AI ASSISTED CODING ASSIGNMENT -02

NAME **:** M.Shiva

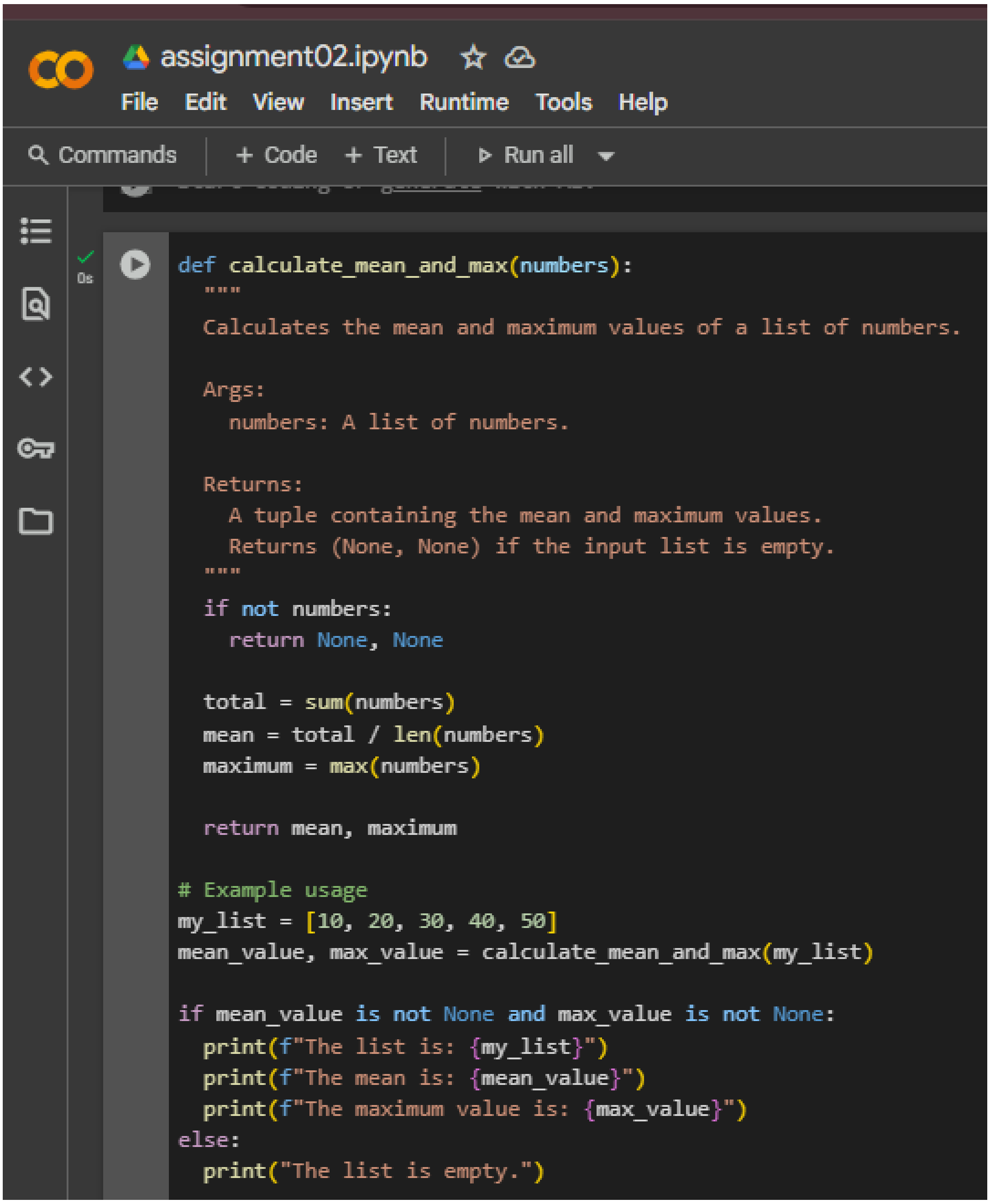
HALL.NO **:** 2403A52377

BATCH.NO **:** AI 14

PROMPT 01 :

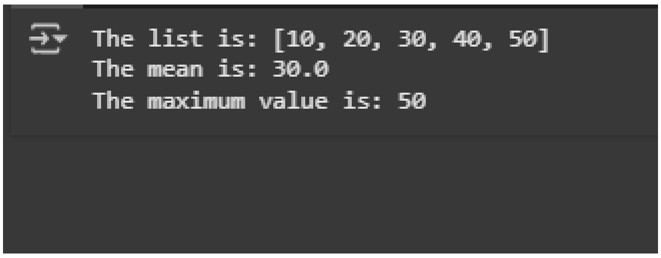
I need a code fuction that reads a list of numbers and calculates the mean and maximum values

