AI Assistant Coding

Assignment 3.1

Name : Hari Priya. G **HT. No :** 2303A51104 **Batch:** 1

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.

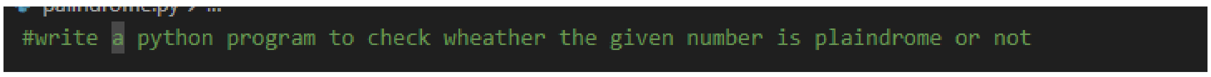
Task:

• Record the AI-generated code.

• Test the code with multiple inputs.

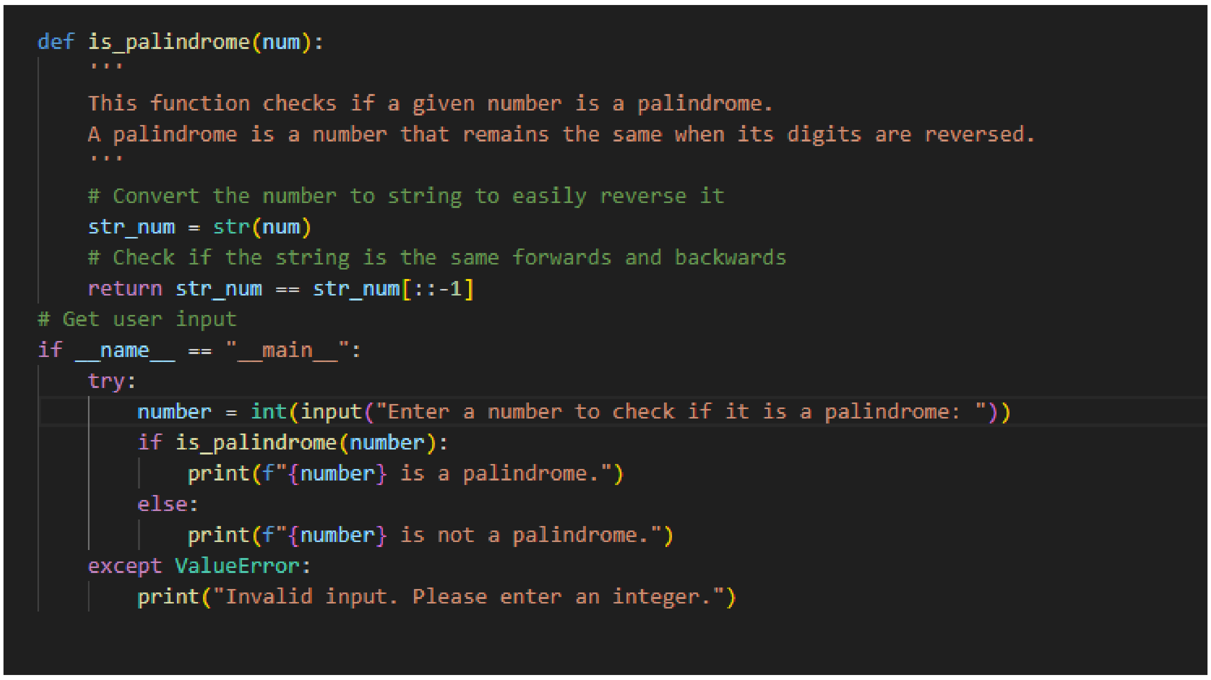
• Identify any logical errors or missing edge-case handling.

Prompt:



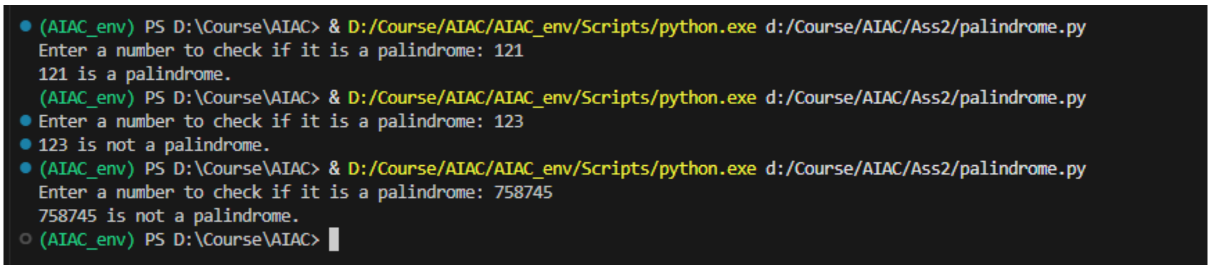
```
#write a python program to check wheather the given number is plaindrome or not
```

