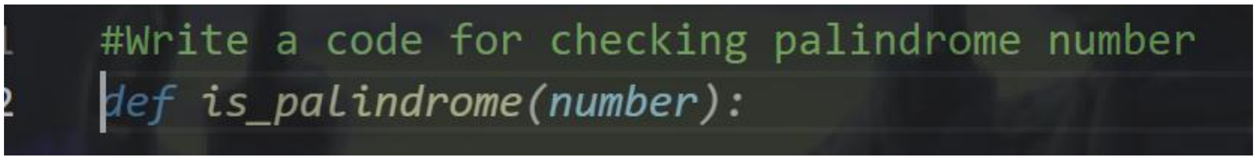
Question 1: Zero-Shot Prompting (Palindrome Number Program)

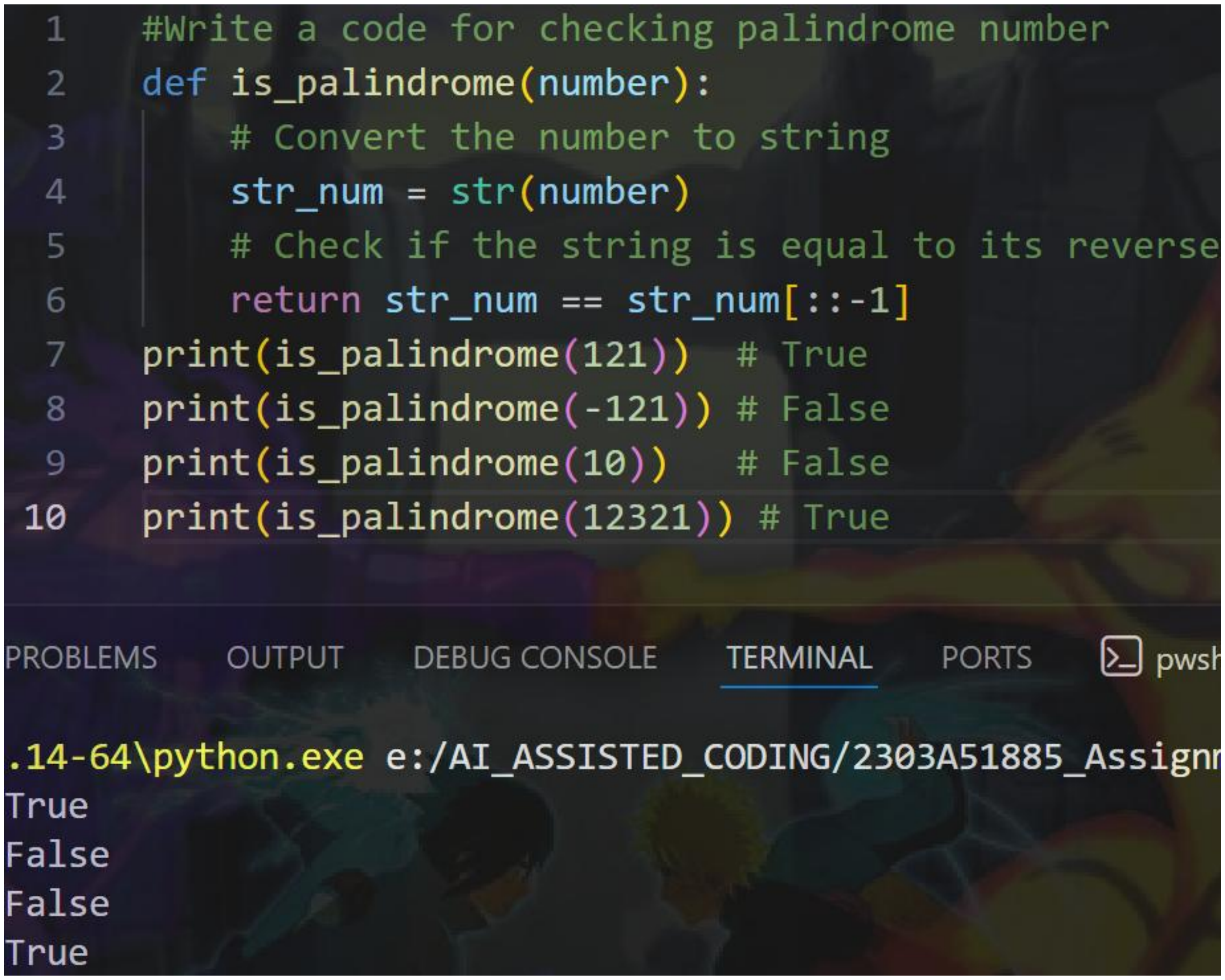
Write a **zero-shot prompt** (without providing any examples) to generate a Python function that checks whether a given number is a palindrome.

