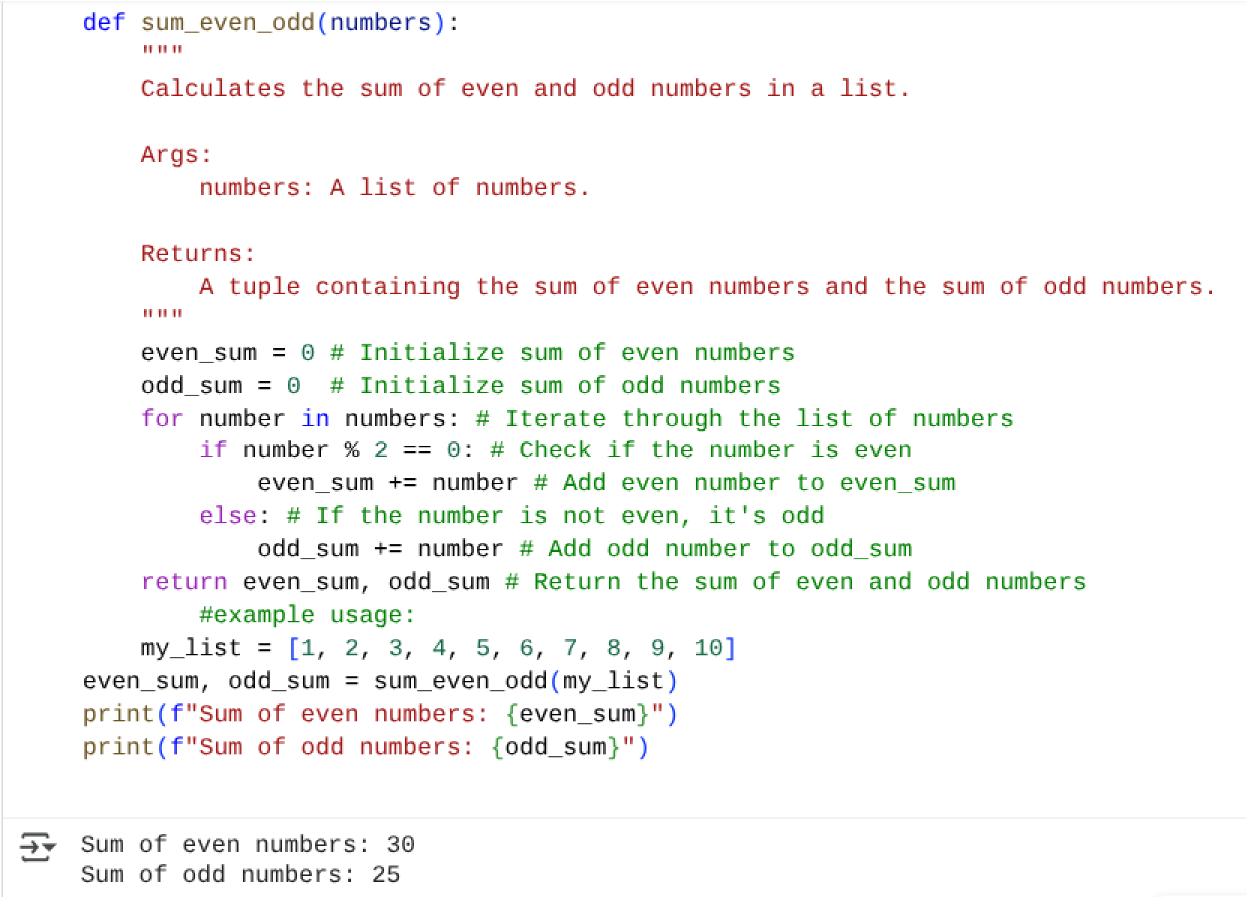
ASSIGNMENT-9.3

Task 1:

write the python function to calculate the sum of even and odd numbers in a given list without comments

Gemini:

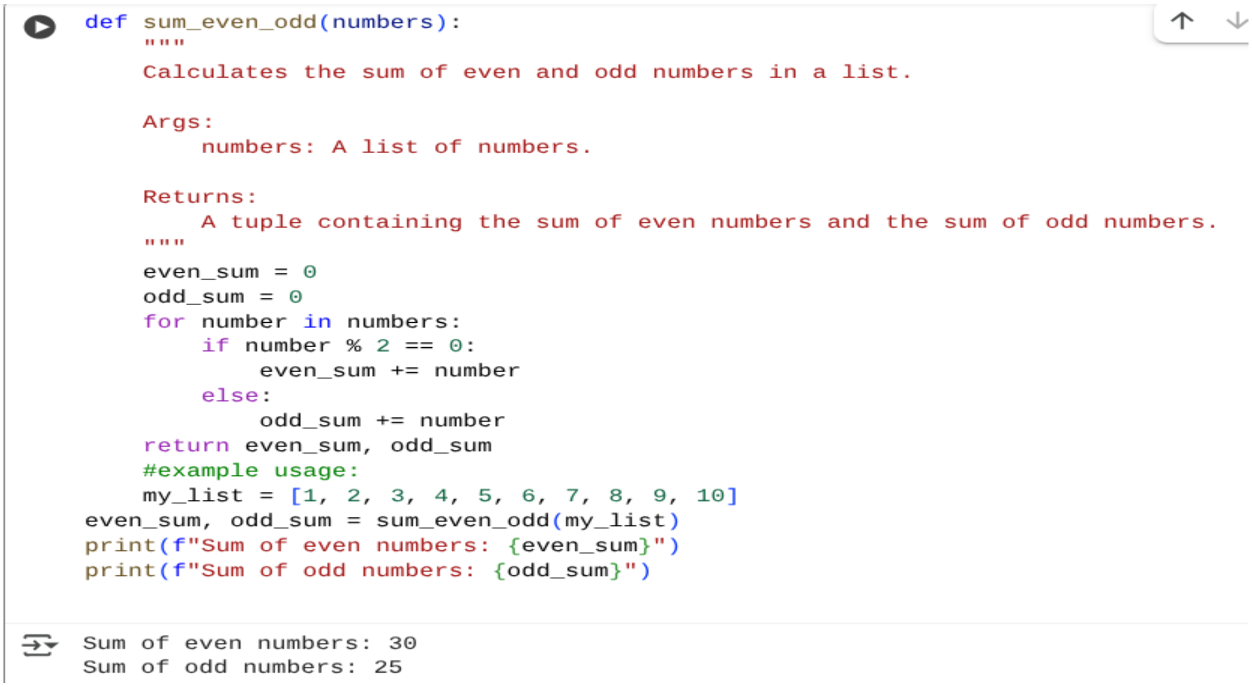
With comments:



```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.
    """
    even_sum = 0 # Initialize sum of even numbers
    odd_sum = 0  # Initialize sum of odd numbers
    for number in numbers: # Iterate through the list of numbers
        if number % 2 == 0: # Check if the number is even
            even_sum += number # Add even number to even_sum
        else: # If the number is not even, it's odd
            odd_sum += number # Add odd number to odd_sum
    return even_sum, odd_sum # Return the sum of even and odd numbers
        #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"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")
```