Code:



```python
def is_palindrome(num):
    '''
    This function checks if a given number is a palindrome.
    A palindrome is a number that remains the same when its digits are reversed.
    '''
    # Convert the number to string to easily reverse it
    str_num = str(num)
    # Check if the string is the same forwards and backwards
    return str_num == str_num[::-1]
# Get user input
if __name__ == "__main__":
    try:
        number = int(input("Enter a number to check if it is a palindrome: "))
        if is_palindrome(number):
            print(f"{number} is a palindrome.")
        else:
            print(f"{number} is not a palindrome.")
    except ValueError:
        print("Invalid input. Please enter an integer.")
```

Output:


```
● (AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/palindrome.py
  Enter a number to check if it is a palindrome: 121
  121 is a palindrome.
  (AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/palindrome.py
● Enter a number to check if it is a palindrome: 123
● 123 is not a palindrome.
● (AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/palindrome.py
  Enter a number to check if it is a palindrome: 758745
  758745 is not a palindrome.
○ (AIAC_env) PS D:\Course\AIAC> ▌
```

Explanation:

The code is handled every thing but edge case like negative and non integers are not handled.

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:

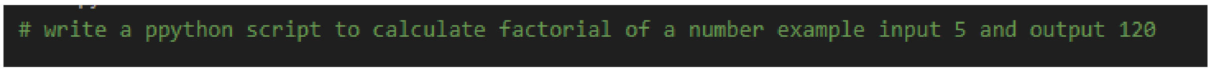
Input: 5 → Output: 120

Task:

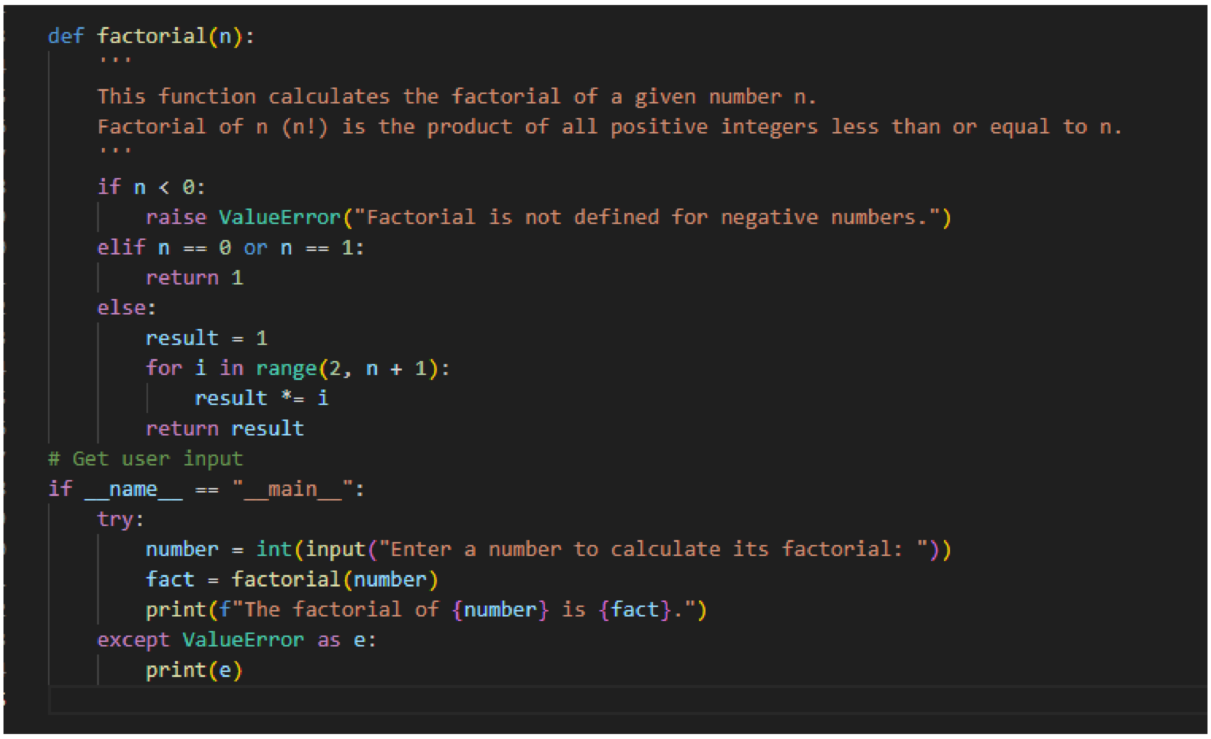
• Compare the generated code with a zero-shot solution.

• Examine improvements in clarity and correctness.

