AI Asst Coding

Lab Assignment-3.1

2303A54012
A.Sanjay
B47A

Question 1: Zero-Shot Prompting – Palindrome Number Program

Prompt Type: Zero-Shot Prompting

Prompt Used:
Write a Python function that checks whether a given number is a palindrome. The function should take an integer as input and return whether it is a palindrome or not.

AI Generated Code:

```
def is_palindrome(num):
    temp = str(num)
    if temp == temp[::-1]:
        return "Palindrome"
    else:
        return "Not Palindrome"
```

Test Inputs and Outputs: - Input: 121 → Output: Palindrome
- Input: 123 → Output: Not Palindrome
- Input: 0 → Output: Palindrome
- Input: -121 → Output: Not Palindrome

Analysis / Observations:
The zero-shot prompt generated a simple and correct solution for positive numbers. However, negative numbers are not considered palindromes, which shows a limitation in edge-case handling.

Question 2: One-Shot Prompting – Factorial Calculation

Prompt Type: One-Shot Prompting

Prompt Used:
Input: 5 → Output: 120

Using the above example, write a Python function to calculate the factorial of a given number. The function should handle valid non-negative integers.

AI Generated Code:

```
def factorial(n):
    if n < 0:
        return "Invalid input"
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result
```

Test Inputs and Outputs: - Input: 5 → Output: 120
- Input: 0 → Output: 1
- Input: 1 → Output: 1

Analysis / Observations:
The one-shot prompt produced clearer and more reliable code than zero-shot prompting. Providing an example helped the AI understand the expected behavior.

Question 3: Few-Shot Prompting – Armstrong Number Check

Prompt Type: Few-Shot Prompting

Prompt Used:
Input: 153 → Output: Armstrong Number
Input: 370 → Output: Armstrong Number
Input: 123 → Output: Not an Armstrong Number

Based on the above examples, write a Python function to check whether a given number is an Armstrong number.

AI Generated Code:

```
def is_armstrong(num):
    digits = str(num)
    power = len(digits)
    total = 0
    for d in digits:
        total += int(d) ** power
    if total == num:
        return "Armstrong Number"
    else:
        return "Not an Armstrong Number"
```

Test Inputs and Outputs: - Input: 153 → Output: Armstrong Number
- Input: 370 → Output: Armstrong Number
- Input: 123 → Output: Not an Armstrong Number

Analysis / Observations:
Few-shot prompting significantly improved the accuracy and structure of the code. Multiple examples helped the AI generalize the logic correctly.

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

Prompt Type: Context-Managed Prompting

Prompt Used:
Write an optimized Python program that classifies a given number as Prime, Composite, or Neither. Ensure proper input validation and efficient logic.

AI Generated Code:

```
def classify_number(n):
    if n <= 1:
        return "Neither Prime nor Composite"
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return "Composite"
    return "Prime"
```

Test Inputs and Outputs: - Input: 2 → Output: Prime
- Input: 4 → Output: Composite
- Input: 1 → Output: Neither Prime nor Composite

Analysis / Observations:
Context-managed prompting resulted in optimized and well-structured code with efficient logic and proper classification.

Question 5: Zero-Shot Prompting – Perfect Number Check

Prompt Type: Zero-Shot Prompting

Prompt Used:
Write a Python function that checks whether a given number is a perfect number.

AI Generated Code:

```python
def is_perfect(num):
    if num <= 0:
        return "Not a Perfect Number"
    total = 0
    for i in range(1, num):
        if num % i == 0:
            total += i
    if total == num:
        return "Perfect Number"
    else:
        return "Not a Perfect Number"
```

Test Inputs and Outputs: - Input: 6 → Output: Perfect Number
- Input: 28 → Output: Perfect Number
- Input: 12 → Output: Not a Perfect Number

Analysis / Observations:
The zero-shot prompt generated correct logic but could be optimized further for performance.

Question 6: Few-Shot Prompting – Even or Odd with Validation

Prompt Type: Few-Shot Prompting

Prompt Used:
Input: 8 → Output: Even
Input: 15 → Output: Odd
Input: 0 → Output: Even

Write a Python program to determine whether a given number is even or odd with proper input validation.

AI Generated Code:

```python
def even_or_odd(num):
    if not isinstance(num, int):
        return "Invalid Input"
    if num % 2 == 0:
        return "Even"
    else:
        return "Odd"
```

Test Inputs and Outputs: - Input: 8 → Output: Even
- Input: 15 → Output: Odd
- Input: -4 → Output: Even
- Input: 3.5 → Output: Invalid Input

Analysis / Observations:
Few-shot prompting improved input handling and output clarity. Validation makes the program more robust.

Conclusion

This experiment demonstrated the importance of prompt engineering in AI-assisted coding. Zero-shot prompting provides basic solutions, while one-shot and few-shot prompting significantly improve clarity and correctness. Context-managed prompts generate optimized and efficient code.