page (opened 1 September 2026): Computer Systems and Networks, including the Internet, listed among Big Ideas at 11–15 percent of exam score on the opened page. Optional later object, not this chapter's floor. Create performance task due 30 April 2027 and exam 14 May 2027 on that fetched page.

[63] *K–12 Computer Science Framework* (2016) FAQ: complement of computer literacy and digital citizenship. Federal Trade Commission, Children's Online Privacy Protection Rule, 16 CFR 312; general compliance 22 April 2026 (passed). COPPA covers operators collecting personal information from children under 13; FTC FAQ: not designed to protect children from viewing particular types of content. Photos of a child are personal information when collected by a covered

When the child misses this week's idea, recast or prompt. Recast: "You found a page. We have not yet checked who made it." Prompt: "Who made this page, and how would we check without staying on it?" If they want to "just Google it," leave the first hit on purpose. You will not interrupt every click. You will not pretend a fluent model-citation is a source.

After you ask, wait. Stahl's think-time is a general-classroom habit. A slow three is a useful minimum. Leaving a page to check can honestly take a minute of work-time. Look at the envelope or the named book, not at the child's face, if the silence is hard.[64]

A generated "screenshot of a website" is a picture to look at together, labelled generated. It is not a visit.

Wrong answers you should be able to hear

1. "I found it on the internet." Offered as a source, with no page, no maker, no check.

What it usually means. The web is still a place-name, not a system of pages with authors.

What to say. "Who made this page, and how would we check without staying on it?" Wait. Open a book you already trust on the same question. Compare. Then go on.

2. The browser *is* the internet. Closing the window felt like shutting the system, or they cannot say that many computers are talking.

What it usually means. Tool and system are one blob. Chapter 3's hallway has not yet stretched.

What to say. "Is this the internet, or a tool for asking the internet?" Close the browser. The family router, if you looked at it last week, is still a box. The system is still there.

---

operator.

[64] Robert J. Stahl, "Using Think-Time and Wait-Time Skillfully in the Classroom," ERIC Digest ED370885, May 1994. Think-time as uninterrupted silence; three seconds a useful minimum, not a cliff; task-completion work-time may run to a minute or more. General classroom, not a computer-science trial. https://files.eric.ed.gov/fulltext/ED370885.pdf.

3. A fluent answer from a model, treated as a page someone opened. "It cited three papers." None are on the table.

What it usually means. Language was mistaken for a witness. Models complete patterns. They do not owe you a page they did not open.

What to say. "Is a fluent answer the same as a page we opened? If the model named a paper, can we find it?" Open the ones that exist. Notice the ones that do not. Dated demonstration, not a scare about every sentence.

4. Public as private because it felt small. A post "just for friends," a photo of a child on a gallery, an address in a caption.

What it usually means. Audience is still "who I had in mind," not "who could see it." Photos of a child are personal information when a covered operator collects them. Prefer local files and parent-held accounts.

What to say. Two paper posts: one meant for Grandma, one meant for a classroom. Sort. "What would Grandma see if this were public?"

5. Browsing as the CS credit. A year of search practice titled Computer Science. A certificate of "internet fluency." A generated tour of "famous websites."

What it usually means. Column A ate Column B. Teach the search. Label it Digital Literacy. Keep the system sentences — many computers, packets, public and private — as Computer Science.

What to say. "If a stranger read the transcript, would they think this hour was a program of looking at pages, or a system of computers talking?" Then one hello, one check, one honest name.

Five-minute parent warm-up

Before the lesson, with a paper envelope, a book you already own, and the family device on a parent-held account, no student in the room.

1. Write a one-line hello on paper. Walk it to another room. Say out loud: "This could be private. A page on a screen might not be."
2. Write today's opening on the scrap: "Who made this page, and how would we check without staying on it?" Say it once. Count a slow three.
3. Write the wrong turn: "I found it on the internet." Next to it, the repair: "Which page? Who made it? What else could we open?"

4. Pick one factual question the book already answers. Do not search yet. Notice that the book is a check you can leave a page for.
5. Put the pencil down. You are ready. The child still talks.

If those five minutes are fluent for you, you can hear a source-shrug, a browser-as-system, and a fake citation. That is the job.

How to teach it this week

Run the computing hour from the front of this book. This chapter fills it with a hello, a key, a page you leave, a public-or-private sort, and — for older students — a citation hunt. Warm-up. Short model. Student attempt. One good question, then wait. Mixed practice. Unaided exit.

Warm-up (3–5 minutes, unaided). Yesterday's packets, if you have chapter 3 behind you: "If we cut the message into pieces, how does the other room know how to put it back?" Then: "If the other room were far away, what would have to be true?" No child public profile. No surprise tabs.

Short model (5–8 minutes). Envelope or parent-held hello. You: "Many computers, far away, talking." Send or receive one real hello, or walk the paper. Then: private or public? Then, if you open a page the parent already chose, ask the opening question. You are showing people and a check, not delivering a lecture on addressing.

Fade as soon as you can. Second sitting: they say whether the hello was private. Third: they suggest how to check a page. Fourth: they leave the page and open the book. Include one incorrect example you say on purpose: "I found it on the internet" as a complete source — then you wrinkle your face and repair it: "Which page? Who made it?" The child hears the miss.

Exact wording you can steal.

On a page: "Who made this page, and how would we check without staying on it?" Then wait.

On what they read: "What did you actually read?" Then wait.

On audience: "What would Grandma see if this were public?" Then wait.

On a model: "Is a fluent answer the same as a page we opened?" Then wait.

On a citation: “If the model named a paper, can we find it?” Then wait.

On a password: “Why not a dictionary word?” — when they are old enough for the list-of-words picture. Then wait.

On a search shrug: “Just finding it is a start. Checking is the hour.”

First try for the student, after the model. One hello still to send or receive, parent-held. Product goal on a sticky: say private or public. You wait. Struggle before rescue. You do not grab the account.

One good question. After the attempt: “Who made this page, and how would we check without staying on it?” — or, if you have not opened a page yet, “Was that hello private or public, and how do you know?”

Student attempt, then mixed practice. Two short judgments of the new type, unaided, after the fade. Then mix: yesterday’s packet rule, last week’s named file, one password-as-key, one page you leave.

Exit ticket (3–5 minutes, unaided). One hello named private or public, or one page with a maker-check said alone. Tool closed. A fluent generated “history of the Internet” is performance. The sentence they can still say is the test.

How to fade help. You send and say private → they say private while you send → they send (parent-held) while you watch → they judge unaided → they ask *you* who made a page. When they can attach people to public-or-private without your prompt, put the scaffold away. Bring it back when leaving-the-page or a citation hunt arrives.

When to stop talking. After one model and one question. You hold the key: is there a hello, a privacy sentence, a check that left the page, and an unaided try? You are not the search box.

Safety this week is accounts. Parent-held login. No child public profile as a five-to-ten default. Parent-visible browser. No surprise tabs. Under thirteen, the parent holds the key; scripts and parent-child talk, not live open chat. COPPA covers operators collecting personal information from children under thirteen. It does not make content safe. It does not apply to teenagers; thirteen-to-eighteen are still a household hour. Photos, video, and audio of a child are personal information when collected by a covered operator. Prefer local files. Geolocation off unless the parent chose it. One family computer is enough.[65]

[65] COPPA as in note 62. Homeschool is not school authorization under unfinalized 2024 NPRM

If two ages share the hour, the younger child's sitting is the hello and the password-as-key. The older child's extra is search-then-leave, public-or-private, or citation hunt, on a second sitting.

Age-band moves

Place by skill, not by birthday. A sixteen-year-old who thinks the browser is the internet is still in the first band of this chapter for the *system*. They may still put Computer Science on a transcript if the year's work includes the system, not if the year is only search. A ten-year-old who can already say a hello was private is ready for a simple leave-the-page, not for a malware catalogue.

Ages 5–10. We can reach other people. Password as a key. Packets at the older half, carried from chapter 3. Parent-held hello. No child public profile. Short sits. Then stop. The child says private or public. You wait. You do not start malware. You do not start a research seminar. You do not start their own email.

Ages 11–14. Protocols across networks and the Internet, in ordinary language: the hallway's rule, now with more than one hall. Public versus private as a sort. Search, then leave the page. One factual question the family already has in a book. Footprint as "what would stay if we walked away." Encryption remains the paper model from chapter 3 unless you chose to repeat it.

Ages 15–18. Scalability and reliability in kitchen language: more people, more machines, the system still has to put messages back; it can be slow; it can fail. Malware in ordinary language: software meant to harm, sneak, or steal; you get it by running something you should not, not by "the Wi-Fi thinking." Privacy of automated collection: a site that remembers you without a hello you chose. Citation hunt: open the sources a model named; find the ones that do not exist. An optional later course may treat systems and networks as one exam piece. That is beyond, not this week's floor. Browser fluency remains literacy.

Named try-it: Far Away, Talking

Time. 10 minutes. Ages 5–10; a reminder at any age.

Materials. A parent-held video call or a parent-sent message to a relative, or a paper "here / there."

---

ed-tech amendments. One family computer is enough; a second machine is optional convenience.

Safety / accounts. Parent holds the account. No child public profile. No surprise contacts.

The fun. A real hello.

The skill. Many computers, far away, talking. Not “open a browser” as the whole idea.

How to run it. Send or receive one hello. “Who did we reach? What had to travel?” Wait. Then: private or public? Ages 11–14 add: how many machines might have helped, even if we cannot see them — ordinary language, not a hidden log we guess.

Named try-it: Password as a Key

Time. 8 minutes.

Materials. A paper lock. A made-up strong password the parent stores. Not a real family password written on the table.

Safety / accounts. Parent stores the real ones. Child does not share, post, or type a real password into a chat.

The fun. The click of private.

The skill. What passwords are and why we use them. Framework cybersecurity at this level is usernames and passwords. Not hashing. Not a list-attack lab.

How to run it. Paper lock. A short nonsense phrase as the toy key. “Who is allowed to know it, and why?” Ages 15–18 may hear the Framework’s intersection example: a program can try a list of ordinary words quickly, so a dictionary word is a weak key. Still a talk, not a procedure for attacking anything.

Named try-it: Search Then Leave the Page

Time. 15 minutes. Ages 11–14; 15–18 as needed.

Materials. One factual question the family already has in a book. A search the parent runs. The book stays on the table.

Safety / accounts. Parent-visible browser. No surprise tabs. Parent holds the account.

The fun. Catching a page that looks sure and is thin.

The skill. Evaluate origin and relevance. One looking move: who made this, and can we leave and check. Not a research seminar. Not a second book on claims.

How to run it. Read the book's sentence first. Then search. Open one page. Ask the opening question. Leave. Open the book again. "What did you actually read? What did the book say?" If they want to stay because the page looks official, that is the miss. Ages 15–18 add: a second independent page, still leaving the first.

Figure 10: A closed laptop, a paper envelope, a named book on a table, an unbranded map edge. No readable browser. No children. No logos.

Debug / talk box

This is an illustration, not a reported family. A page is open, or a hello has been sent. The parent has not announced a definition of a URL.

Opening question

"Who made this page, and how would we check without staying on it?"

Not: "What is a URL?"

Follow-ups (pick three to five)

1. What did you actually read?
2. What would Grandma see if this were public?
3. Is a fluent answer the same as a page we opened?
4. If the model named a paper, can we find it?
5. Why not a dictionary word for a password?

How to wait

Ask. Count a slow three. Look at the book or the envelope if the silence is hard. Leaving a page to check can take a minute of work-time. You do not fill the silence with the maker's name.

What a stuck silence usually means

They think the browser *is* the internet. Or the question was a recitation item. Or they want to "just Google it." Or wait-time has been too short for years. Next move: close the page, open the book, ask the opening question again. Done enough: the child can say the internet is many computers talking, name one privacy habit, and (11–14+) leave a page to check a claim.

Practice that actually builds learning

Blocked, for a new move. The day you introduce far-away talking, only the hello. The day you introduce the key, only the paper lock. The day you introduce leaving the page, only one question the book already answers. Mixing too early makes the child hunt for a school move called "research."

Mixed, for when to use it. Later the same week, the hello reappears next to packet cards and a named file. Mixing is choosing: is this a file in a folder, a message in pieces, or a page with a maker?

Brief retrieval. Two minutes: private or public on a paper post; one maker-check said out loud; one password habit. Cover vocabulary boxes.

One incorrect example to diagnose. After a good check, show a card that says "I found it on the internet" with no page. Ask: "What did this person skip?" After a model-citation, ask whether a fluent list is a page you opened. The child names the miss.

Use this week's talk box. Same opening. Same wait. Same stuck-silence meanings.

Named try-it: Public or Private

Time. 15 minutes. Ages 11–14 and 15–18; a paper version can sit at 5–10.

Materials. Two paper posts: one meant for Grandma, one meant for a classroom. Optional: a third that looks private and could be copied.

Safety / accounts. No real post as the first try. No child's face uploaded to a public gallery.

The fun. Sorting.

The skill. Public versus private as a tradeoff. Not a scare lecture. Not a citizenship course that replaces the system.

How to run it. Sort the two posts. "What would Grandma see if this were public?" Wait. If they sort by how they *feel*, ask who could copy it. Ages 15–18 add automated collection: a site that remembers you without a hello you chose — ordinary language, not a legal memo.

Named try-it: Citation Hunt

Time. 20 minutes. Upper 11–14 and 15–18.

Materials. A named page already on the table (a book chapter, a printed article). A model asked, on the parent's account, for three sources on that idea, *after* the child has read the named page.

Safety / accounts. Parent account. Child does not live in the chat. Ages 5–10 do not do this try-it as live open chat.

The fun. Opening the ones that exist and finding the ones that do not.

The skill. A fluent citation can be fake. Demonstration dated to 2023 models in the study this book uses; do not treat those percentages as a 2026 rate. Not a reason to skip the named page. The page wins.

How to run it. Child reads the named page first. Parent asks the model for three sources. Child and parent try to open each one. Mark: exists / does not / exists but does not say that. "If the model named a paper, can we find it?" Then close the model. The named page stays.

Talk every sitting. At 5–10 a spoken private-or-public is done enough. At 11–14 a maker-check that left the page. At 15–18 a malware sentence in ordinary language, plus one citation that did or did not exist. You are not assigning a history-of-the-Internet essay.

A weekly shape: one new move, blocked; mixed retrieval of the hello; then a page you leave. The year is a map. This week is people, a key, a check.

If the hour was mostly search, write Digital Literacy in your own notes for that sitting. If the hour was many computers talking, packets across the Internet, public and private as a system tradeoff, write Computer Science. Both can happen in one week. They still have two names.

For the student

You are learning that the internet is many computers talking, not a single window. A browser is a tool that asks for a page and shows it. A page is made by someone. You can leave the page to check. A hello can be private or public. A password is a key you do not share.

"I found it on the internet" is a start. It is not the name of a source. The source is a page, a person, a book. You say which.

Tiny worked example

Hello to Grandma, parent-held, one-to-one. Private — unless someone copies it.

A page about plants. Who made it? A name on the page, or no name. How would we check without staying? Open the plant book on the table. Compare one sentence.

That is a check. Staying until the page feels sure is not a check.

Two tries

1. Send or receive one parent-held hello. Say whether it was private or public, and how you know.
2. With a grown-up, open one page about a question a book in the house already answers. Say who made the page — or that you cannot tell. Leave. Open the book. Say what matched and what did not.

Write nothing if writing fights the sitting. Talk the sentence. Your grown-up can remember it.

Explain it back

Tell someone at the table how the internet is many computers talking. Then say how a browser is not the same thing. Then say how you would check a page without staying on it. Then say why a password is a key.

Challenge

Some people say they found it on the internet and stop. Some people think closing the browser shuts the system. Some people trust a fluent list of papers a program wrote. Some people treat a year of searching as Computer Science. What would those four moves mean? How do you look at the *page* instead? How do you ask who made it, and how you would check without staying?

You are allowed to struggle. You may use an envelope, a book, a parent-held hello. You talk. If you get stuck, ask for a hint — not the maker's name already in the grown-up's mouth. Then try again.

A picture a program made of a website is a scene. It is interesting. It is not a visit.

If you are older, try this extra: say in ordinary words what malware is (software meant to harm, sneak, or steal) and one way people try not to run it — do not click a thing you did not ask for, do not download from a stranger's promise. Then, with your grown-up, ask a model for three sources on a page you already read, and see which ones exist. You still do not live in the chat. Your grown-up holds that key.

If it isn't clicking

Three diagnostics. Each one has a next move. None of them is a verdict on talent.

1. The child thinks the browser is the internet, or says "I found it on the internet" as a source, or cannot say whether a hello was private.

The system is not yet attached to people and pages. Next move: envelope first, hello second, question third. Cover the search box. "Was that for Grandma, or for anyone?" "I found it on the internet" is ordinary. Stay with one hello and one book. If this is still the bottleneck after several weeks of short daily checks, stay here. A citation hunt on top of a child who cannot name a maker will not hold. A human who will sit with a book and wait, not a tab that fills the silence, is a reasonable next step if you have tried the small-set work and the check still does not come.

2. The child recites a "how the Internet works" video, or a safety script, and cannot leave a page to check.

The cue is still a performance. Next move: book first, page second, leave third. Cover the caption. "Who made this page, and how would we check without staying on it?" After an attempt, one incorrect example ("this person stayed because it looked official — what did they skip?"). Slow down the "interesting" video until a sentence can be tied to the book. Go ahead once they can judge private-or-public and leave a page. A tutor is useful if recitation remains the default after a couple of weeks of daily page-and-book work *and* the hour has become a fight.

3. The child will not talk, or every sitting becomes "just Google it," and the silence after your question is a wall.

Talk is the product, and a search box is the mind at rest. Next move: shorter sits, one envelope, the opening question, wait a slow three twice. If silence is hard, look at the book. For "just Google it," search *after* the book, then leave. One page a week is enough. If the hour has become a fight after a stretch of daily short sits, a human who will wait, not a chatbot that answers the question, is the release valve.

When to slow down. Hellos still unjudged. Recitation still the default. Sits so long that checking never starts. A generated screenshot doing the work of a visit. A child account as the five-to-ten default. Those are brakes. "Not ready" because of age is the brake this book will not use.

When to go ahead. A hello is an easy private-or-public. The child can leave a page and open a book. A password is a key they do not share. Then productivity software as *use*, honestly titled, has somewhere to sit, and first programs have a table that already knows a system. Being "good at the internet" because a child can search is not a reason to skip the check, and not a reason to title the year Computer

Science.

When to get a human tutor. You have run the hello, or the book-then-page, or the wait after the question, for a stretch of daily sittings, and the same diagnosis is still in the room, and the hour has become a fight. A tutor is a release valve. Keep yourself as the person who can still hear a source-shrug and a browsing year in disguise.

Tools, including AI

Optional helpers for you, the adult. The full rules live in the computing-hour box in the front of this book. This chapter is a pointer, not a second policy paper.

The child talks after the hello first. You hold the account and the key. A tool may explain today's idea *to you* from a named page. If the model and the page disagree, the page wins. Extra isomorphic practice (new paper posts to sort, same public-or-private job) with the answers hidden is allowed. A labelled SCRIPT after the child tried, a hint after an attempt, a diagnosis of a check already done, talk-box questions you vet — those are allowed. Ages 5–10: scripts and parent-child talk, not live open chat.

Leave these out of the hour: a chatbot as the search engine; "write my bibliography"; an unsupervised tab during the try; a model treated as a witness to a page it did not open; a generated screenshot treated as a visit; a certificate of internet fluency; a detector grading the child's sentence. A fluent citation can be fake. Open the named page. The unaided check is the test.[66] If the tool produces the maker's name, the child is a spectator. The same crutch that can raise assisted practice and then fail a closed exam in mathematics is available whenever a fluent answer replaces the child's attempt.[67]

Do not let the model invent a visit, a network trace, or a benchmark. The oracle is the page you opened, the book on the table, or a named source you can actually

[66] Nicholas C. Soderstrom and Robert A. Bjork, "Learning Versus Performance: An Integrative Review," *Perspectives on Psychological Science* 10, no. 2 (2015): 176–199. James Prather et al., "The Widening Gap," ICER 2024 / arXiv 2405.17739: n = 21 CS1; 20 of 21 finished a working program; illusion of competence among struggling students. Not a homeschool internet trial. The sitting condition is the mechanism used here.

[67] Hamsa Bastani et al., "Generative AI without Guardrails Can Harm Learning," *PNAS* 122, no. 26 (2025): e2422633122. High-school mathematics: GPT Base +48 percent on assisted practice, −17 percent on an unaided exam. Not a computing RCT.

find. Do not police the child's sentence with a detector.[68] Hermes Agent is not Hermes 4. Grok is not Grok Bot. Copilot and ChatGPT are not required for a hello. An envelope, a book, and a parent-held account are tools too. Use them, then fade them.

What "done enough" looks like

Placement is by skill, not birthday. A publisher's "grade 8 internet" book is a scope, not a legal grade. The transcript, when you need one, says **Computer Science** for the system — many computers talking, packets, protocols, public and private as a computing tradeoff. It says **Digital Literacy** for browse-and-search. It does not say a brand. It does not say "the internet" as a credit that was only clicking. NCAA will not treat basic internet navigation as computer science. That is a gift to your honesty, not a reason to skip the citizen move.[69]

Checklist before moving on

- The child can say the internet is many computers talking, not a single window.
- One privacy habit in a sentence: password as a key, or a hello judged private or public.
- Ages 11–14+: they can leave a page to check a claim against a book or a second source.
- "I found it on the internet" has receded as a complete source.
- You can hear a browser-as-system, a fake citation, and a browsing year titled Computer Science, and you can ask a good question without taking the account.
- Ages 15–18, if you are there: malware in ordinary language; one sentence about collection you did not choose; optional citation hunt with at least one source opened or found missing.
- Sits can be short. A spoken sentence is enough at 5–10. You did not require a research paper.

If most of that list is true, go on to a real document, honestly titled Digital Literacy,

[68] Weixin Liang et al., "GPT Detectors Are Biased Against Non-Native English Writers," *Patterns* 4, no. 7 (2023): 100779. Average false-positive rate 61.22 percent on human TOEFL essays in that study. One sentence: do not police the child's sentence with a detector.

[69] NCAA HSRC 2026–27, as in note 58. Transcript titles in this book: Computer Science for the system; Digital Literacy / Technology Applications for browse-and-search; never a brand.

even if the birthday says otherwise — and keep Computer Science for the week that is a system or a program they can trace. If the birthday says "twelfth grade" and the page still cannot be left, stay. The next chapter is computer *use*: type, save, a picture, a few slides. It will not steal this chapter's credit name.

A path through computing is the promise. A percentile is not.

Chapter 5

Documents, typing, and slides

Figure 11: A kitchen-table still-life: a printed page of dummy text, a pencil, a paper folder tab. No readable office screen, no brand logos, no people.

Why this matters

This chapter is computer *use*. It is in the book because a family needs it. A child who cannot find a file they made yesterday, or who cannot put a sentence on a page a grandparent could read, is stuck in every later hour of this book. That is worth the struggle. It is not, for that reason, Computer Science.

