Raspberry Pi Foundation

Why kids still need to learn to code in the age of AI

A position paper from the Raspberry Pi Foundation

Philip Colligan, Mark Griffiths, Veronica Cucuiat

June 2025

0. Why do I need to learn this?

It's a question that every teacher needs to be ready to answer. Learning takes time and effort, and, given an already packed curriculum and no shortage of ideas about what else should be added, we need to justify everything that we expect young people to study.

Until recently, the rationale for learning to code seemed clear: it was a pathway to a well-paid job. Tech companies have long bemoaned a shortage of coding skills and there is good evidence that young people who study computer science earn more.[1]

And it wasn't just about access to jobs. Learning to code empowers young people to use technology to solve problems that they care about, to express themselves creatively, and to build businesses that could change the world.

For the past decade, the challenge facing education systems wasn't whether they should be equipping young people with the ability to code, it was how they should go about it.

So has something changed?

The past couple of years have seen huge advances in artificial intelligence (AI), machine learning, robotics, and automation that are already reshaping the way we live, work, learn, and interact. No one really knows how far-reaching these changes will be, but it seems plausible that we are living through one of the most significant waves of technological innovation in history.

One of the fastest developing applications of generative AI has been systems that can generate code. Trained on huge repositories of human-generated code, tools like GitHub's Copilot and Replit's Ghostwriter can now generate increasingly complex code from natural language prompts, ushering in a new era of "vibe-coding".

This is leading many to question whether young people should bother to learn to code and for some to even predict the end of programming as we know it.[2]

So what should we say to the ten year-old who is invited to put together some coding blocks in a computer science class when she asks "Why do I need to learn this?"

In this paper, we put forward five arguments for why we think that kids still need to learn to code in the age of AI.

We argue that even in a world where AI can generate code, we will need skilled human programmers who can think critically, solve problems, and make ethical decisions. Young people need to learn to code because it is the most effective way for them to develop the mental models and fluency to become skilled human programmers.

We argue that learning to code will open up more economic opportunities for young people as the advances in technology expand the range of problems that can be solved through computation. We also make the case for coding as a modern form of literacy that gives young people agency in a world mediated by digital technologies.

And finally, we explain why those who learn to code will be the ones who ultimately shape the future that we all live in and therefore we should open those opportunities to as wide a group as possible.

We do so with humility and in full recognition that we are living through a period of rapid technological change, with all the uncertainty that involves.

But, right now, based on all of the information that we have available, we think that learning to code remains a powerful way of equipping young people with the knowledge, skills, and mindsets to shape the world that they live in.

1. We need humans who are skilled programmers

One of the great things about natural language is that it can cope with ambiguity. We tend to understand each other even when we are a bit loose with our words. So it's not surprising that "code" has often been used as a shorthand for the whole field of computer science or that we say "coding" when we really mean "programming".

We should start with some definitions.

Computer science is the study of how computers work and how we can use them to solve problems. It involves computational thinking and the principles that underpin computation, from how data is represented and processed, to how algorithms are designed and analysed. It also includes topics like networking, web technologies, cyber security, and the social and ethical implications of computing technologies. It is a broad and wide ranging subject that continues to evolve.

Programming is perhaps best understood as the process of formulating a problem in a way that it can be solved by a computer. It sounds simple, but in reality it is a complex and creative process through which human intent is translated into executable computer programs. It is a highly skilled craft that some even describe as a form of art.[3]

Coding is the way that humans give instructions to computers. It is a crucial part of programming, but they aren't the same thing. Programming also involves analysing and understanding problems, identifying and evaluating solutions, designing algorithms, testing and debugging, and much more.

The methods through which humans give instructions to computers have changed dramatically since the first programmers connected switches and wrangled punch cards. Successive waves of innovations have automated different parts of the process, making it easier for humans to get a computer to do their bidding. Compilers, programming languages, operating systems, and libraries are all innovations that use abstraction and automation to make the act of coding easier and quicker.

