AI ASSISTED CODING

ASSIGNMENT-3.4

Name: V. Vyshnavi

Hallticket No.:2303A51968

Batch-14

Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques

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.

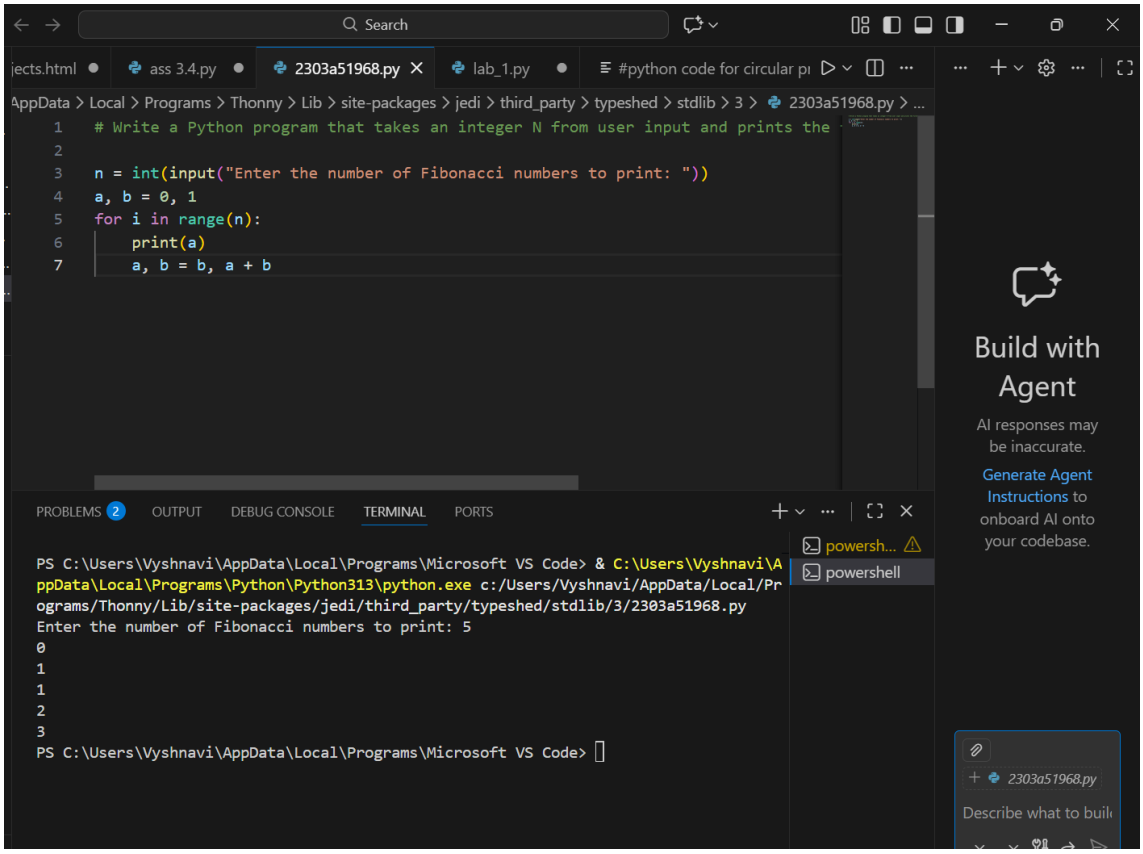
• Correct output for sample input N = 7 → 0 1 1 2 3 5 8

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

Prompt:

Write a Python program that takes an integer N from user input and prints the first N Fibonacci numbers starting from 0

Code and output:



Explanation:

The Fibonacci series is a sequence of numbers where each number is the sum of the two preceding numbers, starting from 0 and 1.
In this program, the user enters a number **N**, and the program generates and prints the first **N Fibonacci numbers**.
The program initializes the first two Fibonacci values and uses a loop to calculate subsequent numbers and display them in a single line separated by spaces.

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.

Expected Output #2

• Copilot-generated function to reverse a list using slicing or loop.

• Output: [3, 2, 1] for input [1, 2, 3]

• Observation on how adding a single example improved Copilot's accuracy.

