ASSIGNMENT

Task Description 1 Basic Docstring Generation
•Write python function to return sum of even and odd numbers in the given list.
•Incorporate manual docstring in code with Google Style
•Use an AI-assisted tool (e.g., Copilot, Cursor AI) to generate a docstring describing the function.
•Compare the AI-generated docstring with your manually written one

*** EXPECTED CODE AND OUTPUT GIVEN BY GOOGLE COLAB :

Code with inline comments :

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

  Args:
    numbers: A list of integers.

  Returns:
    A tuple containing the sum of even numbers and the sum of odd numbers.
  """
  even_sum = 0  # Initialize sum for even numbers
  odd_sum = 0   # Initialize sum for odd numbers

  for number in numbers:  # Iterate through each number in the list
    if number % 2 == 0:  # Check if the number is even
      even_sum += number # Add to even sum if even
    else:
      odd_sum += number  # Add to odd sum if odd

  return even_sum, odd_sum # Return the sums

# Example usage:
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_sum, odd_sum = sum_even_odd_manual_docstring(my_list)
print(f"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")
```

```
Sum of even numbers: 30
Sum of odd numbers: 25
```

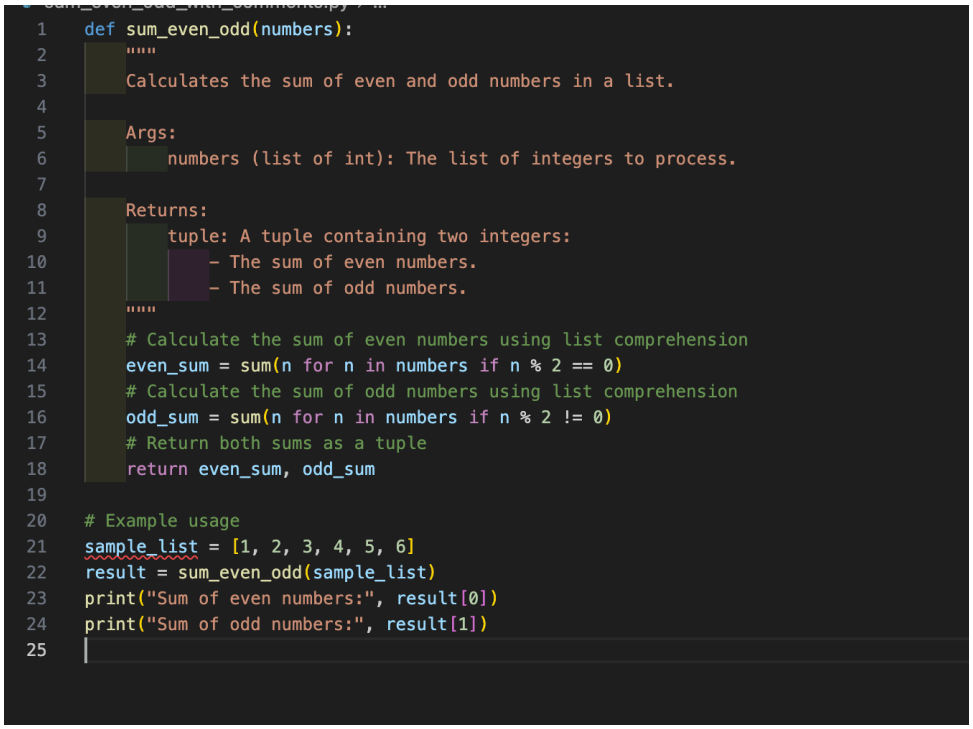
Code and output without comments :

```
def sum_even_odd_ai_docstring(numbers):
  even_sum = 0
  odd_sum = 0

  for number in numbers:
    if number % 2 == 0:
      even_sum += number
    else:
      odd_sum += number

  return even_sum, odd_sum

# Example usage:
my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_sum, odd_sum = sum_even_odd_ai_docstring(my_list)
print(f"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")
```