CODE :



```python
def calculate_mean_and_max(numbers):
  """
  Calculates the mean and maximum values of a list of numbers.

  Args:
    numbers: A list of numbers.

  Returns:
    A tuple containing the mean and maximum values.
    Returns (None, None) if the input list is empty.
  """
  if not numbers:
    return None, None

  total = sum(numbers)
  mean = total / len(numbers)
  maximum = max(numbers)

  return mean, maximum

# Example usage
my_list = [10, 20, 30, 40, 50]
mean_value, max_value = calculate_mean_and_max(my_list)

if mean_value is not None and max_value is not None:
  print(f"The list is: {my_list}")
  print(f"The mean is: {mean_value}")
  print(f"The maximum value is: {max_value}")
else:
  print("The list is empty.")
```

OUTPUT:



```
The list is: [10, 20, 30, 40, 50]
The mean is: 30.0
The maximum value is: 50
```

CODE EXPLANATION:

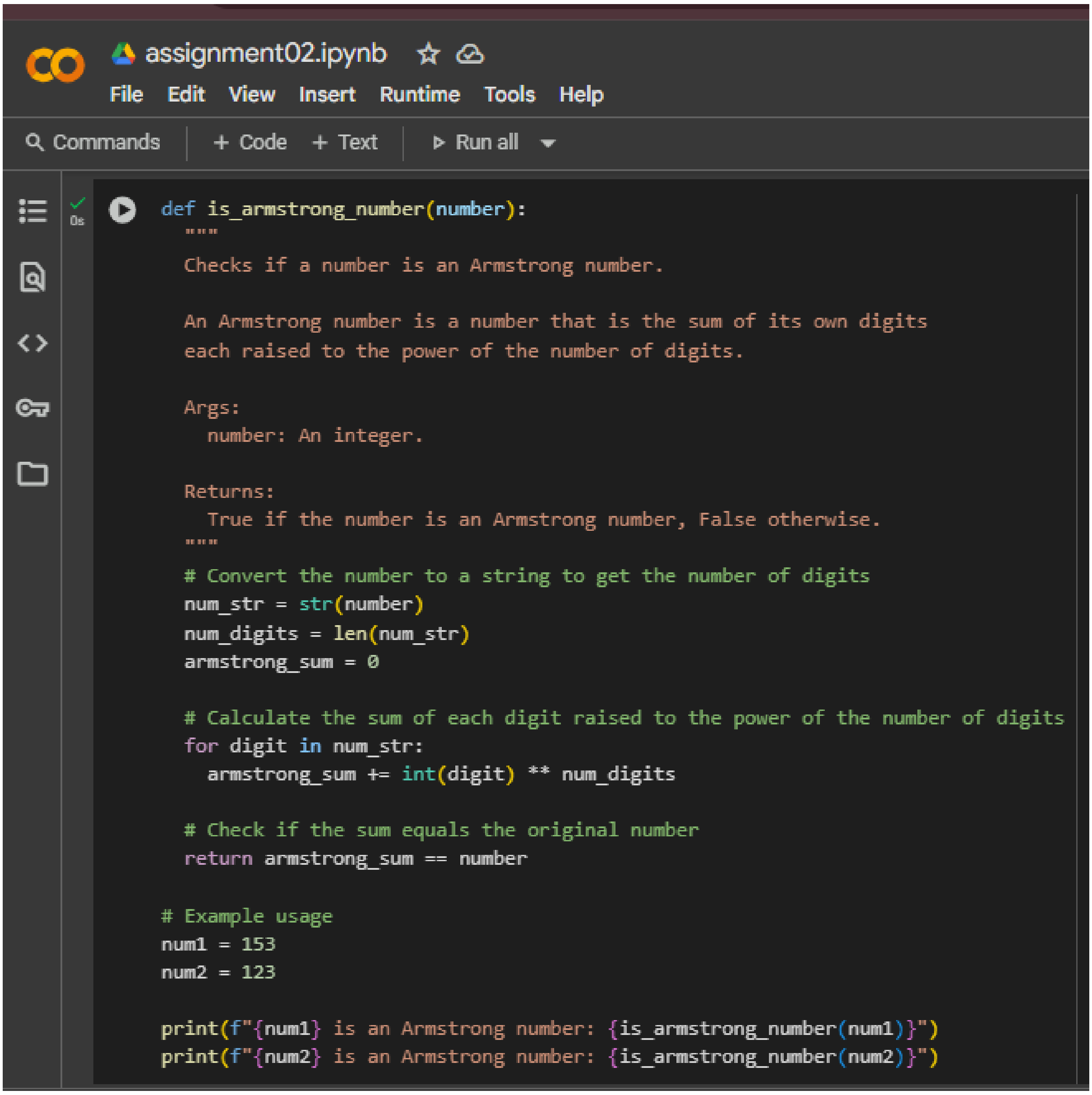
Function: calculate_mean_and_max(numbers)

