This is why the moment you enter the door and change to a new environment is the most crucial. If at this moment, you tell yourself that you are just going to do 20 minutes of coding practice, you will most likely succeed and use your inertia to end up learning for an hour or more. No brain will perceive a 20-minute task as a lot of effort and you end up tricking your brain to take advantage of your evening.

Of course, it’s unlikely that at the beginning, you'll be able to code up Clash of Clans or League of Legends. But you’ll be able to make something interesting. It could be a dice game or a flash-card app. But as long at the end of the tutorial, you’ll have made something you can use and play with, then you’ll be far more motivated to code to the end.

A programming language is simply a tool. It is no different from any other tool in your hardware box. If you want to hammer a nail, you should be using a hammer. If you want to fix your water pipes, you’ll probably need a spanner.

The key to learning to code is all about ramping. You want to be stretched over and over again and for knowledge to be built on previous knowledge. If that ramp is too steep, you’ll get lost. If that ramp is too shallow, you’ll get bored. The right gradient is different for everyone. That’s why we encourage students to use the speed change functionality liberally on our tutorials.

The skill you need to hone is in asking good questions and understanding the answer. There is no point in copy-pasting code from a StackOverflow answer if you have no clue how it works. Because StackOverflow works on a reputation system, it’s in their interest to be as clear as possible in their answer to be marked as correct and collect upvotes.

Lacking in ideas? Be a copycat. Make your own notepad, make your own MSPaint, make your own piano. If you’re into games, make minesweeper, make Tetris, make Flappy Bird. Not only will they be sort of useful, but they’ll also be the perfect opportunity for you to figure out how to do things and get experience in finding help.

You need accountability and commitment to learning. Think back to your university days, would have bothered to finish that essay at 3 AM if nothing depended on it? Would you have gone to any of the lectures if you didn’t care about passing or failing?

What I mean is programming will keep evolving. In order to stay relevant, you have to keep reinventing yourself. The world will keep moving, if you stay in one place, you’ll eventually be left behind.

Do you have a bug in your code that you can’t work out? Sleep on it, play foosball, go for a walk. In 9 out of 10 cases, the solution will become apparent. In the remaining 1 out of 10 cases, you’re just screwed.

Pair programming is an agile software development technique that’s based on very similar principles. For example, a learner and a mentor would sit down at the same workstation and work on a problem. The learner is in charge of writing code and the mentor reviews the code line-by-line as they are written.

The more that you break down problems and define the issue that you’re trying to solve, the easier it is to package your code into bite-sized chunks. The simpler the chunk, the easier it is to tackle.

That’s why there’s a rule in programming that says “never copy paste code that you don’t understand”. So what should you do when you’re confronted with a block of code that solves your problem but you have no clue how it works? Break it down.