Assignment – 4.2

2303A51599

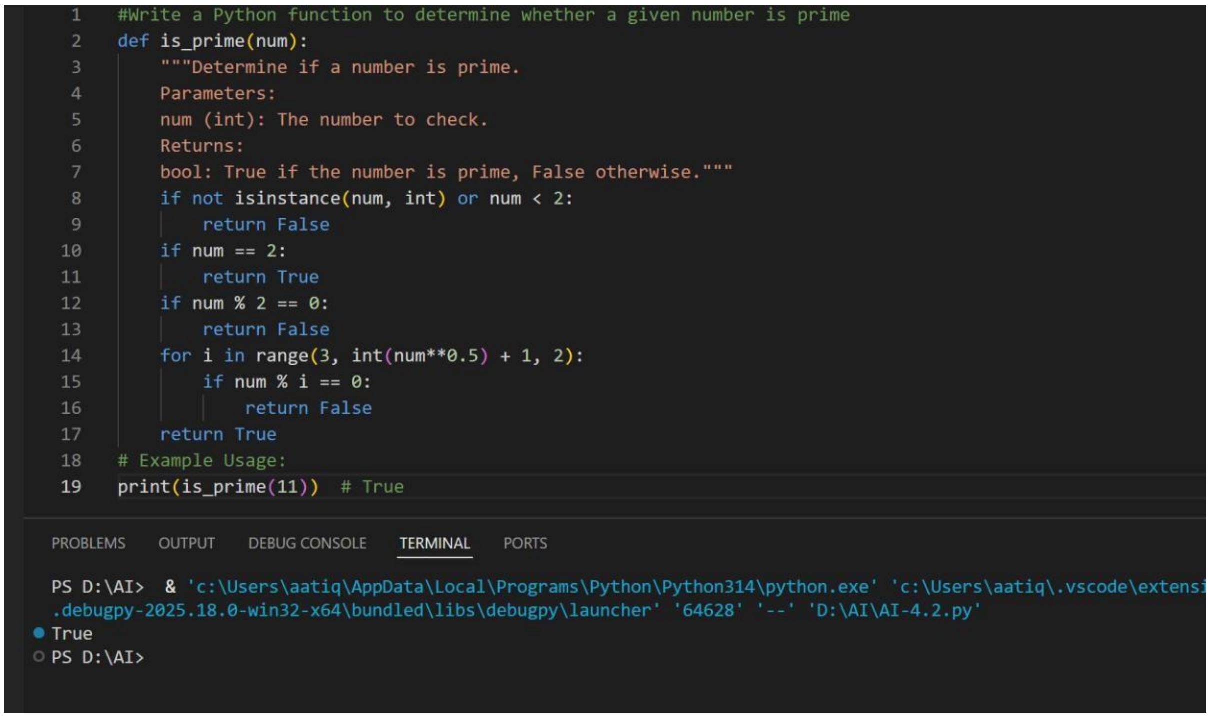
Batch NO:-29

Task 1: Zero-shot: Prompt AI with only the instruction:-

PROMPT :

Write a Python function to determine whether a given number is prime

CODE and OUTPUT :



```python
#Write a Python function to determine whether a given number is prime
def is_prime(num):
    """Determine if a number is prime.
    Parameters:
    num (int): The number to check.
    Returns:
    bool: True if the number is prime, False otherwise."""
    if not isinstance(num, int) or num < 2:
        return False
    if num == 2:
        return True
    if num % 2 == 0:
        return False
    for i in range(3, int(num**0.5) + 1, 2):
        if num % i == 0:
            return False
    return True
# Example Usage:
print(is_prime(11))  # True
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\AI>  & 'c:\Users\aatiq\AppData\Local\Programs\Python\Python314\python.exe' 'c:\Users\aatiq\.vscode\extensi
.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '64628' '--' 'D:\AI\AI-4.2.py'
True
PS D:\AI>
```

Justification :