Computer science is why and how computers work. Typing, a letter, a few slides, and a sheet that adds a column are how a person *uses* a computer to get something done. The *K–12 Computer Science Framework* draws that line on purpose. Computer literacy, on that map, is productivity software, Internet search, and a digital presentation. Those skills matter. They are not the study of computers and of the processes that run on them.[70] California's public-school standards say the same thing in kitchen language: computer science is not learning to type, and it is not learning to use word-processing, spreadsheet, or presentation software.[71] The 2026 CSTA standards say it again: using word processors, slide decks, or generative tools is not computer science. Those tools show up in computing and in every other subject too.[72] NCAA's high-school review chart is a third cut, from athletic eligibility rather than from a classroom map: usage of word processing, spreadsheets, and presentation software does not contribute to an approved computing course.[73]

---

[70] K–12 Computer Science Framework Steering Committee, *K–12 Computer Science Framework* (2016), 12–13, https://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf. Computer science as "the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society." Computer literacy, educational technology, digital citizenship, and information technology distinguished as *use*. "These aspects of computing are distinguished from computer science because they are focused on using computer technologies rather than understanding why they work and how to create those technologies." Access date for URLs in these notes: 1 September 2026.

[71] California Department of Education, *Computer Science Standards for California Public Schools, Kindergarten Through Grade Twelve* (adopted 6 September 2018), 15–16, https://www.cde.ca.gov/be/st/ss/documents/csstandards.pdf. "Computer science is not: learning how to type or use a mouse; learning to use word processing, spreadsheet, or presentation software…"

[72] Computer Science Teachers Association, *2026 CSTA PK–12 Computer Science Standards*, DOI 10.1145/3820482, 2, https://csteachers.org/wp-content/uploads/2026/07/2026-CSTA-PK%E2%80%9312-Computer-Science-Standards.pdf. Computer science "Is Not: using technology tools like word processors, slide decks, or generative AI tools."

[73] NCAA High School Review Committee, *Policies and Procedures (Staff) 2026–27*, 55 and 60, https://ncaaorg.s3.amazonaws.com/committees/ncaa/hsreview/HSRC_PoliciesAndProcedures_Staff.pdf. Content that does not contribute includes "Usage of common programs (e.g. word processing,

A stranger who reads **Computer Science** on a transcript and finds only letters and slides has been misled. Print an honest title: **Digital Literacy**, **Technology Applications**, or **Keyboarding**. Keyboarding is a skill title. It is not Computer Science with a quieter font. Never put a product name in the credit line. Never write Scratch I. Never write a year of slides as Computer Science I.

The win at ages 5–10 is the save / find / revise loop, not a speed score. Instruction can move typing speed; published speed norms are highly variable.[74] Teach short technique. Then get back to a real sentence. At 11–14 a spreadsheet can do work — a total that changes when one cell changes, a small chart of hours practiced. That is still a slice of data literacy, not Algorithms and Programming. At 15–18, word, sheet, and slides are assumed tools for communicating about computing and for working with data. They are still not the Computer Science course. The next chapter is.

This week you can learn how a saved, named file is the literacy win, and how to hear "we did Computer Science — we made slides" as a wrong transcript title, not a cute slip. Today the student can type or dictate one short piece, save it with a real name in a real folder, and open it again.

For the parent: understand it yourself

You do not need to be an office trainer. You do need to understand this week's idea well enough to hear "it's saved because I can still see it" as a wrong picture, and "slides are Computer Science" as a wrong credit, not as cute slips.

Everyday picture. A letter to Grandma. The child types or dictates three true sentences. They save the file with a name a stranger could read — `letter-to-grandma-september.txt` or the house's equivalent — in a folder

---

spreadsheet, gaming, generative artificial intelligence, etc.)" and "Usage of presentation software." NCAA High School Review Committee, *Policies and Procedures 2025–26*: software applications, spreadsheets, website construction, keyboarding, computer repair, and other tech-prep computer courses "will not be approved."

[74] Denise K. Donica, Peter Giroux, and Amber Faust, "Keyboarding Instruction: Comparison of Techniques for Improved Keyboarding Skills in Elementary Students," *Journal of Occupational Therapy, Schools, & Early Intervention* 11, no. 4 (2018). Quasi-experimental K–5 comparison in one district; net words-per-minute change greater where a taught keyboarding curriculum was used, significantly in grades 1, 3, 4, and 5. Authors: "Current keyboarding speed norms are difficult to determine." They summarize Freeman et al. 2005 as showing wide ranges, including fifth grade from 4.7 WPM to 70 WPM. Useful study, not a kitchen-table speed law.

they can find tomorrow. They close the program. They open the folder. They open the file. They change one sentence. They save again. That loop is the hour. Speed is not the hour. A generated "professional" layout is not the hour.

Precise picture. A *document* is a named file a person can find later. A *folder* is a place you chose, not a mystery the machine invented. *Save* is an action, not a feeling. *Revise* is changing the file you can still find. Keyboarding is technique so the keys stop being the whole bottleneck. A spreadsheet is a grid that can compute: change one number, watch a total move. Slides are a few pictures and words that show one idea. ISTE's student standards put this work under Creative Communicator: choose a tool, make something original or remixed, and present it.[75] A National Research Council fluency report, written for undergraduates and still useful as a picture, split *skills* (using today's applications) from *concepts* (how computers and networks work) and *capabilities* (deciding whether to trust a tool).[76] This chapter is the skills strand. Concepts live in the hardware, operating-system, network, internet, first-program, and AI-as-topic chapters. Mixing the strands on the transcript is how a year of Word becomes a fake Computer Science credit.

LibreOffice Writer, Calc, and Impress are enough. They are free, they live on the family computer, and they make ordinary documents, sheets, and slides. Google Workspace is a fit if the family already lives in shared documents and the parent holds the account. Microsoft 365 is a fit if the house already owns it. Pick one. The hour is the save / find / revise loop, not a shopping list of three suites. The name of the tool is ordinary language for a program. It is not a credit title.

Wrong answers you should be able to hear

1. "We did Computer Science. We made slides." A deck about volcanoes, or a newsletter, offered as the computing year.

What it usually means. Use and science collapsed. The Framework's own cited survey already found that a majority of students thought creating documents and

[75] International Society for Technology in Education, *ISTE Standards* (2024, version 4.02), section 1.6 Creative Communicator, https://cms-live-media.iste.org/ISTE_STANDARDS_2024_v02.pdf. Technology-for-learning standards, not computer-science standards; no grade bands.

[76] National Research Council, Committee on Information Technology Literacy, *Being Fluent with Information Technology* (Washington, DC: National Academy Press, 1999), ERIC ED440625, https://files.eric.ed.gov/fulltext/ED440625.pdf. Skills, concepts, and capabilities. Undergraduate-focused; dated tool lists; the three-strand split still useful. The report is not a K–12 curriculum.

searching the Internet *were* computer science.[77] That mix-up is old. It is still easy to make at a kitchen table.

What to say. "This is a real document. That is worth doing. The transcript title is Digital Literacy, or Keyboarding if this was a typing year. Computer Science is the chapter where we make a machine follow steps we can trace."

2. "I can still see it, so it's saved." Or: "It's in the cloud, so we're done." The file has no name. Tomorrow they cannot find it.

What it usually means. The screen was treated as the record. Closing the window was never practiced.

What to say. "Close it. Now open the folder and find it by the name you gave it. If Grandma sat down at this computer, what would she click?"

3. "We'll start programming when they type 30 words a minute." Or the opposite: a year that is only a typing-speed contest, with no real document.

What it usually means. Speed was upgraded to a gate, or speed was upgraded to the year. No opened map in this book requires a magic words-per-minute number before a first program. Instruction helps. A grade-by-grade speed table is not science.

What to say. "Eight minutes of technique, a few times a week, so the keys are not the whole fight. Then a real sentence. We do not wait for a score to start the next chapter."

4. "Just make it look professional." The child wants a model to design the letter. The parent is tempted, because the page looks thin.

What it usually means. Layout was upgraded to the skill. The sentences were not.

What to say. "Grandma needs three true sentences she can find tomorrow. Fancy comes later, if it comes at all. A generated layout is not the child's document."

5. "We should learn all three programs this year." Writer and Docs and Word, as if collecting buttons were literacy.

---

[77] *K–12 Computer Science Framework*, 12, citing a Google and Gallup 2015 survey as reported there: a majority of students believed creating documents and presentations (78 percent) and searching the Internet (57 percent) were computer science activities. Report as the Framework's own cited survey, not as a 2026 poll.

What it usually means. Coverage won. Depth — one named file in one place they can find — lost.

What to say. "One tool we actually use. Save, find, revise. The other buttons will wait."

Five-minute parent warm-up

Before the lesson, at the family computer, no student in the room:

1. Make a tiny file yourself. Name it. Put it in a real folder. Close everything. Open the folder. Open the file. Say out loud: "This is where it lives."
2. Write the sentence you will actually ask: "Where is the file, and what would Grandma see if she opened it?"
3. Write tonight's transcript title on a sticky note: Digital Literacy, or Keyboarding, or Technology Applications. Cross out Computer Science if your hand reached for it.

If those three are fluent for you, you can hear a missing save and a wrong credit. That is the job. The student still types the sentences.

How to teach it this week

The computing hour at the front of this book already taught the shape: warm-up, short model, student attempt, one good question, mixed practice, exit ticket. This chapter fills it with a document. Accounts and safety stay where they were: the parent holds the login if the work is in the cloud; photos of a child are personal information when a covered service collects them; ages 5–10 use scripts and parent-child work, not live open chat as the hour.[78]

Warm-up (3–5 minutes, unaided). Yesterday's file, if it exists: "Open the folder. Find it. Read the first sentence out loud." No new tool. If there is no yesterday's

[78] Children's Online Privacy Protection Rule, 90 Fed. Reg. 16918 (22 April 2025); general compliance date 22 April 2026. Federal Trade Commission staff, "Complying with COPPA: Frequently Asked Questions," https://www.ftc.gov/business-guidance/resources/complying-coppa-frequently-asked-questions. COPPA covers operators collecting personal information from children under 13; it "was not designed to protect children from viewing particular types of content"; it does not apply to teenagers. Photos, video, and audio of a child are personal information when collected by a covered operator. A homeschool is not a school authorization.

file yet, they name the folder you will use today.

Short model (5 minutes). You type or dictate two sentences on *your* page. Think aloud: "I am naming this `garden-notes-tuesday`. I am putting it in Documents, in a folder called School. I am saving. I am closing. I am opening the folder. Here it is. I change one word. I save again." Keep it short. You are showing the loop, not delivering a lecture on menus.

Student attempt. The first try-it below. Product on a sticky note: a named file they can find after they close the program. You wait. Struggle before rescue. You do not grab the keyboard.

One good question. After the attempt: "Where is the file, and what would Grandma see if she opened it?" Then wait. A slow three seconds is a minimum in ordinary classroom research; hunting a letter on a keyboard, or hunting a file in a folder, can honestly take longer — fifteen seconds, or a minute — before you help.[79] Look at the folder, not at the child, if the silence is hard.

Mixed practice. One old sentence to revise. One new save. One "close it and find it." If a younger child is dictating, they still choose the name and watch the save.

Exit ticket (5 minutes, unaided). Close everything. Open the folder. Open the file. Change one word. Save. Tell you the name and the folder. Tool closed means the helper that would write the letter stays closed. You are listening for a file they can find, not for a pretty font.

How to fade help. You save while they watch → they tell you the name while you click → they click, you only watch → they close and reopen unaided. When they can find yesterday's file without a tour of the desktop, the pointing finger goes away. Bring it back when a new kind of file arrives — a sheet, a slide, a PDF.

When to stop talking. After one model and one question. You hold the key: is there a named file in a named place they can open tomorrow? You are not the typist.

---

[79] Robert J. Stahl, "Using Think-Time and Wait-Time Skillfully in the Classroom," ERIC Digest ED370885 (1994), https://files.eric.ed.gov/fulltext/ED370885.pdf. General classroom, not a computing trial. Think-time as uninterrupted silence; typical pause after a question between 0.7 and 1.4 seconds; about three seconds as a minimal wait unless the teacher has sound reasons to reduce it. Student task-completion work-time may be 3–5 seconds, several (15, 20, 30, or 90) seconds, or two or more minutes. No computing-specific wait-time trial was opened for this book.

Age-band moves

Ages 5–10. Short sits. Dictate if the keys are still a fight; they still choose the name and watch the save. One real document this week is enough: a letter, a caption, a list of what they saw on a walk. Eight to ten minutes of keyboard technique a few times a week, on the real keyboard they will use, so the keys stop being the whole bottleneck. Then back to the sentence. A picture they already own can sit in a one-page document. Uploading a child's face to a public gallery is not this hour. Scratch and first programs can start without a speed score. This band's transcript, if you keep one, is Keyboarding or Digital Literacy for the typing-and-document work — never Computer Science for the slides.

Ages 11–14. They type their own sentences. A few slides that show one idea, with one picture the family owns. A small table that computes: five numbers that mean something (pages read, minutes practiced), a total that moves when one cell changes. That table is data literacy. It is not a programming course. They export or print one piece a stranger could open. Sharing a cloud file is a parent decision; the parent still holds the account.

Ages 15–18. Word, sheet, and slides are assumed tools. The work is a named paper or portfolio piece a stranger could find: a filename that says what it is, a folder a counselor could navigate, a PDF if that is how the house delivers work. Interactive charts and cleaner data work can live here as communication about computing. They still are not the Computer Science year. If this is the only computing the student does, title it Digital Literacy or Technology Applications and tell the truth in the description.

Exact wording you can steal

On finding: "Where is the file, and what would Grandma see if she opened it?"

On saving: "Close it. Now open the folder and find it by the name you gave it."

On the credit: "This is a document. The transcript says Digital Literacy. Computer Science is when we make a machine follow steps we can trace."

On technique: "Eight minutes on the home row. Then the letter. We are not collecting a speed score as the year."

On slides: "One idea. A few slides. One picture we already own. That is showing, not Computer Science."

On the sheet: “Change this one number. What happened to the total? That is the machine doing arithmetic you set up. It is still not a program you wrote.”

On the model: “You write the sentences. A helper may talk to me about menus after you have tried. It does not write Grandma’s letter.”

Try-it — One Real Document

Time. 20–30 minutes.

Materials. A word processor. LibreOffice Writer is enough. One named folder the child can see. Paper and a pencil if they draft first.

Safety / accounts. Parent-held account if the file lives in the cloud. Local files on the family computer are a clean default. No child’s face uploaded to a public page.

The fun of it. A letter or a caption a grandparent could actually read.

The skill it builds. Type or dictate, save, find, revise. Not speed. Not Computer Science.

How to run it. Agree the audience and the three sentences before the screen comes on. They type or dictate. They name the file in ordinary words. They save. They close. They find it. They change one sentence. They save again. You ask the opening question. You wait.

Practice that actually builds learning

A stack of typing-speed tests with no real document is assignment, not teaching. A year of themed slide decks with no save / find / revise loop is activity. Practice the loop. Then add one new kind of file when the loop holds.

Blocked, for a new move. The day they first close a file and find it again, two more short saves of the same kind are honest. The day a sheet first computes a total, two more tiny tables, same move. The day they name a file so a stranger could find it, one more export, same move.

Mixed, for when to use it. Later the same week, mix: revise yesterday’s letter, a short technique sit, one slide that shows one idea, one “is this Computer Science

or a document?" question. Choosing *when* to make a sheet instead of a paragraph is part of literacy.

Brief retrieval. Yesterday's filename; the folder; one sentence they can still say without opening it. A short oral at the start of the hour.

One incorrect example to diagnose. Show a beautiful one-page "report" a model wrote, with a filename like `Document1`, sitting on the desktop. Or a transcript line that says Computer Science beside a folder of slides. Ask what Grandma would find, and what a counselor would think the year was. Then an isomorphic task: three true sentences, named, saved, found, unaided.

Kitchen letters and birthday slides can motivate. They do not replace the unaided save.

Try-it — Keyboard as Technique, Short

Time. 8–10 minutes, ages 5–10, a few times a week. Older students who still hunt and peck can use the same sit, then stop.

Materials. The real keyboard they will use. A sentence that is actually theirs, already drafted on paper if that helps.

Safety / accounts. Posture as comfort — feet down if they reach, screen they can see, wrists not jammed. This book is not a physical-therapy protocol. If a child is in pain, stop and ask a person who treats hands.

The fun of it. A sentence that is theirs, coming out of their own fingers.

The skill it builds. Instruction moves speed. Published norms vary so widely that a grade-3 target printed as science would be a guess. Short, regular technique so the keyboard is not the whole bottleneck for one real document.

How to run it. Home-row rest if they will take it. One sentence. You do not hover over wrong letters for the whole eight minutes. You do not extend the sit until they cry. Then the letter from One Real Document, or tomorrow's caption. Technique is a servant of the document. The document is not a servant of a speed chart.

Try-it — Picture and a Few Slides

Time. 20 minutes.

Materials. One photo the family already owns. A slide tool — LibreOffice Impress is enough. Three cards on paper first: the idea, the picture, the one sentence.

Safety / accounts. No child's face uploaded to a public gallery. A family photo in a local file is different from a public post. Parent holds the account if the slides live in the cloud.

The fun of it. Showing one idea.

The skill it builds. Creative communication: choose a tool, make something, present it. Still not a Computer Science credit.

How to run it. One idea, not a ten-slide report. They place the picture. They write the sentence in their own words. They save with a name. They close. They find it. They show you, standing up if that helps, as if Grandma were in the chair. You ask what a stranger would see. Fancy transitions are not the skill.

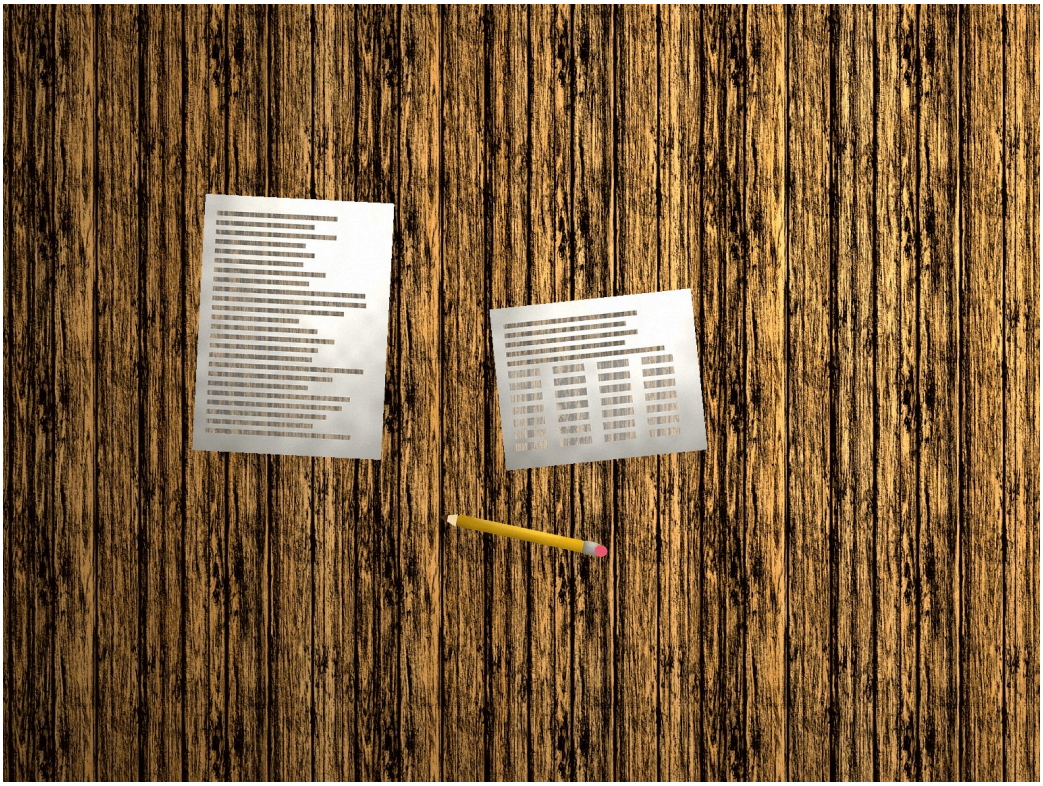

Figure 12: A printed dummy page beside a paper folder tab and a pencil, as if a document had left the screen and become an object. Dummy text only. No readable menus. No logos. No people.

Try-it — A Table That Computes

Time. 20 minutes. Ages 11–14, and 15–18 if a sheet is still new.

Materials. A spreadsheet. LibreOffice Calc is enough. Five numbers that mean something to the family: hours practiced, pages read, temperatures from the window, prices from a grocery list.

Safety / accounts. Same as the document: parent-held cloud, or local. No scraping of other people's data. No "hack the sheet." Five honest numbers.

The fun of it. The total changes when one cell changes.

The skill it builds. A slice of data literacy: a machine doing arithmetic you set up.

Still not Algorithms and Programming. A homeschooler who only lives in sheets has done digital literacy and a slice of data, not a Computer Science course.

How to run it. They type the five numbers. They put a sum in the sixth cell — you may show the sum once on *your* copy. They change one number. They predict the new total *before* they look. They look. They save with a name a stranger could read. Optional: a simple chart of those five numbers, if the chart earns its keep. Two numbers do not need a pie.

Try-it — Named So a Stranger Could Find It

Time. 15 minutes. Ages 15–18, and 11–14 when a portfolio piece exists.

Materials. A paper or a portfolio piece they already made. A way to export or print, including PDF if that is how the house hands work over.

Safety / accounts. The file is theirs. Metadata and a cloud share are parent decisions. Public links to a child's full name and photo are not a default.

The fun of it. "Someone else could open this."

The skill it builds. Files named; assumed tools for communicating about computing. Not worship of any one suite. Not NCAA Computer Science.

How to run it. Rename `Document1` and `Untitled` until the name says what the thing is. Put it in the folder a counselor could navigate. Export or print. Close everything. Hand the machine to you. You find it from the name alone. If you cannot, they rename again.

> **Debug / talk box**
>
> **Opening.** "Where is the file, and what would Grandma see if she opened it?"
>
> Not: "What is a word processor?"
>
> **Follow-ups** 1. What did you change since yesterday? 2. If we cannot find it, what words would you use to describe where it lives? 3. Is this Computer Science, or is this a document? (The honest answer is the document.) 4. What would a stranger read on a transcript that listed this year as Computer Science? 5. If we printed this page, would the sentences still be yours?

How to wait. After you ask, a slow three seconds is a minimum, not a cliff. File-finding and hunting a letter on the keyboard are task-completion work: fifteen, twenty, thirty, or ninety seconds, or a couple of minutes, can be honest before you help.[80] Look at the folder list, not at the child's face, if the silence is hard. Say, if you need a sentence: "Take your time. I'll wait." Then help. You do not grab the keyboard at the first pause.

Stuck silence usually means. The keyboard is the whole bottleneck (then a short technique sit, not a computing lecture); they think saving is optional because the text is still on the screen; they want a model to "make it look professional"; the question was a recitation item in disguise ("what is Save As?"); or they do not yet have the idea that a file is a named object in a place.

Next move. Point at the folder. Offer two choices of name. Repeat the close-and-open loop more slowly. A rehearsed frame — "name, folder, save, close, find" — not a lecture, and not your hands on the keys.

For the student

A computer can hold a page you made. That page is a *file*. A file has a name. It lives in a *folder* you chose. If you can close the program, open the folder, and find your page, you did the real work of this chapter.

This is not the chapter where you teach a machine a new trick. That is coming. This is the chapter where a person you love could sit down, open your file, and read sentences you actually wrote.

