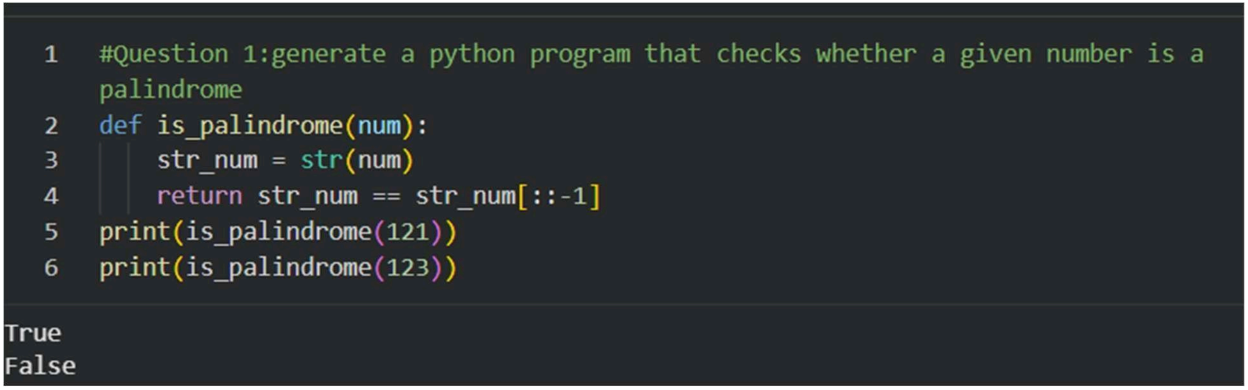
ASSIGNMENT-3.1

Name: Bandana Giri

HT. No: 2303A51590

Batch: 07
Lab Experiment: Prompt Engineering – Improving Prompts and Context Management.
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.

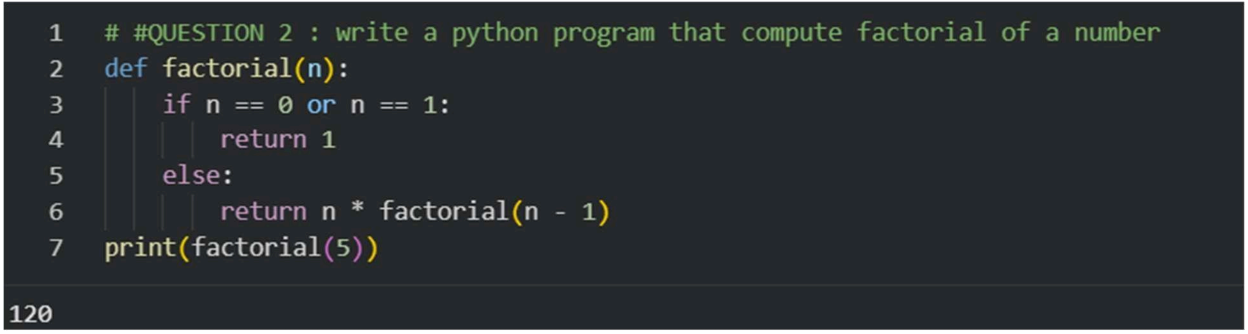


```python
#Question 1:generate a python program that checks whether a given number is a palindrome
def is_palindrome(num):
    str_num = str(num)
    return str_num == str_num[::-1]
print(is_palindrome(121))
print(is_palindrome(123))
```

```
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.
Task:

- Compare the generated code with a zero-shot solution.
- Examine improvements in clarity and correctness.



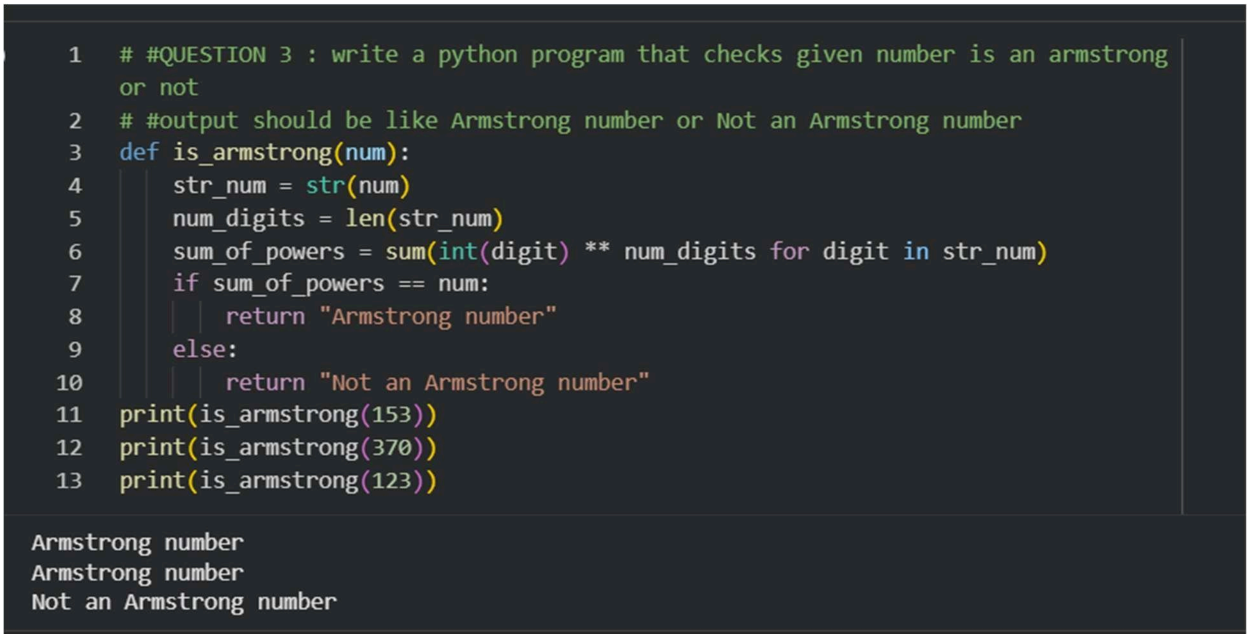
```python
# #QUESTION 2 : write a python program that compute factorial of a number
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(5))
```

