

Hugging Face and Local LLMs

You've probably heard about the large language models (LLMs) provided by big companies – ChatGPT, Copilot, Claude, etc. But there are many models being designed by smaller companies or individuals too.

[Hugging Face](#) is a platform where developers can share models they've created. These models can be downloaded onto your computer and run locally through the terminal or in an interface like [LM Studio](#).



When you run a LLM locally, you get to control how/when it is used and ensure that your data stays on your computer. But local LLMs may be less powerful and take longer than the biggest models since they have limited resources (the processing power of your computer).

Unit 2: Modern CS Techniques

Programming With AI

15-110 – Monday 09/28

Announcements

- Hw4 revisions due **tomorrow noon**
- Exam1 this Wednesday
 - Remember – **different location than normal lectures.**
 - Lecture 1 (2pm) – CUC McConomy
 - Lecture 2 (4pm) – DH 2315

Learning Goals

- Recognize the processes used by **generative artificial intelligence** to produce new text or media based on a prompt.
- **Prompt** and **iterate** using AI tools to create a Python programs.
- **Critique** code generated by an AI tool to identify possible places for improvement.

Reminder:

100% deterministic
Same code, same input == same output every time

Python Code:

```
1 3 + 5
2
```

Compiler/
Interpreter

```
push    rbp
mov     rbp, rsp
sub     rsp, 16
mov     eax, 3
add     eax, 5
mov     DWORD PTR [rbp-4], eax
leave
ret
```

```
101110000000000011
000000000000000000
0000000000000000101
000001010000000000
000000000000000000
```

GenAI
writes
code

You write
the prompt

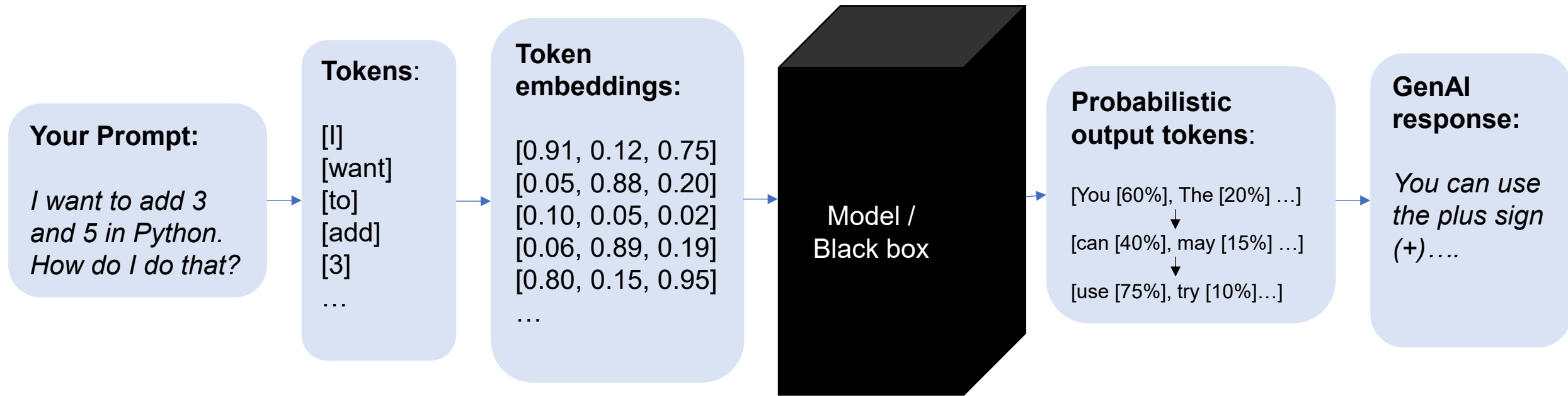
*"I want to add 3 and
5 in Python. How do
I do that?"*

**GenAI is non-deterministic! The same prompt,
run multiple times, even on the same model,
can produce different results.**



Human Intent:
Add 3 and 5

Reminder: Why are GenAIs non-deterministic?



