

## School of Computer Science and Artificial Intelligence

---

Lab Assignment # 3

---

Name of Student : Pavan Juluri

Enrollment No. : 2303A51412

Batch No. 21

---

**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.py > ...
1  #Task1
2  from logging import root
3
4
5  def palindrome(number):
6      str_num = str(number)
7      return str_num == str_num[::-1]
8  print(palindrome(121))
9  print(palindrome(123))
10 print(palindrome(12321))
11 print(palindrome(45654))
12 print(palindrome(789))
13
```

Output:-

```
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:-**

```
Assignment3.py > ...
15  #Task2
16  # input: 5 -> output: 120 write a function to calculate factorial of a number
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(5))
23  print(factorial(0))
24  print(factorial(6))
25
```

**Output:-**

```
120
1
720
```

**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:-**

```
Assignment3.py > CheckArmstrong
27 #Task3
28 # input: 153 -> output: Armstrong
29 # input: 123 -> output: Not Armstrong
30 # input: 370 -> output: Armstrong
31 # write a program check whether function to check if a number is an Armstrong number or not Armstrong number.
32 def is_Armstrong(number):
33     num_str = str(number)
34     num_digits = len(num_str)
35     sum_of_powers = sum(int(digit) ** num_digits for digit in num_str)
36     return sum_of_powers == number
37 def CheckArmstrong(number):
38     if is_Armstrong(number):
39         print("Armstrong")
40     else:
41         print("Not Armstrong")
42 CheckArmstrong(153)
43 CheckArmstrong(123)
44 CheckArmstrong(370)
45 CheckArmstrong(9474)
```

**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:-**

```
Assignment3.py > ...
48     #Task4
49     # write a program on a context-managed that classifies number as prime, composite or neither.
50     class NumberClassifier:
51         def __init__(self, number):
52             self.number = number
53
54         def __enter__(self):
55             if self.number <= 1:
56                 self.classification = "Neither prime nor composite"
57             elif self.number == 2:
58                 self.classification = "Prime"
59             else:
60                 for i in range(2, int(self.number ** 0.5) + 1):
61                     if self.number % i == 0:
62                         self.classification = "Composite"
63                         break
64                 else:
65                     self.classification = "Prime"
66             return self.classification
67
68         def __exit__(self, exc_type, exc_value, traceback):
69             pass
70     with NumberClassifier(7) as classification:
71         print(classification)
72     with NumberClassifier(10) as classification:
73         print(classification)
74     with NumberClassifier(1) as classification:
75         print(classification)
76     with NumberClassifier(13) as classification:
77         print(classification)
78     with NumberClassifier(15) as classification:
79         print(classification)
80     with NumberClassifier(0) as classification:
81         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:-

```
#Task5
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:-

input: 8 -> output: Even

write a function to check if a number is even or odd.

**Code:-**

```
Assignment3.py > ...
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 is even or odd.
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))
```

**Output:-**

```
Even
Odd
Even
Odd
Odd
```