A tiny example. Imagine you want Grandma to know three things about the garden: the tomatoes are red, the basil was spicy, and a rabbit ate one leaf. You type those three sentences. You name the file `garden-for-grandma`. You put it in a

[80] Robert J. Stahl, "Using Think-Time and Wait-Time Skillfully in the Classroom," ERIC Digest ED370885 (1994), https://files.eric.ed.gov/fulltext/ED370885.pdf. General classroom, not a computing trial. Think-time as uninterrupted silence; typical pause after a question between 0.7 and 1.4 seconds; about three seconds as a minimal wait unless the teacher has sound reasons to reduce it. Student task-completion work-time may be 3–5 seconds, several (15, 20, 30, or 90) seconds, or two or more minutes. No computing-specific wait-time trial was opened for this book.

folder called Letters. You save. You close. You open Letters. There it is. You change “spicy” to “lemony” because you tasted again. You save again. Grandma, if she sat in your chair, would see three true sentences and one honest revision.

Try 1. Type or say three true sentences about something in your house. Save them with a name that says what they are. Close everything. Open the folder. Find the file. Read the first sentence out loud.

Try 2. Change one word so the sentence is still true. Save again. Tell someone where the file lives, in ordinary words — not “it’s in there,” but “Letters, then garden-for-grandma.”

Explain it back. In your own words: what is the difference between words you can still see on the screen and a file you can find tomorrow? What would Grandma see if she opened yours?

A challenge. Make a second kind of thing — a short list of numbers that add themselves, or two slides that show one idea with one picture your family already has. Name it. Save it. Close it. Find it. Then say, honestly: was that Computer Science, or was that a document? If you said document, you are right. The next chapter is the other kind of hour.

If it isn’t clicking

No shame in the room. Three common stuck places, and a next move for each.

1. They cannot find yesterday’s file. The desktop is a junk drawer. Every file is `Document1`. Or they are sure it is “in the computer” as a fog.

Next move. Slow the loop. One folder, one file, one name you write on a sticky note and put on the screen bezel. Close. Find. Repeat tomorrow with the same folder. If cloud and local are both in play, pick local for a week so there is one place. The idea is “a named object in a place,” not a tour of every menu.

2. The keyboard is the whole fight. Ten minutes in, they have four words, they are angry, and no one has saved.

Next move. Dictate the sentences. They still name and save. Add the eight-minute technique sit on other days, not as punishment and not as the whole year. If hands hurt, stop. A first program in the next chapter does not require a speed score. Start

that chapter when they can say what they expected a set of steps to do. Keep typing as a servant of real pages.

3. They want the page to look like a magazine, or they want a helper to write it. The sentences are thin. The layout is the whole conversation. Or a generated letter appeared and now they cannot tell you what it said.

Next move. Three true sentences, twelve-point text, no helper. Grandma is the audience. When the sentences are theirs and the file can be found, *then* one picture or one heading. A fluent page nobody in the house can explain is performance. The sentences they can read aloud are the skill.

When to slow down. If they cannot close a file and open it again, stay here. Another week of One Real Document is teaching. Birthday is not placement. A publisher's "grade 6 computer book" is a scope, not a legal grade.

When to go ahead. If they can find yesterday's file, revise one sentence, and tell you this is a document rather than Computer Science, you have enough of this chapter to start first programs. Keep using documents as tools. Stop treating them as the computing course.

When to get a human tutor. If a diagnosed difference with language, vision, or motor skills is making text entry the wall, a person who knows that work is the right help. An occupational therapist, a vision specialist, or a typing tutor is a person. A chatbot is not that person. Dictation, a larger keyboard, or a shorter sit can be honest accommodations. The transcript can still say Keyboarding or Digital Literacy while the description tells the truth about supports.

Tools, including AI

Keep this short. The computing-hour box at the front of the book already named what a helper may and may not do. This chapter points back. It does not rewrite the policy.

Pick one production tool and use it. LibreOffice is a fit when you want free Writer, Calc, and Impress on the family computer with no extra account. Google Workspace is a fit when the family already shares documents and the parent holds the login — including for anyone under 13. Microsoft 365 is a fit when the house already has it. One family computer is enough. You do not need all three suites. You do not need a new machine for typing.

You may, on your account, after the child has tried: ask a helper to explain *to you* a named menu or a named help page from the tool you actually use; make extra practice files with the answer key hidden (a second letter prompt, a second tiny table); draft a labelled SCRIPT for a save / find rehearsal you vet and then perform; give a hint after an attempt ("the name goes at the top of the window, then Save"); diagnose a file they already made ("this filename does not say what the paper is").

The helper does not write the letter. It does not invent Grandma's news. It does not "make it look professional" as a substitute for three true sentences. It does not sit as the child's only partner, especially at ages 5–10. Using a generative tool inside a document is still computer *use*. CSTA's line holds: using the tool is not computer science.[81] A fluent page is performance. The file they can find, and the sentences they can read aloud, are the skill.

Gemini-in-Docs, Copilot-in-Word, and any similar button are optional helpers on the parent's side of the table. They are not a reason to title the year Computer Science. They are not required.

What "done enough" looks like

Placement is by skill, not birthday.

Ages 5–10. They have one real document — typed or dictated — with a name and a folder. They can close it and find it the next day. They can change one sentence and save again. They have had some short technique sits on the real keyboard. They can say, in ordinary words, that this was a page they made, not a program they wrote.

Ages 11–14. The save / find / revise loop holds. They have shown one idea on a few slides with a picture the family owns, saved and found. They have made a small table that computes, and they can predict what happens when one number changes. They can tell you this year is Digital Literacy (or Technology Applications, or Keyboarding) rather than Computer Science.

[81] Computer Science Teachers Association, *2026 CSTA PK–12 Computer Science Standards*, DOI 10.1145/3820482, 2, https://csteachers.org/wp-content/uploads/2026/07/2026-CSTA-PK%E2%80%9312-Computer-Science-Standards.pdf. Computer science "Is Not: using technology tools like word processors, slide decks, or generative AI tools."

Ages 15–18. They can hand a stranger a named file — a paper, a sheet, a short deck — that opens and makes sense without you in the room. Filenames say what the work is. Word, sheet, and slides are tools they use on purpose. If this is the only computing work of the year, the transcript says **Digital Literacy** or **Technology Applications**, and the description names documents, sheets, and presentations. If they also traced programs, that other work has its own line: **Computer Science**. The two lines are not the same line.

Before you move on. One real document, saved, found the next day. An honest title. The next chapter exists because this one is not enough for the title of the book. First programs are Computer Science. This chapter was the use that makes later hours possible.

Chapter 6

First programs

Figure 13: A kitchen-table still-life: colored bricks as code-as-object, a notebook open to a blank trace table, a pencil. No brand logos, no readable programming screen, no people.

Why this matters

This is the chapter without which the title of the book is a lie. Computer science is why and how computers work. A path that never asks a child to make a machine follow steps they can trace is digital literacy with a borrowed name. Using a word processor, a slide deck, or a generative tool is not computer science. Programming is an essential part of computer science, and it is not the whole of it.[82]

A first program is a process written so a very literal follower — a machine, or a person playing machine — can run it without guessing. At ages 5–7 that can start as a recipe said out loud, then as a short sequence of blocks. At 8–11, events, loops, and a first conditional join the sequence. At 11–14, nested control and a procedure they can call twice. At 15–18, lists, modules, a prototype they can talk through with the screen off. None of those maps is homeschool law. None of them requires a named language. College Board's AP Computer Science Principles lets the teacher choose a language. AP Computer Science A requires Java. CSTA stays language-agnostic. NCAA names Java, C++, Python, and Scratch as examples inside an academic programming course.[83] Scratch is a tool. It is not a law. It is not a transcript title.

Unplugged work can show what computing is about. It is a catalyst when it is later run on a machine, not a year without a computer if a computer is available and wanted.[84] Blocks are a ramp, not a toy. In one high-school comparison, students

[82]Computer Science Teachers Association, *2026 CSTA PK–12 Computer Science Standards*, DOI 10.1145/3820482, 2. Computer science is not using word processors, slide decks, or generative AI tools; is not limited to coding; is not being replaced by AI. "CS is what powers AI." Programming includes reading and evaluating AI-generated code. Access date for URLs in these notes: 1 September 2026.

[83]Computer Science Teachers Association, *CSTA K–12 Computer Science Standards, Revised 2017*: Level 1A (ages 5–7) 1A-AP-08 (algorithms for daily processes), 1A-AP-10 (sequences and simple loops), 1A-AP-14 (debug sequences and simple loops); Level 1B (ages 8–11) 1B-AP-10 (sequences, events, loops, conditionals); Level 2 (ages 11–14) 2-AP-10 (flowcharts and/or pseudocode), 2-AP-12 (nested loops and compound conditionals), 2-AP-14 (procedures with parameters); Level 3A (ages 14–16, all students) prototypes, lists, modules, documentation; Level 3B (ages 16–18, specialty) data structures, recursion, languages comparison, AI algorithms. College Board, AP Computer Science Principles and AP Computer Science A course pages, opened 1 September 2026: CSP, teacher chooses the language; CSA, Java required. NCAA High School Review Committee, *Policies and Procedures (Staff) 2026–27*, 55 and 60: programming/coding examples include Java, C++, Python, Scratch.

[84]Tim Bell, "CS Unplugged or Coding Classes?," *Communications of the ACM*, 2021, https://cacm.acm.org/opinion/cs-unplugged-or-coding-classes/. "Unplugged can also be used to

who started in a block form of the same language learned more in five weeks than students who started in text; after ten further weeks in Java, that advantage faded.[85] This is a useful study, not a promise that every home will see the same result. A story or a game the child wants to finish can buy time on the task: middle-school girls using a storytelling version of Alice spent 42 percent more time programming, with equal learning of basic constructs.[86] Debugging is a named practice, not a personality test.[87] Computational thinking, in this book, names those practices — taking a process apart, writing it so a machine could run it, testing, debugging — not a product you must buy, and not a promise that programming will raise every other score.[88]

---

justify poor decisions by treating it as a complete curriculum in itself." "Unplugged never sought to replace computer programming." Catalyst finding reported there from Hermans and Aivaloglou (that study's PDF not opened for this book): no difference in understanding of programming concepts; unplugged-first group showed more self-efficacy and a wider vocabulary of commands. Bell's gloss: Unplugged as catalyst when combined with programming. "CS Unplugged is not a curriculum."

[85]David Weintrop and Uri Wilensky, "Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms," *ACM Transactions on Computing Education* 17, no. 3 (2017). Five-week quasi-experiment, two high-school introductory classes, same Pencil.cc environment; both improved (blocks $d = .96$; text $d = .50$); blocks showed greater learning gains and higher interest in future computing courses; text students viewed the experience as more similar to professional programming. Limitations the authors name: one language family, elective, two intact classes, not random assignment. Not ages 5–10; not Scratch versus Python; not a homeschool kitchen table. David Weintrop and Uri Wilensky, "Transitioning from Introductory Block-Based and Text-Based Environments to Professional Programming Languages in High School Computer Science Classrooms," *Computers & Education* 135 (2019): after ten weeks of Java, block/text differences faded, including attitudinal differences after fifteen weeks. Useful study, not a promise that every home will see the same result.

[86]Caitlin Kelleher, Randy Pausch, and Sara Kiesler, "Storytelling Alice Motivates Middle School Girls to Learn Computer Programming," CHI 2007. Storytelling Alice versus Generic Alice; equal success at learning basic programming constructs; Storytelling Alice users spent 42 percent more time programming. Middle-school girls; not a Python study; not a law of storytelling.

[87]Karen Brennan and Mitchel Resnick, "New Frameworks for Studying and Assessing the Development of Computational Thinking," AERA 2012. Concepts, practices, and perspectives from Scratch activity, not a mandate. Practices include experimenting and iterating, testing and debugging, reusing and remixing, abstracting and modularizing. Debugging developed "through trial and error, transfer from other activities, and the support of others."

[88]Jeannette M. Wing, "Computational Thinking," *Communications of the ACM* 49, no. 3 (2006); Jeannette M. Wing, "Computational Thinking and Thinking about Computing," *Philosophical Transactions of the Royal Society A* 366 (2008): essence of computational thinking is abstraction; it does not require a machine. Peter J. Denning, "Remaining Trouble Spots with Computational Thinking," *Communications of the ACM* 60, no. 6 (2017): an algorithm is "not any sequence of steps, but a series of steps that control some abstract machine or computational model without

The transcript title is **Computer Science**. The provider field may name Scratch, a Python course, or an unplugged-plus-machine year. The credit line does not.

This week you can learn how a loop is "do it again until," and how to hear a green run the child cannot trace as performance, not as the skill. Today the student can make a sequence of three steps a very literal person (or a machine) could run, then change one step and predict what happens before they run it.

For the parent: understand it yourself

You do not need to be a computer scientist. You do need to understand today's idea well enough to hear "the computer is thinking" as a wrong picture, and "it ran, so I understand" as a wrong test, not as cute slips.

Everyday picture. Making a peanut-butter sandwich. You say "put peanut butter on the bread." A very literal person puts the jar on the loaf, lid still on. You laugh, then you write the missing steps: open the jar, pick up the knife, scoop, spread. That is an algorithm in ordinary language: a sequence of steps a follower with no judgment can run. If a step is missing, the sandwich is wrong in a way you can point at. If you say "keep spreading until the bread is covered," you have a loop. If you say "if there is no peanut butter, use jam," you have a conditional. The test is not whether a kind adult could guess what you meant. The test is whether a machine, or a very literal person, could run it.[89]

Precise picture. A *program* is that kind of process, written in a form a computer can run. A *sequence* is steps in order. A *loop* is "do it again until." An *event* is "when this happens, do that" — a key, a click, a sprite touching a wall. A *conditional* is "if this is true, take this branch." A *procedure* (sometimes called a function, or a custom block) is a named chunk you can use more than once. A *bug* is a mismatch between what you expected and what happened. *Debugging* is the practice of finding that mismatch on purpose: say what you expected, look at what it did, try a change, look again.

Blocks postpone punctuation. That is a design choice, not a finding that punctuation is cruelty. Text is not a promotion and not a punishment. A fifteen-year-old

---

requiring human judgment"; the claim of universal value "has little empirical support." Teach computing because computing is worth knowing.

[89] Denning, "Remaining Trouble Spots." Everyday sandwich and toothbrush sequences in this chapter are illustrations, not evidence that any household chore is computer science.

who has never programmed can start in text. A child who is fluent in blocks should expect a transition into text, not a new identity. After a shared stretch of text, the intro modality is not the lasting difference.[90]

Unplugged hours license a start without a machine. They do not license a year that never runs. Bell, writing later about the Unplugged project he helped lead, is blunt: treating Unplugged as a complete curriculum is a poor decision; he wants students programming.[91] Grover and Pea, surveying the field, cautioned that unplugged activities may keep learners from the computational experiences computational thinking actually needs.[92] If a machine is on the table and wanted, plug the idea in so the child sees the same idea *run*.

Pair work at one keyboard can keep a stuck learner talking at 11–14 and 15–18 if you teach the roles. University completion rates from paired CS1 courses are university evidence. After-school game clubs with middle-school girls are a different evidence. This book does not mash them, and it has no opened pair-programming trial at ages 5–10. A parent sitting next to a child is help. It is not automatically

[90] David Weintrop and Uri Wilensky, "Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms," *ACM Transactions on Computing Education* 17, no. 3 (2017). Five-week quasi-experiment, two high-school introductory classes, same Pencil.cc environment; both improved (blocks $d = .96$; text $d = .50$); blocks showed greater learning gains and higher interest in future computing courses; text students viewed the experience as more similar to professional programming. Limitations the authors name: one language family, elective, two intact classes, not random assignment. Not ages 5–10; not Scratch versus Python; not a homeschool kitchen table. David Weintrop and Uri Wilensky, "Transitioning from Introductory Block-Based and Text-Based Environments to Professional Programming Languages in High School Computer Science Classrooms," *Computers & Education* 135 (2019): after ten weeks of Java, block/text differences faded, including attitudinal differences after fifteen weeks. Useful study, not a promise that every home will see the same result.

[91] Tim Bell, "CS Unplugged or Coding Classes?," *Communications of the ACM*, 2021, https://cacm.acm.org/opinion/cs-unplugged-or-coding-classes/. "Unplugged can also be used to justify poor decisions by treating it as a complete curriculum in itself." "Unplugged never sought to replace computer programming." Catalyst finding reported there from Hermans and Aivaloglou (that study's PDF not opened for this book): no difference in understanding of programming concepts; unplugged-first group showed more self-efficacy and a wider vocabulary of commands. Bell's gloss: Unplugged as catalyst when combined with programming. "CS Unplugged is not a curriculum."

[92] Shuchi Grover and Roy Pea, "Computational Thinking in K–12: A Review of the State of the Field," *Educational Researcher* 42, no. 1 (2013). By 2013 the K–12 field still did not agree on what computational thinking was, how to teach it, or how to assess it. On Unplugged: activities "may be keeping learners from the crucial computational experiences involved in CT's common practice."

"pair programming."[93]

Wrong answers you should be able to hear

1. "It ran. I'm done." A green flag, a sprite that moved, a function that printed. The child cannot say what the first step did.

What it usually means. Performance was taken for the skill. A working program can hide an illusion of competence.[94]

What to say. "What did you expect it to do, and what did it do instead? Trace the first step with the screen off."

2. "The computer is thinking." Or: "It knows what I meant."

What it usually means. A machine was given a mind. The missing step was treated as the computer's job.

What to say. "It only does the steps we wrote, in order. If we left a step out, it will not guess. Show me the step it actually ran."

3. "Any list of steps is an algorithm." Brushing teeth offered as computer science because the poster said Sequence.

What it usually means. The word *algorithm* was stretched until it meant "instructions for a person." Useful, reliable computation is narrower: a follower that does not get to use judgment.[95]

---

[93] Charlie McDowell, Linda Werner, Heather Bullock, and Julian Fernald, "The Effects of Pair-Programming on Performance in an Introductory Programming Course," SIGCSE 2002: university introductory course; paired students more likely to complete. Linda Werner and Jill Denning, "Pair Programming in Middle School: What Does It Look Like?," *Journal of Research on Technology in Education* 42, no. 1 (2009); Jill Denner and Linda Werner, "Computer Programming in Middle School: How Pairs Respond to Challenges," 2007: Girls Creating Games, 126 girls, after-school/summer, qualitative. Do not mash university CS1 with grade 3. No opened pair-programming trial at ages 5–10.

[94] James Prather et al., "The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers," ICER 2024, arXiv:2405.17739. n = 21 CS1 students, one C++ problem, Copilot and ChatGPT, 35 minutes; 20 of 21 finished a working program; new metacognitive difficulties (Progression, Interruption, Mislead); struggling students often finished with an illusion of competence. Small single-site lab, not a homeschool trial. A completed program is not evidence of learning.

[95] Denning, "Remaining Trouble Spots." Everyday sandwich and toothbrush sequences in this chapter are illustrations, not evidence that any household chore is computer science.

What to say. “Would a very literal person — or a machine — get this right without asking us what we meant? If they have to guess, we are not done writing.”

4. “Blocks are babyish.” Or: “Text at eight is cruel.” Or: “We have to finish Scratch before anything else.”

What it usually means. A ramp was turned into a caste system, or a tool was turned into a law. The high-school evidence is mixed on purpose: blocks helped early in that study; text students felt closer to what programmers do; the difference faded later.[96]

What to say. “Blocks are a way to see the ideas. Text is a way to see the same ideas with punctuation. We pick the ramp that lets you try today. Scratch is one tool. It is not the name of the year.”

5. “Just add a loop.” Or: “Ask the helper to write the function.” A recast from you, or a paste from a model, and now the sprite circles, and nobody can say why.

What it usually means. The next block was supplied before the child had a prediction. The helper became the programmer.

What to say. “Say what should happen before we run. If a tool suggested a line, explain that line. If you cannot, we put it back and try a smaller step.”

Five-minute parent warm-up

Before the lesson, no student in the room:

1. Write, on paper, three steps for a task in your kitchen. Hand the paper to an imaginary very literal person. Circle the step a machine would get wrong.

[96] David Weintrop and Uri Wilensky, “Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms,” *ACM Transactions on Computing Education* 17, no. 3 (2017). Five-week quasi-experiment, two high-school introductory classes, same Pencil.cc environment; both improved (blocks $d = .96$; text $d = .50$); blocks showed greater learning gains and higher interest in future computing courses; text students viewed the experience as more similar to professional programming. Limitations the authors name: one language family, elective, two intact classes, not random assignment. Not ages 5–10; not Scratch versus Python; not a homeschool kitchen table. David Weintrop and Uri Wilensky, “Transitioning from Introductory Block-Based and Text-Based Environments to Professional Programming Languages in High School Computer Science Classrooms,” *Computers & Education* 135 (2019): after ten weeks of Java, block/text differences faded, including attitudinal differences after fifteen weeks. Useful study, not a promise that every home will see the same result.

Add the missing step.

2. Write the sentence you will actually ask: “What did you expect it to do, and what did it do instead?”
3. Run, if you have a machine, one tiny loop in whatever environment you will offer this week — a clap game is enough if the machine is off. Say out loud: “Do it again until I say stop.” Put the helper that would write the function away.

If those three are fluent for you, you can hear a green run that cannot be traced. That is the job. The student still writes the steps.

How to teach it this week

The computing hour already taught the shape. Fill it with a process that could come out otherwise: the loop might not stop; the sandwich might get the lid; the sprite might miss the wall.

Warm-up (3–5 minutes, unaided). Yesterday’s three steps, said with the screen off. “What was the first thing the machine actually did?” No helper. No new language.

Short model (5–8 minutes). You run a sequence that is *wrong* on purpose on your copy. Think aloud: “I expected the sprite to stop at the wall. It went through. The first step it ran was move. I never said stop. I will add the stop, then I will predict, then I will run.” Keep it short. You are showing debugging, not delivering a lecture on syntax.

Student attempt. Recipe Algorithm, or Do It Again Until, depending on whether a loop is already in the room. Sticky-note product: a sequence they can trace, plus one prediction before a run. You wait. You do not grab the keyboard. You do not finish the function.

One good question. “What did you expect it to do, and what did it do instead?” Then wait. Debugging can honestly take a minute of silence.[97] Look at the script, the cards, or the notebook, not at the child, if the silence is hard.

[97] Robert J. Stahl, “Using Think-Time and Wait-Time Skillfully in the Classroom,” ERIC Digest ED370885 (1994). General classroom, not a computing trial. Task-completion work-time may be several seconds, ninety seconds, or two or more minutes.

Mixed practice. One old sequence to change. One new loop. One "predict before you run." One "is this a missing step, a loop that does not stop, or a different problem?"

Exit ticket (5–10 minutes, unaided). Screen off, or cards down. They say the first three steps. They say what would happen if you changed one. If they used a helper's line, they explain that line or it comes out. A program that runs is performance. A program they can trace is the skill.

How to fade help. You write a wrong sequence and they catch it → they write, you only ask the expectation question → they debug unaided, you still in the chair. When they can predict before a run, the pointing finger goes away. Bring it back when a new idea arrives — a procedure, a list, a first text landing.

When to stop talking. After one model and one question. You hold the key: did they say what they expected before they ran? You are not the second programmer.

