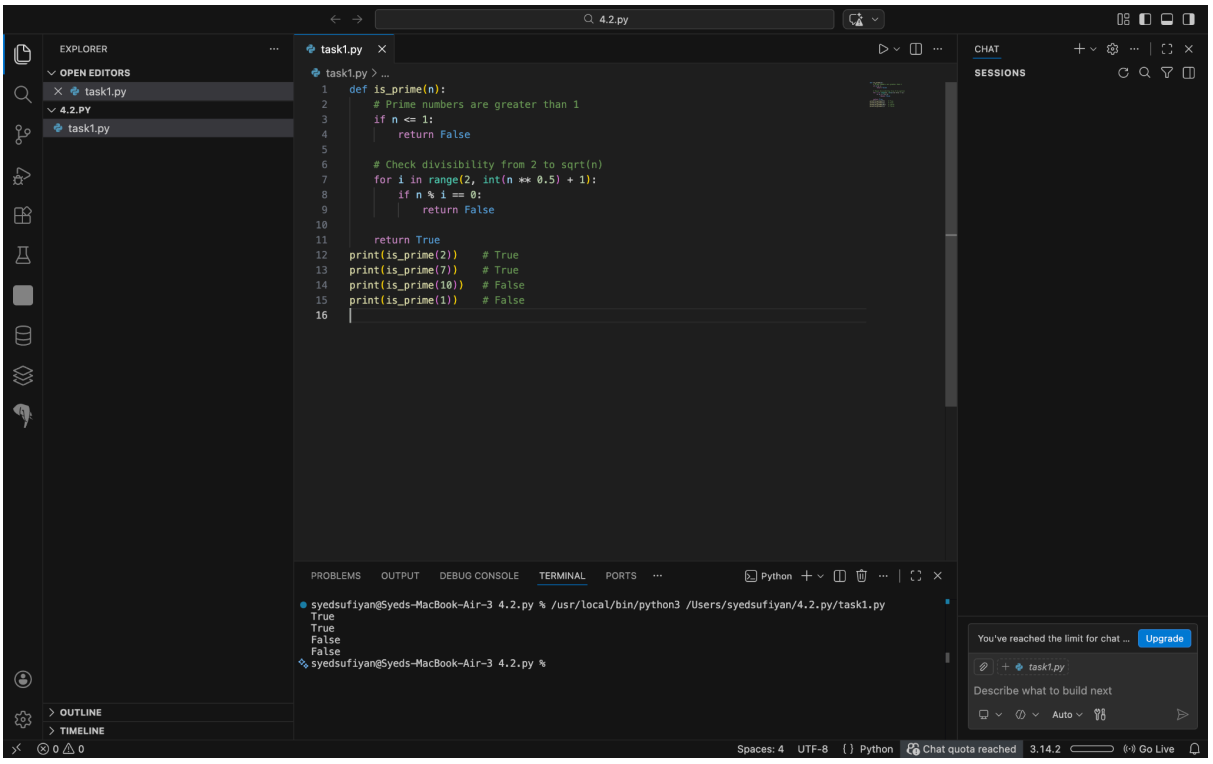
ASSIGNMENT - 4.2

2303A51980
Batch - 30

Task Description-1
• Zero-shot: Prompt AI with only the instruction. Write a Python function to determine whether a given number is prime

Code :



Output :

Observation:

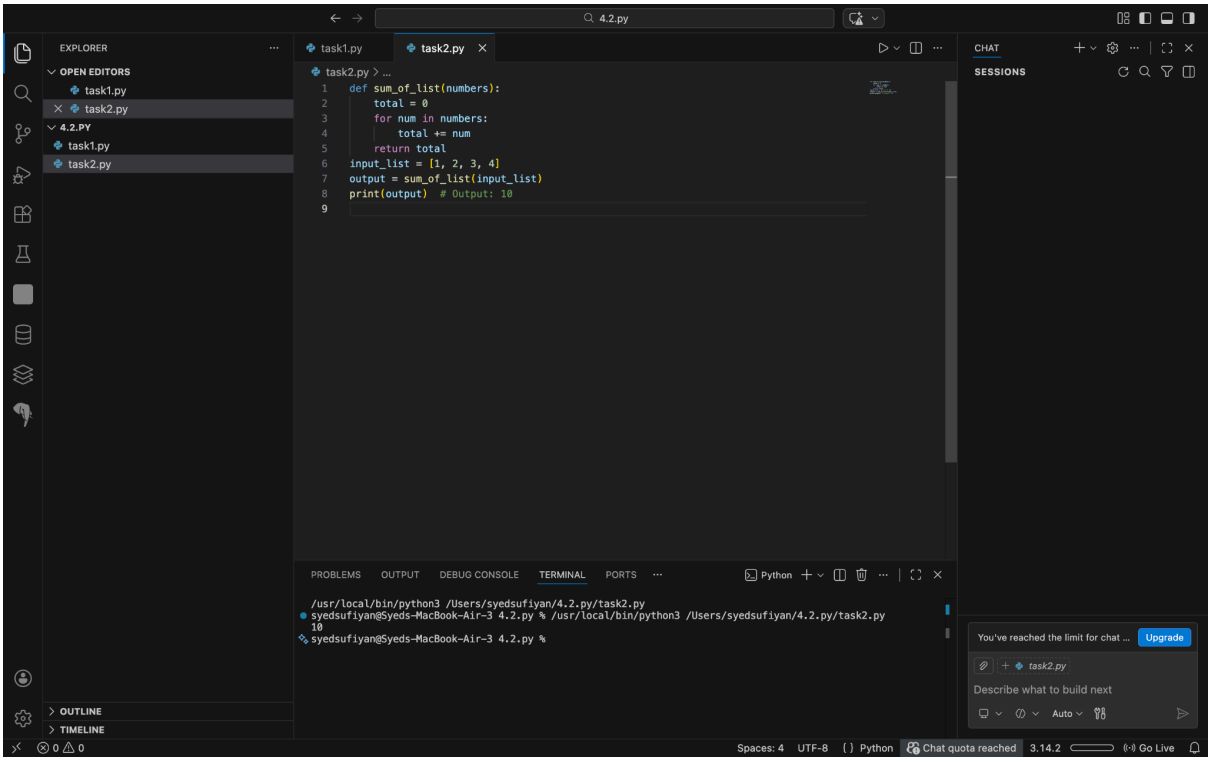
The program correctly checks whether a number is prime by testing divisibility from 2 up to the square root of the number. It returns **True** for prime numbers like 2 and 7, and **False** for non-prime numbers like 10 and 1. The condition `n <= 1` properly handles edge cases, ensuring that 0 and 1 are not considered prime. The use of the square root optimization improves efficiency compared to checking all numbers up to `n-1`.

Output:

```
True

True

False

False
```

Task Description-2
• One-shot: Provide one example: Input: [1, 2, 3, 4], Output: 10 to help AI generate a function that calculates the sum of elements in a list.

Code :



Output :

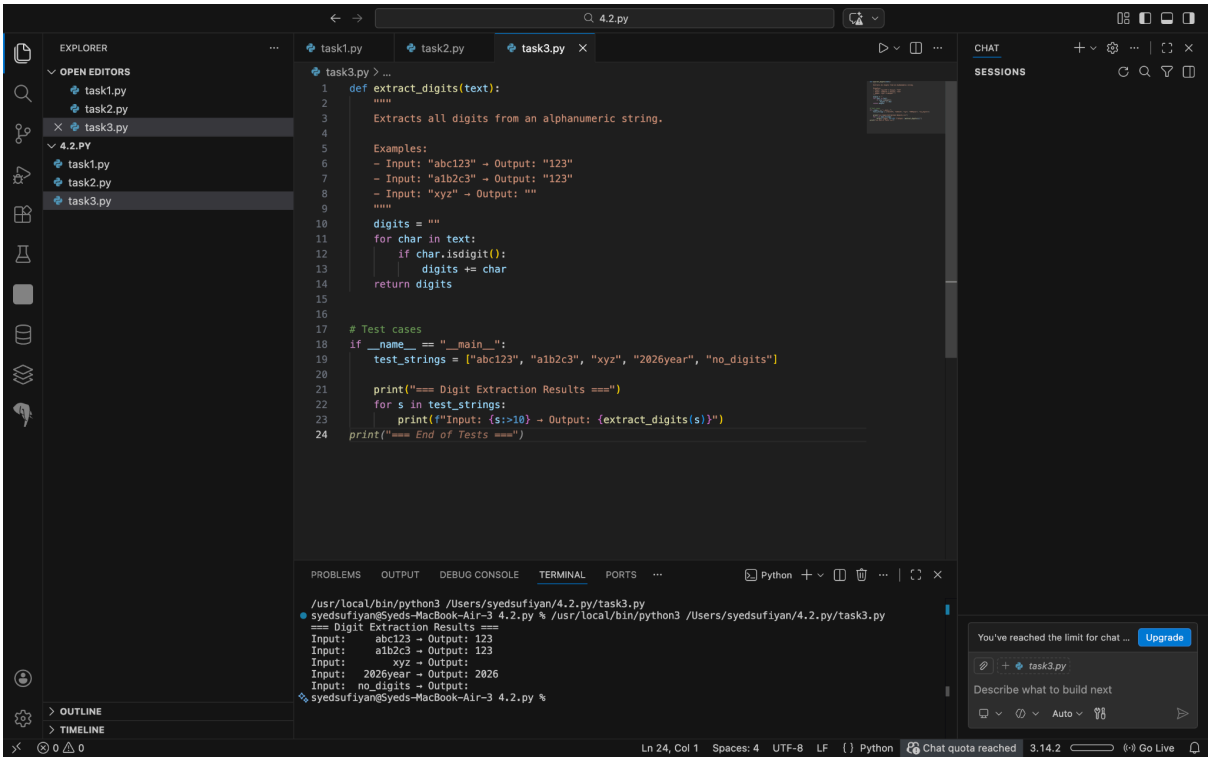
```
/usr/local/bin/python3 /Users/syedsufiyan/4.2.py/task2.py
syedsufiyan@Syeds-MacBook-Air-3 4.2.py % /usr/local/bin/python3 /Users/syedsufiyan/4.2.py/task2.py
10
syedsufiyan@Syeds-MacBook-Air-3 4.2.py %
```