Prompt:

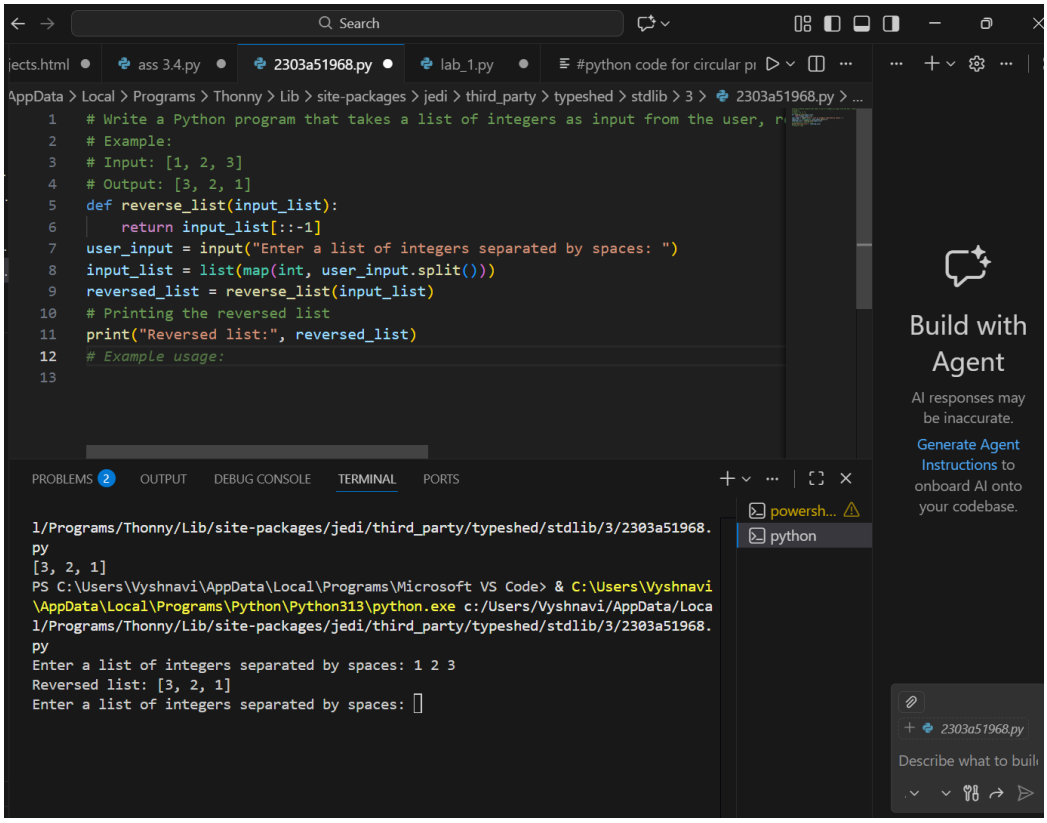
#Write a Python program that takes a list of integers as input from the user, reverses the list using a function (slicing or loop), and prints the reversed list.

Example:

Input: [1, 2, 3]

Output: [3, 2, 1]

Code and output:



Explanation:

input() takes the list from the user as a string.
eval() converts the input string into a Python list.
The list is stored in the variable lst.
lst[::-1] reverses the list using slicing.
-1 as the step value means the list is read from last to first.
print() displays the reversed list as output.

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.

