

# The AI-assisted coding paradigm

AI-ASSISTED CODING FOR DEVELOPERS



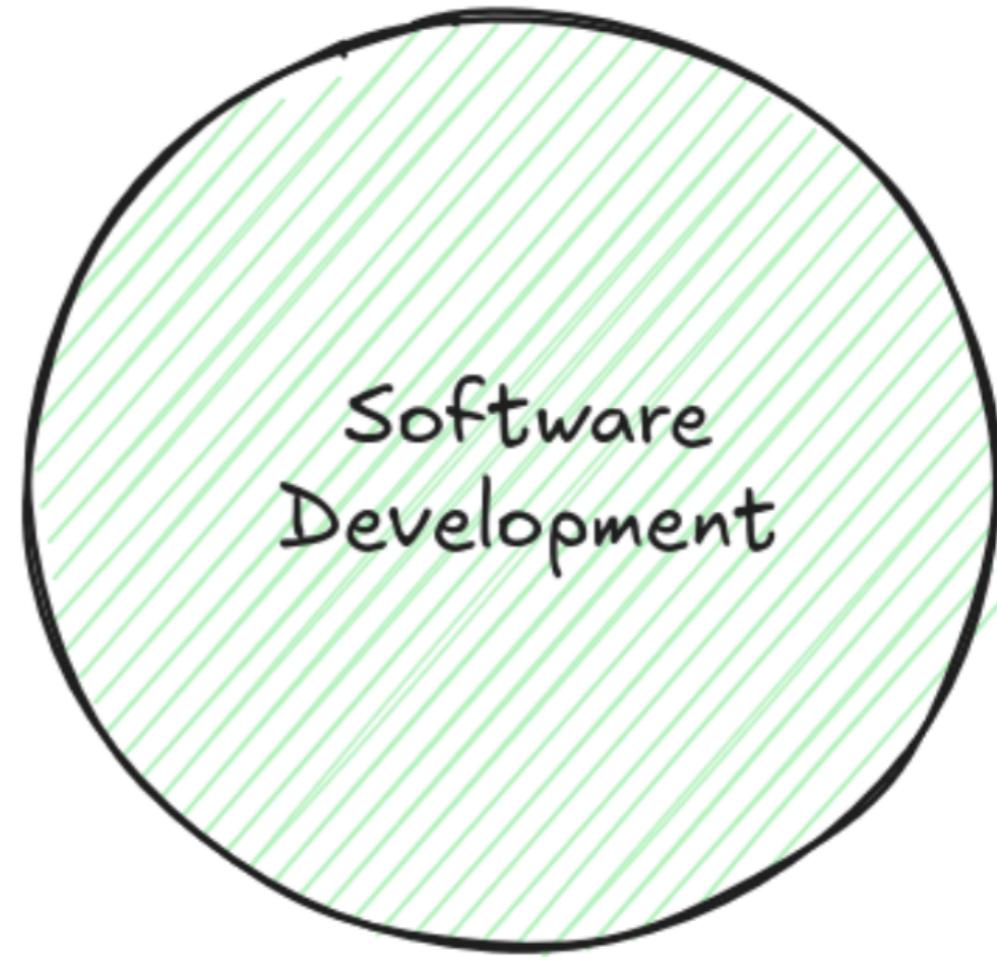
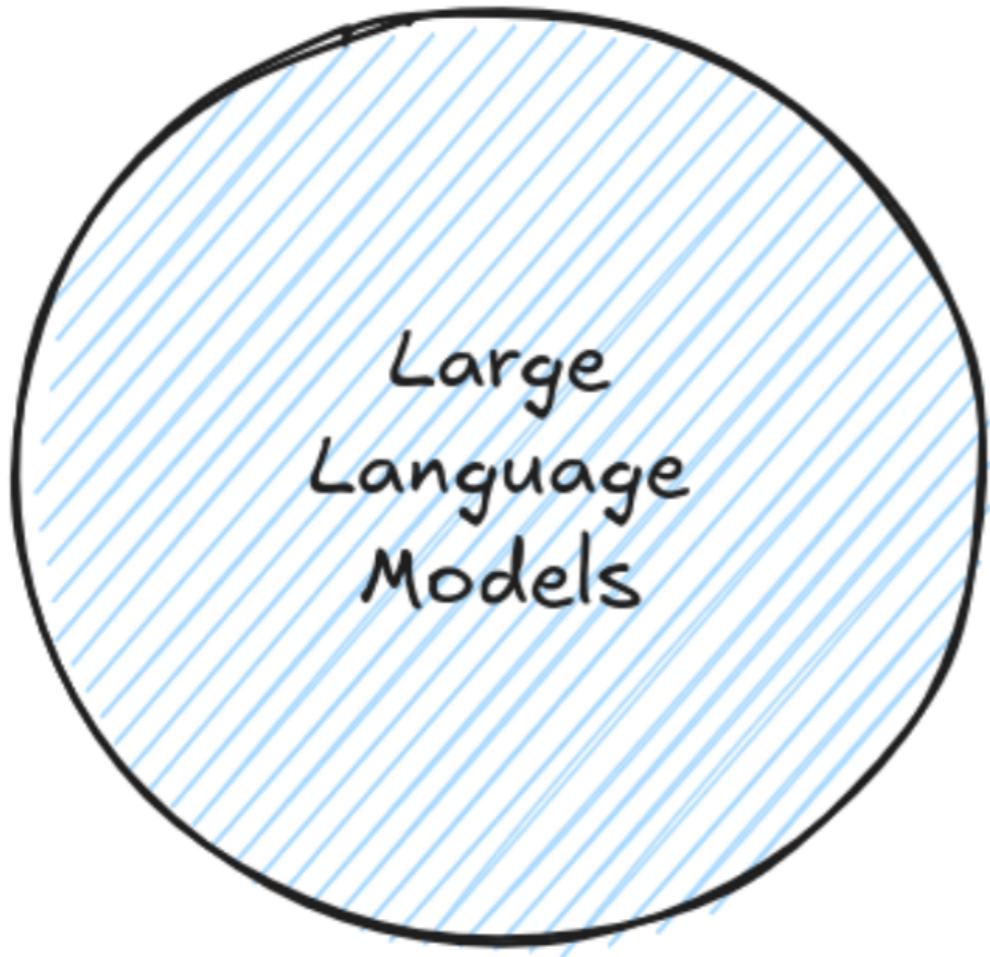
Francesca Donadoni