Age-band moves

Ages 5–10. Unplugged first is enough to *start*: a sandwich, brushing teeth, a clap that repeats until you say stop. Then, if a machine is available and wanted, the same idea in a block environment. ScratchJr is a fit for the younger end; Scratch is a fit when they want a sprite that does something. No opened standard requires Scratch, Blockly, Python, or any named environment at this band.[98] Parent holds the account. Scripts and parent-child talk, not live open chat. Debugging at this age is describing the problem and getting help, not diagnosing a whole system.[99] Sequences, simple loops, then events and a first "if" when those are in reach. A story they care about is a reason to stay in the chair. Pair programming in the university sense is not this band's method; you sit beside them.

Ages 11–14. Nested loops and compound conditionals when the simple ones hold. A procedure they can call twice, with a parameter if they are ready — a chunk with a name, used in two places. Flowcharts or pseudocode on paper can plan a run. A story or a game as motive is honest: the personally meaningful *end* buys

[98] No opened CSTA, Framework, or California standard requires Scratch, Blockly, Python, or any named environment at ages 5–10. Mitchel Resnick et al., "Scratch: Programming for All," *Communications of the ACM* 52, no. 11 (2009): design paper; low floor / high ceiling / wide walls; digital fluency as designing, creating, and remixing. Scratch is context, not a mandate.

[99] *K–12 Computer Science Framework* (2016), Grade 2 troubleshooting: developmentally appropriate troubleshooting is describing the problem and getting help, not diagnosing a system.

time. Scratch is still a fit; it is still not the transcript title. A planned move into text can start here — the same idea, different clothes — without calling the blocks a waste. Two people at one machine can help if you teach driver and navigator and you still ask what they expected. Sharing a project online is a supervision choice, not a default hour.

Ages 15–18. The floor is a prototype they can talk through, lists, modules, short documentation in their own words. Text is an ordinary landing. Python through a taught course is a fit; the python.org tutorial is written for programmers new to Python, not for people new to programming, so parent load is high if the docs are the whole curriculum.[100] Java is required for AP Computer Science A, not for this chapter. Specialty work — classic algorithms, efficiency, recursion, version control — is beyond the floor, not a homework tax on every homeschool senior. Reading a short function they did not paste, after they attempted the problem, belongs here as a computing practice. A late starter is behind in time-on-task, not biologically locked out. Plan the band. Weintrop's high-school novices learned in five weeks in either modality.[101]

Exact wording you can steal

On expectation: "What did you expect it to do, and what did it do instead?"

On tracing: "What is the first step the machine actually ran? Say it with the screen off."

On prediction: "If we change this one input, what should happen? Say it *before*

[100] Python Software Foundation, Python documentation tutorial, python.org. The tutorial "is designed for programmers that are new to the Python language, not beginners who are new to programming."

[101] David Weintrop and Uri Wilensky, "Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms," *ACM Transactions on Computing Education* 17, no. 3 (2017). Five-week quasi-experiment, two high-school introductory classes, same Pencil.cc environment; both improved (blocks $d = .96$; text $d = .50$); blocks showed greater learning gains and higher interest in future computing courses; text students viewed the experience as more similar to professional programming. Limitations the authors name: one language family, elective, two intact classes, not random assignment. Not ages 5–10; not Scratch versus Python; not a homeschool kitchen table. David Weintrop and Uri Wilensky, "Transitioning from Introductory Block-Based and Text-Based Environments to Professional Programming Languages in High School Computer Science Classrooms," *Computers & Education* 135 (2019): after ten weeks of Java, block/text differences faded, including attitudinal differences after fifteen weeks. Useful study, not a promise that every home will see the same result.

we run."

On the literal follower: "Would a very literal person get this right without asking us what we meant?"

On loops: "Do it again until — until what? What makes it stop?"

On procedures: "Can we use this chunk twice, in two places, without copying every brick?"

On the helper: "Point at the named page that says what this word does. We are not asking anyone to write the function."

On the credit: "The year is Computer Science. The tool might be Scratch, or Python, or cards. The tool is not the title."

Try-it — Recipe Algorithm

Time. 10–15 minutes. Ages 5–10 to start; older novices can use it the first week.

Materials. Making a sandwich or brushing teeth as a real sequence, or cards, one step per card. A very literal follower — you, playing machine.

Safety / accounts. Food-safe if food. No unsupervised heat. Cards, not a stove, if a young child is cooking in name only.

The fun of it. Catching a missing step. Being allowed to be the annoying literal person.

The skill it builds. A process for a daily task, written so a follower without judgment can run it. Not any list of steps. Not a year of Unplugged.

How to run it. They write or say the steps. You follow them *exactly*. When the lid stays on the jar, wait. Let them see the miss. They insert the step. They run it again. Then, if a machine is available and wanted, they make a three-step sequence that *runs*. Same idea, now moving.

Practice that actually builds learning

A worksheet of vocabulary — sequence, loop, algorithm — with no run is assignment. A click-until-it-works hour with no prediction is activity. Practice a process they can trace. Then mix.

Blocked, for a new move. The day a loop first stops on purpose, two more tiny loops, same move. The day a procedure is first called twice, two more calls, same chunk. The day they first debug from an expectation, two more broken sequences you prepared on *your* copy.

Mixed, for when to use it. Later the same week: an unplugged repair, a loop on the machine, one "predict before run," one procedure, one "blocks then the same idea in text" if that landing has started.

Brief retrieval. Yesterday's first step; what made the loop stop; one bug they can still describe. A short oral, screen off.

One incorrect example to diagnose. A sprite that circles forever, or a function a helper wrote that prints the right number for a *different* problem. Ask what they expected. Ask whether they can trace it. Then an isomorphic task: three steps, one change, a prediction, a run, unaided.

A story or a game can motivate. It does not replace the unaided trace. Transfer to math or writing, if it happens, is a bonus to check, not the reason for the hour. Near transfer inside programming — a loop here, a loop on a new sprite — is the honest goal.[102]

Try-it — Do It Again Until

Time. 15 minutes.

Materials. A clap, a block environment, or both. A way to stop: a count, a key, a spoken "stop."

Safety / accounts. Parent-held account if online. Ages 5–10 not in live open chat. Claps do not need a login.

The fun of it. The loop that does not stop until you say.

[102] Roy D. Pea and D. Midian Kurland, "On the Cognitive Effects of Learning Computer Programming," Technical Report No. 9 (1984), ERIC ED249919: spontaneous far transfer was not in evidence. Ronny Scherer, Fazilat Siddiq, and Barbara Sánchez-Scherer, "Some Evidence on the Cognitive Benefits of Learning to Code," *Frontiers in Psychology* 12 (2021): learning to code had a strong effect on coding skills and a medium effect on other cognitive skills in the synthesis they report; transfer must be facilitated; effects tend to be lower for trials with active controls; "the transfer debate has not yet to be settled." Coefficients from that 2021 opinion paper summarizing syntheses; not a homeschool finding.

The skill it builds. Sequences and simple loops. Then run it on a machine if one is available. Unplugged as catalyst, not as a replacement.

How to run it. They clap until you say stop. They name the until. Then, on the machine or with cards, they make a follower repeat a move until a condition. They predict the stop *before* they run. If it never stops, that is the lesson, not a crisis. They add the until. They run again.

Try-it — What Did You Expect?

Time. 15 minutes. Every band, often.

Materials. A program that misbehaves — theirs, from this week. Notebook. Pencil.

Safety / accounts. You do not snatch the keyboard. The child stays the person at the keys. Parent still in the chair.

The fun of it. Being a detective. The bug is a mismatch, not a verdict on the child.

The skill it builds. Testing and debugging as a practice. At the early band, describing the problem. Not "never show the code." Not a promise that you will never help.

How to run it. Before anyone clicks, they say what they expected. They run. They look. They write one line: expected / instead. They change one thing. They predict. They run again. You ask the opening question. You wait. If they are stuck, offer two choices ("missing step, or loop that never stops?"), not a lecture, and not your hands.

Try-it — Story or Game as Motive

Time. 25–40 minutes. Ages 11–14; younger if a sprite they care about is already the point; older as a studio, not as AP Computer Science A.

Materials. A block environment. Scratch is a fit, not a law. ScratchJr for younger. Paper for a three-beat story: start, problem, end.

Safety / accounts. Parent-held account. Ages 5–10 not in live open chat. Sharing at 11–14 is a supervision choice. No child's full name and face as a public project default.

The fun of it. A sprite that does something they care about.

The skill it builds. A personally meaningful end buys time on the task, with equal construct learning in the study that measured it.[103] Not a law of storytelling. Not "Scratch I" on a transcript.

How to run it. They name the end before they open the gallery of other people's projects. They build the smallest version that moves. They debug from expectation. Remixing someone else's work is allowed when they can say what they changed. The year is still Computer Science. The brand stays off the credit line.

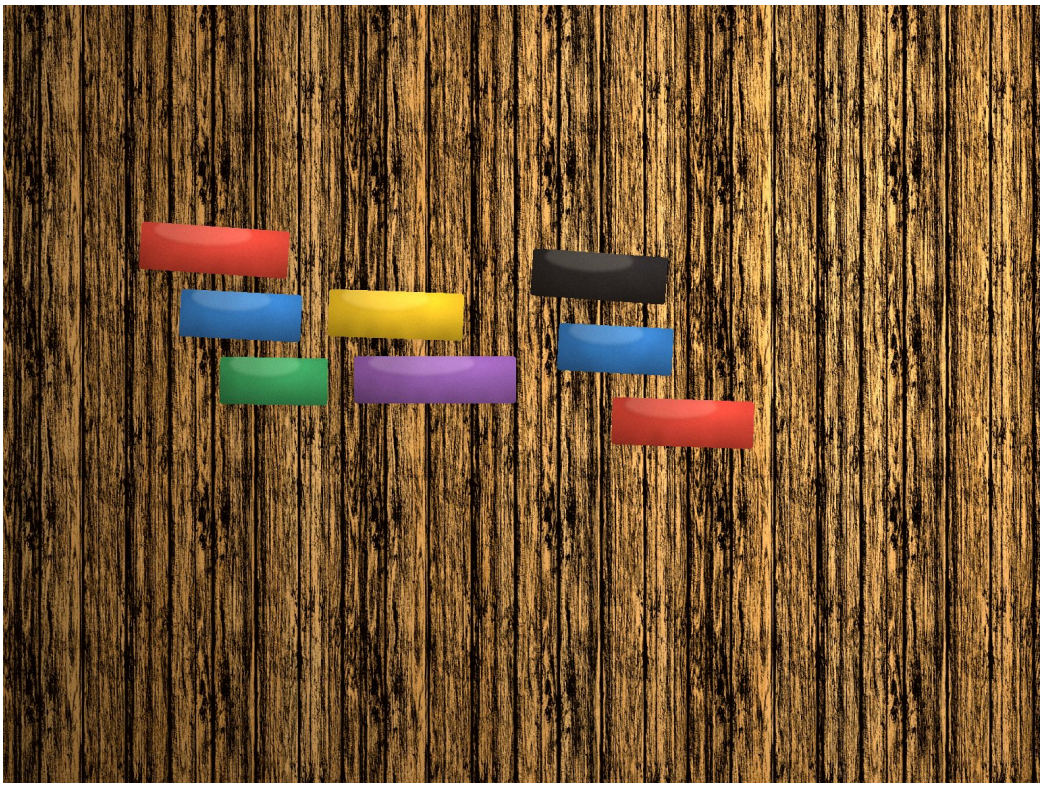

Figure 14: Colored bricks stacked as a process, a notebook trace beside them, a pencil. Code as an object on a table. No readable editor. No logos. No people.

[103] Caitlin Kelleher, Randy Pausch, and Sara Kiesler, "Storytelling Alice Motivates Middle School Girls to Learn Computer Programming," CHI 2007. Storytelling Alice versus Generic Alice; equal success at learning basic programming constructs; Storytelling Alice users spent 42 percent more time programming. Middle-school girls; not a Python study; not a law of storytelling.

Try-it — Procedure You Can Call Twice

Time. 20 minutes. Ages 11–14, and 15–18 if a named chunk is still new.

Materials. Blocks or text. A repeated action that is currently copied: jump twice, greet two sprites, add tax on two prices.

Safety / accounts. Same as the environment they already use. No new child account for this sit.

The fun of it. One chunk, two uses. Changing the chunk once and watching both uses change.

The skill it builds. Procedures with parameters, in ordinary language: a named bundle, handed a value, usable twice. Not AP Computer Science A Java. Not a data-structures course.

How to run it. They find the copied steps. They pull them into one named chunk. They call it twice. They change the chunk. They predict both calls. They run. If the two uses need different numbers, that number is the parameter — the blank the chunk is handed. Screen off: they say what the chunk does.

Try-it — Blocks Then a Text Landing

Time. A planned move, not one sit. Ages 11–14 and 15–18.

Materials. The introductory environment they used. Then a text language the curriculum names. A taught Python path is a fit. Docs-only Python is a steep parent load.

Safety / accounts. Parent holds new logins. Ages 5–10 are not the default audience for this landing.

The fun of it. "The same idea, different clothes."

The skill it builds. The intro modality is not the lasting difference after shared text.[104] Not "blocks were useless." Not "text at eight is cruelty."

[104] David Weintrop and Uri Wilensky, "Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms," *ACM Transactions on Computing Education* 17, no. 3 (2017). Five-week quasi-experiment, two high-school introductory classes, same Pencil.cc environment; both improved (blocks $d = .96$; text $d = .50$); blocks showed greater learning gains and higher interest in future computing courses; text students viewed the experience as more similar to professional programming. Limitations the authors name: one language family, elective, two

How to run it. Pick one idea they already can trace — a loop that stops, a procedure they can call twice. Rebuild it in text, small. They predict. They run. They debug from expectation. You do not rebuild the whole game in a weekend. You land the idea. Then another.

Debug / talk box

Opening. "What did you expect it to do, and what did it do instead?"

Not: "What is a loop?"

Follow-ups 1. What is the first step the machine actually ran? 2. If we change this one input, what should happen — say it *before* we run. 3. Can you trace it with the screen off? 4. Is this a missing step, a loop that does not stop, or a different problem? 5. If a tool suggested a line, can you explain that line?

How to wait. After you ask, a slow three is a minimum, not a cliff. Debugging is task-completion work: a minute of silence can be honest.[105] Look at the script, the cards, or the notebook, not at the child. Say, if you need a sentence: "Take your time. I'll wait." Then a hint. Then a short model on *your* copy. You do not snatch the keyboard at the first red mark.

Stuck silence usually means. Search for the next block or the next word (not refusal); they think a green run is understanding; a recast of "just add a loop" passed unnoticed; wait time has been too short for years; they want a model to write the function; or the question was a recitation item in disguise.

Next move. Point at the first step. Offer two choices. Repeat more slowly. A rehearsed frame — expected / instead / one change / predict / run — not a lecture, and not your hands on the keys.

---

intact classes, not random assignment. Not ages 5–10; not Scratch versus Python; not a homeschool kitchen table. David Weintrop and Uri Wilensky, "Transitioning from Introductory Block-Based and Text-Based Environments to Professional Programming Languages in High School Computer Science Classrooms," *Computers & Education* 135 (2019): after ten weeks of Java, block/text differences faded, including attitudinal differences after fifteen weeks. Useful study, not a promise that every home will see the same result.

[105] Robert J. Stahl, "Using Think-Time and Wait-Time Skillfully in the Classroom," ERIC Digest ED370885 (1994). General classroom, not a computing trial. Task-completion work-time may be several seconds, ninety seconds, or two or more minutes.

For the student

A program is a set of steps written so a machine can run them without guessing. You are allowed to be wrong on the first run. The work is to say what you expected, look at what happened, and change one thing.

Computers do not think. They do not know what you meant. They do the next step you wrote. If a step is missing, they will not fill it in from kindness. That is good news. It means a bug is a mismatch you can find, not a secret the machine is keeping from you.

A tiny example. You want a friend to take three steps: stand up, clap, sit down. You write those three. You play machine for them, or they play machine for you. If you forgot "stand up," the clap happens in the chair. You add the step. You run it again. On a machine, the same idea might be: when the green start is pressed, the cat moves ten, then says hello. If you expected hello *first*, and the cat moved first, you now know the order you actually wrote.

Try 1. Write three steps a very literal person could run. Have someone follow them exactly. Fix the missing step. Then change one step and say what will happen before you run it again.

Try 2. Make something repeat until a stop you name — a clap, a sprite moving, a number adding. If it never stops, that is a bug you can describe. Add the until. Predict. Run.

Explain it back. What did you expect, and what did it do instead? What was the first step the machine (or the literal person) actually did? Can you say it with the screen off?

A challenge. Make a chunk you can use twice — a jump, a greeting, a calculation — and use it in two places. Then, if you have been working in blocks and you are ready, rebuild just that chunk in text. Same idea, different clothes. If a helper offers to write it for you, ask instead for the named page that explains the word you are stuck on. You still write the chunk.

If it isn't clicking

No shame. Three stuck places, and a next move for each.

1. They cannot say what they expected. They click. It moves. They smile or they freeze. Asked to trace, they shrug. "It just did."

Next move. Take the machine away for one sit. Cards or a sandwich. You play the literal person. The mismatch becomes visible. Then a three-step program, screen on, with the prediction *written* before the run. If a helper has been writing the functions, it stays closed until they can trace what is already there. A working paste is not the child's knowledge of the algorithm.[106]

2. The loop never stops, or never starts, and they want you to fix it. Anger, or a quiet push of the keyboard toward you.

Next move. Two choices, spoken: "Does it need an until, or did we never tell it to begin?" Wait. If they cannot choose, you model on *your* copy: a clap that you forget to stop, then the until. They try the same move on theirs. You still do not grab their keys. If two people at one machine would help at 11–14 or 15–18, teach driver and navigator for one sitting, then return to one person at the keys.

3. Blocks feel like a toy, or text feels like a wall. Identity arrived before the idea.

Next move. Same idea, different clothes, on purpose. A loop they already can trace, rebuilt in the other form, small. The high-school study that compared blocks and text is a useful picture, not a verdict on this child.[107] A late starter at fifteen is a novice. Novices learn. They are not too late. If the project is boring, change the

[106] James Prather et al., "The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers," ICER 2024, arXiv:2405.17739. n = 21 CS1 students, one C++ problem, Copilot and ChatGPT, 35 minutes; 20 of 21 finished a working program; new metacognitive difficulties (Progression, Interruption, Mislead); struggling students often finished with an illusion of competence. Small single-site lab, not a homeschool trial. A completed program is not evidence of learning.

[107] David Weintrop and Uri Wilensky, "Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms," *ACM Transactions on Computing Education* 17, no. 3 (2017). Five-week quasi-experiment, two high-school introductory classes, same Pencil.cc environment; both improved (blocks $d = .96$; text $d = .50$); blocks showed greater learning gains and higher interest in future computing courses; text students viewed the experience as more similar to professional programming. Limitations the authors name: one language family, elective, two intact classes, not random assignment. Not ages 5–10; not Scratch versus Python; not a homeschool kitchen table. David Weintrop and Uri Wilensky, "Transitioning from Introductory Block-Based and Text-Based Environments to Professional Programming Languages in High School Computer Science Classrooms," *Computers & Education* 135 (2019): after ten weeks of Java, block/text differences faded, including attitudinal differences after fifteen weeks. Useful study, not a promise that every home will see the same result.

end — a story or a game they want — rather than changing the child's character.

When to slow down. If they cannot trace three steps with the screen off, stay with sequences and simple loops. Another week of Recipe Algorithm and What Did You Expect? is teaching. Unplugged-only is not a complete year if a machine is available and wanted; plug the same idea in, then stop.

When to go ahead. If they can trace a sequence or a loop unaided, describe one debug, and predict before a run, you have enough to add a procedure, or a first text landing, or the next system chapter. Birthday is not placement. A publisher's "grade 6 coding book" is a scope, not a legal grade.

When to get a human tutor. If language, reading, or a diagnosed difference is making text syntax the wall, a person who can sit at the same machine is the right help — a co-op, a tutor, a class with a human. A chatbot that writes the function is the wrong help. North Carolina's homeschool statute already names that other people may instruct; the homeschool still issues the transcript.[108] The title remains Computer Science. The description may tell the truth about supports.

Tools, including AI

Short. Point back to the computing-hour box.

Environments by fit, not by law. Cards and a notebook are enough to start. Scratch and ScratchJr are fits for a first sprite; they are not required and not a credit title. A taught Python path is a fit for a text landing. The python.org tutorial is a named language page, useful when you already know how to program and are new to Python. CS Unplugged is a fit for an idea without a login; then run the idea. Code.org courses are a tool, not the CSTA, and not Computer Science I on a transcript. One family computer is enough.

You may, on your account, after the child has tried: ask a helper to explain *to you* this week's idea from a named language documentation page or a named textbook page — if the helper and the page disagree, the page wins; make extra isomorphic traces with the answer key hidden (a new sprite, same loop); draft a labelled

[108] North Carolina Division of Non-Public Education, "Starting a Home School"; G.S. 115C-563(a) allows persons other than the parent to instruct. The homeschool still issues the transcript. Not a pair-programming finding.

SCRIPT for a debug rehearsal you vet; give a hint after an attempt; diagnose work already done; draft talk-box questions you vet.

For first programs, point at a named language page, not "write my function." The helper does not write the child's program. It does not invent output. It does not sit as the only partner, especially at ages 5–10. A completed program in a few minutes is not evidence of learning.[109] AP Computer Science Principles, if you ever sit it, treats generative help as supplementary: the student must review, understand, ensure it runs, cite it, and explain the code on the exam.[110] That mechanism is the kitchen rule too. If they cannot explain the line, they do not have the skill.

Copilot, ChatGPT, and any similar box are not required editors. They are optional, adult-side, after the attempt.

What "done enough" looks like

Placement by skill, not birthday.

Ages 5–10. A sequence of steps a very literal follower can run, plus a simple loop they can describe as "do it again until." One debug they can say in ordinary words: expected, instead, what they changed. If a machine is available and wanted, the same idea has run on it. They have not been asked to train a neural net. They have not been left in live open chat.

Ages 11–14. Nested control or a compound "if" when the simple ones hold. A procedure they can call twice. A story or game they wanted to finish, traced, not only displayed. Debugging as a habit: prediction before the run. A first text landing is optional and honest, not a promotion ceremony.

Ages 15–18. A prototype, a list, a named module or file they can talk through with the screen off. Text is ordinary if that is the path. They can read a short function

[109] James Prather et al., "The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers," ICER 2024, arXiv:2405.17739. n = 21 CS1 students, one C++ problem, Copilot and ChatGPT, 35 minutes; 20 of 21 finished a working program; new metacognitive difficulties (Progression, Interruption, Mislead); struggling students often finished with an illusion of competence. Small single-site lab, not a homeschool trial. A completed program is not evidence of learning.

[110] College Board, AP Computer Science Principles, AP Students page, opened 1 September 2026: generative AI as a supplementary resource; the student must review, understand, ensure functionality, cite, and explain the code on the exam.

they did not paste and say whether it solves *this* problem. Specialty algorithms and recursion are beyond, not a tax. If they started late, the floor is still a program they can trace, not an apology.

Before you move on. A sequence or a loop they can trace, unaided, plus one debug they can describe. The transcript line is **Computer Science**. The tool's name stays in the description. The next chapter is AI as a *topic* in that same science — pattern in, pattern out — not a substitute for the program they just traced, and not a new credit called AI Fluency.