1. Purpose: Calculates the *mean (average)* and *maximum* values from a list of numbers.
2. Empty list check: If the list is empty, returns (None, None).
3. Total calculation: Uses sum(numbers) to get the total of all elements.
4. Mean calculation: Divides the total by the number of elements (len(numbers)).
5. Max calculation: Finds the largest number in the list using max(numbers).
6. Return values: Returns both the mean and maximum as a tuple (mean, maximum).
7. Creates a sample list: my_list = [10,. Calls the function: mean_value, max_value = calculate_mean_and_max(my_list)`.
8. Checks if results are not None (list is not empty).
9. If values exist, prints:
    - The original list.
    - The mean value.
    - The maximum value.
10. If the list is empty, prints "The list is empty.".

PROMPT 02 :

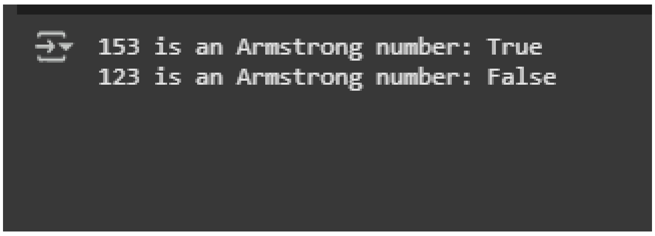
I need python code ,function that checks wether a number is an amstrong number

CODE :



```python
def is_armstrong_number(number):
  """
  Checks if a number is an Armstrong number.

  An Armstrong number is a number that is the sum of its own digits
  each raised to the power of the number of digits.

  Args:
    number: An integer.

  Returns:
    True if the number is an Armstrong number, False otherwise.
  """
  # Convert the number to a string to get the number of digits
  num_str = str(number)
  num_digits = len(num_str)
  armstrong_sum = 0

  # Calculate the sum of each digit raised to the power of the number of digits
  for digit in num_str:
    armstrong_sum += int(digit) ** num_digits

  # Check if the sum equals the original number
  return armstrong_sum == number

# Example usage
num1 = 153
num2 = 123

print(f"{num1} is an Armstrong number: {is_armstrong_number(num1)}")
print(f"{num2} is an Armstrong number: {is_armstrong_number(num2)}")
```

OUTPUT :



```
153 is an Armstrong number: True
123 is an Armstrong number: False
```

CODE EXPLANATION :

Function: is_armstrong_number(number)

1. Purpose: Checks if a number is an *Armstrong number*.
2. Convert to string: num_str = str(number) to handle digits easily.

3. Get digit count: num_digits = len(num_str) stores total digits.
4. Initialize sum: armstrong_sum = 0.
5. Loop through digits:
    - Convert each to int.
    - Raise to num_digits power.
    - Add to armstrong_sum.
6. Compare with original number:
   return armstrong_sum == number → True if Armstrong, else False.