The latest wave of AI technologies have made it possible for a human to describe what they want to a large language model (LLM) and have their natural language instructions translated into code in the programming language of their choice.

Is this new wave of AI-powered coding tools simply the latest chapter in the history of automation and abstraction? Or has something more fundamental changed?

For sure, there is compelling evidence that these AI systems can dramatically increase the productivity of a skilled human programmer. This seems to be happening in two main ways: by automating routine tasks, helping them achieve what they want to do quicker; and by helping them explore options, suggesting different ways to solve a particular problem that the programmer can evaluate and select.[4]

While the ability of these AI systems to generate code is genuinely impressive (and improving all the time), they haven't automated the process — or art — of programming. The evidence suggests that skilled human programmers will continue to have a critical role to play in translating messy real-world problems into a form that can be solved through computation.[5]

Human programmers are needed to provide effective prompts to AI systems; to critically evaluate the suggestions that they produce[6]; and to use 'systems thinking' to understand how a code component fits into the wider software architecture that exists or which is being built.[7]

LLMs are probabilistic systems designed to generate statistically acceptable outputs. There is no guarantee of accuracy or relevance, and generating more code faster isn't necessarily a good thing. One study analysed 211 million lines of code that had been changed by AI code assistants and identified "multiple signs of eroding code quality" and concluded that developers should "emphasize their still-uniquely human ability to 'simplify' and 'consolidate' code they understand".[8]

While AI-powered coding tools lower the barrier to anyone being able to generate code, it still takes a skilled programmer to know what good quality, safe, and ethical code looks like.

This expertise is important when it comes to applying critical thinking to working with a generative AI system. In one study, knowledge workers with higher confidence in their ability to complete the task without generative AI tools were more likely to apply critical thinking skills when using these tools and evaluating their outputs.[9]

It's also worth remembering that AI systems are trained on repositories of existing code and "will not be able to generate programs that deal with new machine architectures, new programming frameworks, or solve new real-work problems".[10]

For all of those reasons, we will still need humans who are skilled programmers.

2. Learning to code is an essential part of learning to program

In an era of AI-generated code, it's tempting to assume that the hard work of actually writing code is becoming obsolete. Surely we can just teach young people how to vibe code?

The reality is that coding is still the most effective way we know for young people to develop the computational thinking skills that enable them to become an effective programmer.

The fact that code (either graphical or text-based) does not use natural language as an input is arguably a feature and not a bug. The friction introduced in the conversion of human reasoning into a rigid expression of logic is where the learning and development of computational thinking occurs.

An analogy helps make the point here. For good reasons we ask students to write creative stories — even though generative AI can produce compelling narratives — because the act of writing builds literacy. It helps learners engage with language, understand sentence construction, experiment with style, and develop techniques like characterisation and dialogue. These skills can't be fully absorbed by observing a finished product, they emerge through the process of doing.

In the same way, coding is the hands-on practice that allows young people to internalise programming concepts. It enables them to try things out, see what works, and observe the results of their thinking.

Through writing code, a young person builds mental models that allow them to better understand how computers work, their potential, and their constraints.[11] This includes developing an understanding of what types of problems are amenable to computational approaches and how to translate a messy real-world problem into precise computational components. They learn how to read code and critically evaluate it, enabling them to improve both their own and other people's code, including code produced by an AI system.

This reflects the broader body of research on how expertise is developed. For any skill, the process of learning is gradual, with learners acquiring new information, which is then connected and organised into schemas or structured mental models that allow for increasingly sophisticated understanding and application.[12]

In the context of coding, a beginner might first encounter isolated ideas like variables, loops, or conditionals. Over time, with support and repetition, these concepts become connected. Learners might begin to understand how to perform calculations on a list, structure a loop to automate a task, and use conditional logic to guide a program's behaviour. Eventually, they can apply these mental models to solve real-world problems like writing a program to calculate payroll after a tax change.

