AI Assisted Coding

Assignment 3.4

Name: P. Vineeth Kumar

Hall ticket no: 2303A51256

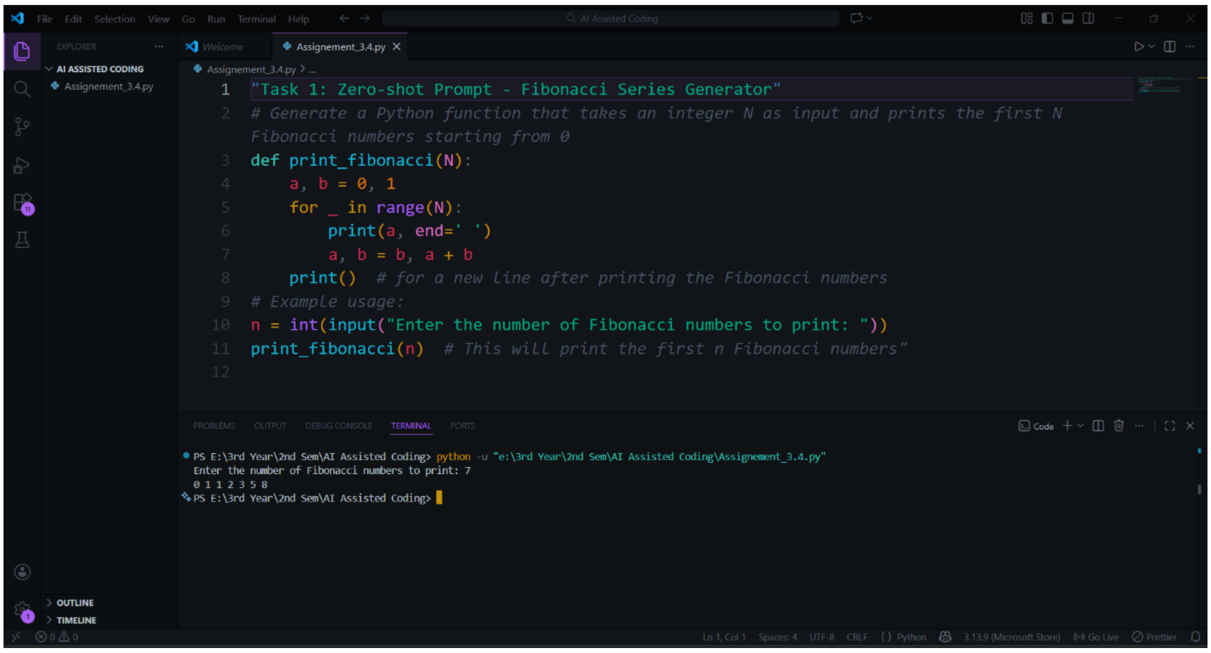
Batch no: 19

Task 1: Zero-shot Prompt – Fibonacci Series Generator

Prompt:

Generate a Python function that takes an integer N as input and prints the first N Fibonacci numbers starting from 0.

Code & Output:



```python
"Task 1: Zero-shot Prompt - Fibonacci Series Generator"
# Generate a Python function that takes an integer N as input and prints the first N Fibonacci numbers starting from 0
def print_fibonacci(N):
    a, b = 0, 1
    for _ in range(N):
        print(a, end=' ')
        a, b = b, a + b
    print()  # for a new line after printing the Fibonacci numbers
# Example usage:
n = int(input("Enter the number of Fibonacci numbers to print: "))
print_fibonacci(n)  # This will print the first n Fibonacci numbers"
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignement_3.4.py"
Enter the number of Fibonacci numbers to print: 7
0 1 1 2 3 5 8
PS E:\3rd Year\2nd Sem\AI Assisted Coding>
```

Explanation:

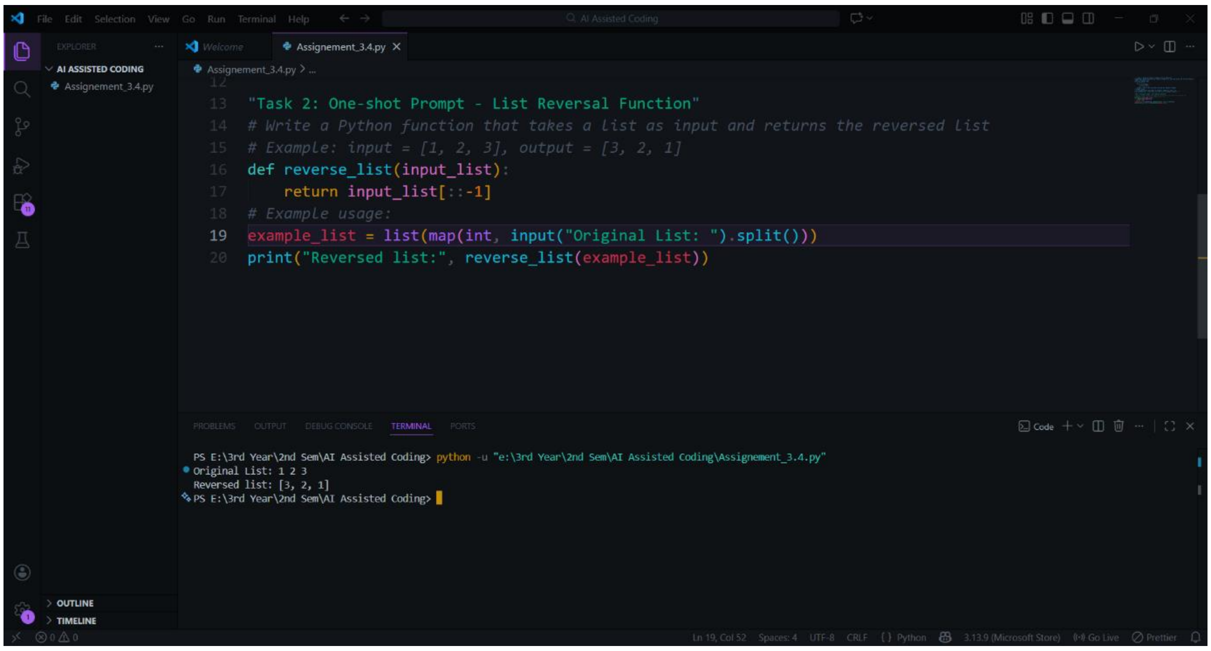
In this task, a zero-shot prompt was used where only the problem description was provided without any examples. Based on this instruction, the AI generated a function to print the Fibonacci series. The program starts with the initial values and iteratively calculates the next numbers in the sequence. This task demonstrates that the AI can correctly understand and solve a problem even when no examples are given.

Task 2: One-shot Prompt – List Reversal Function

Prompt:

Write a Python function that takes a list as input and returns the reversed list.
Example: input = [1, 2, 3], output = [3, 2, 1]

Code & Output:



```python
"Task 2: One-shot Prompt - List Reversal Function"
# Write a Python function that takes a list as input and returns the reversed list
# Example: input = [1, 2, 3], output = [3, 2, 1]
def reverse_list(input_list):
    return input_list[::-1]
# Example usage:
example_list = list(map(int, input("Original List: ").split()))
print("Reversed list:", reverse_list(example_list))
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignement_3.4.py"
Original List: 1 2 3
Reversed list: [3, 2, 1]
PS E:\3rd Year\2nd Sem\AI Assisted Coding>
```

Explanation:

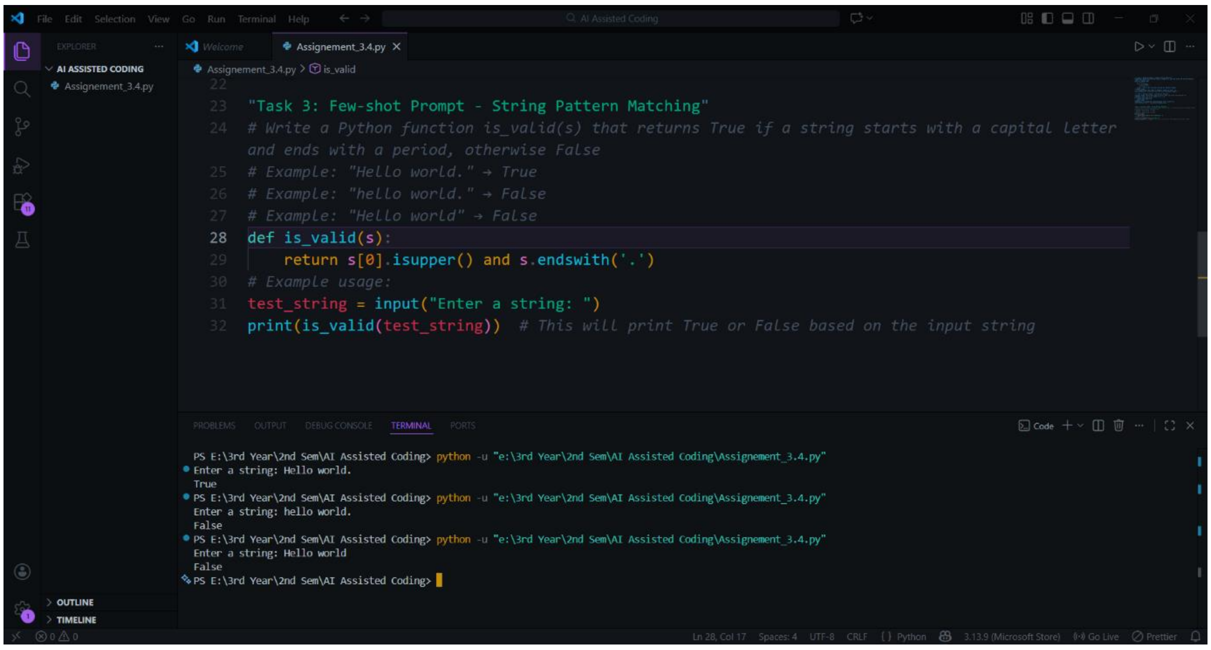
In this task, a one-shot prompt was used by providing a single example along with the task description. The example helped the AI clearly understand the expected input and output format. As a result, the generated solution accurately reverses the list. This shows that adding one example improves the clarity and correctness of the AI-generated code.

Task 3: Few-shot Prompt – String Pattern Matching

Prompt:

Write a Python function is_valid(s) that returns True if a string starts with a capital letter and ends with a period.
Examples:
"Hello world." → True
"hello world." → False
"Hello world" → False

Code & Output:



```python
"Task 3: Few-shot Prompt - String Pattern Matching"
# Write a Python function is_valid(s) that returns True if a string starts with a capital letter and ends with a period, otherwise False
# Example: "Hello world." → True
# Example: "hello world." → False
# Example: "Hello world" → False
def is_valid(s):
    return s[0].isupper() and s.endswith('.')
# Example usage:
test_string = input("Enter a string: ")
print(is_valid(test_string))  # This will print True or False based on the input string
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignement_3.4.py"
Enter a string: Hello world.
True
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignement_3.4.py"
Enter a string: hello world.
False
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignement_3.4.py"
Enter a string: Hello world
False
PS E:\3rd Year\2nd Sem\AI Assisted Coding>
```

Explanation:

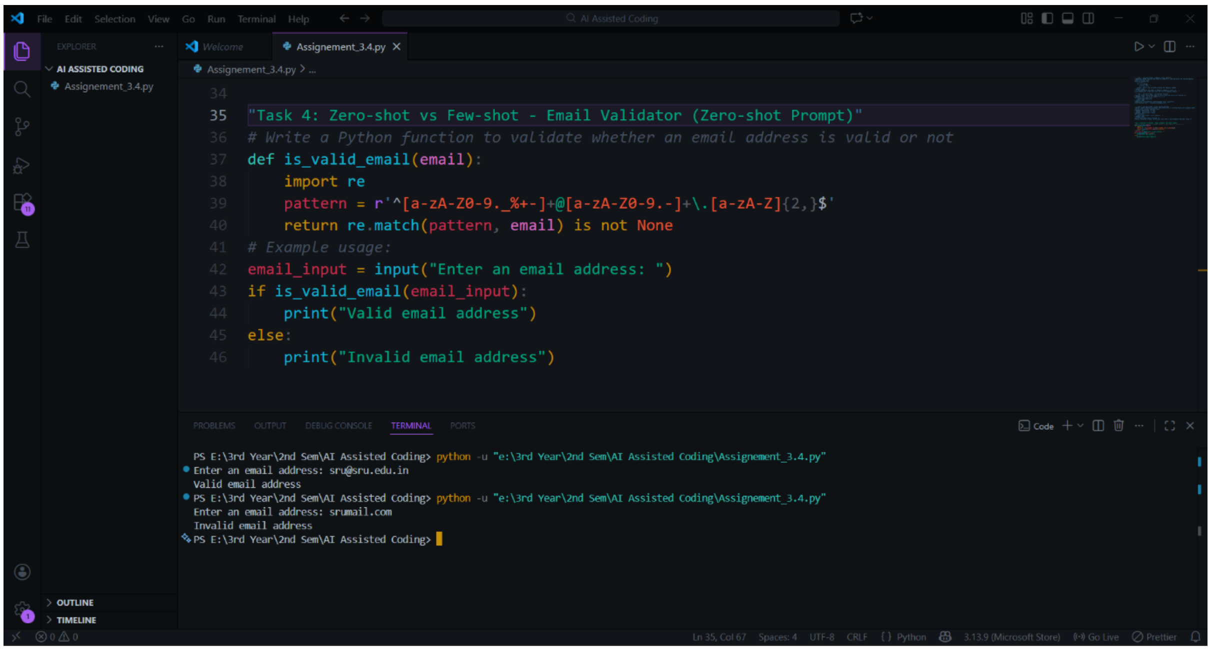
In this task, few-shot prompting was used by providing multiple examples. These examples guided the AI to identify both conditions correctly: the string must start with a capital letter and end with a period. The presence of multiple examples helped the AI generate a more precise and reliable solution compared to zero-shot or one-shot prompting.

