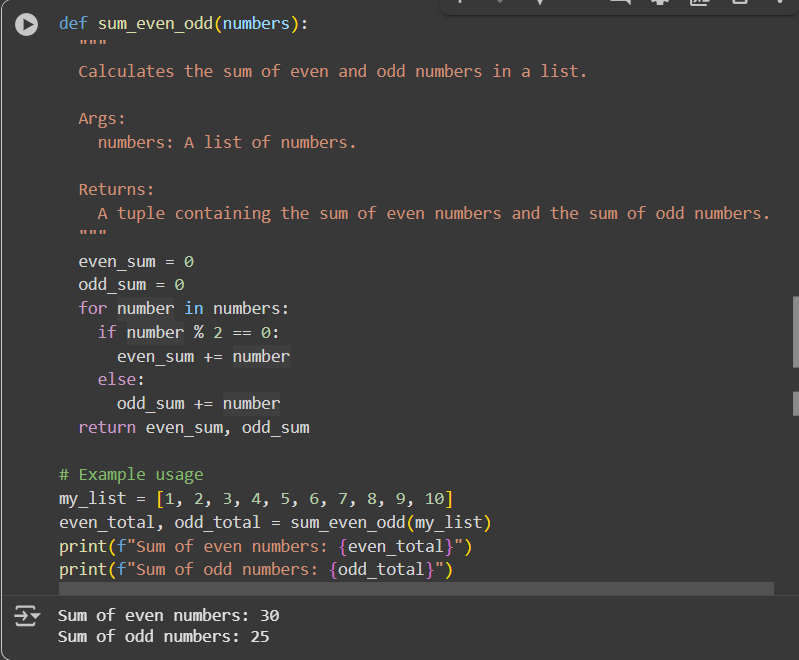
**AI ASSISTED CODING LAB 9**

**Hall ticket number : 2403A52373**

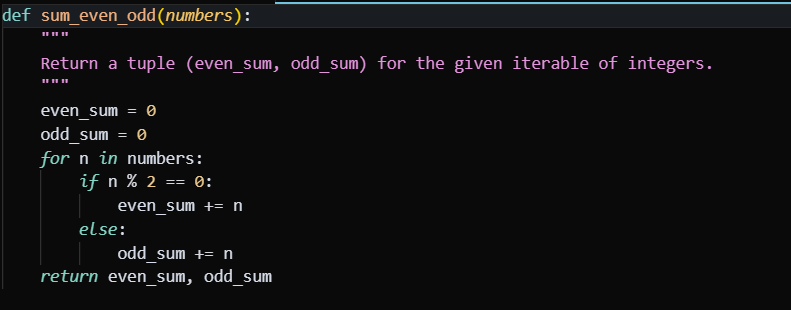
**Name :KODURU ESWAR REDDY**

**Prompt 01: generate a Python function to return the sum of even and odd numbers in the given list.**

**Code&Output:  
Gemini:**

****

**Cursor:**

****

**Code explanation:**

1. Function Definition:

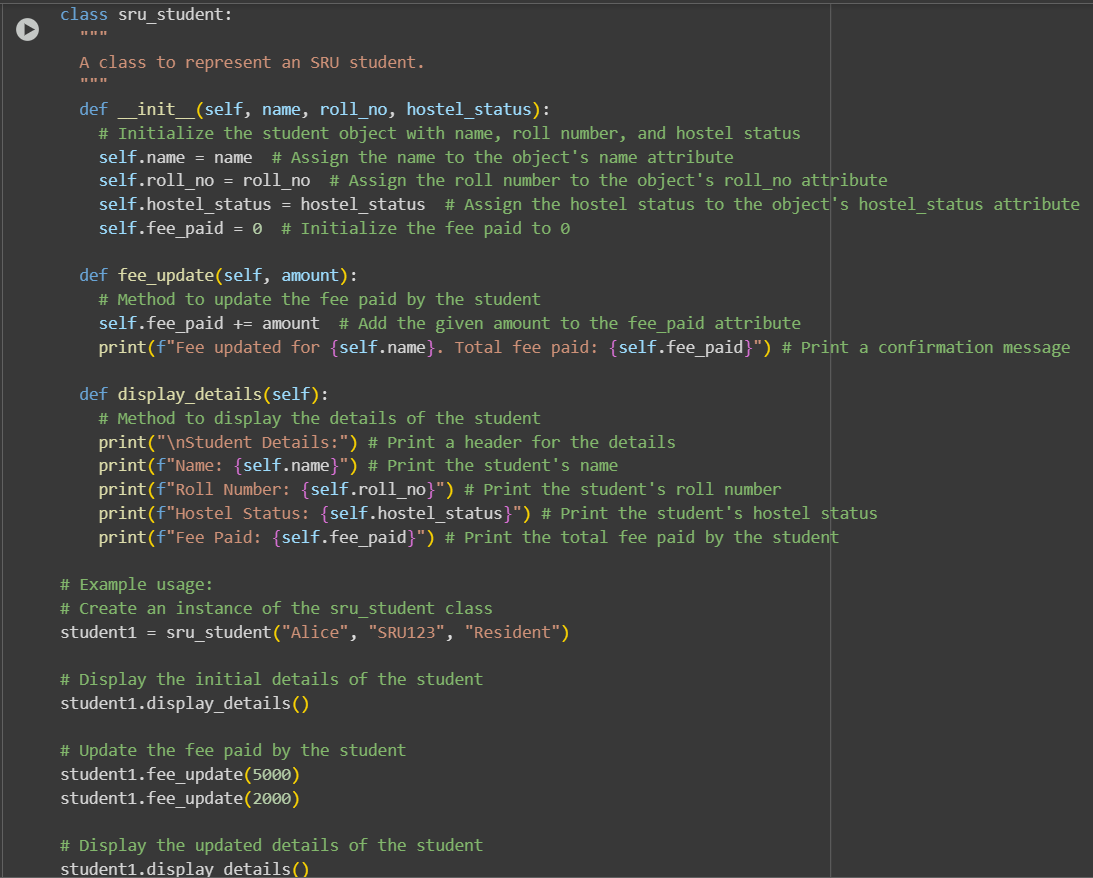
* + def sum\_even\_odd(numbers): defines a function named sum\_even\_odd that accepts one argument, numbers.
  + The docstring within triple quotes explains what the function does, its arguments (Args), and what it returns (Returns).

