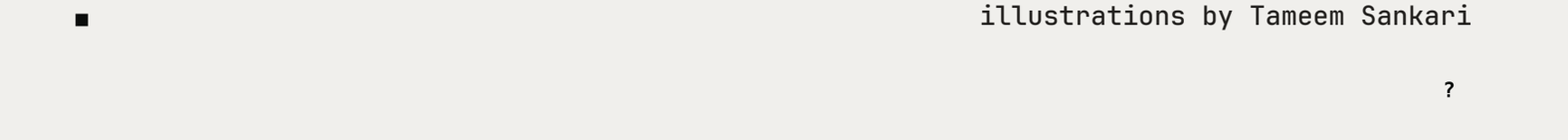
Someone else's code

AI coding agents are overwhelming the internet's unsung human maintainers and eroding the open-source ecosystem that supports our digital lives.

After the vibe-code revolution, who's going to pick up the garbage in the morning?

by Sam Learner



illustrations by Tameem Sankari

If you are reading the digital version of this story, you have just used a small piece of software from a project called cURL. If you are reading it in print, but you have used a smartphone, laptop or tablet today, you have used cURL. If, somehow, you haven't touched a browser or app, but you've used a modern car, printer, TV or gaming device, you've used cURL. You and the majority of humans on Earth use it several times every day, though only a tiny fraction of its users are aware of its existence.

While online, you are constantly requesting packets of information from other machines. But these requests need to follow specific protocols over destination, routing, format, permission; cURL administers these details. First released in 1996, it is now a piece of the invisible plumbing of the internet, small but prevalent, part of the many thousands of lines of code that run constantly in the background of our digital lives. Its maintainers estimate that its component, libcurl, has been installed more than 20 billion times. Even its admirers are rarely conscious of it as they use it, just as it would be impossible always to consider the pumps that make water flow when we open a sink tap.

It is also an open-source software project. The cURL source code is freely available on the software platform GitHub for anyone to see. Its permissive licence guarantees that everyone has "permission to use, copy, modify and distribute this software for any purpose with or without fee". The project also accepts, and depends upon, contributions from a wider community of developers – cURL's website lists more than 3,000 contributors. Its most prolific and consistent contributor is its creator, Daniel Stenberg, a Swedish software developer, who runs the project full time.

"We're not actually owned by anything, we're not owned by any company," Stenberg told me. "We're just a name on the internet pretty much, and a bunch of people trying to make this product as good as possible."

Stenberg is a maintainer, a steward of software that requires constant care and attention. For code to be more than an inscrutable pile of text, it needs to run in an environment – device, browser, application. As those environments change, code must change too. Somebody, whether its creator or others recruited to help, either works to maintain it or sees it die a slow death and leaves its users to deal with the consequences.

Stenberg, aged 55, is fortunate among this guild of maintainers; his project has attracted notoriety, funding and a large pool of eager contributors. Many others carry on unheralded and unpaid. They are the "load-bearing internet people", as open-source pioneer Eric S Raymond called them, those who maintain software critical for the internet to function, without organisational support or a budget. In the complex web of software packages that our digital lives depend on, these LBIP operate behind the scenes to keep things ticking. The webcomic xkcd once poked fun at this precarious dynamic, depicting a tall pile of blocks labelled "all modern digital infrastructure", supported by a single narrow block that "some random person in Nebraska has been thanklessly maintaining since 2003".

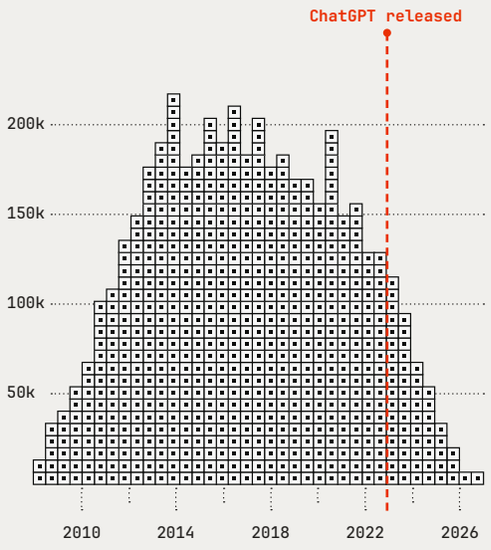
This puts pressure on maintainers, who often act as single points of failure for their projects. "I've been the only full-time person on cURL for a long time," Stenberg said. "All of the others have been volunteers and they tend to come and go over time. I don't know exactly who is going to be around tomorrow and I can't depend on that."

Recently, Stenberg and the load-bearing internet people have faced a new problem. The availability of AI coding tools, which spit out code in response to plain-language prompts, has led to a huge influx of proposed code contributions. Most of them are not very good. Some appear useful, but miss important context or don't function quite right. Some function fine, but are not particularly helpful and still take time to review. These submissions come from people who mean well, and from people chasing the clout of contributing to a foundational piece of software. They come from professional developers, as well as from so-called vibe coders, writing exclusively with AI tools, sometimes without the ability to understand the output.