Crucially, this progression requires deliberate focus and cognitive effort. In terms of learning to code, this includes reading, modifying, writing, and explaining code, and testing it for errors. These activities are what help learners move knowledge from short-term to long-term memory, building the fluency and confidence that define genuine programming expertise. Without these experiences, a deep understanding of programming simply doesn't form.

Alongside that, there is clearly huge potential for AI technologies to transform teaching and learning across all subjects and figuring out effective pedagogical strategies for integrating AI is one of the most important areas of innovation in modern education. There's also an important role for schools in exposing young people to the real-world applications of AI technologies and how they are changing the world, and this should include the tools that are helping programmers write code quicker.[13]

But our job as educators isn't to teach young people to use the tools employed by industry today. The challenge (and opportunity) is how to integrate AI technologies into teaching and learning while ensuring that young people are still developing the foundational knowledge and skills that will enable them to thrive in a world where AI is ubiquitous.

That is why learning to code remains an essential part of learning to program.

3. Learning to code will open up even more opportunities in the age of AI

There's no doubt that AI will reshape the labour market. From transport to the creative industries and professional services, we are already witnessing the early impacts of AI-fuelled automation (replacing tasks) and augmentation (enhancing human capability) across the economy. Programming is no exception.

So what happens if AI is used to generate more and more of the code that humans used to write? Will we just need far fewer programmers?

History suggests not. One of the consistent lessons from previous waves of technology-driven change is that, as the cost of a technology falls, demand for it increases.[14]

As we have already observed, successive waves of innovations have made it easier and more productive for humans to give instructions to computers (compilers, programming languages, operating systems, etc.). By dramatically lowering both barriers and costs, these innovations made it easier to build software, leading to many more problems being solved through computation, spawning whole new industries and economic opportunities.[15]

Generative AI is likely to follow the same pattern. It will undoubtedly change how code is written, but it will also increase the reach of programming (and computational approaches) across the economy and into new domains, creating demand for humans who are skilled programmers.

This is reflected in current labour market predictions. While some lower-level or routine coding roles will decline, according to the U.S. Bureau of Labor Statistics, employment is projected to grow by 18% in software development, 33% in information security, and 36% in data science, between 2023 and 2033.[16]

We're already seeing a greater need for people who can combine programming with domain-specific skills. In the life sciences, innovations in personalisation and co-management of health are increasingly fuelled by wearable devices and huge amounts of data.[17] In modern agriculture, farmers are integrating data captured by sensors, drones, and satellites to optimise crop yields, requiring them to blend traditional knowledge with the ability to interact with complex digital systems.[18] In the social sciences and humanities, computational methods are enabling new kinds of inquiry, from large-scale textual analysis to social network modelling.[19]

We can't predict all the roles that will exist in the future: famously, 60% of the jobs in the US economy in 2018 did not exist in 1940.[20] But we know that digital technologies are becoming more important across every sector of the economy and AI is accelerating that trend.

To ensure young people can access the opportunities these changes will create, they need a foundational understanding of computer science and programming. That is why the OECD's most recent international assessment of what 15-year-olds know and can do is focused on "students' capacity to engage in an iterative process of knowledge building and problem solving using computational tools".[21]

Coding is no longer just for software engineers, it's becoming a core skill that enables people to work effectively and think critically in a world shaped by intelligent machines. As a recent report on human skills for an AI age puts it, young people should aim to develop "the dual capabilities of humans and AI, working together and with humans in the driving seat, rather than a mutually diminishing duel between ourselves and machines".[22]

4. Coding is a literacy that helps young people have agency in a digital world

Just as reading and writing unlocks opportunity, creativity, and understanding, coding gives young people a new way to express themselves, to learn, and to make sense of the world. That is why coding and programming are increasingly recognised as a modern form of literacy, essential whether or not it is a central part of your working life.[23]