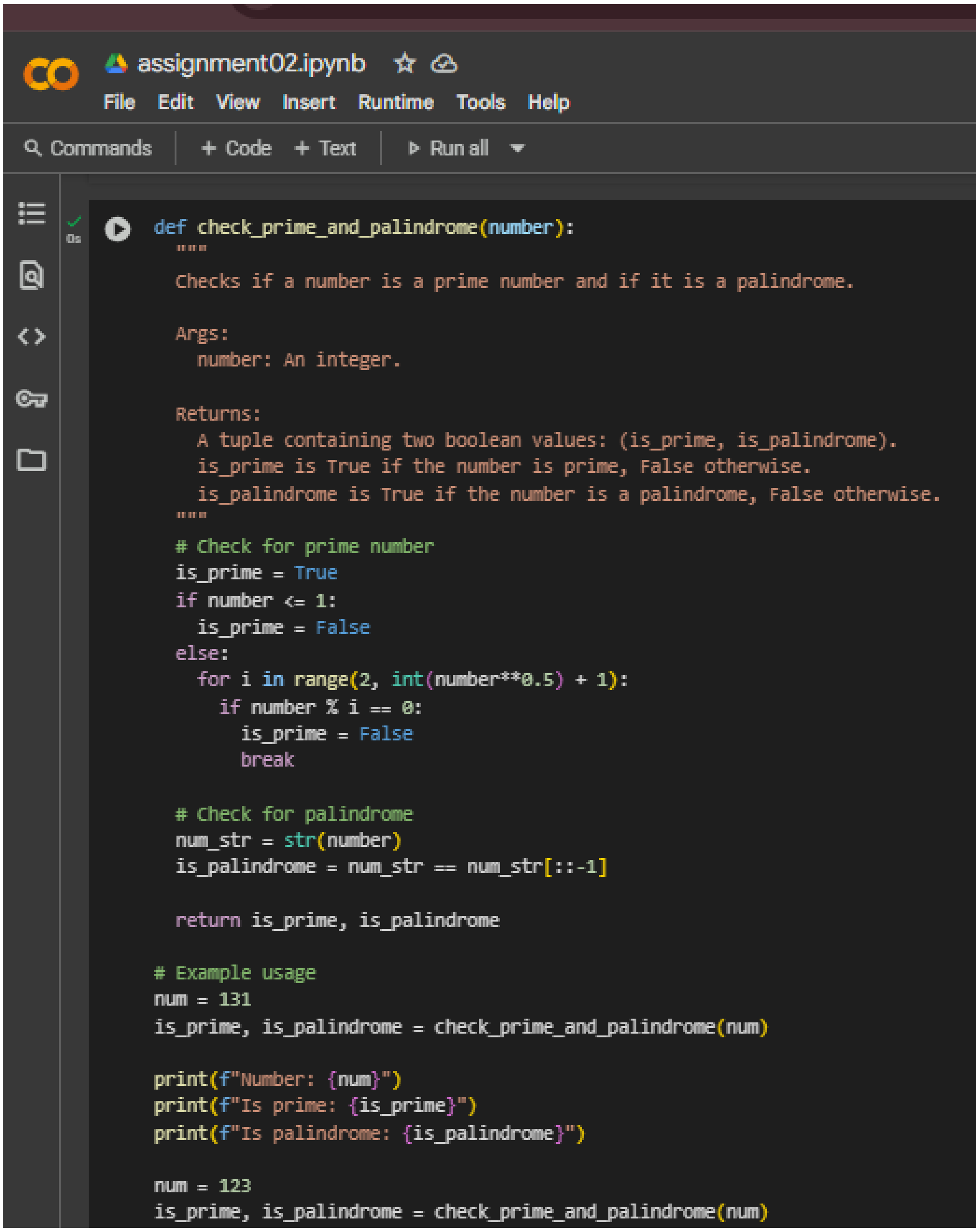
Example usage

7. Checks if 153 and 123 are Armstrong numbers using the function.
8. Prints results in format:
   "XXX is an Armstrong number: True/False".

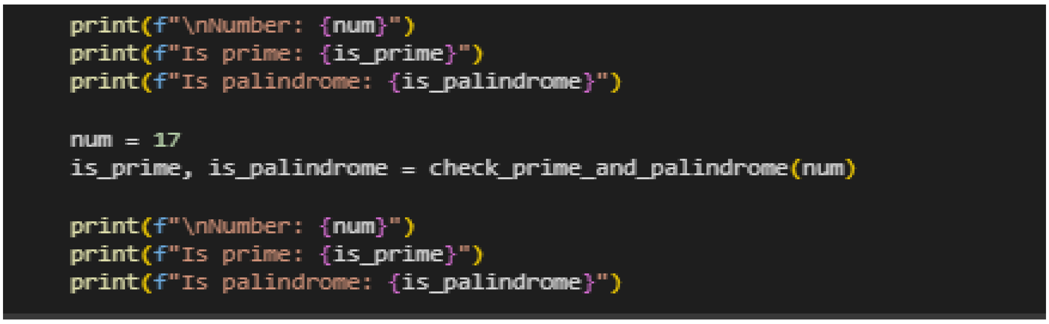
PROMPT 03 :

I need a python code , function check the number is prime number or a palindrome

CODE :

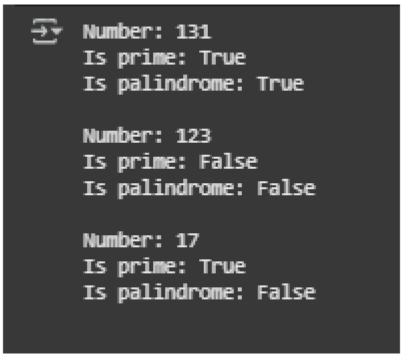


```
def check_prime_and_palindrome(number):
  """
  Checks if a number is a prime number and if it is a palindrome.

  Args:
    number: An integer.

  Returns:
    A tuple containing two boolean values: (is_prime, is_palindrome).
    is_prime is True if the number is prime, False otherwise.
    is_palindrome is True if the number is a palindrome, False otherwise.
  """
  # Check for prime number
  is_prime = True
  if number <= 1:
    is_prime = False
  else:
    for i in range(2, int(number**0.5) + 1):
      if number % i == 0:
        is_prime = False
        break

  # Check for palindrome
  num_str = str(number)
  is_palindrome = num_str == num_str[::-1]

  return is_prime, is_palindrome

# Example usage
num = 131
is_prime, is_palindrome = check_prime_and_palindrome(num)

print(f"Number: {num}")
print(f"Is prime: {is_prime}")
print(f"Is palindrome: {is_palindrome}")

num = 123
is_prime, is_palindrome = check_prime_and_palindrome(num)
```



```
print(f"\nNumber: {num}")
print(f"Is prime: {is_prime}")
print(f"Is palindrome: {is_palindrome}")

num = 17
is_prime, is_palindrome = check_prime_and_palindrome(num)

print(f"\nNumber: {num}")
print(f"Is prime: {is_prime}")
print(f"Is palindrome: {is_palindrome}")
```