Chapter 7

Artificial intelligence as a topic

Figure 15: A kitchen-table still-life: two piles of cards (red triangles in one, mixed shapes in the other), a notebook, a folded PARENT KEY. No chatbot screen, no tutor avatar, no people.

Why this matters

Artificial intelligence is a *topic* in computer science. It is not a substitute for computer science. It is not the whole book. Using a generative tool is not computer science. Computer science is what powers AI. Programming now includes reading and evaluating code a model emitted. A high-school specialty in AI, where a public-school map names one, sits *after* a foundational computing year, not instead of it.[111]

The adult picture this chapter uses is simple on purpose. A system that "learns" is doing statistical inference from examples people usually supplied. It does not think the way a human does. Perception, representation, learning, interaction, and impact are the five big ideas a joint AAAI and CSTA project put on a poster for teachers.[112] You do not need the poster on the fridge. You need the kitchen version: pattern in, pattern out; lopsided examples, lopsided guesses; a fluent paragraph can be wrong; a fluent function can solve a different problem.

Ages 5–10: not magic, not alive, not a friend. Sort a pile. Change the pile. See the guess change. Scripts and parent-child talk, not live open chat as the hour. The 2017 CSTA standards placed "describe how AI drives systems" and "implement an AI algorithm" at the high-school specialty level, not at kindergarten.[113] The 2026 standards distribute AI literacy earlier *and* keep a specialty. Quote both. Teach literacy. Leave the specialty for a student who already has a program they can trace.

Ages 11–14: training data and patterned mistakes. A fluent citation can be fake. A fluent function can solve a different problem.

---

[111] Computer Science Teachers Association, *2026 CSTA PK–12 Computer Science Standards*, DOI 10.1145/3820482, 2 and 4. Computer science is not using generative AI tools; CS is what powers AI; AI distributed across concepts; programming includes reading and evaluating AI-generated code; high-school specialty AIN after foundational CS; not a curriculum. Access date for URLs in these notes: 1 September 2026.

[112] AI4K12 / AAAI / CSTA, "Five Big Ideas in Artificial Intelligence," poster v.2 (2019), https://ai4k12.org/wp-content/uploads/AI4K12_Five_Big_Ideas_Poster_v2.pdf. Perception; Representation and Reasoning ("do not think the way a human does"); Learning (machine learning as statistical inference; training data usually supplied by people); Natural Interaction; Societal Impact. CC BY-NC-SA 4.0. Initiative homepage: https://ai4k12.org/.

[113] Computer Science Teachers Association, *CSTA K–12 Computer Science Standards, Revised 2017*, Level 3B: describe how artificial intelligence drives many software and physical systems; implement an algorithm for artificial intelligence at 3B, not at K–2.

Ages 15–18: a sort that always does the same steps is not the same object as a chat model that samples. Agents are not chatbots. Reading a short function they did not paste is a computing outcome. Building ChatGPT is not the homework.

The transcript title, when this work is real, is **Computer Science**. This book does not invent a kitchen-table badge called AI Fluency. NCAA will count AI *development, design, and refinement* inside an academic programming course submitted as math or science. Usage of generative tools will not.[114]

The tools box in this chapter points back to the computing hour. It does not reprint another book. Names stay un-mashed: Hermes 4 is not Hermes Agent. Grok is not Grok Bot. There is no Grok Bot education product on the pages this book opened.

This week you can learn how a model guesses from examples people chose, and how to hear a fluent paragraph as likely, not as true. Today the student can sort a pile, change the training pile, and say how the guess changed — then tell whether the guesser is a person.

For the parent: understand it yourself

You do not need to be a machine-learning engineer. You do need to understand today's idea well enough to hear "it knows" as a wrong picture, and "we did Computer Science — we chatted with a model" as a wrong credit, not as cute slips.

Everyday picture. A pile of cards. You show a child, or a very literal sorting rule, ten red triangles and say "triangle." You then hold up a blue triangle. If the rule was secretly "red," the blue one is sorted as "not." The guess followed the examples. The examples were lopsided. That is pattern in, pattern out. Nobody in the room has built a mind. You changed the pile. The next guess changed. That is the hour.

Precise picture. An *algorithm*, in the sense Chapter 6 already used, is a process a machine can run without judgment. A classical sort — always compare these two, always swap if out of order — does the same steps on the same input. A *model* trained on examples is different: it has fitted a pattern, and the next output is a guess. A chat model samples; the same prompt can yield a different paragraph. A

[114] NCAA High School Review Committee, *Policies and Procedures (Staff) 2026–27*, 55 and 60. Contributes: artificial intelligence (development, design, refinement). Does not contribute: usage of common programs including generative artificial intelligence.

chatbot is a conversational interface. An *agent* is a system that can take a sequence of actions on a computer — files, tools, a browser — not merely return a paragraph. Hermes Agent is a self-hosted agent. Hermes 4 is a chat and reasoning model family, and the people who make the agent say Hermes 4 is not the model to put *inside* the agent.[115] Grok is a conversational assistant. Grok Bot is an always-on cloud-computer agent. A "Game artist" job that makes pictures is pictures, not a computing course.[116] El Salvador's announced classroom program used Grok the assistant in public schools. That is not Grok Bot, and it is not a United States homeschool product.[117] Three objects, three jobs. Mashing the names is how a parent buys the wrong thing and calls it CS.

Machine learning, on the AI4K12 poster, is statistical inference. Training data is usually supplied by people. Agents do not think the way a human does.[118] Early grades notice attributes and that some programs make predictions. Students are not expected to understand supervised versus unsupervised learning in the first years, and independent training is out of scope there. Later, a student may train a tiny classifier with you in the room. CSTA's middle-grades language asks students to interpret and assess outputs that *assist*, not to build AI systems. High school: why the same prompt can yield a different paragraph; evaluate generated output; students are not expected to build or debug AI models as the floor.[119]

---

[115]Nous Research, Hermes Agent homepage, https://hermes-agent.org/, fetched 1 September 2026: self-hosted agent, February 2026, MIT license; not a chatbot; not a school product. Nous Portal docs, https://hermes-agent.nousresearch.com/docs/integrations/nous-portal: Hermes 4 is a chat/reasoning model family and is not recommended for use inside Hermes Agent.

[116]xAI, "Introducing Grok Bot," 11 August 2026, https://x.ai/news/introducing-grok-bot; "Grok Bot is now included with more plans," 26 August 2026, https://x.ai/news/grok-bot-more-plans; Grok Bot FAQ, https://docs.x.ai/grok-bot/faq. Always-on cloud-computer agent; jobs listed include a Game artist that makes pictures; no education product on pages opened 1 September 2026. Grok the assistant is a different object.

[117]xAI, "xAI and El Salvador Pioneer the World's First Nationwide AI Education Program," 11 December 2025, https://x.ai/news/el-salvador-partnership. Grok the assistant in a ministry program; announcement, not a trial; not Grok Bot; not a U.S. homeschool product.

[118]AI4K12 / AAAI / CSTA, "Five Big Ideas in Artificial Intelligence," poster v.2 (2019), https://ai4k12.org/wp-content/uploads/AI4K12_Five_Big_Ideas_Poster_v2.pdf. Perception; Representation and Reasoning ("do not think the way a human does"); Learning (machine learning as statistical inference; training data usually supplied by people); Natural Interaction; Societal Impact. CC BY-NC-SA 4.0. Initiative homepage: https://ai4k12.org/.

[119]CSTA and AI4K12, *AI Learning Priorities for All K–12 Students* (2025), https://csteachers.org/ai-priorities/. CSTA 2026 identifiers as opened for this book: early grades notice attributes and predictions; students not expected to understand supervised vs. unsupervised at the first elementary band; independent training out of scope at the next; a later

A fluent citation can be fabricated. In one study of short literature reviews, 55 percent of GPT-3.5 citations and 18 percent of GPT-4 citations were fabricated.[120] Detector scores are not a grade: seven detectors, in another study, flagged human TOEFL essays at an average false-positive rate around 61 percent.[121] Do not police the child's program or paragraph with a detector. If a child can paste a prompt and receive a fluent program, the same crutch that showed up in a high-school *math* trial is available — extra practice with a model that hands over answers, then a drop when the model is taken away — and a parent who is not a programmer will not always hear it.[122] That math trial is not a coding trial. The mechanism is the warning. Then we teach tracing.

Wrong answers you should be able to hear

1. "It knows." Or: "It's alive." Or: "It's my friend." A voice from a speaker, a paragraph that uses their name.

What it usually means. A tool was given a mind, a life, or a relationship. The examples disappeared from the story.

What to say. "What examples did we show it, and what did it guess about the new one? Is this a person, a toaster, or a pattern-completer?"

2. "We did Computer Science. We used ChatGPT." Or a certificate of AI

---

elementary band permits training a classifier. Middle grades: justify procedural vs. rule-based vs. data-driven; use an AI tool to generate outputs that assist, interpret and assess, not build AI systems. High school: deterministic vs. probabilistic; evaluate AI-generated output; students not expected to build or debug AI models as the floor.

[120]William H. Walters and Esther Isabelle Wilder, "Fabrication and Errors in the Bibliographic Citations Generated by ChatGPT," *Scientific Reports* 13 (2023): 14045. Eighty-four short literature reviews; 55 percent of GPT-3.5 citations and 18 percent of GPT-4 citations fabricated; of the real ones, 43 percent and 24 percent had substantive errors. 2023 models; this book does not update those rates to later models.

[121]Weixin Liang, Mert Yuksekgonul, Yining Mao, Eric Wu, and James Zou, "GPT Detectors Are Biased against Non-Native English Writers," *Patterns* 4, no. 7 (2023): 100779; preprint arXiv:2304.02819. Seven detectors; 91 human TOEFL essays; average false-positive rate 61.22 percent.

[122]Hamsa Bastani et al., "Generative AI without Guardrails Can Harm Learning: Evidence from High School Mathematics," *Proceedings of the National Academy of Sciences* 122, no. 26 (2025): e2422633122. Nearly 1,000 Turkish high-school *math* students. GPT Base: about +48 percent on practice, −17 percent on an unaided exam; GPT Tutor: about +127 percent on practice, statistically indistinguishable from control on the unaided exam; Base 51 percent correct. Math trial, not a coding trial. One-sentence crutch warning only.

Fluency printed from a website.

What it usually means. Use and science collapsed, the way slides collapsed in Chapter 5. CSTA's line is the same line: using the tool is not CS.[123]

What to say. "Using the tool is computer use. Computer Science is when we can say what examples it had, how a guess can be wrong, and — if we read a function — whether it solves this problem. There is no AI Fluency line on our transcript."

3. "The citation looks real, so it is real." A bibliography a model wrote. A function that runs for a different assignment.

What it usually means. Fluency was taken for truth, or a green run was taken for the right problem.

What to say. "Open the named page. If we cannot open it, it comes off. If we cannot explain the function with the screen off, it is not ours."

4. "The detector says they cheated." A percent. A red flag. A fight.

What it usually means. A tool that fails on human writing was upgraded to a judge.[124]

What to say. "We do not grade with a detector. We ask them to trace, to point at the examples, to explain the line. That is the test."

5. "Hermes, Grok, ChatGPT — it's all the same AI." Or: "Let's put them on a companion chatbot and call it the computing hour."

What it usually means. Product names collapsed, or a companion was upgraded to a course.

What to say. "A chatbot answers in language. An agent takes actions on a computer. Pictures are pictures. We name the object. Ages 5–10, we talk with a script. We do not park an unsupervised chat as the partner."

---

[123] Computer Science Teachers Association, *2026 CSTA PK–12 Computer Science Standards*, DOI 10.1145/3820482, 2 and 4. Computer science is not using generative AI tools; CS is what powers AI; AI distributed across concepts; programming includes reading and evaluating AI-generated code; high-school specialty AIN after foundational CS; not a curriculum. Access date for URLs in these notes: 1 September 2026.

[124] Weixin Liang, Mert Yuksekgonul, Yining Mao, Eric Wu, and James Zou, "GPT Detectors Are Biased against Non-Native English Writers," *Patterns* 4, no. 7 (2023): 100779; preprint arXiv:2304.02819. Seven detectors; 91 human TOEFL essays; average false-positive rate 61.22 percent.

Five-minute parent warm-up

Before the lesson, no student in the room:

1. Make two piles of paper: triangles and not-triangles, all red. Hold up a blue triangle. Say out loud what a lopsided pile will guess. Change the pile to include blue. Say how the guess should change.
2. Write the sentence you will actually ask: "What examples did we show it, and what did it guess about the new one?"
3. Write three names on a sticky note — chatbot, agent, pictures — and one line each. Fold a PARENT KEY over any helper you might use later. Ages 5–10: the helper stays a script in your mouth, not a live window.

If those three are fluent for you, you can hear "it knows." That is the job. The student still sorts the pile.

How to teach it this week

The computing hour already taught the shape, including the AI box. This chapter fills the hour with pattern in, pattern out. It does not reopen the box as a policy paper.

Warm-up (3–5 minutes, unaided). Yesterday's pile, or yesterday's sentence: "What examples did we show it?" No live chat. No new product.

Short model (5–8 minutes). You sort on *your* cards. Think aloud: "I trained only on red triangles. I expected the blue triangle to be 'not,' because red was the hidden pattern. I was wrong about triangles, because my examples were lopsided. I will add blue triangles and try again." Keep it short. You are showing a guess that can be wrong, not delivering a lecture on neural nets.

Student attempt. Sort Then Change the Pile, or Living vs a Tool if the pile already happened. Sticky-note product: a sentence in their own words — it guesses from examples, and it can be wrong. You wait.

One good question. "What examples did we show it, and what did it guess about the new one?" Then wait. Look at the piles, not at the child, if the silence is hard.[125]

[125] Robert J. Stahl, "Using Think-Time and Wait-Time Skillfully in the Classroom," ERIC Digest ED370885 (1994). General classroom, not a computing trial.

Mixed practice. One old pile. One changed pile. One "person, toaster, or pattern-completer?" One, at 11–14 or 15–18, "did this function solve *this* problem?"

Exit ticket (5–10 minutes, unaided). Cards down. "It guesses from examples we gave it, and it can be wrong." Older: name one difference between a sort that always does the same steps and a chat model that samples. Helper closed. A fluent paragraph about AI is performance. The pile they can explain is the skill.

How to fade help. You sort while they watch → they tell you which pile before you place the card → they sort, you only ask about examples → they change the pile unaided and say how the guess should move. When they can say "lopsided examples, lopsided guesses," the pointing finger goes away. Bring it back when a new object arrives — a function they did not paste, an agent versus a chatbot.

When to stop talking. After one model and one question. You hold the key: did they point at the examples? You are not the second guesser, and you are not the companion.

Age-band moves

Ages 5–10. Scripts and parent-child talk. Not live open chat. Not a child account on an open model. Unplugged piles, living versus a tool, a voice assistant the parent already owns and turns on for one minute if that helps the sort — then it goes off. They are not expected to train a network. They are not asked to build a chatbot. Evaluate when a tool is or is not a helpful resource as a *talk*, not as an unsupervised session.[126] COPPA covers operators collecting personal information from children under 13; it does not make content safe; it does not apply to teenagers. The parent still holds the account.[127]

Ages 11–14. A tiny classifier, two piles, parent-run. The child labels. Nobody creates a child account on an open chatbot for this sit. Talk about who is harmed if the pile is lopsided — a *talk*, with examples you chose, not a doom lecture. A fluent citation is checked against a named page. A short function they attempted,

[126] CSTA and AI4K12, *AI Learning Priorities*, grades 3–5: evaluate when AI is or is not a helpful resource — a talk, not an unsupervised session.

[127] Children's Online Privacy Protection Rule, 90 Fed. Reg. 16918 (22 April 2025); general compliance 22 April 2026. Federal Trade Commission staff, "Complying with COPPA: Frequently Asked Questions." Under 13; not designed to protect children from particular content; does not apply to teenagers. Photos, video, and audio as personal information when collected by a covered operator.

then a model-emitted version they read, is honest overlap with Chapter 6. Interpret and assess outputs that assist. They are not building an AI system this week.

Ages 15–18. Deterministic versus probabilistic in ordinary language: why the same prompt can yield a different paragraph. Evaluate generated output. Read a function they did not paste. Name three objects — agent, chatbot, pictures — from pages the parent opened, without installing the agent. NIST's risk-management framework is a voluntary adult picture, not kitchen law.[128] California's state guidance is one named state's advice, not a fifty-state mandate.[129] Stanford's college AI course forbids using an agentic coding tool to build a repository from a proposal; that is college policy, not a homeschool statute. Import the mechanism: if they cannot explain the code, they do not have the skill.[130] AP Computer Science Principles, if you sit it, treats generative help as supplementary and requires explanation on the exam.[131] Specialty AI coursework is beyond the floor.

Exact wording you can steal

On examples: "What examples did we show it, and what did it guess about the new one?"

On lopsided data: "If we had only shown orange cats, what happens to a black cat?"

On living versus a tool: "Is this a person, a search engine, or a pattern-completer?"

On a function: "Did this function solve *this* problem? Can you explain it with the screen off?"

On responsibility: "Who chose the examples? Who is responsible for the guess?"

[128] National Institute of Standards and Technology, *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*, NIST AI 100-1, January 2023, DOI 10.6028/NIST.AI.100-1. Voluntary, not kitchen-table law.

[129] California Department of Education AI guidance (letter 9 January 2026; last reviewed 25 June 2026 as opened). One named state. Education Code § 33308.5: guidance is not binding on local educational agencies in the sense that statute names. Not a fifty-state mandate and not homeschool law.

[130] Stanford CS221, Spring 2026, course collaboration policy as opened for this book: collaborator with a transcript; using an agentic coding tool to build a repository from a project proposal is not allowed. College course, not a homeschool statute.

[131] College Board, AP Computer Science Principles, AP Students page, opened 1 September 2026. Generative AI as a supplementary resource; student must review, understand, ensure functionality, cite, and explain the code on the exam.

On names: “A chatbot answers in language. An agent takes actions on a computer. Pictures are pictures.”

On the credit: “Using the tool is not Computer Science. Saying how the guess was made can be.”

On detectors: “We do not grade with a detector. We ask you to trace.”

Try-it — Sort Then Change the Pile

Time. 15 minutes. Ages 5–10; older as a first sitting if the idea is new.

Materials. Shape cards or toy animals. A “Triangles / Not Triangles” pile. Start with all one color.

Safety / accounts. Unplugged. No child chatbot. No live open window.

The fun of it. The guess that breaks when you only trained on red.

The skill it builds. Pattern in, pattern out. Lopsided examples, lopsided guesses. Not neural nets.

How to run it. They sort with you. You hold up a new shape. They guess. You reveal that the hidden pattern was color, or size, not the name on the pile. They change the pile. They guess again. They say, in their own words, what the examples were. You ask the opening question. You wait.

Practice that actually builds learning

A vocabulary quiz on “machine learning” with no pile is assignment. An hour of chatting with a model with no talk about examples is use, and it is not this chapter’s Computer Science. Practice the guess you can explain. Then mix.

Blocked, for a new move. The day a lopsided pile first produces a wrong guess, two more lopsided piles, same move. The day they first catch “it knows,” two more living-versus-tool sorts. The day they first reject a function they cannot explain, two more short reads, same move.

Mixed, for when to use it. Later the same week: an unplugged pile, a living-versus-tool sort, one citation check at 11–14, one function read at 15–18, one “name the object.”

Brief retrieval. Yesterday's examples; yesterday's wrong guess; one sentence — it can be wrong. A short oral, cards down.

One incorrect example to diagnose. A fluent paragraph with a fabricated citation, or a function that runs for a different problem, or a printout titled AI Fluency. Ask what examples were shown. Ask whether the named page opens. Then an isomorphic pile or an isomorphic function, unaided.

Kitchen voice assistants can motivate. They do not replace the unaided sentence. A companion chatbot is not a computing hour.

Try-it — Living vs a Tool

Time. 10 minutes. Ages 5–10.

Materials. A person in the room. A toaster or a hammer. A voice assistant the parent already owns, optional, parent-run for one minute.

Safety / accounts. Parent turns the assistant on and off. No child account. No companion session. No "talk to your friend the speaker" as the hour.

The fun of it. Sorting. Being allowed to say "not a person."

The skill it builds. Living versus nonliving versus a tool that talks. Not a friend. AI4K12's early picture, in ordinary language.

How to run it. Three cards: person, tool, pattern-completer. They sort the toaster, the person, the speaker. They say what the speaker does when the parent asks for a timer. They say whether it is alive. If they say friend, you wait, then: "What examples does it have? Who turned it on?" The speaker goes off. The talk stays with you.

Try-it — Tiny Classifier, Two Piles

Time. 25 minutes. Ages 11–14.

Materials. A parent-run demo, or cards if you would rather stay unplugged. Two label piles the child helps build: photos the family owns, or shapes, or words.

Safety / accounts. Parent holds the account. Child labels. Nobody creates a child account on an open chatbot. Photos of a child are personal information when a

covered operator collects them; family photos on a local machine, used for a sit and then put away, are a cleaner default than an upload.

The fun of it. The guess that changes when the pile changes.

The skill it builds. Training data and mistakes. A talk: if this pile is lopsided, who gets the wrong guess? Not "build ChatGPT."

How to run it. They label twenty examples. They try a new one. They change five labels or add five of the missing kind. They try again. They write one sentence about who would be harmed if the pile stayed lopsided — a local, concrete harm (the blue triangles always called "not"), not a national op-ed. You ask the opening question. You wait.

Figure 16: Two piles of cards on a table, a notebook with a blank trace line, a folded paper marked PARENT KEY. Dummy shapes, not a product screen. No logos. No people.

Try-it — Read a Function You Did Not Paste

Time. 20 minutes. Upper 11–14 and 15–18.

Materials. A short problem they already attempted in Chapter 6's sense. After they tried, a short model-emitted function on *your* account, labelled GENERATED, on paper or a file they did not type.

Safety / accounts. Parent runs the helper. The child does not paste the assignment into an unsupervised chat. Ages 5–10 skip this try-it.

The fun of it. Marking: runs / does not run; this problem / a different problem; I can explain every line / I cannot.

The skill it builds. Reading and evaluating AI-generated code. You do not grade with a detector.

How to run it. They attempted first. You print or show the generated function, labelled. They mark the three pairs. Screen off: they explain a line, or they say they cannot. If they cannot, the function does not become the solution. They go back to their attempt, or they point at a named language page for one word. The computing-hour rule holds: a fluent paste is performance.

Try-it — Name Three Objects

Time. 15 minutes. Ages 15–18.

Materials. Pages the parent already opened: an agent page, a chatbot page, a pictures-job page. Paper. Three columns.

Safety / accounts. The child does not install the agent. The child does not create a new cloud-computer. Parent-owned pages, perhaps printed. No product-sheet dump. No education SKU invented because a job list included "Game artist."

The fun of it. Catching the mash. Being the person who can say "that name is a different object."

The skill it builds. Agents versus chatbots versus pictures, in ordinary language. Not a product review. Not a shopping list.

How to run it. They write three rows: what it takes as input, what it does, whether it takes actions on a computer. They match Hermes Agent, Hermes 4, Grok, Grok Bot, and a pictures job to those rows using the pages in front of them — not from

memory of advertisements. If a name is missing from the pages you opened, it stays off the table. You ask who would be responsible if the agent moved a file. You wait.