- GenAI does not think the way we do; it predicts the next best word based on patterns in its enormous training text

AI Changes How We Program

Programming is no longer about writing every line yourself. The process of programming has changed to:

- **communicating** to GenAI what you need
- **understanding** what it produces
- **evaluating** and **improving** it

Think of yourself as a manager, and GenAI as an eager junior assistant.

- It is fast and knowledgeable, but easily confused and error-prone
- It needs your judgment and guidance, and often, clarification in small, bite-sized steps

Our goal is to work with GenAI while still thinking like a programmer.

Let's do an experiment:

To see some of the capabilities and limitations of GenAI, we will go to a GenAI and write the following prompt:

“Write a program that calculates the average of my grades”



vague, unclear prompt

Rules for Coding with AI

Rule 1: Be specific

With Yourself

- Know exactly what the code should do before you start
- Think through the inputs/parameters: how many, their types, what they mean
- Think through the output/return value
- Plan the algorithm at a high level, in plain English

With the AI

- Vague prompt = vague result. Be precise, clear, and as detailed as possible
- State your assumptions so the model doesn't have to guess
- You'd do the same with your junior assistant!

Rule 2: Read, trace, understand

You cannot evaluate AI code if you don't understand what it's supposed to do.

Carefully **read** what the AI produces; **understand** it, and **trace** how it works.

- Walk through the code line by line to see what each part is actually doing.
- Trace execution with print statements to confirm the code does what you think it does.

If something's unclear, ask the AI to explain it, or turn to other resources like your course materials or Python documentation.

Never use code you don't understand!

Rule 3: Test, verify, iterate

Run the code and test it with `assert` to verify it works as intended.

- Test with common, general inputs
- Test with edge cases and unique inputs

If the code doesn't work, use the AI to iterate and refine the solution.

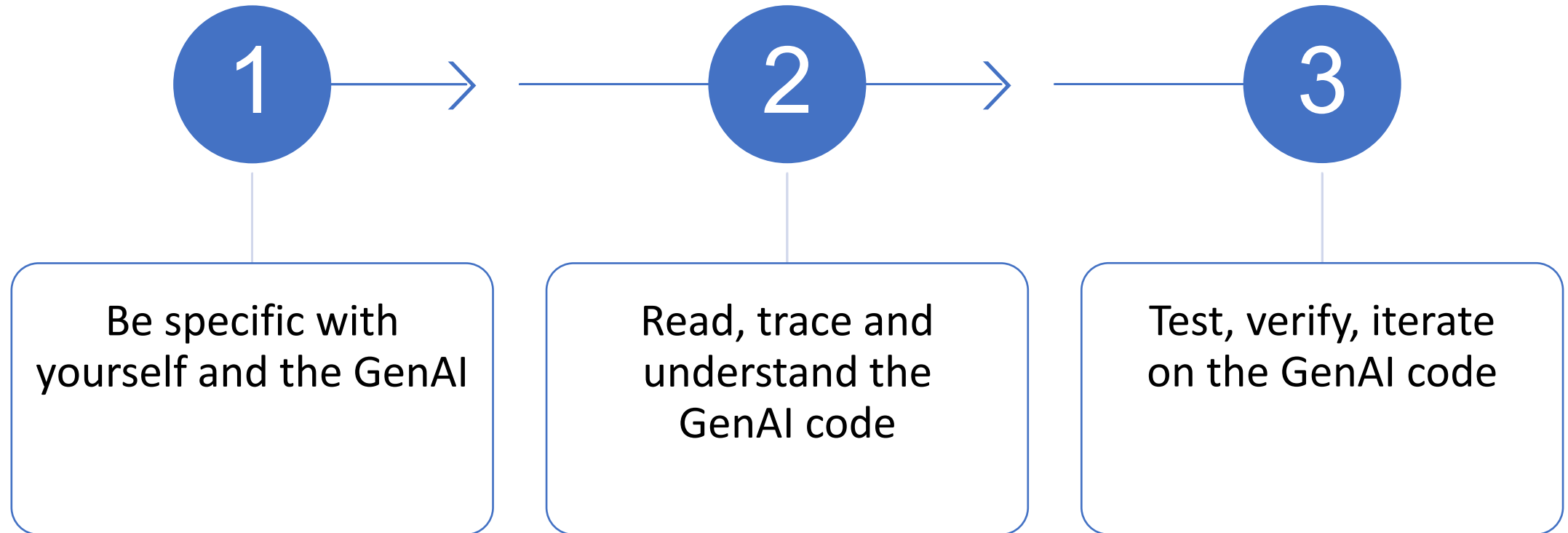
- Again, be as clear as possible what you want it to fix.

