Lab-4.2 (Assignment-04)

Course : AI Assisted Coding

Topic:- Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques

Student Details:

Name : Batti Chandan Singh

Hall Ticket No : 2303A51515

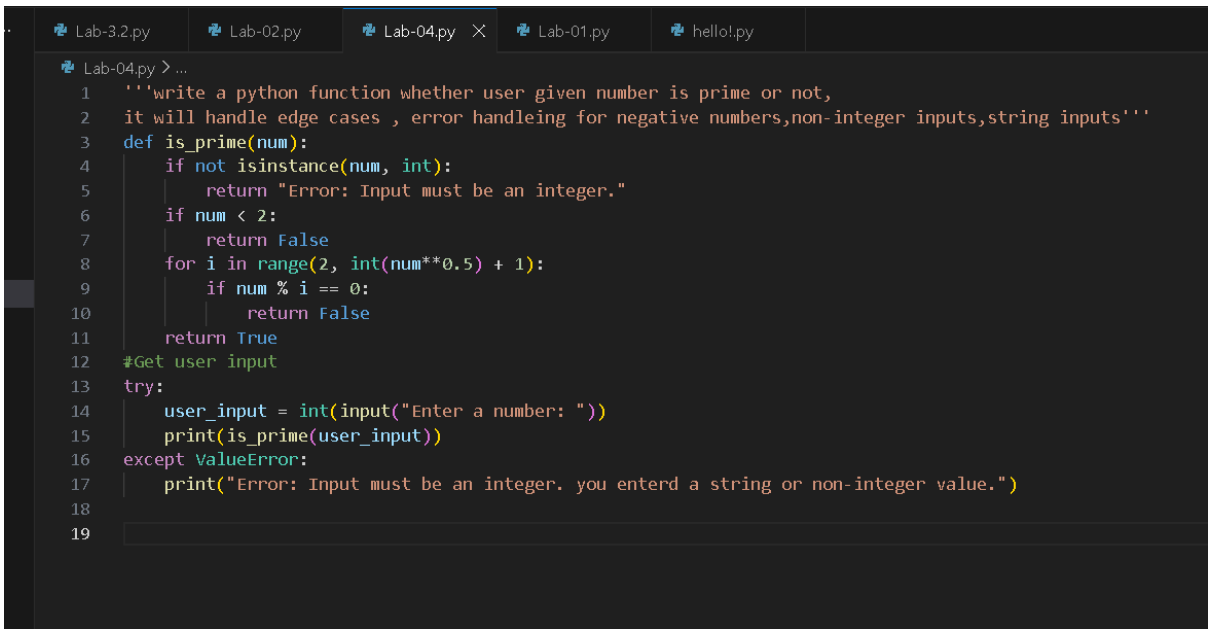
Batch : 22

Task-01

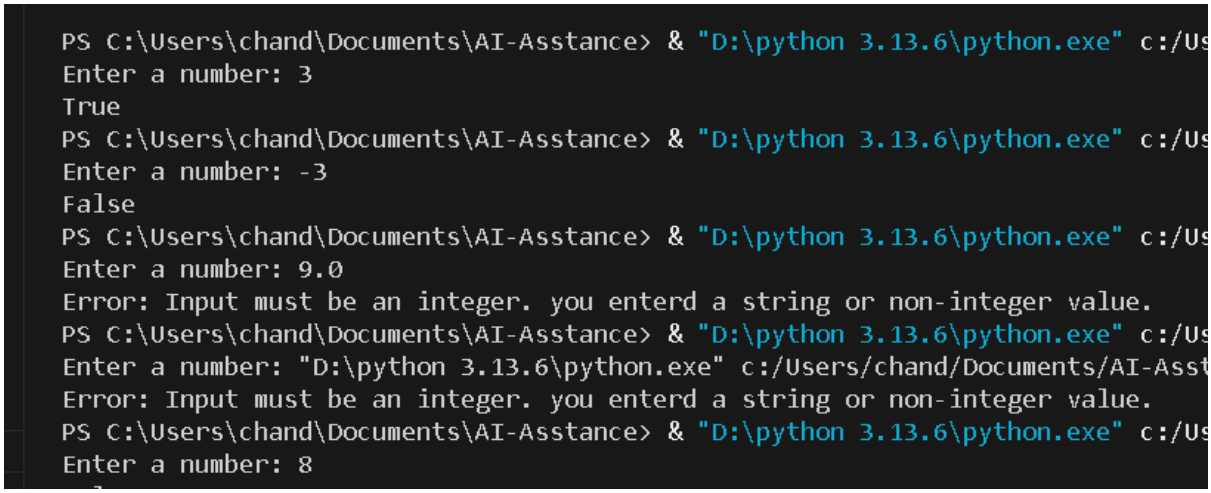
Final Optimal Prompt

write a python function whether user given number is prime or not, it will handle edge cases , error handling for negative numbers, non-integer inputs , string inputs