Zero-shot prompt



```
#Write a code for checking palindrome number
def is_palindrome(number):
```

Code:



```
#Write a code for checking palindrome number
def is_palindrome(number):
    # Convert the number to string
    str_num = str(number)
    # Check if the string is equal to its reverse
    return str_num == str_num[::-1]
print(is_palindrome(121))  # True
print(is_palindrome(-121)) # False
print(is_palindrome(10))   # False
print(is_palindrome(12321)) # True
```

```
.14-64\python.exe e:/AI_ASSISTED_CODING/2303A51885_Assign
True
False
False
True
```

Improvements or edge cases:

So for the above code the edges cases or like it is not handling any strings or float values so when we give a string or float number we have to return enter a valid number only. This is the edge case that is missing in the code.

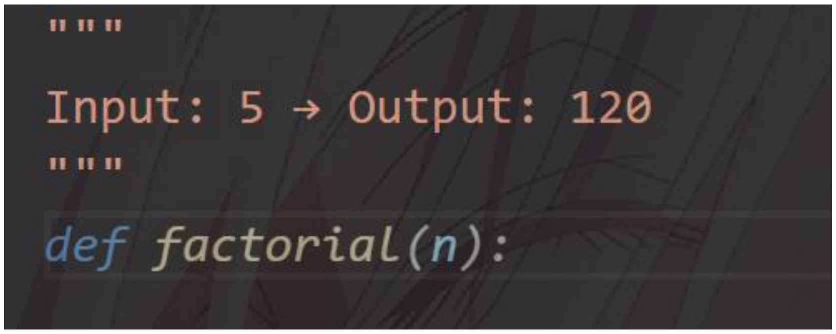
Question 2: One-Shot Prompting (Factorial Calculation)

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:

