

ChatGPT for Coding: A Beginner's Cheat Sheet

Use ChatGPT as a coding helper, not an unquestioned authority. Ask for small steps, check the result, and stay in control of your code.

Start with a clear request

Tell ChatGPT what you are building, the programming language, what should go in, what should come out, and any rules it must follow. Mention your experience level and ask for an explanation if you need one. Clear details make the answer easier to check.

A useful first prompt

I am learning Python. Write a small function that converts Celsius to Fahrenheit. Include one example call, explain the formula in plain English, and do not use external libraries.

A simple working loop

1. Describe one small task and the result you expect.
2. Read the code and ask questions about anything unclear.
3. Run it in a safe environment, check the output, then adjust one thing at a time.

Generate code in small steps

Build a little, run a little, and keep each change understandable.

Ask for one useful piece at a time

A request like 'build my whole app' leaves too many decisions open. Start with a single function, screen, or small change. State what it should do, what it must not do, and how you will know it works. Add complexity only after the simple version passes your checks.

Example: a tiny Python function

```
def double(value):  
    return value * 2  
  
print(double(4)) # Expected: 8
```

This is intentionally small: one input, one operation, and one visible expected result. Ask ChatGPT to explain what each line does, then run the example yourself.

A prompt you can adapt

In Python, write a function that [specific task]. It should accept [input] and return [output]. Keep it beginner-friendly, use only the standard library, and show two test examples including one edge case. Explain any assumptions before giving the code.

Iterate without losing track

- Ask for the smallest change that solves the next problem.
- Keep a copy of working code before trying a larger edit.
- If the answer is too advanced, ask for a simpler version and a short explanation of each new idea.

Debug and understand code

Give useful evidence, then ask for reasoning you can follow.

When something breaks

Share the exact error message, the relevant short section of code, what you expected, and what happened instead. Include the language and version if you know them. Remove passwords, keys, personal data, and private information first. Ask ChatGPT to explain the likely cause and suggest one minimal fix at a time.

Debugging prompt

This Python code should [expected behaviour], but it [actual behaviour]. The error is [paste a redacted error]. Explain the likely cause in plain English, then suggest the smallest safe change. Do not rewrite unrelated parts.

Ask for an explanation, not just an answer

- Ask what each unfamiliar line or keyword does.
- Ask what assumptions the code makes about its input.
- Ask for a small example with the input, steps, and output shown.
- If the explanation is confusing, ask for a simpler analogy or a step-by-step trace.

Check the fix

A confident explanation can still be wrong. Compare the proposed change with your original goal, run the code, and check both an ordinary input and an edge case. If the error remains, share the new result and continue from the latest evidence rather than applying several guesses at once.

Test, protect, and check the details

Good coding habits matter more than a polished answer.

Before you use generated code

- Review every line. Do not run code you cannot explain, especially commands that delete files, change permissions, or access the internet.
- Test in a safe, separate environment with sample data. Back up important work first.
- Check the output, test likely edge cases, and ask someone experienced to review important or production code.
- Treat answers as suggestions. ChatGPT can produce code that looks plausible but is incorrect, outdated, or insecure.

Keep secrets and private data out

Never paste passwords, API keys, access tokens, private keys, confidential source code, or personal information into a chat. Replace sensitive values with clearly labelled placeholders, such as `API_KEY_HERE`. If a secret has already been shared, follow the service's guidance to revoke or rotate it.

Dependencies and licences

Before installing a suggested package, check that it is genuine, maintained, compatible with your project, and obtained from a trusted source. Review its documentation and security notices. Code and dependencies can have licence conditions: check the licence for any copied code and each package, and get appropriate advice before using them in commercial or distributed projects. ChatGPT's answer is not legal advice.

Remember: ask, inspect, test.

Use ChatGPT to learn and explore. You remain responsible for the code you choose to run and share.