The influx is clogging the plumbing. For years, cURL operated a "bug bounty" programme, a system that paid outside developers who identified legitimate security flaws. (This is deeply technical stuff; the largest bounty ever paid was $4,660 for a flaw that made "curl overflow a heap-based buffer in the SOCKS5 proxy handshake".) But in January, Stenberg announced that, due to an "explosion of AI slop reports", cURL was ending the programme. He explained that "the never-ending slop submissions take a serious mental toll to manage and sometimes also a long time to debunk. Time and energy that is completely wasted while also hampering our will to live."

HUMANS DON'T ASK HUMANS CODING QUESTIONS ANY MORE

Total monthly questions asked on Stack Overflow's forums



©FT Source: Stack Exchange

Stenberg's experience is notable, but not unique. Across the open-source ecosystem, AI code tools are disrupting a set of norms that has rewarded developers for building or contributing to software that benefits everyone. In the past, contributions to open-source projects were bullet points on a developer's CV. But are you a contributor if your contribution was authored by Claude or Codex? And if the "community of users" for your project is increasingly made up of bots, will you still be motivated to build or maintain it? And if not, then what happens when we open the sink tap?

BY NATURE, DEVELOPERS ARE ALWAYS building on top of other people's work. When you make a sandwich, you don't usually start by baking your own bread. It's the same in software development, where you're never truly starting from scratch. Programming languages and the libraries, packages and frameworks built on top of them are all ultimately the same thing: Someone Else's Code. After all, why solve a problem that somebody else has already solved?

Open-source software is "a precondition for most software to be able to exist", said Stenberg. "In order to work the way we do today, you need a lot of common pieces that you can glue together to build your software... It's not a question of whether it's good or bad. This is now what we have."

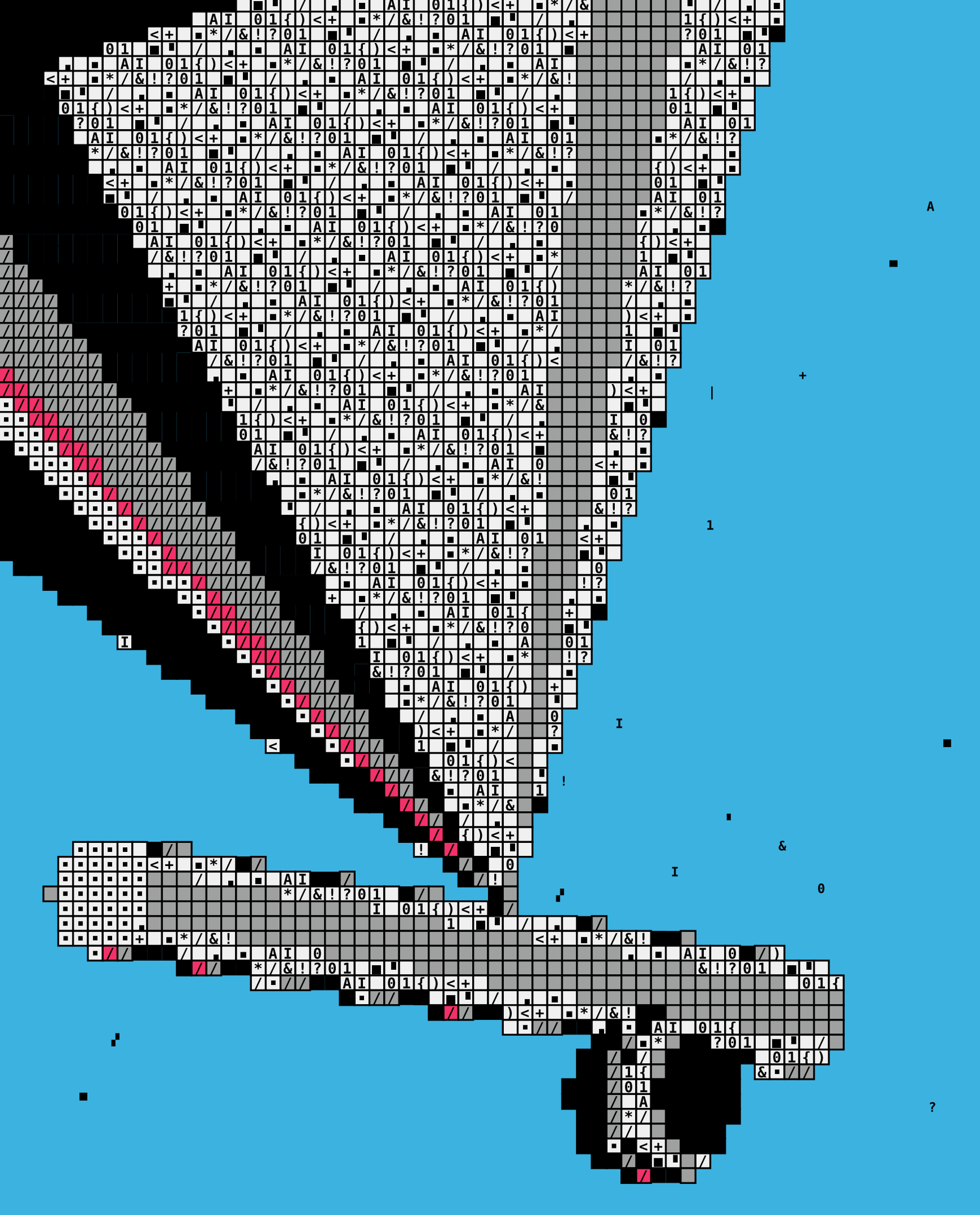
Generally this works well. So well that every piece of software you engage with, whether open-source or proprietary itself, sits on top of a web of open-source packages. The Financial Times app, for example, rests on top of more than 2,000 other packages. One of those is a tool called axios (fetches data), which depends on a tool called babel (translates code), which depends on Node (a form of the language JavaScript), which depends on the open-source language Python. But there's a risk. When you build on top of Someone Else's Code, it becomes a dependency and dependencies can go stale.