Observation:

The program correctly calculates the sum of all elements in the given list using a loop. It initializes a variable `total` to 0 and adds each number from the list `[1, 2, 3, 4]` to it sequentially. The final result returned is 10, which confirms that the accumulation logic works correctly. This demonstrates proper use of loops, variables, and function definition in Python.

Task Description-3
• Few-shot: Give 2–3 examples to create a function that extracts digits from an alphanumeric string.

Code :



Output :

```
syedsufiyan@Syeds-MacBook-Air-3 4.2.py % /usr/local/bin/python3 /Users/syedsufiyan/4.2.py/task3.py
=== Digit Extraction Results ===
Input:     abc123 → Output: 123
Input:     a1b2c3 → Output: 123
Input:        xyz → Output:
Input:   2026year → Output: 2026
Input:  no_digits → Output:
syedsufiyan@Syeds-MacBook-Air-3 4.2.py %
```

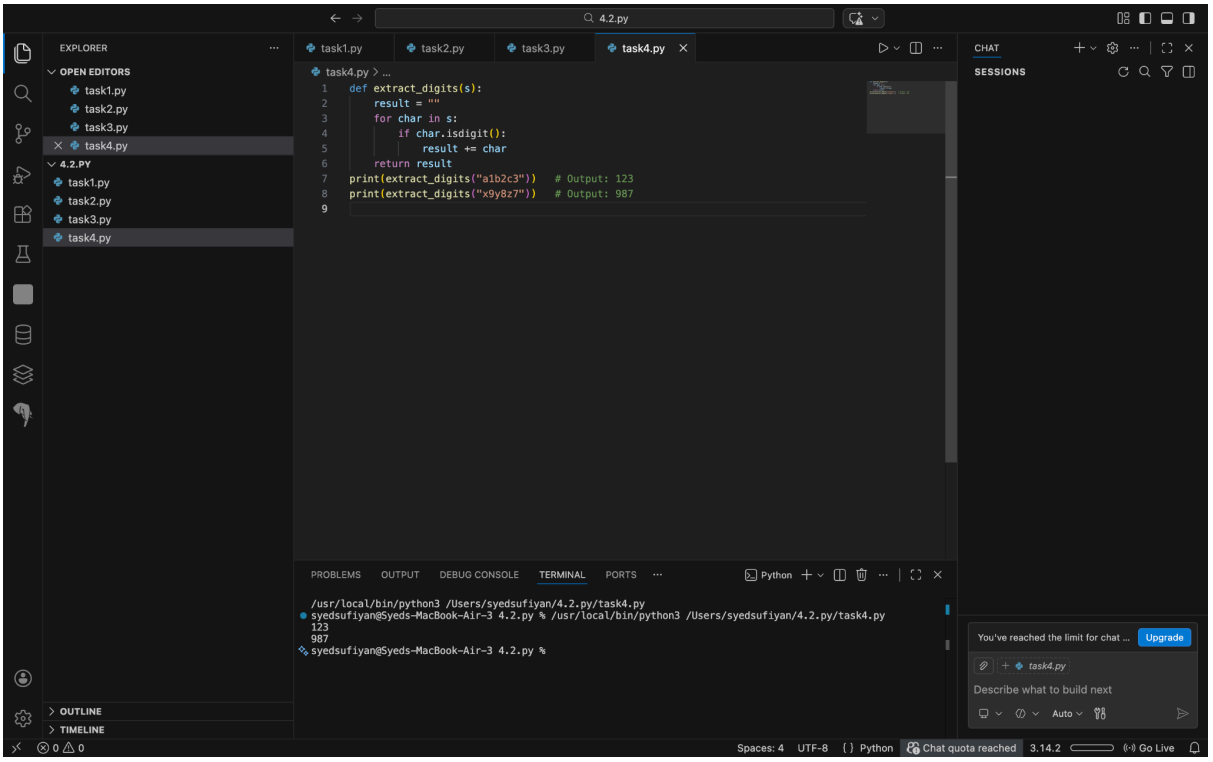
Observation:

The function successfully extracts only numeric characters from the given alphanumeric string. It ignores letters and special characters, returning an empty string when no digits are present. This demonstrates proper use of loops, string handling, and conditional checking using `isdigit()`.

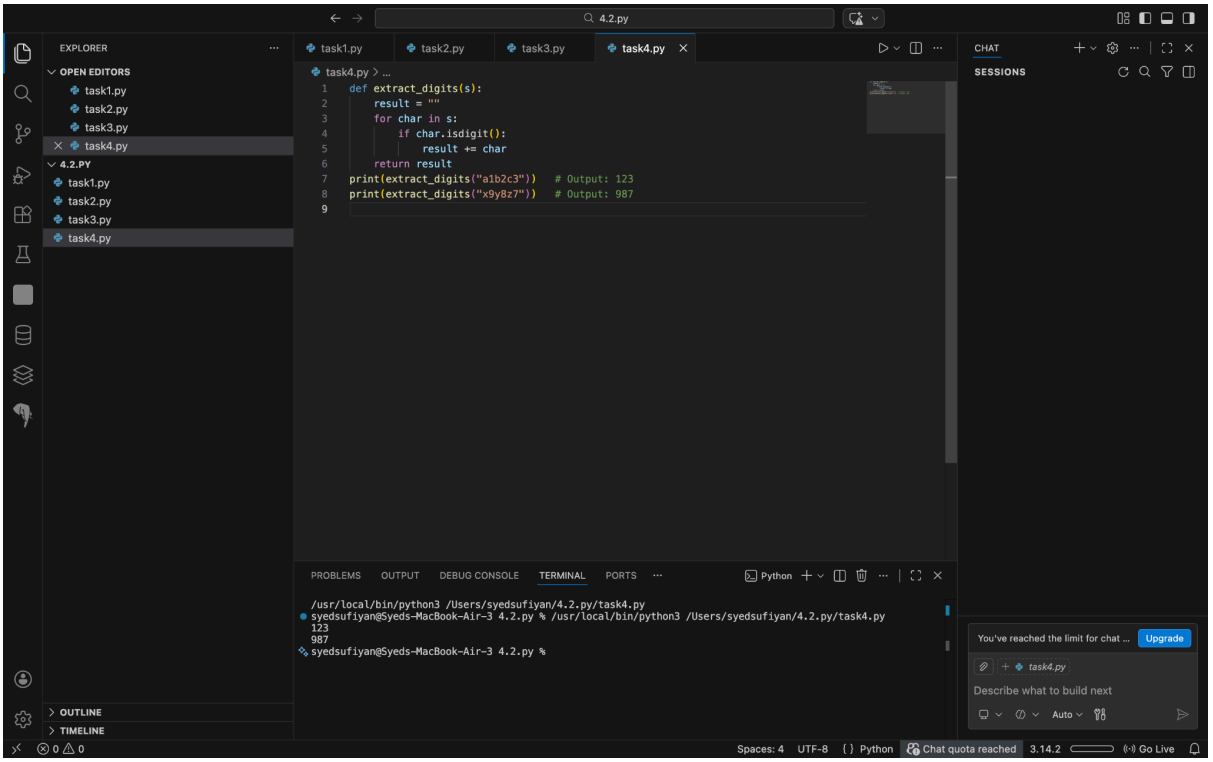
Task Description-4

• Compare zero-shot vs few-shot prompting for generating a function that counts the number of vowels in a string