AI Curriculum Manager, DataCamp

# Course goals

- Define how AI supports multiple coding tasks
  - Code completion
  - Debugging
  - Test generation
  - Documentation
- Choose the right model for your coding needs
- Craft the best prompts for coding purposes
- Build robust AI pipelines to boost your coding productivity

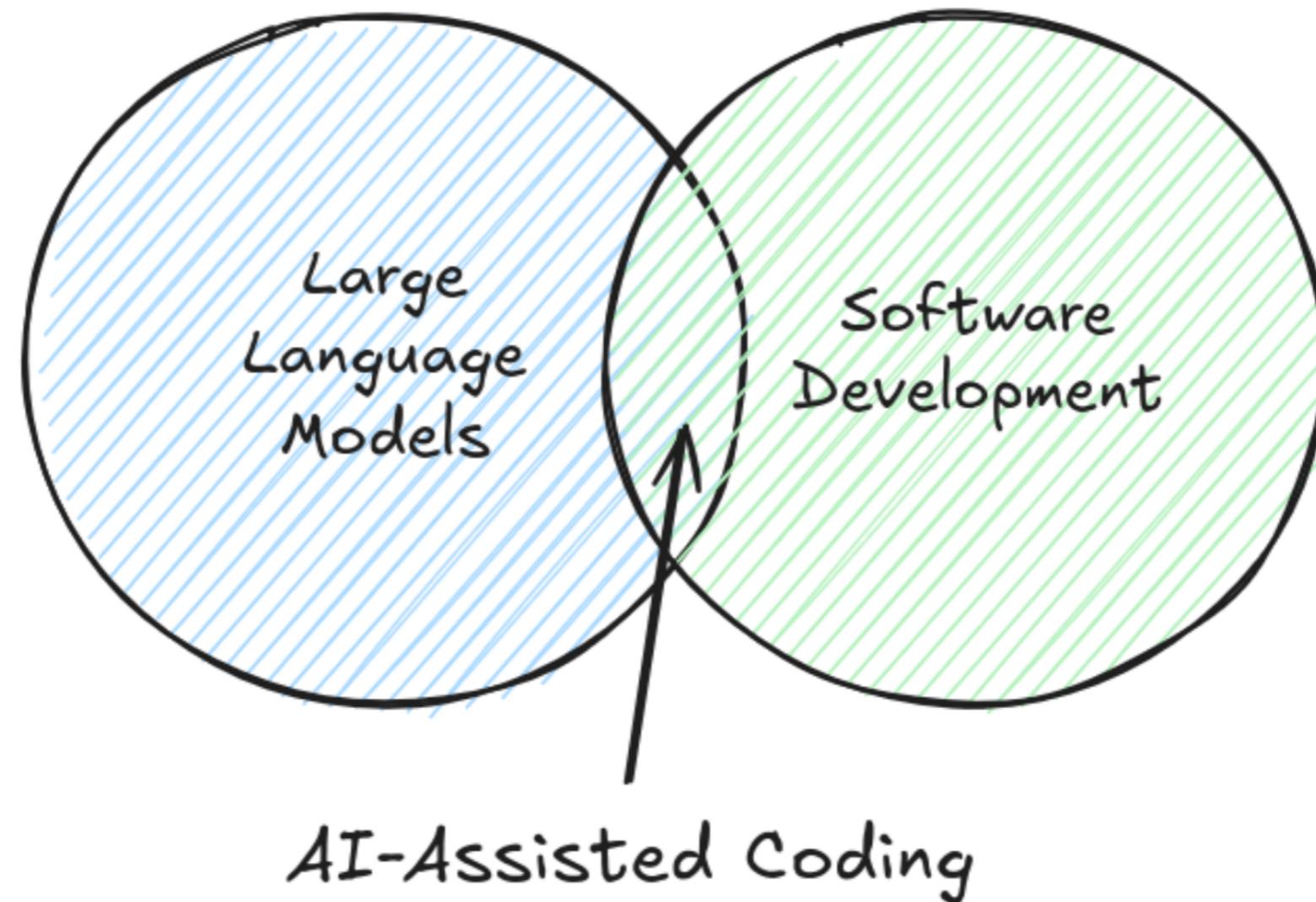
# The AI-assisted coding paradigm



*"Can you help me organizing a 2-day trip to Barcelona?"*

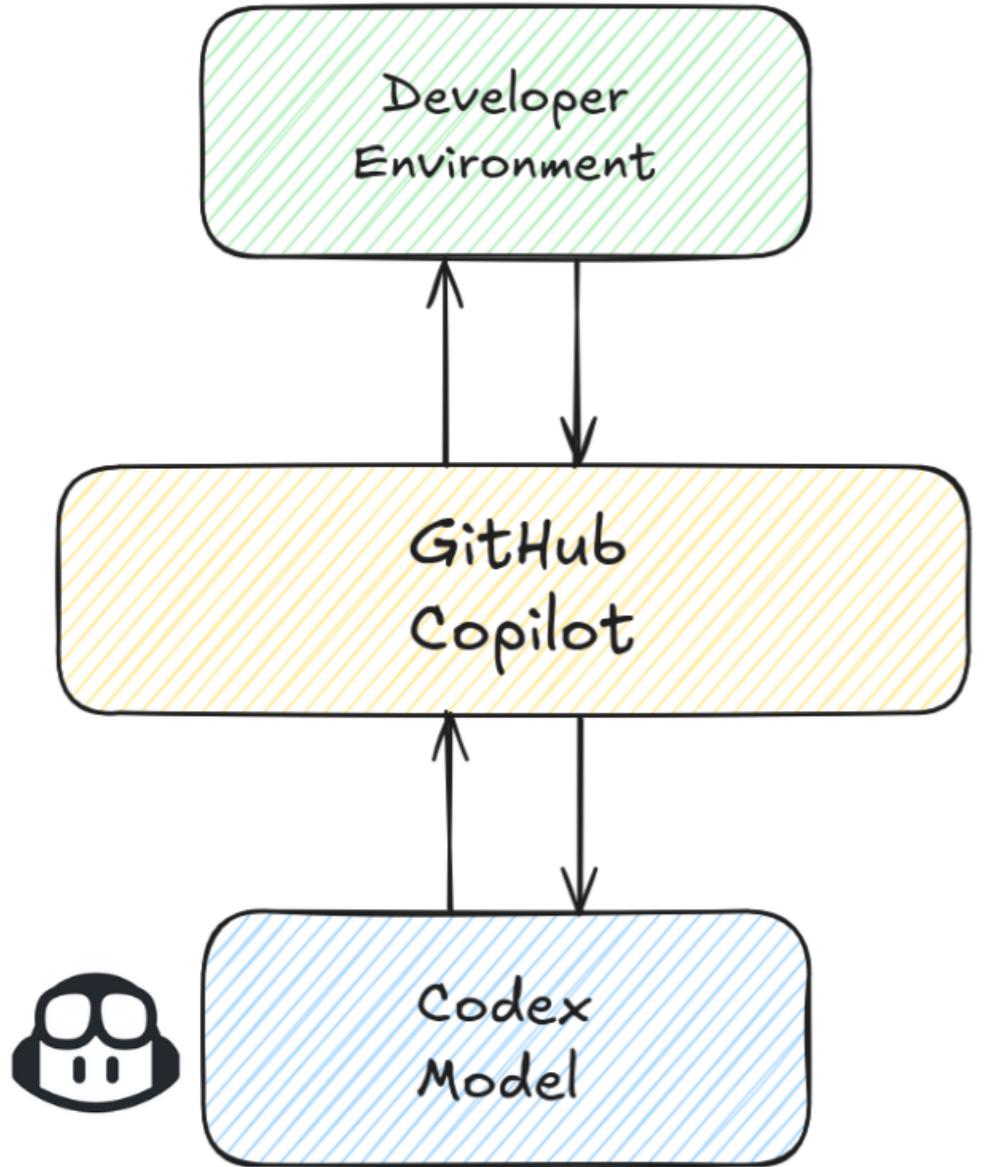
*"I need to finish this function by tomorrow"*

