ASSIGNMENT-5

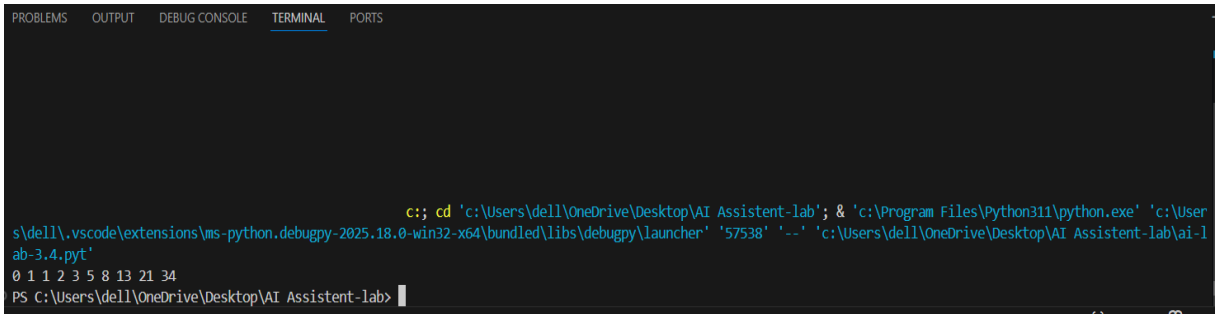
Task-1: Zero-shot Prompt – Fibonacci Series Generator

Prompt: #write a python function to print the to print the first N Fibonacci numbers.

Code and Output:

```python
#write a python program function to print the frist n Fibonacci numbers.
def print_fibonacci(n):
    a, b = 0, 1
    for _ in range(n):
        print(a, end=' ')
        a, b = b, a + b
# Example usage:
n = 10
print_fibonacci(n)

```



Justification:

- Copilot correctly understood the task without any example
- Generated a valid Fibonacci logic using iteration
- Shows that zero-shot prompting works well for common problems

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

Prompt: #write a python function to reverse a list

#Example: Input: [3,2,1] Output: [1,2,3]

Code and Output:

```python
#write a python fuction to reverse a list using slicing or loop
def reverse_list(lst):
    # Using slicing to reverse the list
    return lst[::-1]
# Example usage:
example_list = [1, 2, 3]
reversed_list = reverse_list(example_list)
print(f"Original list: {example_list}")
print(f"Reversed list: {reversed_list}")
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

c:; cd 'c:\Users\dell\OneDrive\Desktop\AI Assistent-lab'; & 'c:\Program Files\Python311\python.exe' 'c:\Users\dell\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '52781' '--' 'c:\Users\dell\OneDrive\Desktop\AI Assistent-lab\ai la-3.4.pyt'
Original list: [1, 2, 3]
Reversed list: [3, 2, 1]
PS C:\Users\dell\OneDrive\Desktop\AI Assistent-lab>
```

Justification:

- Adding one example improved accuracy
- Copilot used Python slicing, which is clean and efficient
- One-shot prompts guide Copilot toward better solutions

Task-3: String Pattern Matching

Prompt:

#write a python python function o check if a string starts with a capital letter and ends with a period.

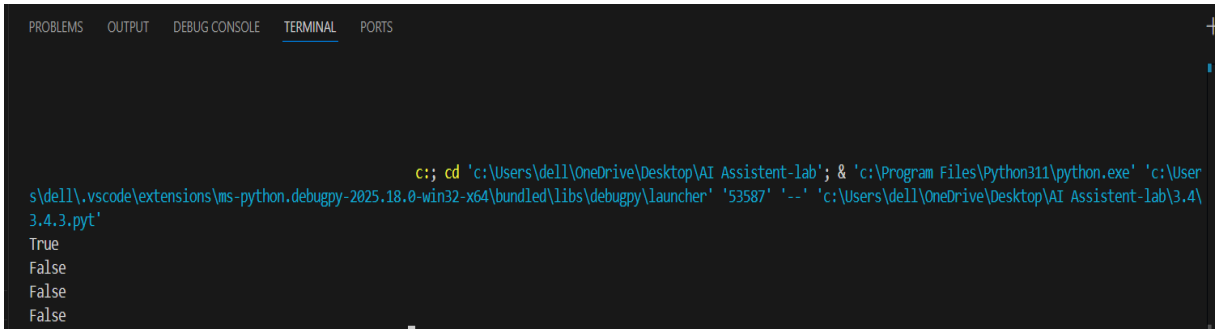
#Example: Input: "Coding." Output: True

#Example: Input: "coding." Output: False

#Example: Input: "Coding" Output: False

Code and Output:

```python
#write a python fuction to check if a string starts with a capital letter and ends with a period
def check_string_format(s):
    if s and s[0].isupper() and s.endswith('.'):
        return True
    return False
# Example usage:
print(check_string_format("Hello world."))
print(check_string_format("hello world."))
print(check_string_format("Hello world"))
print(check_string_format(""))
```



Justification:

- Multiple examples helped Copilot understand the pattern clearly
- Few-shot prompting produces more accurate and specific logic

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

Prompt for zero-shot: write a python function to check an email validation.