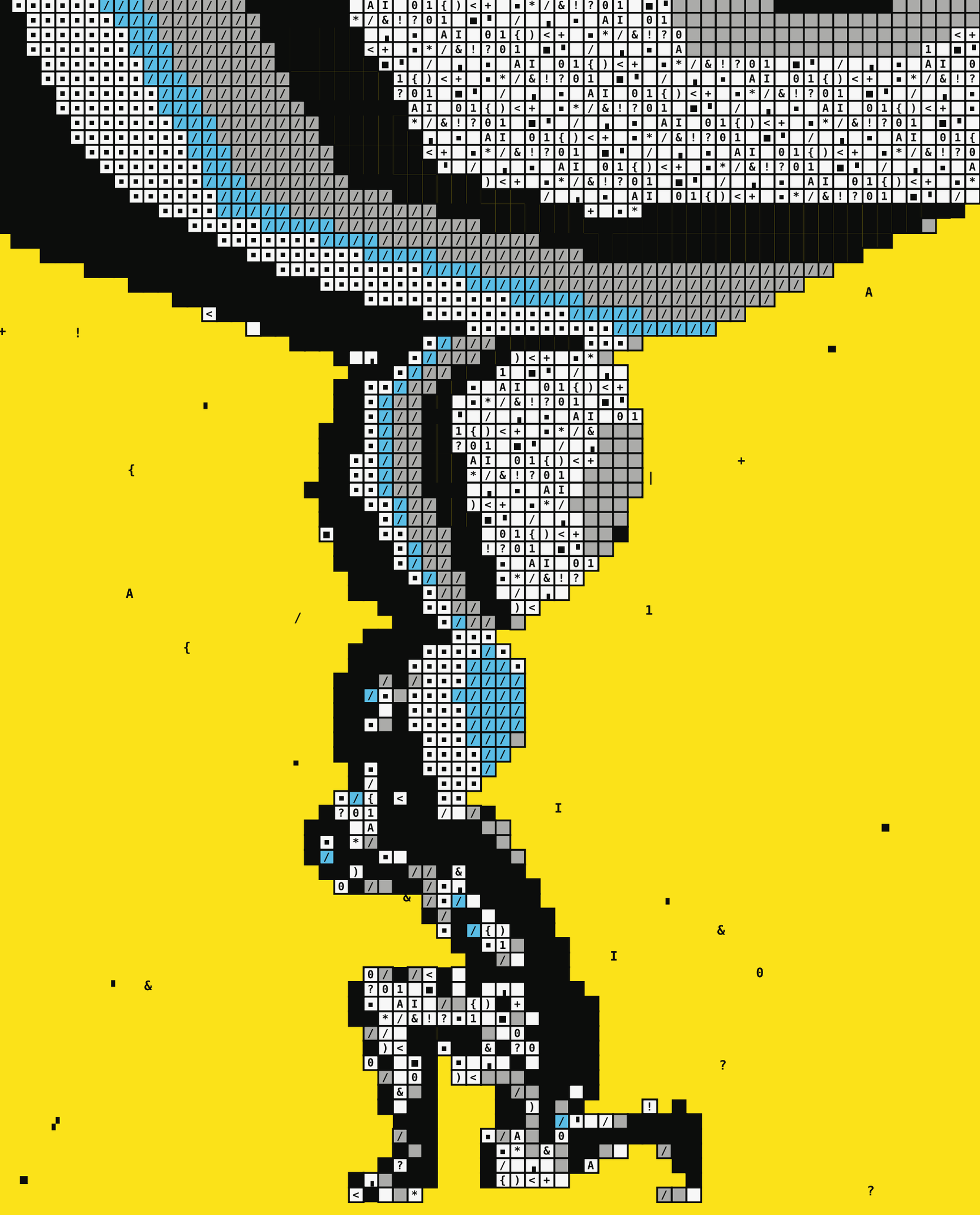
In my career writing code, I've mostly acted as a sandwich maker within the software-as-a-sandwich ecosystem. I build user-facing applications or websites with ingredients provided by the open-source supply chain. I'm burdened by the occasional angry customer when sandwiches stop coming, or sometimes bristle at the way someone uses my recipe.