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

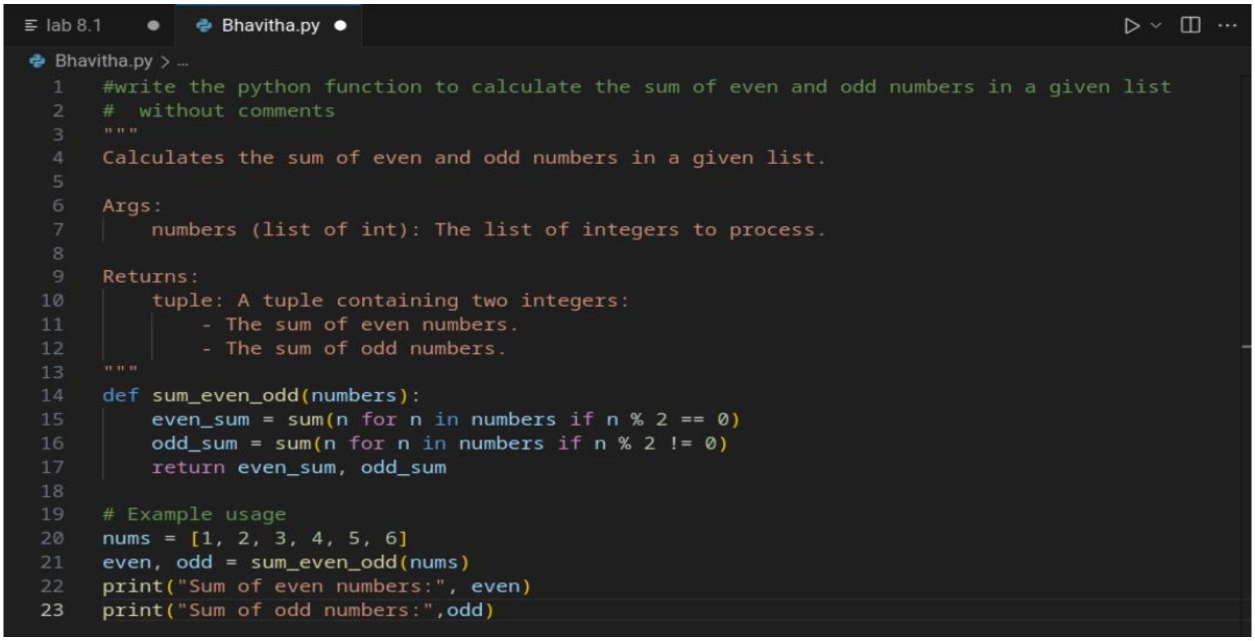
Without comments:



```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.
    """
    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(my_list)
print(f"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")
```

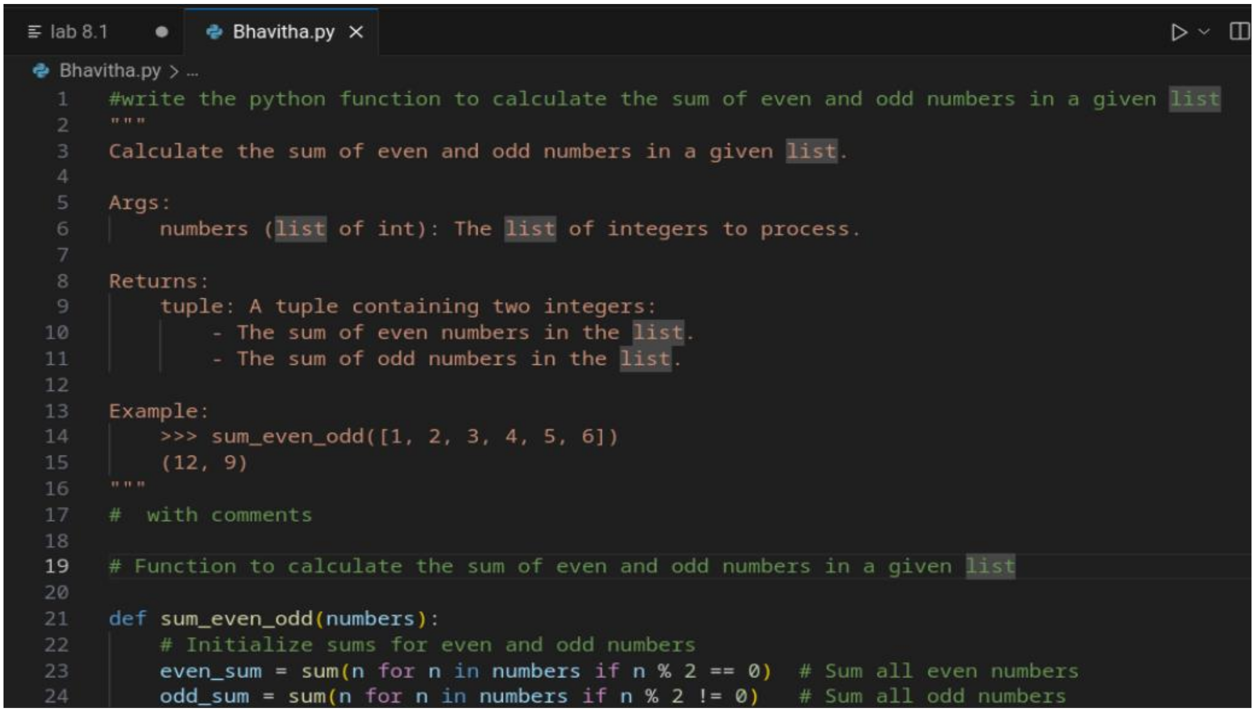
```
Sum of even numbers: 30
Sum of odd numbers: 25
```

Github Copilot: Without comments:

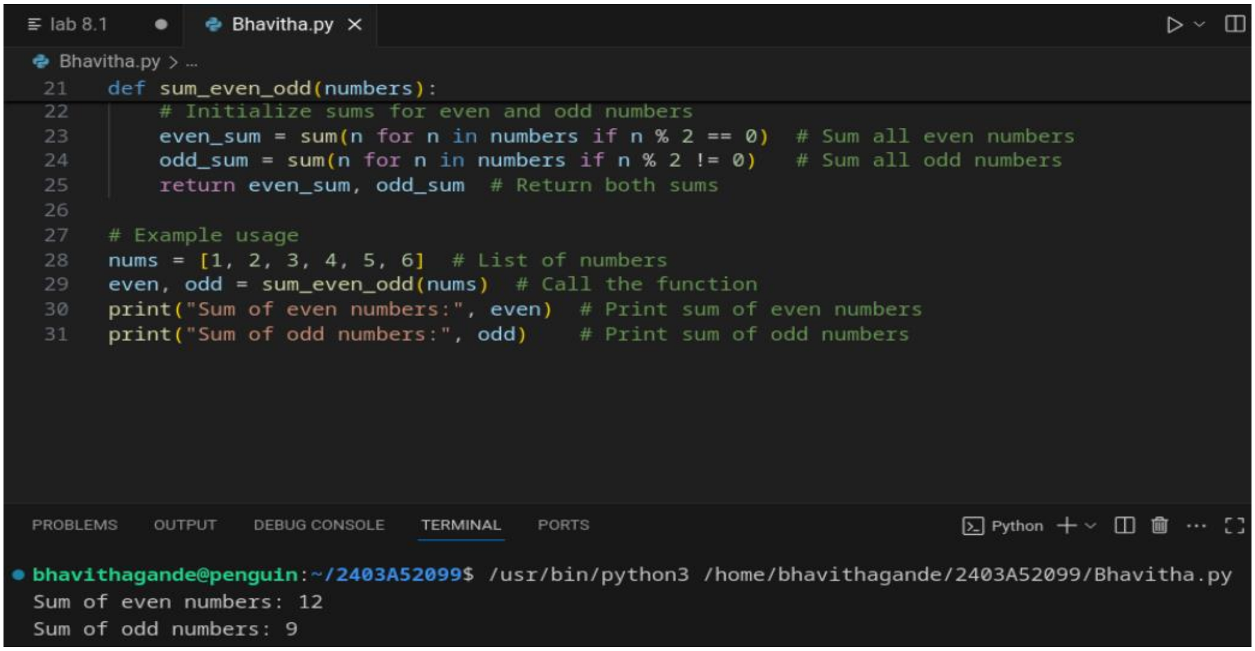


```python
#write the python function to calculate the sum of even and odd numbers in a given list
#  without comments
"""
Calculates the sum of even and odd numbers in a given list.

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.
"""
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
nums = [1, 2, 3, 4, 5, 6]
even, odd = sum_even_odd(nums)
print("Sum of even numbers:", even)
print("Sum of odd numbers:",odd)
```

With comments:



```python
#write the python function to calculate the sum of even and odd numbers in a given list
"""
Calculate the sum of even and odd numbers in a given list.

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

Returns:
    tuple: A tuple containing two integers:
        - The sum of even numbers in the list.
        - The sum of odd numbers in the list.

Example:
    >>> sum_even_odd([1, 2, 3, 4, 5, 6])
    (12, 9)
"""
#  with comments

# Function to calculate the sum of even and odd numbers in a given list

def sum_even_odd(numbers):
    # Initialize sums for even and odd numbers
    even_sum = sum(n for n in numbers if n % 2 == 0)  # Sum all even numbers
    odd_sum = sum(n for n in numbers if n % 2 != 0)   # Sum all odd numbers
```