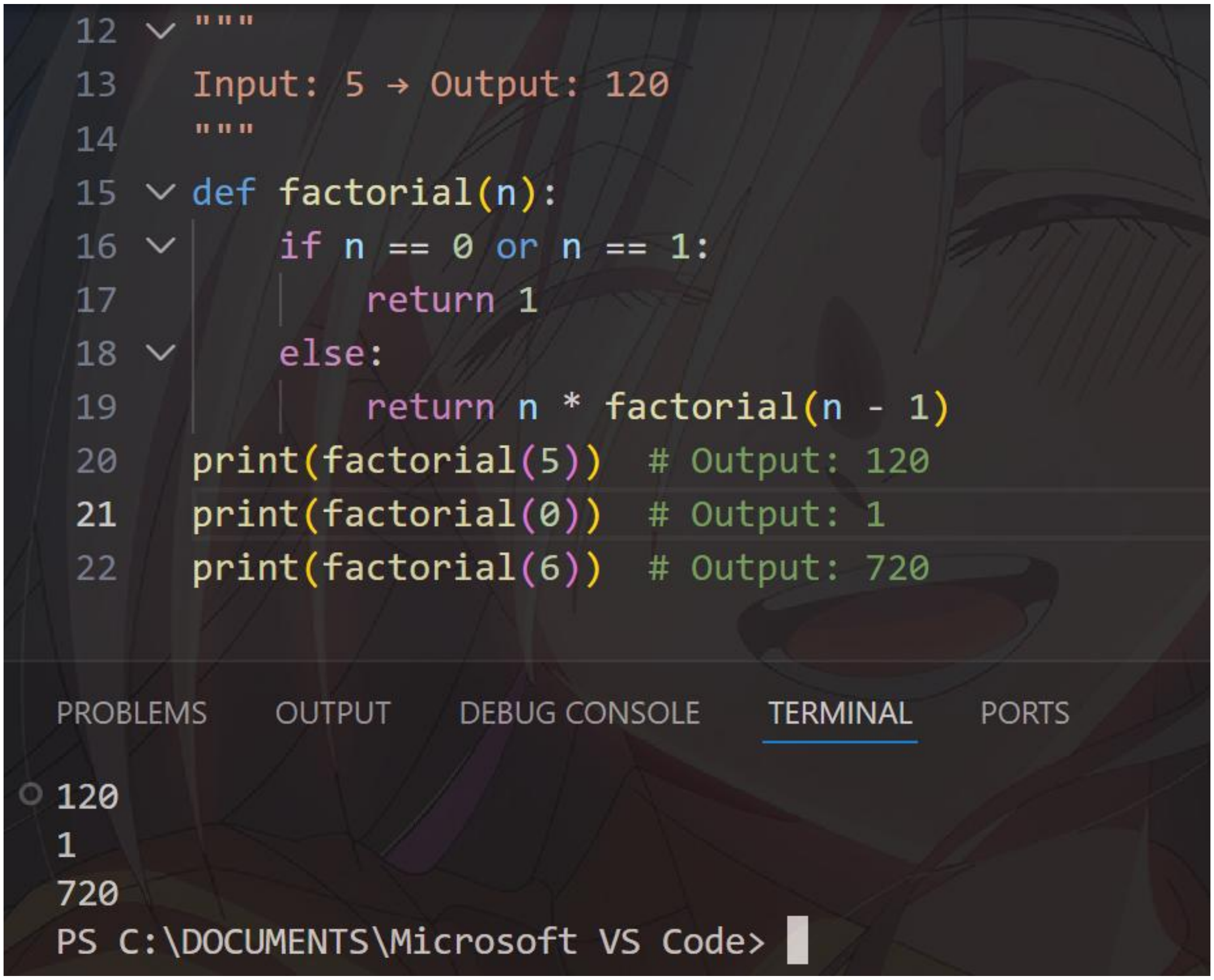
Input: 5 → Output: 120

<u>**One-shot prompt**</u>



```python
"""
Input: 5 → Output: 120
"""
def factorial(n):
```

Code:



```python
"""
Input: 5 → Output: 120
"""
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(5))  # Output: 120
print(factorial(0))  # Output: 1
print(factorial(6))  # Output: 720
```

```
120
1
720
PS C:\DOCUMENTS\Microsoft VS Code>
```

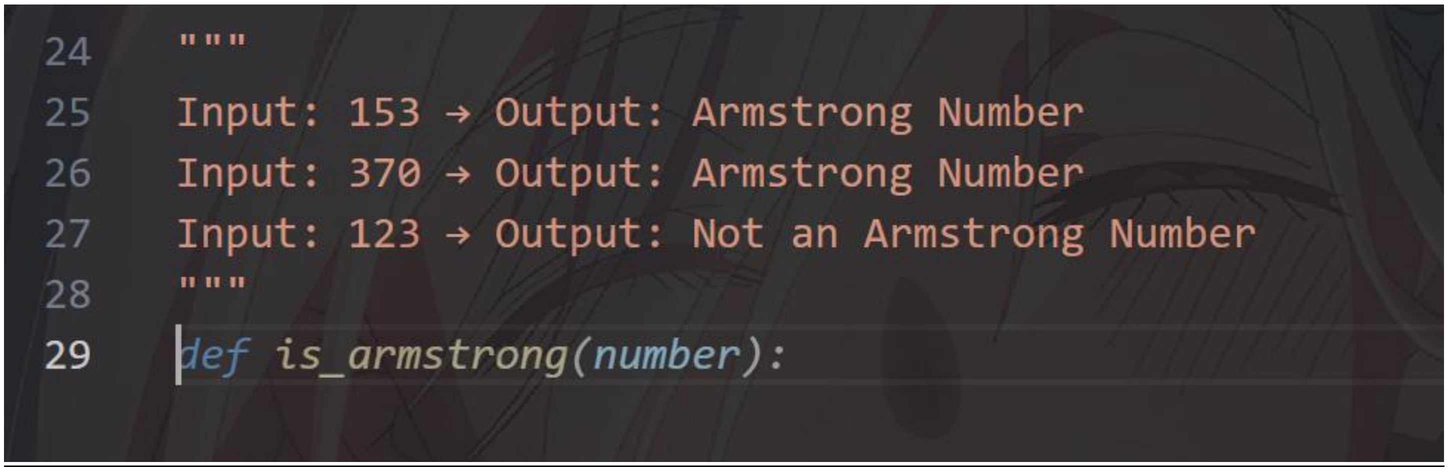
Question 3: Few-Shot Prompting (Armstrong Number Check)

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:

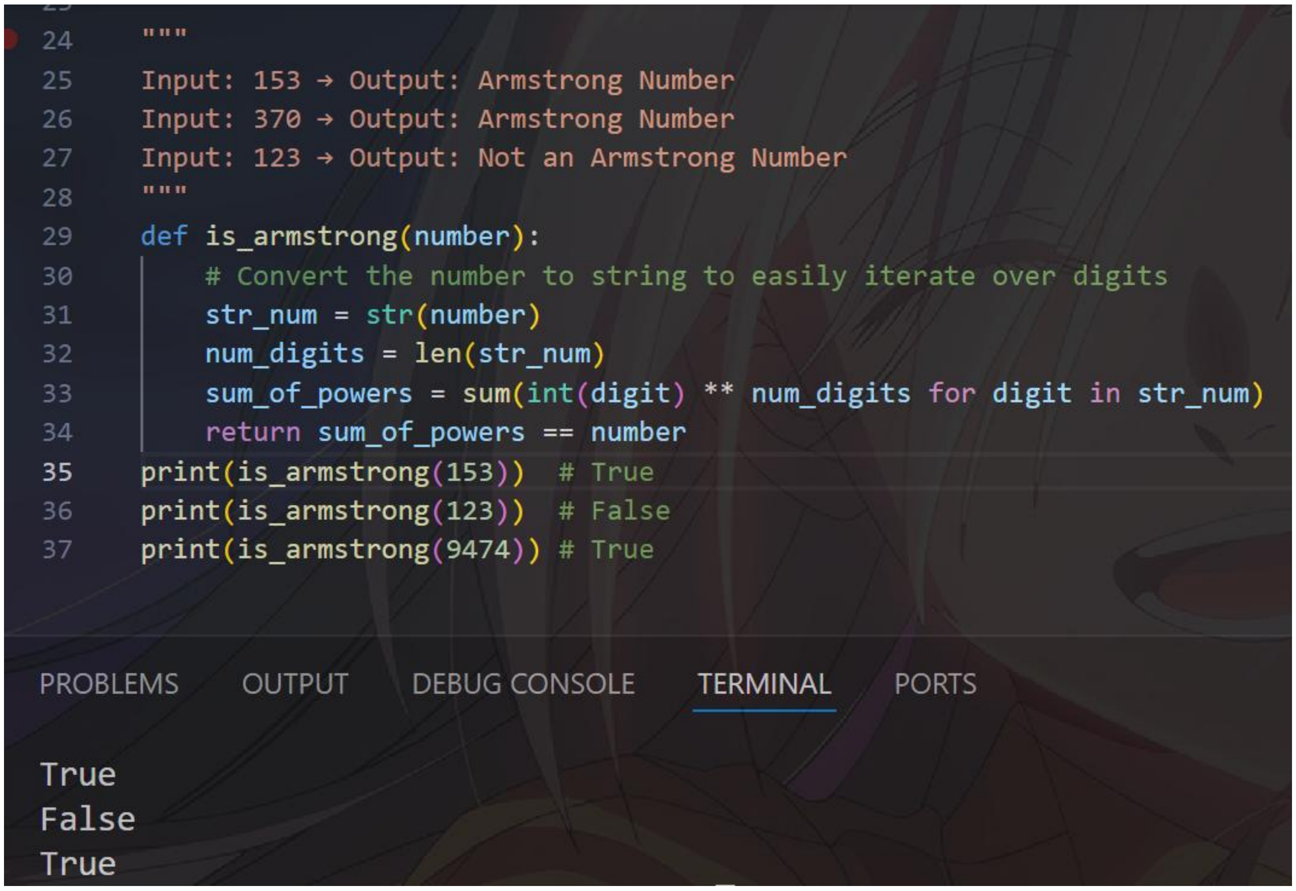
- Input: 153 → Output: Armstrong Number
- Input: 370 → Output: Armstrong Number
- Input: 123 → Output: Not an Armstrong Number

Few-Shot Prompt:



```
"""
Input: 153 → Output: Armstrong Number
Input: 370 → Output: Armstrong Number
Input: 123 → Output: Not an Armstrong Number
"""
def is_armstrong(number):
```

Code:



```
"""
Input: 153 → Output: Armstrong Number
Input: 370 → Output: Armstrong Number
Input: 123 → Output: Not an Armstrong Number
"""
def is_armstrong(number):
    # Convert the number to string to easily iterate over digits
    str_num = str(number)
    num_digits = len(str_num)
    sum_of_powers = sum(int(digit) ** num_digits for digit in str_num)
    return sum_of_powers == number
print(is_armstrong(153))  # True
print(is_armstrong(123))  # False
print(is_armstrong(9474)) # True
```