1. Initialization:
   * even\_sum = 0 and odd\_sum = 0 . Initialize two variables to store the running totals of even and odd numbers, starting at zero.
2. Iterating through the list:
   * for number in numbers: starts a loop that goes through each number in the input numbers list.
3. Checking for even or odd:
   * if number % 2 == 0: checks if the current number is even by using the modulo operator (%). If the remainder when divided by 2 is 0, the number is even.
   * If the number is even, even\_sum += number adds the number to the even\_sum.
   * else: handles the case where the number is not even (meaning it's odd).
   * If the number is odd, odd\_sum += number adds the number to the odd\_sum.
4. Returning the sums:
   * return even\_sum, odd\_sum returns a tuple containing the final calculated even\_sum and odd\_sum.
5. Example Usage:
   * my\_list = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] creates a sample list.
   * even\_total, odd\_total = sum\_even\_odd(my\_list) calls the sum\_even\_odd function with my\_list and unpacks the returned tuple into two variables, even\_total and odd\_total.
   * The print statements display the calculated sums of even and odd numbers using an f-string for formatted output.

**Prompt 02:**generate a python program for sru\_student class with attributes like name, roll no., hostel\_status and fee\_update method and display\_details method.with comments for each line

**Code:**

**GEMINI:**

****

**CURSOR:**

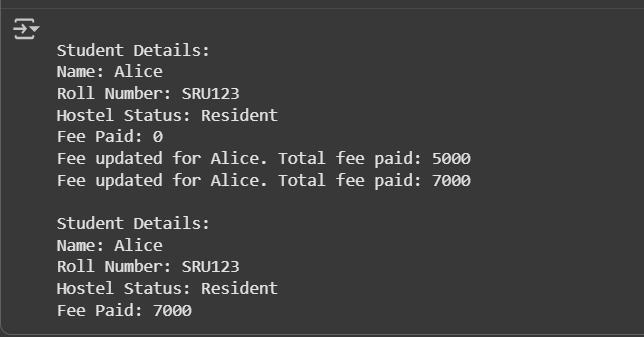
**A screen shot of a computer program

AI-generated content may be incorrect.**

**Code explanation:**

1. Class Definition:
   * class sru\_student: defines a new class named sru\_student.
   * The docstring provides a brief description of the class.
2. \_\_init\_\_ Method (Constructor):
   * def \_\_init\_\_(self, name, roll\_no, hostel\_status): is the constructor method. It's called when you create a new object (instance) of the sru\_student class.
   * self refers to the instance of the class being created.
   * name, roll\_no, and hostel\_status are parameters that you pass when creating a student object.
   * self.name = name, self.roll\_no = roll\_no, and self.hostel\_status = hostel\_status assign the values passed to the constructor to the object's attributes (properties).
   * self.fee\_paid = 0 initializes the fee\_paid attribute to 0 for every new student object.
3. fee\_update Method:
   * def fee\_update(self, amount): defines a method called fee\_update that takes self (the object itself) and an amount as input.
   * self.fee\_paid += amount adds the given amount to the fee\_paid attribute of the specific student object.
   * The print statement confirms the fee update and shows the total fee paid.
4. display\_details Method:
   * def display\_details(self): defines a method called display\_details that takes only self as input.
   * This method prints the details of the student object, including their name, roll number, hostel status, and the total fee paid.
5. Example Usage:
   * student1 = sru\_student("Alice", "SRU123", "Resident") creates an instance of the sru\_student class named student1 with the specified details.
   * student1.display\_details() calls the display\_details method on the student1 object to show its initial information.
   * student1.fee\_update(5000) and student1.fee\_update(2000) call the fee\_update method to add fee amounts to student1.
   * The second student1.The display\_details() call displays the updated details, including the total fee paid.