Programming allows learners to create with technology, not just consume it. Whether they're designing a game, building an animation, or automating a task, they're not just following instructions, they are bringing their own ideas into the world. This ability to shape the digital world is a powerful form of self-expression and builds confidence that technology is something they can control.[24]

Coding supports learning across the curriculum. It provides a concrete, interactive way to explore abstract ideas, making them more engaging and easier to understand. It allows students to model complex systems, test hypotheses, and interact with data in meaningful ways. From algebra to science and poetry, providing students with the ability to program enables them to "learn about everything else".[25]

This broader view of the role of learning to code — that it isn't just about preparing young people for work — has also been shown to increase participation, making computer science more relevant and accessible to a wider range of young people.

But perhaps most importantly, learning to code is about power.

In a world increasingly mediated by software, algorithms, and automated decision-making, young people who don't understand or interrogate these systems are at a disadvantage. They lack the conceptual framework to ask questions like: What data is being used? What assumptions underpin the model? Can the outcome be contested?

Coding provides the foundation for that understanding, fostering a kind of epistemic agency: the ability to ask how and why a system does what it does. Even a basic understanding of conditionals, loops, and data structures enables learners to see software as constructed and fallible. They are more likely to question a decision made by an AI system, to demand transparency, and to advocate for their rights.

As AI interfaces become more capable of taking natural language instructions, it is tempting to believe that understanding what is happening inside the black box is unnecessary, but that is a dangerous fallacy. The more intuitive the interface, the easier it is to mistake automation for objectivity or accuracy.

Providing young people with a solid grounding in computational literacy developed through coding, helps ensure that they have agency. Without it, they risk being manipulated by systems they don't understand. As Rushkoff said: "Program, or be programmed".[26]

5. The kids who learn to code will shape the future

The idea that "kids don't need to learn to code" carries the real and present danger that we will consolidate the power to shape the world into the hands of an even smaller and less representative group.

Since the earliest days of computing, programming has been championed by educators and computer scientists as a democratising force — empowering anyone to become critically engaged citizens and effective contributors to a digital society.[27] The reality is that access to the opportunities to learn about computer science, programming, and coding has remained deeply unequal, both within and between countries.[28]

That has helped create a technology sector that doesn't reflect the broad diversity of human backgrounds, perspectives, and experiences. And we are all living with the consequences.

While talent is everywhere, the opportunities to develop and apply it are not. Industry and society are missing out on the creativity and entrepreneurialism of significant portions of the population.

Teams with diverse perspectives are more likely to create inclusive designs, identify biases, and build solutions that serve a broad range of users. From facial recognition algorithms to hiring platforms and health tools, we've already seen the consequences of design decisions made without broad participation.

The unprecedented levels of wealth that can accrue to individuals and organisations in the technology sector create powerful second- and third-order effects. A small group of individuals gets to decide where to deploy the capital they have built: which businesses they invest in, which philanthropic causes they fund, and which political campaigns they turbo charge with donations.

That's why the question of who gets to learn to code is so consequential, for everyone.

6. The work we need to do

The good news is that we have made real progress. Over the past decade and more, a global movement of educators, non-profits, philanthropists, businesses, and individuals have been advocating for change and working with policy makers, school systems, and teachers to make it happen.

It is hard work. Across the globe, education systems are under extraordinary pressure. Still recovering from the impact of the COVID pandemic, they are being asked to do more with fewer resources. At the same time, they are expected to deliver the knowledge, skills, and values that will equip the next generation for a world of constant technological, economic, and environmental change.

Time and again we ask teachers to teach a subject that they didn't study themselves and we almost always fail to give them the time and investment in their professional development that they need to succeed.

The risk of the idea that "AI will do the coding for us" is that it suggests that policy makers, school system leaders, and funders should step back from the hard work at just the moment when we need to redouble our efforts.

