Lesson 04 Demo 03
Analyzing AI Responses to Different Prompt Styles

Objective: To analyze how different prompt styles influence AI-generated code responses and determine the most effective techniques for guiding AI assistance

Tools required: VS Code with GitHub Copilot

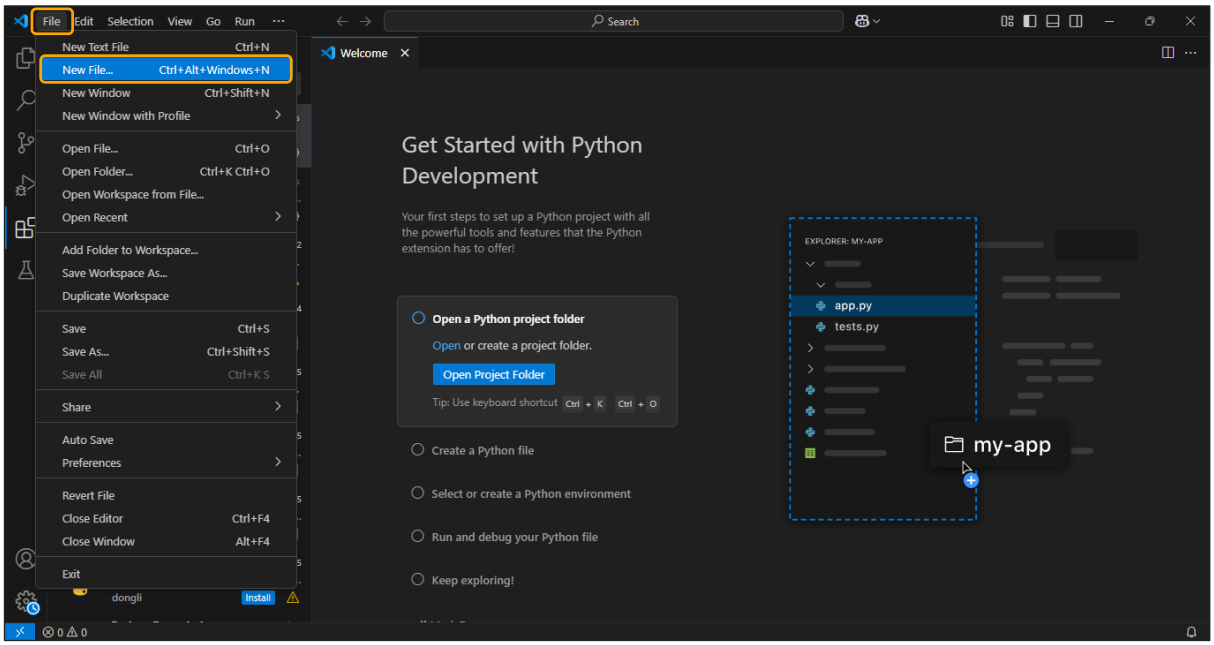
Prerequisites: None

Steps to be followed:

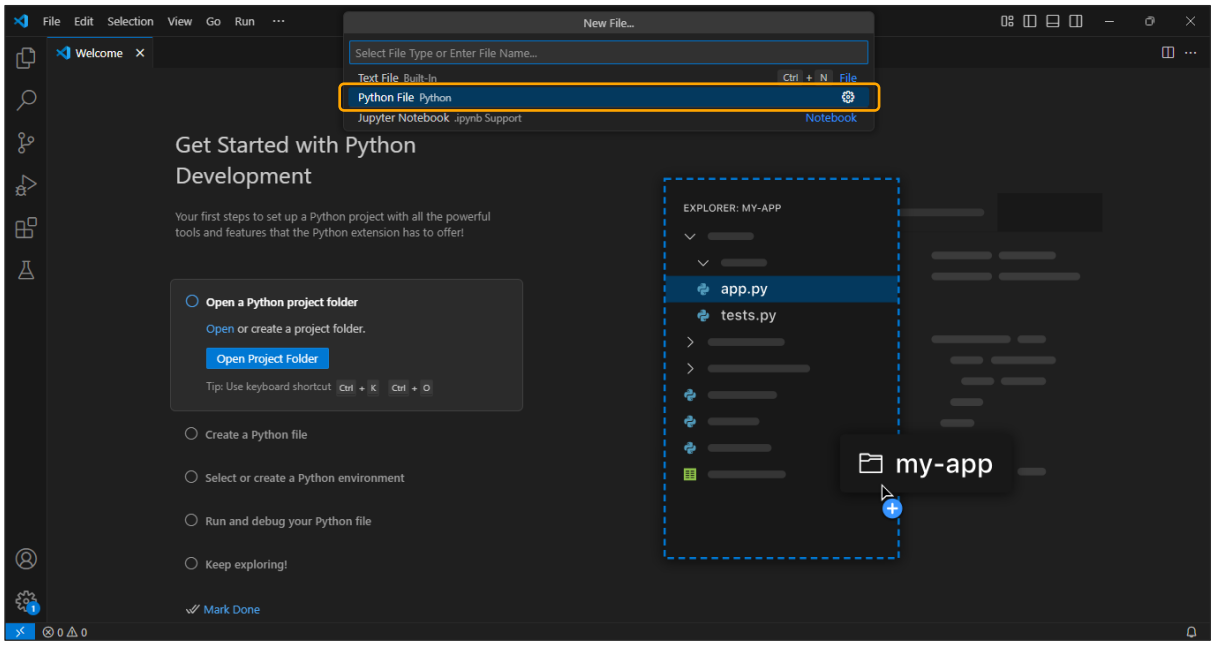
1. Launch VS Code and open a new file
2. Create a function using inline comments
3. Create a function using docstring-based prompts
4. Create a function using contextual hints
5. Create a function using partial code completion

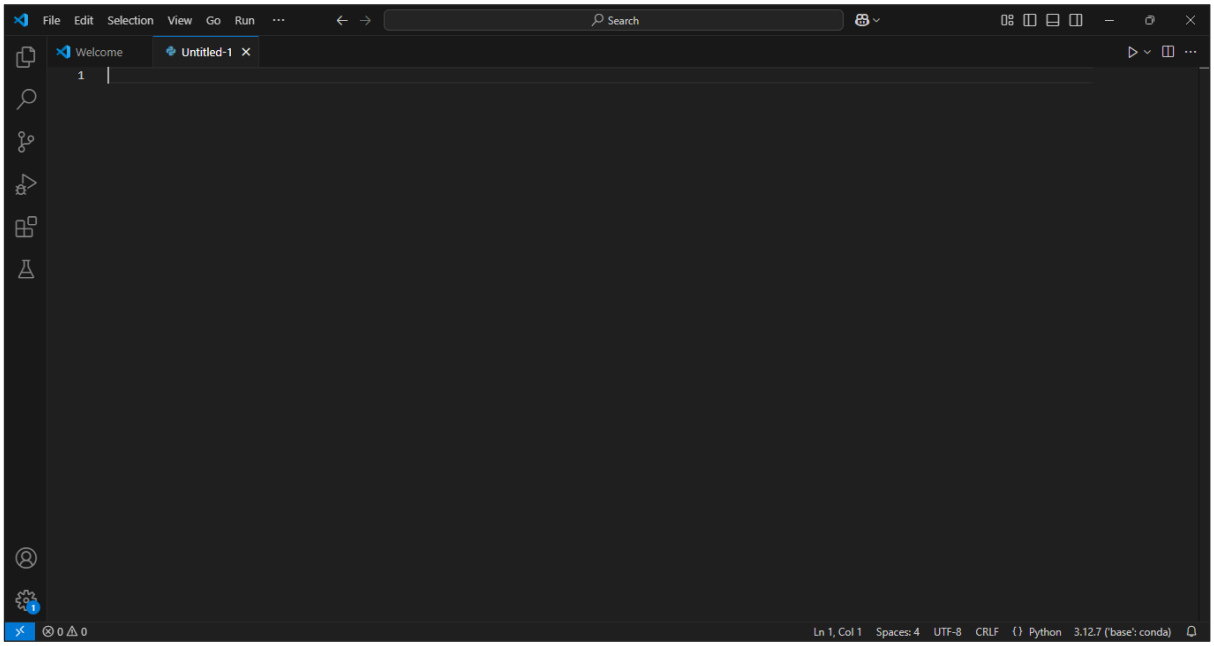
Step 1: Launch VS Code and open a new file