OUTPUT :



```
Number: 131
Is prime: True
Is palindrome: True

Number: 123
Is prime: False
Is palindrome: False

Number: 17
Is prime: True
Is palindrome: False
```

CODE EXPLANATION :

Function: check_prime_and_palindrome(number)

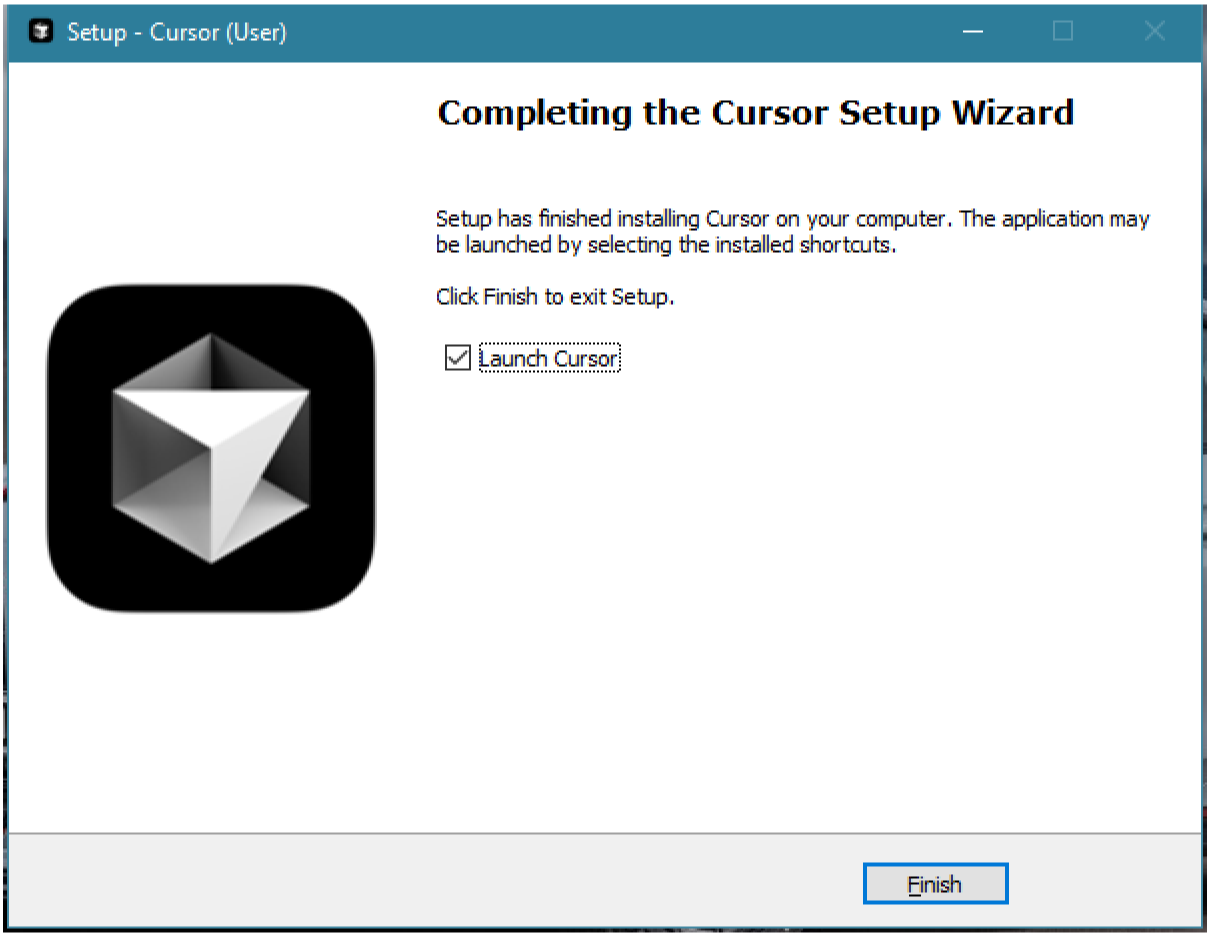
1. Purpose: Checks if a number is prime and if it is a palindrome.
2. Prime check:
    - If number ≤ 1 → Not prime.
    - Else → Check divisibility from 2 to √number.
    - If divisible → Not prime, else prime.
3. Palindrome check:
    - Convert number to string: num_str = str(number).
    - Compare with reversed string: num_str[::-1].
    - If equal → palindrome.
4. Return values:
    - (is_prime, is_palindrome) as a tuple of booleans.