However, sometimes the AI won't be able to fix the problem

- You'll need to step in yourself with print statements or a debugger, which is exactly why understanding the code matters.

You are always the one responsible for correctness!

Summary of Rules



Let's practice:

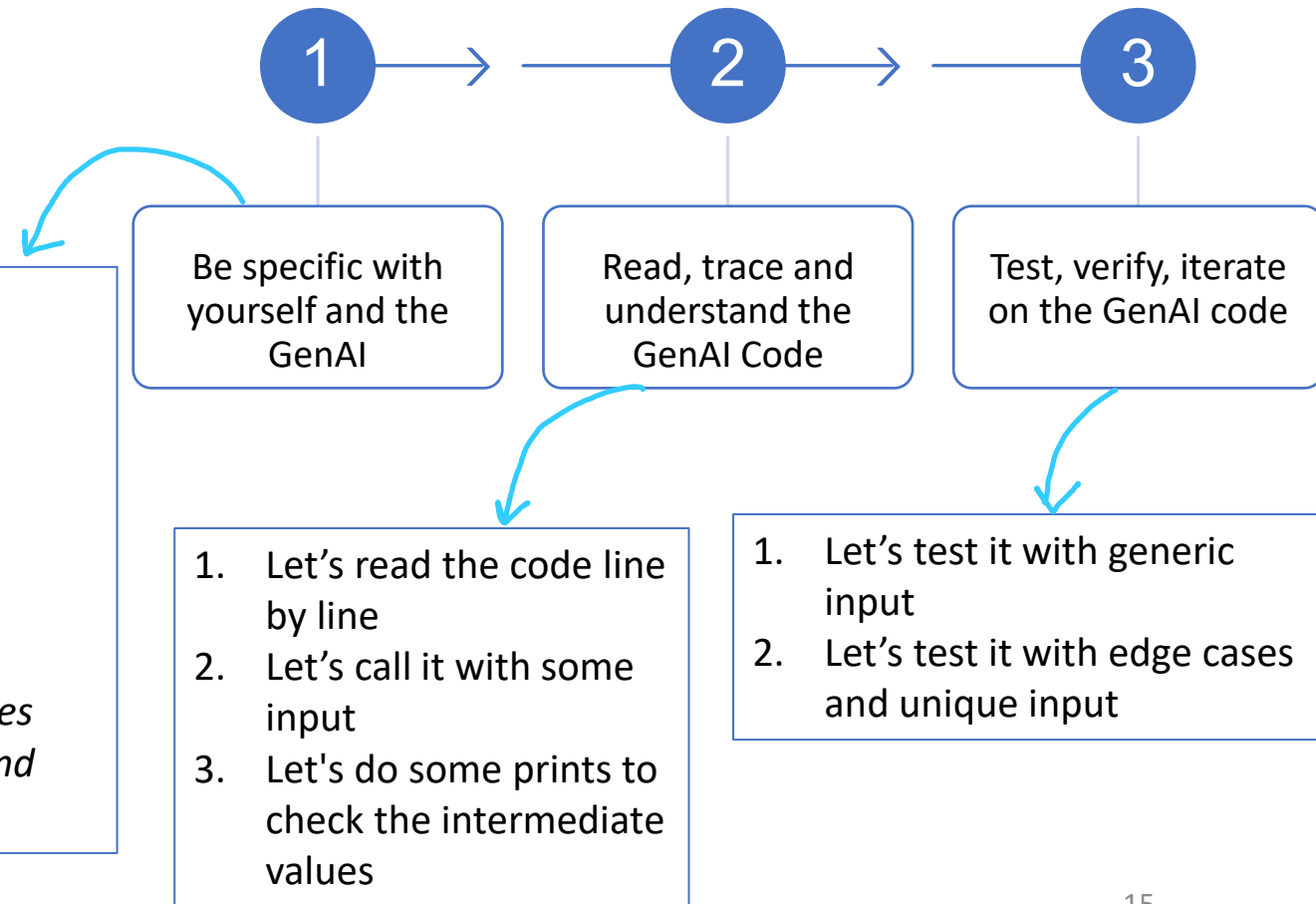
Today, we'll focus on one function at a time, using small exercises.