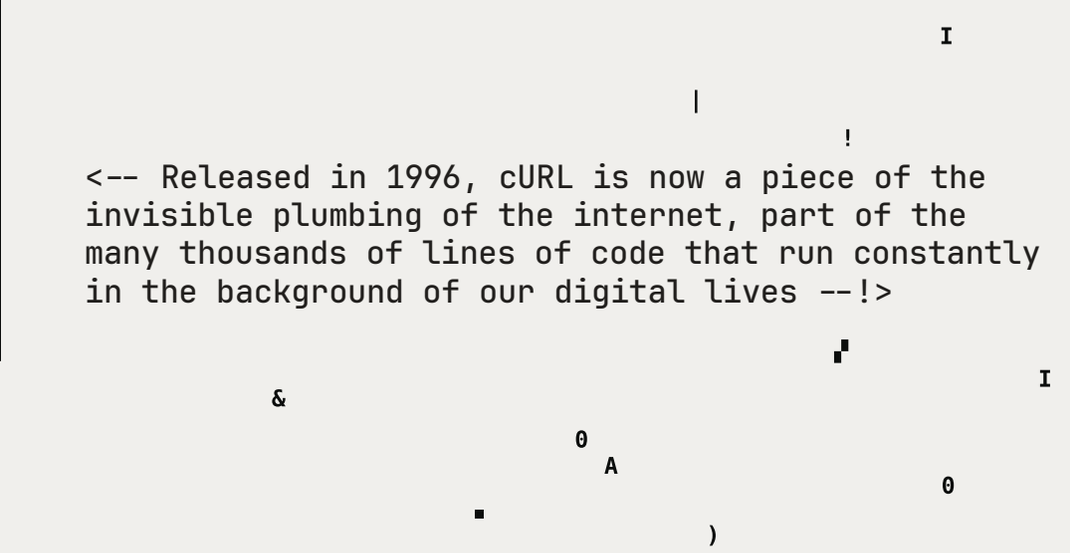
But my contributions and headaches are trivial compared with the bakers who make my bread, not to mention those who invented bread. When something goes wrong with their product, it is the sandwich makers they have to answer to. Many sandwich makers prefer not to think about how the bread is made; they just want it to show up, lest they have to face a crowd of hungry customers who care even less about baking.

Suppose, for example, that a new local rule banned bread baked with certain kinds of wheat – it could cause clashes down the supply chain. The sandwich maker could need a new baker, or the baker could need a new mill, or the mill could need a new farmer.

These "dependency problems" occur constantly, because of new rules, or new recipes, or because an exhausted dough maker simply retires. Maybe their machines keep running for a while, giving their customers time to adjust, but sooner or later







something breaks and there's nobody around to fix it. When that happens, the sandwich makers, many of whom were only vaguely aware of the source of their bread, will have a problem to solve, as they deal with frustrated customers who just want to eat.

If all of this sounds fairly nightmarish, you can understand why maintaining a project like cURL – perhaps a flour mill in this analogy – is a full-time job for Stenberg. This is also why he has relied on a network of skilled volunteers, grants and bug bounties to make sure it's running smoothly.

Now bring AI tools into the kitchen. Anyone can make a sandwich, promise the makers of the sandwich-a-tron. And there's truth to it: many people can now make delicious "vibe" sandwiches without paying a deli. Some deli owners protest that the machine-made sandwiches aren't the same quality, though many buy their own sandwich-a-trons for handling simple sandwiches. Almost everybody agrees that the sandwich-a-tron excels at common recipes and that this will free up time for deli owners to focus on their more artisanal recipes.

But the sandwich-a-tron does some strange things to the delicate supply chain. It orders bread from bakeries that have been closed for years, or that never existed at all. It shows a strong preference for tomatoes from a small farmer in France. The farmer has always answered the phone for customers who made reasonable requests or offered suggestions on harvesting equipment or boxes for shipping. But suddenly he's getting dozens of new requests and complaints every day, suggesting all kinds of pesticides and fertilisers that aren't legal in France. This sudden growth in customers strains the tomato farmer, especially since his compensation comes in the form of donations or goodwill from his community, something that these robo-customers don't offer.

MOST OPEN-SOURCE CODE LIVES ON THE developer platform GitHub. Projects there sit in code repositories, or "repos", which are often public and where multiple developers can contribute simultaneously. Their code is reconciled using an underlying technology called "Git", originally developed by open-source pioneer Linus Torvalds, the creator of the Linux operating system. Usually, anyone on GitHub can flag problems and request improvements through a feature called "issues". Potential collaborators submit draft changes in the form of "pull requests", which are reviewed and approved by a trusted maintainer of the project before being "merged" into the codebase.

Not everyone in the open-source community likes GitHub. Earlier generations of the "free software" movement bristled at the idea of open-source development becoming intertwined with a platform that was for-profit and not itself open-source, a fight that some advocates carry on today. One such organisation is the Software Freedom Conservancy, which protests against GitHub's corporate ownership – it was acquired by Microsoft in 2018 – and its use of code hosted on the site to train AI products, sometimes ignoring the terms of licences attached to projects.

