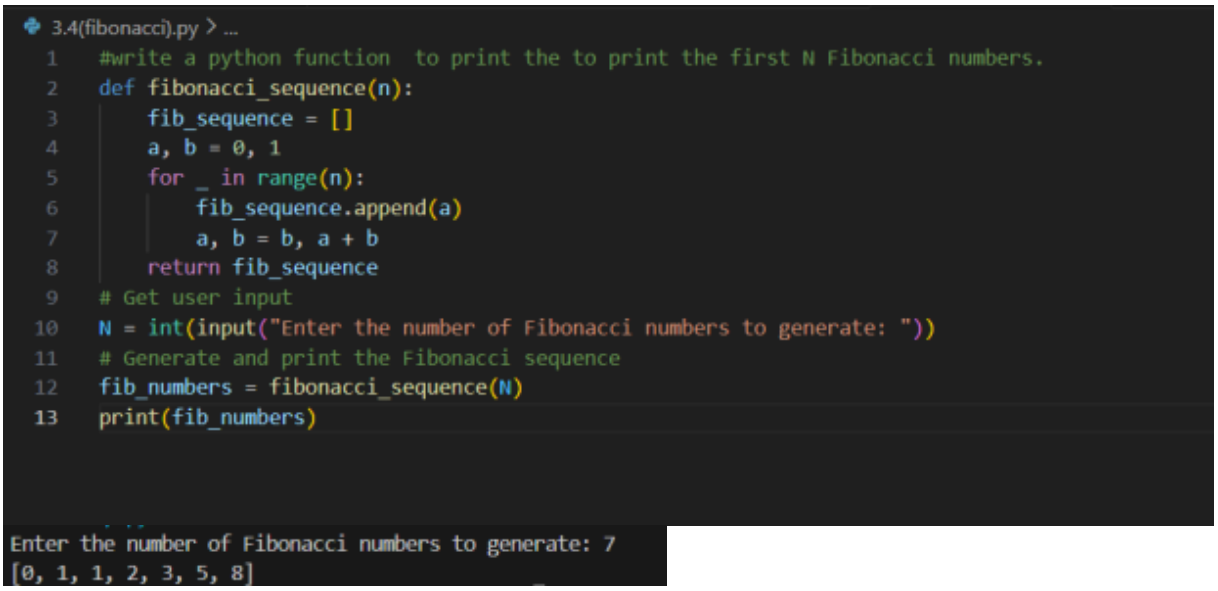
ASSIGNMENT-5

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:



```
3.4(fibonacci).py > ...
1   #write a python function  to print the to print the first N Fibonacci numbers.
2   def fibonacci_sequence(n):
3       fib_sequence = []
4       a, b = 0, 1
5       for _ in range(n):
6           fib_sequence.append(a)
7           a, b = b, a + b
8       return fib_sequence
9   # Get user input
10  N = int(input("Enter the number of Fibonacci numbers to generate: "))
11  # Generate and print the Fibonacci sequence
12  fib_numbers = fibonacci_sequence(N)
13  print(fib_numbers)
```

```
Enter the number of Fibonacci numbers to generate: 7
[0, 1, 1, 2, 3, 5, 8]
```

Justification:

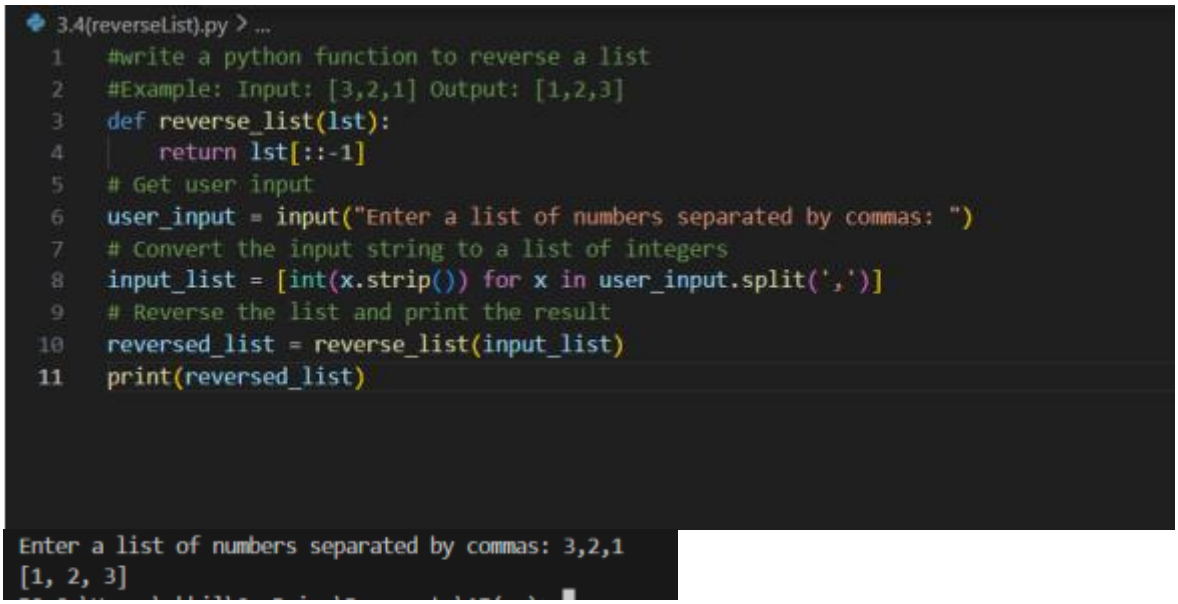
- Copilot correctly understood the task without any example
- Generated a valid Fibonacci logic using iteration
- Shows that zero-shot prompting works well for common problems

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:



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:

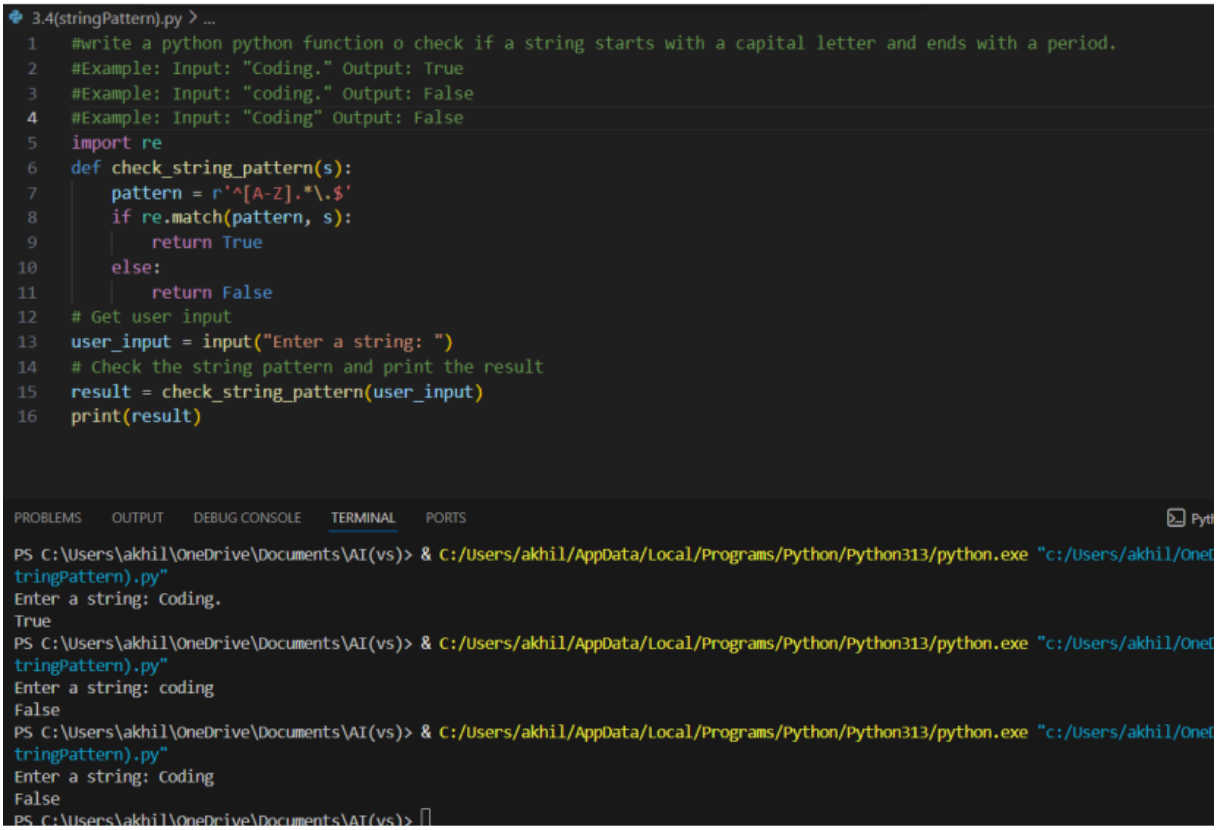
#write a python python function o check if a string starts with a capital letter and ends with a period.