For all of the reasons we have set out in this paper, we need to challenge the false narrative that AI is removing the need for kids to learn to code. Instead we need to:

- Define a broad and diverse computer science curriculum that embraces AI, but which prioritises the foundational knowledge and skills that will enable young people to thrive in a fast-changing world
- Reimagine computer science as both a standalone subject in its own right and a cross-curricular and interdisciplinary endeavour, integrated into the sciences, mathematics, language, arts, and humanities
- Create more opportunities for hands-on and creative learning, empowering young people to use digital technologies to bring their ideas into the world, building their confidence and agency
- Invest in teachers with high-quality curriculum and classroom resources that are grounded in research and effective pedagogy, and give them the time and support for ongoing professional development
- Mobilise governments, companies, foundations, and philanthropists to increase investment in the programs that ensure that every young person has the opportunity to learn to code

The Raspberry Pi Foundation was founded in 2008 with the purpose to democratise computing, so that every young person could realise their full potential through the power of digital technologies. As we step into the age of AI, that mission feels more relevant and urgent than ever.

Colligan, P., Griffiths, M., Cucuiat, V. (2025) *Why kids still need to learn to code in the age of AI.* Raspberry Pi Foundation.

References

[1] Liu, J., Conrad, C., & Blazar, D. (2024). *Computer science for all? The impact of high school computer science courses on college majors and earnings* (EdWorkingPaper No. 24-904). Annenberg Institute at Brown University. https://doi.org/10.26300/k0w5-pg15

[2] Welsh, M. (2022). *The end of programming*. Communications of the ACM, 66(1), 34–35. https://doi.org/10.1145/3570220

[3] Knuth, D. E. (1974). *Computer programming as an art*. Communications of the ACM, 17(12), 667–673. https://www.cs.tufts.edu/~nr/cs257/archive/don-knuth/as-an-art.pdf

[4] Barke, S., James, M. B., & Polikarpova, N. (2023). *Grounded Copilot: How programmers interact with code-generating models*. Proceedings of the ACM on Programming Languages, 7(OOPSLA1), Article 78, 1–27. https://doi.org/10.1145/3586030

[5] O'Reilly, T. (2025). *The End of Programming as We Know It*. O'Reilly Radar. https://www.oreilly.com/radar/the-end-of-programming-as-we-know-it/

[6] Karaci Deniz, B., Gnanasambandam, C., Harrysson, M., Hussin, A., & Srivastava, S. (2023). *Unleashing developer productivity with generative AI*. McKinsey & Company. https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/unleashing-developer-productivity-with-generative-ai

[7] Du, X., Liu, M., Wang, K., Wang, H., Liu, J., Chen, Y., Feng, J., Sha, C., Peng, X., & Lou, Y. (2024). *Evaluating large language models in class-level code generation*. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE '24)* (pp. 1–13). ACM. https://doi.org/10.1145/3597503.3639219

[8] GitClear. (2025). *AI Copilot code quality: 2025 look back at 12 months of data*.

[9] Lee, H.-P. (H.), Sarkar, A., Tankelevitch, L., Drosos, I., Rintel, S., Banks, R., & Wilson, N. (2025). *The impact of generative AI on critical thinking: Self-reported reductions in cognitive effort and confidence effects from a survey of knowledge workers*. In *CHI Conference on Human Factors in Computing Systems (CHI '25)* (pp. 1–23). Association for Computing Machinery. https://doi.org/10.1145/3706598.3713778

[10] Yellin, D. (2023). *The premature obituary of programming: Why deep learning will not replace programming*. Communications of the ACM, 66(2), 41–44. https://cacm.acm.org/opinion/the-premature-obituary-of-programming/

[11] Kim, D. Y. J. (2023). *Redefining computer science education: Code-centric to natural language programming with AI-based no-code platforms* (arXiv No. 2308.13539). arXiv. https://arxiv.org/abs/2308.13539

[12] Hattie, J., & Yates, G. C. R. (2014). *Visible learning and the science of how we learn*. Routledge.