```
Sum of even numbers: 30
Sum of odd numbers: 25
```

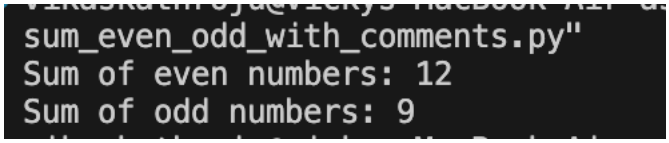
*** EXPECTED CODE AND OUTPUT GIVEN BY GITHUB COPILOT :

Code with inline comments :



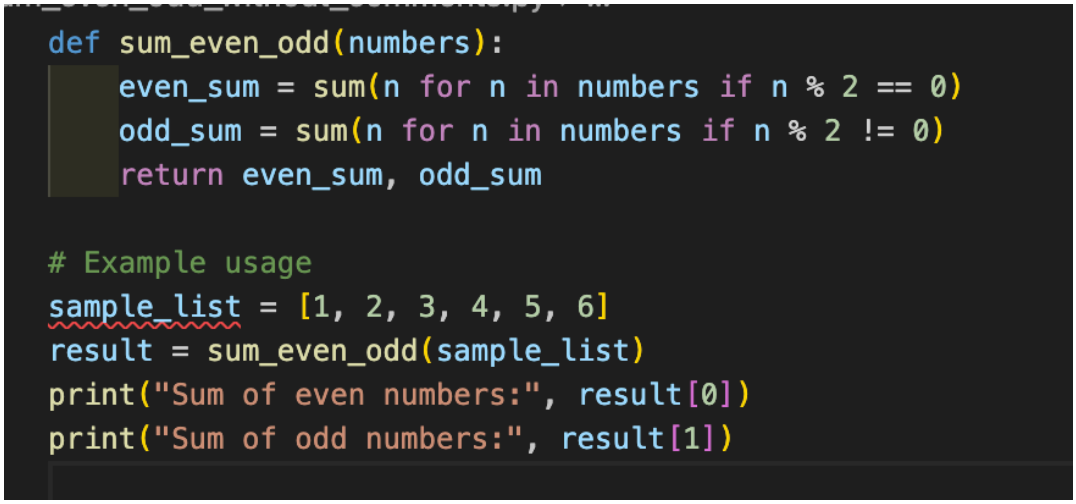
```python
def sum_even_odd(numbers):
    """
    Calculates the sum of even and odd numbers in a list.

    Args:
        numbers (list of int): The list of integers to process.

    Returns:
        tuple: A tuple containing two integers:
            - The sum of even numbers.
            - The sum of odd numbers.
    """
    # Calculate the sum of even numbers using list comprehension
    even_sum = sum(n for n in numbers if n % 2 == 0)
    # Calculate the sum of odd numbers using list comprehension
    odd_sum = sum(n for n in numbers if n % 2 != 0)
    # Return both sums as a tuple
    return even_sum, odd_sum

# Example usage
sample_list = [1, 2, 3, 4, 5, 6]
result = sum_even_odd(sample_list)
print("Sum of even numbers:", result[0])
print("Sum of odd numbers:", result[1])

```

Output :



```
sum_even_odd_with_comments.py"
Sum of even numbers: 12
Sum of odd numbers: 9
```

Code without comments :



```python
def sum_even_odd(numbers):
    even_sum = sum(n for n in numbers if n % 2 == 0)
    odd_sum = sum(n for n in numbers if n % 2 != 0)
    return even_sum, odd_sum

# Example usage
sample_list = [1, 2, 3, 4, 5, 6]
result = sum_even_odd(sample_list)
print("Sum of even numbers:", result[0])
print("Sum of odd numbers:", result[1])
```

Output :

```
sum_even_odd_without_comments.py"
Sum of even numbers: 12
Sum of odd numbers: 9
```

Task Description 2 Automatic Inline Comments

•Write python program for sru_student class with attributes like name, roll no.,hostel_status and fee_update method and display_details method.
•Write comments manually for each line/code block
•Ask an AI tool to add inline comments explaining each line/step.
•Compare the AI-generated comments with your manually written one

*** EXPECTED CODE AND OUTPUT GIVEN BY GOOGLE COLAB :

Code and output with inline comments :

```python
class sru_student_manual_comments: # Define a class named sru_student_manual_comments
    def __init__(self, name, roll_no, hostel_status): # Constructor to initialize object attributes
        self.name = name # Assign the provided name to the object's name attribute
        self.roll_no = roll_no # Assign the provided roll_no to the object's roll_no attribute
        self.hostel_status = hostel_status # Assign the provided hostel_status to the object's hostel_status attribute
        self.fee_paid = 0 # Initialize fee_paid attribute to 0

    def fee_update(self, amount): # Method to update the fee paid
        self.fee_paid += amount # Add the provided amount to the fee_paid attribute
        print(f"Fee updated for {self.name}. Total fee paid: {self.fee_paid}") # Print a confirmation message

    def display_details(self): # Method to display student details
        print("Student Details:") # Print a header for student details
        print(f"Name: {self.name}") # Print the student's name
        print(f"Roll No.: {self.roll_no}") # Print the student's roll number
        print(f"Hostel Status: {self.hostel_status}") # Print the student's hostel status
        print(f"Fee Paid: {self.fee_paid}") # Print the total fee paid