"The internet and everything else is basically built on software that was created with something of an ideological basis… and then all of that work was published online and became extremely useful to companies," Karen Sandler, the conservancy's executive director, told me. "Code is being gobbled up into LLMs [large language models] as training data and then you have a whole body of ideological work which is being labelled as non-copyrighted, potentially to be worked into proprietary software."

Despite these reservations, GitHub today is about as integral to software development as Adobe products are to design work. Alternatives exist, but hosting a project elsewhere generally implies some kind of ideological stance. By and large, for the present generation of developers, GitHub is simply the place where software development happens.

EARLIER THIS YEAR, A GROUP OF researchers led by Miklós Koren, an economist at Central European University in Vienna, released a paper titled "Vibe Coding Kills Open Source". They argue that in a world where the consumers of software packages are increasingly AI coding agents, the incentives that drive the collaborative model of open-source development collapse.

"We started with models that are quite standard in international trade, where firms are producing different quality products and they compete in the market… and we realised it's not a bad analogy for open source," Koren told me. "There might not be a lot of money involved, but every developer wants their package to be popular."

I understand this well. Before working in newsrooms, I developed and released a set of personal open-source projects. One in particular, which visualised the downstream paths of rivers, was a disorientating success. The project is still more widely viewed than every story I've worked on at the FT combined and, judged by eyeball count, might be the most successful thing I ever create. I released it just months before interviewing for my job and, while I developed the tool for free, I can draw a straight line between its success and my employment today.

But Koren found that these socio-economic reasons for developers to contribute to open-source projects degrade when human users are replaced with bots, which install the software without providing positive interactions. In the researchers' model, this outweighs the fact that AI tools may help individual developers write code faster. Over time, the bots erode the open-source ecosystem.

"The question for the developer," Koren said, "is if I want to be popular among humans, why should I write something that is only used by machines?"

In experiments with six coding models, the researchers found that open-source software packages that were frequently recommended by models saw large increases in download numbers – but that activity did not translate into the engagement that typically sustains maintainers. The paper concludes that, "under the traditional business model, where developer revenue depends entirely on direct user engagement, the open-source ecosystem cannot survive widespread AI adoption".

A user base full of bots is one problem introduced by coding agents. Another is that community-driven development itself has become more difficult, as maintainers wade through a swamp of AI-generated contributions to their projects.

The maintainers I spoke with agreed that AI contributions were frequently useless or irrelevant. Would-be contributors using AI tools are more likely to make "extractive contributions", where the time and energy required for maintainers to review them outweigh their value. AI coding agents have dramatically reduced the cost of authoring an extractive contribution, but reviewing one still requires a human touch. Given these increasing demands on their time, many open-source maintainers have grappled with how, and if, they should be handling outside contributions in the AI era.

Steve Ruiz has decided to shut them out entirely. He's the creator of tldraw, a popular digital whiteboard tool. Facing a wall of low-quality, AI-generated code to review, he announced that decision in January. In a blog post, Ruiz posed a philosophical question about the future of collaborative development: "If writing the code is the easy part, why would I want someone else to write it?"

If everybody is using the same tools for writing code anyway, what is the value of collaboration, especially with those who don't know their way around your project? In the past, a new contributor would gradually build knowledge about your project and hopefully one day become a trusted maintainer. But if they simply point a tool at it, there's no reason to believe that accepting their contribution will lead to any growth in their knowledge of, or investment in, your project.

And why should other developers, maintainers and potential employers assign any value to those contributions? If contributions are the currency of the open-source community, they are deflating to the point of being worthless. "The social contract and social practices around an open-source contribution are just obliterated," Ruiz said.

Many maintainers told me that they found coding tools useful for some components of maintenance, such as parsing large codebases. But they are tools that facilitate collaboration with

<-- If the 'community of users' for your project is increasingly made up of bots, will you still be motivated to build or maintain it? And if not, then what happens when we open the sink tap? --!>

a machine, not collaboration among humans. To the extent that the sandwich-a-tron can recreate sandwiches in my deli's style, it's better that I'm the one operating it. If I bring in someone new to push the machine's buttons, they will never really learn to make my sandwiches. And then who is going to take over the deli?

GUIDO VAN ROSSUM, THE CREATOR OF THE Python programming language, has left the deli before. After releasing the language in the early 1990s, he assumed the title "benevolent dictator for life", a tongue-in-cheek moniker that established him as the final decision maker on the project, a system of governance that many open-source projects have adopted since. But two decades as BDFL involved making sometimes controversial decisions, the stress of which wore on him. The fallout over a new syntax feature, the infamous "walrus operator", was the final straw that led him to step down from the position in 2018.

"Maintainers are under a lot of pressure," van Rossum told me. "Every project ends up sort of having an emergency meeting where they make decisions on what to do with the endless flow of AI slop." Besides their volume, he noted that large language models have a tendency to touch many parts of a file not directly related to the problem that they are trying to solve, making each review more tedious.