1.1 Launch VS Code and click on **File** and then **New File**



1.2 Select the **Python File** option from the name bar on top and a new Python file named Untitled-1 will open

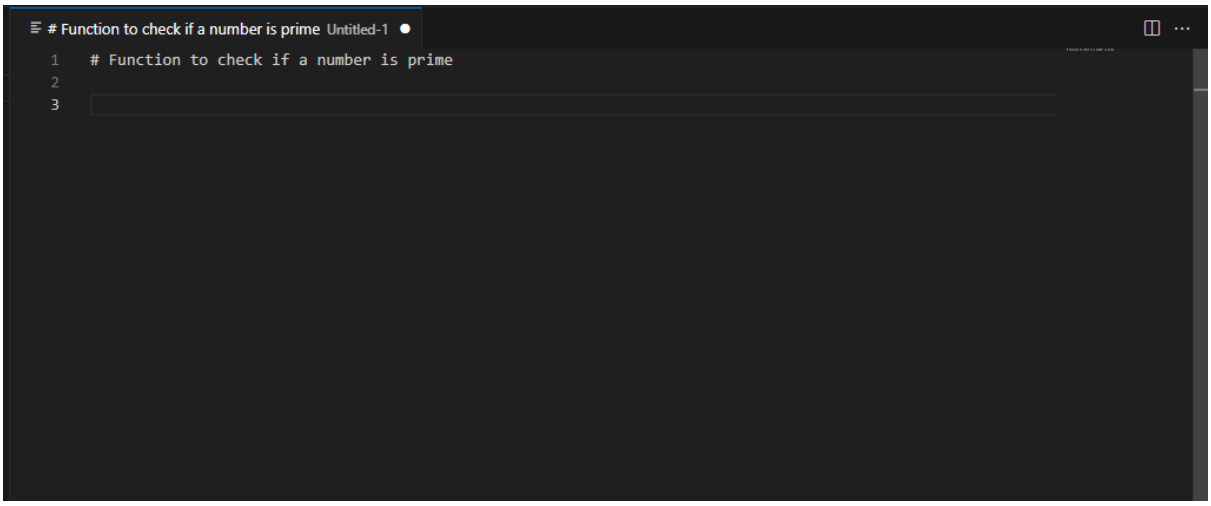




Step 2: Create a function using inline comments

2.1 Type the following comment before defining a function and press enter to get suggestions:

 # Function to check if a number is prime



2.2 Pause and observe how Copilot suggests the function's body:

Note: You can press **tab** on your keyboard to accept the AI-suggested code or you can modify and run it with test values.

Step 3: Create a function using docstring-based prompts

3.1 Clear the previous function, type a new one with a structured docstring, and press enter to get suggestions

Example input:

def is_prime(n):

"""

This function checks whether a number is prime.

It returns True if the number is prime, otherwise False.

"""

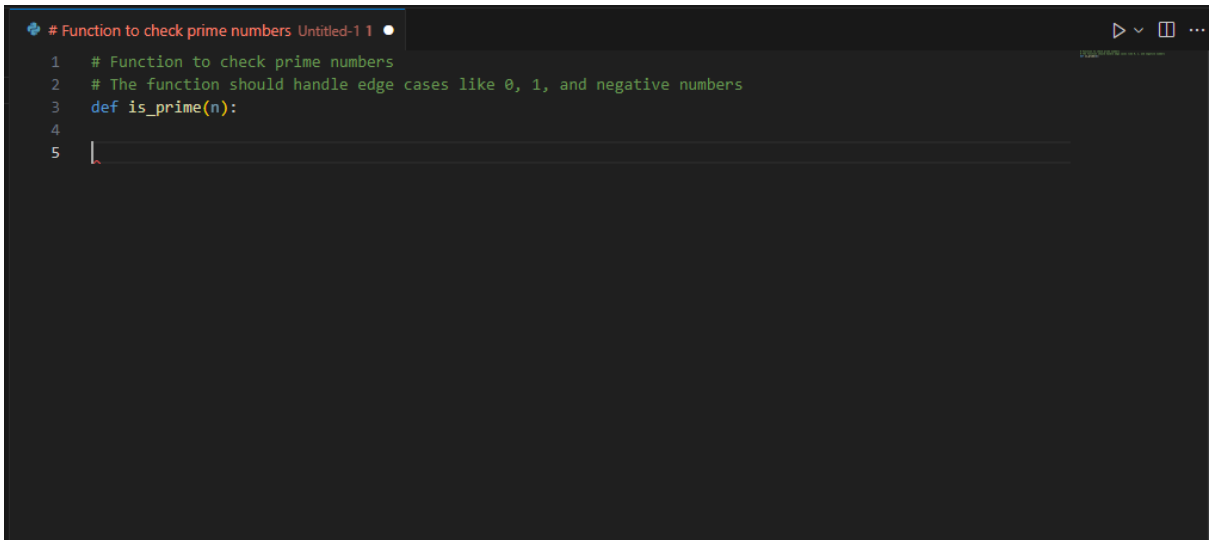
3.2 Pause and observe how Copilot suggests the function's body