# Example Usage:
student1 = sru_student_manual_comments("Alice", "SRU123", "Resident") # Create an instance of the class
student1.display_details() # Display the initial details of student1
student1.fee_update(5000) # Update the fee for student1
student1.fee_update(10000) # Update the fee again for student1
student1.display_details() # Display the updated details of student1
```

```
Student Details:
Name: Alice
Roll No.: SRU123
Hostel Status: Resident
Fee Paid: 0
Fee updated for Alice. Total fee paid: 5000
Fee updated for Alice. Total fee paid: 15000
Student Details:
Name: Alice
Roll No.: SRU123
Hostel Status: Resident
Fee Paid: 15000
```

Code and output without comments :

```python
class sru_student_ai_comments:
    def __init__(self, name, roll_no, hostel_status):
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee_paid = 0

    def fee_update(self, amount):
        self.fee_paid += amount
        print(f"Fee updated for {self.name}. Total fee paid: {self.fee_paid}")

    def display_details(self):
        print("Student Details:")
        print(f"Name: {self.name}")
        print(f"Roll No.: {self.roll_no}")
        print(f"Hostel Status: {self.hostel_status}")
        print(f"Fee Paid: {self.fee_paid}")

# Example Usage:
student2 = sru_student_ai_comments("Bob", "SRU456", "Day Scholar")
student2.display_details()
student2.fee_update(3000)
student2.fee_update(7000)
student2.display_details()
```

```
Student Details:
Name: Bob
Roll No.: SRU456
Hostel Status: Day Scholar
Fee Paid: 0
Fee updated for Bob. Total fee paid: 3000
Fee updated for Bob. Total fee paid: 10000
Student Details:
Name: Bob
Roll No.: SRU456
Hostel Status: Day Scholar
Fee Paid: 10000
```

*** EXPECTED CODE AND OUTPUT GIVEN BY GITHUB COPILOT :

Code with inline comments :

```python
class sru_student:
    # Initialize the student object with name, roll number, and hostel status
    def __init__(self, name, roll_no, hostel_status):
        self.name = name  # Store the student's name
        self.roll_no = roll_no  # Store the student's roll number
        self.hostel_status = hostel_status  # Store the student's hostel status
        self.fee_paid = False  # Track if the fee has been updated/paid

    # Method to update the fee status
    def fee_update(self, status):
        self.fee_paid = status  # Update the fee_paid attribute

    # Method to display all details of the student
    def display_details(self):
        print(f"Name: {self.name}")  # Print the student's name
        print(f"Roll No: {self.roll_no}")  # Print the student's roll number
        print(f"Hostel Status: {self.hostel_status}")  # Print the student's hostel status
        print(f"Fee Paid: {'Yes' if self.fee_paid else 'No'}")  # Print fee status

# Example usage
student1 = sru_student("Rahul", 101, "Yes")  # Create a student object
student1.fee_update(True)  # Update fee status to paid
student1.display_details()  # Display all details
```

Output :

```
Name: Rahul
Roll No: 101
Hostel Status: Yes
Fee Paid: Yes
```

Code without comments :

```python
class sru_student:
    def __init__(self, name, roll_no, hostel_status):
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee_paid = False

    def fee_update(self, status):
        self.fee_paid = status

    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Roll No: {self.roll_no}")
        print(f"Hostel Status: {self.hostel_status}")
        print(f"Fee Paid: {'Yes' if self.fee_paid else 'No'}")

student1 = sru_student("Rahul", 101, "Yes")
student1.fee_update(True)
student1.display_details()
```

Output :

```
vikaskathroju@vickys-Mac
t:2"
Name: Rahul
Roll No: 101
Hostel Status: Yes
Fee Paid: Yes
```

Task Description 3
•Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).
•Incorporate manual docstring in code with NumPy Style
- Use AI assistance to generate a module-level docstring + individual function docstrings.
- Compare the AI-generated docstring with your manually written one.

Expected Output :

Students learn structured documentation for multi-function scripts

***EXPECTED CODE AND OUTPUT PROVIDED BY GOOGLE COLAB :

Code with inline comments :

```python
# This module provides basic calculator functions.

def add(x, y):
    """
    Adds two numbers.

    Parameters
    ----------
    x : float
        The first number.
    y : float
        The second number.