Code Screenshot



Output Screenshot



Explanation/Justification/Observation (100 words / 5 – 6 sentence)

In this task, I used a zero-shot prompt to generate a Python function that checks whether a given number is prime or not. Initially, even though the logic and syntax were correct, the output did not fully match my expectations. I realized that zero-shot prompting provides instructions without examples, which can sometimes lead to partial or unexpected results. When handling edge cases like negative numbers, non-integer inputs, and string values, the AI required very clear and detailed instructions. This task helped me understand the limitations of zero-shot prompting. It also showed the importance of examples in improving output accuracy

Task-02

Final Optimal Prompt

#write the python function to print the sum of elements in a list Example input : [1,2,3,4] output : 10

Code Screenshot

```python
#---Question 02/05 Lab-04.py---#
'''#write the python function to print the sum of elements in a list E
def sum_of_elements(lst):
    if not all(isinstance(x, (int, float)) for x in lst):
        return "Error: All elements in the list must be numbers."
    return sum(lst)
#Get user input
user_input = input("Enter a list of numbers separated by commas: ")
try:
    num_list = [float(x.strip()) for x in user_input.split(',')]
    print(sum_of_elements(num_list))
except ValueError:
    print("Error: Please enter valid numbers separated by commas.")
```

Output Screenshot

```
PS C:\Users\chand\Documents\AI-Asstance> & "D:\python 3.13
Enter a list of numbers separated by commas: 1,2,3,4
10.0
PS C:\Users\chand\Documents\AI-Asstance> & "D:\python 3.13
Enter a list of numbers separated by commas: 2,9.0,3.4
14.4
PS C:\Users\chand\Documents\AI-Asstance> & "D:\python 3.13
Enter a list of numbers separated by commas: 3.4,4.5,0.5,0
9.2
PS C:\Users\chand\Documents\AI-Asstance> & "D:\python 3.13
Enter a list of numbers separated by commas: k,e,r,t
Error: Please enter valid numbers separated by commas.
PS C:\Users\chand\Documents\AI-Asstance>
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence)

In this task, a Python function was designed to calculate the sum of elements in a user-provided list. The function includes validation to ensure that all elements are numeric, which helps prevent runtime errors. Error handling is implemented to manage cases where the user enters invalid inputs such as strings or mixed data types. GitHub Copilot assisted in generating concise logic using built-in functions like sum(). This task demonstrates how AI-

assisted coding helps in writing clean and efficient functions. It also emphasizes the importance of input validation in real-world programs.

Task-03

Final Optimal Prompt

write a python function to print the number which is present in given user input string,and handle the error cases,

Example input : "abc123xyz" output : 123, edge case example input : "abcxyz" output : "No numbers found

Code Screenshot

```
#---Question 03/05 Lab-04.py---#
'''write a python function to print the number which is prese
#Example input : "abc123xyz" output : 123, edge case example
def extract_number(s):
    numbers = ''.join(filter(str.isdigit, s))
    return numbers if numbers else "No numbers found"
#Get user input
user_input = input("Enter a string: ")
print(extract_number(user_input))
```

Output Screenshot

```
PS C:\Users\chand\Documents\AI-Asstance> & "D:\pytl
Enter a string: chandu
No numbers found
PS C:\Users\chand\Documents\AI-Asstance> & "D:\pytl
Enter a string: chandu09
09
PS C:\Users\chand\Documents\AI-Asstance> & "D:\pytl
Enter a string: chandu.09
09
PS C:\Users\chand\Documents\AI-Asstance> & "D:\pytl
Enter a string: chandu(9.0)
90
PS C:\Users\chand\Documents\AI-Asstance> & "D:\pytl
Enter a string: 9
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence)

This task focuses on extracting numeric values from a given user input string. The function handles edge cases where no digits are present by returning an appropriate message.