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

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

#Example: Input: "Coding" Output: False

Code and Output:



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.
def is_valid_email(email):
    import re
    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: ")
# Check email validation and print the result
validation_result = is_valid_email(user_input)
print(validation_result)
```

```
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: coding@
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: gmail.com
False
PS C:\Users\akhil\OneDrive\Documents\AI(vs)>
```

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
#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)
```

```
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.

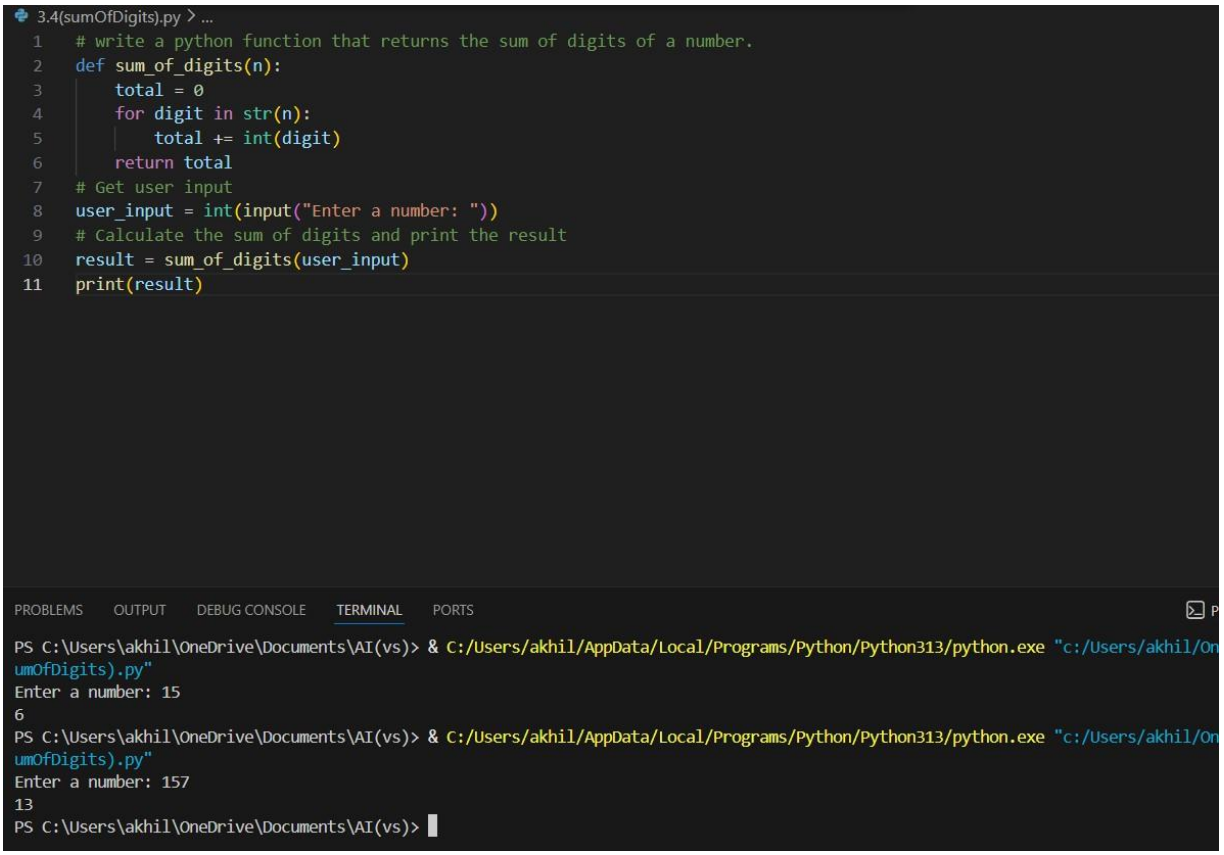
Zero-shot -- Basic, incomplete validation

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:

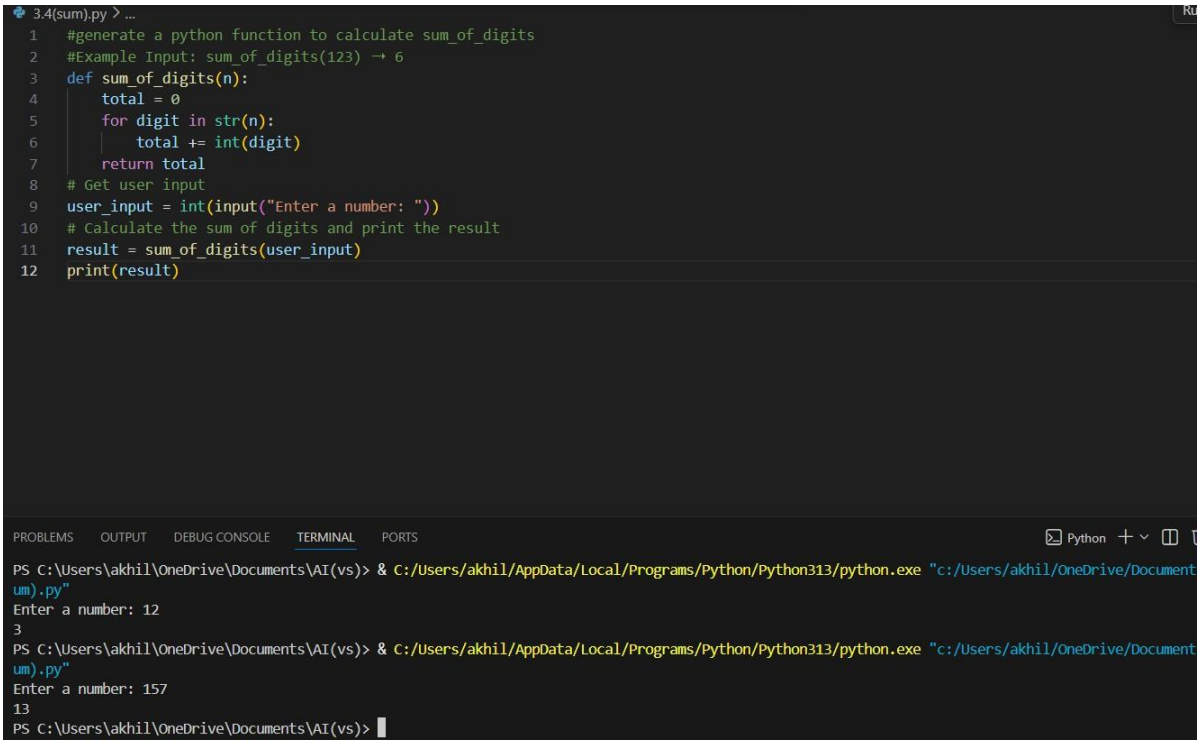


0

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