# The AI-assisted coding paradigm

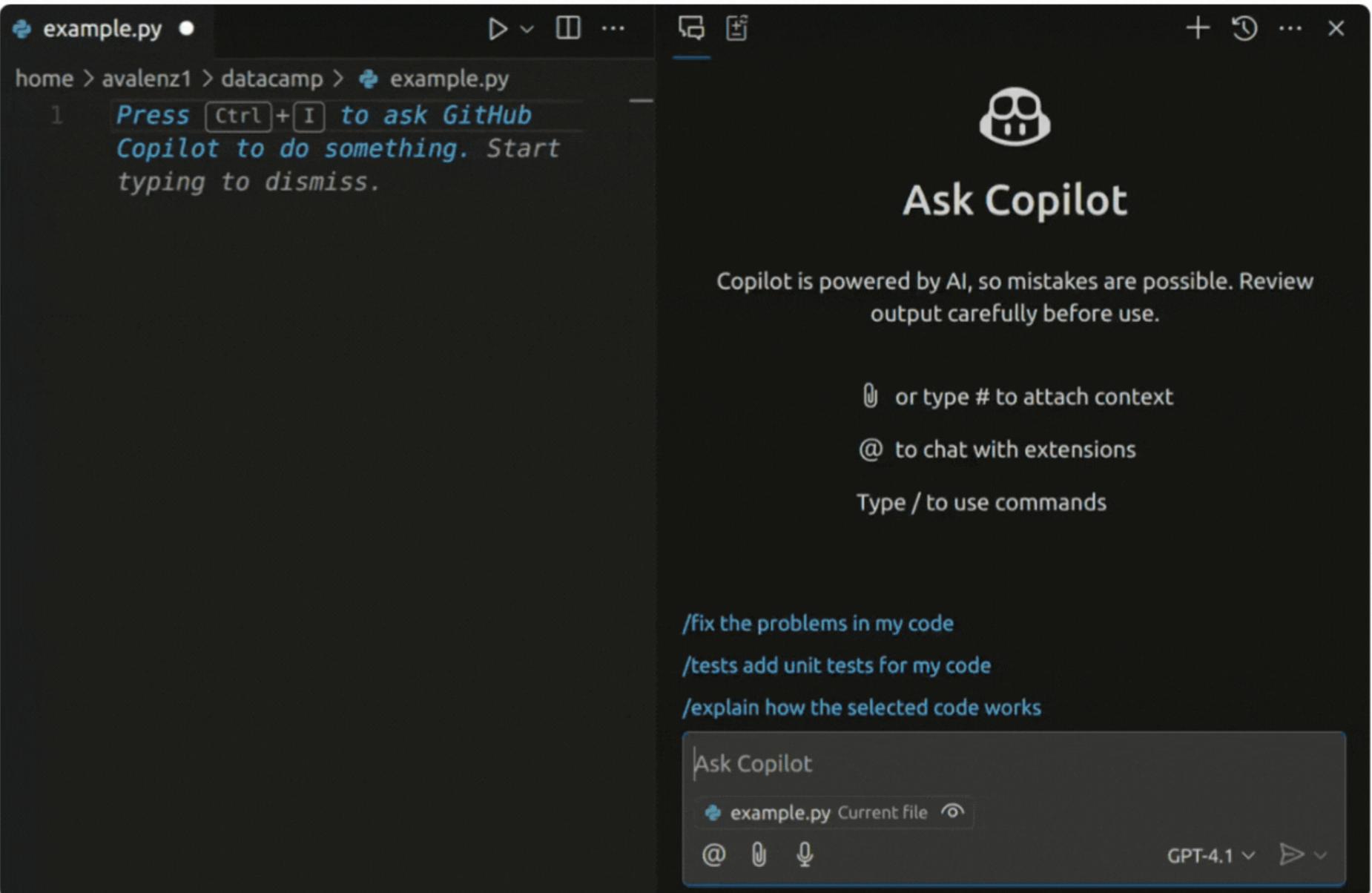
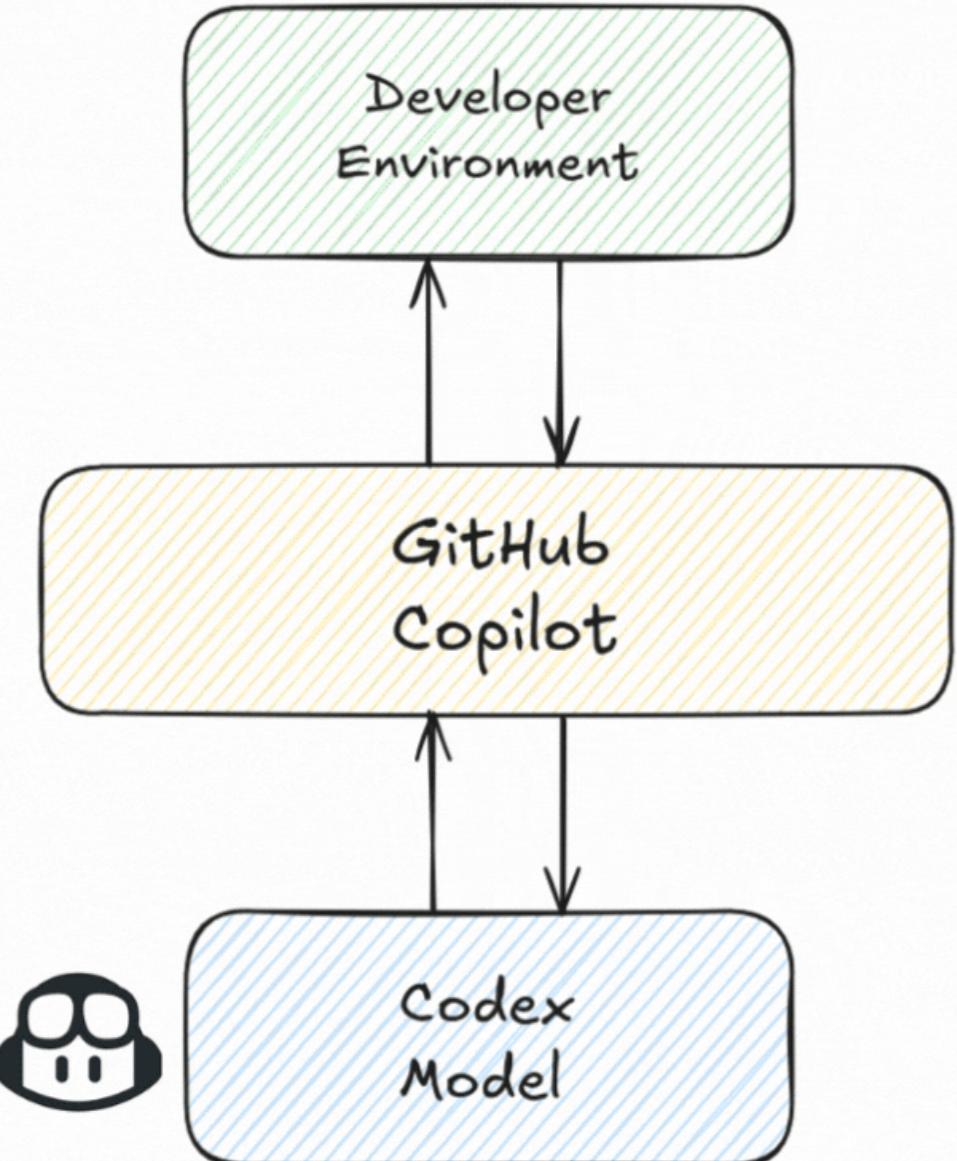


