Week2(3.1) -Monday-(AI Assisted Coding)

Lab Experiment: Prompt Engineering – Improving Prompts and Context Management

2303A51729

Batch-11

Lab Objectives

1. To understand and apply different prompt engineering techniques for generating Python programs using AI-assisted tools.

2. To analyze the impact of context and examples on the accuracy and efficiency of AI-generated code.

3. To develop and refine real-world Python applications through iterative prompt improvement.

Lab Outcomes

1. Students will be able to design effective prompts to generate

correct and optimized Python code.

2. Students will be able to compare and evaluate AI-generated

solutions produced using different prompting strategies.

3. Students will be able to implement and document real-world

Python applications using AI-assisted coding tools.

Experiment – Prompt Engineering Techniques

Task Description

Design and refine prompts using different prompting strategies to

generate Python programs for basic computational problems.

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.

```
# Write a zero-shot prompt (without providing any examples) to generate
# a Python function that checks whether a given number is a palindrome.
# A palindrome is a number that reads the same forwards and backwards.
# give user input and output of the code.
def is_palindrome(number):
    """
    Check if a given number is a palindrome.

    Parameters:
    number (int): The number to check.

    Returns:
    bool: True if the number is a palindrome, False otherwise.
    """
    str_num = str(number)
    return str_num == str_num[::-1]
# Example usage
if __name__ == "__main__":
    user_input = int(input("Enter a number: "))
    if is_palindrome(user_input):
        print(f"{user_input} is a palindrome.")
    else:
        print(f"{user_input} is not a palindrome.")
```

Explanation of the code.

The function converts the number to a string and checks if it is equal to its reverse.

Output:

```
PS C:\Users\Ajay Kumar\OneDrive\Desktop\Ai> & "C:/Program Files/Python39/python.exe" "c:/Users/Ajay Kumar/OneDrive/Desktop/Ai/lab(3.5).py"
Enter a number: 121
121 is a palindrome.
PS C:\Users\Ajay Kumar\OneDrive\Desktop\Ai> & "C:/Program Files/Python39/python.exe" "c:/Users/Ajay Kumar/OneDrive/Desktop/Ai/lab(3.5).py"
Enter a number: 2653
2653 is not a palindrome.
PS C:\Users\Ajay Kumar\OneDrive\Desktop\Ai>
```

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.

```python
# 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.
# give user input and output of the code with no comments.
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
# Example usage
if __name__ == "__main__":
    user_input = int(input("Enter a number to compute its factorial: "))
    result = factorial(user_input)
    print(f"The factorial of {user_input} is {result}.")
```

Explanation of the code.

The function uses recursion to compute the factorial of the given number by multiplying it with the factorial

of the number minus one until it reaches the base case of 0 or 1.

Output:

```
PS C:\Users\Ajay Kumar\OneDrive\Desktop\Ai> & "C:/Program Files/Python39/python.exe" "c:/Users/Ajay Kumar/OneDrive/De
sktop/Ai/lab(3.1).py"
Enter a number to compute its factorial: 5
The factorial of 5 is 120.
PS C:\Users\Ajay Kumar\OneDrive\Desktop\Ai> 7
7
PS C:\Users\Ajay Kumar\OneDrive\Desktop\Ai> & "C:/Program Files/Python39/python.exe" "c:/Users/Ajay Kumar/OneDrive/De
sktop/Ai/lab(3.1).py"
Enter a number to compute its factorial: 4
The factorial of 4 is 24.
PS C:\Users\Ajay Kumar\OneDrive\Desktop\Ai>
```

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

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

to guide the AI in generating a Python function to check whether a

given number is an Armstrong number.

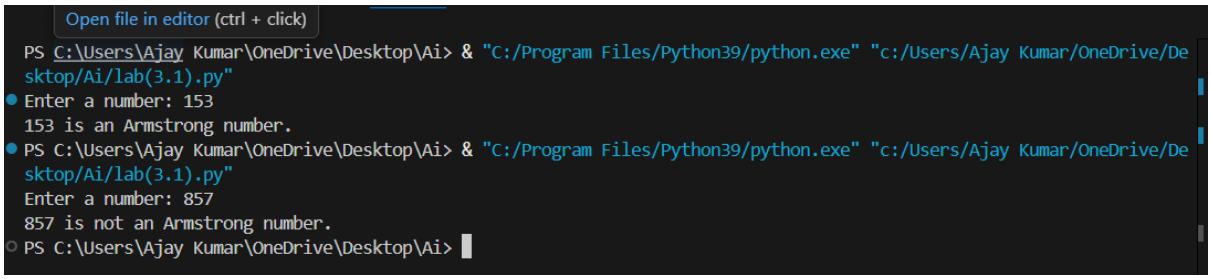
```python
# Question 3: Few-Shot Prompting (Armstrong Number Check)
# Write a few-shot prompt by providing multiple input-output examples
# to guide the AI in generating a Python function to check whether a
# given number is an Armstrong number.
# An Armstrong number is a number that is equal to the sum of its own digits
# each raised to the power of the number of digits.
# give uncommented code with user input and output of the code.
def is_armstrong_number(number):
    str_num = str(number)
    num_digits = len(str_num)
    sum_of_powers = sum(int(digit) ** num_digits for digit in str_num)
    return sum_of_powers == number
# Example usage
if __name__ == "__main__":
    user_input = int(input("Enter a number: "))
    if is_armstrong_number(user_input):
        print(f"{user_input} is an Armstrong number.")
    else:
        print(f"{user_input} is not an Armstrong number.")
```

Explanation of the code.