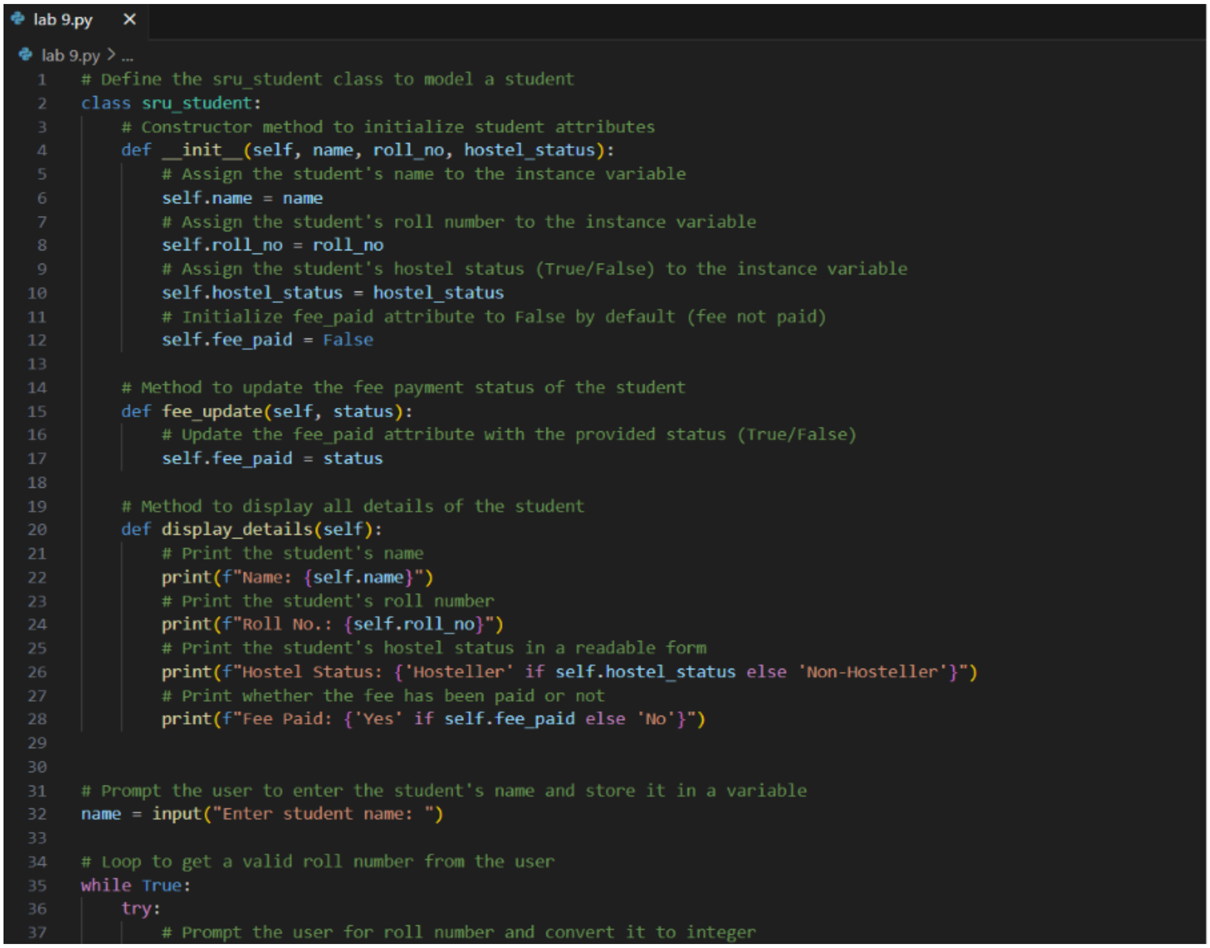
```python
def sum_even_odd(numbers):
    # Initialize sums for even and odd numbers
    even_sum = sum(n for n in numbers if n % 2 == 0)  # Sum all even numbers
    odd_sum = sum(n for n in numbers if n % 2 != 0)   # Sum all odd numbers
    return even_sum, odd_sum  # Return both sums

# Example usage
nums = [1, 2, 3, 4, 5, 6]  # List of numbers
even, odd = sum_even_odd(nums)  # Call the function
print("Sum of even numbers:", even)  # Print sum of even numbers
print("Sum of odd numbers:", odd)    # Print sum of odd numbers
```

```
bhavithagande@penguin:~/2403A52099$ /usr/bin/python3 /home/bhavithagande/2403A52099/Bhavitha.py
Sum of even numbers: 12
Sum of odd numbers: 9
```

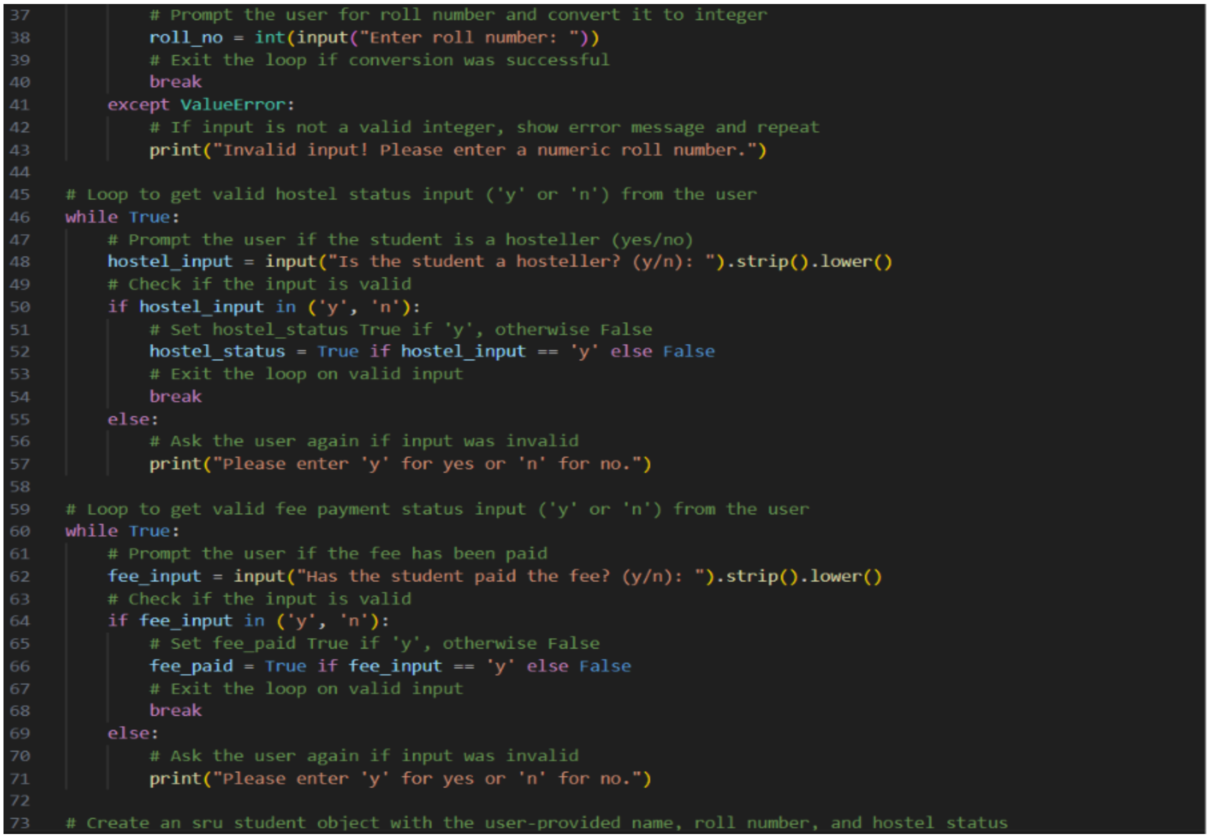
Task-2:

Generate 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/step.

