3.1 - AI Assistant Coding Assignment

Name : p. Nandhini

Roll.No : 2303A51695

Batch : 28

Question 1: Zero-Shot Prompting (Palindrome Number Program)

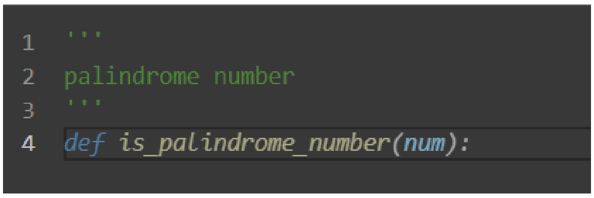
Write a zero-shot prompt (without providing any examples) to generate

a Python function that checks whether a given number is a palindrome.

Task:

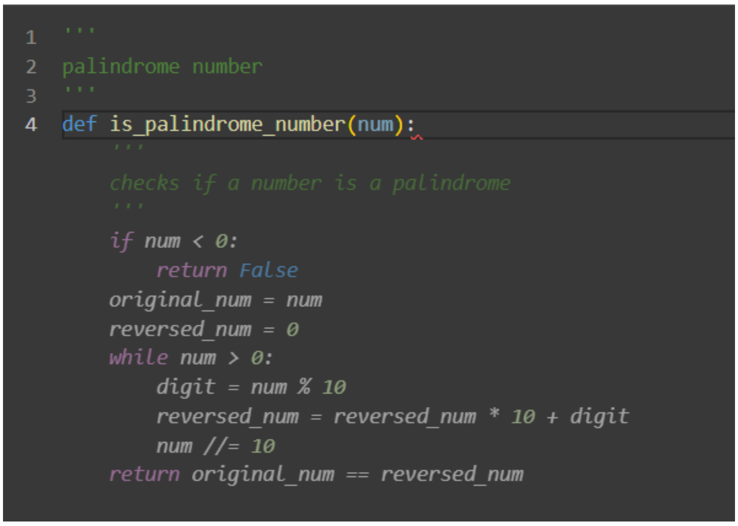
- **Record the AI-generated code.**
- **Test the code with multiple inputs.**
- **Identify any logical errors or missing edge-case handling.**

Prompt:



```python
'''
palindrome number
'''
def is_palindrome_number(num):
```

Output:



```python
'''
palindrome number
'''
def is_palindrome_number(num):
    '''
    checks if a number is a palindrome
    '''
    if num < 0:
        return False
    original_num = num
    reversed_num = 0
    while num > 0:
        digit = num % 10
        reversed_num = reversed_num * 10 + digit
        num //= 10
    return original_num == reversed_num
```

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.

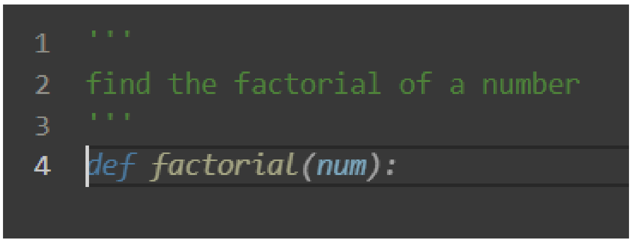
Example:

Input: 5 → Output: 120

Task:

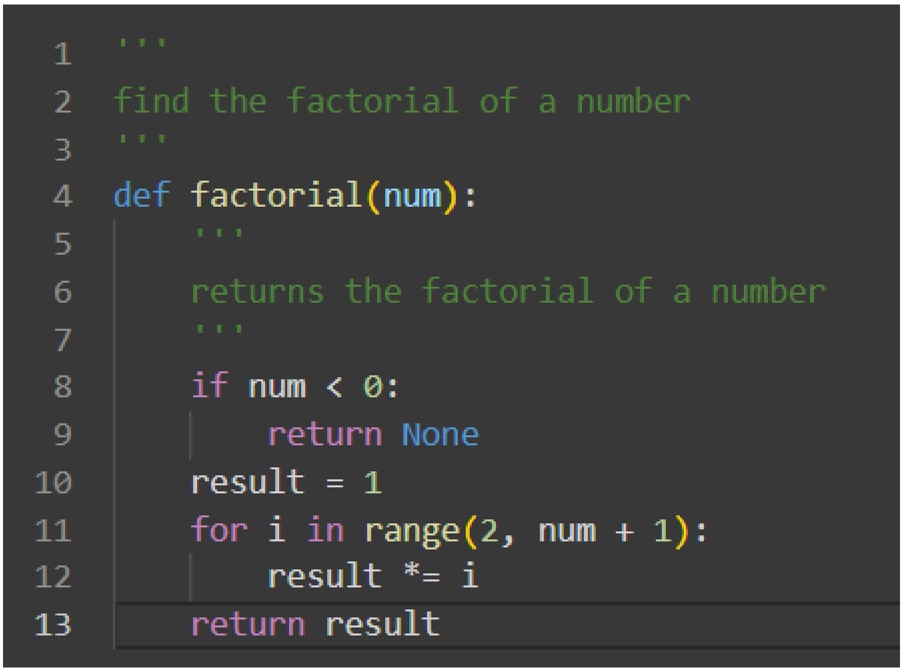
- **Compare the generated code with a zero-shot solution.**
- **Examine improvements in clarity and correctness.**