Next week, we'll shift to working with bigger, more complex projects.

Example 1:

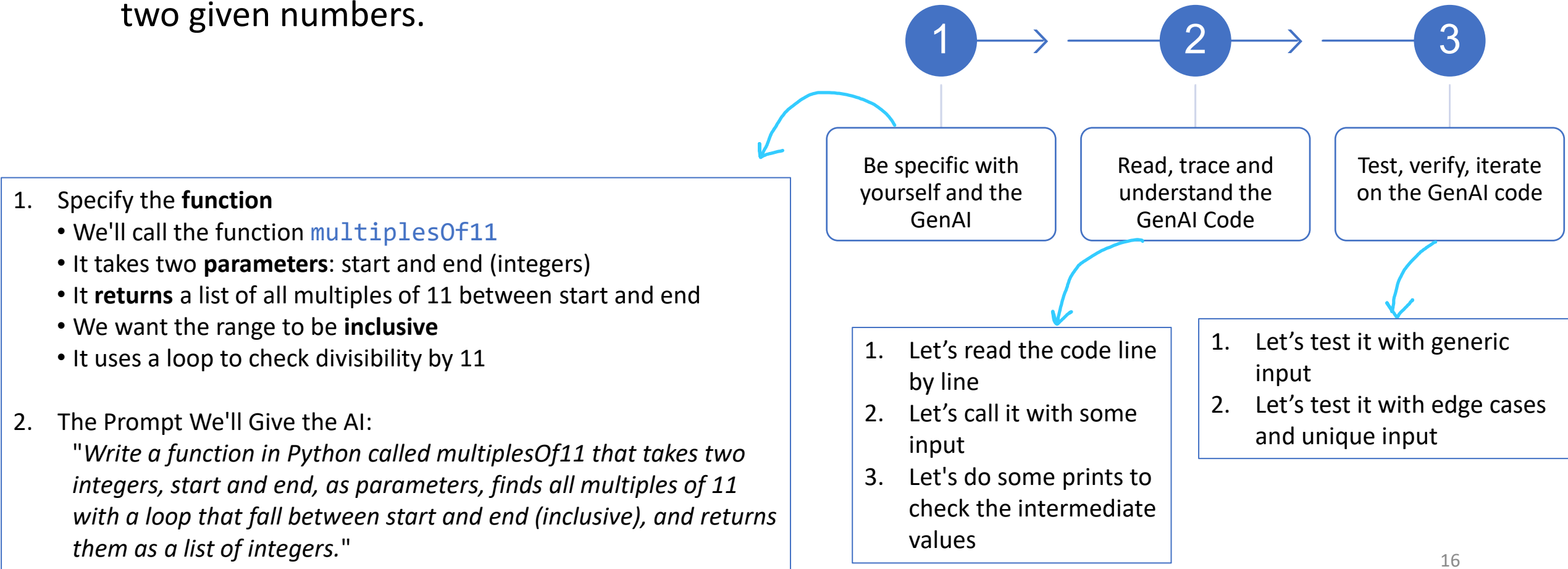
We want to create a function that will convert temperatures from Celsius to Fahrenheit.

1. Specify the **function**:
 - We'll call the function `convertToF`
 - It takes one **parameter**: the temperature in Celsius, as a float
 - It **returns** the temperature in Fahrenheit, as a float
 - It uses the standard Celsius-to-Fahrenheit conversion formula
2. The **prompt** for the GenAI:
"Write a function in Python called convertToF that takes a float representing the temperature in Celsius as a parameter, calculates the conversion of this temperature from Celsius to Fahrenheit, and returns the temperature in Fahrenheit as a float."