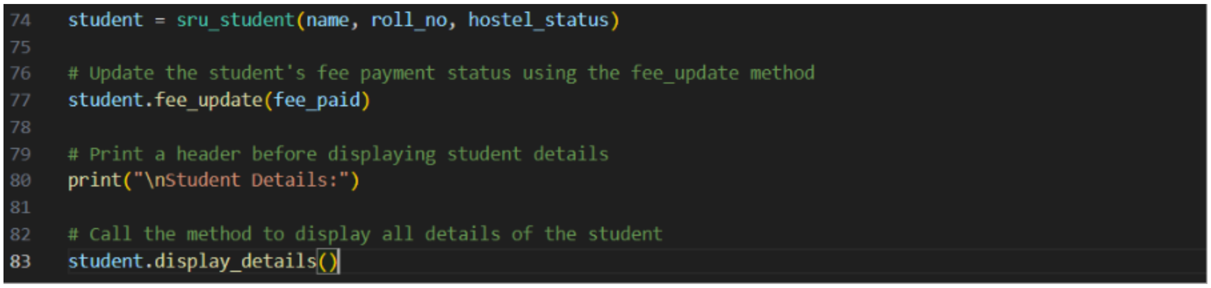
GitHub Copilot:



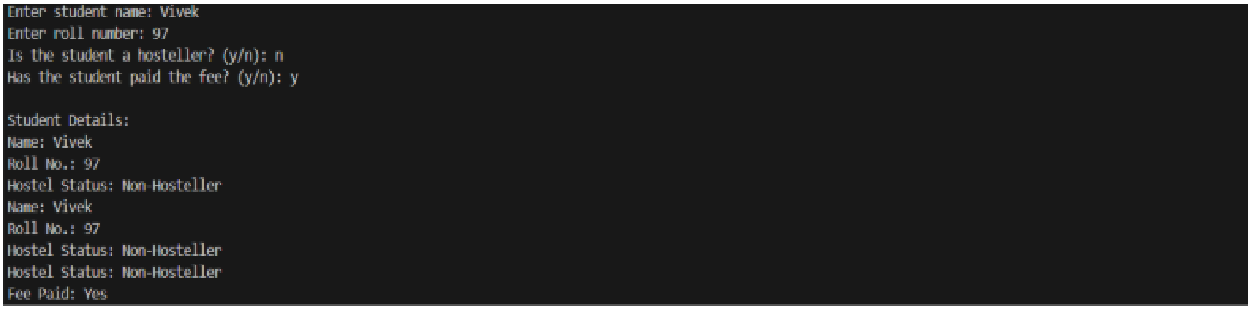
```python
# Define the sru_student class to model a student
class sru_student:
    # Constructor method to initialize student attributes
    def __init__(self, name, roll_no, hostel_status):
        # Assign the student's name to the instance variable
        self.name = name
        # Assign the student's roll number to the instance variable
        self.roll_no = roll_no
        # Assign the student's hostel status (True/False) to the instance variable
        self.hostel_status = hostel_status
        # Initialize fee_paid attribute to False by default (fee not paid)
        self.fee_paid = False

    # Method to update the fee payment status of the student
    def fee_update(self, status):
        # Update the fee_paid attribute with the provided status (True/False)
        self.fee_paid = status

    # Method to display all details of the student
    def display_details(self):
        # Print the student's name
        print(f"Name: {self.name}")
        # Print the student's roll number
        print(f"Roll No.: {self.roll_no}")
        # Print the student's hostel status in a readable form
        print(f"Hostel Status: {'Hosteller' if self.hostel_status else 'Non-Hosteller'}")
        # Print whether the fee has been paid or not
        print(f"Fee Paid: {'Yes' if self.fee_paid else 'No'}")


# Prompt the user to enter the student's name and store it in a variable
name = input("Enter student name: ")

# Loop to get a valid roll number from the user
while True:
    try:
        # Prompt the user for roll number and convert it to integer
```



```python
        # Prompt the user for roll number and convert it to integer
        roll_no = int(input("Enter roll number: "))
        # Exit the loop if conversion was successful
        break
    except ValueError:
        # If input is not a valid integer, show error message and repeat
        print("Invalid input! Please enter a numeric roll number.")

# Loop to get valid hostel status input ('y' or 'n') from the user
while True:
    # Prompt the user if the student is a hosteller (yes/no)
    hostel_input = input("Is the student a hosteller? (y/n): ").strip().lower()
    # Check if the input is valid
    if hostel_input in ('y', 'n'):
        # Set hostel_status True if 'y', otherwise False
        hostel_status = True if hostel_input == 'y' else False
        # Exit the loop on valid input
        break
    else:
        # Ask the user again if input was invalid
        print("Please enter 'y' for yes or 'n' for no.")

# Loop to get valid fee payment status input ('y' or 'n') from the user
while True:
    # Prompt the user if the fee has been paid
    fee_input = input("Has the student paid the fee? (y/n): ").strip().lower()
    # Check if the input is valid
    if fee_input in ('y', 'n'):
        # Set fee_paid True if 'y', otherwise False
        fee_paid = True if fee_input == 'y' else False
        # Exit the loop on valid input
        break
    else:
        # Ask the user again if input was invalid
        print("Please enter 'y' for yes or 'n' for no.")

# Create an sru student object with the user-provided name, roll number, and hostel status
```

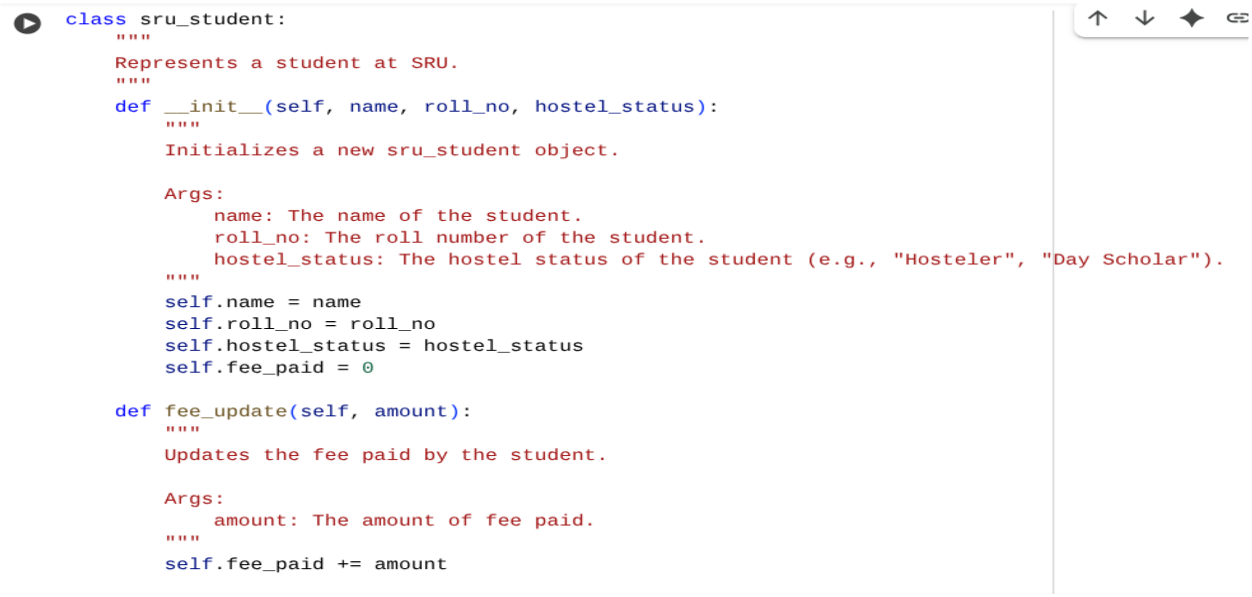


```python
student = sru_student(name, roll_no, hostel_status)

# Update the student's fee payment status using the fee_update method
student.fee_update(fee_paid)

# Print a header before displaying student details
print("\nStudent Details:")

# Call the method to display all details of the student
student.display_details()
```

