Hi, I'm edunham, and this is a crash course on skills that you can use to turn technical expertise into personal and professional success.

Who here prefers computers over people?

Don't be shy, I can't be the only one here.

But a key tech skill is communication.

Software is a thing that people make for other people to use.

Writing code is communication with your users.
They might be people, or other programs... Which are also written by people!

One communication skill is to look for the overlap between what someone else wants, and what you want.

That takes some empathy, and it's the essence of negotiation.

Negotiation is where you explore how you and someone else can build a thing that meets both your needs -- whether that thing is a job description, a salary, a deadline, or the specs of a new product.

Never be afraid to negotiate.

Also, here's your daily reminder that it's ok to ask for help. Your time is valuable. Don't waste it on problems that a quick question or search could easily solve.

Most people like talking about themselves, and enjoy giving advice.

I also value advice from people I don't want to emulate, because it helps me understand how they got where they are.

While others can be helpful, you’re ultimately responsible for your own wellbeing.

Wearing yourself down by tolerating a toxic environment, or overworking yourself into exhaustion and burnout, won't accomplish any of your goals.

So stand up for yourself.

Your boss's job is to make sure your team delivers its product.

Your company's job is to accomplish its mission, which probably involves making money.

And none of these things are easy. The real world plays by the rules of physics and sociology and economics, and those rules don't let everybody succeed all the time.

While schoolwork is often about learning to get the same answers as everyone else, most work in the industry is about building something new or doing something differently.

In other words, you’re searching through all possible solutions to a problem to figure out which ones will work.

You'll find a lot of solutions that look like they should work, but don't.

That's ok.

The smarter and faster you search through possible solutions, the sooner you'll find something that works.

Find the tools let you skip to the interesting and important parts of a problem.

Find an editor that works for you, and customize it how you need. Keep your code in version control. Learn to learn new languages, and pick the best for each task.

Avoid insulting the tools that others choose. Instead, ask how that tool won the competition for their attention.

To avoid doing same year of work 20 times in a row, you've got to quit doing things once they get easy for you. But they've still gotta get done. Delegate to a person, or write a bot.

When you’re doing things that challenge you, you'll sometimes get stuck. Learn strategies to get unstuck. I like to try approaches from other fields to see what new questions they uncover.

Attack your problem with a technique from an art or a sport or a game, and see whether it helps or where it gets you. Isn't that smart?

It’s ok to value “being smart”, but learn the difference between a growth mindset and a fixed minset. Carol Dweck's TED talks explain it great.

And finally, I'm sure all of you here will encounter problems that nobody has ever seen before.

Please share what you learn from them.

If everyone helps teach others, we'll all be rewarded with a better industry to work in.

Thanks to everybody who shared their advice for this talk, and special thanks to the mentors who've helped shape my career.