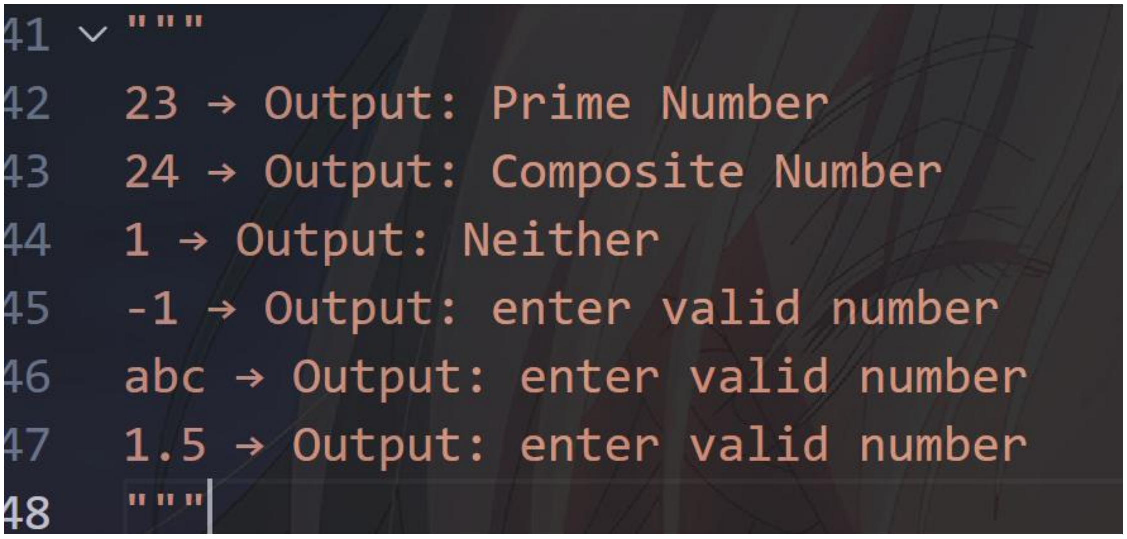
```
True
False
True
```

(Optional Extension)

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

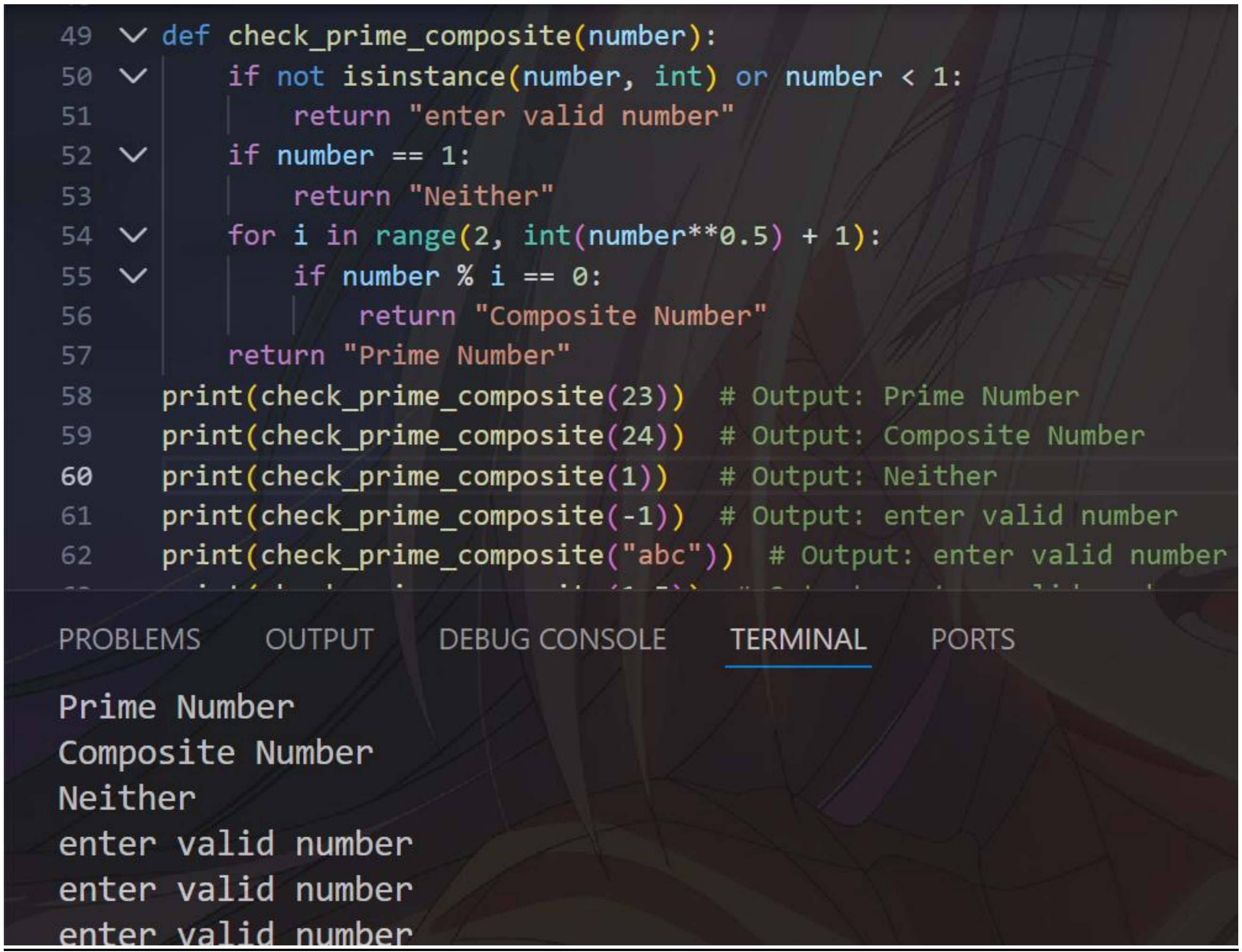
Design a **context-managed prompt** with clear instructions and constraints to generate an optimized Python program that classifies a number as **prime, composite, or neither**.