Code :



Output :

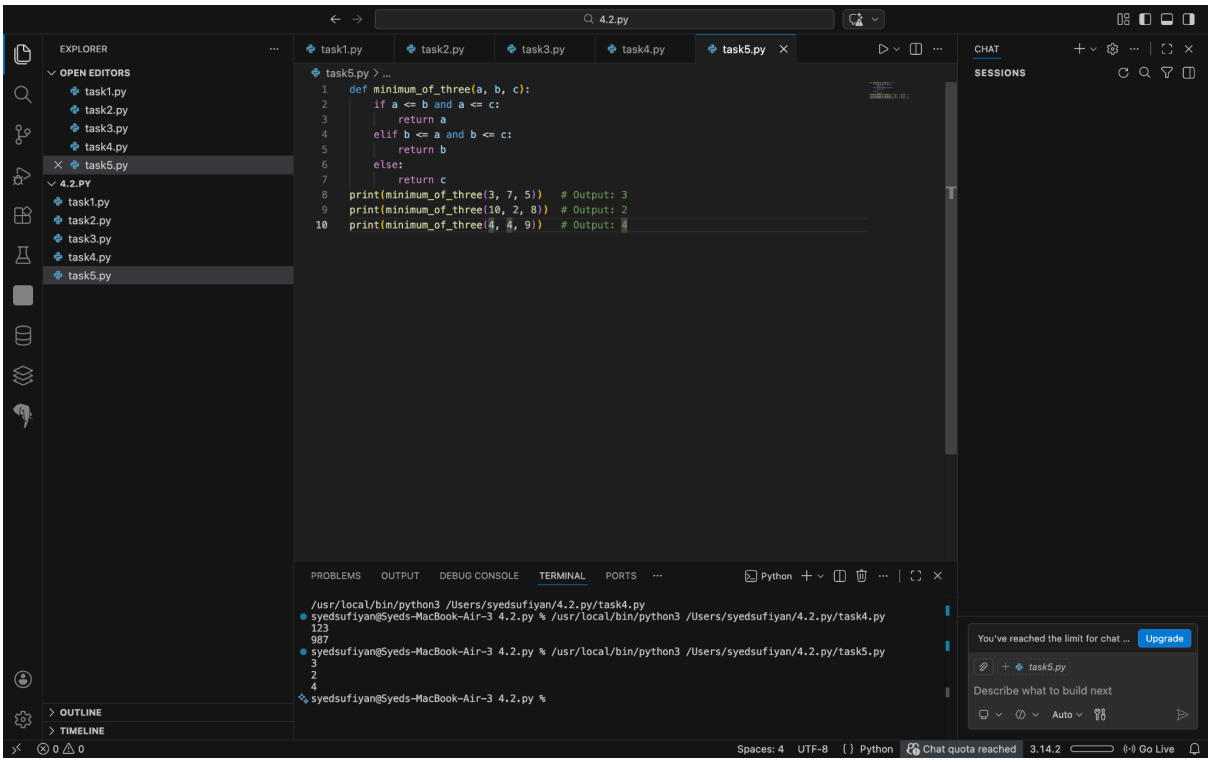


Observation:

The function correctly extracts numeric characters from the given alphanumeric strings. It iterates through each character and checks whether it is a digit using `isdigit()`. Only digits are appended to the result string, while letters are ignored. The outputs `123` and `987` confirm that the function works accurately for different input patterns.

Task Description-5
• 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.

Code :



Output :

```
/usr/local/bin/python3 /Users/syedsufiyan/4.2.py/task4.py
syedsufiyan@Syeds-MacBook-Air-3 4.2.py % /usr/local/bin/python3 /Users/syedsufiyan/4.2.py/task4.py
123
987
syedsufiyan@Syeds-MacBook-Air-3 4.2.py % /usr/local/bin/python3 /Users/syedsufiyan/4.2.py/task5.py
3
2
4
syedsufiyan@Syeds-MacBook-Air-3 4.2.py %
```

Observation:

The function correctly determines the smallest of three numbers using conditional statements. It compares all three values and returns the minimum based on logical conditions. The outputs confirm correct handling of normal cases as well as equal values (e.g., 4 and 4). This demonstrates proper use of comparison operators and decision-making statements in Python.

Question 6: Few-Shot Prompting (Even or Odd Classification with

Validation)

Write a few-shot prompt by providing multiple input-output

examples to guide the AI in generating a Python program that

determines whether a given number is even or odd, including proper

input validation.