Prompt:


```
# write a ppython script to calculate factorial of a number example input 5 and output 120
```

Code:



```python
def factorial(n):
    '''
    This function calculates the factorial of a given number n.
    Factorial of n (n!) is the product of all positive integers less than or equal to n.
    '''
    if n < 0:
        raise ValueError("Factorial is not defined for negative numbers.")
    elif n == 0 or n == 1:
        return 1
    else:
        result = 1
        for i in range(2, n + 1):
            result *= i
        return result
# Get user input
if __name__ == "__main__":
    try:
        number = int(input("Enter a number to calculate its factorial: "))
        fact = factorial(number)
        print(f"The factorial of {number} is {fact}.")
    except ValueError as e:
        print(e)
```

Explanation:

Aspect	**Zero-Shot**	**One-Shot**
Prompt detail	Minimal	Includes example
Code clarity	Basic	More structured
Edge-case handling	Often missing	Usually included
Accuracy	Moderate	Higher
Reliability	Depends on model	Improved due to guidance

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

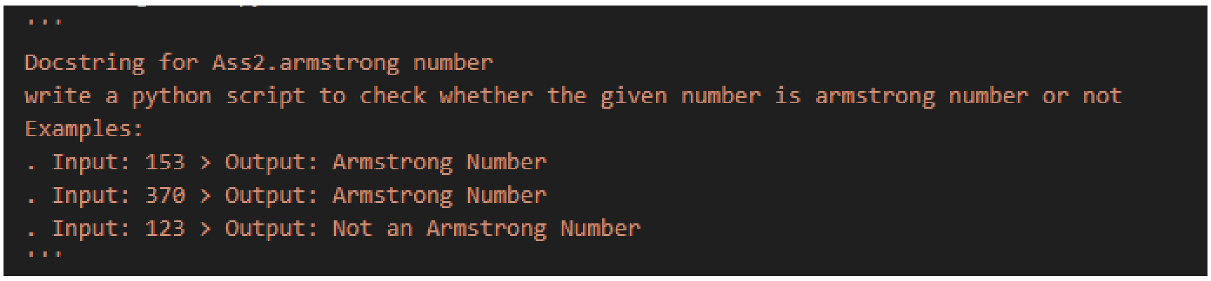
Task:

. Analyze how multiple examples influence code structure and

accuracy.

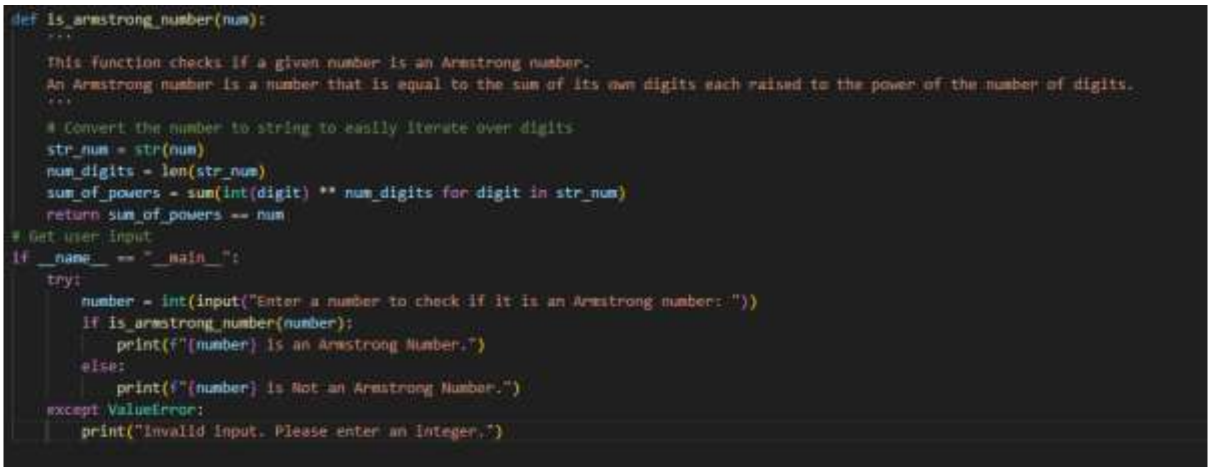
. Test the function with boundary values and invalid inputs.

Prompt:



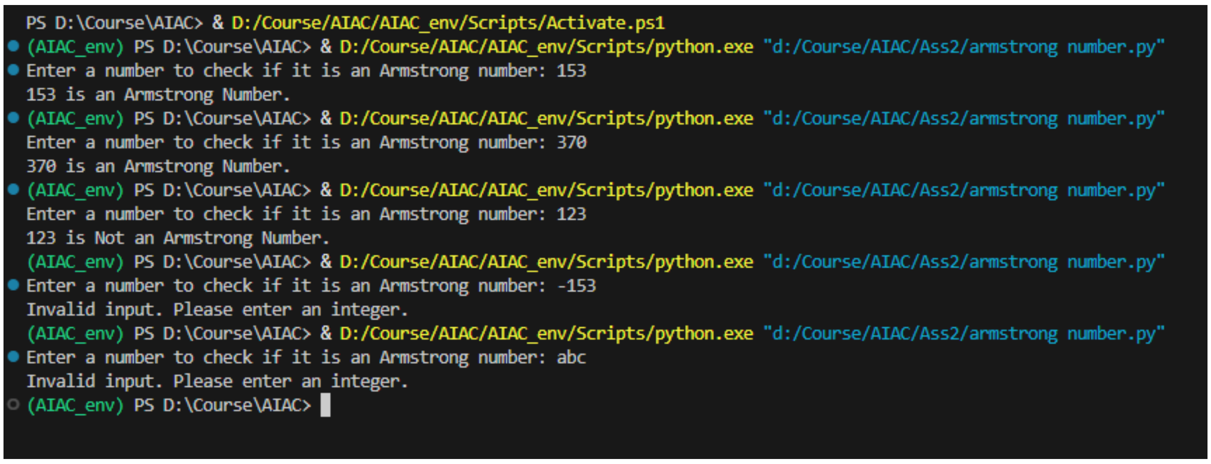
```
'''
Docstring for Ass2.armstrong number
write a python script to check whether the given number is armstrong number or not
Examples:
. Input: 153 > Output: Armstrong Number
. Input: 370 > Output: Armstrong Number
. Input: 123 > Output: Not an Armstrong Number
'''
```

Code:



```python
def is_armstrong_number(num):
    '''
    This function checks if a given number is an Armstrong number.
    An Armstrong number is a number that is equal to the sum of its own digits each raised to the power of the number of digits.
    '''
    # Convert the number to string to easily iterate over digits
    str_num = str(num)
    num_digits = len(str_num)
    sum_of_powers = sum(int(digit) ** num_digits for digit in str_num)
    return sum_of_powers == num
# Get user input
if __name__ == "__main__":
    try:
        number = int(input("Enter a number to check if it is an Armstrong number: "))
        if is_armstrong_number(number):
            print(f"{number} is an Armstrong Number.")
        else:
            print(f"{number} is Not an Armstrong Number.")
    except ValueError:
        print("Invalid input. Please enter an integer.")
```

Output:



```
PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/Activate.ps1
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe "d:/Course/AIAC/Ass2/armstrong number.py"
Enter a number to check if it is an Armstrong number: 153
153 is an Armstrong Number.
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe "d:/Course/AIAC/Ass2/armstrong number.py"
Enter a number to check if it is an Armstrong number: 370
370 is an Armstrong Number.
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe "d:/Course/AIAC/Ass2/armstrong number.py"
Enter a number to check if it is an Armstrong number: 123
123 is Not an Armstrong Number.
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe "d:/Course/AIAC/Ass2/armstrong number.py"
Enter a number to check if it is an Armstrong number: -153
Invalid input. Please enter an integer.
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe "d:/Course/AIAC/Ass2/armstrong number.py"
Enter a number to check if it is an Armstrong number: abc
Invalid input. Please enter an integer.
(AIAC_env) PS D:\Course\AIAC>
```

Explanation:

Testing boundary values:

-153 : not an integer

“abc”:not an integer

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.

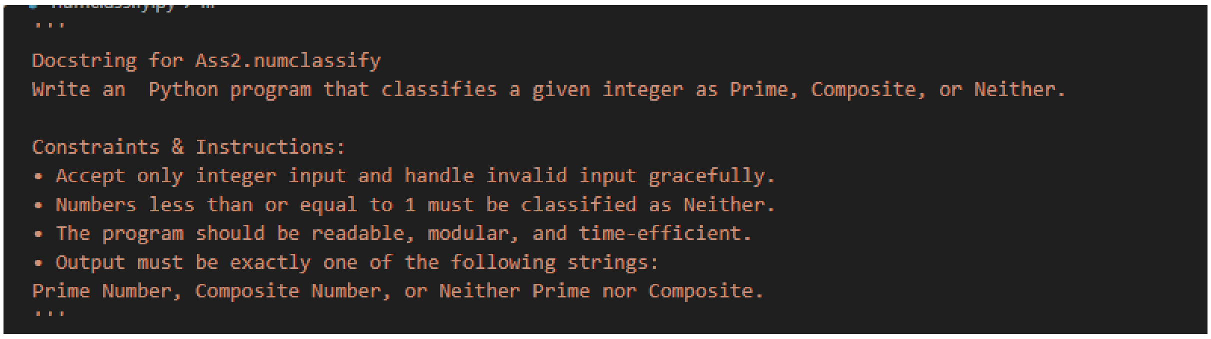
Task:

. Ensure proper input validation.

. Optimize the logic for efficiency.