Note: Press tab to accept the function or modify and run it with test values

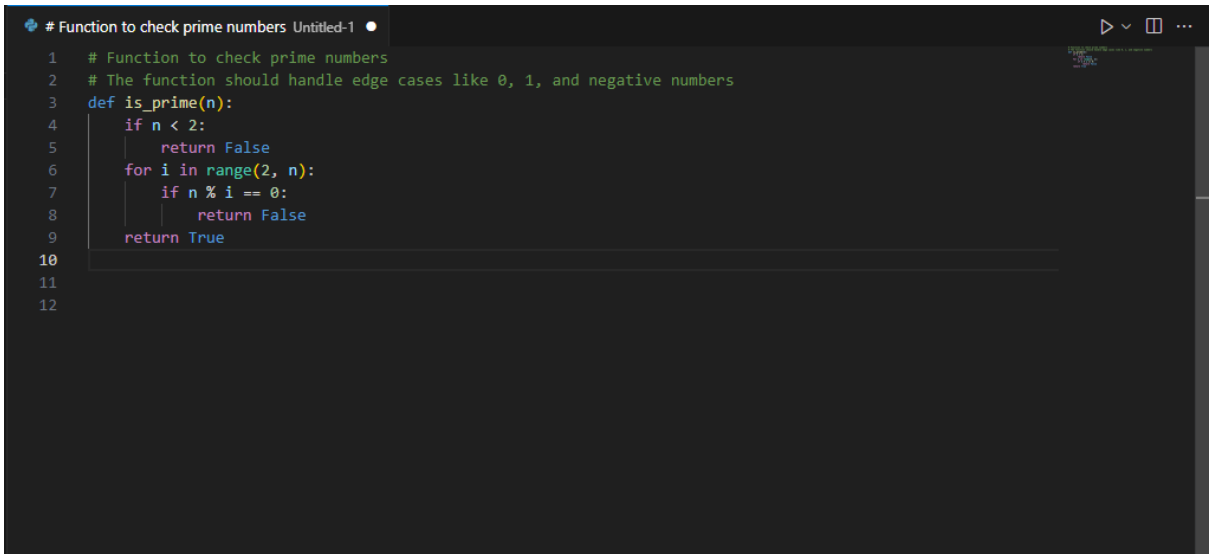
Step 4: Create a function using contextual hints

4.1 Delete the previous function, write a new one with more descriptive comments, and press enter to get suggestions:

Example input:
 # Function to check prime numbers
 # The function should handle edge cases like 0, 1, and negative numbers
def is_prime(n):