__Multi- Shot prompt:__



```
"""
23 → Output: Prime Number
24 → Output: Composite Number
1 → Output: Neither
-1 → Output: enter valid number
abc → Output: enter valid number
1.5 → Output: enter valid number
"""
```

__Code__



```python
def check_prime_composite(number):
    if not isinstance(number, int) or number < 1:
        return "enter valid number"
    if number == 1:
        return "Neither"
    for i in range(2, int(number**0.5) + 1):
        if number % i == 0:
            return "Composite Number"
    return "Prime Number"
print(check_prime_composite(23))  # Output: Prime Number
print(check_prime_composite(24))  # Output: Composite Number
print(check_prime_composite(1))   # Output: Neither
print(check_prime_composite(-1))  # Output: enter valid number
print(check_prime_composite("abc"))  # Output: enter valid number
```

```
Prime Number
Composite Number
Neither
enter valid number
enter valid number
enter valid number
```

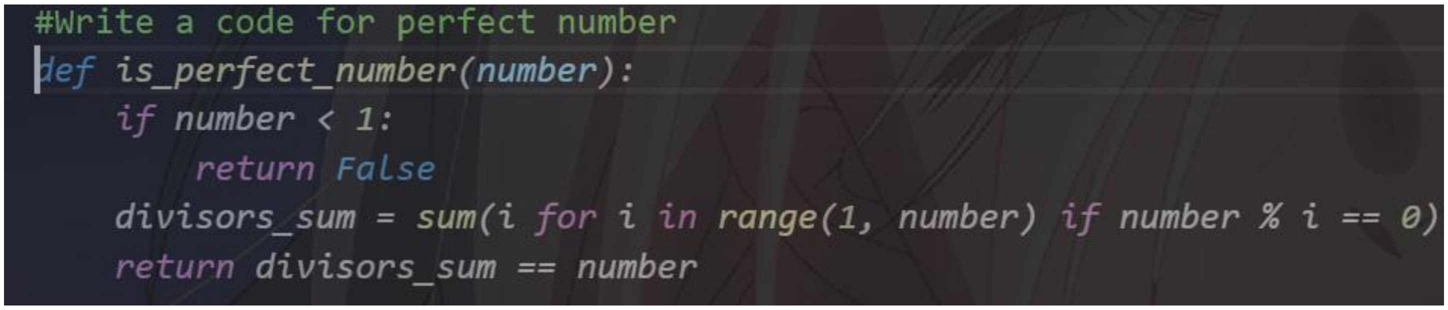
Using Multi-Shot prompt

Multi-Short prompting is a technique where we give multiple examples as a prompt by which we observes the keywords and trends in the values and the AI helps us in coding it. So Multi Shot helps us in the code

