**Task 1: Zero-shot Prompt – Fibonacci Series Generator**

**Task Description #1**

• Without giving an example, write a single comment prompt asking GitHub Copilot to generate a Python function to print the first N Fibonacci numbers.

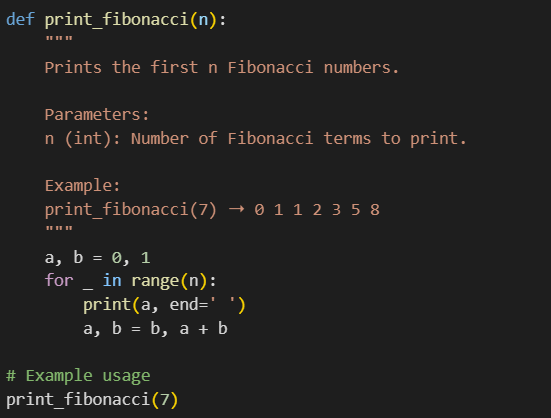
**Expected Output #1**

**•** A complete Python function generated by Copilot without any example provided.

• Correct output for sample input N = 7 ➝ 0 1 1 2 3 5 8

• Observation on how Copilot understood the instruction with zero context.

Code:



Output:



Explanation:

a and b start as 0 and 1 , the first two Fibonacci numbers.

• The for loop runs n times (how many numbers you want).

• Each time:

• It prints the current value of a .

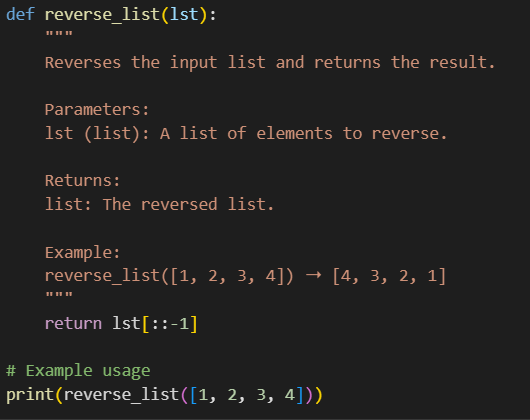
• Then updates a and b to the next two numbers in the sequence.

**Task 2: One-shot Prompt – List Reversal Function**

**Task Description #2**

• Write a comment prompt to reverse a list and provide one example below the comment to guide Copilot

Code:



Output:



Explanation:

def reverse\_list(lst): """ Reverses the input list and returns the result. Parameters: lst (list): A list of elements to reverse. Returns: list: The reversed list. Example: reverse\_list([1, 2, 3, 4]) ➝ [4, 3, 2, 1] Explanation: This function uses Python's slicing syntax lst[::-1] to reverse the list. - lst[:] means "take all elements" - The step -1 means "go backwards" So lst[::-1] grabs the list from end to start, flipping the order. It's a clean and efficient way to reverse any list. """ return lst[::-1] # Example usage print(reverse\_list([1, 2, 3, 4])) # Output: [4, 3, 2, 1]

**Task 3: Few-shot Prompt – String Pattern Matching**

**Task Description #3**

• Write a comment with 2–3 examples to help Copilot understand how to check if a string starts with a capital letter and ends with a period.

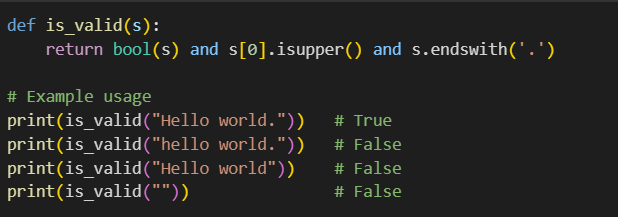
**Expected Output #3**

• A function is\_valid() that checks the pattern.

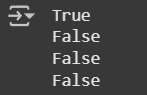
• Output: True or False based on input.

• Students reflect on how multiple examples guide Copilot to generate more accurate code.

Code:



Output:



Explanation:

""" Function: is\_valid(s) Purpose: Checks if a string starts with a capital letter and ends with a period. Logic: - bool(s): Ensures the string is not empty. - s[0].isupper(): Checks if the first character is uppercase. - s.endswith('.'): Checks if the string ends with a period. Returns: True if all conditions are met, otherwise False. Examples: is\_valid("Hello world.") ➝ True is\_valid("hello world.") ➝ False is\_valid("Hello world") ➝ False is\_valid("") ➝ False """

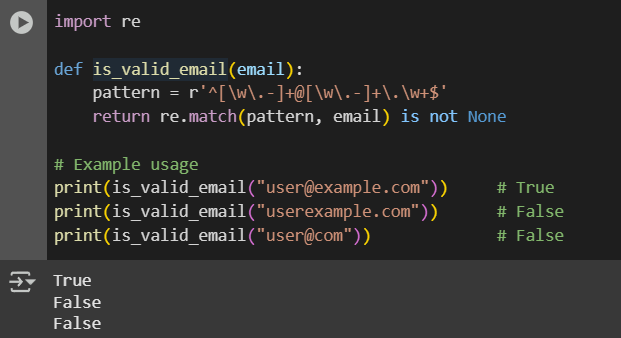
**Task 4: Zero-shot vs Few-shot – Email Validator**

Task Description #4

• First, prompt Copilot to write an email validation function using zero-shot (just the task in comment).