```
120
```

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.
Task:

- Analyze how multiple examples influence code structure and accuracy.

- Test the function with boundary values and invalid inputs. (Optional Extension)



```python
# #QUESTION 3 : write a python program that checks given number is an armstrong or not
# #output should be like Armstrong number or Not an Armstrong number
def is_armstrong(num):
    str_num = str(num)
    num_digits = len(str_num)
    sum_of_powers = sum(int(digit) ** num_digits for digit in str_num)
    if sum_of_powers == num:
        return "Armstrong number"
    else:
        return "Not an Armstrong number"
print(is_armstrong(153))
print(is_armstrong(370))
print(is_armstrong(123))
```

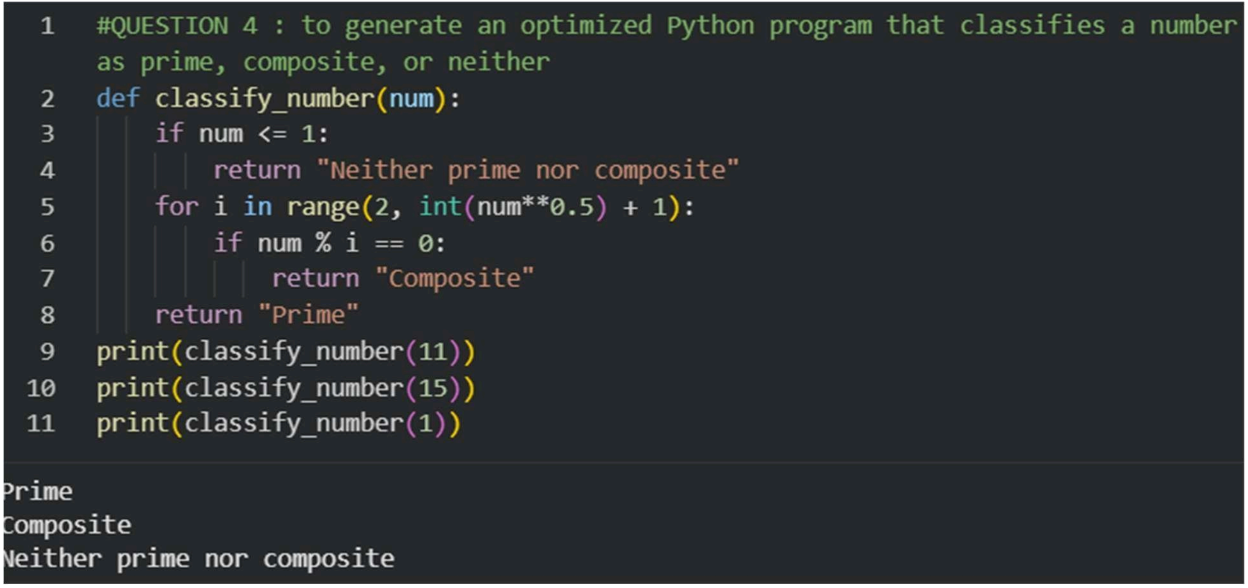
```
Armstrong number
Armstrong number
Not an Armstrong number
```

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.