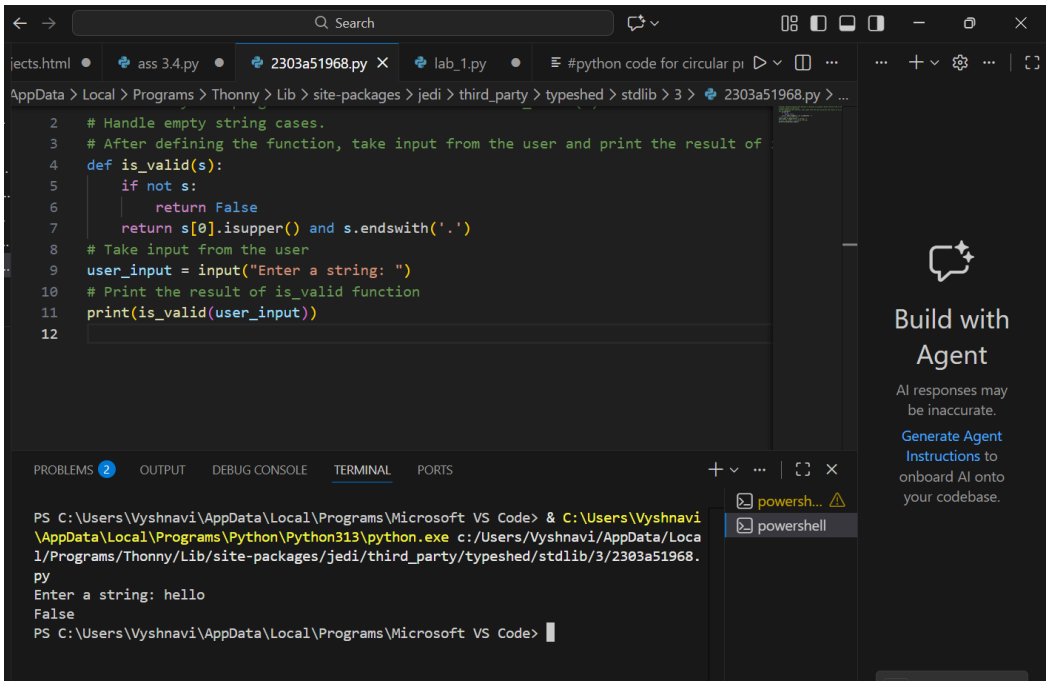
Prompt:

Write a Python program that defines a function is_valid(s) which returns True if the string starts with a capital letter and ends with a period.

Handle empty string cases.

After defining the function, take input from the user and print the result of is_valid(s).

Code and output:



Explanation:

- The function is_valid(s) checks whether a given string satisfies two conditions.
- First, it checks if the string is empty. If it is empty, the function returns False because an empty string cannot start with a capital letter or end with a period.
- The function then checks the first character of the string using isupper() to verify that it starts with a capital letter.
- Next, it uses the endswith('.') method to check whether the string ends with a period.
- Both conditions are combined using the logical and operator.
- If both conditions are true, the function returns True; otherwise, it returns False.
- After defining the function, the program takes input from the user.
- Finally, it prints the result returned by the is_valid(s) function.

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.

Expected Output #4

• Compare both outputs:

Zero-shot may result in basic or generic validation.

Few-shot gives detailed and specific logic (e.g., @ and domain

checking).

• Submit both code versions and note how few-shot improves

reliability

Prompt:

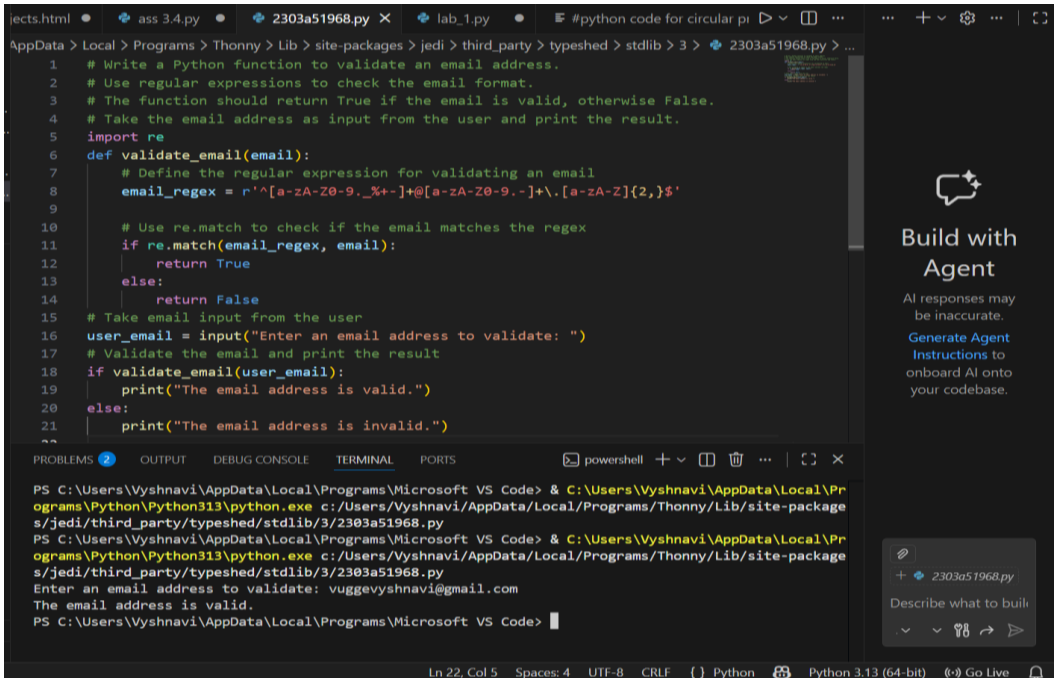
Write a Python function to validate an email address.