• Then, rewrite the prompt using few-shot examples

Zero shot:



Explanation:

"""

Function: is\_valid\_email(email)

Purpose:

Validates whether a string is a properly formatted email address.

Logic:

- Uses regex to match:

- Starts with letters, digits, dots or hyphens

- Contains '@'

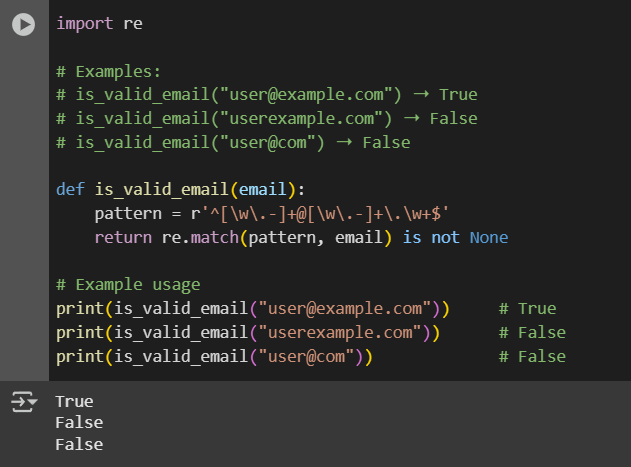
- Followed by domain name and a dot-extension

Returns:

True if the string matches the pattern, otherwise False.

"""

Few shot:



Explanation:

"""

Function: is\_valid\_email(email)

Purpose:

Checks if the input string is a valid email format using regex.

Why Few-shot Works:

- Examples clarify what counts as valid vs invalid.

- Copilot learns to reject missing '@' or domain parts.

- Helps avoid false positives and improves accuracy.

Regex Pattern:

- ^[\w\.-]+ : Starts with word characters, dots, or hyphens

- @ : Must contain '@'

- [\w\.-]+ : Domain name

- \.\w+ : Dot followed by extension (e.g., .com, .org)

Returns:

True if valid, False otherwise.

"""

**Task 5: Prompt Tuning – Summing Digits of a Number**

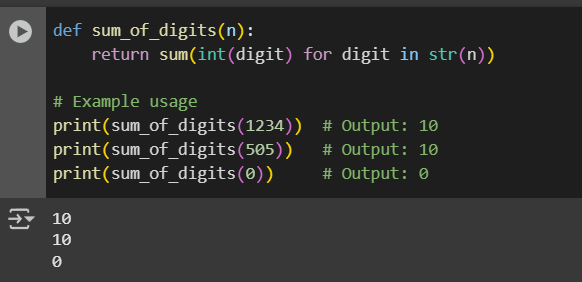
**Task Description #5**

• Experiment with 2 different prompt styles to generate a function that returns the sum of digits of a number.

Style 1: Generic task prompt

Style 2: Task + Input/Output example

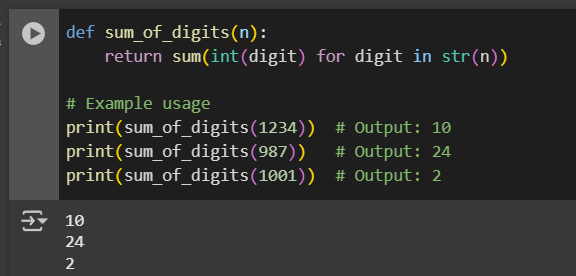
Style1:



Explanation:

""" Function: sum\_of\_digits(n) Purpose: Calculates the sum of all digits in a given number. Logic: - Converts the number to a string: str(n) - Iterates over each character (digit) - Converts each digit back to int and sums them Examples: sum\_of\_digits(1234) ➝ 1 + 2 + 3 + 4 = 10 sum\_of\_digits(505) ➝ 5 + 0 + 5 = 10 sum\_of\_digits(0) ➝ 0 This method is clean and works for any non-negative integer. """

Style2:



Explanation:

""" Function: sum\_of\_digits(n) Purpose: Returns the sum of digits in a number. Why Prompt Tuning Helps: - The example guides Copilot to infer the expected behavior. - It avoids confusion with summing numbers or digits incorrectly. - Ensures the function handles multi-digit numbers properly. Logic: - Convert number to string to access each digit. - Use a generator expression to convert each digit to int. - Sum them using Python’s built-in sum(). Examples: sum\_of\_digits(1234) ➝ 10 sum\_of\_digits(987) ➝ 9 + 8 + 7 = 24 sum\_of\_digits(1001) ➝ 1 + 0 + 0 + 1 = 2 """