AI ASSISTED CODING

LAB-9

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.
- **GOOGLE COLAB**

```python
def sum_even_odd(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.
  """
  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

my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_sum, odd_sum = sum_even_odd(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
```

Inline comments:

```python
def sum_even_odd(numbers):
  sum_even = 0  # Initialize sum of even numbers to 0
  sum_odd = 0   # Initialize sum of odd numbers to 0
  for number in numbers: # Iterate through each number in the input list
    if number % 2 == 0: # Check if the number is even
      sum_even += number # If even, add to sum_even
    else:
      sum_odd += number # If odd, add to sum_odd
  return sum_even, sum_odd # Return the sum of even and odd numbers

my_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] # Example list of numbers
even_sum, odd_sum = sum_even_odd(my_list) # Call the function and store the results
print(f"Sum of even numbers: {even_sum}") # Print the sum of even numbers
print(f"Sum of odd numbers: {odd_sum}") # Print the sum of odd numbers
```

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

VS CODE

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

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

    Returns:
        tuple: A tuple containing (sum_of_even, sum_of_odd).
    """
    sum_even = sum(num for num in numbers if num % 2 == 0)
    sum_odd = sum(num for num in numbers if num % 2 != 0)
    return sum_even, sum_odd
```

INLINE COMMENTS

```python
def sum_even_odd(numbers):
    # This function returns the sum of even and odd numbers in the given list
    """
    Returns the sum of even and odd numbers in the given list.

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

    Returns:
        tuple: A tuple containing (sum_of_even, sum_of_odd).
    """
    # Calculate the sum of even numbers using a generator expression
    sum_even = sum(num for num in numbers if num % 2 == 0)
    # Calculate the sum of odd numbers using a generator expression
    sum_odd = sum(num for num in numbers if num % 2 != 0)
    # Return both sums as a tuple (sum_of_even, sum_of_odd)
    return sum_even, sum_odd
```

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.

```python
class SRU_STUDENT:
  """Represents a student at SRU."""

  def __init__(self, name, roll_no, hostel_status):
    """Initializes a new SRU_STUDENT object.

    Args:
      name: The name of the student.
      roll_no: The roll number of the student.
      hostel_status: The hostel status of the student (e.g., 'resident', 'day scholar').
    """
    self.NAME = name
    self.ROLL_NO = roll_no
    self.HOSTEL_STATUS = hostel_status
    self.FEE_STATUS = "Pending" # Initial fee status

  def FEE_UPDATE(self, status):
    """Updates the fee status of the student.

    Args:
      status: The new fee status (e.g., 'Paid', 'Pending').
    """
    self.FEE_STATUS = status
    print(f"Fee status for {self.NAME} (Roll No: {self.ROLL_NO}) updated to: {self.FEE_STATUS}")

  def display_details(self):
    """Displays the details of the student."""
    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 Status: {self.FEE_STATUS}")
    print("-----------------------")

# Example usage:
# student1 = SRU_STUDENT("Alice", "SRU123", "resident")
# student1.display_details()
# student1.FEE_UPDATE("Paid")
# student1.display_details()
```

INLINE COMMENTS

```python
class SRU_STUDENT:
  # Class docstring explaining the purpose of the class

  def __init__(self, name, roll_no, hostel_status):
    # Initializes a new SRU_STUDENT object.

    # Args:
    # name: The name of the student.
    # roll_no: The roll number of the student.
    # hostel_status: The hostel status of the student (e.g., 'resident', 'day scholar').

    self.NAME = name  # Assign the provided name to the instance's NAME attribute
    self.ROLL_NO = roll_no # Assign the provided roll_no to the instance's ROLL_NO attribute
    self.HOSTEL_STATUS = hostel_status # Assign the provided hostel_status to the instance's HOSTEL_STATUS attribute
    self.FEE_STATUS = "Pending" # Initialize the FEE_STATUS attribute to "Pending"

  def FEE_UPDATE(self, status):
    # Updates the fee status of the student.

    # Args:
    # status: The new fee status (e.g., 'Paid', 'Pending').

    self.FEE_STATUS = status # Update the FEE_STATUS attribute with the provided status
    print(f"Fee status for {self.NAME} (Roll No: {self.ROLL_NO}) updated to: {self.FEE_STATUS}") # Print a confirmation message

  def display_details(self):
    # Displays the details of the student.

    print("--- Student Details ---") # Print a header for the 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 Status: {self.FEE_STATUS}") # Print the student's fee status
    print("-----------------------") # Print a footer for the student details