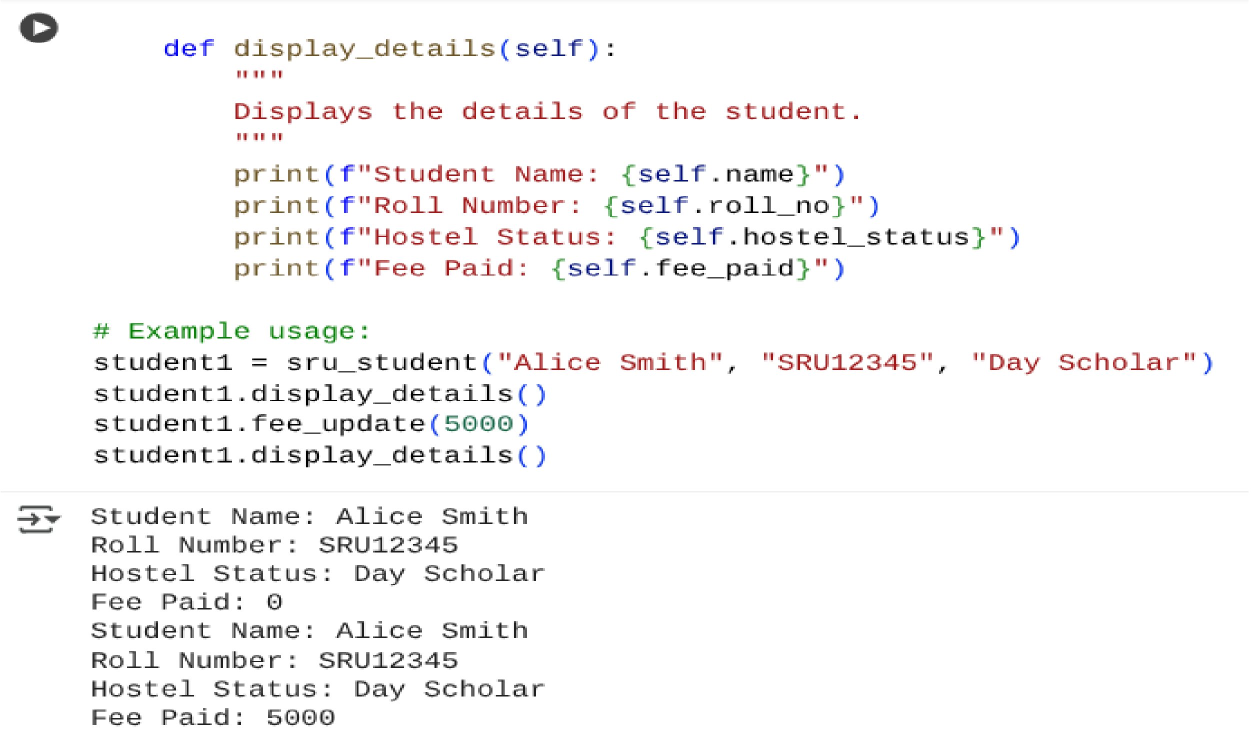


```
Enter student name: Vivek
Enter roll number: 97
Is the student a hosteller? (y/n): n
Has the student paid the fee? (y/n): y

Student Details:
Name: Vivek
Roll No.: 97
Hostel Status: Non-Hosteller
Name: Vivek
Roll No.: 97
Hostel Status: Non-Hosteller
Hostel Status: Non-Hosteller
Fee Paid: Yes
```

Gemini: Without comments:



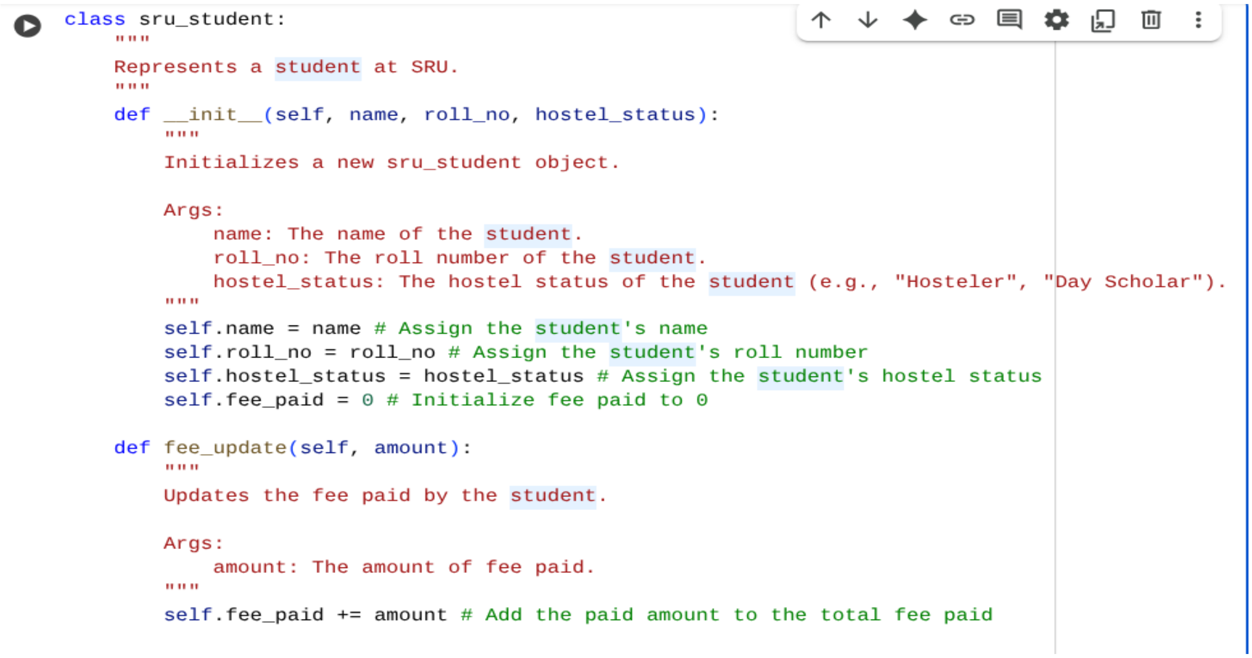
```python
class sru_student:
    """
    Represents a student at SRU.
    """
    def __init__(self, name, roll_no, hostel_status):
        """
        Initializes a new sru_student object.

        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., "Hosteler", "Day Scholar").
        """
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee_paid = 0

    def fee_update(self, amount):
        """
        Updates the fee paid by the student.

        Args:
            amount: The amount of fee paid.
        """
        self.fee_paid += amount
```



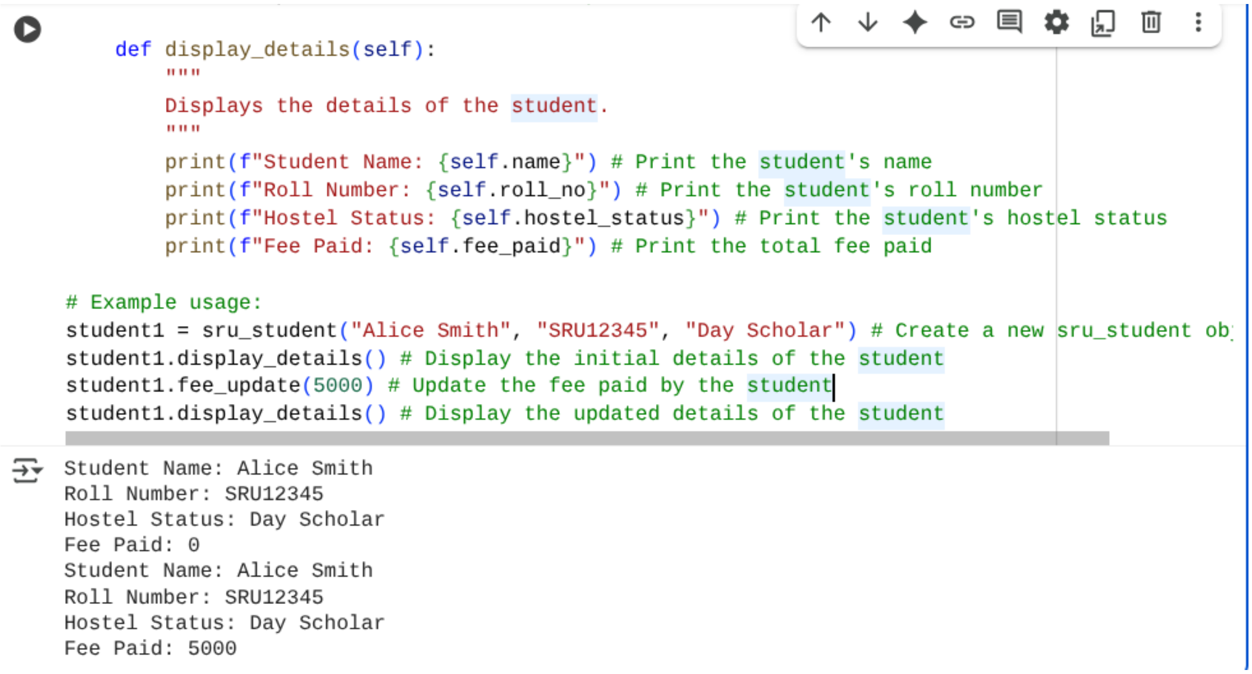
```python
    def display_details(self):
        """
        Displays the details of the student.
        """
        print(f"Student Name: {self.name}")
        print(f"Roll Number: {self.roll_no}")
        print(f"Hostel Status: {self.hostel_status}")
        print(f"Fee Paid: {self.fee_paid}")

# Example usage:
student1 = sru_student("Alice Smith", "SRU12345", "Day Scholar")
student1.display_details()
student1.fee_update(5000)
student1.display_details()
```

```
Student Name: Alice Smith
Roll Number: SRU12345
Hostel Status: Day Scholar
Fee Paid: 0
Student Name: Alice Smith
Roll Number: SRU12345
Hostel Status: Day Scholar
Fee Paid: 5000
```