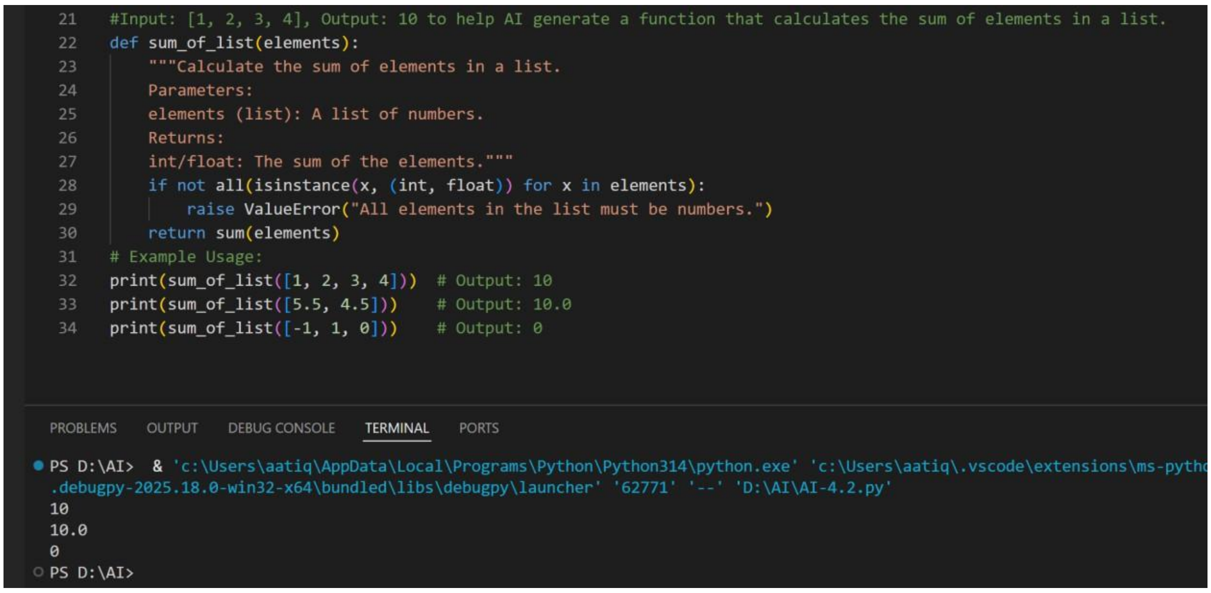
In this task, the AI was given only a clear instruction without any examples. This approach is called zero-shot prompting. Based on its general understanding of mathematics and Python, the AI was able to write a function to check whether a number is prime. The function correctly verifies divisibility and also considers special cases like numbers less than or equal to 1. This shows that zero-shot prompting works well for basic and commonly known problems, although the solution may not always be the most optimized or detailed.

Task 2: One-shot: Provide one example: Input: [1, 2, 3, 4], Output: 10:-

PROMPT :

Input: [1, 2, 3, 4], Output: 10 to help AI generate a function that calculates the sum of elements in a list.

CODE and OUTPUT :



```python
#Input: [1, 2, 3, 4], Output: 10 to help AI generate a function that calculates the sum of elements in a list.
def sum_of_list(elements):
    """Calculate the sum of elements in a list.
    Parameters:
    elements (list): A list of numbers.
    Returns:
    int/float: The sum of the elements."""
    if not all(isinstance(x, (int, float)) for x in elements):
        raise ValueError("All elements in the list must be numbers.")
    return sum(elements)
# Example Usage:
print(sum_of_list([1, 2, 3, 4]))  # Output: 10
print(sum_of_list([5.5, 4.5]))    # Output: 10.0
print(sum_of_list([-1, 1, 0]))    # Output: 0
```

```
PS D:\AI>  & 'c:\Users\aatiq\AppData\Local\Programs\Python\Python314\python.exe' 'c:\Users\aatiq\.vscode\extensions\ms-pytho
.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62771' '--' 'D:\AI\AI-4.2.py'
10
10.0
0
PS D:\AI>
```

Justification :

In this task, one-shot prompting was applied by providing one input-output

example: Input: [1, 2, 3, 4]