Use regular expressions to check the email format.

The function should return True if the email is valid, otherwise False.

Take the email address as input from the user and print the result.

Code and output:



Explanation:

- The re module is used for email pattern matching.
- The function is_valid_email() checks whether the email format is valid.
- A regular expression defines the correct email structure.
- re.match() verifies the email against the pattern.
- The function returns **True** for a valid email and **False** otherwise.
- The email is taken as input and the result is printed.

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

Expected Output #5

• Two versions of the sum_of_digits() function.

• Example Output: sum_of_digits(123) → 6

• Short analysis: which prompt produced cleaner or more

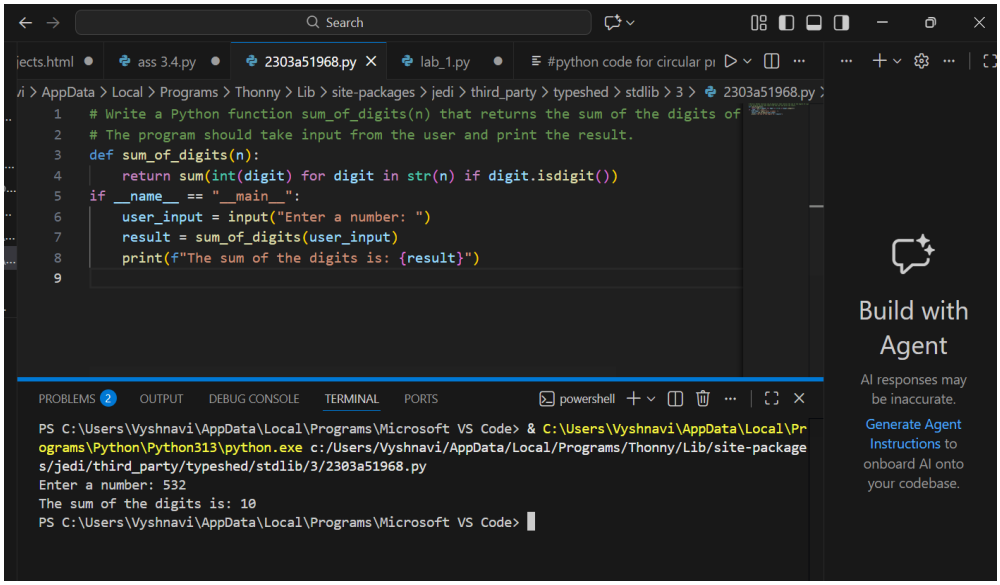
optimized code and why?

Prompt:

Write a Python function sum_of_digits(n) that returns the sum of the digits of a given number.

The program should take input from the user and print the result.

Code and output:



Explanation:

- The function sum_of_digits() calculates the total of all digits in a given number.
- The number is taken as input from the user.
- Each digit is extracted using modulo (%) and added to a sum variable.
- Integer division (//) is used to remove the last digit after each step.
- Finally, the total sum of digits is printed.