```python
#QUESTION 4 : to generate an optimized Python program that classifies a number as prime, composite, or neither
def classify_number(num):
    if num <= 1:
        return "Neither prime nor composite"
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return "Composite"
    return "Prime"
print(classify_number(11))
print(classify_number(15))
print(classify_number(1))
```

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

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

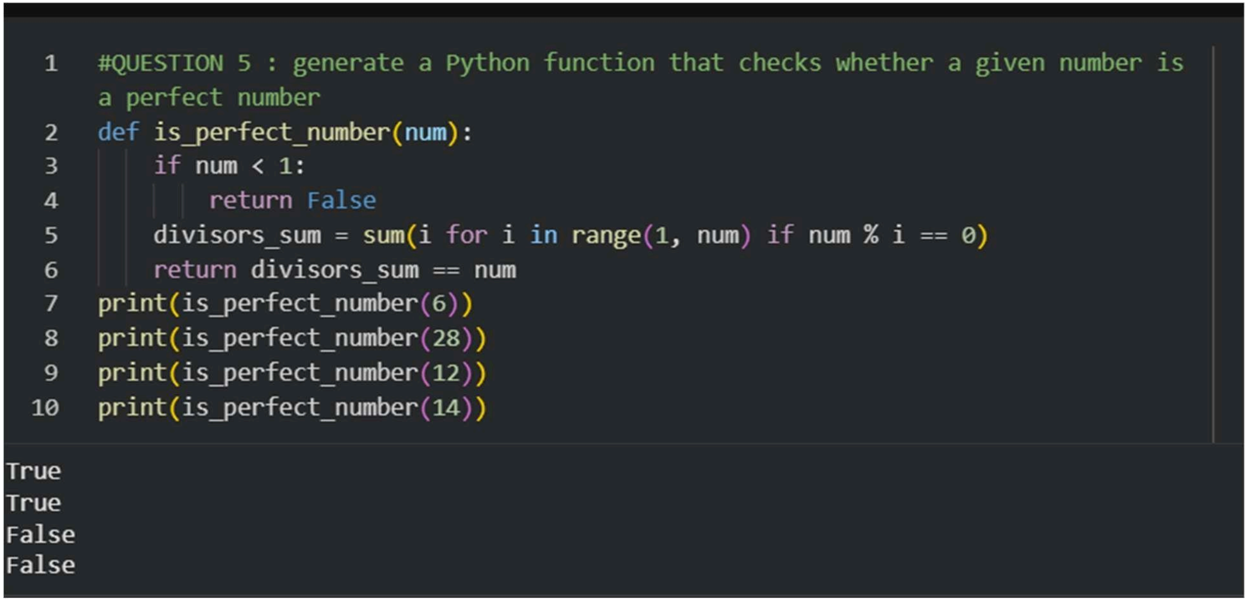
Write a zero-shot prompt (without providing any examples) to generate a Python function that checks whether a given number is a

perfect number.
Task:

• Record the AI-generated code.

• Test the program with multiple inputs.

• Identify any missing conditions or inefficiencies in the logic.



```python
#QUESTION 5 : generate a Python function that checks whether a given number is a perfect number
def is_perfect_number(num):
    if num < 1:
        return False
    divisors_sum = sum(i for i in range(1, num) if num % i == 0)
    return divisors_sum == num
print(is_perfect_number(6))
print(is_perfect_number(28))
print(is_perfect_number(12))
print(is_perfect_number(14))
```

```
True
True
False
False
```

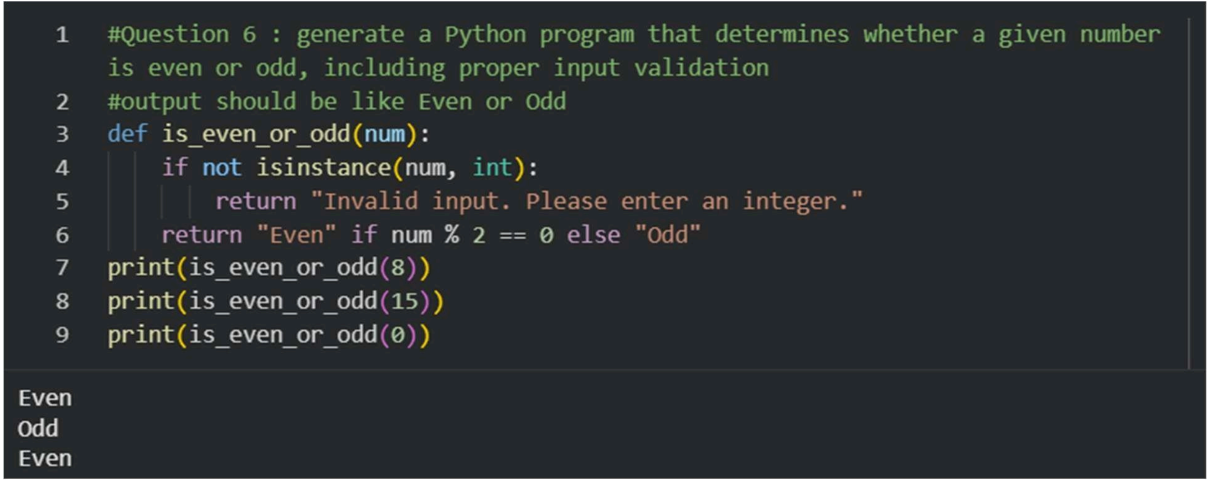
Question 6: Few-Shot Prompting (Even or Odd Classification with Validation)

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.

Task:

• Analyze how examples improve input handling and output clarity.

• Test the program with negative numbers and non-integer inputs.



```python
#Question 6 : generate a Python program that determines whether a given number is even or odd, including proper input validation
#output should be like Even or Odd
def is_even_or_odd(num):
    if not isinstance(num, int):
        return "Invalid input. Please enter an integer."
    return "Even" if num % 2 == 0 else "Odd"
print(is_even_or_odd(8))
print(is_even_or_odd(15))
print(is_even_or_odd(0))
```

```
Even
Odd
Even
```