

Advanced prompt engineering for coding assistance

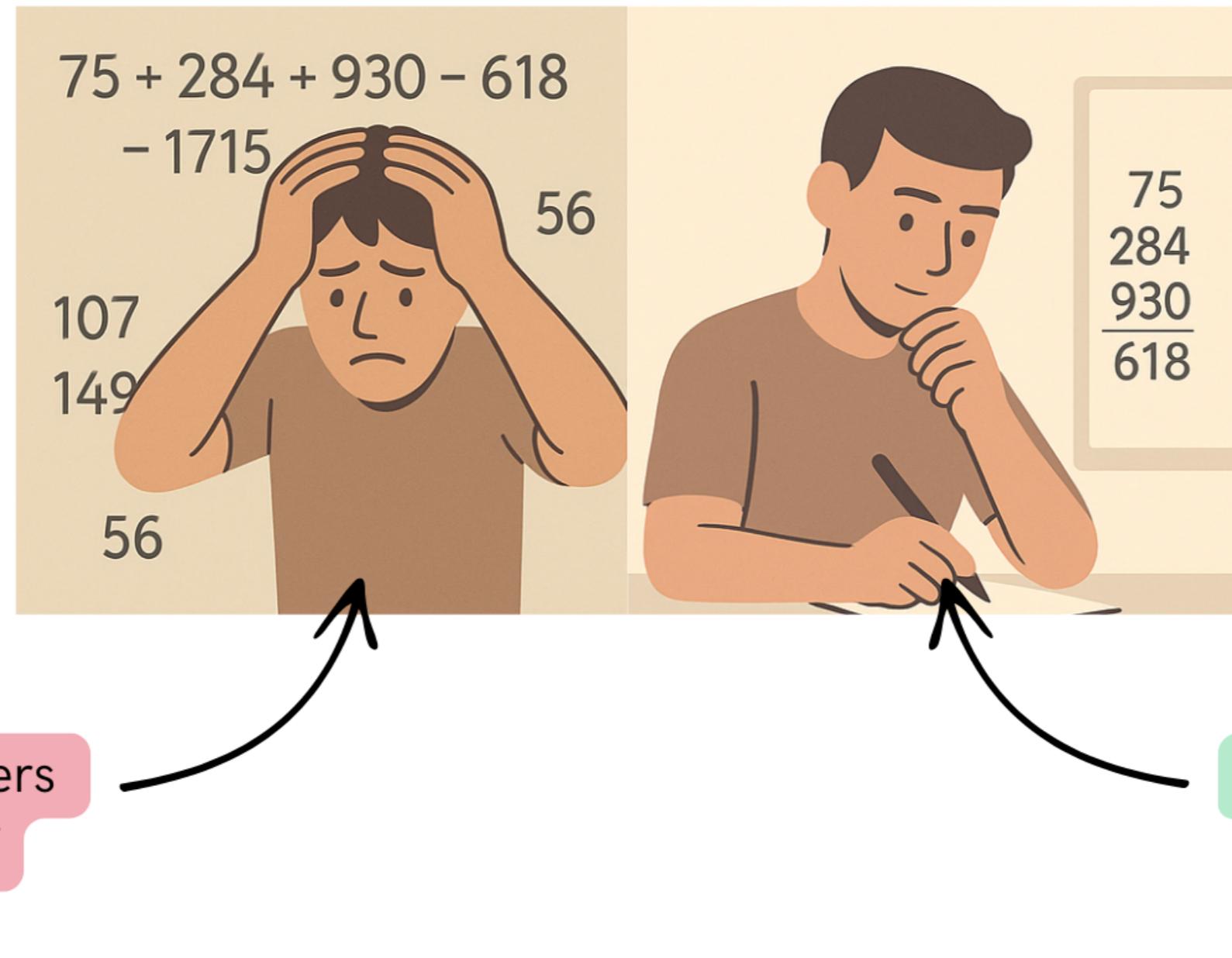
AI-ASSISTED CODING FOR DEVELOPERS



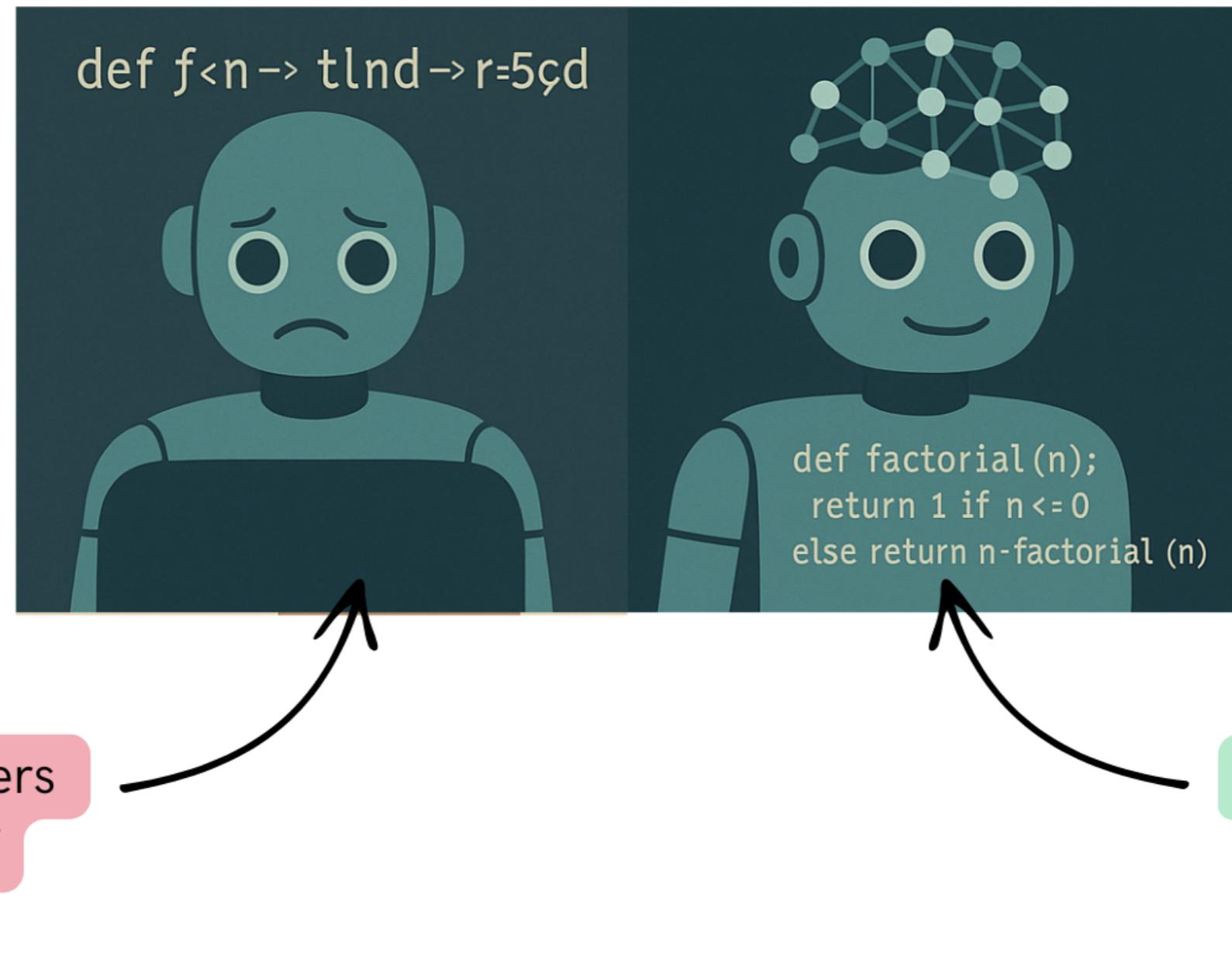
Francesca Donadoni

AI Curriculum Manager, DataCamp

Chain-of-thought



Chain-of-thought



Chain-of-thought

1st Chain-of-Thought approach

<Regular Prompt>

+

"Let's think step by step"

Large Language Models are Zero-Shot Reasoners

Takeshi Kojima
The University of Tokyo
t.kojima@weblab.t.u-tokyo.ac.jp

Machel Reid
Google Research*

Shixiang Shane Gu
Google Research, Brain Team

Yutaka Matsuo
The University of Tokyo

Yusuke Iwasawa
The University of Tokyo

Abstract

Pretrained large language models (LLMs) are widely used in many sub-fields of natural language processing (NLP) and generally known as excellent *few-shot* learners with task-specific exemplars. Notably, chain of thought (CoT) prompting,

<https://arxiv.org/pdf/2205.11916.pdf>

Chain-of-thought

- CoT forces the model to explicitly compute the intermediate steps
 - More accurate response

