Assignment – 4.2

2303A510I9

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 :

C: > Users > Ramya Sri > Downloads > # Write a Python function to determine w.py > ...

```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

```

Output:

```
determine w.py"
PS C:\Users\Ramya Sri\Desktop\Devoops 2026\20-01-2026> & "C:/Users/Ramya Sri/AppData/Local/Programs/Python
"c:/Users/Ramya Sri/Downloads/# Write a Python function to determine w.py"
True
```

Justification :

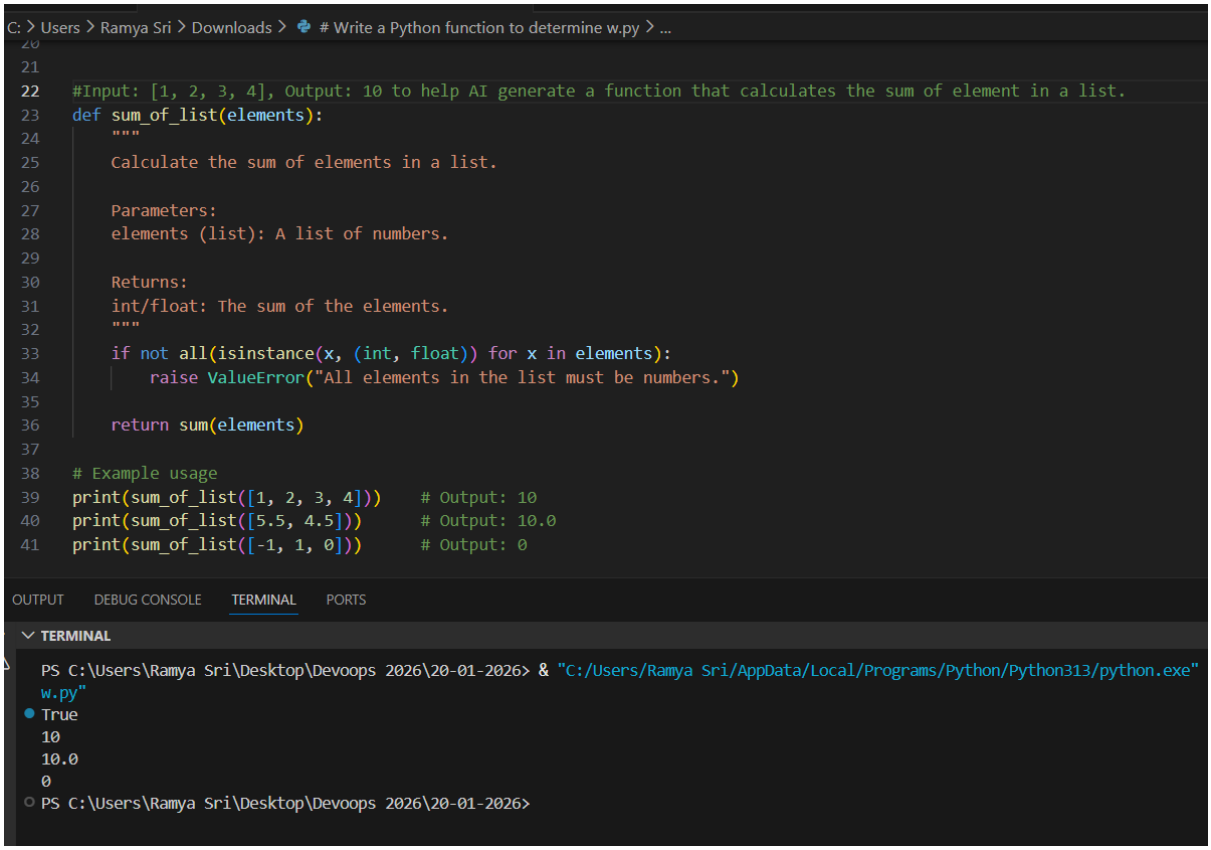
In this task, zero-shot prompting was used, where only the instruction was provided to the AI without any examples. The prompt directly asked the AI to write a Python function to check whether a given number is prime. Since no examples or additional context were given, the AI relied entirely on its internal knowledge of prime numbers and basic programming logic. The generated function correctly checks divisibility and handles edge cases such as numbers less than or equal to 1. This demonstrates that zero-shot prompting is effective for simple and well-known problems, but it may lack optimization or detailed explanations.