Example 2:

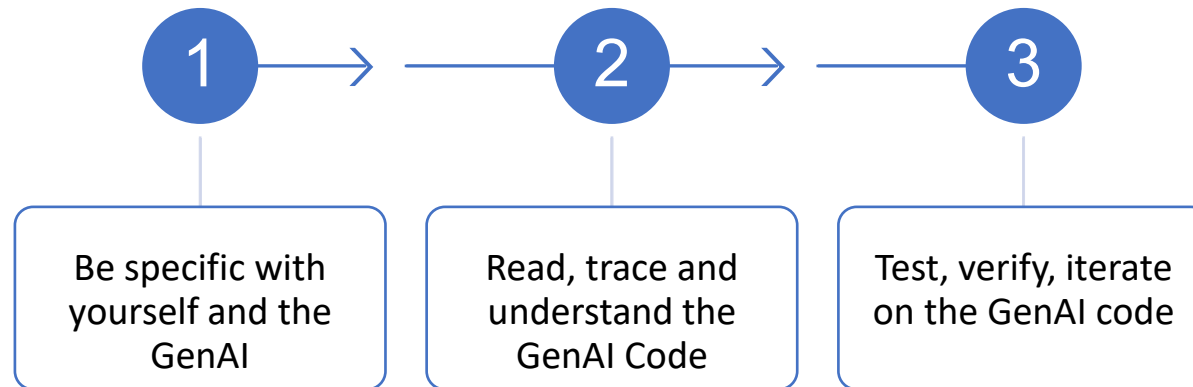
Find all multiples of 11 between two given numbers.



You try:

We want to add all the even numbers between 2 given numbers.

For example, between 10 and 21 we have $10+12+14+16+18+20 = 90$



Example 3: Let's do together

Given a list of strings, remove all words that start with the letter "a".

1. Specify the **function**

- We'll call the function `removeAWords`
- It takes one **parameter**: words, a list of strings
- It **returns** a new list containing only the words that do not start with the letter "a" (case-insensitive)
- It does not modify the original list

• The Prompt We'll Give the AI:

"Write a function in Python called removeAWords that takes a list of strings called words as a parameter, removes all words that start with the letter 'a' (case-insensitive), and returns a new list containing the remaining words, without modifying the original list."

1
Be specific with yourself and the GenAI

2
Read, trace and understand the GenAI Code

3
Test, verify, iterate on the GenAI code

1. Let's read the code line by line
2. Let's call it with some input
3. Let's do some prints to check the intermediate values

1. Let's test it with generic input
2. Let's test it with edge cases and unique input

GenAI Use In Industry

Coding with AI at scale

Different models for different jobs

- Many GenAI systems exist, and while the underlying architecture is largely similar, each is trained differently, giving it distinct strengths.
- Most providers also offer multiple models within their own lineup.
- Industry teams pick the model suited to the task at hand: deep reasoning and architecture design, optimized for fast code completion, multimodal/documentation-heavy work, etc

Multi-Agent Systems

In industry, it's common to run multiple AI agents in parallel, each handling a different part of a project. Once each agent's output is verified, the pieces are merged into a single, cohesive result.

Agentic Coding

An AI plans, writes, tests, and fixes code across many steps with minimal human supervision, functioning more like a junior engineer delegated a task than a simple autocomplete tool.

IDE-Integrated AI

Modern IDEs have AI built directly into the tool, offering inline suggestions as you type, and agent modes that can edit multiple files at once.

The AI is right there in your workflow, and companies often invest heavily in training these models on their own codebase and data.

But the core process stays the same: read, understand, and test.

Worth exploring on your own, though not required for this class; the browser-based AI we use is fine for now.

Learning Goals

- Recognize the processes used by **generative artificial intelligence** to produce new text or media based on a prompt.
- **Prompt** and **iterate** using AI tools to create a Python programs.
- **Critique** code generated by an AI tool to identify possible places for improvement.