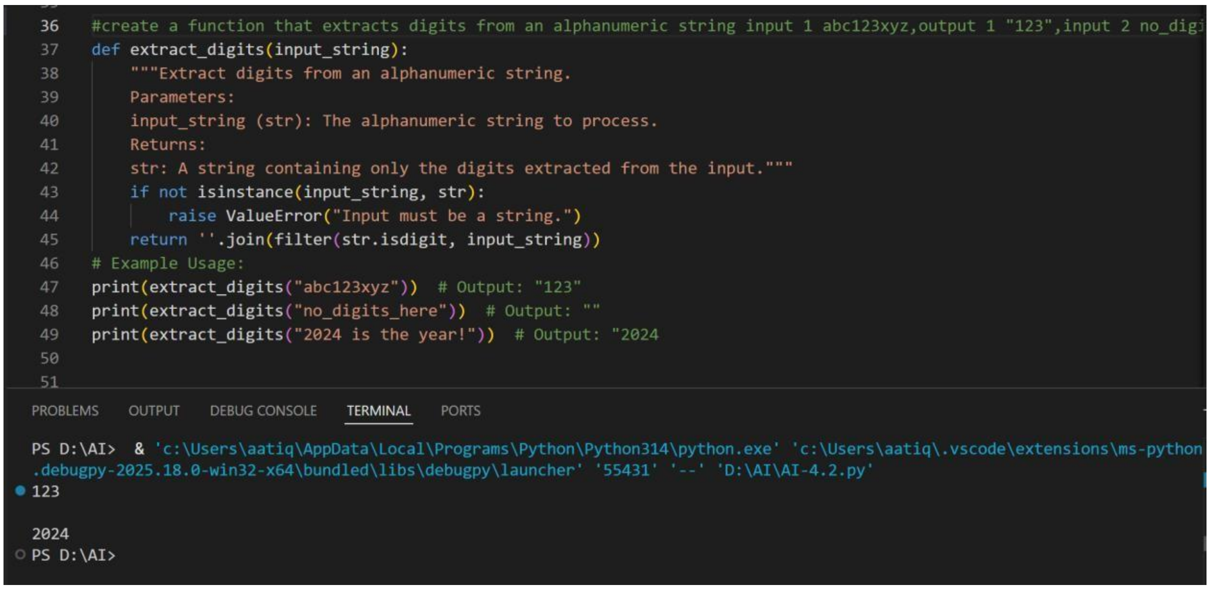
Output: 10

Here, one example input and output were provided before asking the AI to generate the function. This one-shot prompting helped the AI clearly understand that the task was to
compute the sum of elements in a list. With just one example, the AI was able to recognize the expected behavior and produce an accurate solution. This demonstrates that even a single example can significantly reduce confusion and guide the AI toward the correct logic.
Task 3: Few-shot: Give 2–3 examples:-

PROMPT :

create a function that extracts digits from an alphanumeric string input 1 abc123xyz,output 1 "123",input 2 no_digits_here output 2 "".

CODE and OUTPUT :



```python
#create a function that extracts digits from an alphanumeric string input 1 abc123xyz,output 1 "123",input 2 no_digi
def extract_digits(input_string):
    """Extract digits from an alphanumeric string.
    Parameters:
    input_string (str): The alphanumeric string to process.
    Returns:
    str: A string containing only the digits extracted from the input."""
    if not isinstance(input_string, str):
        raise ValueError("Input must be a string.")
    return ''.join(filter(str.isdigit, input_string))
# Example Usage:
print(extract_digits("abc123xyz"))  # Output: "123"
print(extract_digits("no_digits_here"))  # Output: ""
print(extract_digits("2024 is the year!"))  # Output: "2024
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\AI>  & 'c:\Users\aatiq\AppData\Local\Programs\Python\Python314\python.exe' 'c:\Users\aatiq\.vscode\extensions\ms-python
.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55431' '--' 'D:\AI\AI-4.2.py'
123

2024
PS D:\AI>
```

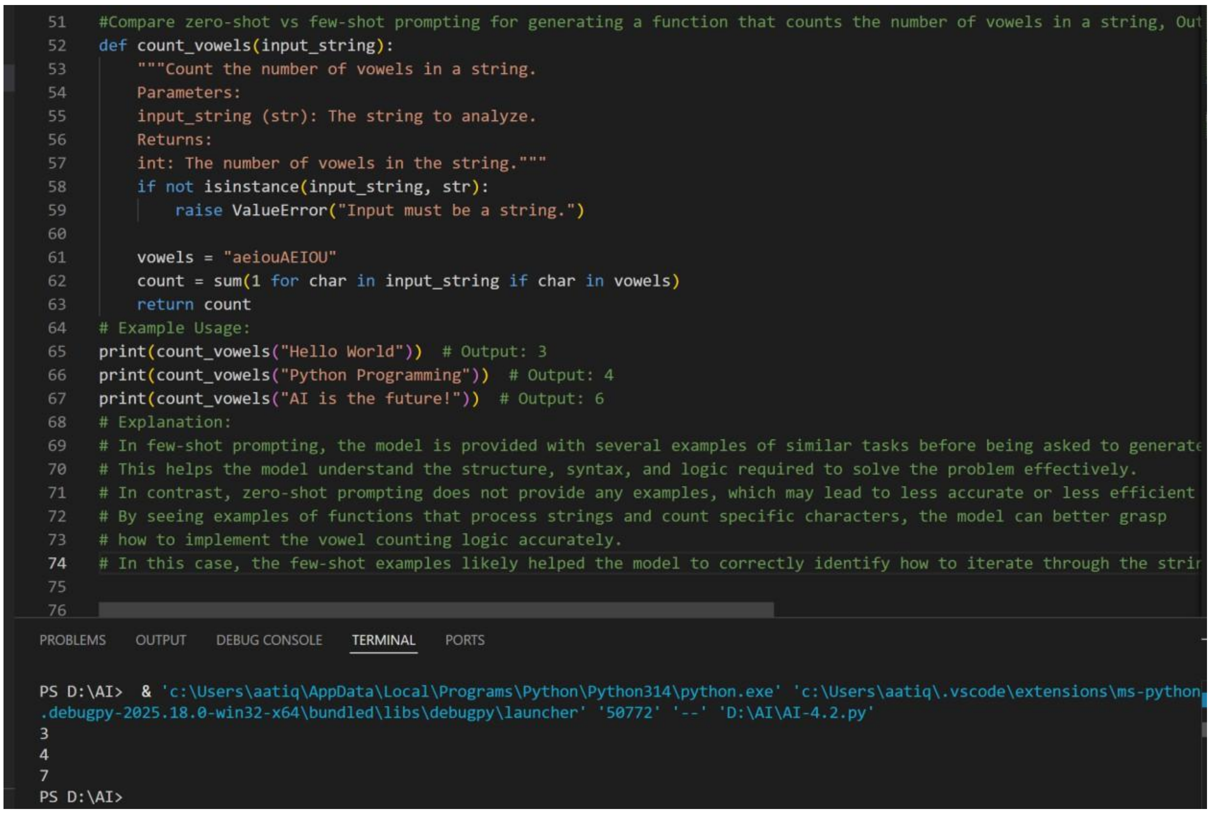
Justification :