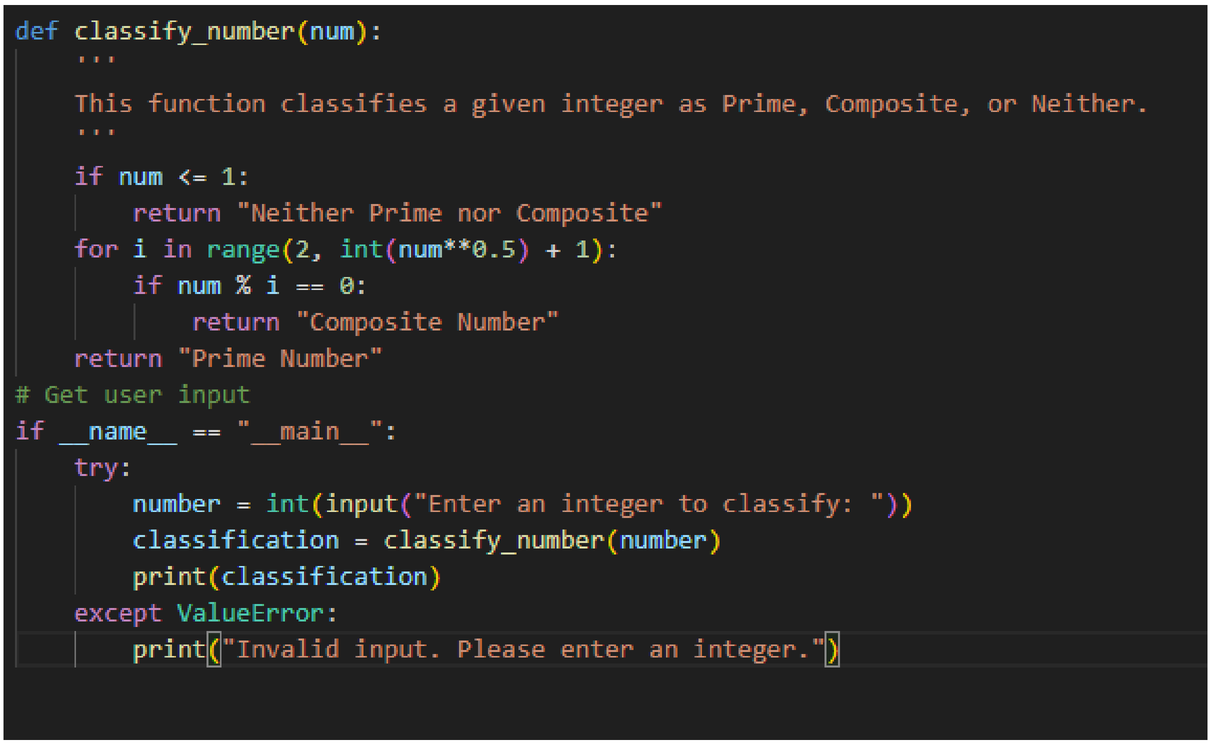
. Compare the output with earlier prompting strategies.

Prompt:



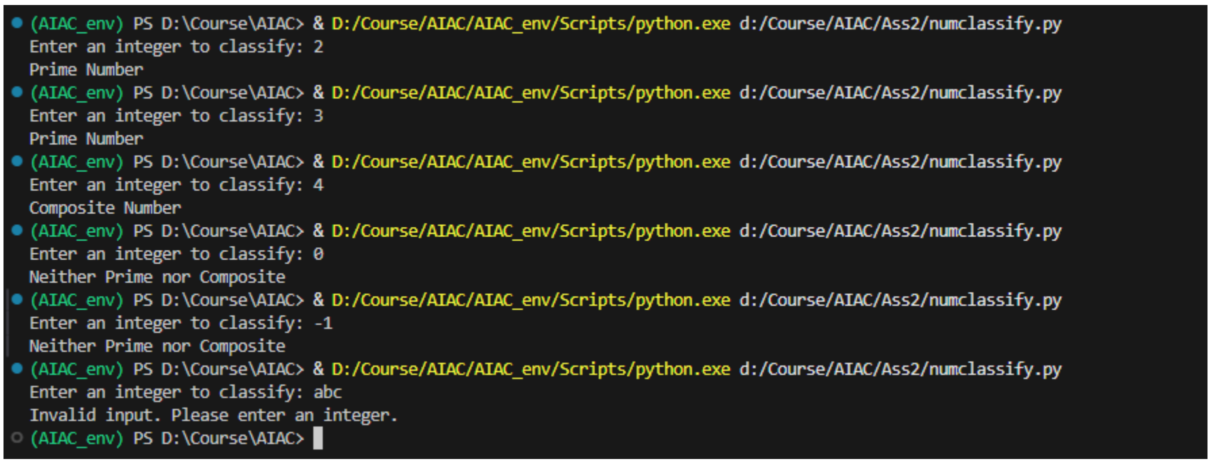
```
'''
Docstring for Ass2.numclassify
Write an  Python program that classifies a given integer as Prime, Composite, or Neither.

Constraints & Instructions:
• Accept only integer input and handle invalid input gracefully.
• Numbers less than or equal to 1 must be classified as Neither.
• The program should be readable, modular, and time-efficient.
• Output must be exactly one of the following strings:
Prime Number, Composite Number, or Neither Prime nor Composite.
'''
```