4.2 Pause and observe how Copilot suggests the function's body

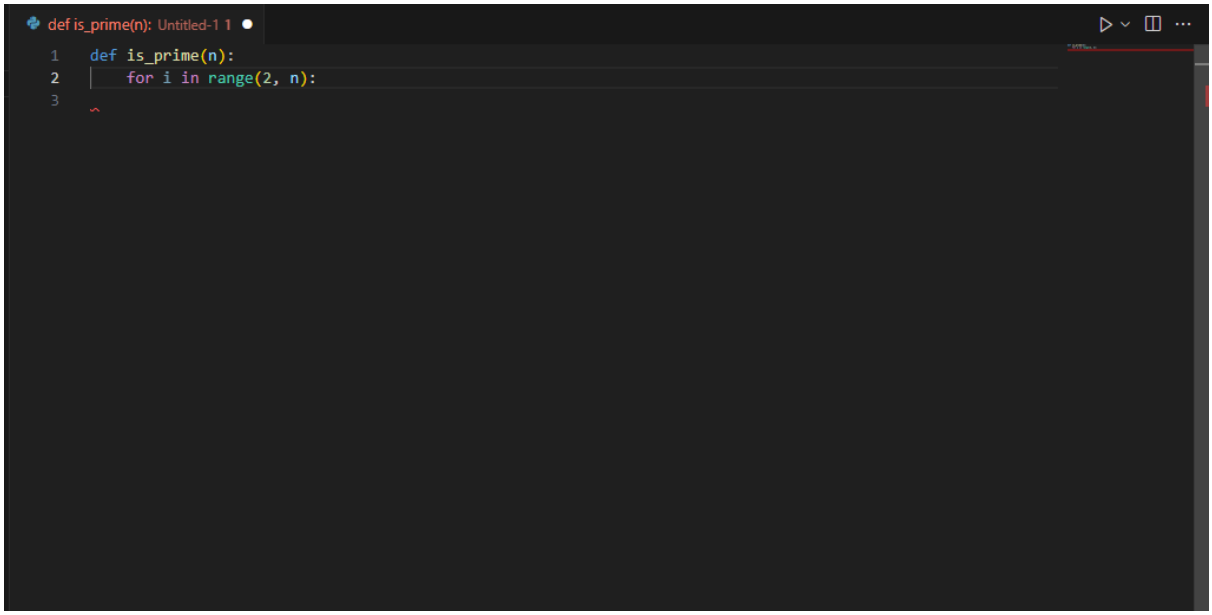


Note: Press tab to accept the function or modify and run it with test values

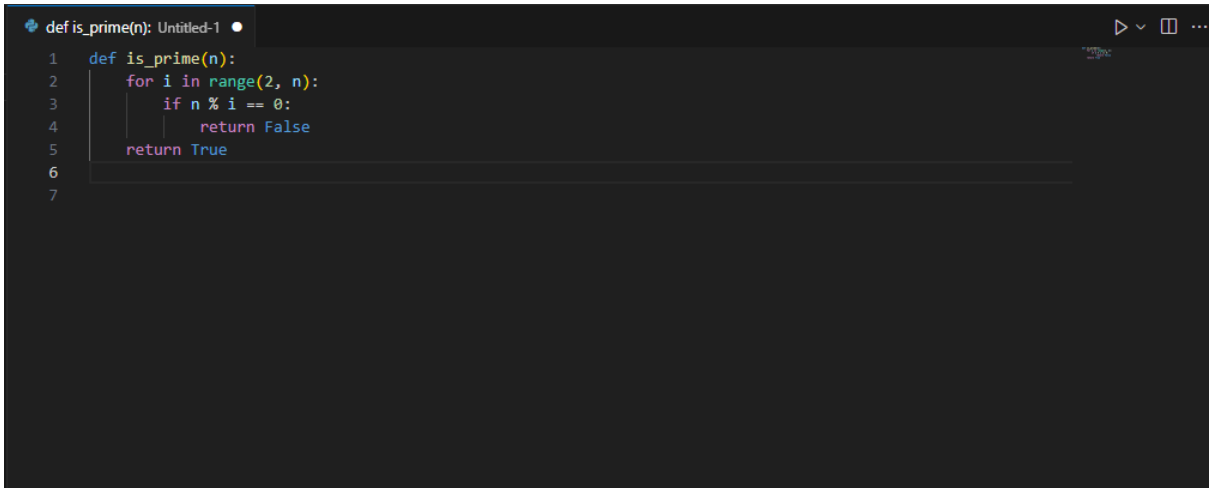
Step 5: Create a function using partial code completion

5.1 Type the function header and start the loop without completing it

Example input:
def is_prime(n):
for i in range(2, n):



5.2 Pause and observe how Copilot suggests the function's body



Note: Press tab to accept the function or modify and run it with test values

By following these steps, you have successfully tested different prompt styles and understood that structured docstrings and contextual hints lead to more accurate and optimized AI-generated code, while inline comments and partial code provide only basic

completions. Experimenting with these styles helps developers optimize AI-assisted coding for better accuracy, readability, and maintainability.