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 ✗

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

(Output) There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls. ✓

Chain-of-thought

😊 Normal Prompt

"Write a Python function that checks if a string is a palindrome"

☐ CoT Prompt

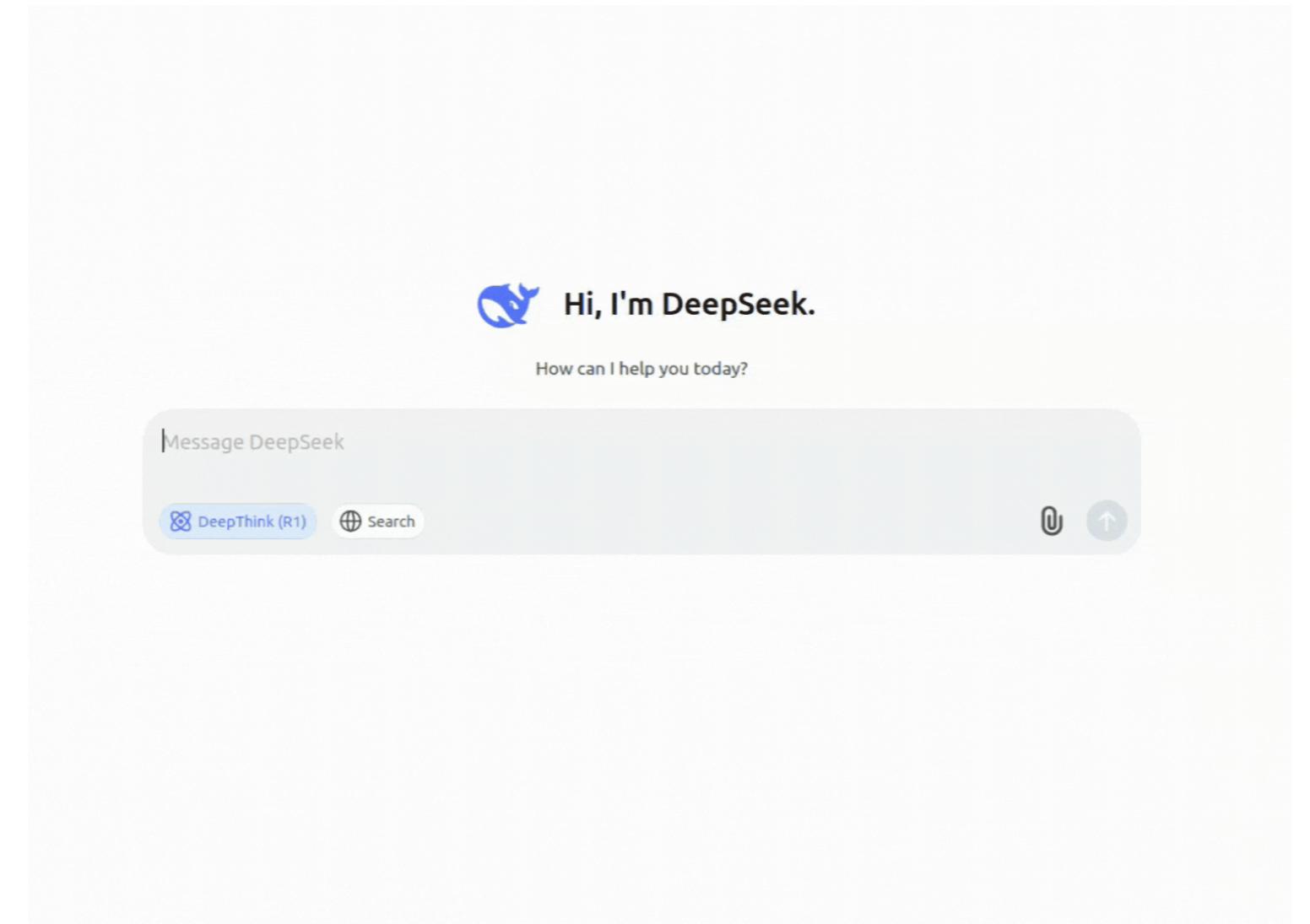
"Check if a string is a palindrome.

- Step 1: Normalize the string (lowercase, remove non-alphanumeric characters).
- Step 2: Reverse the cleaned string.
- Step 3: Compare the original and reversed versions.