Debug / talk box

Opening. "What examples did we show it, and what did it guess about the new one?"

Not: "What is machine learning?"

Follow-ups 1. If we had only shown orange cats, what happens to a black cat? 2. Is this a person, a search engine, or a pattern-completer? 3. Did this function solve *this* problem? 4. Can you explain it with the screen off? 5. Who is responsible for the guess?

How to wait. After you ask, a slow three is a minimum, not a cliff.[132] Look at the piles, the labelled GENERATED page, or the notebook, not at the child. Say, if you need a sentence: "Take your time. I'll wait." Then a hint. Then a short model on *your* cards.

Stuck silence usually means. They think the model *knows*; the question was a recitation item; they want to chat; they have never seen a lopsided pile, so there is nothing to hang the guess on; or a helper already wrote the paragraph about AI and there is nothing left to think.

Next move. Point at two cards. Offer two piles. Repeat the change-the-pile move more slowly. A rehearsed frame — examples / guess / what would change the guess — not a lecture, and not a live companion window.

For the student

A computer can guess from examples. The examples are usually chosen by people. If the examples are lopsided, the guess is lopsided. That is not magic. That is not a mind. That is not a friend.

This chapter is part of Computer Science because you are looking at how a kind of

[132] Robert J. Stahl, "Using Think-Time and Wait-Time Skillfully in the Classroom," ERIC Digest ED370885 (1994). General classroom, not a computing trial.

computing system works. Chatting with a model, by itself, is using a tool. Using a tool is useful. It is not the same thing.

A tiny example. You have a pile of red triangles labelled "triangle" and a pile of red squares labelled "not." Someone holds up a blue triangle. If the hidden pattern in your examples was *red*, you might put the blue triangle in "not." You were not mean. You followed the pile. When we add blue triangles to "triangle," the next blue one should land differently. The guess moved because the examples moved.

Try 1. Sort a pile. Write what the examples were. Guess about a new one. Then change the pile on purpose and say how the guess should change *before* you test it.

Try 2. Sort three things: a person, a toaster, and a talking speaker (or a picture of one). Which is living? Which is a tool? Which completes a pattern? If you want to say the speaker is a friend, stop and answer: who turned it on, and what examples does it have?

Explain it back. What examples did we show it, and what did it guess about the new one? Who chose the examples? What would you change in the pile to change the guess?

A challenge. If you have already tried to write a small program, read a short function you did not type — labelled GENERATED, after you tried — and mark: runs or not; this problem or a different one; I can explain every line or I cannot. If you cannot explain it, it is not yours yet. Point at a named page for one word, or go back to your own attempt. If you are older, name three objects from pages on the table: a chatbot, an agent, a pictures tool. Catch anyone in the room who mashes the names.

If it isn't clicking

No shame. Three stuck places, and a next move for each.

1. They think it knows. "Because it's AI." "Because it's smart." The pile never got a turn.

Next move. Cards only for a week. No speaker. No window. Change one pile until they can say the guess moved *because the examples moved*. Then, if a speaker is useful, one minute, parent-run, then off. The sentence we want in the mouth is "it

guesses from examples, and it can be wrong."

2. They want to chat. The hour becomes a conversation with a window. The cards sit untouched. Or a younger child asks to be alone with a companion.

Next move. The window stays closed at ages 5–10. You have a script. You are the partner. At 11–14 and 15–18, a helper may talk to *you* after they tried, on your account, from a named page. It does not sit as their only partner. If they wanted a friend, that is a different need than a computing hour. The computing hour is piles, traces, and names.

3. Fluency fooled the room. A beautiful explanation of neural nets they cannot connect to a pile. A bibliography that does not open. A function that runs for the wrong problem. A detector score that started a fight.

Next move. Named page or it comes off. Screen-off trace or the function is not theirs. Detectors stay unused as grades.[133] [134] Go back to two piles. Then one function. Small. Unaided.

When to slow down. If they cannot say that a guess comes from examples and can be wrong, stay with cards. Another week of Sort Then Change the Pile is teaching. Kindergarteners do not train neural nets in this book.

When to go ahead. If they can explain a lopsided guess, tell a speaker from a person, and — at the older bands — reject a function they cannot trace, you have enough of this topic to return to programs, systems, and an honest transcript. Specialty AI is beyond. Using the tool every day, if the house chooses, is still use. Title it as use.

When to get a human tutor. If you want a high-school specialty that actually designs or evaluates models, a person who can hear a wrong evaluation is the right help — a dual-enrollment course, a tutor, a class. A chatbot that praises

[133] William H. Walters and Esther Isabelle Wilder, "Fabrication and Errors in the Bibliographic Citations Generated by ChatGPT," *Scientific Reports* 13 (2023): 14045. Eighty-four short literature reviews; 55 percent of GPT-3.5 citations and 18 percent of GPT-4 citations fabricated; of the real ones, 43 percent and 24 percent had substantive errors. 2023 models; this book does not update those rates to later models.

[134] Weixin Liang, Mert Yuksekgonul, Yining Mao, Eric Wu, and James Zou, "GPT Detectors Are Biased against Non-Native English Writers," *Patterns* 4, no. 7 (2023): 100779; preprint arXiv:2304.02819. Seven detectors; 91 human TOEFL essays; average false-positive rate 61.22 percent.

fluent nonsense is the wrong help. College AI rules are college rules. Import the mechanism, not the syllabus as law.[135]

Tools, including AI

Short. This box points back to the computing hour. The child looks, tries, traces, and writes. The tool does not write the program, invent a trace, or sit as the only partner.

You may, on your account, after the child has tried: ask a helper to explain *to you* this week's idea from a named page (the AI4K12 poster, a CSTA sentence, a language doc); make extra isomorphic piles or extra short functions to read, with the answer key hidden; draft a labelled SCRIPT for a parent-child talk you vet and then perform — the default at ages 5–10; give a hint after an attempt; diagnose work already done; draft talk-box questions you vet.

You may not: park an unsupervised chatbot as the child's only partner (especially ages 5–10); ask a model to write the child's program or homework; invent a native-expert identity you cannot check; grade code or writing from an unopened detector; issue a certificate of competence or fluency; invent data, traces, or benchmark numbers.

Hermes 4 is not Hermes Agent. Grok is not Grok Bot and is not a classroom program in another country. There is no Grok Bot education product in this book's opened pages. Game artist makes pictures, not a CS course. Copilot and ChatGPT are not required editors. One family computer is enough. Raspberry Pi is optional hardware, not a moral upgrade.

Using the helper is not the Computer Science. The pile they can explain, the function they can trace, and the names they can keep un-mashed are.

What "done enough" looks like

Placement by skill, not birthday.

[135] Stanford CS221, Spring 2026, course collaboration policy as opened for this book: collaborator with a transcript; using an agentic coding tool to build a repository from a project proposal is not allowed. College course, not a homeschool statute.

Ages 5–10. “It guesses from examples we gave it, and it can be wrong.” They can sort living versus a tool versus a person. They have not been left in live open chat. They have not been asked to train a neural net.

Ages 11–14. They can change a training pile and say how the guess should move. They can talk, in a local example, about a lopsided pile and a wrong guess. They can check a fluent citation against a named page. They can mark a short generated function as this problem or a different one.

Ages 15–18. They can name a difference between a sort that always does the same steps and a chat model that samples. They can evaluate one generated output: runs / explains / I would not use it for this. They can keep agent, chatbot, and pictures from collapsing into one word. They do not need a specialty AI year for this to be enough. They do not get a transcript line called AI Fluency.

Before you move on. The opening question has an answer in their mouth that points at examples. The tools box stayed a pointer back to the hour. The next chapter is how you sequence a path and how you name it so a stranger is not misled. First programs remain Computer Science. This topic, taught as systems, can live on that same line. A year of chat cannot.

Chapter 8

Choose and sequence

Figure 17: A kitchen-table still-life: four year-boxes on paper labelled with dummy titles Digital Literacy, Computer Science, AP CSP, AP CSA; a folded talk-box slip. No brand logos, no portraits, no readable product screens.

Why this matters

A parent choosing a computing path does not need a federal flowchart. They need a calendar and an honest transcript. Start with a device they can name, a file they can find, and a recipe-algorithm they can trace. Stay with systems plus a first program long enough that a stranger who reads **Computer Science** finds algorithms, programs, data, and impacts — not only typing. Keep productivity in the year as computer *use* with an honest title. Optional exams, if you sit them, sit *beyond* that floor. Then teach this week's try-it.

If the child is six, short sits and objects. If the child is fourteen with no prior hour, they are a novice. High-school novices in one comparison learned in five weeks in either blocks or text.[136] They are not too late. Age is time-on-task and prior knowledge, not a native-coder window. A late start is a planning fact. It is not a closed biological boat.

A sequence that runs Digital Literacy, then Computer Science, then an optional AP Computer Science Principles year is a college-ready *construction*. It is not a description of what most homeschoolers had already done when the federal survey last asked. Computer science that year — listed under Science as "computer science (e.g., computer programming)" — was taught in 16 percent of homeschool households overall, and in 17 percent of grades 9–12. Among grades 9–12, 37 percent had ever been taught it during home instruction.[137] That is context, not a scolding. You are one of those tables. There is no national computer-science minutes number in these pages. There is no national diploma. None of Texas, North Carolina, Pennsylvania, Virginia, or New York requires computer science or tech-

[136] David Weintrop and Uri Wilensky, "Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms," *ACM Transactions on Computing Education* 17, no. 3 (2017). High-school novices in a five-week introductory comparison learned in either modality. Useful study, not a kitchen stopwatch and not a native-coder claim. Access date for URLs in these notes: 1 September 2026.

[137] Jiashan Cui and Rachel Hanson, *Homeschooling in the United States: Results from the 2012 and 2016 Parent and Family Involvement Survey*, NCES 2020-001 (Washington, DC: National Center for Education Statistics, 2019), Tables 8 and 9, https://nces.ed.gov/pubs2020/2020001.pdf. Table 9, subjects taught *that year*: computer science (e.g., computer programming), listed under Science, 16 percent overall; 11 percent (interpret with NCES's caution flag) in K–2, 18 percent in 3–5, 17 percent in 6–8, 17 percent in 9–12. Table 8, grades 9–12 *ever* taught during home instruction: computer science 37 percent. The 2023 First Look (NCES 2024-113) did not republish subject-taught tables. Ellena Sempeles and Jiashan Cui, *Parent and Family Involvement in Education: 2023*, NCES 2024-113: 3.4 percent homeschooled, approximately 1,765,000 students ages 5–17 with a K–12 grade equivalent, 2022–23.

nology for homeschool.[138] Lawful without a CS year is legality. A college-ready computing path is a different stack. You can choose the stack. You cannot pretend a year of slides was the stack.

AP Computer Science Principles and AP Computer Science A are possible beyonds. They are not the floor. Students may take them in either order. This chapter will not sell them. Notes at the back name what the opened course pages said.

This week you can hold two columns — use and CS — and see that a grade-9 start that holds through 12 is a path. Today the student can say last week's unaided line (a named part, a saved file, or a traced loop) without the book, then help write next year's transcript title in ordinary English.

For the parent: understand it yourself

You do not need to be a counselor. You do need to understand this week's idea well enough to hear "Scratch I" as a wrong credit title, and "colleges all require CS" as a claim this book will not make, not as cute slips.

Everyday picture. Four boxes on a piece of paper. Box one: Digital Literacy — files, keyboard, a document a stranger could find. Box two: Computer Science — hardware as a system, an operating system as software, a network as packets, a first program they can trace, AI as pattern in and pattern out. Box three, optional: AP Computer Science Principles. Box four, optional: AP Computer Science A. You are allowed to draw only two boxes. You are allowed to draw the first two twice, because depth is the hour. You are not allowed to label box one Computer Science because the slides looked busy.

[138] Texas Education Agency, "Home Schooling" and "Alternative Schooling": *Leeper* list is reading, spelling, grammar, mathematics, and good citizenship — computer science is not named; the State does not award a homeschool diploma. North Carolina DNPE: annual test in English grammar, reading, spelling, and mathematics — not CS; five-hour day is a recommendation. Pennsylvania 24 P.S. § 13-1327.1: elementary and secondary lists do not name CS; graduation 9–12 is four English, three math, three science, three social studies, two arts and humanities. Virginia Department of Education, "Home Instruction": parent writes a list of subjects; CS is not required. Virginia diploma seals page: no Computer Science seal; STEM Seal is a public-diploma object. New York 8 NYCRR 100.10: grades 1–6 do not name CS; grades 9–12 have three units of electives where a CS year may sit. This chapter is not legal advice. Read your current department page.

Precise picture. Public maps distinguish a floor for all students from specialty and from college-board exams. CSTA's 2017 levels 1A through 3A are for all students; 3B is specialty. The *K–12 Computer Science Framework* is a high-level map, not standards, not a curriculum, not national law, and not a homeschool mandate. AP and career-technical courses sit outside that floor.[139] Your kitchen table is not required to be a public-school map. Using the map as a *picture* of sequence is honest: look and save first; programs and systems next; specialty and AP later if they are wanted.

University of California A–G has **no computer-science subject area**. Computer science and computer programming are named among college-preparatory electives in area **G** (one year required of electives). Area **D**: a CS course can be used as an *additional* science — third year and beyond — not as a substitute for the two years that must cover two of biology, chemistry, and physics. Area **C**: a CS course with substantial mathematical content (induction, proof techniques, or other discrete mathematics) could be approved; a course whose primary focus is coding methods alone would not.[140] How a California homeschooler gets a course onto an A–G list is a local question this book does not pretend to have closed. UC A–G is a California public-university admissions map, also used by CSU. It is not Texas *Leeper*, not North Carolina's test list, and not a homeschool statute.

NCAA Division I asks for sixteen core courses. Computer science is not named as an additional discipline the way world language is. A student can meet Division I without a CS year.[141] Academic programming *may* count as math or science if

[139] *K–12 Computer Science Framework* (2016), 14 and FAQ at https://k12cs.org/faq/: concepts and practices are not specific measurable performance expectations and not curriculum; "The framework statements are not standards." CSTA 2017: Levels 1A–3A for all students; 3B specialty. CSTA 2026: not a curriculum. AP and CTE sit outside the Framework floor as this book uses that map.

[140] University of California Office of the President, A–G Policy Resource Guide, Subject Area G (college-preparatory elective; computer science and computer programming named), Subject Area D (science; CS as additional science, third year and beyond), Subject Area C (mathematics; substantial mathematical content required; "a course with primary focus on coding methods alone would not qualify"), opened 1 September 2026, https://hs-articulation.ucop.edu/guide/. No computer-science subject area. Homeschool pathway onto an A–G list not closed in this book's sources.

[141] NCAA Eligibility Center, Division I Academic Requirements fact sheet, September 2023, still live 1 September 2026, http://fs.ncaa.org/Docs/eligibility_center/Student_Resources/DI_ReqsFactSheet.pdf. Sixteen core courses: 4 English, 3 math (Algebra I or higher), 2 science including one year of lab if offered, 1 extra English/math/science, 2 social science, 4 additional from listed areas *or*

it qualifies for graduation credit in that area and it is an academic programming course. Keyboarding, office applications, and usage of generative tools will not be approved.[142] The sample homeschool transcript in the toolkit has no Computer Science line. That absence is a fact about the sample, not a prohibition. If you want NCAA-readable CS, you add a title that matches the worksheet, in math or science. There is no Computer Science CLEP. The nearest exam is Information Systems, listed under Business — not a high-school CS year, not NCAA core.[143]

Google CS First turned down on 30 June 2025. It is not a live course to sequence into.[144] One family computer is enough. A helper is not a reason to skip the first program.

Wrong answers you should be able to hear

1. "This year was Computer Science. We typed and made slides, and we asked a model for help." The folder is documents. There is no traced program.

What it usually means. Use and science collapsed. Chapter 5 already named this miss. It still happens at planning time.

What to say. "If a stranger read next year's transcript, would they think this year was typing or a program you can trace? We will title the typing Digital Literacy.

---

world language / comparative religion / philosophy. Computer science is not named in the additional-discipline parenthetical.

[142] NCAA High School Review Committee, *Policies and Procedures 2025–26* and *Policies and Procedures (Staff) 2026–27*. CS courses may be approved if they qualify for graduation credit in mathematics or science and are an academic programming course. Contributes: operating systems, networks and hardware design; programming/coding (Java, C++, Python, Scratch, etc.); data structures; algorithms; cybersecurity; AI (development, design, refinement); robotics; emerging computing and impact on society. Does not contribute: keyboarding/typing; basic internet navigation; usage of word processing, spreadsheet, gaming, generative AI; website maintenance; presentation software. NCAA Eligibility Center, *Home School Toolkit 2025–26*: sample transcript has no Computer Science line; course name on the worksheet must match the transcript title; CLEP, audited courses, and credit-by-exam are not NCAA-approved core courses.

[143] College Board, CLEP Exam Topics, https://clep.collegeboard.org/clep-exams, opened 1 September 2026: no Computer Science CLEP. CLEP Information Systems, listed under Business: material usually taught in an introductory business course; approximately 100 questions / 90 minutes; ACE recommended score 50 for 3 semester hours; $97; does not emphasize hardware design or language-specific programming. Not AP CSP, not AP CSA, not NCAA core.

[144] Google CS First shutdown: turned down 30 June 2025; data deleted. Do not sequence it as a live 2026 course. Experience CS named on that shutdown page as a Scratch-based successor; live syllabus not used as a source here.

We will put Computer Science on the year that has algorithms and a program."

2. "They're fourteen and they never coded, so we missed the window." Or: "They're eight, so they are a native programmer and the older one is not."

What it usually means. A pipeline claim was upgraded to biology. Opportunity helps. A closed window is not in the opened sources.

What to say. "They are a novice. Novices learn. We start with a recipe-algorithm and a machine if we have one. The eight-year-old does not own a different brain. They own more hours if we give them hours."

3. "AP is the real Computer Science, so we should skip the kitchen year." Or: "CSP then CSA is the national order."

What it usually means. Beyond was upgraded to floor, or a catalogue was upgraded to a statute. Opened pages say the two AP courses may be taken in either order.[145]

What to say. "AP is a possible next. The floor is a program they can trace and a system they can talk about. We will not skip the floor to buy an acronym."

4. "Colleges require CS." Or: "NCAA requires a CS year." Or: "We'll CLEP out of CS."

What it usually means. A local ask was upgraded to a national law, or a business exam was upgraded to computer science.

What to say. "UC has no CS area. NCAA does not require a CS year; programming may count as math or science if we submit it honestly. There is no CS CLEP. We will look up *this* college, *this* year."

[145] College Board, AP Computer Science Principles and AP Computer Science A, AP Students pages, opened 1 September 2026, https://apstudents.collegeboard.org/courses/ap-computer-science-principles and https://apstudents.collegeboard.org/courses/ap-computer-science-a. CSP: first-semester introductory college computing; no previous coding required on the opened FAQ; teacher chooses the language; Big Ideas and exam weights on the page (Creative Development 10–13%; Data 17–22%; Algorithms and Programming 30–35%; Computer Systems and Networks 11–15%; Impact of Computing 21–26%). Create performance task due Friday, 30 April 2027; exam Friday, 14 May 2027, as printed on the fetched page. CSA: Java required; units on the opened page include Using Objects and Methods 15–25%; Selection and Iteration 25–35%; Class Creation 10–18%; Data Collections 30–40%; exam Wednesday, 12 May 2027. Students may take the two courses in either order. Independent study possible; College Board recommends a course because of the performance task. Credit is institution-specific. Catalogue detail belongs here in the notes, not as a sales pitch in the chapter body. Homeschool AP Course Audit path not closed in this book's sources.

5. "Unplugged is our CS year, and we do not need a computer." Or: "We finished a Code.org course, so the transcript says Computer Science I."

What it usually means. A catalyst was upgraded to a curriculum, or a tool was upgraded to a credit.[146]

What to say. "Cards can start the idea. Then we run it. Completing a course from a website is work. The credit title is still Computer Science, or Digital Literacy if the work was use. The brand stays off the line."

Five-minute parent warm-up

Before the lesson, paper, no student in the room:

1. Draw this year's two columns: Use | Computer Science. Put last month's actual hours in the cells. If one cell is empty, that is information, not a verdict.
2. Write the sentence you will actually ask: "If a stranger read next year's transcript, would they think this year was typing or a program you can trace?"
3. Write next year's proposed title in ordinary English. Cross out any product name that snuck in. Cross out AI Fluency if it snuck in.

If those three are fluent for you, you can hear a wrong credit. That is the job. The student still says last week's unaided line.

How to teach it this week

This chapter's "try-its" are decision objects. They still have time, materials, a fun, and a skill. The computing hour's shape still holds: you model a choice on *your* paper, they try, you ask one good question, you wait.

Warm-up (3–5 minutes, unaided). Last week's unaided line, without the book: a named part, a saved file, or a traced loop. If they cannot say it, that is the gate, not a shame. You now know where to sit.

Short model (5 minutes). You hold up a dummy transcript line that is wrong: "Scratch I, A." Think aloud: "A stranger cannot map that. I will write Computer

[146] Tim Bell, "CS Unplugged or Coding Classes?," *Communications of the ACM*, 2021. Unplugged is not a complete curriculum; catalyst with programming.

Science, and I will put Scratch in the description. I will not write Computer Science for a year that was only typing."

Student attempt. Transcript Title Drill, then One-Path Calendar if time. Sticky-note product: one true title for next year. You wait.

One good question. "If a stranger read next year's transcript, would they think this year was typing or a program you can trace?" Then wait.[147] Look at the dummy line, not at the child, if the silence is hard.

Mixed practice. One old unaided line. One title repair. One "unplugged idea, then the machine." One optional box — AP, UC, NCAA — only if that box is actually in this family's next two years.

Exit ticket (5 minutes, unaided). They write next year's title in ordinary English. You write the description in one sentence: documents, or programs they can trace. Helper closed. A polished plan a model wrote is performance. The title they can defend is the skill.

How to fade help. You repair a dummy line → they catch the next dummy → they propose next year's title unaided. When they can tell Digital Literacy from Computer Science, the pointing finger goes away. Bring it back when a new object arrives — AP, NCAA, a college page.

When to stop talking. After one model and one question. You hold the key: is the title true? You are not a college-counseling franchise.

Age-band moves

Ages 5–10. The calendar is yours. They still help: "What can you still do without looking?" A named part. A file they can find. A three-step recipe. Short sits. Unplugged-to-machine is the gate you watch — the idea, then a run, if a machine is available and wanted. They do not need NCAA worksheets. They do not need AP names. They need the two columns in your head so you do not title their typing Computer Science later.

Ages 11–14. They sit with you for One-Path Calendar and Transcript Title Drill. They can say whether last month was use or CS. A first text landing, if it has started, is a planned move, not a panic. Co-ops and tutors are a real path; in the

[147] Robert J. Stahl, "Using Think-Time and Wait-Time Skillfully in the Classroom," ERIC Digest ED370885 (1994). General classroom, not a computing trial.

last federal subject-table year, 23 percent of homeschool households used a tutor and 31 percent a co-op.[148] The homeschool still issues the transcript. Brands stay off the title.

Ages 15–18. College-Ask Lookup is for this band. NCAA Honesty Check is only if the student is in the Eligibility Center process. AP boxes are optional. Dual enrollment, if it exists, is a *college* transcript plus a designation on yours. A late novice still starts at the recipe and the trace, even if the transcript year is eleventh grade. High-school *titles* can still be Computer Science while the description tells the truth about supports. Specialty and AP are beyond. Skipping the first program because a helper can emit one is the wrong beyond.