GitHub Copilot effectively suggested using string filtering techniques to identify digits. This approach simplifies the logic and improves readability. Error-free execution is ensured even when the input contains only characters. Through this task, I learned how AI tools can assist in string manipulation problems while maintaining robustness and clarity.

Task-04

Final Optimal Prompt

Zero Short Prompt:

#1.write the python function count the number of vowels in the given user input string, handle edge cases and error handling for non-string inputs.

One Short Prompt:

#2.write the python function count the number of vowels in the given user input string, handle edge cases and error handling for non-string inputs.

#Example input : "Hello World" output : Hello World have 3 vowels

Code Screenshot

```python
#1.write the python function count the number of vowels
def count_vowels(s):
    if not isinstance(s, str):
        return "Error: Input must be a string."
    vowels = 'aeiouAEIOU'
    count = sum(1 for char in s if char in vowels)
    return count
#Get user input
user_input = input("Enter a string: ")
print(count_vowels(user_input))

#2.write the python function count the number of vowels
#Example input : "Hello World" output : Hello World have
def count_vowels_with_message(s):
    if not isinstance(s, str):
        return "Error: Input must be a string."
    vowels = 'aeiouAEIOU'
    count = sum(1 for char in s if char in vowels)
    return f'"{s}" has {count} vowels'
#Get user input
user_input = input("Enter a string: ")
print(count_vowels_with_message(user_input))
```

Output Screenshot

```
PS C:\Users\chand\Documents\AI-Asst
Enter a string: chandu
2
Enter a string: chandu
"chandu" has 2 vowels
PS C:\Users\chand\Documents\AI-Asst
Enter a string: chandu12
2
Enter a string: chandu12
"chandu12" has 2 vowels
PS C:\Users\chand\Documents\AI-Asst
Enter a string: ()
0
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence)

In this task, I analyzed the difference between zero-shot and one-shot prompting using AI-assisted coding. When a zero-shot prompt was used, the AI generated a solution based only on instructions, which sometimes resulted in outputs that did not fully match expectations. By providing a one-shot prompt with a single example, the AI clearly understood the required input-output pattern. This improved accuracy and reduced ambiguity in the generated code. The task helped me understand that examples play a critical role in guiding AI behavior. Overall, one-shot prompting produces more reliable and predictable results than zero-shot prompting.

Tack-05

Final Optimal Prompt

#write the python function to determine the minimum of three user given numbers, handle edge cases and error handling for non-numeric inputs.

Code Screenshot

```python
def find_minimum(a, b, c):
    try:
        a = float(a)
        b = float(b)
        c = float(c)
    except ValueError:
        return "Error: All inputs must be numeric."
    return min(a, b, c)
#Get user input
user_input = input("Enter three numbers separated by commas: ")
try:
    num1, num2, num3 = [x.strip() for x in user_input.split(',')]
    print(find_minimum(num1, num2, num3))
except ValueError:
    print("Error: Please enter exactly three numbers separated by commas.")
```

Output Screenshot

```
Enter three numbers separated by commas: 1,2,3
1.0
PS C:\Users\chand\Documents\AI-Asstance> & "D:\python
Enter three numbers separated by commas: 1.2,2.3,4.5
1.2
PS C:\Users\chand\Documents\AI-Asstance> & "D:\python
Enter three numbers separated by commas: 1,2,y
Error: All inputs must be numeric.
PS C:\Users\chand\Documents\AI-Asstance> & "D:\python
Enter three numbers separated by commas: -0,09,5
-0.0
PS C:\Users\chand\Documents\AI-Asstance> & "D:\python
Enter three numbers separated by commas: -1,90,-222
-222.0
PS C:\Users\chand\Documents\AI-Asstance>
```

Explanation/Justification/Observation (100 words / 5 – 6 sentence)

This task involves determining the minimum of three user-provided numbers using a Python function. The function converts inputs into numeric values and handles non-numeric input through exception handling. GitHub Copilot assisted in generating a clean and reliable solution using built-in functions like min(). Error handling ensures the program does not

crash when invalid input is given. This task demonstrates how AI tools support safe coding practices. It also shows the benefit of modular design for simple decision-making problems.