"Write a function that implements this logic"

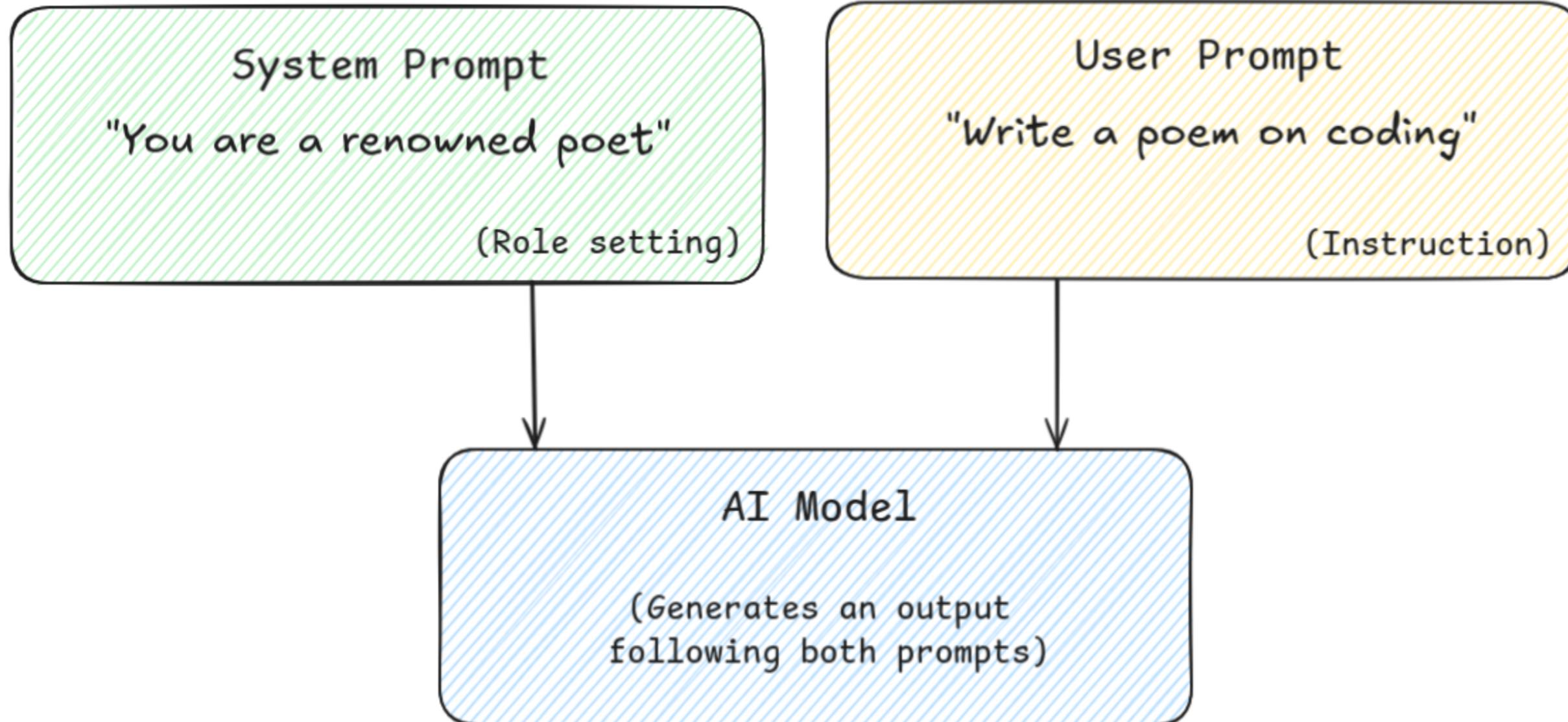
Reasoning models

- Perform Chain-of-Thought reasoning by default
- Self-verify intermediate steps
- Iterate over their own reasoning



System roles

- *Role definitions have been shown to consistently improve output quality and accuracy*



System roles

□ Coding Tutor

"You are a friendly programming tutor. Explain each concept in simple terms, using analogies when helpful. Highlight common mistakes to avoid"

□ Senior Software Engineer

"You are a senior software engineer helping to debug code. First, identify what the code is trying to do. Then, analyze where and why it might be failing"

Let's practice!