*"Can you help me writing a C function that [...]?"*

# Coding models



# Coding models



# Coding models

The screenshot shows a code editor window for a Python file named `example.py`. The code defines a function `square` that returns the square of its argument `x`. A cursor is positioned at the end of the second line, where the model has suggested the closing brace of the function definition.

**User input:**

```
def square(x):  
    return x * x
```

**Model suggestion:**

A yellow callout box highlights the closing brace of the function definition, with a curved arrow pointing from it to a callout box labeled "Model suggestion".

**GitHub Copilot Suggestions:**

1 Suggestions

**Suggestion 1:**

```
return x * x
```

**Accept suggestion 1**

# Coding models

User input:

"Write a Python function to compute the square of a value"

Model response:

```
def square(value):  
    return value ** 2
```

A screenshot of a GitHub Copilot interface. At the top, it shows a file named "example.py" with a status dot indicating it's open. Below the file path "home > avalenz1 > datacamp > example.py", there is a blue placeholder text: "Press Ctrl+I to ask GitHub Copilot to do something. Start typing to dismiss." The main area of the interface is currently empty, suggesting the user has not yet typed anything.

# Coding models



**GPT**  
(OpenAI)



**CodeGen**  
(Salesforce)

# Coding models

General models (with coding capabilities)



**GPT**

(OpenAI)



**Claude**  
(Anthropic)



**Gemini**  
(Google)

# Coding models

General models (with coding capabilities)



**Claude**  
(Anthropic)



**GPT**  
(OpenAI)



**Gemini**  
(Google)

Code-specific models (fine-tuned for coding)



**CodeGen**  
(Salesforce)



**CodeLLaMa**  
(Meta AI,  
LLaMa model)



**DeepSeek-Coder**  
(DeepSeek,  
DeepSeek model)

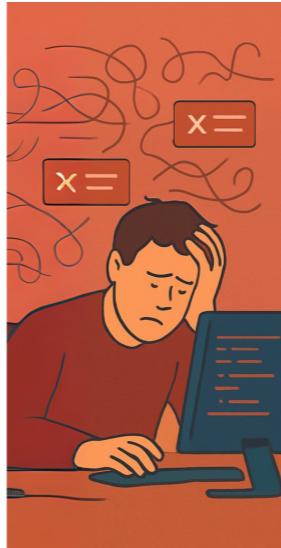
# Choosing the model

*Which model is best for my task?*

# Choosing the model

*Which model is best for my task?*

Task type	Best suited models
Complex tasks requiring deep reasoning	o-series, Claude Opus, Gemini Ultra



# Choosing the model

*Which model is best for my task?*

Task type	Best suited models
Complex tasks requiring deep reasoning	o-series, Claude Opus, Gemini Ultra
Simple tasks requiring low latency	o-mini, Gemini Flash



# Choosing the model

*Which model is best for my task?*

Task type	Best suited models
Complex tasks requiring deep reasoning	o-series, Claude Opus, Gemini Ultra
Simple tasks requiring low latency	o-mini, Gemini Flash
General coding tasks	GPT, Claude Sonnet