Example usage

5. Test with numbers 131, 123, and 17.
6. Print:
    - The number.
    - Whether it's prime.
    - Whether it's a palindrome.

PROMPT 04 :

i need a pyhton code , function sum of the first N natural numbers and test the output



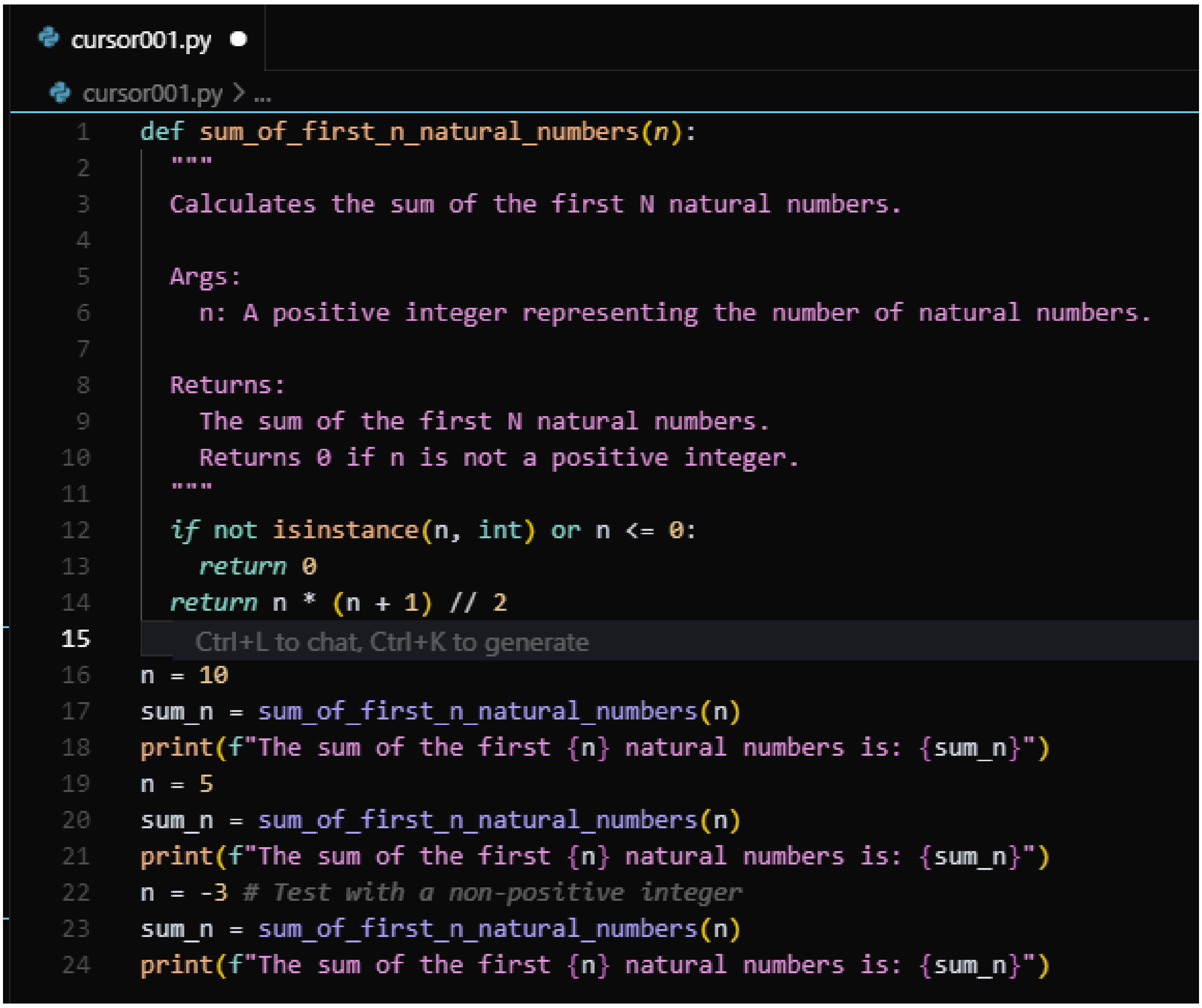
Completing the Cursor Setup Wizard

Setup has finished installing Cursor on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