AI-ASSISTED CODING FOR DEVELOPERS

Documenting code with AI

AI-ASSISTED CODING FOR DEVELOPERS



Francesca Donadoni
AI Curriculum Manager, DataCamp

AI-Powered docstring generation



AI-Powered docstring generation

- User prompt: *Write a docstring for this function*

```
def sales_trend(self):  
    ...  
    if increasing > decreasing:  
        return "Increasing trend"  
    else:  
        return "Decreasing or stable trend"
```

- Model's response:

```
def sales_trend(self):  
    """Analyze sales data to determine the overall trend"""  
    [...]
```

AI-Powered docstring generation

- **User prompt:** *Generate a docstring for this function that includes:*
 - *A simple explanation of what the function does.*
 - *A description of each parameter and its type.*
 - *A description of the return value and its type.*
- **Model's response:**

Here's a complete docstring you can use for your function:

```
def sales_trend(self):  
    """  
        Analyze the sales data to determine whether there is an increasing trend  
        or a decreasing/stable trend over time.  
        [...]  
    """
```

AI-Powered docstring generation

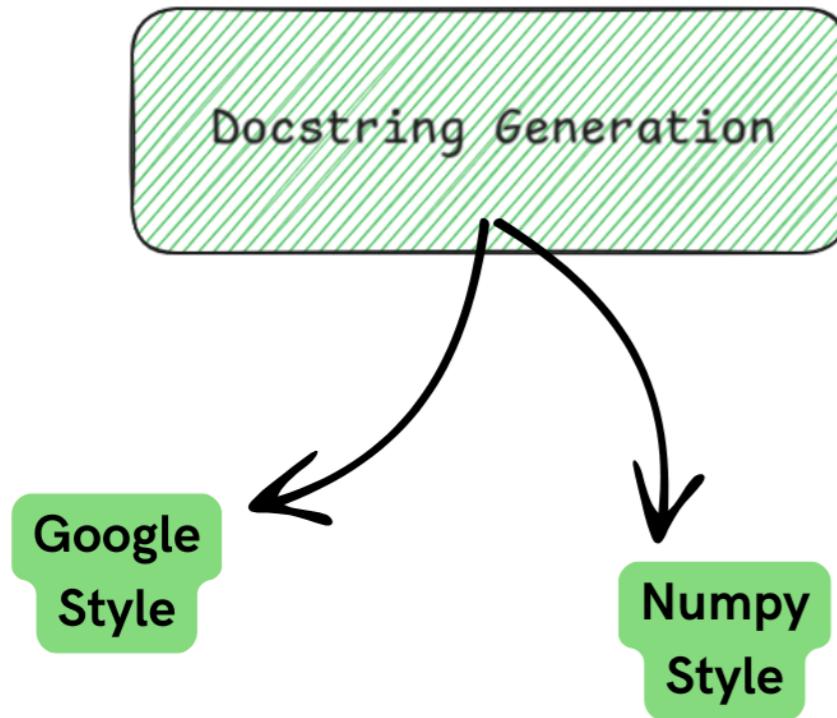
Here's a complete docstring you can use for your function:

```
def sales_trend(self):
    """
    Analyze the sales data to determine whether there is an increasing trend
    or a decreasing/stable trend over time.

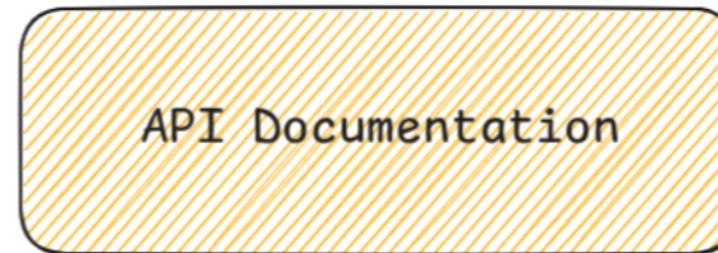
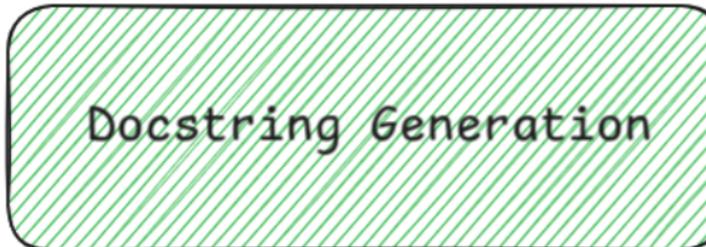
    Parameters
    -----
    self : object
        An object that contains an attribute `daily_sales` (list of int or float),
        which represents daily sales values in chronological order.
    [...]
    """