# Example usage:
# student1 = SRU_STUDENT("Alice", "SRU123", "resident") # Create an instance of the SRU_STUDENT class
# student1.display_details() # Call the display_details method to show initial details
# student1.FEE_UPDATE("Paid") # Call the FEE_UPDATE method to change the fee status
# student1.display_details() # Call the display_details method again to show updated details
```

Vs code

ai.py > lab9.py > SRU_Student > display_details

```python
class SRU_Student:
    """
    SRU_Student class represents a student with the following attributes:
    - name: Name of the student
    - roll_no: Roll number of the student
    - hostel_status: Boolean indicating if the student is in hostel

    Methods:
    - fee_update(amount): Updates the student's fee status by the given amount.
    - display_details(): Displays the student's details.
    """

    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):
        """Updates the student's fee status by the given amount."""
        self.fee_paid += amount

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

INLINE COMMENTS

ai.py > lab9.py > SRU_Student > display_details

```python
class SRU_Student:
    # SRU_Student class represents a student with attributes and methods for fee management

    def __init__(self, name, roll_no, hostel_status):
        # Initialize the student's name, roll number, and hostel status
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        # Initialize fee_paid to 0
        self.fee_paid = 0

    def fee_update(self, amount):
        # Add the given amount to the student's fee_paid
        self.fee_paid += amount

    def display_details(self):
        # Print all details of the student
        print(f"Name: {self.name}")
        print(f"Roll No: {self.roll_no}")
        print(f"Hostel Status: {'Yes' if self.hostel_status else 'No'}")
        print(f"Fee Paid: {self.fee_paid}")
```

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.

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

  Args:
    x: The first number.
    y: The second number.

  Returns:
    The sum of x and y.
  """
  return x + y

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

  Args:
    x: The first number.
    y: The second number.

  Returns:
    The difference between x and y.
  """
  return x - y

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

  Args:
    x: The first number.
    y: The second number.

  Returns:
    The product of x and y.
  """
  return x * y

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

  Args:
    x: The first number (dividend).
    y: The second number (divisor).
```

```
    Divides the first number by the second.

    Args:
      x: The first number (dividend).
      y: The second number (divisor).

    Returns:
      The result of the division.

    Raises:
      ZeroDivisionError: If the divisor is zero.
    """
    if y == 0:
      raise ZeroDivisionError("Cannot divide by zero")
    return x / 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)}")
```

Inline comments

```python
def add(x, y):
  # Adds two numbers.

  # Args:
  #   x: The first number.
  #   y: The second number.

  # Returns:
  #   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.

  # Args:
  #   x: The first number.
  #   y: The second number.

  # Returns:
  #   The difference between x and y.
  return x - y # Return the difference between x and y

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

  # Args:
  #   x: The first number.
  #   y: The second number.

  # Returns:
  #   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.
```

```
    # Args:
    #   x: The first number (dividend).
    #   y: The second number (divisor).

    # Returns:
    #   The result of the division.

    # Raises:
    #   ZeroDivisionError: If the divisor is zero.
    if y == 0: # Check if the divisor is zero
      raise ZeroDivisionError("Cannot divide by zero") # Raise an error if the divisor is zero
    return x / y # Return the result of the division

# 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)}")
```

VS CODE :DOCSTRING

i.py > lab9.py > ...

```python
def add(a, b):
    """
    This module provides basic calculator functions: addition, subtraction, multiplication, and division.
    Functions:
        add(a, b): Returns the sum of a and b.
        subtract(a, b): Returns the difference of a and b.
        multiply(a, b): Returns the product of a and b.
        divide(a, b): Returns the quotient of a divided by b. Raises ValueError if b is zero.
    Example:
    """
    """Return the sum of a and b."""
    return a + b

def subtract(a, b):
    """Return the difference of a and b."""
    return a - b

def multiply(a, b):
    """Return the product of a and b."""
    return a * b

def divide(a, b):
    """Return the quotient of a divided by b. Raises ValueError if b is zero."""
    if b == 0:
        raise ValueError("Cannot divide by zero.")
    return a / b

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

INLINE COMMENTS

ai.py > lab9.py > ...

```python
def add(a, b):
    # Return the sum of a and b
    return a + b

def subtract(a, b):
    # Return the difference of a and b
    return a - b

def multiply(a, b):
    # Return the product of a and b
    return a * b

def divide(a, b):
    # Return the quotient of a divided by b
    # Raise ValueError if b is zero to avoid division by zero
    if b == 0:
        raise ValueError("Cannot divide by zero.")
    return a / b

# Example usage
if __name__ == "__main__":
    x, y = 10, 5
    # Print results of calculator functions
    print("Add:", add(x, y))
    print("Subtract:", subtract(x, y))
    print("Multiply:", multiply(x, y))
    print("Divide:", divide(x, y))
```