In this task, few-shot prompting was used by providing two to three examples showing how digits should be extracted from alphanumeric strings. These examples clearly demonstrated the expected pattern and output format. With multiple examples, the AI was able to
understand that: Only numeric characters should be extracted All other alphabetic or special characters should be ignored As a result, the generated function accurately returns only the digits from the input string. This task proves that few-shot prompting is very effective for pattern-based problems, where examples help the AI generalize the solution correctly.
Task 4: Compare zero-shot vs few-shot prompting

PROMPT :

Compare zero-shot vs few-shot prompting for generating a function that counts the number of vowels in a string, Output comparison + student explanation on how examples helped the model.

CODE and OUTPUT :



```python
#Compare zero-shot vs few-shot prompting for generating a function that counts the number of vowels in a string, Out
def count_vowels(input_string):
    """Count the number of vowels in a string.
    Parameters:
    input_string (str): The string to analyze.
    Returns:
    int: The number of vowels in the string."""
    if not isinstance(input_string, str):
        raise ValueError("Input must be a string.")

    vowels = "aeiouAEIOU"
    count = sum(1 for char in input_string if char in vowels)
    return count
# Example Usage:
print(count_vowels("Hello World"))  # Output: 3
print(count_vowels("Python Programming"))  # Output: 4
print(count_vowels("AI is the future!"))  # Output: 6
# Explanation:
# In few-shot prompting, the model is provided with several examples of similar tasks before being asked to generate
# This helps the model understand the structure, syntax, and logic required to solve the problem effectively.
# In contrast, zero-shot prompting does not provide any examples, which may lead to less accurate or less efficient
# By seeing examples of functions that process strings and count specific characters, the model can better grasp
# how to implement the vowel counting logic accurately.
# In this case, the few-shot examples likely helped the model to correctly identify how to iterate through the strir
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS D:\AI>  & 'c:\Users\aatiq\AppData\Local\Programs\Python\Python314\python.exe' 'c:\Users\aatiq\.vscode\extensions\ms-python
.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '50772' '--' 'D:\AI\AI-4.2.py'
3
4
7
PS D:\AI>
```

Justification :

This task demonstrates the difference between zero-shot and few-shot prompting when counting vowels in a string. With zero-shot prompting, the AI produced a basic solution but did not always consider variations like uppercase letters. In contrast, the few-shot approach included examples that showed how vowels are counted in different situations. These examples helped the AI generate a more complete and accurate function. This comparison clearly shows that providing examples improves the quality and reliability of the output.