```

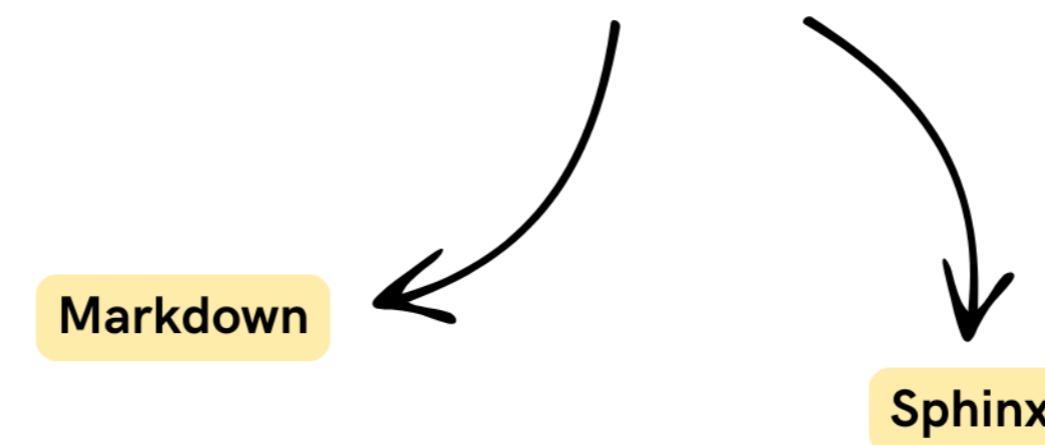
Adapting to documentation styles



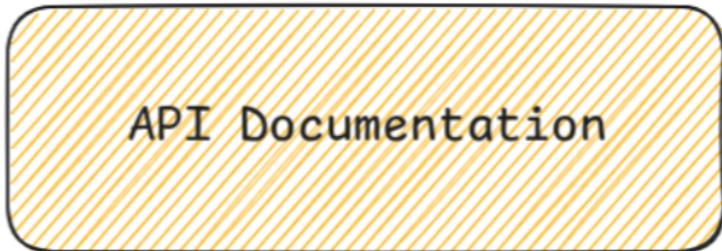
Adapting to documentation styles



"Analyze this class and generate an API reference entry including method names, descriptions, and parameter details"



Summarizing code into READMEs



Summarizing code into READMEs

- **User prompt:** *Generate a README file in Markdown that explains the SalesAnalyzer class and its methods. Include a sample code snippet to help users analyze the sales trend over a random period.*
- **Model's response:**

Here's a polished README.md draft for your SalesAnalyzer class:

SalesAnalyzer

The SalesAnalyzer class provides a simple way to analyze daily sales data and detect whether the sales trend is generally increasing or decreasing/stable. This tool can be useful for businesses or individuals who want a quick, high-level overview of their sales performance over time.

