AI Assistant Coding

Assignment-3.1

Name:K.Trinay Prasad

HT No:2303A52070

Batch:32

Question 1: Zero-Shot Prompting (Palindrome Number Program)

Write a zero-shot prompt (without providing any examples) to generate

a Python function that checks whether a given number is a palindrome.

Task:

• Record the AI-generated code.

• Test the code with multiple inputs.

• Identify any logical errors or missing edge-case handling.

Prompt:

```
#Write a Python function to check whether a given number is a palindrome or not.
```

Code:

```python
def is_palindrome(number):
    """
    Check if a given number is a palindrome.

    :param number: The number to check.
    :return: True if the number is a palindrome, False otherwise.
    """
    # Convert the number to a string and check if it's equal to its reverse
    str_num = str(number)
    # Compare the string with its reverse
    return str_num == str_num[::-1]
#Get user input
num = int(input("Enter a number: "))
#Check if the number is a palindrome
if is_palindrome(num):
    print(f"{num} is a palindrome.")
else:
    print(f"{num} is not a palindrome.")
```

Output:

```
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter a number: 121
121 is a palindrome.
PS D:\AI Assicoding>
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter a number: 123
123 is not a palindrome.
PS D:\AI Assicoding>
```

Explanation:

The code is handled everything but edge case like negative and non-integers are not handled.

Question 2: One-Shot Prompting (Factorial Calculation)

Write a one-shot prompt by providing one input-output example and

ask the AI to generate a Python function to compute the factorial of a

given number.

Example:

Input: 5 → Output: 120

Task:

• Compare the generated code with a zero-shot solution.

• Examine improvements in clarity and correctness.

Prompt:

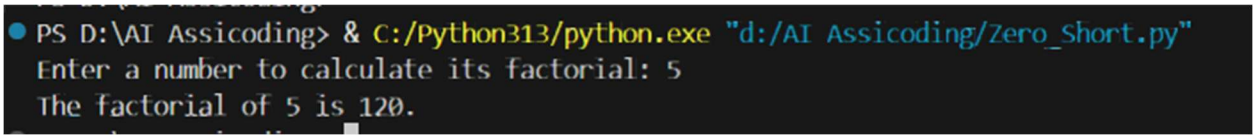
```
#Write a python script to calculate the factorial of a number example input 5 output 120
```

Code:

```
def factorial(n):
    """
    Calculate the factorial of a given number.

    :param n: The number to calculate the factorial for.
    :return: The factorial of the number.
    """
    if n < 0:
        return "Factorial is not defined for negative numbers."
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result
    #Get user input
num = int(input("Enter a number to calculate its factorial: "))
#Calculate and print the factorial
result = factorial(num)
print(f"The factorial of {num} is {result}.")
```

Output:



Explanation:

Aspect	Zero-Shot	One-Shot
Prompt detail	Minimal	Includes example
Code clarity	Basic	More structured
Edge-case handling	Often missing	Usually included
Accuracy	Moderate	Higher
Reliability	Depends on model	Improved due to guidance

Question 3: Few-Shot Prompting (Armstrong Number Check)

Write a few-shot prompt by providing multiple input-output examples to guide the Al in generating a Python function to check whether a given number is an Armstrong number.

Examples:

. Input: 153 > Output: Armstrong Number

. Input: 370 > Output: Armstrong Number

. Input: 123 > Output: Not an Armstrong Number

Task:

. Analyze how multiple examples influence code structure and

accuracy.

. Test the function with boundary values and invalid inputs.

Prompt:

```
'''
Docstring for Ass2.armstrong number
write a python script to check whether a given number is an armstrong number or not
Examples:
.Input:153 > Output:153 is an armstrong number
.Input: 370 > Output:370 is an armstrong number
.Input:123 > Output:123 is not an armstrong number
'''
```

Code:

```
def is_armstrong(number):
    """
    Check if a given number is an Armstrong number.

    :param number: The number to check.
    :return: True if the number is an Armstrong number, False otherwise.
    """
    # Convert the number to a string to easily iterate over digits
    str_num = str(number)
    num_digits = len(str_num)
    # Calculate the sum of each digit raised to the power of num_digits
    sum_of_powers = sum(int(digit) ** num_digits for digit in str_num)
    # An Armstrong number is equal to the sum of its own digits each raised to the power of the number of digits
    return sum_of_powers == number
#Get user input
num = int(input("Enter a number to check if it's an Armstrong number: "))
#Check if the number is an Armstrong number
if is_armstrong(num):
    print(f"{num} is an Armstrong number.")
else:
    print(f"{num} is not an Armstrong number.")
```

Output:

```
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter a number to check if it's an Armstrong number: 153
153 is an Armstrong number.
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter a number to check if it's an Armstrong number: 370
370 is an Armstrong number.
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter a number to check if it's an Armstrong number: 123
123 is not an Armstrong number.
PS D:\AI Assicoding>
```

Question 4: Context-Managed Prompting (Optimized Number Classification)

Design a context-managed prompt with clear instructions and constraints to generate an optimized Python program that classifies a number as prime, composite, or neither.

