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

Lab Experiment: Prompt Engineering – Improving Prompts and Context Management

Hall ticket No:2303A51734

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.

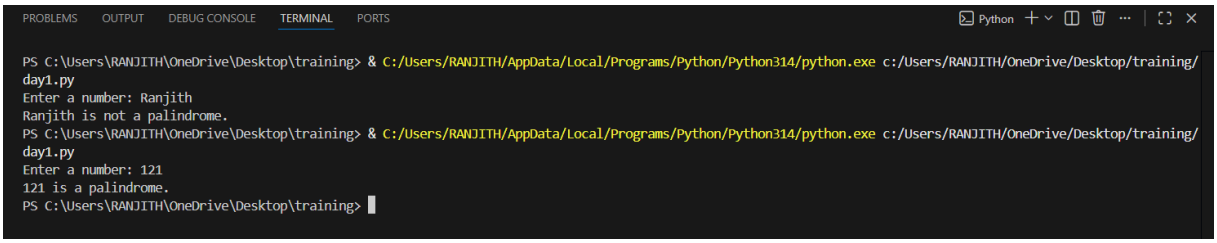
• Identify any logical errors or missing edge-case handling.

```python
number = int(input("Enter a number: "))
original = str(number)
reversed_num = original[::-1]

if original == reversed_num:
    print(f"{number} is a palindrome.")
else:
    print(f"{number} 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:



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.

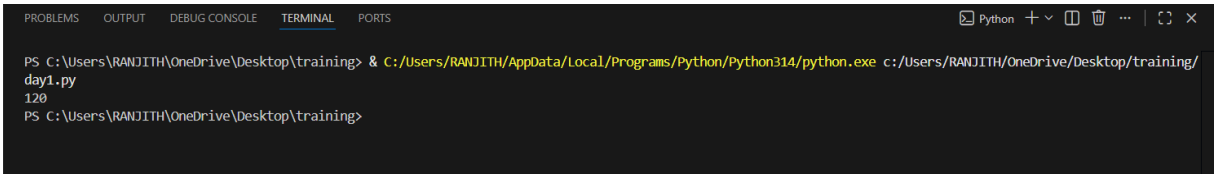
```python
def factorial(n):
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers.")
    if n == 0 or n == 1:
        return 1
    result = 1
    for i in range(2, n + 1):
        result *= i
    return result
print(factorial(5))
```

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:



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.

day1.py > armstrong_number

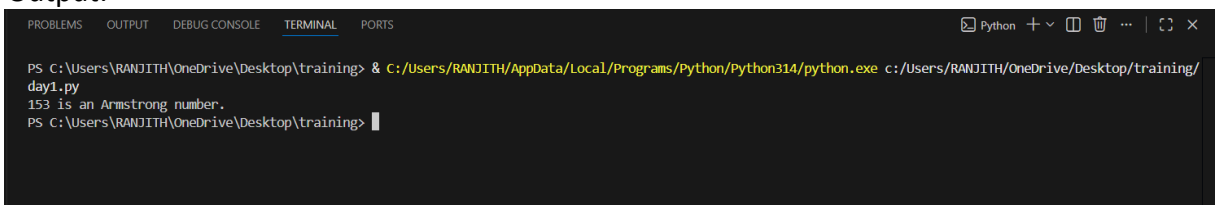
```python
def armstrong_number(num):
    num_str = str(num)
    num_digits = len(num_str)
    sum_of_powers = sum(int(digit) ** num_digits for digit in num_str)
    return sum_of_powers == num
if __name__ == "__main__":
    test_num = 153
    if armstrong_number(test_num):
        print(f"{test_num} is an Armstrong number.")
    else:
        print(f"{test_num} 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
def cla(parameter) number: Any
    if number <= 1:
        return "neither"
    for i in range(2, int(number**0.5) + 1):
        if number % i == 0:
            return "composite"
    return "prime"


if __name__ == "__main__":
    user_input = int(input("Enter a number: "))
    classification = classify_number(user_input)
    print(f"{user_input} is classified as {classification}.")

```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS                                                    Python

PS C:\Users\RANJITH\OneDrive\Desktop\training> & C:/Users/RANJITH/AppData/Local/Programs/Python/Python314/python.exe c:/Users/RANJITH/OneDrive/Desktop/training/
day1.py
Enter a number: 121
121 is classified as composite.
PS C:\Users\RANJITH\OneDrive\Desktop\training> & C:/Users/RANJITH/AppData/Local/Programs/Python/Python314/python.exe c:/Users/RANJITH/OneDrive/Desktop/training/
day1.py
Enter a number: 7
7 is classified as prime.
PS C:\Users\RANJITH\OneDrive\Desktop\training>
```

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

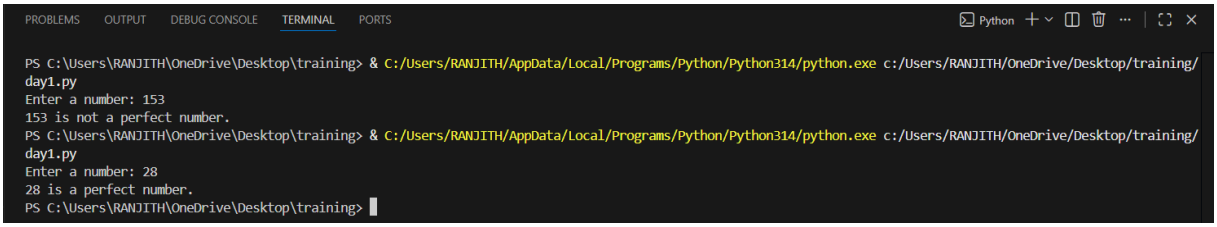
Write a zero-shot prompt (without providing any examples) to generate a Python function that checks whether a given number is a perfect number.

Task:

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

```python
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

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.")

```

Output:



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.

day1.py > ...

```python
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"


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.")

```

Output