Commit messages from code diffs

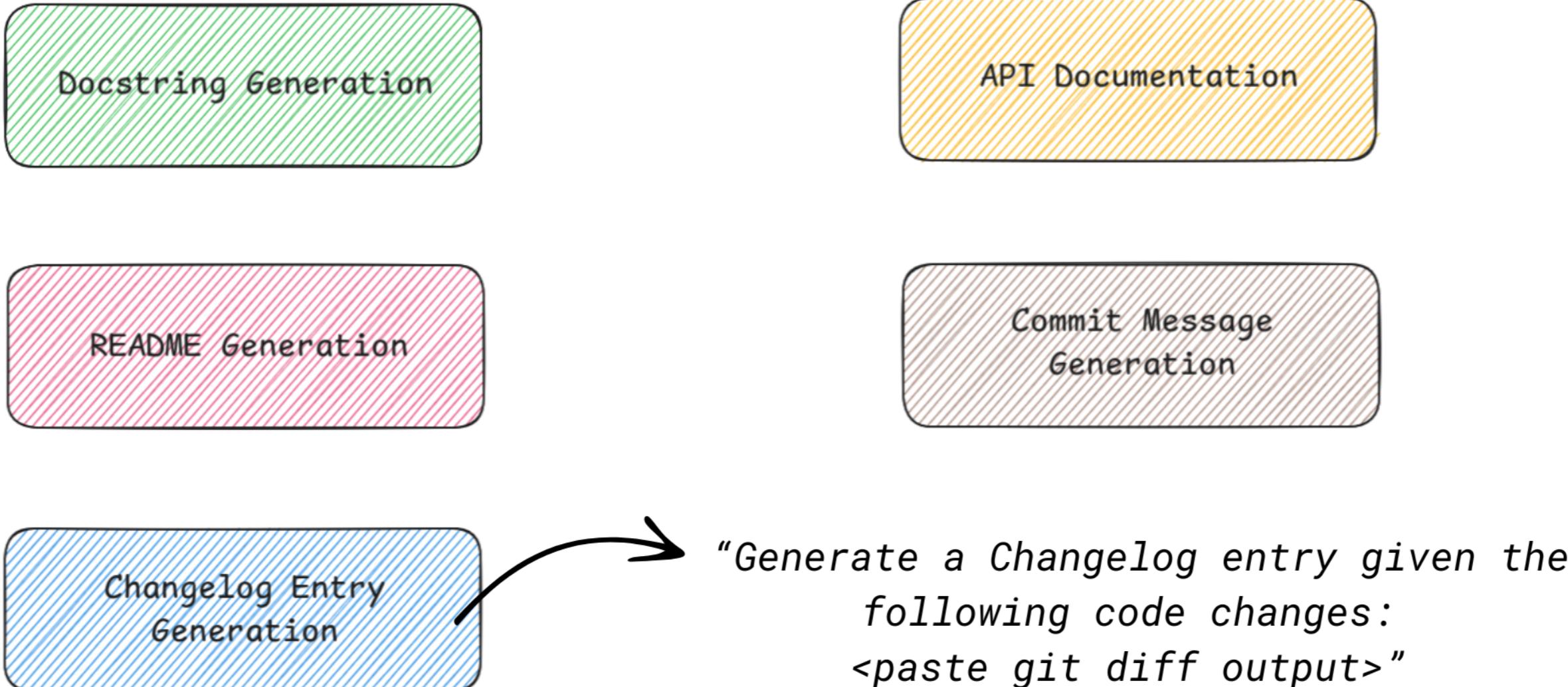


Commit messages from code diffs

- **User prompt:** Write a meaningful commit message given this git diff:

```
(base) datacamp@test:~/datacamp/docstring-generation$ git diff  
+  
+ Raises:  
+     ValueError: If daily_sales contains non-numeric values.  
+  
+     """  
-     return sum(self.daily_sales)  
+     if not all(isinstance(sale, (int, float)) for sale in self.daily_sales):  
+         raise ValueError("All sales figures must be numeric.")  
+     return sum(sale for sale in self.daily_sales)
```

Automating release notes



Other use cases

Docstring Generation

API Documentation

README Generation

Commit Message
Generation

Changelog Entry
Generation

?

Let's practice!

AI-ASSISTED CODING FOR DEVELOPERS

Writing tests and securing code with AI

AI-ASSISTED CODING FOR DEVELOPERS



Francesca Donadoni
AI Curriculum Manager, DataCamp

Test-driven prompting

☺ Normal Prompting

Write a Python function to parse email addresses.

□ Test-Driven Prompting

Write a Python function to parse email addresses. It should pass the following tests:

- *Accepts a valid address like "user@example.com".*
- *Rejects invalid input like "user@@domain".*
- *Raises a ValueError for empty input.*

Test-driven prompting

- **User prompt:** *Generate unit tests in Python for a function that processes user-submitted form data and inserts it into a SQL database.*
- **Model response:**

Here's a ready-to-run pytest suite that covers validation, normalization, SQL-injection safety, and transaction behavior for a function that processes form data and inserts a record into a SQL database.

```
[...]
# ----- Test DB fixtures -----

@pytest.fixture()
def conn():
    """In-memory SQLite DB with a users table."""
    c = sqlite3.connect(":memory:") [...]
```

Test-driven prompting

- **User prompt:** Generate Python unit tests using pytest for the following function. Include cases for empty input, SQL keywords in the input, and special characters. Assume a mock database connection. <INSERT FUNCTION>
- **Model response:**

Here's a self-contained pytest test module that uses mocks (no real DB) and covers empty input, SQL keywords, and special characters. It also verifies commit/close behavior and that exceptions are swallowed.

```
[...]
@pytest.fixture
def mock_sqlite_connect(monkeypatch):
    """
    Patch sqlite3.connect to return a MagicMock connection + cursor.
    [...]
```