[13] Denny, P., Prather, J., Becker, B. A., Finnie-Ansley, J., Hellas, A., Leinonen, J., Luxton-Reilly, A., Reeves, B. N., Santos, E. A., & Sarsa, S. (2024). *Computing education in the era of generative AI*. Communications of the ACM, 67(2), 56–67. https://cacm.acm.org/research/computing-education-in-the-era-of-generative-ai/

[14] Allglen. (2025). Jevons paradox: *Why junior developers shouldn't fear AI*. Stackademic. https://blog.stackademic.com/jevons-paradox-why-junior-developers-shouldnt-fear-ai-b7daf86c5590

[15] Cihon, P., & Demirer, M. (2023). *How AI-powered software development may affect labor markets*. Brookings Institution. https://www.brookings.edu/articles/how-ai-powered-software-development-may-affect-labor-markets/

[16] U.S. Bureau of Labor Statistics. (2023). *Fastest growing occupations*. https://www.bls.gov/emp/tables/fastest-growing-occupations.htm

[17] UK BioIndustry Association. (2022). *Life Sciences 2030 Skills Strategy* https://www.bioindustry.org/static/uploaded/3acc684d-f590-4e80-ab90472b1d96dee7.pdf

[18] Trendov, N. M., Varas, S. & Zeng, M. (2019). *Digital technologies in agriculture and rural areas – Status report*. Rome. Licence: cc by-nc-sa 3.0 igo.

[19] University College London. (n.d.). UCL Centre for Digital Humanities. https://www.ucl.ac.uk/digital-humanities/

[20] Autor, D., Chin, C., Salomons, A., & Seegmiller, B. (2022). *New frontiers: The origins and content of new work, 1940–2018*. MIT Department of Economics. https://economics.mit.edu/sites/default/files/2022-11/ACSS-NewFrontiers-20220814.pdf

[21] Organisation for Economic Co-operation and Development. (2024). *PISA 2025 learning in the digital world assessment framework: Second draft*. OECD Publishing. https://www.oecd.org/content/dam/oecd/en/topics/policy-sub-issues/learning-in-the-digital-world/PISA%202025%20Learning%20in%20the%20Digital%20World%20Assessment%20Framework%20-%20Second%20Draft.pdf

[22] Chatfield, T. (2025). *Human skills for an AI age: How today's business schools can develop tomorrow's leaders (White paper)*. Sage. https://doi.org/10.4135/wp254070

[23] Vee, A. (2017). *Coding literacy: How computer programming is changing writing*. MIT Press.

[24] Kafai, Y. B., & Burke, Q. (2014). *Connected code: Why children need to learn programming*. MIT Press. https://doi.org/10.7551/mitpress/9992.001.0001

[25] Guzdial, M. (2022). *Providing students with computational literacy for learning about everything*. In S.-C. Kong & H. Abelson (Eds.), *Computational thinking education in K–12: Artificial intelligence literacy and physical computing* (pp. 29–48). MIT Press. https://doi.org/10.7551/mitpress/13375.003.0005

[26] Rushkoff, D. (2010). *Program or Be Programmed: Ten Commands for a Digital Age*. OR Books.

[27] Vogel, S., Santo, R., & Ching, D. (2017). *Visions of computer science education: Unpacking arguments for and projected impacts of CS4All initiatives*. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 609–614). ACM. https://dl.acm.org/doi/10.1145/3017680.3017755

[28] Denner, J., & Campe, S. (2023). *Equity and inclusion in computer science education: Research on challenges and opportunities*. In S. Sentance, E. Barendsen, & C. Schulte (Eds.), *Computer science education: Perspectives on teaching and learning in school* (pp. 85–100). Bloomsbury Academic. https://doi.org/10.5040/9781350296947.ch-008

Raspberry Pi
Foundation

www.raspberrypi.org

Raspberry Pi Foundation UK registered charity 1129409