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

Zero-shot Prompt:

Write a Python function to validate whether an email address is valid or not.

Code & Output:



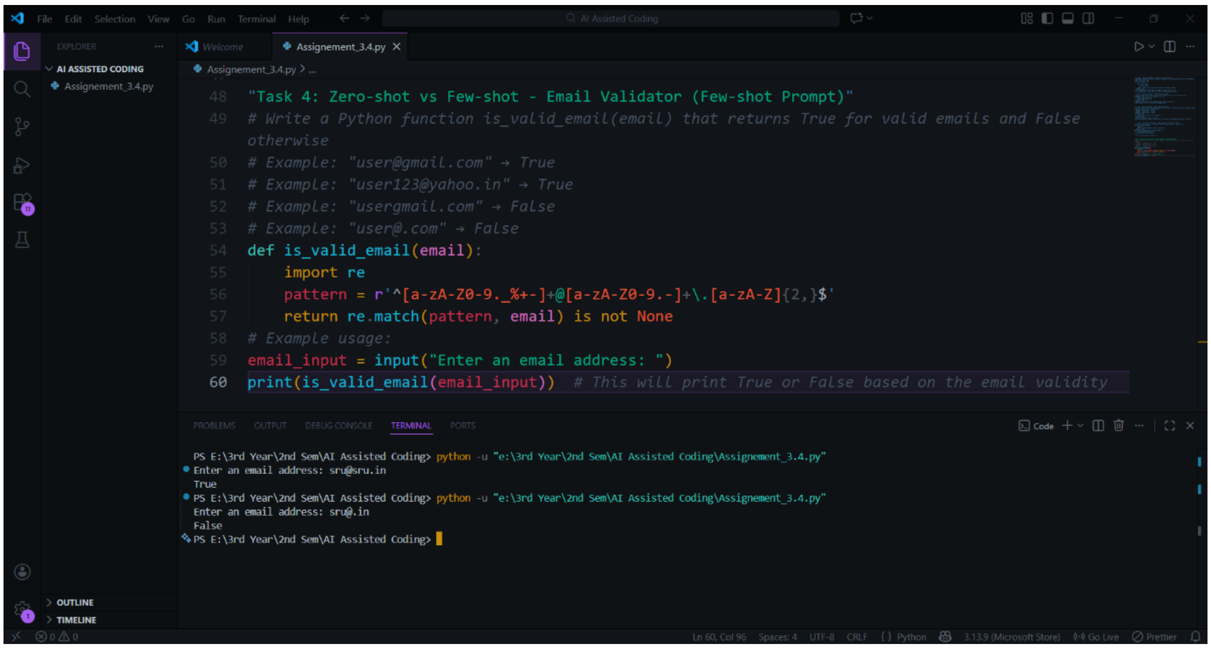
```python
"Task 4: Zero-shot vs Few-shot - Email Validator (Zero-shot Prompt)"
# Write a Python function to validate whether an email address is valid or not
def is_valid_email(email):
    import re
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    return re.match(pattern, email) is not None
# Example usage:
email_input = input("Enter an email address: ")
if is_valid_email(email_input):
    print("Valid email address")
else:
    print("Invalid email address")
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignement_3.4.py"
Enter an email address: sru@sru.edu.in
Valid email address
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignement_3.4.py"
Enter an email address: srumail.com
Invalid email address
PS E:\3rd Year\2nd Sem\AI Assisted Coding>
```

Few-shot Prompt:

Write a Python function is_valid_email(email) that returns True for valid emails and False otherwise.
Examples:
"user@gmail.com" → True
"user123@yahoo.in" → True
"usergmail.com" → False
"user@.com" → False

Code & Output:



```python
"Task 4: Zero-shot vs Few-shot - Email Validator (Few-shot Prompt)"
# Write a Python function is_valid_email(email) that returns True for valid emails and False otherwise
# Example: "user@gmail.com" → True
# Example: "user123@yahoo.in" → True
# Example: "usergmail.com" → False
# Example: "user@.com" → False
def is_valid_email(email):
    import re
    pattern = r'^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$'
    return re.match(pattern, email) is not None
# Example usage:
email_input = input("Enter an email address: ")
print(is_valid_email(email_input))  # This will print True or False based on the email validity
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignement_3.4.py"
Enter an email address: sru@sru.in
True
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignement_3.4.py"
Enter an email address: sru@.in
False
PS E:\3rd Year\2nd Sem\AI Assisted Coding>
```

Explanation:

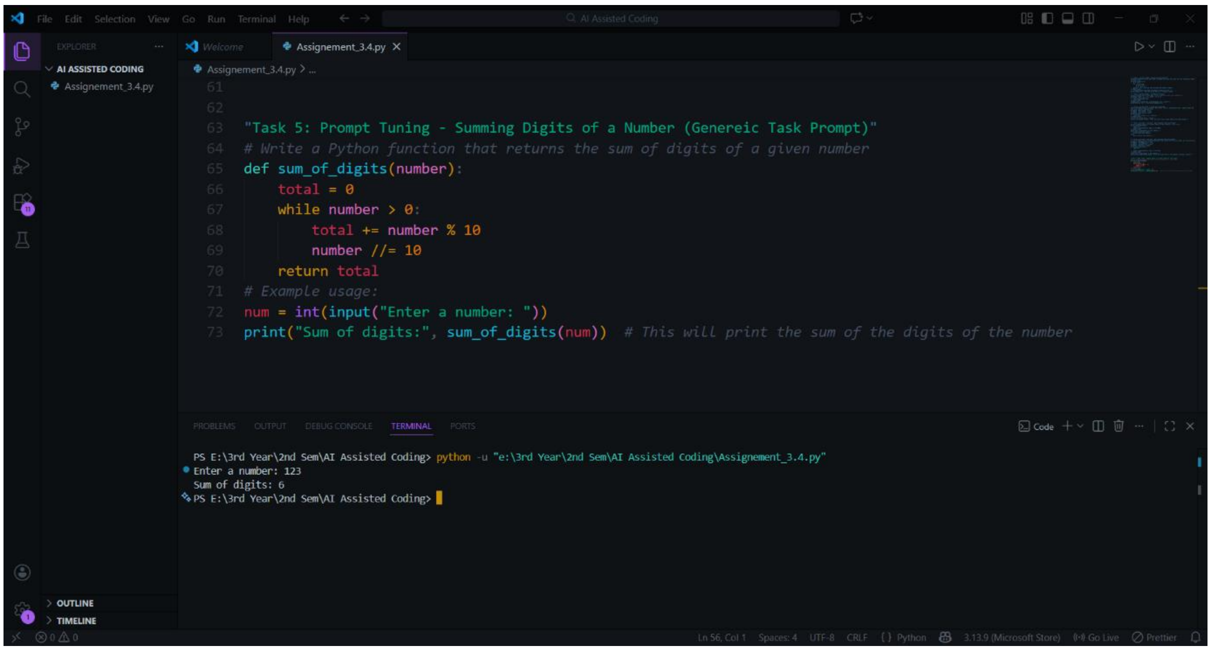
In the zero-shot prompt, the AI produced a basic email validation logic because no examples were provided. In contrast, the few-shot prompt included valid and invalid examples, which helped the AI understand the structure of an email address more clearly. As a result, the few-shot approach generated a more accurate and reliable email validation solution.

Task 5: Prompt Tuning – Summing Digits of a Number

Style 1: Generic Task Prompt

Prompt:
Write a Python function that returns the sum of digits of a given number.