# **Let's practice!**

**AI-ASSISTED CODING FOR DEVELOPERS**

# Prompt engineering basics

AI-ASSISTED CODING FOR DEVELOPERS



Francesca Donadoni

AI Curriculum Manager, DataCamp

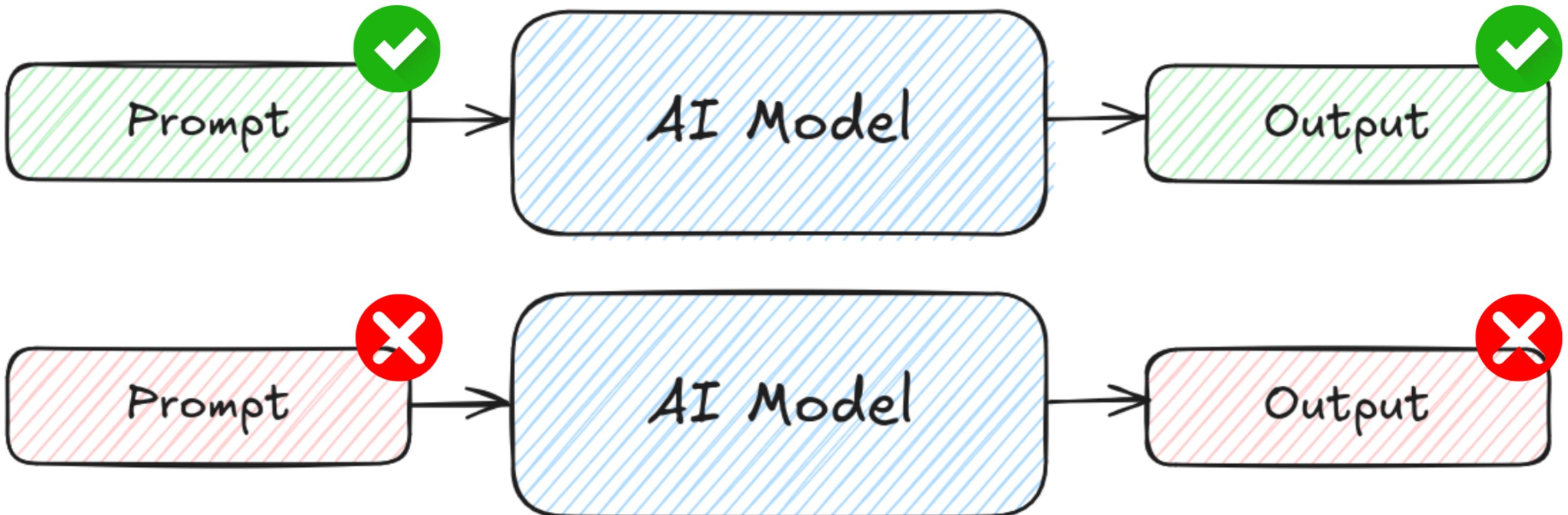
# What is a prompt?

*Prompt: The input or instruction we give to an AI model*



# What is a prompt?

*Prompt: The input or instruction we give to an AI model*



# What is a prompt?

*Driver: "Where to?"*

*Character: "Anywhere"*

**PROMPT:** Vague, unclear for the model



# What is a prompt?

*Driver: "Where to?"*

*Character: "Take me to 23 Main Street. Avoid the highway"*

**PROMPT:** Detailed, clear for the model



# Prompt quality

## Bad prompt

"Fix this code"

## Good prompt

"Fix this Python function to handle division by zero errors. Keep the structure the same, and only change the except block"

# Prompt quality

## 😊 Simple prompt

"Generate a Python function to reverse a string"

## ☐ Advanced prompt

"Generate a Python function that reverses a string, handles Unicode characters, and ignores trailing whitespace"

# Components of a prompt

**PROMPT** = **INSTRUCTION** + **CONTEXT** + **CONSTRAINT**

*Fix this Python function to handle division by zero errors. Keep the structure the same, and only change the except block.*

# Components of a prompt

## INSTRUCTION

→ *Fix this Python function*

## CONTEXT

→ *to handle division by zero errors.*

## CONSTRAINT

→ *Keep the structure the same, and only change the except block.*

# **Let's practice!**

**AI-ASSISTED CODING FOR DEVELOPERS**

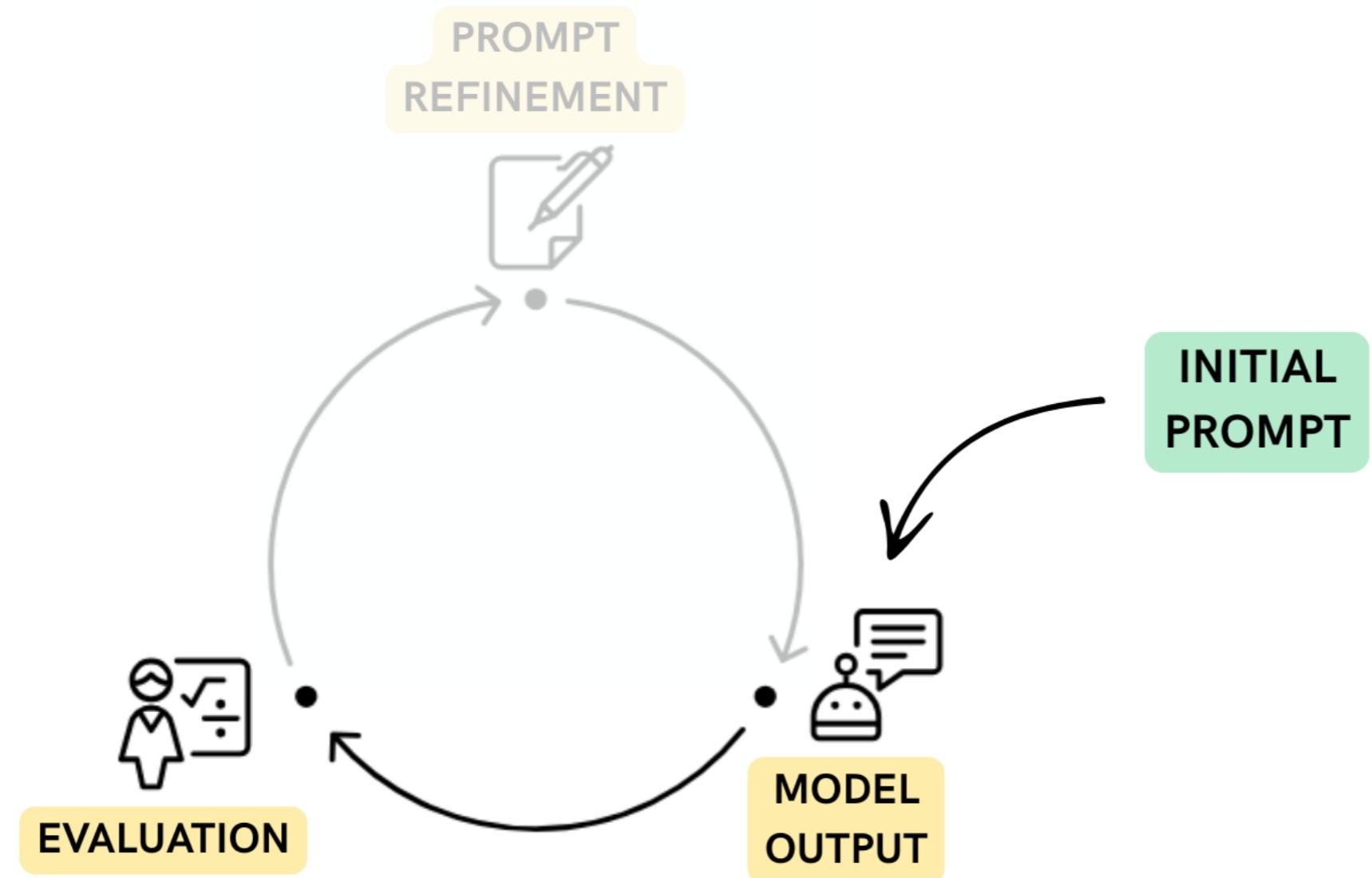
# Adding context for smarter AI responses