Task 5: Use few-shot prompting with 3 sample inputs:-

PROMPT :

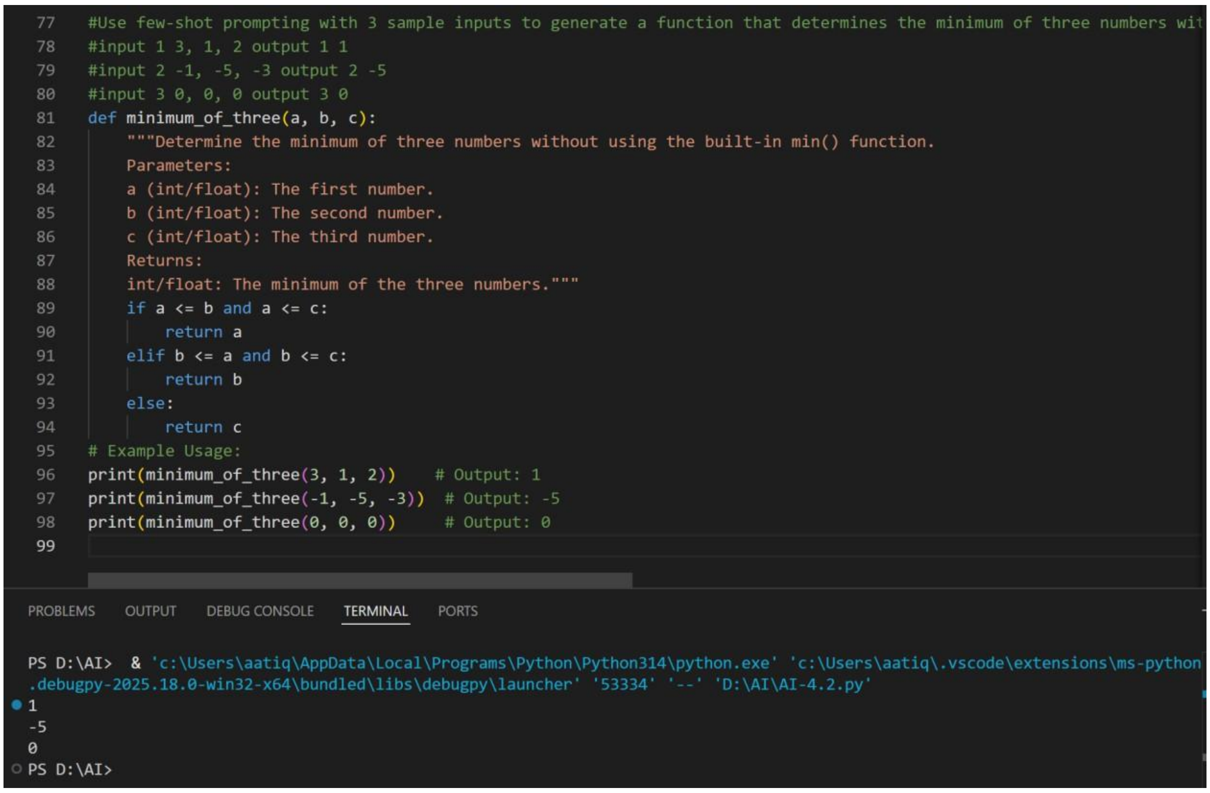
Use few-shot prompting with 3 sample inputs to generate a function that determines the minimum of three numbers without using the built-in min() function,A function that handles all cases with correct logic based on example patterns.

input 1 3, 1, 2 output 1 1

input 2 -1, -5, -3 output 2 -5

input 3 0, 0, 0 output 3 0

CODE and OUTPUT :



```python
#Use few-shot prompting with 3 sample inputs to generate a function that determines the minimum of three numbers wit
#input 1 3, 1, 2 output 1 1
#input 2 -1, -5, -3 output 2 -5
#input 3 0, 0, 0 output 3 0
def minimum_of_three(a, b, c):
    """Determine the minimum of three numbers without using the built-in min() function.
    Parameters:
    a (int/float): The first number.
    b (int/float): The second number.
    c (int/float): The third number.
    Returns:
    int/float: The minimum of the three numbers."""
    if a <= b and a <= c:
        return a
    elif b <= a and b <= c:
        return b
    else:
        return c
# Example Usage:
print(minimum_of_three(3, 1, 2))    # Output: 1
print(minimum_of_three(-1, -5, -3))  # Output: -5
print(minimum_of_three(0, 0, 0))     # Output: 0
```

```
PS D:\AI>  & 'c:\Users\aatiq\AppData\Local\Programs\Python\Python314\python.exe' 'c:\Users\aatiq\.vscode\extensions\ms-python
.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53334' '--' 'D:\AI\AI-4.2.py'
1
-5
0
PS D:\AI>
```

Justification :

In this task, three sample inputs were provided to guide the AI in writing a function that finds the minimum of three numbers without using the built-in min() function. The examples helped the AI understand how to compare values in different cases, including negative numbers and equal values. Based on these patterns, the AI implemented correct conditional logic. This task proves that few-shot prompting helps the model learn logical decision-making directly from examples.