With comments:



```
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., "Hosteler", "Day Scholar").
        """
        self.name = name # Assign the student's name
        self.roll_no = roll_no # Assign the student's roll number
        self.hostel_status = hostel_status # Assign the student's hostel status
        self.fee_paid = 0 # Initialize fee paid to 0

    def fee_update(self, amount):
        """
        Updates the fee paid by the student.

        Args:
            amount: The amount of fee paid.
        """
        self.fee_paid += amount # Add the paid amount to the total fee paid
```



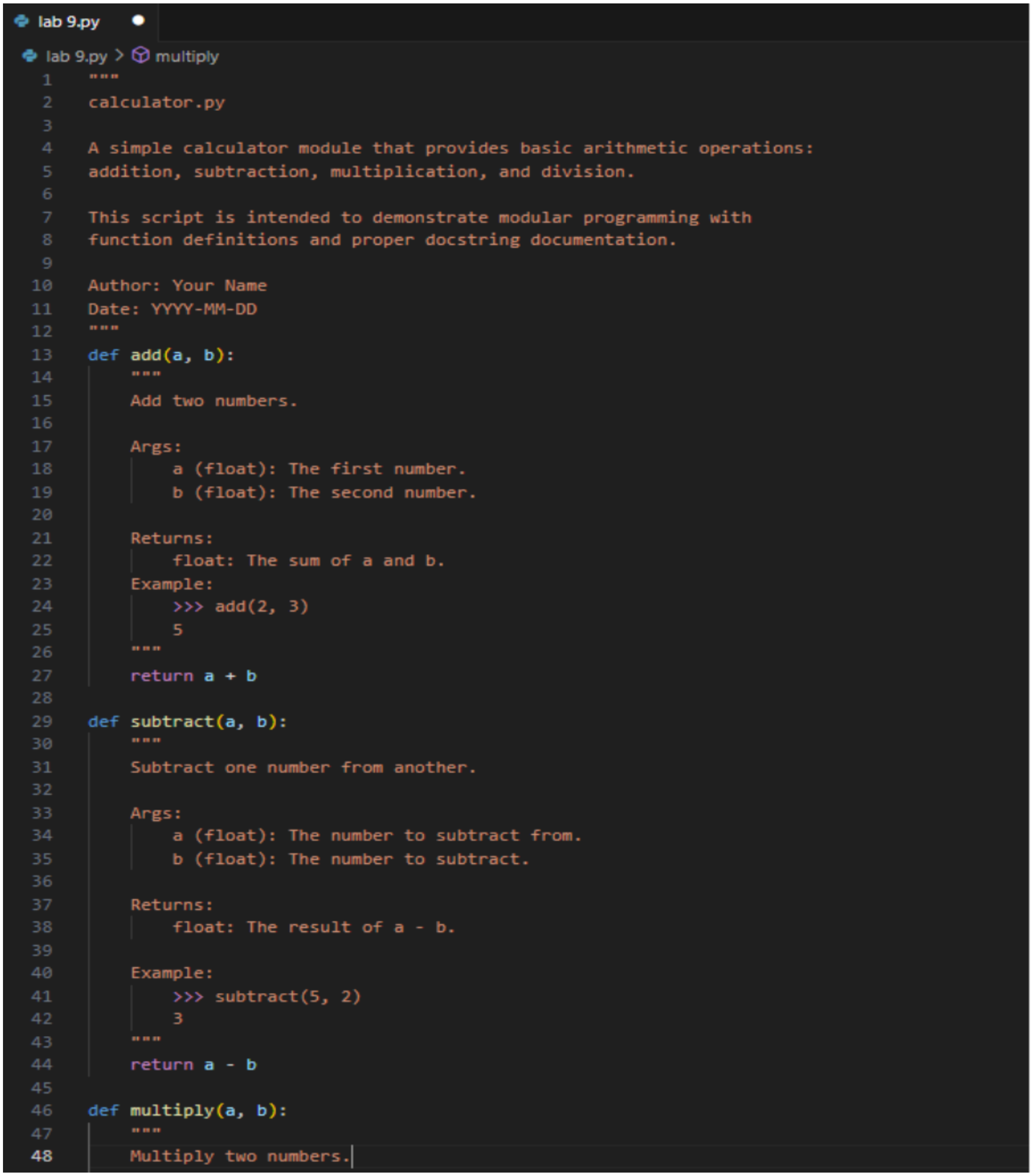
```
    def display_details(self):
        """
        Displays the details of the student.
        """
        print(f"Student Name: {self.name}") # Print the student's name
        print(f"Roll Number: {self.roll_no}") # Print the student's roll number
        print(f"Hostel Status: {self.hostel_status}") # Print the student's hostel status
        print(f"Fee Paid: {self.fee_paid}") # Print the total fee paid

# Example usage:
student1 = sru_student("Alice Smith", "SRU12345", "Day Scholar") # Create a new sru_student ob
student1.display_details() # Display the initial details of the student
student1.fee_update(5000) # Update the fee paid by the student
student1.display_details() # Display the updated details of the student
```

```
Student Name: Alice Smith
Roll Number: SRU12345
Hostel Status: Day Scholar
Fee Paid: 0
Student Name: Alice Smith
Roll Number: SRU12345
Hostel Status: Day Scholar
Fee Paid: 5000
```