Task:

. Ensure proper input validation.

. Optimize the logic for efficiency.

. Compare the output with earlier prompting strategies.

Prompt:

```
'''
Docstring for Ass2.numclassify
write a python script to classify a given integer as prime, composite, or neither.

Constraints & Instructions:
- Accept only integer input and handle invalid inputs gracefully.
- Number less than or equal to 1 must be classified as 'neither'.
- The program should be readable,modular,and time-efficient.
-Output must be exactly one of the following strings: 'prime', 'composite', or 'neither'.

'''
```

Code:

```python
def classify_number(n):
    '''
    This function classifies a given integer as 'prime', 'composite', or 'neither'.
    '''
    if num<=1:
        return "Neither prime nor composite"
    for i in range(2,int(n**0.5)+1):
        if n%i==0:
            return "Composite"
    return "Prime"
#Get user input
if __name__ == "__main__":
    try:
        num = int(input("Enter an integer to classify: "))
        classification = classify_number(num)
        print(f"The number {num} is classified as: {classification}")
    except ValueError:
        print("Invalid input. Please enter a valid integer.")
```

Output:

```
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter an integer to classify: 1
The number 1 is classified as: Neither prime nor composite
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter an integer to classify: 2
The number 2 is classified as: Prime
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter an integer to classify: 4
The number 4 is classified as: Composite
The number 4 is classified as: Composite
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter an integer to classify: 0
Enter an integer to classify: 0
The number 0 is classified as: Neither prime nor composite
The number 0 is classified as: Neither prime nor composite
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter an integer to classify: -1
The number -1 is classified as: Neither prime nor composite
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter an integer to classify: a
Invalid input. Please enter a valid integer.
PS D:\AI Assicoding>
```

Explanation:

Zero-shot prompting is basic and simple code but it misses examples. One-shot prompting improves logic and structure with partial validation and moderate performance. Few-shot prompting gives clearer and more accurate code by learning from multiple examples, though efficiency is average. Context-managed prompting delivers highly optimized, fully validated, and professionally structured code.

Question 5: Zero-Shot Prompting (Perfect Number Check)

Write a zero-shot prompt (without providing any examples) to generate a Python function that checks whether a given number is a perfect number.

Task:

Record the Al-generated code.

Test the program with multiple inputs.

Identify any missing conditions or inefficiencies in the logic.

Prompt:

```
#Write a python program to check wether a given number is perfect or not.
```

Code:

```python
def is_perfect(n):
    """
    Check if a given number is a perfect number.

    :param n: The number to check.
    :return: True if the number is a perfect number, False otherwise.
    """
    if n <= 1:
        return False
    # Calculate the sum of proper divisors
    sum_of_divisors = sum(i for i in range(1, n) if n % i == 0)
    # A perfect number is equal to the sum of its proper divisors
    return sum_of_divisors == n
#Get user input
if __name__ == "__main__":
    try:
        num = int(input("Enter a number to check if it's a perfect number: "))
        if is_perfect(num):
            print(f"{num} is a perfect number.")
        else:
            print(f"{num} is not a perfect number.")
    except ValueError:
        print("Invalid input. Please enter a valid integer.")
```

Output:

```
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter a number to check if it's a perfect number: 2
2 is not a perfect number.
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter a number to check if it's a perfect number: 6
6 is a perfect number.
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter a number to check if it's a perfect number: 28
28 is a perfect number.
Enter a number to check if it's a perfect number: 28
28 is a perfect number.
28 is a perfect number.
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter a number to check if it's a perfect number: -1
-1 is not a perfect number.
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter a number to check if it's a perfect number: abc
Invalid input. Please enter a valid integer.
PS D:\AI Assicoding>
```

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 Al 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.

Prompt:

```
'''
Docstring for Ass2.evenodd
write a python script to check whether a given number is even or odd.
examples:
.Input:4 > Output:4 is even number
.Input:7 > Output:7 is odd number
'''
```

Code:

```python
def is_even_odd(number):
    """
    Check if a given number is even or odd.

    :param number: The number to check.
    :return: A string indicating whether the number is 'even' or 'odd'.
    """
    if number % 2 == 0:
        return "even"
    else:
        return "odd"
#Get user input
if __name__ == "__main__":
    try:
        num = int(input("Enter a number to check if it's even or odd: "))
        result = is_even_odd(num)
        print(f"{num} is an {result} number.")
    except ValueError:
        print("Invalid input. Please enter a valid integer.")
```

Output:

```
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter a number to check if it's even or odd: 2
2 is an even number.
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter a number to check if it's even or odd: 5
5 is an odd number.
Enter a number to check if it's even or odd: 5
5 is an odd number.
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter a number to check if it's even or odd: 0
0 is an even number.
PS D:\AI Assicoding> & C:/Python313/python.exe "d:/AI Assicoding/Zero_Short.py"
Enter a number to check if it's even or odd: -1
-1 is an odd number.
```