Van Rossum thinks that many people submitting low-effort slop are motivated simply by the idea of writing "Python contributor" on their CV. (By some measures, Python is the most popular language in the world, including among coding models.) He noted, however, that for prominent projects like his, "there has always been a lot of motivation, even before LLMs, to give fly-by contributions that weren't of high value. The number of people who just ran a spellchecker over documentation and turned the output into a pull request is also quite embarrassing."

In many ways, fully automated AI-generated contributions are no different than these spellchecker contributions: shortcuts for producing low-value changes that seek to reap the rewards of participating in a communal project without meaningfully improving it.

JOSH COMEAU IS A DEVELOPER AND EDU-cator based in Montreal. After working in several tech roles, including a stint with online education giant Khan Academy, he began offering his own web-development courses in 2020. But when he released a new course this year, enrolments were only a third of the level of previous launches. His peers in technical education were experiencing similar declines.

"It takes me about two years to make a course," Comeau told me. "If I start work on my fourth course today, it's not going to be ready until late 2027 and who knows if there's going to be an audience." If the current level was the new baseline "that would still actually be worth doing, but it's hard to know if you've hit a new floor or not".

Comeau worries about a generation of developers whose coding education is supplanted by interactions with LLMs, which often try to solve problems directly and narrowly, rather than by providing broader context.

"AIs will know how to answer the question that you have, but you don't know what questions you should be asking that you're not asking," he said. The lack of a bigger-picture understanding "seems like a serious skill deficit that's going to become a problem when the current generation of developers, who learnt before these tools existed, start to retire or move on".

Comeau thinks of coding with AI as driving on the highway with cruise control. The car will keep moving forward, but still requires a driver behind the wheel. This is how he squares stories of significant productivity gains from experienced developers with countless frustrated accounts from novice developers who, while trying to use an AI tool to build a piece of software, hit a wall that they can't get past.

This aligns with one of the conclusions of a November 2025 paper that sought to identify code committed to GitHub as human-written or AI-generated, and to examine trends in AI adoption. The researchers noted that while AI eased developers' expansion into new domains, "experienced programmers capture nearly all of these productivity and exploration gains, widening rather than closing the skill gap".

"To learn how to code well, to be able to verify the outputs of a large language model, you need to understand something about the code," Johannes Wachs, one of the paper's authors, told me. He said that when he teaches first-semester programming courses, his students learn material that would be very easy for LLMs. "I don't envy the students," he said. "They have this homework assignment and there's a red button at the end of the desk: 'Press this and your pain goes away.'"

From a distance, cruise control might look like a self-driving car. A world without drivers will be a problem if self-driving cars turn out to be elusive. But if you think that the technology is around the corner, why spend time learning to drive?

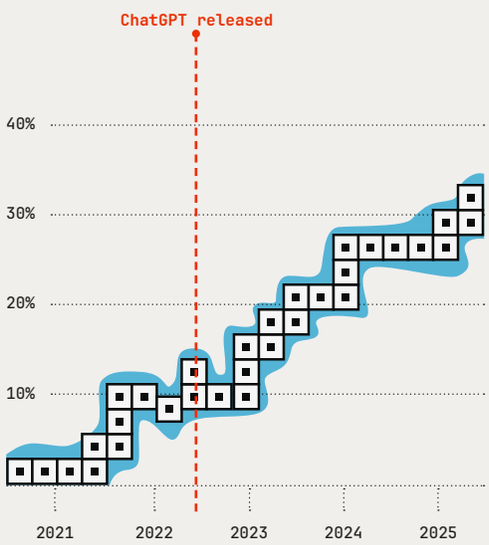
Complicating things further, many of the non-AI resources that students and developers turned to in the past are disappearing. The same factors shrinking the audience for Comeau's courses also make it less worthwhile to write a blog post or participate on a forum.

Perhaps most notably, Stack Overflow, the once-ubiquitous question-and-answer site for programmers, has seen its activity decline to nearly nothing since the launch of modern chatbots. The forum was where developers turned when they were stuck on a problem. Just before the launch of ChatGPT in 2022, there were more than 100,000 monthly questions asked on the site; last month, there were fewer than 1,500. And the response rate, once around 80 per cent, has been cut in half. A Stack Overflow post was often rich with discussion and written by someone who was equally stuck. The posts' volume and variety were staggering. In 15 years of writing code, it's difficult for me to remember a time when its forums didn't provide context for an issue I was trying to solve, if not an outright solution.