AI-ASSISTED CODING FOR DEVELOPERS

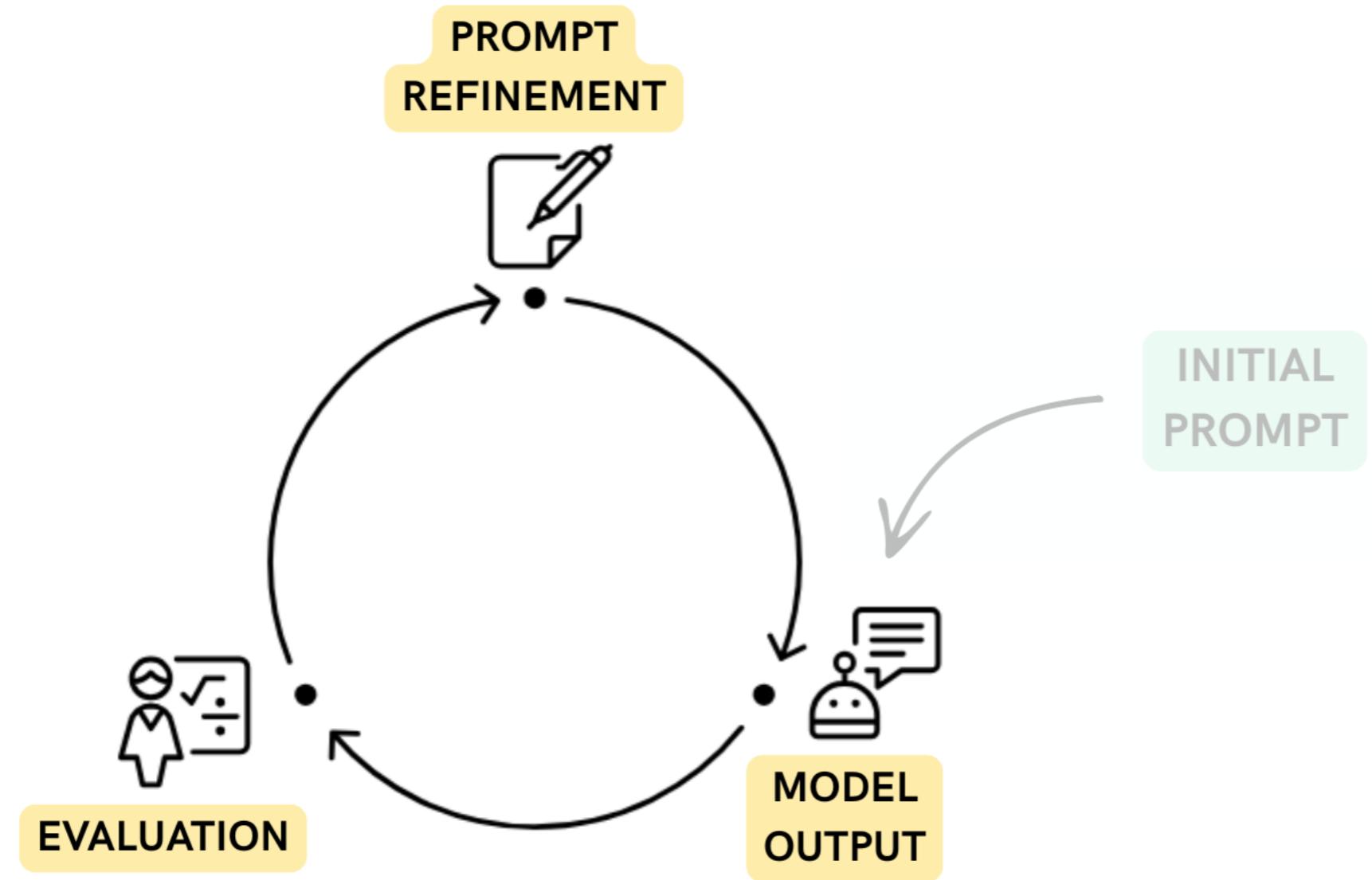
Francesca Donadoni  
AI Curriculum Manager, DataCamp