    Returns
    -------
    float
        The sum of x and y.
    """
    return x + y # Return the sum of x and y

def subtract(x, y):
    """
    Subtracts the second number from the first.

    Parameters
    ----------
    x : float
        The first number.
    y : float
        The second number.

    Returns
    -------
    float
```

```
        -------
        float
            The difference between x and y.
        """
        return x - y # Return the difference between x and y

    def multiply(x, y):
        """
        Multiplies two numbers.

        Parameters
        ----------
        x : float
            The first number.
        y : float
            The second number.

        Returns
        -------
        float
            The product of x and y.
        """
        return x * y # Return the product of x and y

    def divide(x, y):
        """
        Divides the first number by the second.

        Parameters
        ----------
        x : float
            The numerator.
        y : float
            The denominator.
```

```
        Returns
        -------
        float
            The quotient of x and y.

        Raises
        ------
        ValueError
            If the denominator is zero.
        """
        if y == 0: # Check if the denominator is zero
            raise ValueError("Cannot divide by zero!") # Raise an error if dividing by zero
        return x / y # Return the quotient of x and y

    # Example Usage:
    num1 = 10
    num2 = 5

    print(f"{num1} + {num2} = {add(num1, num2)}")
    print(f"{num1} - {num2} = {subtract(num1, num2)}")
    print(f"{num1} * {num2} = {multiply(num1, num2)}")
    print(f"{num1} / {num2} = {divide(num1, num2)}")

    # Example of division by zero
    try:
        print(f"{num1} / 0 = {divide(num1, 0)}")
    except ValueError as e:
        print(e)
```

```
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2.0
Cannot divide by zero!
```

Code without comments :

```python
def add(x, y):
  return x + y

def subtract(x, y):
  return x - y

def multiply(x, y):
  return x * y

def divide(x, y):
  if y == 0:
    raise ValueError("Cannot divide by zero!")
  return x / y

num1 = 10
num2 = 5

print(f"{num1} + {num2} = {add(num1, num2)}")
print(f"{num1} - {num2} = {subtract(num1, num2)}")
print(f"{num1} * {num2} = {multiply(num1, num2)}")
print(f"{num1} / {num2} = {divide(num1, num2)}")

try:
    print(f"{num1} / 0 = {divide(num1, 0)}")
except ValueError as e:
    print(e)
```

```
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2.0
Cannot divide by zero!
```

*** EXPECTED CODE AND OUTPUT PROVIDED BY GITHUB COPILOT :

Code with inline comments :

```python
"""
Calculator module for basic arithmetic operations.

This module provides functions to add, subtract, multiply, and divide two numbers.

"""

def add(a, b):
    """
    Add two numbers.

    Parameters
    ----------
    a : float or int
        First number.
    b : float or int
        Second number.

    Returns
    -------
    float or int
        The sum of a and b.
    """
    # Return the sum of a and b
    return a + b

def subtract(a, b):
    """
    Subtract one number from another.

    Parameters
    ----------
    a : float or int
```

```
    a : float or int
        First number.
    b : float or int
        Second number.

    Returns
    -------
    float or int
        The result of a minus b.
    """
    # Return the difference of a and b
    return a - b

def multiply(a, b):
    """
    Multiply two numbers.

    Parameters
    ----------
    a : float or int
        First number.
    b : float or int
        Second number.

    Returns
    -------
    float or int
        The product of a and b.
    """
```

```
    # Return the product of a and b
    return a * b

def divide(a, b):
    """
    Divide one number by another.

    Parameters
    ----------
    a : float or int
        Numerator.
    b : float or int
        Denominator.

    Returns
    -------
    float or int
        The result of a divided by b.

    Raises
    ------
    ValueError
        If b is zero.
    """
    # Check for division by zero
    if b == 0:
        raise ValueError("Cannot divide by zero.")
    # Return the result of a divided by b
    return a / b
```

```
# Example usage
x, y = 10, 5
print("Add:", add(x, y))
print("Subtract:", subtract(x, y))
print("Multiply:", multiply(x, y))
print("Divide:", divide(x, y))
```

Output :

```
vikaskathroju@vickys
Add: 15
Subtract: 5
Multiply: 50
Divide: 2.0
```

Code without comments :

```
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    """
    if b == 0:
        raise ValueError("Cannot divide by zero.")
    return a / b

x, y = 10, 5
print("Add:", add(x, y))
print("Subtract:", subtract(x, y))
print("Multiply:", multiply(x, y))
print("Divide:", divide(x, y))
```

Output :

```
Add: 15
Subtract: 5
Multiply: 50
Divide: 2.0
```