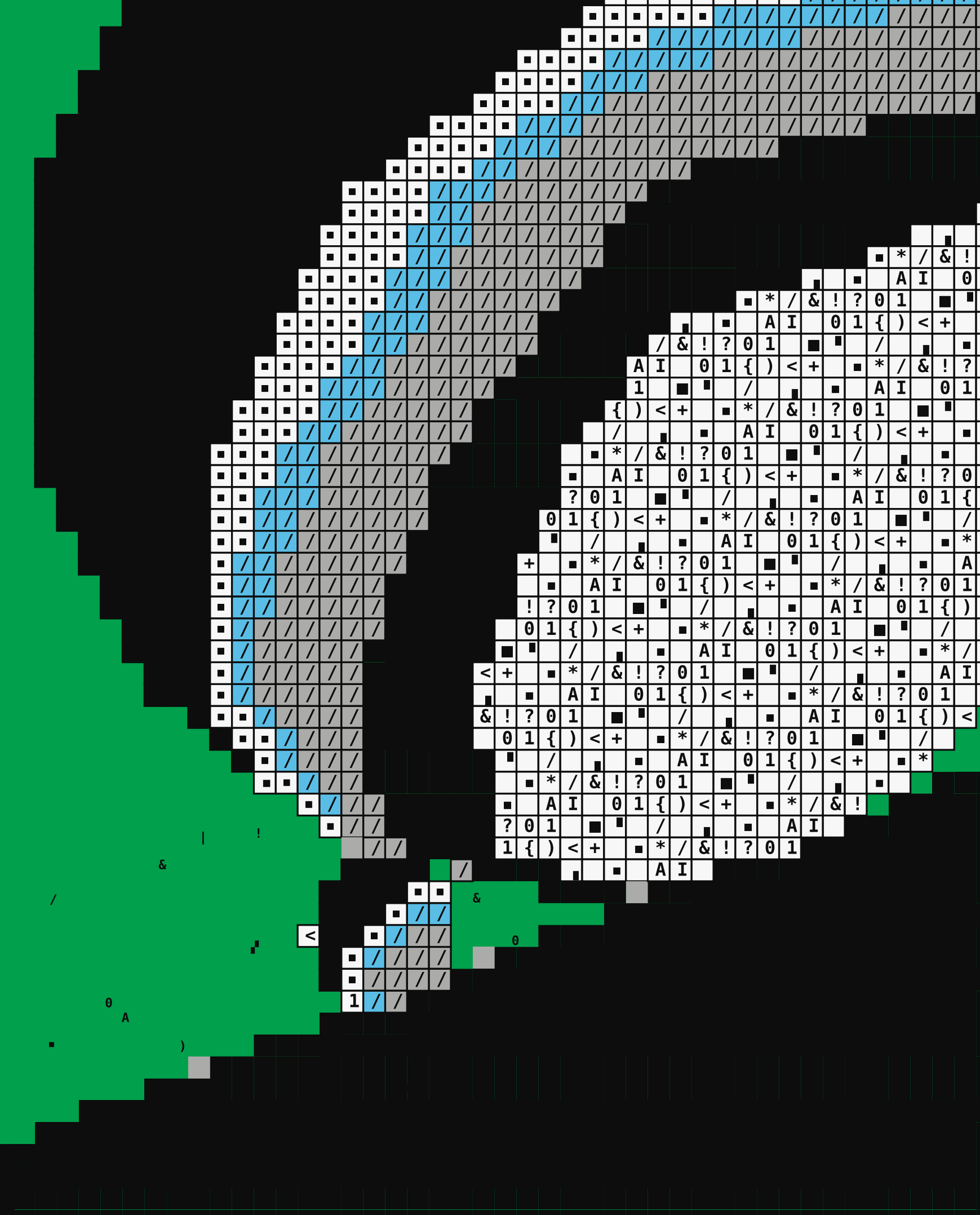
This rich context also made it incredibly useful training data for the AI models that have replaced it. While there are orders of magnitude more lines of code on GitHub than on Stack Overflow, the accompanying code documentation is often sparse. So in the same way that Reddit forums have had an outsized influence on the everyday advice or answers given by LLMs, two decades of Stack Overflow posts form the backbone of models' programming advice.

CODEBASES ARE INCREASINGLY FULL OF AI-GENERATED CODE

Estimated % of Python code committed by US GitHub users that is AI-generated



©FT Source: Daniotti, Wachs, Feng and Neffke, 2025. The researchers used a classifier model to identify code written with substantial AI assistance in over 30 million contributions; shaded area is a 95% confidence interval



<-- It's easier to write code than ever before. But there's a lot more to software development than writing code, and that culture is now under strain from AI tools trained on their output --!>

A paper co-authored by Johannes Wachs in 2023 analysed the forum's decline and questioned whether, by crowding out their own future training data, models would become less effective from Stack Overflow's absence. Now he believes that this was the wrong way of thinking about the problem. "There is still human-interaction data being generated, but that's private now," he said. "OpenAI and Anthropic learn a lot from their users."

Stack Overflow's public forums are dead, replaced by private chatbot conversations. "We have to accept that software development has changed completely, you understand, right?" Prashanth Chandrasekar, the site's chief executive, told me. In June, Stack Overflow announced an answer service for coding bots. Its human archives are still useful, but they grow staler every day, as the tools and problems of software development continue to change. It is no longer a place where you can expect to find answers. Instead those answers live in the private, conversational data that belongs to large tech companies.

"UNTIL VERY RECENTLY," RICH HARRIS told me, "the idea that you would effectively pay rent to a Silicon Valley company for the privilege of being able to write software would have been considered completely absurd."