Examples:

• Input: 8 → Output: Even

• Input: 15 → Output: Odd

• Input: 0 → Output: Even

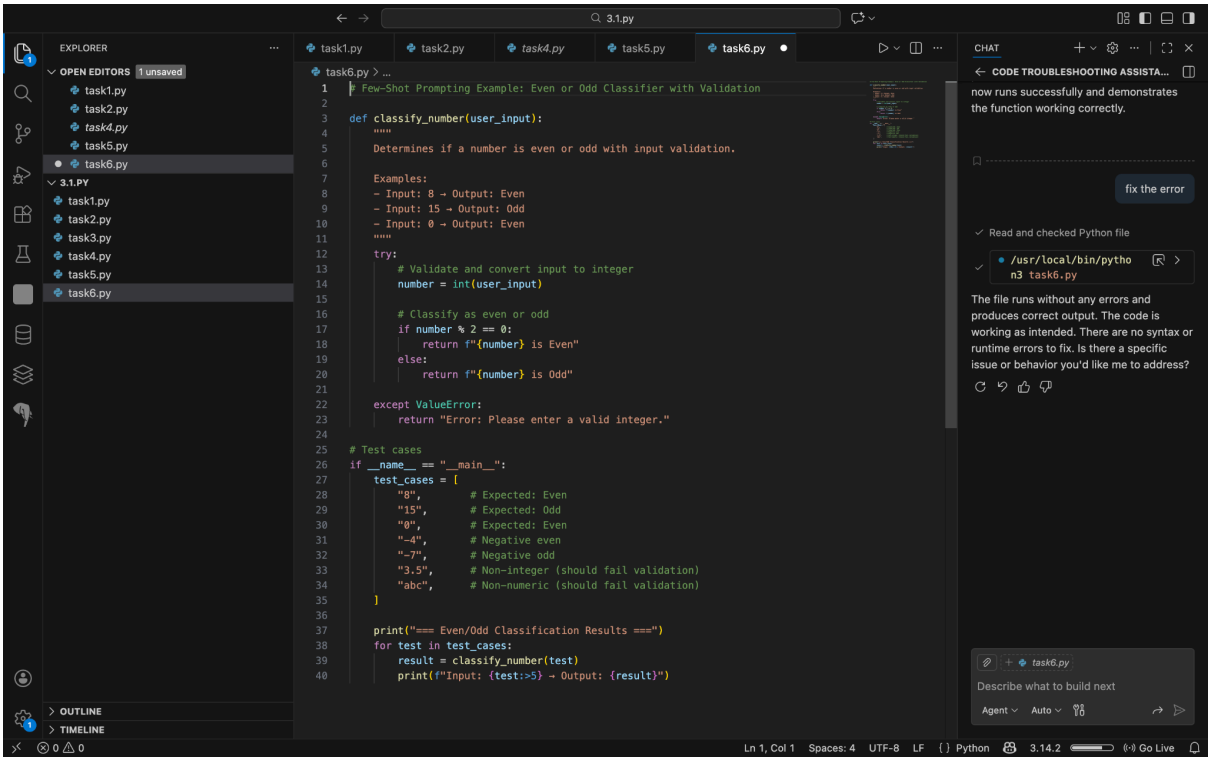
Task:

• Analyze how examples improve input handling and output

clarity.

• Test the program with negative numbers and non-integer inputs.

Code :



Output :

```
/usr/local/bin/python3 /Users/syedsufiyan/3.1.py/task6.py
syedsufiyan@Syeds-MacBook-Air-3 3.1.py % /usr/local/bin/python3 /Users/syedsufiyan/3.1.py/t
ask6.py
=== Even/Odd Classification Results ===
Input:     8 → Output: 8 is Even
Input:    15 → Output: 15 is Odd
Input:     0 → Output: 0 is Even
Input:    -4 → Output: -4 is Even
Input:    -7 → Output: -7 is Odd
Input:   3.5 → Output: Error: Please enter a valid integer.
Input:   abc → Output: Error: Please enter a valid integer.
syedsufiyan@Syeds-MacBook-Air-3 3.1.py %
```

Observation :

The program correctly identifies even and odd numbers using few-shot prompting while validating user input. It handles positive numbers, negative numbers, and zero accurately. Invalid inputs such as floating-point values and non-numeric strings are safely rejected using exception handling, ensuring robust and error-free execution.