SR UNIVERSITY

School of Computer Science and Artificial Intelligence

Lab Assignment # 3.1

Name of Student : GINNE SAI TEJA
Enrollment No. : 2303A51526
Batch No. : 22

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.

Code:-

ASSIGNMENT-4.py

C: > AIAC-LAB ASSIGNMENT-1 > ASSIGNMENT-4.py > ...

```python
#task-1
from logging import root


def palindrome(number):
    str_num = str(number)
    return str_num == str_num[::-1]
print(palindrome(121))
print(palindrome(123))
print(palindrome(12321))
print(palindrome(45654))
print(palindrome(789))
```

Output:-

```
\bundled\libs\debugpy\launcher' '6357
True
False
True
True
False
PS C:\AIAC-LAB ASSIGNMENT-1>
```

SR UNIVERSITY

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 function factorial(n) that returns the factorial of a non-negative integer n.

Code:-

```
#task-2
#Write a Python function factorial(n) that returns the factorial of a non-negative integer n.
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(5))
print(factorial(0))
print(factorial(7))
```

Output:-

```
120
1
5040
```

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.
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.
(Optional Extension)

Prompt:-
Write a Python function is_armstrong(n) that returns True if n is an Armstrong number and False otherwise.

Code:-

```python
#task-3
#Write a Python function is_armstrong(n) that returns True if n is an Armstrong number and False otherwise.
def is_armstrong(n):
    num_str = str(n)
    num_digits = len(num_str)
    sum_of_powers = sum(int(digit) ** num_digits for digit in num_str)
    return sum_of_powers == n
print(is_armstrong(153))
print(is_armstrong(9474))
print(is_armstrong(123))
print(is_armstrong(370))
print(is_armstrong(9475))
print(is_armstrong(0))
print(is_armstrong(1))
```

Output:-

```
True
True
False
True
False
True
True
```

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:-

Write a Python program using a context manager that classifies a number as prime, composite, or neither.

Code:-

```python
#task-4
#Write a Python program using a context manager that classifies a number as prime, composite, or neither.
class NumberClassifier:
    def __init__(self, number):
        self.number = number

    def __enter__(self):
        return self.classify_number()

    def __exit__(self, exc_type, exc_value, traceback):
        pass

    def classify_number(self):
        if self.number < 2:
            return "neither"
        for i in range(2, int(self.number ** 0.5) + 1):
            if self.number % i == 0:
                return "composite"
        return "prime"
with NumberClassifier(7) as classification:
    print(f"7 is {classification}.")
with NumberClassifier(10) as classification:
    print(f"10 is {classification}.")
with NumberClassifier(1) as classification:
    print(f"1 is {classification}.")
with NumberClassifier(0) as classification:
    print(f"0 is {classification}.")
with NumberClassifier(13) as classification:
    print(f"13 is {classification}.")
```

Output:-

```
7 is prime.
10 is composite.
1 is neither.
0 is neither.
13 is prime.
PS C:\Users\SaiTeja>
```

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.

Code:-

```python
#task-5
def perfect_number(n):
    if n < 2:
        return False
    divisors_sum = sum(i for i in range(1, n) if n % i == 0)
    return divisors_sum == n
print(perfect_number(6))
print(perfect_number(28))
print(perfect_number(12))
print(perfect_number(496))
print(perfect_number(15))
```

Output:-

```
True
True
False
True
False
```

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.
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:-
Write a Python function check_even_odd(n) that prints "Even" if n is even and "Odd" if n is odd.

Code:-

```python
#task-6
#Write a Python function check_even_odd(n) that prints "Even" if n is even and "Odd" if n is odd.
def check_even_odd(n):
    if n % 2 == 0:
        print("Even")
    else:
        print("Odd")
check_even_odd(4)
check_even_odd(7)
check_even_odd(0)
check_even_odd(-3)
```

Output:-

```
Even
Odd
Even
Odd
```