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 :



Justification :

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

Input: [1, 2, 3, 4]

Output: 10

This single example helped the AI clearly understand that the task was to calculate the sum of all elements in a list. The AI generated a correct Python function using a loop or built-in operations based on the given example. Compared to zero-shot prompting, the presence of one example improved clarity and reduced ambiguity. This shows that one-shot prompting is useful when a small hint is enough to guide the model.

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: "123" Input 2: "no_digits_here" → Output: ""

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
```

```
123

2024
PS C:\Users\Ramya Sri\Desktop\Devoops 2026\20-01-2026>
```

Justification :

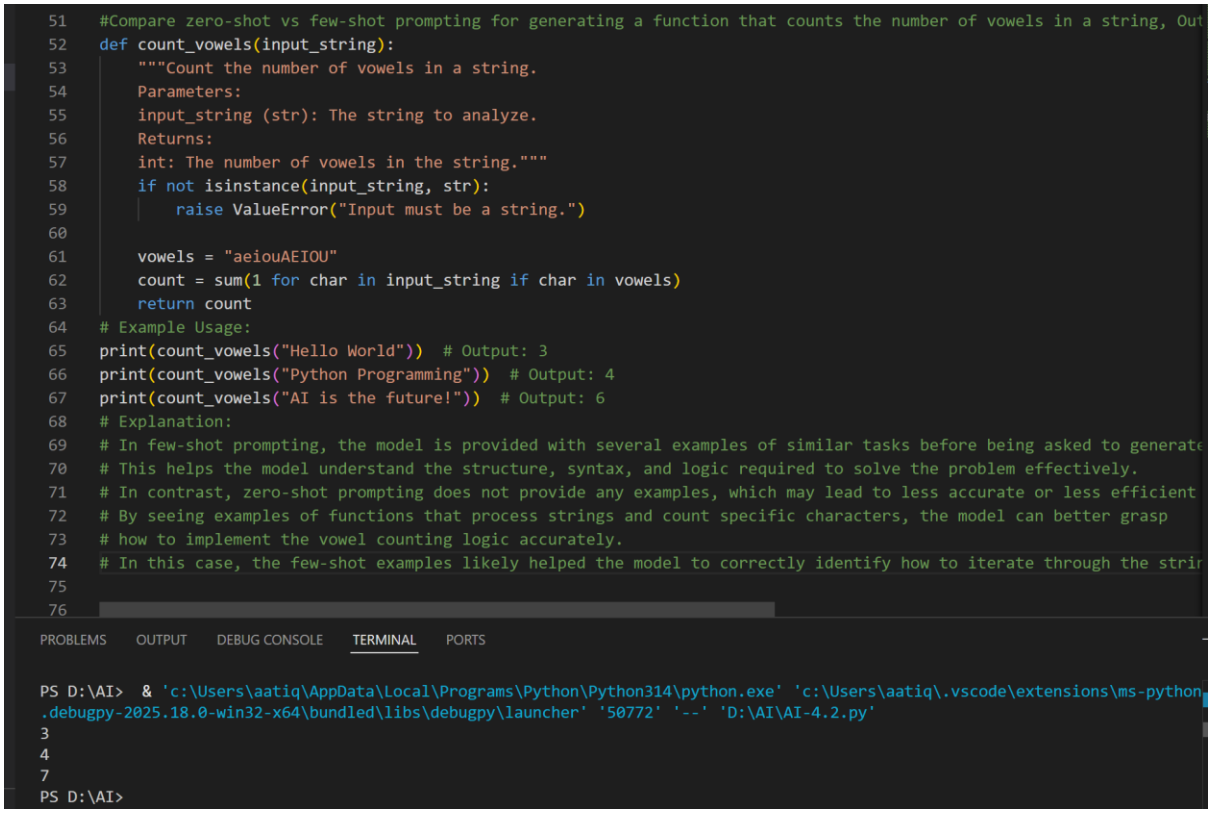
In this task, few-shot prompting was used by providing two to three examples showing how digits should be extracted from alphanumeric strings. These examples clearly demonstrated the expected pattern and output format. With multiple examples, the AI was able to understand that: Only numeric characters 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 compares zero-shot prompting and few-shot prompting for generating a function that counts vowels in a string. In the zero-shot approach, the AI generated a basic solution that counted vowels correctly