Prompting for security

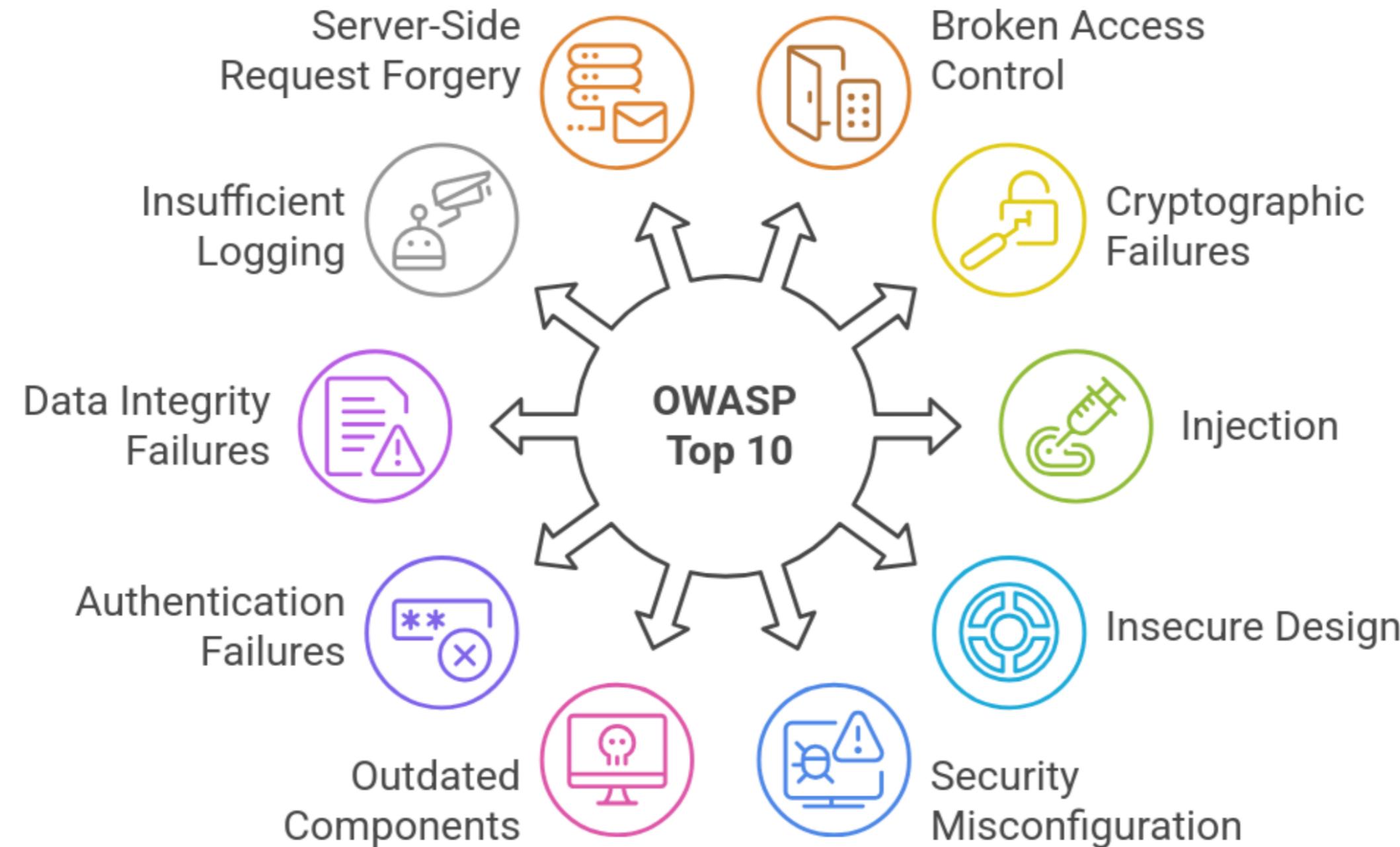
☺ Normal Prompting

Scan the following Python function for potential vulnerabilities and suggest safer alternatives if any are found.

□ Test-Driven Prompting

Review this code for common web security issues, including SQL injection, XSS, and input validation flaws.

OWASP Top 10



OWASP Top 10

- **User prompt:** Audit the following function for the OWASP Top 10 vulnerabilities.

```
def login(username, password):  
    query = f"SELECT * FROM users WHERE username = '{username}' \  
        AND password = '{password}'"  
    result = db.execute(query)  
    return result.fetchone()
```

- **Model response:**

Here's a focused OWASP Top 10 audit of your login function, plus a safer rewrite.

1. A03: Injection. Direct string interpolation into SQL makes this trivially SQL-injectable.

Example payloads:

- `username = "admin' --"` → bypasses password check. [...]

Let's practice!

AI-ASSISTED CODING FOR DEVELOPERS