Exact wording you can steal

On the stranger: “If a stranger read next year’s transcript, would they think this year was typing or a program you can trace?”

On titles: “Computer Science. Digital Literacy. Keyboarding. AP Computer Science Principles. AP Computer Science A. Those are the names. The tool goes in the description.”

On the late start: “You are a novice. That is a starting place. It is not a window that closed.”

On AP: “Possible next. Not the floor. Either order if we sit both.”

On UC: “There is no CS letter. Elective, extra science, or math-heavy. Coding-only is not the math area.”

On NCAA: “Programming may count as math or science if we submit it there. Typing and office apps and chatting with a model will not.”

On Unplugged: “The idea, then the machine if we have one and want one.”

On the helper: “A fluent plan is not a year we taught. We still need the program they can trace.”

[148] Cui and Hanson, NCES 2020-001, Table 3: mother provided most instruction in 78 percent of households; tutor 23 percent; co-op 31 percent (2016). 2023 subject tables were not republished in the First Look this book uses.

Try-it — One-Path Calendar

Time. 20 minutes. Parent and older student; parent alone if the child is young.

Materials. Paper with year-boxes. Pencils. Dummy titles: Digital Literacy; Computer Science; optional AP Computer Science Principles; optional AP Computer Science A.

Safety / accounts. No login. No product website required for this sit.

The fun of it. Seeing the path. Being allowed to draw only two boxes.

The skill it builds. Use labelled as use; CS labelled as CS. Not a college-counseling franchise.

How to run it. Fill last year from the folder, not from hope. Fill next year with one true title. Circle AP only if you mean to sit it. Leave NCAA and UC as notes in the margin if they apply. If the CS box would be a lie, write Digital Literacy and plan the first program. You ask the opening question. You wait.

Practice that actually builds learning

A stack of standards printouts with no calendar is assignment. A Pinterest sequence with no folder is activity. Practice an honest title. Then mix.

Blocked, for a new move. The day they first catch "Scratch I," two more dummy lines, same move. The day an unplugged idea first runs on a machine, one more idea, same gate.

Mixed, for when to use it. Later the same week: a title drill, a calendar repair, last week's unaided trace, one college page if that is real for this house.

Brief retrieval. Yesterday's unaided line; next year's title; one thing that is *not* Computer Science. A short oral.

One incorrect example to diagnose. A dummy transcript: "Code.org 101, A" beside a folder of Hour of Code certificates, or "AP Computer Science" as a vague third course, or "AI Fluency." Ask what a stranger would think the year was. Then an isomorphic repair, unaided.

Kitchen internships and robotics clubs can motivate. They do not replace the title. Other computing topics — game engines, robotics careers — get a pointer, not a

secret ninth teaching chapter.

Try-it — Transcript Title Drill

Time. 10 minutes. All bands at the table; the writing is yours at 5–10.

Materials. A dummy line on paper. A list of the locked titles.

Safety / accounts. No login.

The fun of it. Catching "Scratch I." Being the person who saves the stranger from a brand.

The skill it builds. Computer Science versus Digital Literacy versus Keyboarding versus AP Computer Science Principles versus AP Computer Science A. Never a brand.

How to run it. You read five dummy lines, including at least one honest CS year and one slides-only year. They sort. They rewrite the false ones. Numbered levels — Computer Science I — are ordinary English if the family uses them; the credit is still Computer Science, not Python I, unless a college dual-enrollment transcript already uses that name.

Try-it — Unplugged-to-Machine Gate

Time. 10 minutes.

Materials. Last month's Unplugged idea (binary cards, a sorting line, a recipe-algorithm) and a machine, if available and wanted.

Safety / accounts. Parent-held account if the run needs one. No new child account for a ten-minute gate. No exploit on the family router.

The fun of it. The same idea, running.

The skill it builds. Unplugged as catalyst, not a year without a computer.[149]

How to run it. They say the idea with the machine off. They run the smallest version that moves. If there is no machine, they say what they would run, and you plan the week a machine is in the room. A year that never runs is a year you will

[149] Tim Bell, "CS Unplugged or Coding Classes?," *Communications of the ACM*, 2021. Unplugged is not a complete curriculum; catalyst with programming.

not title Computer Science unless the work was really systems on paper *and* you can defend the claim. This book would rather you plug it in.

Try-it — College-Ask Lookup

Time. 15 minutes. Ages 15–18, and 11–14 if a parent is planning early.

Materials. One named college's page, or UC A–G G/D/C, or the current NCAA fact sheet. Paper.

Safety / accounts. Parent opens the page. No child account on an admissions portal required for this sit.

The fun of it. A real number for *this* family.

The skill it builds. UC does not require CS. NCAA does not require a CS year. Two AP names exist. None of that is a kitchen law, and none of it is "colleges require CS" as a universal.

How to run it. Write the question before you open the page: Does *this* place ask for a CS year? If UC is in view, write G, extra D, math-heavy C. If NCAA is in view, write "not required; programming as math or science if honest." If the page is silent, the answer is silent — a missing ask stays missing. Credit for an AP exam is institution-specific. You record what the page said, dated.

Try-it — NCAA Honesty Check

Time. 15 minutes. Only if the student is in the Eligibility Center process.

Materials. The dummy title plus the contribute / does-not list from the current High School Review Committee chart. The Core-Course Worksheet if you have it.

Safety / accounts. No login required to read a public chart. Do not file a fake worksheet.

The fun of it. Catching Word-as-CS before a stranger does.

The skill it builds. Programming as math or science; keyboarding will not be approved. Worksheet name must match the transcript.[150]

[150] NCAA High School Review Committee, *Policies and Procedures 2025–26* and *Policies and Procedures (Staff) 2026–27.* CS courses may be approved if they qualify for graduation credit in

How to run it. Sort last year's computing work into contributes / does not. If the year was typing, office apps, or chatting with a model, it does not go on a CS, math, or science core line as computing. If the year was academic programming, pick math or science as the area you will actually submit, and make the titles match. CLEP, audited courses, and credit-by-exam are not NCAA-approved core courses.

Debug / talk box

Opening. "If a stranger read next year's transcript, would they think this year was typing or a program you can trace?"

Not: "What is CSTA?"

Follow-ups 1. What can you still do without looking? 2. Is this Digital Literacy, or Computer Science? 3. If we stopped in June, would "Computer Science" be true? 4. Is AP the floor, or beyond? 5. Where does the tool's name belong — title, or description?

How to wait. After you ask, a slow three is a minimum, not a cliff.[151] Look at the dummy line or the year-boxes, not at the child. Say, if you need a sentence: "Take your time. I'll wait."

Stuck silence usually means. The first language of computing never had a program — which is the gate, not a shame; the question was a recitation item about a standards body; they think a certificate from a website is a credit; wait time has been too short; or a helper already wrote a four-year plan and there is nothing left to think.

Next move. Point at last week's unaided line. Offer two titles. Repeat the drill more slowly. A rehearsed frame — what we did / what a stranger would think / the true title — not a lecture, and not a brand.

---

mathematics or science and are an academic programming course. Contributes: operating systems, networks and hardware design; programming/coding (Java, C++, Python, Scratch, etc.); data structures; algorithms; cybersecurity; AI (development, design, refinement); robotics; emerging computing and impact on society. Does not contribute: keyboarding/typing; basic internet navigation; usage of word processing, spreadsheet, gaming, generative AI; website maintenance; presentation software. NCAA Eligibility Center, *Home School Toolkit 2025–26*: sample transcript has no Computer Science line; course name on the worksheet must match the transcript title; CLEP, audited courses, and credit-by-exam are not NCAA-approved core courses.

[151] Robert J. Stahl, "Using Think-Time and Wait-Time Skillfully in the Classroom," ERIC Digest ED370885 (1994). General classroom, not a computing trial.

For the student

Your transcript is a picture of work you did. A stranger will not have sat at your table. They will have a page. The page has to tell the truth.

Computer Science means you made a machine follow steps you can trace, or you can talk about a system — hardware, software, a network, data, impacts — as something that works, not only as buttons you pressed. Digital Literacy and Keyboarding are real titles for real work. They are not lesser people. They are different work.

A tiny example. Last year you typed letters to Grandma and made three slides about a garden. That is Digital Literacy. This year you wrote a loop that stopped when you said, and you can say what you expected it to do. That is Computer Science. If you put last year on the page as Computer Science, a stranger would think you traced programs you did not trace.

Try 1. Say last week's unaided line without this book: a named part, a saved file, or a traced loop. If you cannot, say so. That tells us where to sit tomorrow.

Try 2. Help write next year's title in ordinary English. If a product name gets on the line, take it off. Put it in the sentence that describes *how* you learned.

Explain it back. If a stranger read next year's transcript, would they think this year was typing or a program you can trace? What can you still do without looking?

A challenge. Repair five dummy titles. Catch "Scratch I," "Code.org 101," "AP Computer Science" as a vague third course, "AI Fluency," and one honest Computer Science year that should stay. Then, if college or NCAA is in your actual next two years, open one named page with a parent and write what *that* page asks — not what a rumor asks.

If it isn't clicking

No shame. Three stuck places, and a next move for each.

1. There was never a program. The calendar is full of typing, slides, and chat. The word Computer Science is already on last year's page.

Next move. Repair the title. Digital Literacy is an honest year. Then Chapter 6, this month, even if the birthday is fifteen. A novice is a novice. Five weeks of an

introductory environment has been enough for high-school beginners to learn in a classroom study; your kitchen is not that classroom, and it is proof of possibility, not a stopwatch.[152] The gate is a program they can trace, not a scolding about 2016 survey percents.

2. The brand is doing the naming. Scratch I. Code.org 101. A completion badge in the title column.

Next move. Transcript Title Drill until the locked list is in the mouth. The provider field, or the description, may name the tool. The credit may not.

3. Beyond arrived too early, or too loudly. AP as the first hour. A college CS course as a ninth-grade panic. A helper emitting a four-year plan nobody can defend. "Colleges require CS" with no page open.

Next move. College-Ask Lookup for *this* family. One page. Dated. AP stays optional. UC's missing CS area stays missing. NCAA's missing required year stays missing. The floor remains a traced program and an honest title.

When to slow down. If they cannot say last week's unaided line, the calendar waits. The hour returns to the file, the part, or the loop. Coverage is a map. Depth is the hour.

When to go ahead. If they can defend next year's title, tell use from CS, and trace last week's program or find last week's file, you have enough sequence to keep walking. Records, in the next chapter, is how you put that walk on a page a stranger can file.

When to get a human tutor. If AP, dual enrollment, or NCAA paperwork is the actual next object, a person who has done that paperwork is the right help — a coordinator, a counselor, a homeschool umbrella that already files NCAA worksheets. A model that invents a CS CLEP is the wrong help. There is no CS CLEP.[153]

---

[152] David Weintrop and Uri Wilensky, "Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms," *ACM Transactions on Computing Education* 17, no. 3 (2017). High-school novices in a five-week introductory comparison learned in either modality. Useful study, not a kitchen stopwatch and not a native-coder claim. Access date for URLs in these notes: 1 September 2026.

[153] College Board, CLEP Exam Topics, https://clep.collegeboard.org/clep-exams, opened 1 September 2026: no Computer Science CLEP. CLEP Information Systems, listed under Business: material usually taught in an introductory business course; approximately 100 questions / 90 minutes; ACE recommended score 50 for 3 semester hours; $97; does not emphasize hardware design

Tools, including AI

Short. Point back to the computing hour.

You may, on your account, after the child has tried: ask a helper to explain *to you* a named page (UC A–G G, the NCAA fact sheet, an AP Students page); make extra dummy transcript lines with the answer key hidden; draft a labelled SCRIPT for a title-drill you vet; give a hint after an attempt; diagnose a calendar already drawn; draft talk-box questions you vet.

The helper does not write the four-year plan as if it had been taught. It does not invent a college requirement. It does not invent a CS CLEP. It does not put a brand on the credit line. It does not skip the first program because it can emit one. A fluent schedule is performance. The title that matches the folder is the skill.

Code.org is a tool, not law. Scratch is a tool, not a transcript title. CS First is retired. One family computer is enough.

What “done enough” looks like

Placement by skill, not birthday.

A one-page family calendar with use and CS named separately. AP, NCAA, and UC as optional boxes, not as the hour. Next year’s title in ordinary English, from the locked list. Last week’s unaided line still in the mouth.

Ages 5–10. The parent can hold the two columns. The child can say what they can still do without looking. Unplugged ideas have a plan to run.

Ages 11–14. The student can catch a brand in a dummy title and tell Digital Literacy from Computer Science.

Ages 15–18. The student can look up one named college, UC, or NCAA page and write what it actually asks. If they are in the Eligibility Center process, they can sort last year’s work into contributes / does not.

Before you move on. If we stopped in June, the word Computer Science would be true only if a program they can trace, or a system they can talk through, is in the folder. Typing has its own honest name. Records is next: how a stranger reads the page. Resources after that: programs by fit, not by rank.

---

or language-specific programming. Not AP CSP, not AP CSA, not NCAA core.

Chapter 9

Records

The computing *work* can be as serious as any school's. The *credit* is a family claim until a stranger can use it — a college admissions reader, an NCAA Eligibility Center staffer, a public-school counselor placing a transfer. That stranger will not have sat at your kitchen table. They will have a page.

This chapter is how you make that page. Lawful, safe, and readable to a stranger are three jobs. Keep them in view.

This week you can take last year's computing folder and write one course line a counselor would recognize. Today the student can list the programs they can still trace, or the documents they can still find.

Illustration. A folder spread with dummy titles only: Digital Literacy, Computer Science, AP Computer Science Principles, Keyboarding. Dummy text. No real names. No portraits. No brand spines.

What you are making

You are making a record of computing, not a national diploma.

There is no federal homeschool diploma and no national computer-science credit.

Texas, North Carolina, and New York do not issue one.[154] [155] [156] The NCAA Eligibility Center treats proof of graduation as a family or umbrella document.[157] Pennsylvania has its own supervisor or approved-organization diploma after listed graduation courses; those courses do not name computer science.[158] A parent-issued diploma can be a real piece of paper. It is not a registrar's national CS credit.

Four other objects belong in their own envelopes. Dual-enrollment Computer Science lives on a *college* transcript. AP scores are exam objects. CLEP Information Systems, if anyone sits it, is a business exam, not a high-school CS year. A state-compliance packet — a Pennsylvania evaluator letter, a Virginia evidence-of-progress file, a North Carolina annual test (English and math, not CS), a New York IHIP and quarterlies — is what your statute asked for. A college that wants a transcript is asking for a different envelope.

Virginia prints diploma seals for public diplomas. There is no Computer Science seal on the opened seals page. The STEM Seal is a public-diploma object. This book does not invent a CS seal for a homeschool credential.[159]

---

[154]Texas Education Agency, "Home Schooling" and "Alternative Schooling" (accessed 1 September 2026). No state homeschool diploma. *Leeper* list: reading, spelling, grammar, mathematics, good citizenship — computer science is not named. Transfer treated like an unaccredited private school.

[155]North Carolina DNPE, diplomas FAQ and "Starting a Home School" (last modified 3 August 2026; accessed 1 September 2026). State issues neither diploma nor transcript. Annual test: English grammar, reading, spelling, mathematics — not CS. Five-hour day is a recommendation. G.S. 115C-563(a) allows non-parent instructors.

[156]NYSED, "Home Instruction Questions and Answers"; 8 NYCRR 100.10 (Cornell LII). No local or Regents diploma for home instruction. Grades 1–6 do not name CS; grades 9–12, three units of electives; a unit is 6,480 minutes. Placement by need.

[157]NCAA Eligibility Center, *Home School Toolkit 2025–26*. Proof of graduation is a family or umbrella artifact. Sample transcript has no Computer Science line. Course name on the worksheet must match the transcript title. Evaluation after documents are in *and* an NCAA school has placed the student on an Institutional Request List. Homeschool means parent, tutor, or umbrella creates, instructs, assesses, and awards grades. Dual-enrollment designated on the homeschool transcript as a college course, plus the official college transcript. CLEP, audited courses, and credit-by-exam are not NCAA-approved core. Hawaii and New York: a homeschool-issued diploma is not accepted as proof of graduation.

[158]24 P.S. § 13-1327.1. Elementary and secondary do not name CS. Graduation 9–12: four English, three math, three science, three social studies, two arts and humanities. Pennsylvania-specific supervisor or approved-organization diploma.

[159]Virginia Department of Education, "Home Instruction"; Code of Virginia § 22.1-254.1; "Diploma Seals" (accessed 1 September 2026). Notice by 15 August; evidence of progress by

Titles a stranger can read

Print these and only these as credit names:

Computer Science. Algorithms, programs, systems, data, impacts. Not typing.

Digital Literacy / Technology Applications. Files, accounts, search, safety habits, documents, spreadsheets at a citizen level.

Keyboarding. Typing and text entry. A skill title, not Computer Science.

AP Computer Science Principles. The College Board course and exam of that name.

AP Computer Science A. The College Board Java course and exam of that name.

Those names travel. Never a brand. Never Scratch I. Never Code.org 101. Never "AP Computer Science" as a vague third course. Never AI Fluency as a kitchen-table badge. Never a year of Word labelled Computer Science.

The title is the genre. The program you used is the method. Scratch, CS Unplugged, a Python course, LibreOffice, a co-op — those may appear in a description. They are not the course name. "Computer Science — Scratch" is still a brand in the title column. Curriculum provider is a separate field. Put the tool there.

Numbered levels (Computer Science I, Computer Science II) are ordinary high-school English if the family uses them. The credit is still Computer Science, not Python I, unless a college dual-enrollment transcript already uses that name.

NCAA's sample homeschool transcript has **no Computer Science line**. That absence is a fact about the sample, not a prohibition.[160] If a family wants NCAA-readable CS, they add a title that matches the worksheet, in math or science. There

1 August. Named seals include Governor, Board of Education, CTE, STEM, Civics, Biliteracy, Science and the Environment. No Computer Science seal. STEM Seal is a public-diploma object.

[160] NCAA Eligibility Center, *Home School Toolkit 2025–26*. Proof of graduation is a family or umbrella artifact. Sample transcript has no Computer Science line. Course name on the worksheet must match the transcript title. Evaluation after documents are in *and* an NCAA school has placed the student on an Institutional Request List. Homeschool means parent, tutor, or umbrella creates, instructs, assesses, and awards grades. Dual-enrollment designated on the homeschool transcript as a college course, plus the official college transcript. CLEP, audited courses, and credit-by-exam are not NCAA-approved core. Hawaii and New York: a homeschool-issued diploma is not accepted as proof of graduation.

is no CS checkbox on the opened Core-Course Worksheet subject list. The course name on the worksheet must match the transcript title.

What belongs on the computing line

For each high-school computing course, write:

- the **title** from the list above;
- the **year**, and the grade equivalent you actually used;
- the **credit** — commonly 1.0 for a year; New York's "unit" is 6,480 minutes, not a brand;
- the **grade** and the scale (A = 90–100, or whatever scale you truly used);
- a **brief description** of texts or environments, programs traced, systems studied, writing, and grading — not a brand as the course name;
- **who taught and who graded**;
- your **signature** and a date;
- **external artifacts** when they exist: an AP score; a dual-enrollment college transcript; a state-required test or evaluator letter. Not a website completion badge as the credit.

Name the work. Name the tracing. Name the documents if the year was literacy. A sentence that says only a product name is a brand, not a course.

Here is an illustration, not a reported family:

> **Computer Science** · 2025–26 · 1.0 credit · Grade: A (A = 90–100, B = 80–89)
>
> High-school computer science: algorithms and programs the student can trace, files and operating systems as software, networks as packet paths, data, impacts of computing. Environment: block-based introductory work, then a text landing in Python. Unplugged ideas run on a machine. Writing: two unaided traces with the screen off, one short paper from named sources on a computing impact. Grading: 40 percent programs and traces, 40 percent unaided writing and tests, 20 percent discussion. Instructor: [parent name]. Signature: _____ Date: _____

And an honest literacy year:

> **Digital Literacy** · 2025–26 · 1.0 credit · Grade: B (A = 90–100, B = 80–89)
>
> Documents, sheets, and slides a stranger could open; files named and found; short keyboard technique; search and source habits. Tools: a word processor, a spreadsheet, a slide tool on the family computer. Grading: 50 percent named files the student can still find, 30 percent unaided writing, 20 percent discussion. Instructor: [parent name]. Signature: _____ Date: _____

Change the environments and the counts to what is in *your* folder. The shape stays. The folder is not the transcript. The transcript is the map to the folder.

If the year was typing, title it **Keyboarding**. If the year mixed use and CS, split the lines or tell the truth in the description — and do not let the slides carry the Computer Science noun.

The law is a pattern, not one form

U.S. homeschooling is state law. There is no federal computer-science office. Dates and subject lists change. Read your current department page and the statute it cites. A color-coded chart on a membership site is not your statute.[161] Five official-page examples show the range. None of them requires computer science for homeschool.

Texas. The Agency does not regulate home-school programs.[162] Parents follow a bona fide written curriculum: reading, spelling, grammar, mathematics, and good citizenship. Computer science is not named. Lawful without a CS course is legality, not a college-ready plan. Texas does not award a homeschool diploma. Transfer is treated like an unaccredited private school.

North Carolina. Notice of Intent. Nine calendar months. Each year, a nationally standardized test in English grammar, reading, spelling, and mathematics — not

[161] Official department pages illustrate a pattern, not a fifty-state code. This chapter is not legal advice. A membership chart is not a statute.

[162] Texas Education Agency, "Home Schooling" and "Alternative Schooling" (accessed 1 September 2026). No state homeschool diploma. *Leeper* list: reading, spelling, grammar, mathematics, good citizenship — computer science is not named. Transfer treated like an unaccredited private school.

computer science.[163] The home school, not the State, issues the diploma. A five-hour day is a recommendation, not the statute. Persons other than the parent may instruct. The homeschool still issues the transcript.

Pennsylvania. Elementary and secondary lists do not name CS. Graduation in grades 9–12 requires four English, three math, three science, three social studies, two arts and humanities — not CS.[164] A supervisor or approved organization may issue a Pennsylvania homeschool diploma. That credential is Pennsylvania's.

Virginia. Notice by 15 August, a parent-written list of subjects, and evidence of progress by 1 August.[165] Computer science is not named as required. There is no CS diploma seal.

New York. Letter of intent, an Individualized Home Instruction Plan, quarterly reports, and an annual assessment. Grades 1–6 do not name CS. Grades 9–12 have three units of electives where a CS year *may* sit. A unit is 6,480 minutes.[166] Home instruction does not yield a local or Regents diploma. Placement is by need, not birthday. The legislature has not authorized part-time public instructional participation.

If you live in none of those states, you still have a department page. Open it.

If they transfer

Transfer is a local placement problem. Texas districts commonly assess mastery. New York leaves grade placement to the principal. North Carolina notes that a conventional-school principal has no legal obligation to accept home-school credit,

[163] North Carolina DNPE, diplomas FAQ and "Starting a Home School" (last modified 3 August 2026; accessed 1 September 2026). State issues neither diploma nor transcript. Annual test: English grammar, reading, spelling, mathematics — not CS. Five-hour day is a recommendation. G.S. 115C-563(a) allows non-parent instructors.