Code & Output:



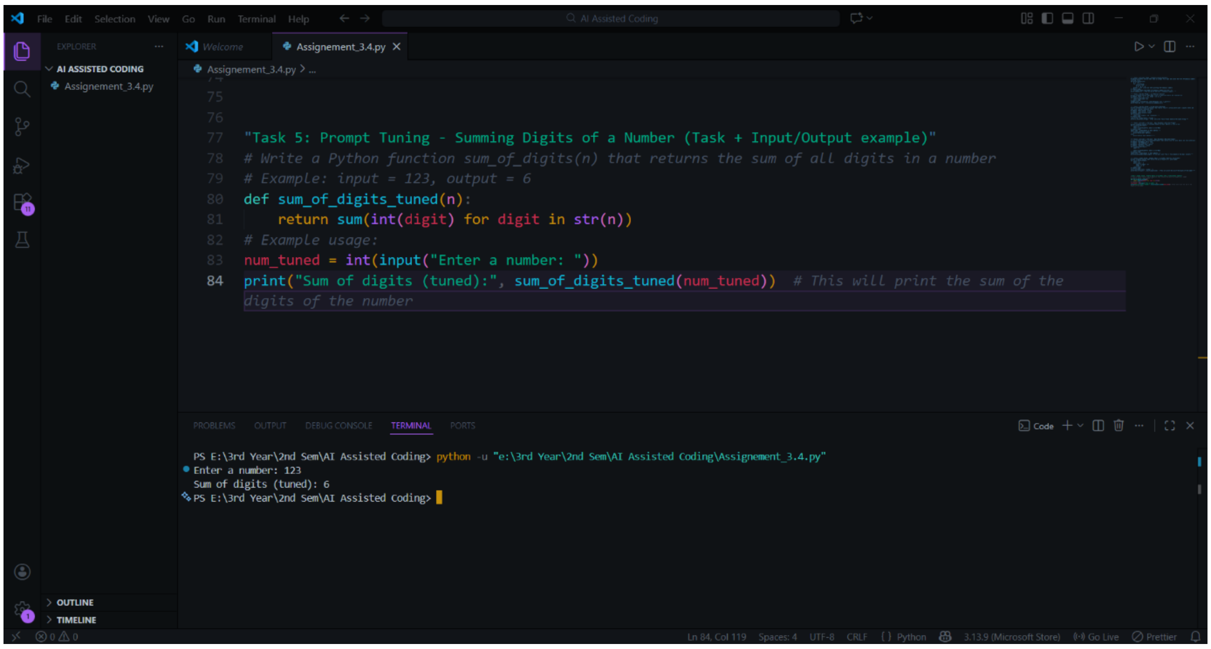
```python
"Task 5: Prompt Tuning - Summing Digits of a Number (Genereic Task Prompt)"
# Write a Python function that returns the sum of digits of a given number
def sum_of_digits(number):
    total = 0
    while number > 0:
        total += number % 10
        number //= 10
    return total
# Example usage:
num = int(input("Enter a number: "))
print("Sum of digits:", sum_of_digits(num))  # This will print the sum of the digits of the number
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignement_3.4.py"
Enter a number: 123
Sum of digits: 6
PS E:\3rd Year\2nd Sem\AI Assisted Coding>
```

Style 2: Task + Input/Output Example Prompt

Prompt:
Write a Python function sum_of_digits(n) that returns the sum of all digits in a number.
Example: input = 123, output = 6

Code & Output:



File Edit Selection View Go Run Terminal Help
AI Assisted Coding
EXPLORER
AI ASSISTED CODING
Assignement_3.4.py
Welcome
Assignement_3.4.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
OUTLINE
TIMELINE
Ln 84, Col 119 Spaces: 4 UTF-8 CRLF Python 3.13.9 (Microsoft Store) Go Live Prettier

```python
"Task 5: Prompt Tuning - Summing Digits of a Number (Task + Input/Output example)"
# Write a Python function sum_of_digits(n) that returns the sum of all digits in a number
# Example: input = 123, output = 6
def sum_of_digits_tuned(n):
    return sum(int(digit) for digit in str(n))
# Example usage:
num_tuned = int(input("Enter a number: "))
print("Sum of digits (tuned):", sum_of_digits_tuned(num_tuned))  # This will print the sum of the digits of the number
```

```
PS E:\3rd Year\2nd Sem\AI Assisted Coding> python -u "e:\3rd Year\2nd Sem\AI Assisted Coding\Assignement_3.4.py"
Enter a number: 123
Sum of digits (tuned): 6
PS E:\3rd Year\2nd Sem\AI Assisted Coding>
```

Explanation:

In this task, two different prompt styles were used. The generic prompt resulted in a straightforward solution, while the prompt with an input/output example produced a cleaner and more optimized implementation. This task highlights how prompt tuning can significantly improve code quality and readability.