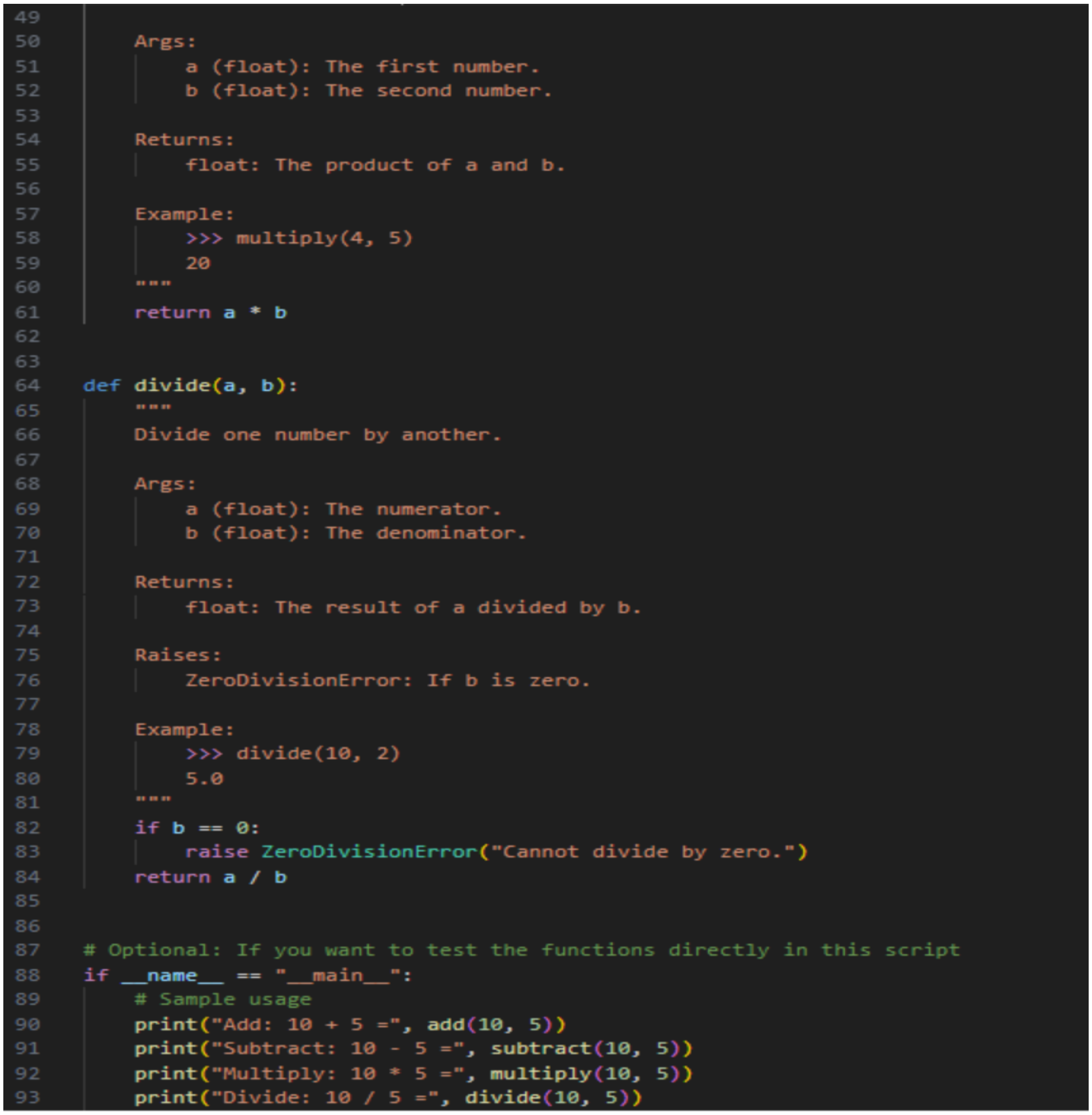
Task-3:

Github copilot:

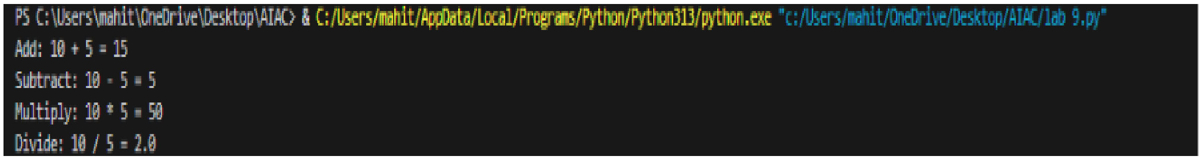
Generate a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide). Include module-level docstring + individual function docstrings.



```python
"""
calculator.py

A simple calculator module that provides basic arithmetic operations:
addition, subtraction, multiplication, and division.

This script is intended to demonstrate modular programming with
function definitions and proper docstring documentation.

Author: Your Name
Date: YYYY-MM-DD
"""
def add(a, b):
    """
    Add two numbers.

    Args:
        a (float): The first number.
        b (float): The second number.

    Returns:
        float: The sum of a and b.
    Example:
        >>> add(2, 3)
        5
    """
    return a + b

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

    Args:
        a (float): The number to subtract from.
        b (float): The number to subtract.

    Returns:
        float: The result of a - b.

    Example:
        >>> subtract(5, 2)
        3
    """
    return a - b

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



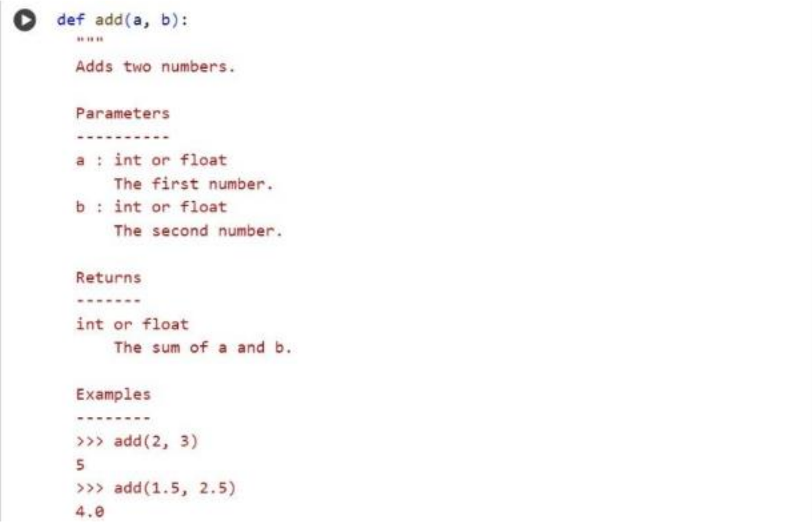
```python

    Args:
        a (float): The first number.
        b (float): The second number.

    Returns:
        float: The product of a and b.

    Example:
        >>> multiply(4, 5)
        20
    """
    return a * b


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

    Args:
        a (float): The numerator.
        b (float): The denominator.

    Returns:
        float: The result of a divided by b.

    Raises:
        ZeroDivisionError: If b is zero.

    Example:
        >>> divide(10, 2)
        5.0
    """
    if b == 0:
        raise ZeroDivisionError("Cannot divide by zero.")
    return a / b