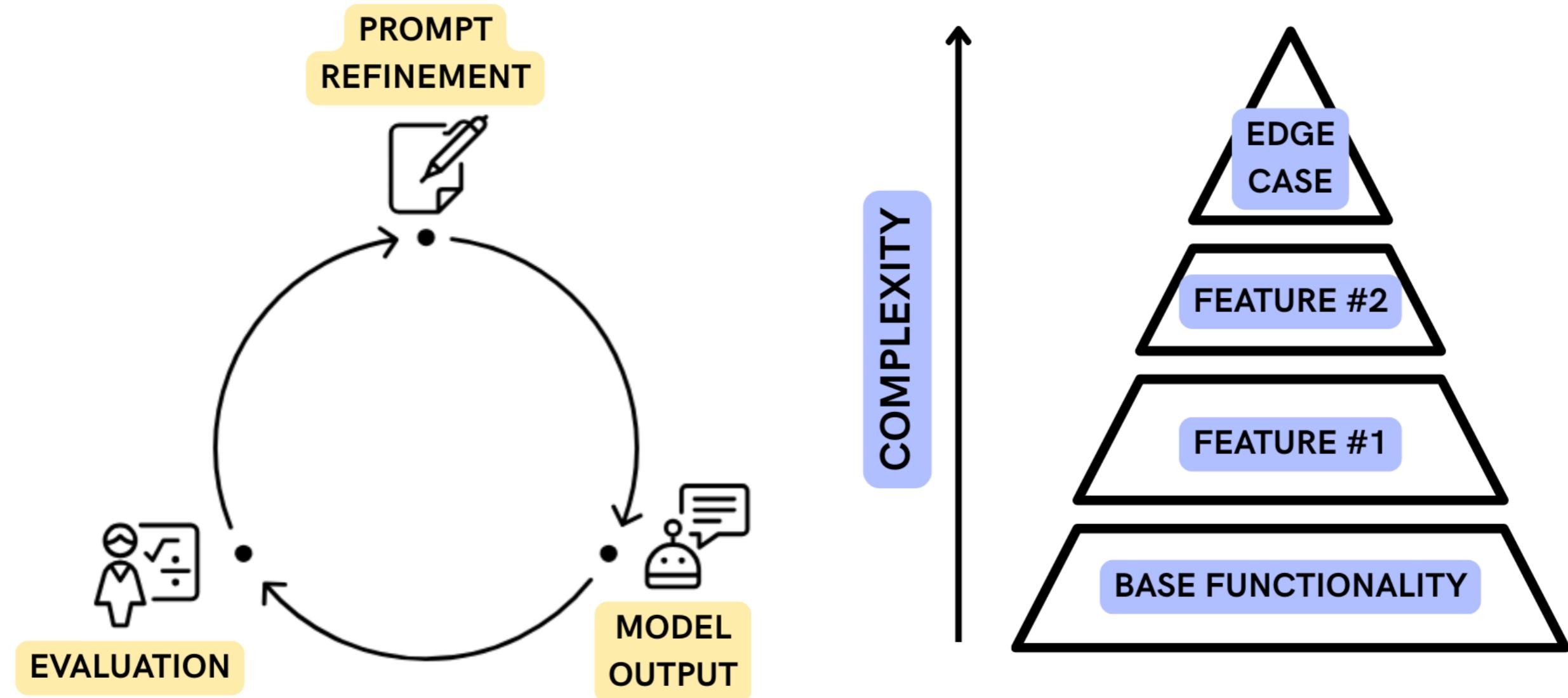
# Prompt refinement



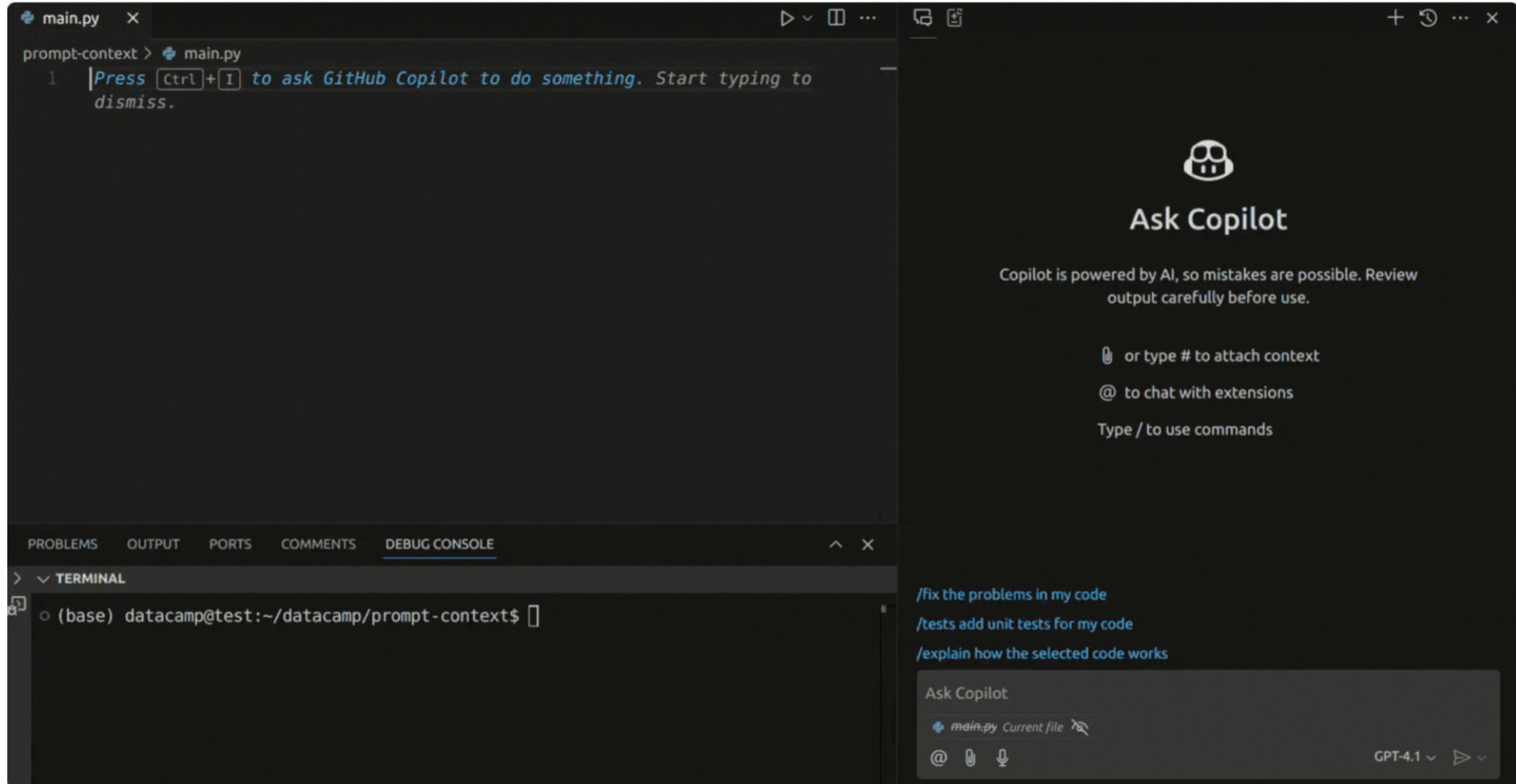
# Prompt refinement



# Prompt refinement



# Prompt refinement



# Contextual prompting

*Providing the right context so the model has all the necessary information to complete the task.*

```
1 # utils.py
2 def fetch_data_from_api(url):
3     ...
4
5 # PROMPT: Write a function in main.py that uses `fetch_data_from_api`
6 # from utils.py to get and parse user data.
7
```

CODE INJECTION



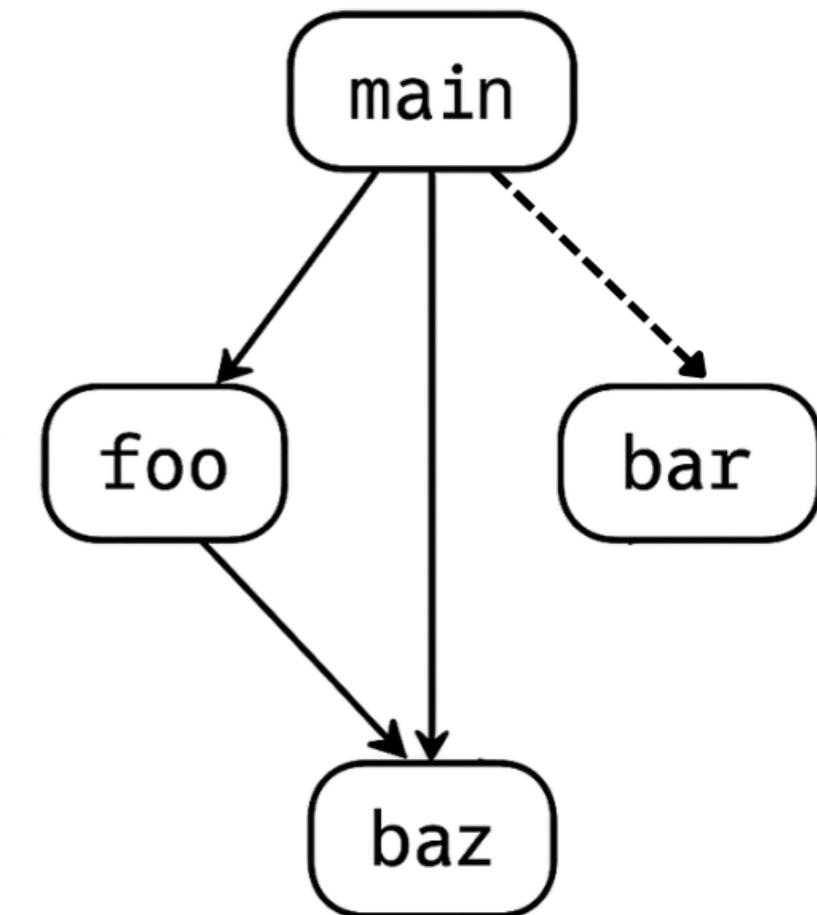
# Contextual prompting

*Providing the right context so the model has all the necessary information to complete the task.*

```
1 # utils.py
2 v def fetch_data_from_api(url):
3 ...
4
5 # PROMPT: Write a function in main.py that uses `fetch_data_from_api`
6 # from utils.py to get and parse user data.
7
```

CODE INJECTION

CALL GRAPH



# Contextual prompting

The screenshot shows a Visual Studio Code (VS Code) interface with a floating 'Ask Copilot' window. The code editor displays a Python script named 'main.py' containing a function to validate email addresses. The 'Ask Copilot' window provides AI-generated documentation for the function, including examples of edge cases and validation rules. It also includes a help section with keyboard shortcuts and command examples.

**Code Editor:**

```
1 import re
2
3 test_emails = [
4     "user@example.com",
5     "user@.com",
6     "user@domain..com",
7     ".user@domain.com",
8     "user@-domain.com"
9 ]
10
11 def is_valid_email(email):
12     """
13         Validates an email address using a regular expression.
14         Returns True if valid, False otherwise.
15         Edge cases handled:
16             - Username cannot start/end with dot