Code:



```python
def classify_number(num):
    '''
    This function classifies a given integer as Prime, Composite, or Neither.
    '''
    if num <= 1:
        return "Neither Prime nor Composite"
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            return "Composite Number"
    return "Prime Number"
# Get user input
if __name__ == "__main__":
    try:
        number = int(input("Enter an integer to classify: "))
        classification = classify_number(number)
        print(classification)
    except ValueError:
        print("Invalid input. Please enter an integer.")
```

Output:



```
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/numclassify.py
Enter an integer to classify: 2
Prime Number
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/numclassify.py
Enter an integer to classify: 3
Prime Number
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/numclassify.py
Enter an integer to classify: 4
Composite Number
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/numclassify.py
Enter an integer to classify: 0
Neither Prime nor Composite
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/numclassify.py
Enter an integer to classify: -1
Neither Prime nor Composite
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/numclassify.py
Enter an integer to classify: abc
Invalid input. Please enter an integer.
(AIAC_env) PS D:\Course\AIAC>
```

Explanation:

Zero-shot prompting is basic and simple code but it misses examples.
One-shot prompting improves logic and structure with partial validation and moderate performance.
Few-shot prompting gives clearer and more accurate code by learning from multiple examples, though efficiency is average.
Context-managed prompting delivers highly optimized, fully validated, and professionally structured 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.

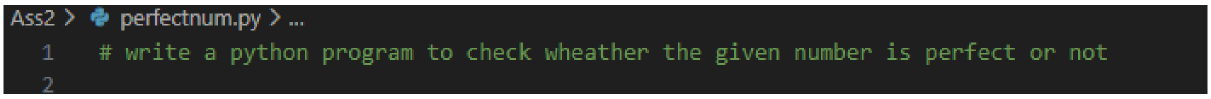
Task:

Record the AI-generated code.

Test the program with multiple inputs.

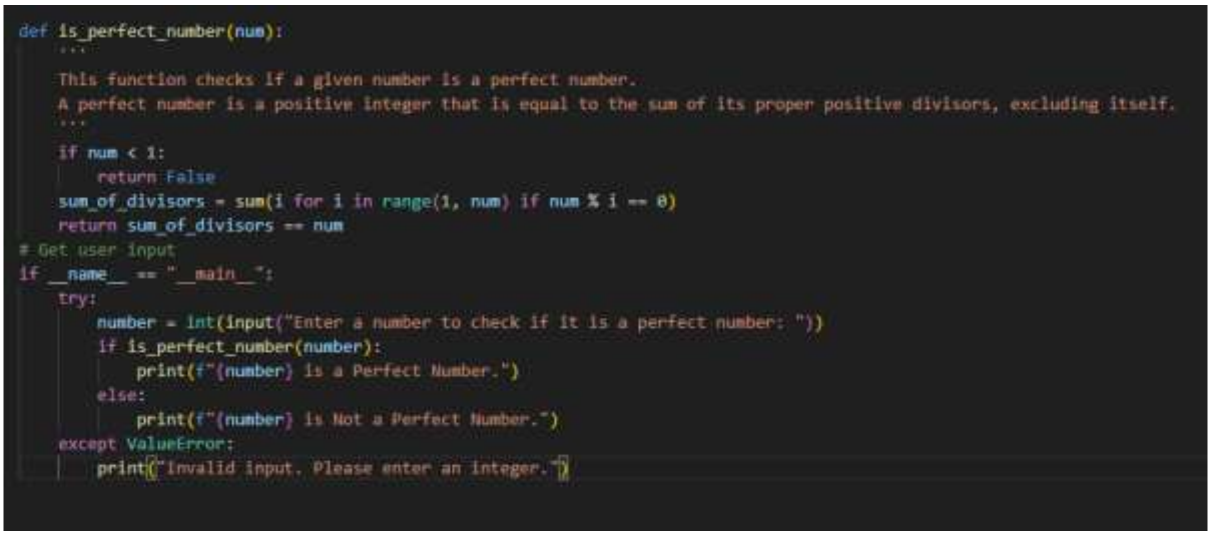
Identify any missing conditions or inefficiencies in the logic.