# Optional: If you want to test the functions directly in this script
if __name__ == "__main__":
    # Sample usage
    print("Add: 10 + 5 =", add(10, 5))
    print("Subtract: 10 - 5 =", subtract(10, 5))
    print("Multiply: 10 * 5 =", multiply(10, 5))
    print("Divide: 10 / 5 =", divide(10, 5))
```



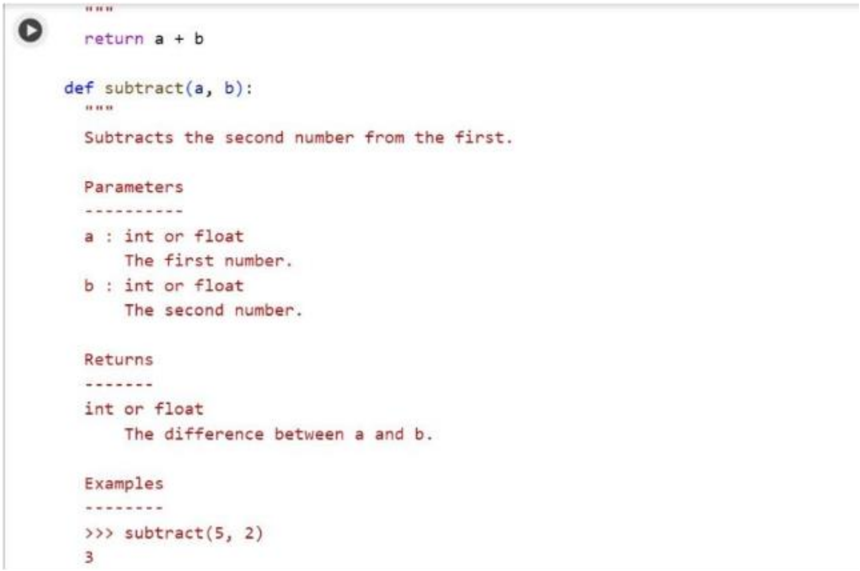
```
PS C:\Users\mahit\OneDrive\Desktop\AIAC> & C:/Users/mahit/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/mahit/OneDrive/Desktop/AIAC/lab 9.py"
Add: 10 + 5 = 15
Subtract: 10 - 5 = 5
Multiply: 10 * 5 = 50
Divide: 10 / 5 = 2.0
```

Gemini:

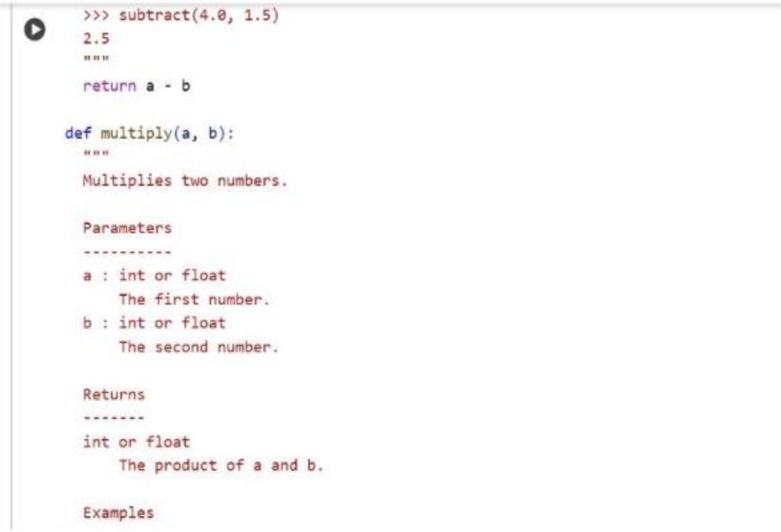
Generate a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide) . Write docstring in code with NumPy Style



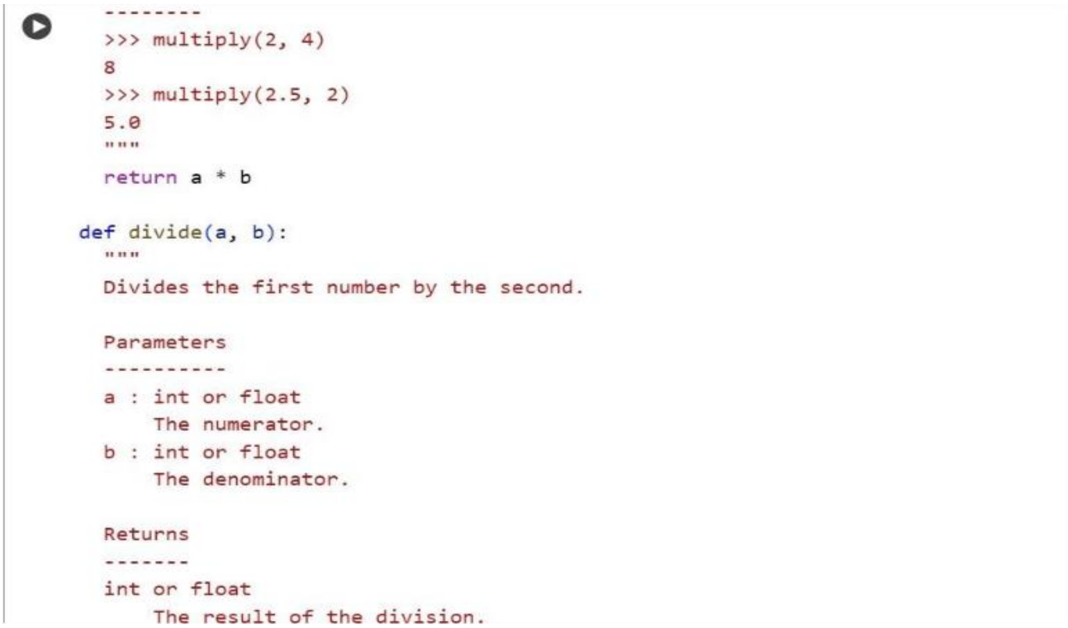
```
def add(a, b):
    """
    Adds two numbers.

    Parameters
    ----------
    a : int or float
        The first number.
    b : int or float
        The second number.

    Returns
    -------
    int or float
        The sum of a and b.

    Examples
    --------
    >>> add(2, 3)
    5
    >>> add(1.5, 2.5)
    4.0
```



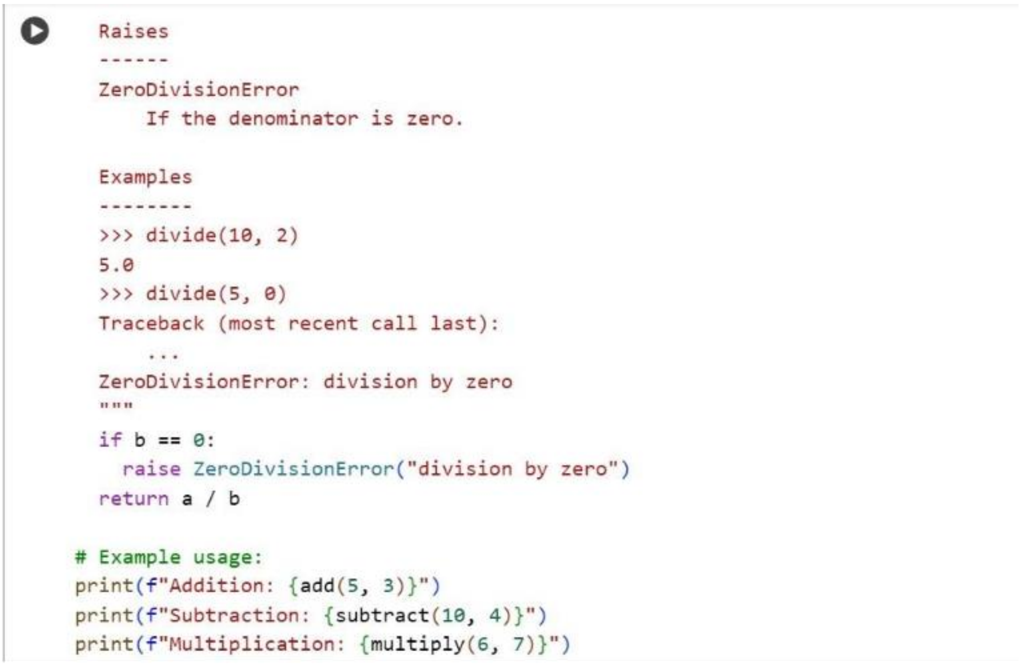
```
    """
    return a + b

def subtract(a, b):
    """
    Subtracts the second number from the first.

    Parameters
    ----------
    a : int or float
        The first number.
    b : int or float
        The second number.

    Returns
    -------
    int or float
        The difference between a and b.

    Examples
    --------
    >>> subtract(5, 2)
    3
```



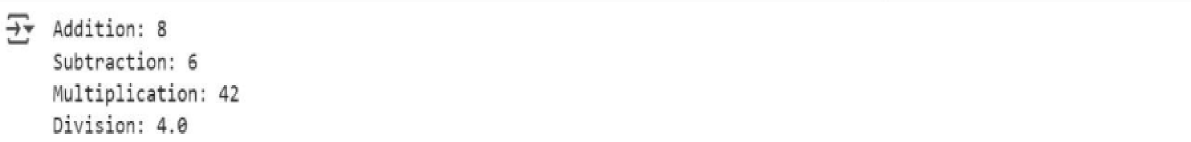
```
  >>> subtract(4.0, 1.5)
  2.5
  """
  return a - b

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

  Parameters
  ----------
  a : int or float
      The first number.
  b : int or float
      The second number.

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

  Examples
```



```
  --------
  >>> multiply(2, 4)
  8
  >>> multiply(2.5, 2)
  5.0
  """
  return a * b

def divide(a, b):
  """
  Divides the first number by the second.

  Parameters
  ----------
  a : int or float
      The numerator.
  b : int or float
      The denominator.

  Returns
  -------
  int or float
      The result of the division.
```



```
  Raises
  ------
  ZeroDivisionError
      If the denominator is zero.

  Examples
  --------
  >>> divide(10, 2)
  5.0
  >>> divide(5, 0)
  Traceback (most recent call last):
      ...
  ZeroDivisionError: division by zero
  """
  if b == 0:
    raise ZeroDivisionError("division by zero")
  return a / b

# Example usage:
print(f"Addition: {add(5, 3)}")
print(f"Subtraction: {subtract(10, 4)}")
print(f"Multiplication: {multiply(6, 7)}")
```



```
Addition: 8
Subtraction: 6
Multiplication: 42
Division: 4.0
```