Prompt:



```python
'''
find the factorial of a number
'''
def factorial(num):
```

Output:



```python
'''
find the factorial of a number
'''
def factorial(num):
    '''
    returns the factorial of a number
    '''
    if num < 0:
        return None
    result = 1
    for i in range(2, num + 1):
        result *= i
    return result
```

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.

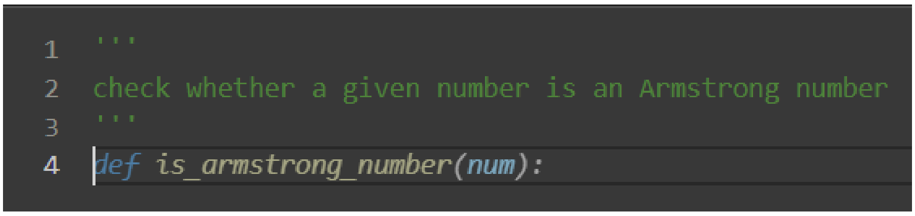
Examples:

- **Input: 153 → Output: Armstrong Number**
- **Input: 370 → Output: Armstrong Number**
- **Input: 123 → Output: Not an Armstrong Number**

Task:

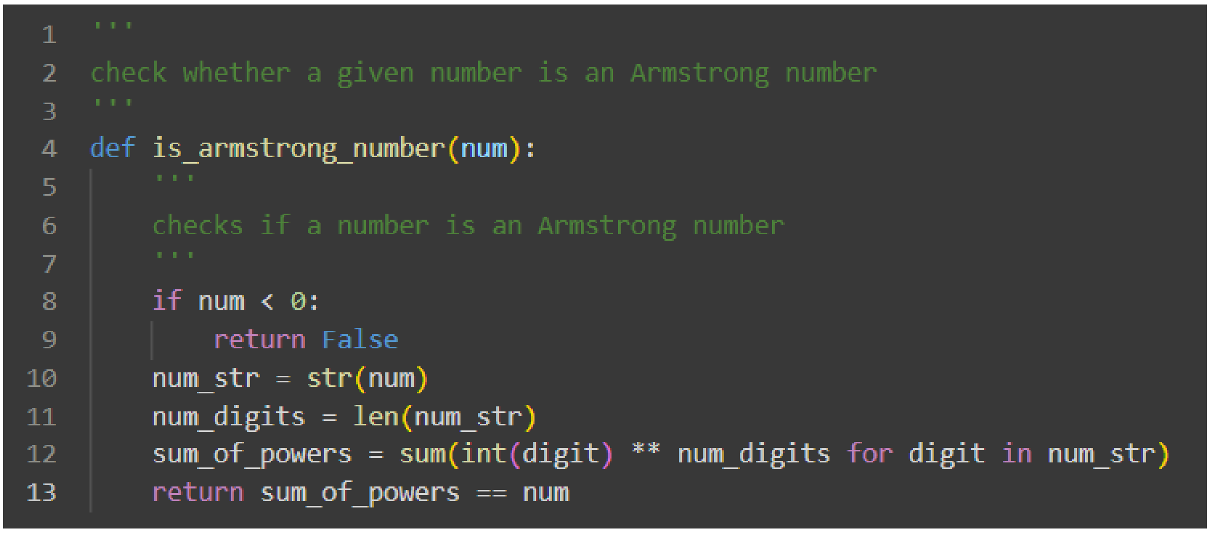
- **Analyze how multiple examples influence code structure and accuracy.**
- **Test the function with boundary values and invalid inputs.**

Prompt:



```python
'''
check whether a given number is an Armstrong number
'''
def is_armstrong_number(num):
```

Output:



```python
'''
check whether a given number is an Armstrong number
'''
def is_armstrong_number(num):
    '''
    checks if a number is an Armstrong number
    '''
    if num < 0:
        return False
    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
```