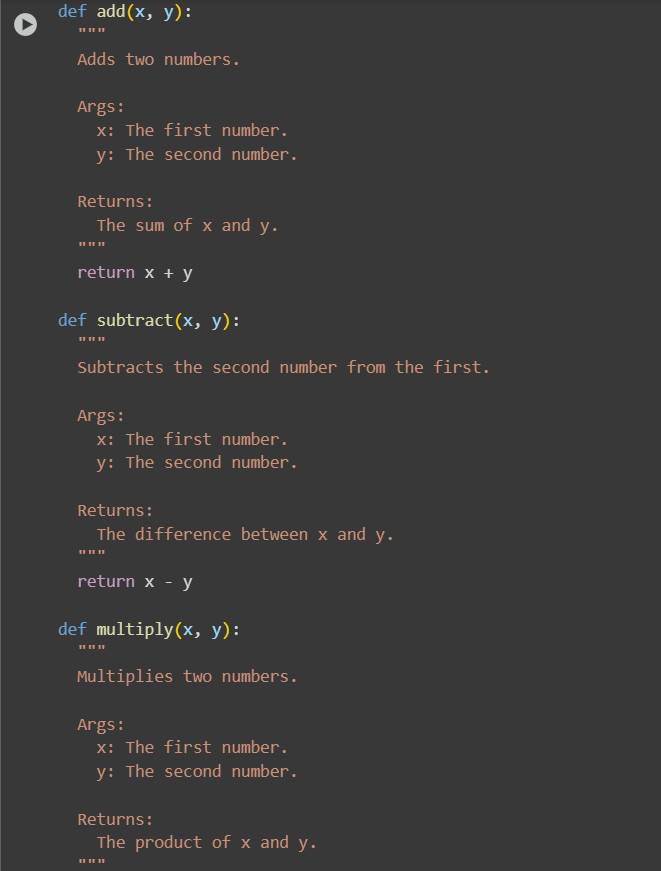
**Output:**

****

**Prompt 03:**Generate a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide) with docstrings.

**Code&OUTPUT:**

**GEMINI:**

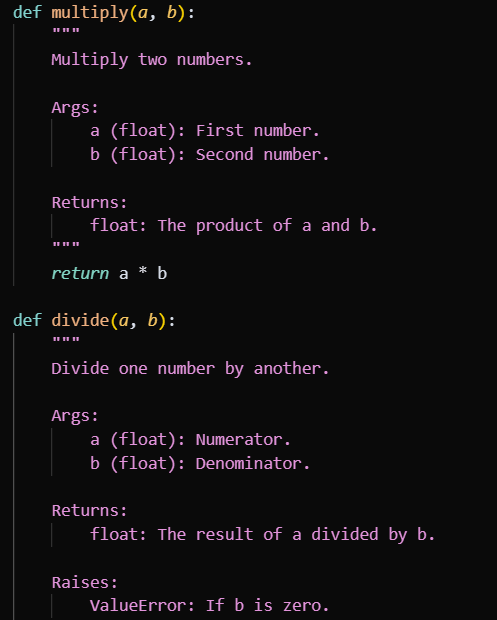
****

**A screenshot of a computer program

AI-generated content may be incorrect.**

**CURSOR:A screenshot of a computer program

AI-generated content may be incorrect.**

**  
A screen shot of a computer

AI-generated content may be incorrect.**

**Code explanation:**

1. add(x, y):
   * Takes two arguments, x and y.
   * The docstring explains that it adds two numbers and specifies the arguments and return value.
   * return x + y calculates the sum of x and y and returns the result.
2. subtract(x, y):
   * Takes two arguments, x and y.
   * The docstring explains that it subtracts the second number from the first and specifies the arguments and return value.
   * return x-y calculates the difference between x and y and returns the result.
3. multiply(x, y):
   * Takes two arguments, x and y.
   * The docstring explains that it multiplies two numbers and specifies the arguments and return value.
   * return x\*y calculates the product of x and y and returns the result.
4. divide(x, y):
   * Takes two arguments, x and y.
   * The docstring explains that it divides the first number by the second, specifies the arguments and return value, and notes that it raises a ZeroDivisionError if the divisor is zero.
   * if y == 0: checks if the second number (y) is zero.
   * If y is zero, raise ZeroDivisionError("Cannot divide by zero!") to prevent division by zero.
   * If y is not zero, return x / y calculates the division of x by y and returns the result.
5. Example Usage:
   * num1 = 10 and num2 = 5 . Set up two variables with example numbers.
   * The print statements demonstrate calling each of the functions with num1 and num2 and display the results using f-strings for clear output.
   * The commented-out line # print(f"{num1} / 0 = {divide(num1, 0)}") shows how you could call the divide function with zero as the divisor, which would trigger the ZeroDivisionError.