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

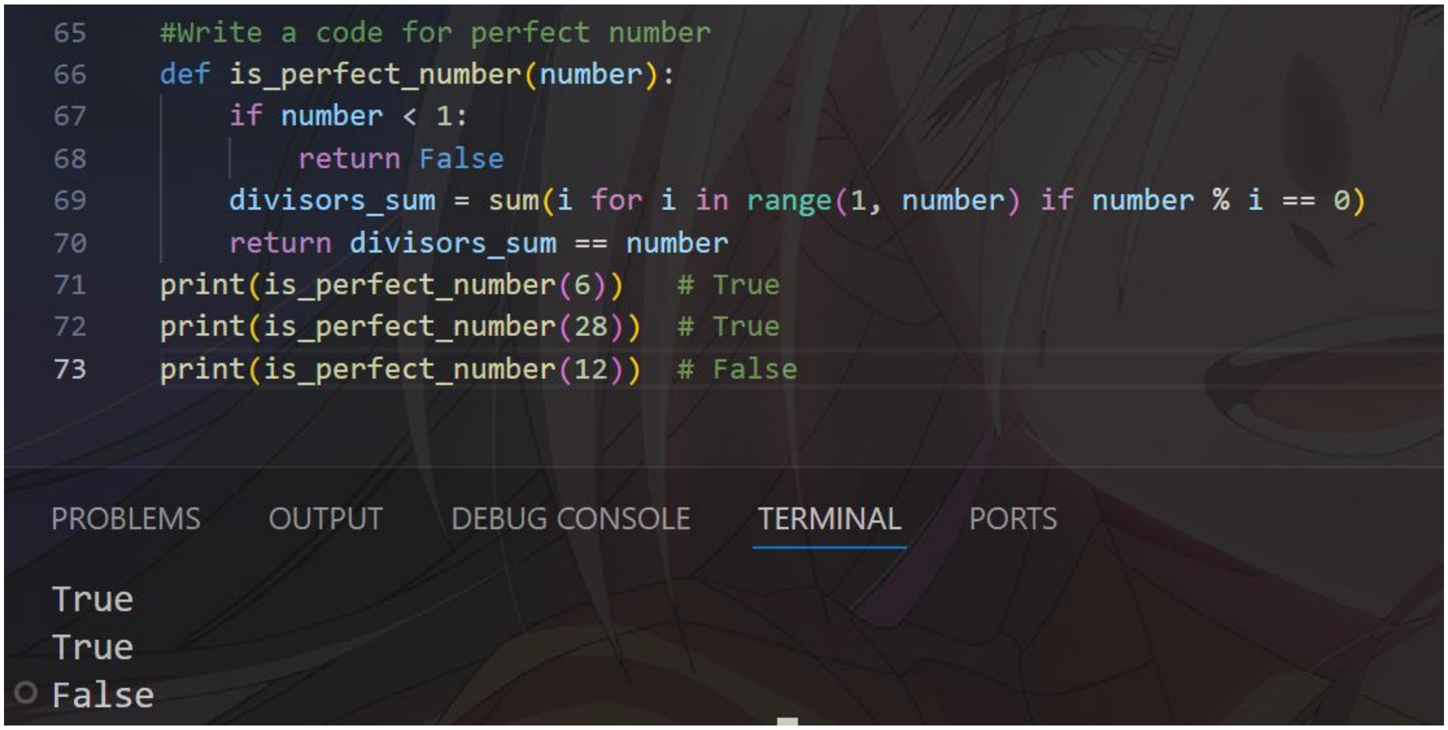
Write a zero-shot prompt (without providing any examples) to generate a Python function that checks whether a given number is a perfect number.

<u>**Zero-Shot Prompt**</u>



```python
#Write a code for perfect number
def is_perfect_number(number):
    if number < 1:
        return False
    divisors_sum = sum(i for i in range(1, number) if number % i == 0)
    return divisors_sum == number
```

Code:



```python
#Write a code for perfect number
def is_perfect_number(number):
    if number < 1:
        return False
    divisors_sum = sum(i for i in range(1, number) if number % i == 0)
    return divisors_sum == number
print(is_perfect_number(6))   # True
print(is_perfect_number(28))  # True
print(is_perfect_number(12))  # False
```

```
True
True
False
```

Improvements:

It is missing the negative inputs case and float input case and strings case

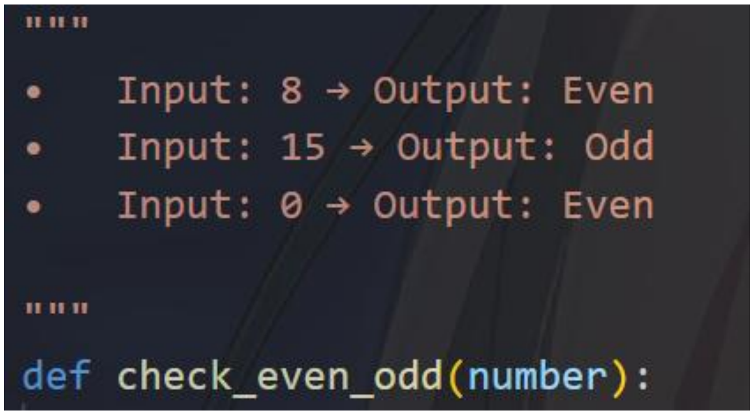
Question 6: Few-Shot Prompting (Even or Odd Classification with Validation)

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

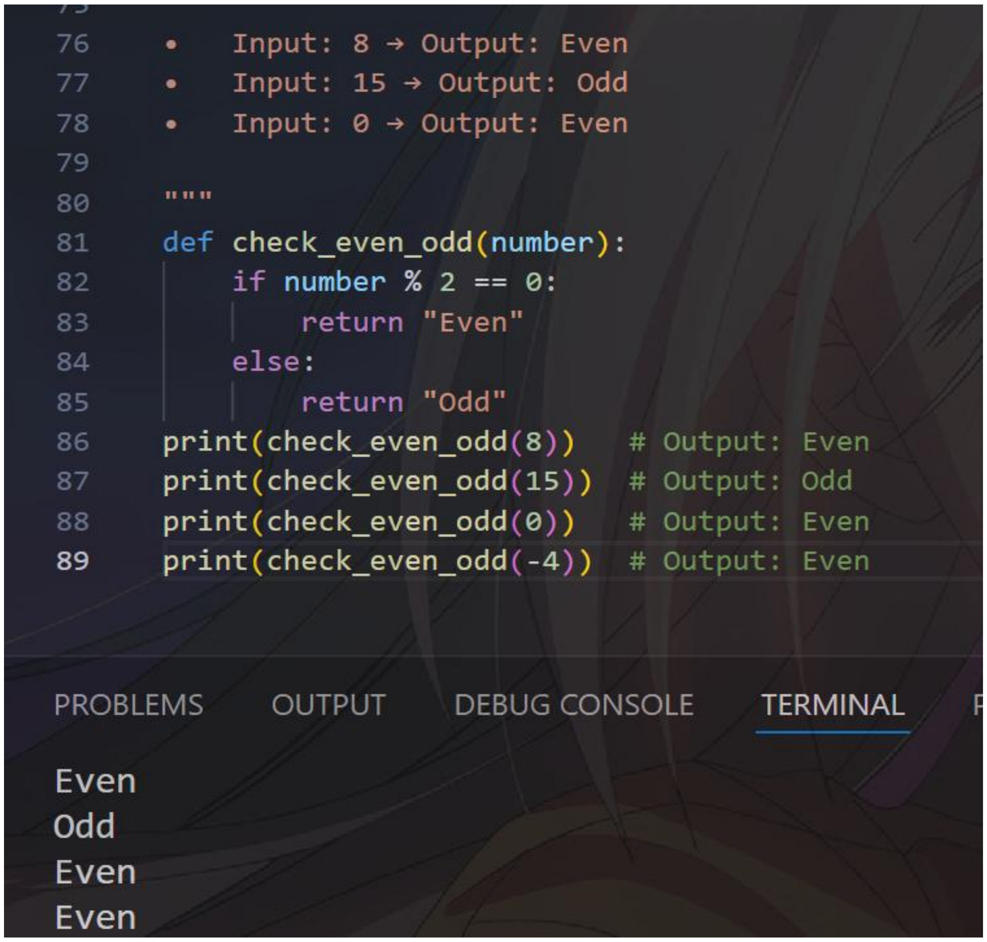
Few-Shot Prompting



```
"""
•   Input: 8 → Output: Even
•   Input: 15 → Output: Odd
•   Input: 0 → Output: Even

"""
def check_even_odd(number):
```

Code:



```
•   Input: 8 → Output: Even
•   Input: 15 → Output: Odd
•   Input: 0 → Output: Even

"""
def check_even_odd(number):
    if number % 2 == 0:
        return "Even"
    else:
        return "Odd"
print(check_even_odd(8))    # Output: Even
print(check_even_odd(15))   # Output: Odd
print(check_even_odd(0))    # Output: Even
print(check_even_odd(-4))   # Output: Even
```

```
Even
Odd
Even
Even
```