10 Rules for Learning to Code [cheatsheet-version]

You can find the original (a *longer*, more detailed) version here: https://brupy.notion.site/10-rules-for-learning-to-code

✅ Rule #1 – The 30-Minute Rule

- Don't aim for hours, aim for **30 minutes a day**.
- Getting started is the hardest part — once you begin, momentum will carry you.
- Build a streak: mark an "X" on the calendar for every day you code.
- Protect the chain. Your brain hates breaking it.

Consistency > intensity. Small wins daily add up.

✅ Rule #2 – Code to *Create*, Not Just to *Learn*

- Don't just copy code from tutorials — build something.
- Motivation comes from *making*, not memorizing.
- Even small projects (dice game, flashcards, timer app) are more fun than solving math problems you don't care about.
- Projects = practice + creativity + something you can actually use.

Code is most powerful when it creates.

✅ Rule #3 – There's No "Perfect" Programming Language (But Python Comes Close)

- No "best language" exists — only the right tool for the job.

- But Python is a **Swiss Army knife**:
  - Websites → Django / Flask
  - Data Science → Pandas, NumPy
  - AI & ML → TensorFlow, PyTorch
  - Automation → requests, BeautifulSoup
  - Hardware → Raspberry Pi, MicroPython
- Concepts you learn in Python transfer to ANY other language.

Start with Python. Build anything. Level up anywhere.

✅ Rule #4 – Beware of the "Draw an Owl" Tutorial

- Avoid tutorials that skip steps or throw in "simply" without explaining.
- Copy-pasting code without understanding teaches nothing.
- If a tutorial feels way over your head → it's not you, it's the tutorial.
- Learning should feel like climbing a **ramp**: steady, not too steep, not too flat.

Good tutorials stretch you, not crush you.

✅ Rule #5 – Everyone Uses Google (and That's Okay)

- Feeling like you're not "good enough" = **imposter syndrome** (70% of people feel it).
- Even Google, Amazon, and Microsoft engineers use **Stack Overflow, Google**, & **AI** daily.
- Your real skill is **problem-solving** and knowing **how to ask good questions** — not memorizing 10,000 functions.
- Debug first → then Google → then Stack Overflow.
- Learn from the answer instead of just copy-pasting.
- Learn how to properly utilize AI for code checking and debugging.

Great coders aren't walking encyclopedias. They're great thinkers.

✅ Rule #6 – Be a Copycat

- Don't just ***read*** books — **build things**.
- Stuck on ideas? Copy **existing apps & games.**
- Copycats teach you how things ***actually*** **work**.
- **The best part**: you'll find tons of help online since others have built them too.

Copy first. Create later. That's how you learn.

✅ Rule #7 – Get Into the Habit of Chunking

- Big ideas feel overwhelming → break them into **small chunks** called **modules**.
- Solve one chunk at a time → they add up to the full project.

- The **simpler** the chunk, the **easier** it is to code.
- Chunking turns impossible ideas into **step-by-step progress**.

Don't build the whole robot. Build one function at a time.

✅ Rule #8 – Get a Mentor (Like Me 😉)

- A mentor **accelerates** your growth like nothing else.
- Good mentors don't just give answers — they teach you how to think.
- With my course, **you're not learning alone**:
  - Access our private **Discord server**.
  - Get help, ask questions, and connect with peers.
  - I'll be there to guide you too.
- Learn, share, teach → grow faster together.

Information is cheap. A mentor shows you how to *think*.

✅ Rule #9 – Learn to Walk Away From Your Code

- Coding ≠ nonstop typing. It's mostly thinking.
- Staring at your screen *is* working.
- Stuck on a bug? Step away. Sleep on it. Walk it off.
- 9 times out of 10, the solution shows up when you're not forcing it.
- Code less, think more. Bad code written in a rush = painful refactoring later.

Sometimes the smartest move is to stop coding.

✅ Rule #10 – Play the Game, Don't Just Run the Drills

- Fundamentals matter, but projects that exercise those fundamentals are where the real learning happens.
- Start building from **Day 1** — no project is "too early."
- Projects boost confidence, pride, self-esteem, and creativity.
- Use **Git/GitHub** to track your growth and showcase your work.
- Keeping all your projects organized and well documented makes you a better candidate.
- Employers make decisions based more-so on an applicants Portfolio than anything else.

Drills teach you skills. Projects let you play the game.