[164] 24 P.S. § 13-1327.1. Elementary and secondary do not name CS. Graduation 9–12: four English, three math, three science, three social studies, two arts and humanities. Pennsylvania-specific supervisor or approved-organization diploma.

[165] Virginia Department of Education, "Home Instruction"; Code of Virginia § 22.1-254.1; "Diploma Seals" (accessed 1 September 2026). Notice by 15 August; evidence of progress by 1 August. Named seals include Governor, Board of Education, CTE, STEM, Civics, Biliteracy, Science and the Environment. No Computer Science seal. STEM Seal is a public-diploma object.

[166] NYSED, "Home Instruction Questions and Answers"; 8 NYCRR 100.10 (Cornell LII). No local or Regents diploma for home instruction. Grades 1–6 do not name CS; grades 9–12, three units of electives; a unit is 6,480 minutes. Placement by need.

especially in grades 10–12.[167] A brand in the title column will not be read as Computer Science. Another reason the title should already look like Computer Science, or Digital Literacy, when that is the work.

College-ready objects — not the course

Scores help a stranger believe the transcript. They do not replace the program the student can trace.

AP Computer Science Principles and **AP Computer Science A** are optional exam objects. There is no course titled simply "AP Computer Science." Students may take the two in either order. CSP's opened page describes a first-semester introductory college course in computing; the teacher chooses the language; a Create performance task sits beside the exam. CSA is a Java programming course. Credit is set by each college.[168] Put the score on its own line. Until you have confirmed how a home course may carry the AP title, the safer transcript line is **Computer Science**, with the exam as a separate artifact. Site rules and a homeschool AP Course Audit path are local questions this book does not close.

SAT Subject Tests were discontinued in June 2021. They are not an object you can still sit.

CLEP has no Computer Science exam. Information Systems is listed under Business: material usually taught in an introductory business course; about 100 questions in 90 minutes; ACE's typical recommendation is a score of 50 for three semester hours; $97 on the opened page. It does not emphasize hardware design or language-specific programming.[169] It is not NCAA core. It is not a high-school CS year. Audited, CLEP, and credit-by-exam courses are not NCAA-approved core courses.

[167]TEA "Home Schooling"; NYSED Q&A; North Carolina DNPE records FAQ (accessed 1 September 2026). A conventional-school principal has no legal obligation to accept home-school credit, especially in grades 10–12.

[168]College Board, AP Computer Science Principles and AP Computer Science A, AP Students pages (accessed 1 September 2026). CSP Create due Friday, 30 April 2027; CSP exam Friday, 14 May 2027; CSA exam Wednesday, 12 May 2027, as printed on the fetched pages. Either order. Credit is institution-specific. Homeschool AP Course Audit path and testing-site rules not closed in this book's sources.

[169]College Board, CLEP Exam Topics and CLEP Information Systems (accessed 1 September 2026). No Computer Science CLEP. Information Systems listed under Business; ACE typical recommendation score 50, 3 semester hours; $97; not NCAA core.

Dual enrollment is a college course taken during high school. Federal statistics define it as postsecondary credit during high school, excluding AP and IB — not a homeschool rate.[170] Access is local. Map the course to **Computer Science**, with the college named, or use the college's actual course name on a designated college line. Send the official college transcript. NCAA wants the college course designated on the homeschool transcript, plus that separate college transcript.[171]

UC A–G, if California public-university admission is in view, has no CS area. G elective; D additional science year 3+; C only with substantial math. Coding-only does not qualify as C. How a homeschooler gets onto an A–G list is not closed here. Look up the current guide in the year you apply.[172]

NCAA, if athletics are in view

If Division I or Division II sports might be in the picture, computing planning starts in ninth grade *only if* you want a core-course line. Computer science is **not** a required year and **not** additional-core like world language.

Division I asks for sixteen core-course credits. The additional-discipline parenthetical names world language, comparative religion, or philosophy — not CS.[173] A student can meet Division I without a computing year.

Academic programming **may** be approved if it qualifies for graduation credit in mathematics or science **and** it is an academic programming course. Operating

[170] National Center for Education Statistics, *Dual Enrollment: Participation and Characteristics*, NCES 2019-176. Dual/concurrent enrollment as postsecondary credit during high school, excluding AP and IB. Not a homeschool computing rate.

[171] NCAA Eligibility Center, *Home School Toolkit 2025–26*. Proof of graduation is a family or umbrella artifact. Sample transcript has no Computer Science line. Course name on the worksheet must match the transcript title. Evaluation after documents are in *and* an NCAA school has placed the student on an Institutional Request List. Homeschool means parent, tutor, or umbrella creates, instructs, assesses, and awards grades. Dual-enrollment designated on the homeschool transcript as a college course, plus the official college transcript. CLEP, audited courses, and credit-by-exam are not NCAA-approved core. Hawaii and New York: a homeschool-issued diploma is not accepted as proof of graduation.

[172] University of California Office of the President, A–G Policy Resource Guide, areas C, D, and G (accessed 1 September 2026). No CS subject area. Homeschool A–G list pathway not closed here.

[173] NCAA Division I Academic Requirements fact sheet, September 2023, still live 1 September 2026. Sixteen core courses; CS not named as additional discipline.

systems, networks, hardware design, programming, data structures, algorithms, cybersecurity, AI as development and design, robotics, and computing's impact on society can contribute. Keyboarding, basic internet navigation, usage of word processing, spreadsheets, gaming, generative AI, website maintenance, and presentation software will not.[174]

The homeschool packet includes a transcript, an administrator statement that instruction followed *state* law, a Core-Course Worksheet for each core course, and proof of graduation. The worksheet title equals the transcript title. Evaluation begins after those documents are in **and** an NCAA school has placed the student on an Institutional Request List.[175] There is no preapproved homeschool computing curriculum. A full-time online school with online teachers is not a homeschool on that toolkit's definition.

CLEP is not a workaround. Hawaii and New York have a graduation-proof rule, not a CS rule: the Eligibility Center cannot accept proof of graduation from a homeschool-issued diploma in those states.[176]

Place by skill, not birthday

A publisher's "grade 6 computer book" is a scope, not a legal grade. New York already says instruction should match the student's needs and previous achieve-

[174]NCAA High School Review Committee, *Policies and Procedures 2025–26* and Staff Policies 2026–27. Academic programming may count as math or science if submitted there. Keyboarding, office applications, and generative-AI *usage* will not contribute.

[175]NCAA Eligibility Center, *Home School Toolkit 2025–26*. Proof of graduation is a family or umbrella artifact. Sample transcript has no Computer Science line. Course name on the worksheet must match the transcript title. Evaluation after documents are in *and* an NCAA school has placed the student on an Institutional Request List. Homeschool means parent, tutor, or umbrella creates, instructs, assesses, and awards grades. Dual-enrollment designated on the homeschool transcript as a college course, plus the official college transcript. CLEP, audited courses, and credit-by-exam are not NCAA-approved core. Hawaii and New York: a homeschool-issued diploma is not accepted as proof of graduation.

[176]NCAA Eligibility Center, *Home School Toolkit 2025–26*. Proof of graduation is a family or umbrella artifact. Sample transcript has no Computer Science line. Course name on the worksheet must match the transcript title. Evaluation after documents are in *and* an NCAA school has placed the student on an Institutional Request List. Homeschool means parent, tutor, or umbrella creates, instructs, assesses, and awards grades. Dual-enrollment designated on the homeschool transcript as a college course, plus the official college transcript. CLEP, audited courses, and credit-by-exam are not NCAA-approved core. Hawaii and New York: a homeschool-issued diploma is not accepted as proof of graduation.

ment.[177]

Can the child find a file they made yesterday and tell you it is a document, not a program? If that is the work, they are still in a literacy year, whatever their age. Can they say what they expected a process to do, and what it did instead? If not, they are still in a first-programs stage, whatever the transcript year. High-school *titles* can still be Computer Science while the description tells the truth about supports.

A proficiency note, if you can honestly hear the skill, is a class-performance note. It is not an unofficial CSTA badge. This book will not print "meets national computer-science standards" or "CSTA certified" as a transcript line. Those sentences are unsafe.

A template, a signature, and the truth

A tool, including an AI helper on your side of the table, may draft a **template**: blank lines for title, year, credit, grade, environments, programs traced, writing, grading, signature. That is a form. It is not a record until you fill it from the folder.

You read every line. You check the titles against the locked list and the traces against the notebook. Then you sign. Hours that were not taught are a false statement. A polished program the child cannot trace is not the child's computer science. A generated data table is not Figure 1. The tool may format. You attest.

Unsafe lines to keep off the page: "Meets national computer-science standards." "CSTA certified." "SAT Subject Test." A Code.org completion badge as a credit. "AP Computer Science" as a vague third course. "We CLEP'd CS." "Google CS First" as a current path. "AI Fluency." A year of Word labelled Computer Science.

For the student

Your transcript is a picture of work you did. You saved files, or you traced programs, or both. When a parent asks you to list what you finished this year, name the documents you can still find, the programs you can still trace with the screen

[177] NYSED, "Home Instruction Questions and Answers"; 8 NYCRR 100.10 (Cornell LII). No local or Regents diploma for home instruction. Grades 1–6 do not name CS; grades 9–12, three units of electives; a unit is 6,480 minutes. Placement by need.

off, and two bugs you actually fixed. If a line on the page is not true, say so before anyone signs.

What “done enough” looks like

You can hand a stranger one page per high-school computing course. The title is from the list a counselor already knows. Computer Science appears only when algorithms, programs, systems, data, or impacts were the work. Digital Literacy and Keyboarding appear when use was the work. The description names environments, traces or documents, writing, and grading, and you would still sign it tomorrow morning. The folder matches the page. That is enough.

Chapter 10

Resources

This chapter names programs families actually use. It does not rank them. It does not sell them. It is a fit list so you can match a stack to the table you already have: how much of the teaching you will carry, whether the hour is unplugged or blocks or text, whether the work is computer *use* or computer *science*, whether a high-school credit shape is even in view, and whether a parent who is not a programmer can hear a wrong turn.

A "grade 6 coding book" is a publisher's scope, not a legal grade. Placement is by skill. The transcript, as Chapter 9 said, still says Computer Science, or Digital Literacy, or Keyboarding. Combining two honest programs does not mint a new title.

This week you can name *why* the tools on your machine fit this child. Today the student can open one environment that is actually in the house, or run one Unplugged idea you already have the cards for.

Illustration. A table of books as objects, a brick-of-code still-life, a small single-board computer as an optional object — not a shopping ad. No brand logos as ads. No portraits. No readable product screens.

How to read a program

Five questions do the work.

Parent load. Is the adult running a game with cards, sitting beside a sprite, hosting a teacher-portal year, or building lessons around language documentation written

for people who already program?

Unplugged, blocks, or text. Cards and string? A block environment? A text language? A planned landing from one to the next?

Use or science. Documents, sheets, slides, typing — Digital Literacy or Keyboarding. Algorithms, programs, systems, data, impacts — Computer Science. Both can live in a year. They still want honest names.

High-school credit shape. Does this path describe a year a stranger could file as Computer Science? Or is this an elementary hour, a month of lessons, or a university course borrowed at the kitchen table?

Can you hear a wrong turn? A non-programmer parent needs to notice "it ran, so I understand," "the computer is thinking," and "slides are CS." If the environment hides every mismatch, pick a smaller ramp.

Where a company makes a strong claim — hours, grade bands, alignment with CSTA or the Framework, a safe community — that claim is the company's. This chapter reports what the page said. It does not certify the slogan.

This chapter names programs the opened pages named. It does not print a year-by-year cycle or a lesson count as if this book had measured them. Open the current page if you use the program, and write the transcript from the work the child actually did.

Apps are practice inside a course, not the course. Tutors and co-ops are a real path: in the last federal subject-table year, 23 percent of homeschool households used a tutor and 31 percent a co-op.[178] North Carolina's statute already names that other people may instruct. The homeschool still issues the transcript. This book does not invent a market size.

One family computer is enough. A second machine, or a small single-board computer, is optional convenience, not a moral upgrade.

[178] Jiashan Cui and Rachel Hanson, *Homeschooling in the United States: Results from the 2012 and 2016 Parent and Family Involvement Survey*, NCES 2020-001, Table 3: tutor 23 percent; co-op 31 percent (2016). Access date for URLs in these notes: 1 September 2026.

CS Unplugged — University of Canterbury

CS Unplugged introduces building blocks of how computers work without using a computer at all: cards, string, crayons. The materials are offered under a Creative Commons license. The project names an "at home" path.[179]

Fit. Ages 5–10 especially; 11–14 when the family wants CS ideas without a login. Parent load: you run a game, not a compiler. A parent who is not a programmer can hear a missing step in a card sort.

What it is not. A high-school Computer Science credit by itself. Not NCAA programming. Not AP. Bell, writing later about the project, says Unplugged is not a complete curriculum and never sought to replace programming.[180] Use it as a catalyst. Then, if a machine is available and wanted, run the same idea.

Transcript. If the year also includes programs they can trace, **Computer Science**, with Unplugged in the description. If the year never left the cards and you cannot defend a systems claim, do not stretch the noun.

ScratchJr and Scratch

ScratchJr is a free app from DevTech at Boston College and the Scratch Foundation, described on the opened page for ages 5–7, with a named curriculum (CAL).[181] Scratch, from the Scratch Foundation and developed at the MIT Media Lab, is a free block-based environment. "Safe, moderated online community" is the foundation's own safety language. User-count figures are foundation copy,

[179]Tim Bell, Ian H. Witten, and Mike Fellows, *Computer Science Unplugged* (2015 revision), https://classic.csunplugged.org/documents/books/english/CSUnplugged_OS_2015_v3.1.pdf. Creative Commons. Tim Bell, "CS Unplugged or Coding Classes?," *Communications of the ACM*, 2021: not a complete curriculum; catalyst with programming. Project pages name an at-home path.

[180]Tim Bell, Ian H. Witten, and Mike Fellows, *Computer Science Unplugged* (2015 revision), https://classic.csunplugged.org/documents/books/english/CSUnplugged_OS_2015_v3.1.pdf. Creative Commons. Tim Bell, "CS Unplugged or Coding Classes?," *Communications of the ACM*, 2021: not a complete curriculum; catalyst with programming. Project pages name an at-home path.

[181]ScratchJr, DevTech (Boston College) and Scratch Foundation, opened product page: free app; ages 5–7 as stated there; CAL curriculum named.

not an NCES homeschool count.[182]

Fit. ScratchJr: ages 5–10, parent beside the child. Scratch: the publisher's positioning is roughly 8–14; younger with a parent; older as a studio, not as AP Computer Science A. Parent load: sit beside the first sprite; fade as they can say what they expected.

What it is not. A transcript title. Not live open internet chat as the 5–10 default hour. Not AP. NCAA names Scratch as an *example language inside* an academic programming course — the course is still titled Computer Science.[183] K–5 still prefers scripts and parent-child work.

Transcript. Computer Science, with Scratch or ScratchJr in the description or the provider field. Never Scratch I.

Code.org Computer Science Fundamentals

Code.org's Computer Science Fundamentals is a set of elementary courses (A–F on the opened page), described as about a month, 13–17 lessons designed for 45-minute periods, with Express paths. The page claims alignment with K–5 CSTA and the Framework. That alignment sentence is the company's.[184]

Fit. Ages 5–10; Express into 11–14. Parent load: a portal hour you can sit beside. A tool, not law. Completing Course F is not Computer Science I.

What it is not. A high-school credit. Not AP Computer Science Principles. Not "the CSTA course." Not a transcript title. Code.org sat on the Framework steering committee. That is authorship, not a statute.

Transcript. Computer Science if the work was programs they can trace; otherwise do not let a completion certificate mint the noun. Never Code.org 101.

---

[182] Scratch Foundation / MIT Media Lab, Scratch, opened product pages: free; block-based; community safety language is the foundation's. User-count figures are not an NCES homeschool count.

[183] NCAA High School Review Committee, *Policies and Procedures (Staff) 2026–27*: Scratch named as an example language inside academic programming, not as a required K–12 environment.

[184] Code.org, Computer Science Fundamentals, opened course page: Courses A–F; about a month; 13–17 lessons designed for 45-minute periods; Express paths; alignment with K–5 CSTA and the Framework claimed on the page — company claim, not a finding of this book.

CMU CS Academy

Carnegie Mellon University's CS Academy offers free graphics-based Python: Exploring (described as about 40 hours, middle school); CS1 (described as about 180 hours, eighth/ninth or high school); 15-111 honors with a named college-credit *opportunity*. The page names a homeschool-educator teacher-account path.[185] Hour counts and the credit opportunity are the program's sentences, not a finding of this book, and not a guarantee that a given homeschooler can transcript the college credit.

Fit. Ages 11–14, Exploring or CS1; ages 15–18, CS1 or 15-111. Parent load: a teacher portal; you still hear a green run the child cannot trace. Text landing with graphics as motive.

What it is not. Automatically NCAA core. Not AP Computer Science A (Java versus Python). Not "CMU CS1" on the transcript.

Transcript. Computer Science. The provider field may say CMU CS Academy.

Bootstrap

Bootstrap puts computing into algebra, data science, physics, and game design. Integration with AP Computer Science Principles is named as possible on opened pages.[186]

Fit. Ages 11–14 and 15–18 when the family already has an algebra or data hour. Parent load follows the host subject: if you can hear a wrong equation, you can hear a wrong expression.

What it is not. Automatically a Computer Science transcript year. It may be Algebra with computing inside it. Title from the work.

Transcript. Computer Science if the year was academic programming they can trace. **Algebra** (or the math noun you actually taught) if the computing was in

[185] Carnegie Mellon University CS Academy, opened pages: Exploring ~40 hours, middle school; CS1 ~180 hours, 8th/9th or high school; 15-111 honors with a named college-credit opportunity; homeschool educator teacher-account path. Hour counts and credit opportunity are the program's sentences. Whether 15-111 is transcriptable for a given homeschooler is a local question this book does not close.

[186] Bootstrap, opened pages: computing into algebra, data science, physics, game design; AP CSP integration named as possible.

service of the math credit. Not both nouns for the same hours unless you split the description honestly.

Python — the language, not a curriculum

Python.org hosts the language and a tutorial. The tutorial is designed for programmers who are new to Python, not for beginners who are new to programming.[187]

Fit. Ages 15–18 inside a CS year, or 11–14 after blocks, *with a curriculum in front of the docs*. Parent load is high if the documentation is the whole program.

What it is not. A homeschool curriculum. Not a transcript title.

Transcript. Computer Science. Never "Python I" unless a college dual-enrollment transcript already uses that name.

LibreOffice, Google Workspace, Microsoft 365

LibreOffice is free Writer, Calc, and Impress on the family computer.[188] Google Docs and Google Workspace are shared documents; the company names AI features on current pages. Microsoft 365 is the suite a house may already own.

Fit. Digital Literacy production. Pick one. Under-13 accounts are a safety object: the parent holds the login.

What it is not. Computer Science. Not NCAA CS. Using a generative button inside a document is still use.

Transcript. Digital Literacy or **Technology Applications** or **Keyboarding**. Never the suite's name.

[187] Python Software Foundation, python.org tutorial: designed for programmers new to Python, not beginners new to programming.

[188] The Document Foundation, LibreOffice: Writer, Calc, Impress, free. Google Workspace / Google Docs and Microsoft 365 named as suites a family may already use. SKU and price lists not used as a source here. NCAA: usage of word processing, spreadsheet, and presentation software does not contribute to an approved computing core course.

Harvard CS50

CS50's OpenCourseWare (cs50.harvard.edu/x/2026 on the page this book opened) is a university course: eleven weeks; C, then Python, then SQL, then web.[189]

Fit. Ages 15–18, post-algebra, as a college-shaped object a motivated student can attempt. Parent load is high. This is **not a homeschool hour**.

What it is not. A year of "Harvard" on the transcript. Not NCAA core by itself. Not AP.

Transcript. Computer Science, or the college's actual name if dual enrollment. Never CS50 as the credit title.

Raspberry Pi Foundation — optional hardware

The Raspberry Pi Foundation publishes optional hardware and named education lines (Computing Curriculum, Ada, Experience CS, Code Club). A small board can be a lovely object. It is not required. One family computer is enough.

Google CS First's shutdown page names Experience CS as a Scratch-based successor for ages 8–14, intended free for 2025–26. This book did not open a live syllabus, so it does not describe lesson counts.[190]

Fit. A family that wants a second, inexpensive machine for tinkering, or a club night. Parent load: you still host safety and the talk after the try.

What it is not. A shopping requirement. Not a moral upgrade. Not a transcript title.

---

[189] Harvard CS50, https://cs50.harvard.edu/x/2026, opened 1 September 2026: university OpenCourseWare; eleven weeks; C then Python then SQL then web. University course, not a homeschool hour.

[190] Raspberry Pi Foundation education lines named on opened pages: Computing Curriculum, Ada, Experience CS, Code Club. Experience CS named on the Google CS First shutdown page as a Scratch-based successor for ages 8–14, intended free for 2025–26. Live syllabus not used as a source; lesson counts not reprinted here.

Retired: Google CS First

Google CS First turned down on **30 June 2025**. Data was deleted.[191] It is not a live free path. If an older sibling's folder still holds CS First projects, those projects may be described as past work. Do not enroll a younger child there as a current course.

Combining honestly

A family may combine. Honest combinations follow the products' own jobs.

CS Unplugged and ScratchJr in the early years, when a recipe-algorithm and a first sprite are the hour. Scratch or Code.org Fundamentals when blocks are the ramp. CMU CS Academy or Bootstrap when a text landing or an algebra hour is already in view. Python docs behind a taught path, not instead of one. LibreOffice or the suite the house already owns for Digital Literacy. CS50 later, if a motivated high-schooler wants a university-shaped object, titled Computer Science.

The child who cannot yet save a file and find it tomorrow is still in a literacy year, whatever the birthday. The high-schooler who cannot yet say what they expected a program to do is still in a first-programs stage, whatever the transcript year. High-school *titles* can still be Computer Science while the description tells the truth about supports.

Combining does not mint a new course title. The transcript still says **Computer Science**, or **Digital Literacy**, or **Keyboarding**. The description may say "block-based introductory programs; unplugged ideas run on a machine; then a Python landing; documents in a free office suite." A stranger can file that. A kitchen nickname is a kitchen nickname.

Game engines and robotics careers are real topics. They get a pointer, not a secret ninth teaching chapter. If the work was academic programming, the title is still Computer Science. If the work was using a tool, the title is still use.

[191] Google CS First shutdown page: turned down 30 June 2025; data deleted. Not a live 2026 course.

What “done enough” looks like

You can say, in one sentence, why this year’s computing stack fits *this* child: parent load you can actually carry, unplugged or blocks or text on purpose, use and science named separately, a high-school credit shape only where the work is a high-school survey, and a machine situation that still leaves one family computer enough.

The programs have names. The transcript has titles a stranger can read. The two lists are not the same list.

A Note on Sources

Studies named in the chapters are listed in Notes, in one series at the back. That is where the full citations live, so the teaching pages can stay a teaching voice.

Some items the research behind this book did not open stay out of the teaching voice. I did not invent a coefficient, a statute, or a product feature to fill a hole. If a study is in the chapter, it is in the notes. If we could not open it, it is not used as a finding here.

Program hours, credit claims, and product descriptions in the resources chapter are taken from the companies’ own pages. Access date 1 September 2026.