Prompt:



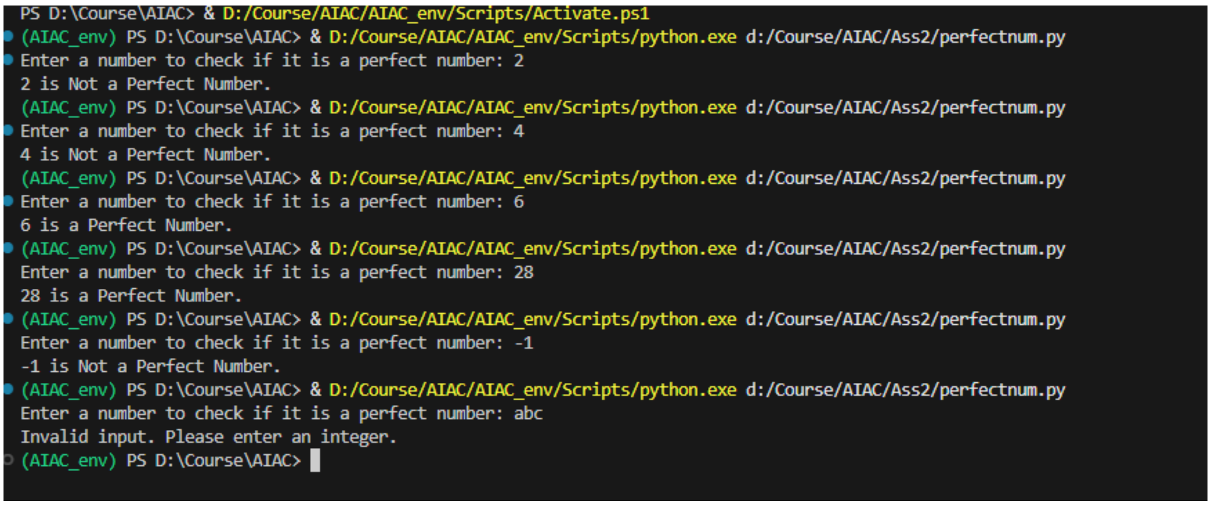
```
Ass2 > perfectnum.py > ...
# write a python program to check wheather the given number is perfect or not
```

Code:



```python
def is_perfect_number(num):
    '''
    This function checks if a given number is a perfect number.
    A perfect number is a positive integer that is equal to the sum of its proper positive divisors, excluding itself.
    '''
    if num < 1:
        return False
    sum_of_divisors = sum(i for i in range(1, num) if num % i == 0)
    return sum_of_divisors == num
# Get user input
if __name__ == "__main__":
    try:
        number = int(input("Enter a number to check if it is a perfect number: "))
        if is_perfect_number(number):
            print(f"{number} is a Perfect Number.")
        else:
            print(f"{number} is Not a Perfect Number.")
    except ValueError:
        print("Invalid input. Please enter an integer.")
```

Output:



```
PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/Activate.ps1
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/perfectnum.py
Enter a number to check if it is a perfect number: 2
2 is Not a Perfect Number.
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/perfectnum.py
Enter a number to check if it is a perfect number: 4
4 is Not a Perfect Number.
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/perfectnum.py
Enter a number to check if it is a perfect number: 6
6 is a Perfect Number.
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/perfectnum.py
Enter a number to check if it is a perfect number: 28
28 is a Perfect Number.
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/perfectnum.py
Enter a number to check if it is a perfect number: -1
-1 is Not a Perfect Number.
(AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/perfectnum.py
Enter a number to check if it is a perfect number: abc
Invalid input. Please enter an integer.
(AIAC_env) PS D:\Course\AIAC> 
```

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

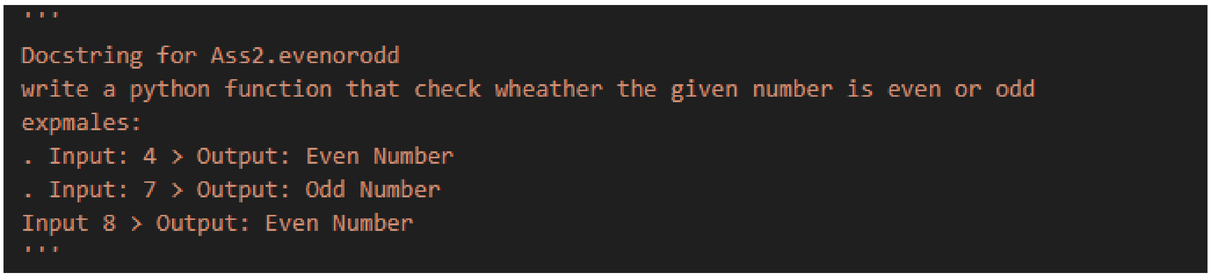
. Input: 0 > Output: Even

Task:

Analyze how examples improve input handling and output clarity.