Code and Output:

```python
#write a python function to check an email validation function using zero-shot
import re
def is_valid_email(email):
    # Define a regex pattern for validating an email
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    # Use re.match to check if the email matches the pattern
    if re.match(pattern, email):
        return True
    return False
# Example usage:
print(is_valid_email("test@example.com"))  # True
print(is_valid_email("invalid-email"))     # False
print(is_valid_email("test@example"))      # False
print(is_valid_email("test@.com"))         # False
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\dell\OneDrive\Desktop\AI Assistent-lab> c:; cd 'c:\Users\dell\OneDrive\Desktop\AI Assistent-lab'; & 'c:\Program Files\Python311\python.exe' 'c:\Users\dell\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58874' '--' 'c:\Users\dell\OneDrive\Desktop\AI Assistent-lab\3.4\3.4.4.pyt'
True
False
False
False
```

Prompt for few-shot: #write a python function to check an email validation.

#Example: Input: "coding@gmail.com" Output: True

#Example: Input: "codinggmail.com" Output: False

#Example: Input: "coding@gmailcom" Output: False

Code and Output:

```python
3.4(emailValidation).py > ...
#write a python function to check an email validation.
#Example: Input: "coding@gmail.com" Output: True
#Example: Input: "codinggmail.com" Output: False
#Example: Input: "coding@gmailcom" Output: False
import re
def validate_email(email):
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    if re.match(pattern, email):
        return True
    else:
        return False
# Get user input
user_input = input("Enter an email address: ")
# Validate the email and print the result
is_valid = validate_email(user_input)
print(is_valid)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Python

PS C:\Users\akhil\OneDrive\Documents\AI(vs)> & C:/Users/akhil/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/akhil/OneDrive/Documents/AI(vs)/3.4(emailValidation).py"
Enter an email address: coding@gmail.com
True
PS C:\Users\akhil\OneDrive\Documents\AI(vs)> & C:/Users/akhil/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/akhil/OneDrive/Documents/AI(vs)/3.4(emailValidation).py"
Enter an email address: codinggmail.com
False
PS C:\Users\akhil\OneDrive\Documents\AI(vs)> & C:/Users/akhil/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/akhil/OneDrive/Documents/AI(vs)/3.4(emailValidation).py"
Enter an email address: coding
False
PS C:\Users\akhil\OneDrive\Documents\AI(vs)>
```

Justification:

- Zero-shot prompting gives basic and incomplete validation because no examples are provided.

- Few-shot prompting gives better logic and more accurate results by checking username and domain using examples.

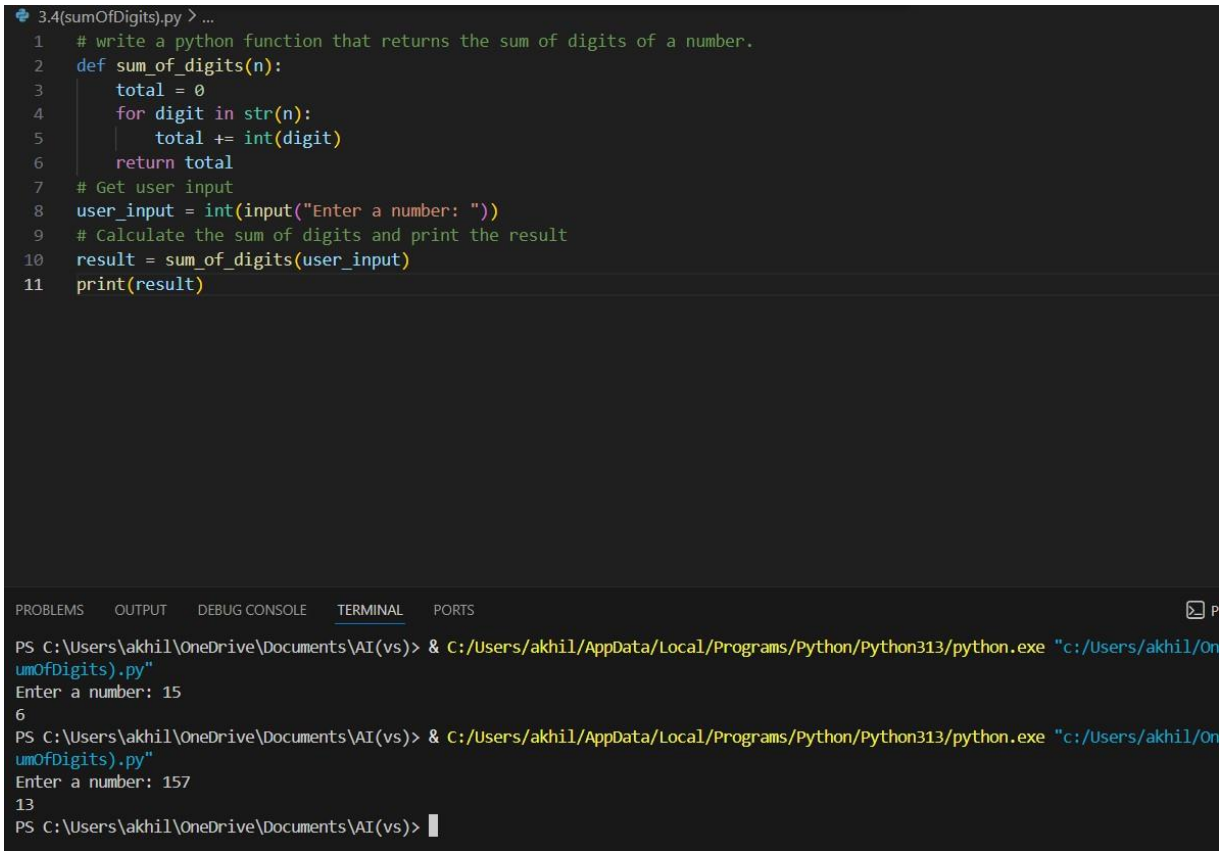
Zero-shot -- Basic, incomplete validation

