SR UNIVERSITY

School of Computer Science and Artificial Intelligence

Lab Assignment # 3

Name of Student : Gunturu Devi Prasad
Enrollment No. : 2303A51538
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:-

Assignment3.1.py > ...

C:\Users\Devi Prasad Gunturu\Desktop\Ai Assistant coding\Assignment3.1.py

```
from logging import root


def palindrome(number):
    str_num = str(number)
    return str_num == str_num[::-1]
print(palindrome(1221))
print(palindrome(12334))
print(palindrome(0123210))
print(palindrome(7456547))
print(palindrome(97899))

```

Output:-

```
sktop\Ai Assistant coding'; & 'c:\Users\Devi Prasad Gunturu\AppData\Local\Progr
ython.exe' 'c:\Users\Devi Prasad Gunturu\.vscode\extensions\ms-python.debugpy-2
dled\libs\debugpy\launcher' '63542' '--' 'c:\Users\Devi Prasad Gunturu\Desktop\
signment3.1.py'
True
False
True
True
False
```

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:input: 5 => output: 120 write a function to calculate factorial of a number

Code:-

```
15  #Task2
16  # input: 5 -> output: 120 write a function to calculate factorial of a
17  def factorial(n):
18      if n == 0 or n == 1:
19          return 1
20      else:
21          return n * factorial(n - 1)
22  print(factorial(7))
23  print(factorial(0))
24  print(factorial(3))
25
```

Output:-

```
120
1
720
```

SR UNIVERSITY

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:input: 370 => output: Armstrong write a function to check if a number is an Armstrong number.

Code:-

```
#Task3
# input: 153 -> output: Armstrong
# input: 123 -> output: Not Armstrong
# input: 370 -> output: Armstrong
# write a program check whether function to check if a number is an
def is_Armstrong(number):
    num_str = str(number)
    num_digits = len(num_str)
    sum_of_powers = sum(int(digit) ** num_digits for digit in num_s
    return sum_of_powers == number
def CheckArmstrong(number):
    if is_Armstrong(number):
        print("Armstrong")
    else:
        print("Not Armstrong")
CheckArmstrong(173)
CheckArmstrong(1283)
CheckArmstrong(3706)
CheckArmstrong(94745)

```

Output:-

```
Armstrong
Not Armstrong
Armstrong
Armstrong
```

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 program on a context-managed that classifies number as prime, composite or neither.

Code:-

```python
#Task4
# write a program on a context-managed that classifies number as prime, composite or neither.
class NumberClassifier:
    def __init__(self, number):
        self.number = number

    def __enter__(self):
        if self.number <= 1:
            self.classification = "Neither prime nor composite"
        elif self.number == 2:
            self.classification = "Prime"
        else:
            for i in range(2, int(self.number ** 0.5) + 1):
                if self.number % i == 0:
                    self.classification = "Composite"
                    break
            else:
                self.classification = "Prime"
        return self.classification

    def __exit__(self, exc_type, exc_value, traceback):
        pass
with NumberClassifier(7) as classification:
    print(classification)
with NumberClassifier(10) as classification:
    print(classification)
with NumberClassifier(1) as classification:
    print(classification)
with NumberClassifier(13) as classification:
    print(classification)
with NumberClassifier(15) as classification:
    print(classification)
with NumberClassifier(0) as classification:
    print(classification)
```

Output:-

```
Prime
Composite
Neither prime nor composite
Prime
Composite
Neither prime nor composite
```

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:-

```
83
84  #Task5
85  def perfect_number(n):
86      if n < 2:
87          return False
88      divisors_sum = sum(i for i in range(1, n) if n % i == 0)
89      return divisors_sum == n
90  print(perfect_number(6))
91  print(perfect_number(28))
92  print(perfect_number(12))
93  print(perfect_number(496))
94  print(perfect_number(15))
95
96
```

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:input: 8 ->
output: Even input: 15
-> output: Odd input: 0
-> output: Even
write a function to check if a number is even or odd.

```
96
97  #Task6
98  # input: 8 -> output: Even
99  # input: 15 -> output: Odd
100 # input: 0 -> output: Even write a function to check if a number
101 def even_or_odd(number):
102     return "Even" if number % 2 == 0 else "Odd"
103 print(even_or_odd(8))
104 print(even_or_odd(6.2))
105 print(even_or_odd(6/3))
106 print(even_or_odd(5**0.5))
107 print(even_or_odd(3.14159))
```

Code:Output:-

SR UNIVERSITY

```
Even
Odd
Even
Odd
Odd
```