Harris is the creator of an open-source web-development framework called Svelte. He developed Svelte while at The New York Times, where he worked as a graphics editor for nearly a decade. Svelte is a direct competitor to React, the framework giant built and maintained by Meta, and it commands a cult-like following and an active set of contributors. These days, Harris maintains the project with the support of the cloud platform Vercel, which has employed him since 2021.

Harris is concerned that software developers could soon be stuck with a different kind of "dependency" problem. Even if the "self-driving cars" do arrive soon, they'll be owned by a few tech giants who charge a lot of money for them. You may still wish you had learnt to drive so that you didn't need to pay a taxi to take you a few minutes down a quiet, straight road.

"One of the beautiful things about software development is that it's completely permissionless," Harris said. "Anyone can do it, and the tools that we use to create software are themselves fully available open-source software. Now, suddenly, it's completely normal to have to pay $100 or $200 per month in order to do software development."

Harris hopes that, in the future, open-source AI models will be able to compete with frontier subscription models like Claude and Codex. But he acknowledges that the technology is not yet there to run on consumer-grade machines. "The biggest risk for me would be if we can't achieve that transition," he said.

Every developer I spoke with wondered whether AI coding agents were simply another layer of abstraction in a field with a long history of abstracting away details. Developers rarely touch assembly language directly any more, the low-level code that sits just above the 0s and 1s. Languages built on top of it, such as C, handled many tedious details, and languages built on C, like van Rossum's Python, smoothed things down even further. Perhaps prompting an LLM is just another step on this ramp of abstraction.

To me, it feels like more than that. Computer science classes or programming interviews often have students write in something called "pseudocode", a style of capturing the logic of a program without getting bogged down in a particular syntax. When I guide an LLM through a familiar task, I feel like I'm instructing it with loose pseudocode. But when I venture into a task that I don't know well, when I am tempted to press the "red button" that Wachs described, it feels different. Instead of a programmer, I become a product manager – or even a customer – of Someone Else's Code. Features like power steering take us further from the road, but we're still driving. When are we no longer driving?

I WONDERED HOW OPEN-SOURCE advocates, some of whom bristle at the idea of code even being hosted on a for-profit platform, felt about code itself being written by large tech companies' products at the cost of a monthly fee. Karen Sandler, at least, was more conflicted than I expected.

"If free-software developers don't use these tools at all, the pace of free-software development might be slower than that of proprietary software, and we'll fall behind," she said. While some in the community support outright bans on AI-generated code, others feel that the cat is out of the bag. "We have to use these tools, so how do we use them ethically?" Sandler said.

"There is something that has the potential to be extremely democratising about these tools," she continued. "The goal was to have freedom with respect to your software. What's cool about these tools is they have the chance to allow anybody to be able to modify the software that they rely on."

This is a core irony. It's easier to write code than ever before, and experts can work more quickly. But there's a lot more to software development than writing code, and the culture that built the software we depend on is under strain from AI tools trained on their output. The public and communal nature of open-source software is not quixotic; it is an asset crucial to its success. Everything now suggests a system that is more private and centralised, where software is built more in conjunction with a tech company's product than with each other.

"In general, writing code the first time was never the problem for any project," says cURL's Stenberg. "The challenge for any project is maintaining it over time, fixing bugs over time. If you don't understand it, you have to rely on the AI to fix all of the problems, going forward."

He continued: "They are not that good at fixing the problems; they're much better at finding the problems."

A paper published this March surveyed recent developer attitudes from a wide array of forum discussions. "Our findings frame AI slop as a tragedy of the commons," the researchers concluded, "where individual productivity gains externalise costs on to reviewers, maintainers and the broader community."

NEARLY 60 YEARS AGO, THE ARTIST Mierle Laderman Ukeles wrote an essay titled "Manifesto for Maintenance Art 1969!". She argued that the work of maintenance is often unappreciated and devalued – "a drag; it takes all the fucking time" – next to the more celebrated work of creation. "After the revolution, who's going to pick up the garbage on Monday morning?" asked Ukeles, who has served as the artist in residence for the New York City Department of Sanitation since 1977. She argued that society "confers lousy status" on those who do maintenance.

This is perhaps truer than ever in the era of the "creator economy" and tech industrialists who burn and build and move fast and break things. To the "creators" go wealth and prestige while the maintainers work in the background, scraping rust and justifying their existence to a society that takes them for granted.

Add to this a generation of generative AI tools designed and marketed as instruments of creation. These tools have made it trivial to make images, write novels, build software and pollute the internet with slop. Rarely has a tech CEO touted an AI tool that will "supercharge maintenance". Maintenance is careful work and not easily shortcut.

The creators have unleashed another weapon on the maintainers: turning loose legions of people to build new buildings and bridges without any thought to architecture or city planning. The slapdash builders construct monstrosities and leave them to rot. Worse, they show up at carefully planned structures and offer improvements in such numbers that the truly important maintenance work gets drowned out.

It is easier to destroy than to create, and it is easier to create than to maintain. Too often, we don't appreciate maintenance until something breaks, or the maintainers until they step away. FT

Sam Learner is a graphics journalist on the FT's visual storytelling team