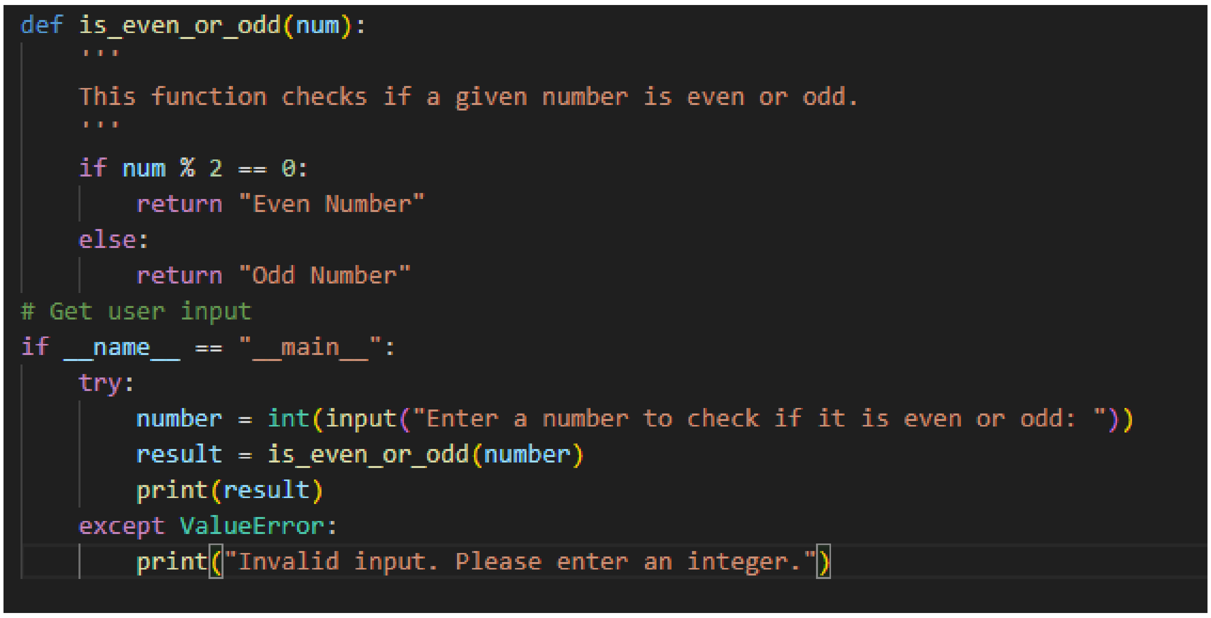
Test the program with negative numbers and non-integer inputs.

Prompt:



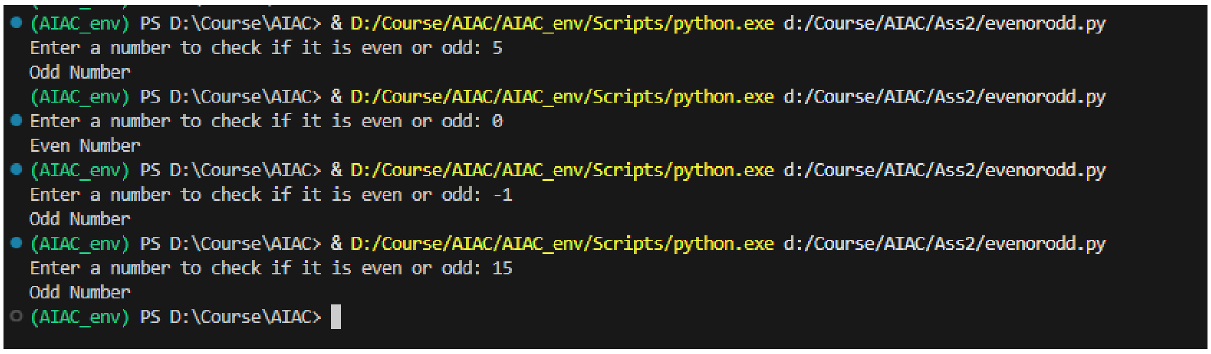
```
'''
Docstring for Ass2.evenorodd
write a python function that check wheather the given number is even or odd
expmales:
. Input: 4 > Output: Even Number
. Input: 7 > Output: Odd Number
Input 8 > Output: Even Number
'''
```

Code:



```python
def is_even_or_odd(num):
    '''
    This function checks if a given number is even or odd.
    '''
    if num % 2 == 0:
        return "Even Number"
    else:
        return "Odd Number"
# Get user input
if __name__ == "__main__":
    try:
        number = int(input("Enter a number to check if it is even or odd: "))
        result = is_even_or_odd(number)
        print(result)
    except ValueError:
        print("Invalid input. Please enter an integer.")
```

Output:



```
● (AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/evenorodd.py
  Enter a number to check if it is even or odd: 5
  Odd Number
  (AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/evenorodd.py
● Enter a number to check if it is even or odd: 0
  Even Number
● (AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/evenorodd.py
  Enter a number to check if it is even or odd: -1
  Odd Number
● (AIAC_env) PS D:\Course\AIAC> & D:/Course/AIAC/AIAC_env/Scripts/python.exe d:/Course/AIAC/Ass2/evenorodd.py
  Enter a number to check if it is even or odd: 15
  Odd Number
○ (AIAC_env) PS D:\Course\AIAC> 
```