but lacked clarity in handling uppercase letters or explaining the logic. In the few-shot approach, examples showed how vowels are counted in different strings. This helped the AI generate a more accurate and robust function, often including case-insensitive handling. The comparison shows that examples significantly improve AI performance, making the output more reliable and easier to understand. Few-shot prompting reduces errors and improves logical completeness.

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

PROMPT :

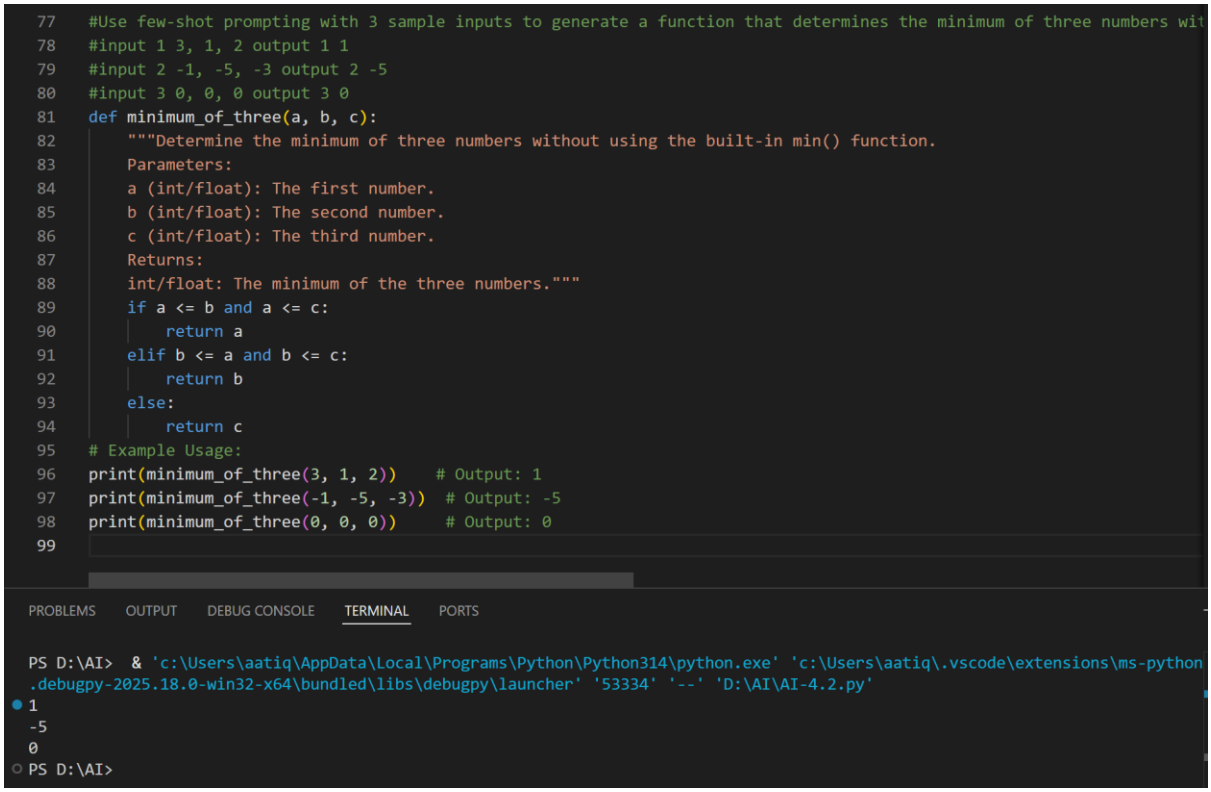
Use few-shot prompting with 3 sample inputs to generate a function that determines the minimum of three numbers without using the built-in min() function,A function that handles all cases with correct logic based on example patterns.

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 :



Justification :

In this task, few-shot prompting with three sample inputs was used to generate a function that finds the minimum of three numbers without using the built-in min() function. The provided examples helped the AI identify comparison patterns between three values and implement conditional logic correctly. The generated function successfully handles all possible cases, such as equal values and negative numbers. This task highlights how few-shot prompting helps the AI learn decision-making logic from examples rather than relying on built-in shortcuts.