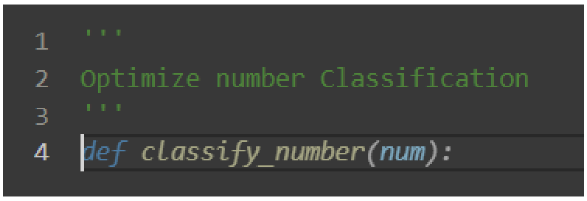
Question 4: Context-Managed Prompting (Optimized Number Classification)

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:

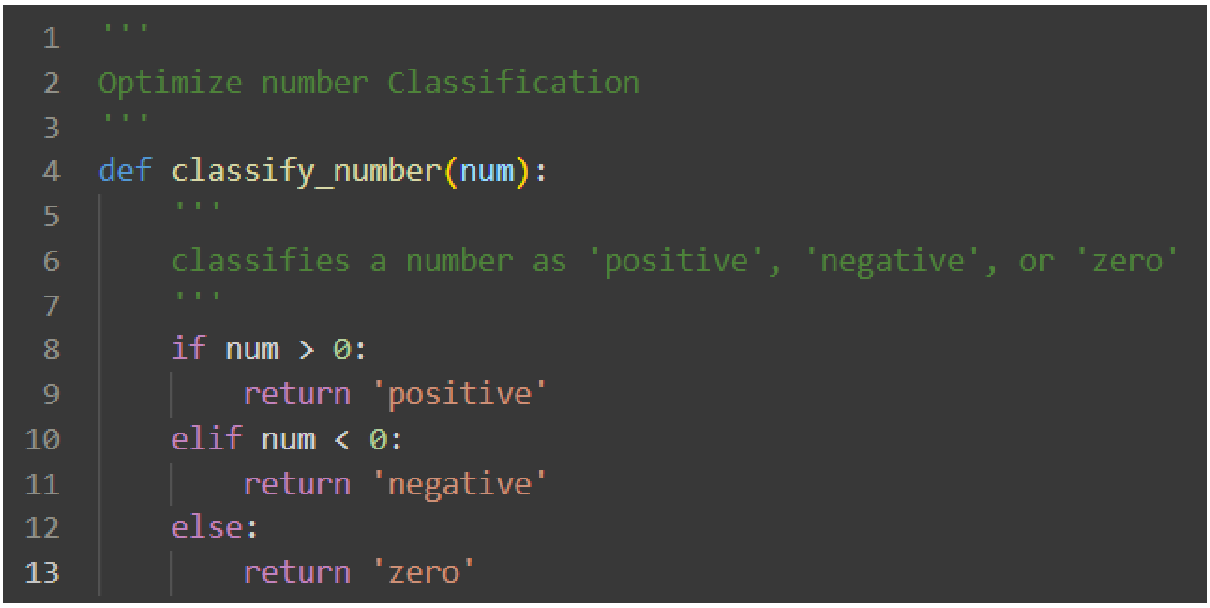
- **Ensure proper input validation.**
- **Optimize the logic for efficiency.**
- **Compare the output with earlier prompting strategies.**

Prompt:



```
'''
Optimize number Classification
'''
def classify_number(num):
```

Output:



```
'''
Optimize number Classification
'''
def classify_number(num):
    '''
    classifies a number as 'positive', 'negative', or 'zero'
    '''
    if num > 0:
        return 'positive'
    elif num < 0:
        return 'negative'
    else:
        return 'zero'
```

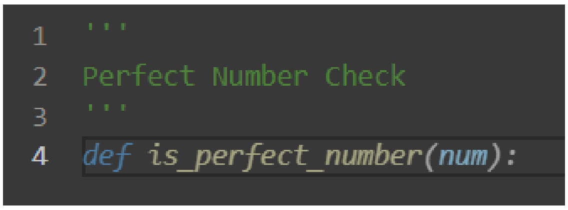
Question 5: Zero-Shot Prompting (Perfect Number Check)

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:

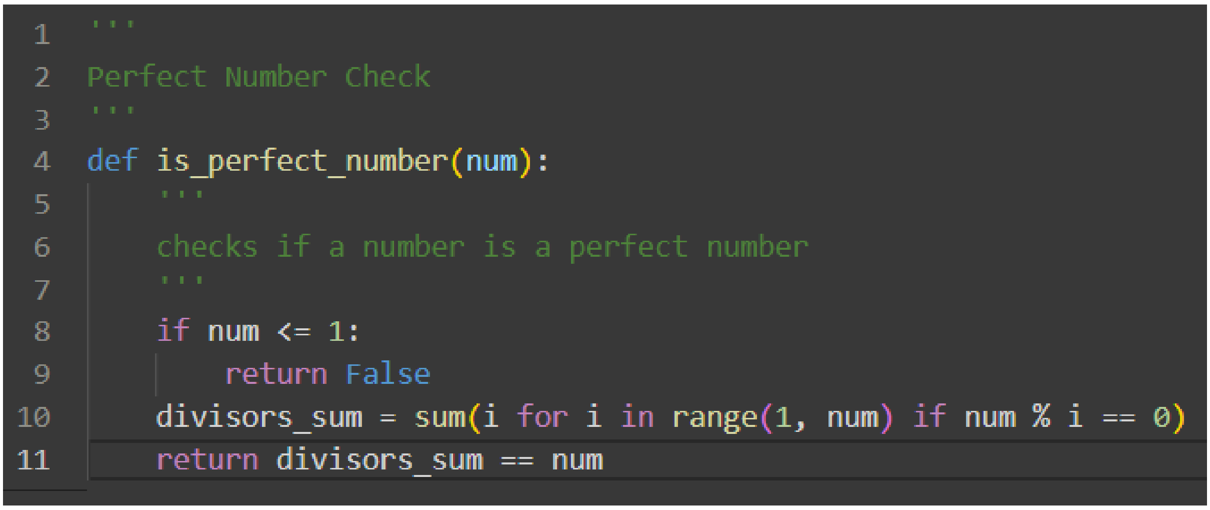
- **Record the AI-generated code.**
- **Test the program with multiple inputs.**
- **Identify any missing conditions or inefficiencies in the logic.**

Prompt:



```python
'''
Perfect Number Check
'''
def is_perfect_number(num):
```

Output:



```python
'''
Perfect Number Check
'''
def is_perfect_number(num):
    '''
    checks if a number is a perfect number
    '''
    if num <= 1:
        return False
    divisors_sum = sum(i for i in range(1, num) if num % i == 0)
    return divisors_sum == num
```

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.

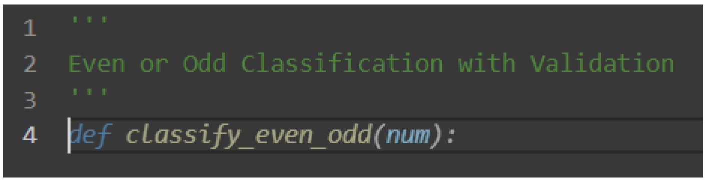
Examples:

- **Input: 8 → Output: Even**
- **Input: 15 → Output: Odd**
- **Input: 0 → Output: Even**

Task:

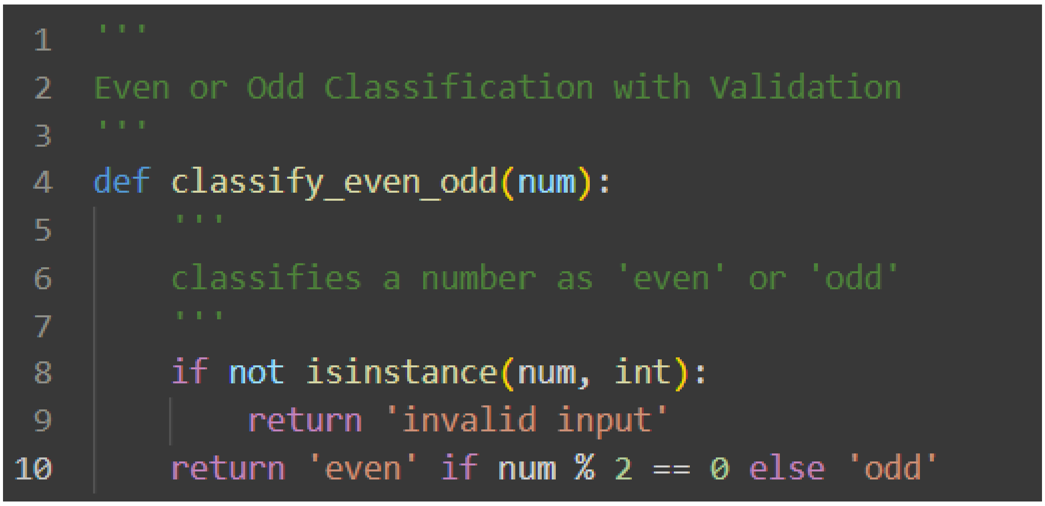
- **Analyze how examples improve input handling and output clarity.**
- **Test the program with negative numbers and non-integer inputs.**

Prompt:



```python
'''
Even or Odd Classification with Validation
'''
def classify_even_odd(num):
```

Output:



```python
'''
Even or Odd Classification with Validation
'''
def classify_even_odd(num):
    '''
    classifies a number as 'even' or 'odd'
    '''
    if not isinstance(num, int):
        return 'invalid input'
    return 'even' if num % 2 == 0 else 'odd'
```