The function calculates the sum of each digit raised to the power of the number of digits

and checks if it is equal to the original number.

Output:



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

```python
# 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
# give uncommented code with user input and output of the code.
def classify_number(number):
    if number <= 1:
        return "neither"
    for i in range(2, int(number**0.5) + 1):
        if number % i == 0:
            return "composite"
    return "prime"
# Example usage
if __name__ == "__main__":
    user_input = int(input("Enter a number: "))
    classification = classify_number(user_input)
    print(f"{user_input} is classified as: {classification}.")
# Explanation of the code.
# The function checks if the number is less than or equal to 1 (neither),
# then checks for factors up to the square root of the number to classify it as prime or composite.
```

Output:

```
PS C:\Users\Ajay Kumar\OneDrive\Desktop\Ai> & "C:/Program Files/Python39/python.exe" "c:/Users/Ajay Kumar/OneDrive/De
sktop/Ai/lab(3.1).py"
Enter a number: 153
153 is an Armstrong number.
PS C:\Users\Ajay Kumar\OneDrive\Desktop\Ai> & "C:/Program Files/Python39/python.exe" "c:/Users/Ajay Kumar/OneDrive/De
sktop/Ai/lab(3.1).py"
Enter a number: 857
857 is not an Armstrong number.
PS C:\Users\Ajay Kumar\OneDrive\Desktop\Ai> & "C:/Program Files/Python39/python.exe" "c:/Users/Ajay Kumar/OneDrive/De
sktop/Ai/lab(3.1).py"
Enter a number: 5
5 is classified as: prime.
```

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 AI-generated code.
- Test the program with multiple inputs.
- Identify any missing conditions or inefficiencies in the logic.

```python
# generate a Python function that checks whether a given number is a
# perfect number.
# A perfect number is a positive integer that is equal to the sum of its
# proper positive divisors, excluding itself.
def is_perfect_number(number):
    if number < 1:
        return False
    divisors_sum = sum(i for i in range(1, number) if number % i == 0)
    return divisors_sum == number
# Example usage
if __name__ == "__main__":
    user_input = int(input("Enter a number: "))
    if is_perfect_number(user_input):
        print(f"{user_input} is a perfect number.")
    else:
        print(f"{user_input} is not a perfect number.")
# Explanation of the code.
# The function calculates the sum of all proper divisors of the number
# and checks if it is equal to the original number.
```

Output:

```
PS C:\Users\Ajay Kumar\OneDrive\Desktop\Ai> & "C:/Program Files/Python39/python.exe" "c:/Users/Ajay Kumar/OneDrive/De
sktop/Ai/lab(3.1).py"
Enter a number: 5
5 is classified as: prime.
PS C:\Users\Ajay Kumar\OneDrive\Desktop\Ai> & "C:/Program Files/Python39/python.exe" "c:/Users/Ajay Kumar/OneDrive/De
sktop/Ai/lab(3.1).py"
Enter a number: 28
28 is a perfect number.
PS C:\Users\Ajay Kumar\OneDrive\Desktop\Ai> & "C:/Program Files/Python39/python.exe" "c:/Users/Ajay Kumar/OneDrive/De
sktop/Ai/lab(3.1).py"
Enter a number: 56
56 is not a perfect number.
PS C:\Users\Ajay Kumar\OneDrive\Desktop\Ai>
```

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.

```python
# 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.
def classify_even_odd(number):
    if not isinstance(number, int):
        return "Invalid input. Please enter an integer."
    return "even" if number % 2 == 0 else "odd"
# Example usage
if __name__ == "__main__":
    try:
        user_input = int(input("Enter an integer: "))
        classification = classify_even_odd(user_input)
        print(f"{user_input} is {classification}.")
    except ValueError:
        print("Invalid input. Please enter a valid integer.")
# Explanation of the code.
# The function checks if the input is an integer and classifies it as even or odd.
# It also includes error handling for invalid inputs.
```

Output

```
PS C:\Users\Ajay Kumar\OneDrive\Desktop\Ai> & "C:/Program Files/Python39/python.exe" "c:/Users/Ajay Kumar/OneDrive/
sktop/Ai/lab(3.1).py"
Enter an integer: 7
7 is odd.
```