17             - No consecutive dots in username
18             - No leading dot in username
19     """
20     pattern = ...

```

**Ask Copilot Window:**

**Ask Copilot**

Copilot is powered by AI, so mistakes are possible. Review output carefully before use.

ⓘ or type # to attach context  
@ to chat with extensions  
Type / to use commands

/fix the problems in my code  
/tests add unit tests for my code  
/explain how the selected code works

main.py Current file

@ ⓘ

GPT-4.1 ▶

# Contextual prompting

Minimal prompt: "I am getting a ConnectionError in my python script. How can I fix it?"

Model response:

To help you fix the `ConnectionError`, please follow these steps:

1. **Check your internet connection** – Make sure your machine is online.
2. **Verify the URL/host** – Ensure the URL or host you are connecting to is correct and reachable.
3. **Handle exceptions** – Use a `try-except` block to catch and handle `requests.exceptions.ConnectionError` if you are using the `requests` library.
4. **Check firewall/proxy settings** – Ensure your network/firewall/proxy is not blocking the connection.
5. **Update dependencies** – Make sure your libraries (like `requests`) are up to date:

```
pip install --upgrade requests
```

6. **Retry logic** – Implement retry logic for transient errors.

# Contextual prompting

Full context: "I am getting the following error in my python script: <paste error message>. How can I fix it?"

Model response:

You have a typo in your URL. Change:

```
response = requests.get("https://jsonplaceholdererr.typicode.
```

to:

```
import requests
response = requests.get("https://jsonplaceholder.typicode.c
print(response.json())
```

This will fix the `NameResolutionError`.

# **Let's practice!**

**AI-ASSISTED CODING FOR DEVELOPERS**