Few-shot -- Better logic, checks username & domain

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

Prompt1: write a python function that returns the sum of digits of a number.

Code and Output:

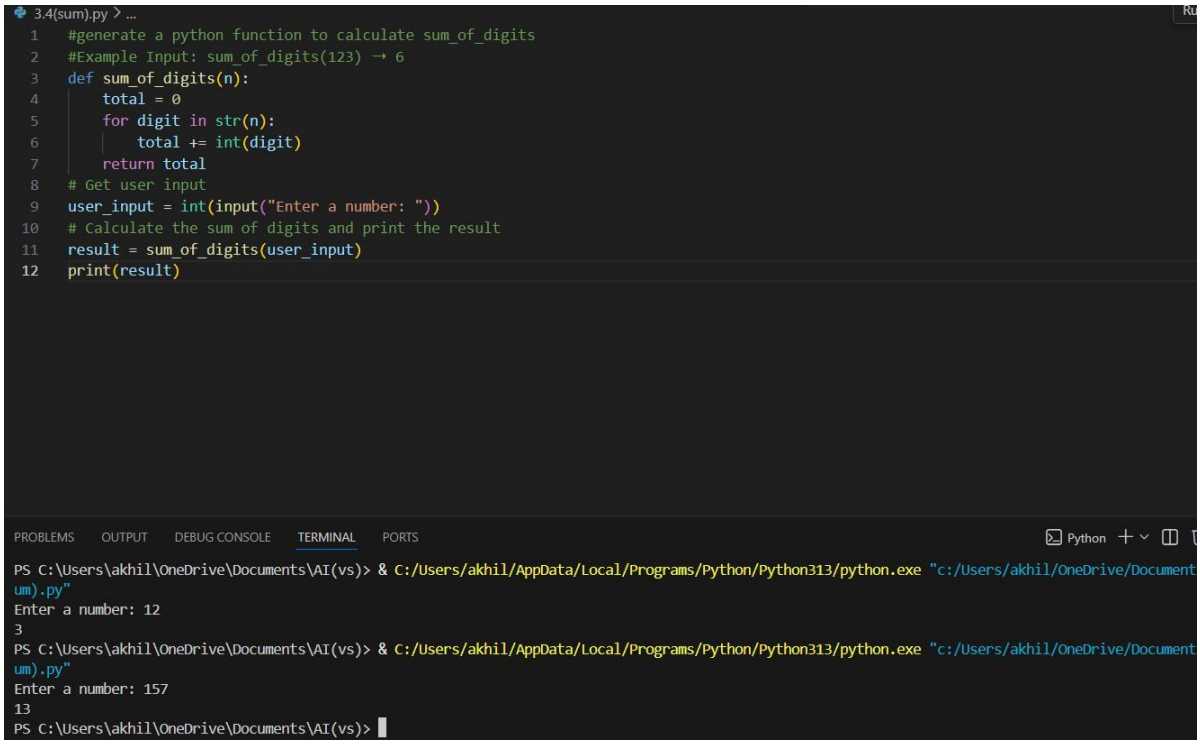


0

Prompt2: #generate a python function to calculate sum_of_digits

#Example Input: sum_of_digits(123) → 6

Code and Output:



Justification:

- Prompt2 produced cleaner and optimized code
- Example guided Copilot to use Pythonic one-line solution
- Prompt tuning improves code quality and readability