Should I still learn how to code?

# 2024-02-29

[Cassie KozyrkovCassie Kozyrkov • 1.• 1.CEO at Data Scientific, Google's first Chief Decision Scientist, Decision Advisor, Keynote Speaker (makecassietalk.com), LinkedIn Top VoiceCEO at Data Scientific, Google's first Chief Decision Scientist, Decision Advisor, Keynote Speaker (makecassietalk.com), LinkedIn Top Voice](https://www.linkedin.com/in/kozyrkov?miniProfileUrn=urn%3Ali%3Afs_miniProfile%3AACoAAAGgwQ8BQtrd2DWgF9p17g_qyzt7zSA0kzU)

[22 Std. • Bearbeitet • 22 Std. • Bearbeitet •](https://www.linkedin.com/in/kozyrkov?miniProfileUrn=urn%3Ali%3Afs_miniProfile%3AACoAAAGgwQ8BQtrd2DWgF9p17g_qyzt7zSA0kzU" \t "_self)

Should you still learn to code?  
  
... 🤔 ... 🤔 🤔 🤔  
  
Okay, some context!  
  
This week, NVIDIA’s CEO [Jensen Huang](https://www.linkedin.com/in/ACoAABMznFkB_XPgkYHnUl33kIHTWt1DtMAV6Pg) seems to be popping out of every corner of our social media feeds with this bombshell: kids no longer need to learn programming. He says that instead, AI will enable everyone to code in human language.  
  
Am I yay or nay on this? Let's take a beat. I like to stir the pot with a nice contrarian "neither" ...so here goes:  
  
I'm taking us back to the 11th century for this one. Omar Khayyam, a brilliant mathematician of the time, tried and failed to find the roots to the general cubic equation—a simple task for most modern high schoolers.  
  
Why did he fail? Because he was writing it out in paragraphs rather than symbolic notation. It was too loose; too muddled. It lacked precision. Today's innocent little equation  
  
x^3+bx=c  
  
would have been expressed by dear Omar as "a cube and a number of sides are equal to a number."  
  
Ew, no wonder it took centuries to make progress. Over time, mathematics ended up condensing down to a less expressive, but more precise, language.  
  
We face the same problem with code today. For inspiration and creative tasks, prompt engineering in English may be enough. Let me say that another way... if the quality of your results doesn't matter to you, don't bother with precise expressions. But look me in the eye and tell me you don't care about control of the quality of your results. Go on. I dare you. Tell me your mission-critical engineering project that changes lives at scale can afford a little shrug and mumble.  
  
Didn't think so. If you require accuracy and precision from your model, your code will have to become more precise as well.  
  
Programming is the ability to know what you want a machine to do and then to express those instructions clearly. Ideally, you also make it easy for collaborators to build on it without wanting to gauge their eyes out.  
  
Programming has always been about instruction-giving. I’ve long held that most programming languages are unnecessarily unpleasant to use. That will change. The need for precise thinking and precise expression won't. So, Jensen is right in that someone else's precise programming language is pretty annoying to learn and maybe you won't need to suffer through it. He's wrong in that tomorrow's language of programming will be the same one you use when hanging out with your buddies. It won't. It'll be yours, but it'll be the most precise way you know to express yourself, bounded by the gravity of rules that prevent an explosion of ambiguous meanings on your journey to bringing your vision to life.  
  
TL;DR: If you're interested in building, you should still learn to code. But not as an exercise in learning a particular language. Learn to think and speak precisely, it's a hard skill but one that'll never go out of fashion.