AI ASSISTANT CODING
Assignment – 4.2

Name: K. Shailaja

H.T No.: 2303A52155

Batch: 34

Question:

Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques
Lab Objectives:
• To explore and apply different levels of prompt examples in AI-assisted code
generation.
• To understand how zero-shot, one-shot, and few-shot prompting affect AI output
quality.
• To evaluate the impact of context richness and example quantity on AI performance.
• To build awareness of prompt strategy effectiveness for different problem types.

Lab Outcomes (LOs):
After completing this lab, students will be able to:
• Use zero-shot prompting to instruct AI with minimal context.
• Use one-shot prompting with a single example to guide AI code generation.
• Apply few-shot prompting using multiple examples to improve AI responses.
• Compare AI outputs across the three prompting strategies.

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

Prompt:

Generate a python program to determine whether the given number is prime only based on conditions without relying on examples

```python
prime_checker.py > ...
1    def is_prime(n):
4        if n == 2:
5            return True
6        if n % 2 == 0:
7            return False
8        # Check odd divisors up to sqrt(n)
9        for i in range(3, int(n**0.5) + 1, 2):
10           if n % i == 0:
11               return False
12       return True
13   def main():
14       """Main function to get user input and check if number is prime."""
15       try:
16           num = int(input("Enter a number to check if it's prime: "))
17           if is_prime(num):
18               print(f"{num} is a PRIME number!")
19           else:
20               print(f"{num} is NOT a prime number.")
21       except ValueError:
22           print("Error: Please enter a valid integer.")
23   if __name__ == "__main__":
24       main()
25
```

Expected Output-1

• A basic Python function to check if a number is prime, demonstrating correct logical
conditions without relying on examples or additional context

```
Enter a number to check if it's prime: 6
6 is NOT a prime number.
```

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.

Prompt:

Generate a python program to sum of elements in a list i will provide you an example input and output

input: [1,2,3,4]

output:10

provide the code in such a way so that the instructions have to be taken by the code in the same way as given example give me an error free code with optimization.

```python
list_sum.py > sum_list
1    def sum_list(elements):
2        """
3        Calculate the sum of all elements in a list.
4
5        Args:
6            elements: A list of numbers to sum
7
8        Returns:
9            The sum of all elements in the list
10       """
11       return sum(elements)
12
13
14   def main():
15       """Main function to get list input and calculate sum."""
16       try:
17           # Get input from user in list format: [1,2,3,4]
18           user_input = input("Enter a list of numbers (e.g., [1,2,3,4]): ").strip()
19
20           # Parse the input string to convert it to a list
21           elements = eval(user_input)
22
23           # Validate that input is a list
24           if not isinstance(elements, list):
25               print("Error: Please enter a valid list format.")
26               return
27
```

```
list_sum.py > sum_list
14    def main():
27
28            # Check if list is empty
29            if not elements:
30                print("Error: List cannot be empty.")
31                return
32
33            # Verify all elements are numbers
34            if not all(isinstance(x, (int, float)) for x in elements):
35                print("Error: List must contain only numbers.")
36                return
37
38            # Calculate and display the sum
39            result = sum_list(elements)
40            print(f"input: {elements}")
41            print(f"output: {result}")
42
43        except (ValueError, SyntaxError):
44            print("Error: Please enter a valid list format like [1,2,3,4]")
45        except Exception as e:
46            print(f"Error: {str(e)}")
47
48
49    if __name__ == "__main__":
50        main()
51
```

Expected Output-2

• A correct conversion function guided by the single example.

```
Enter a list of numbers (e.g., [1,2,3,4]): [4,5,6,2,7]
input: [4, 5, 6, 2, 7]
output: 24
```

Task Description-3

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

Prompt:

write a program that extract digits from an alphanumeric string.

Examples:

"jb54jdjk" ---> 54

"abcd098ijh" ---> 098

"mno653prg" ---> 653

```
  AIAC_Assignment-4.2 > ...
   1    def extract_digits(s):
   2        """
   3        Extracts digits from an alphanumeric string and returns them as a string.
   4
   5        Args:
   6            s: An alphanumeric string
   7
   8        Returns:
   9            A string formed by concatenating all digits in the string
  10        """
  11        digits = ''.join(char for char in s if char.isdigit())
  12        return digits if digits else "0"
  13
  14
  15    # Test cases
  16    print(extract_digits("jb54jdjk"))        # 54
  17    print(extract_digits("abcd098ijh"))      # 098
  18    print(extract_digits("mno653prg"))       # 653
  19    print(extract_digits("ach12bg"))         # 12
  20    print(extract_digits("th9n1ljfm"))       # 91
  21    print(extract_digits("asd986lkj"))       # 986
```

Expected Output-3

• Accurate function that returns only the digits from alphanumeric string.

```
iran\OneDrive\Desktop\AIAC Assignments\AIAC_Assignment
54
098
653
12
91
986
```

Task Description-4

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

Prompt:

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

```
25    # Zero-shot prompting example
26    def count_vowels_zero_shot(s):
27        """Count vowels in a string without examples."""
28        return sum(1 for char in s.lower() if char in 'aeiou')
29
30
31    # Few-shot prompting example with test cases
32    def count_vowels_few_shot(s):
33        """
34        Count the number of vowels in a string.
35
36        Examples:
37            "hello" -> 2 (e, o)
38            "programming" -> 3 (o, a, i)
39            "aeiou" -> 5
40        """
41        return sum(1 for char in s.lower() if char in 'aeiou')
42
43
44    # Test both approaches
45    print(count_vowels_zero_shot("hello"))           # 2
46    print(count_vowels_few_shot("hello"))            # 2
47    print(count_vowels_zero_shot("programming"))     # 3
48    print(count_vowels_few_shot("programming"))      # 3
```

Expected Output-4

• Output comparison + student explanation on how examples helped the model.

```
2
2
3
3
```

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.

Prompt:

Generate a function that determines the minimum of three numbers without using the built-in min () function.

Example: input: 4 5 3

 output: minimum among given numbers is 3

input: 5 4 7

output: minimum among given numbers is 4

input: 10 200 1000

output: minimum among given numbers is 10

first analyze the given example then generate the code accordingly give me the optimized code using valid conditions

```python
def find_minimum(a, b, c):
    """Function to find the minimum among three numbers."""
    # Compare a with b first
    if a <= b:
        # If a is less than or equal to b, compare a with c
        if a <= c:
            return a
        else:
            return c
    else:
        # If b is less than a, compare b with c
        if b <= c:
            return b
        else:
            return c


def main():
    """Main function to get input and find minimum of three numbers."""
    try:
        # Get three numbers from user
        print("Enter three numbers separated by spaces:")
        user_input = input().strip()

        # Parse input
        numbers = list(map(int, user_input.split()))
```

```
26        numbers = list(map(int, user_input.split()))
27
28        # Validate that we have exactly 3 numbers
29        if len(numbers) != 3:
30            print("Error: Please enter exactly 3 numbers.")
31            return
32
33        a, b, c = numbers
34
35        # Find and display the minimum
36        minimum = find_minimum(a, b, c)
37        print(f"input: {a} {b} {c}")
38        print(f"output: minimum among given numbers is {minimum}")
39
40    except ValueError:
41        print("Error: Please enter valid numbers separated by spaces.")
42    except Exception as e:
43        print(f"Error: {str(e)}")
44
45
46 if __name__ == "__main__":
47    main()
48
```

Expected Output-5

• A function that handles all cases with correct logic based on example patterns.

```
Enter three numbers separated by spaces:
4 2 8
input: 4 2 8
output: minimum among given numbers is 2
```