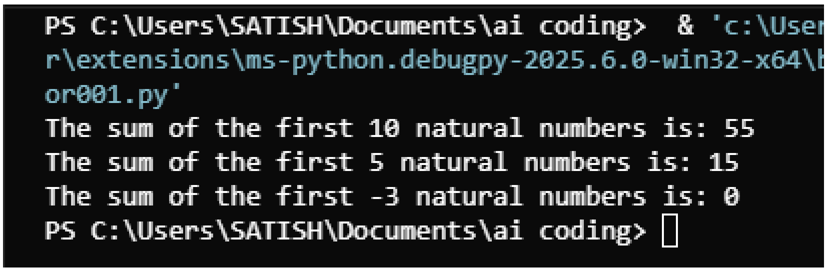
☑ Launch Cursor

CODE :



```python
def sum_of_first_n_natural_numbers(n):
  """
  Calculates the sum of the first N natural numbers.

  Args:
    n: A positive integer representing the number of natural numbers.

  Returns:
    The sum of the first N natural numbers.
    Returns 0 if n is not a positive integer.
  """
  if not isinstance(n, int) or n <= 0:
    return 0
  return n * (n + 1) // 2

n = 10
sum_n = sum_of_first_n_natural_numbers(n)
print(f"The sum of the first {n} natural numbers is: {sum_n}")
n = 5
sum_n = sum_of_first_n_natural_numbers(n)
print(f"The sum of the first {n} natural numbers is: {sum_n}")
n = -3 # Test with a non-positive integer
sum_n = sum_of_first_n_natural_numbers(n)
print(f"The sum of the first {n} natural numbers is: {sum_n}")
```

OUTPUT :



```
PS C:\Users\SATISH\Documents\ai coding>  & 'c:\User
r\extensions\ms-python.debugpy-2025.6.0-win32-x64\
or001.py'
The sum of the first 10 natural numbers is: 55
The sum of the first 5 natural numbers is: 15
The sum of the first -3 natural numbers is: 0
PS C:\Users\SATISH\Documents\ai coding> 
```

CODE EXPLANATION :

Function Purpose:
Calculates the sum of the first n natural numbers using the mathematical formula:

Sum=n(n+1)2\text{Sum} = \frac{n(n+1)}{2}Sum=2n(n+1)

Input Validation:
Checks if the input n is a positive integer.

- If not, returns 0.

Sum Calculation:
Uses integer division (//) to compute the sum efficiently without loops.

Function Calls:

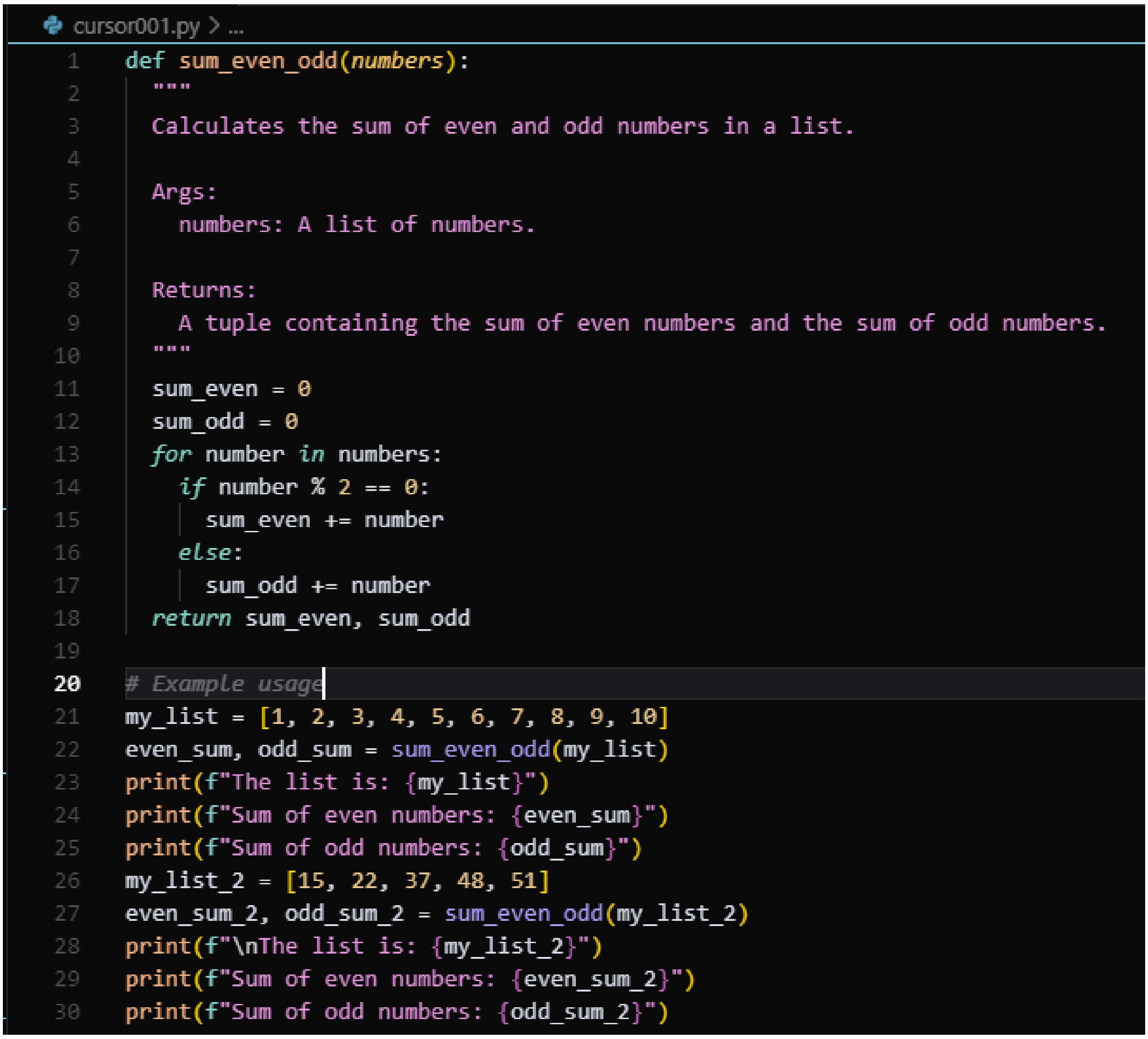
- n = 10 → Prints the sum of first 10 natural numbers.
- n = 5 → Prints the sum of first 5 natural numbers.
- n = -3 → Invalid input, so it prints 0.

Output:
Demonstrates how the function behaves for valid and invalid inputs.

PROMPT 05 :

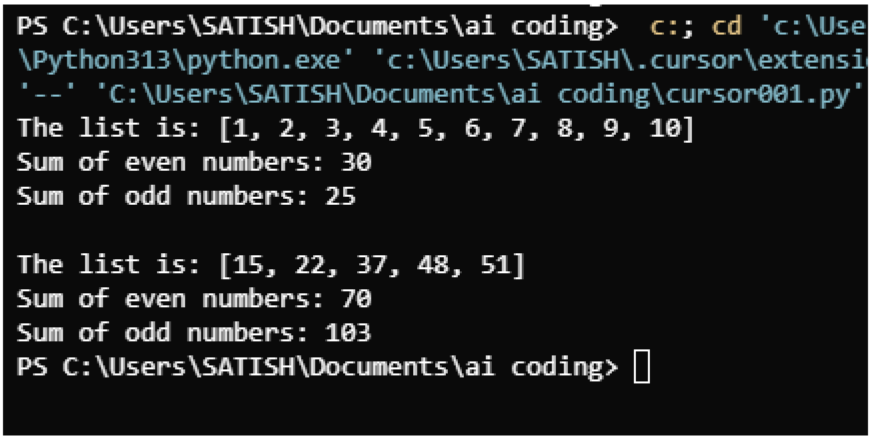
i need a pyhton code , sum of even numbers and odd numbes

CODE :



```python
def sum_even_odd(numbers):
  """
  Calculates the sum of even and odd numbers in a list.

  Args:
    numbers: A list of numbers.

  Returns:
    A tuple containing the sum of even numbers and the sum of odd numbers.
  """
  sum_even = 0
  sum_odd = 0
  for number in numbers:
    if number % 2 == 0:
      sum_even += number
    else:
      sum_odd += number
  return sum_even, sum_odd

# Example usage
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_sum, odd_sum = sum_even_odd(my_list)
print(f"The list is: {my_list}")
print(f"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")
my_list_2 = [15, 22, 37, 48, 51]
even_sum_2, odd_sum_2 = sum_even_odd(my_list_2)
print(f"\nThe list is: {my_list_2}")
print(f"Sum of even numbers: {even_sum_2}")
print(f"Sum of odd numbers: {odd_sum_2}")
```

OUTPUT :



```
PS C:\Users\SATISH\Documents\ai coding>  c:; cd 'c:\Use
\Python313\python.exe' 'c:\Users\SATISH\.cursor\extensi
'--' 'C:\Users\SATISH\Documents\ai coding\cursor001.py'
The list is: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
Sum of even numbers: 30
Sum of odd numbers: 25

The list is: [15, 22, 37, 48, 51]
Sum of even numbers: 70
Sum of odd numbers: 103
PS C:\Users\SATISH\Documents\ai coding> 
```

CODE EXPLANATION :

Function Purpose:
Calculates and returns the sum of even and odd numbers from a given list.

Parameters:

- numbers: A list of integers.

Initialization:

- sum_even and sum_odd are both initialized to 0.

Iteration and Condition:

- For each number in the list:
  - If the number is even (number % 2 == 0), it's added to sum_even.
  - If the number is odd, it's added to sum_odd.

Return Value:
Returns a tuple: (sum_even, sum_odd).

Example Usage:

- For my_list = [1, 2, 3, ..., 10]:
  - Even sum = 2 + 4 + 6 + 8 + 10 = 30
  - Odd sum = 1 + 3 + 5 + 7 + 9 = 25
- For my_list_2 = [15, 22, 37, 48, 51]:
  - Even sum = 22 + 48 = 70
  - Odd sum = 15 + 37 + 51 = 103

Output:
Displays the original list and the respective even and odd sums.