**ASSIGNMENT – 9.3**

*Name – Sriram Sushanth*

*Roll – 2403a54079*

*Batch(DS) – 03*

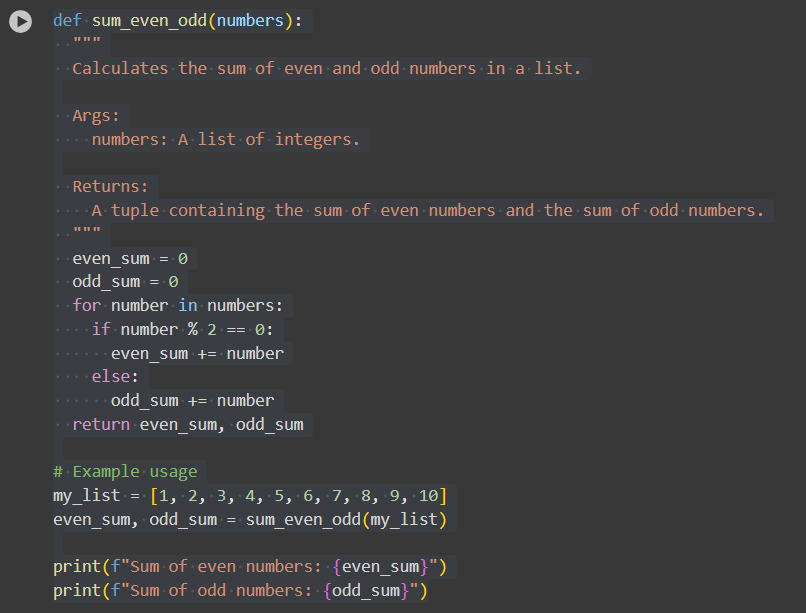
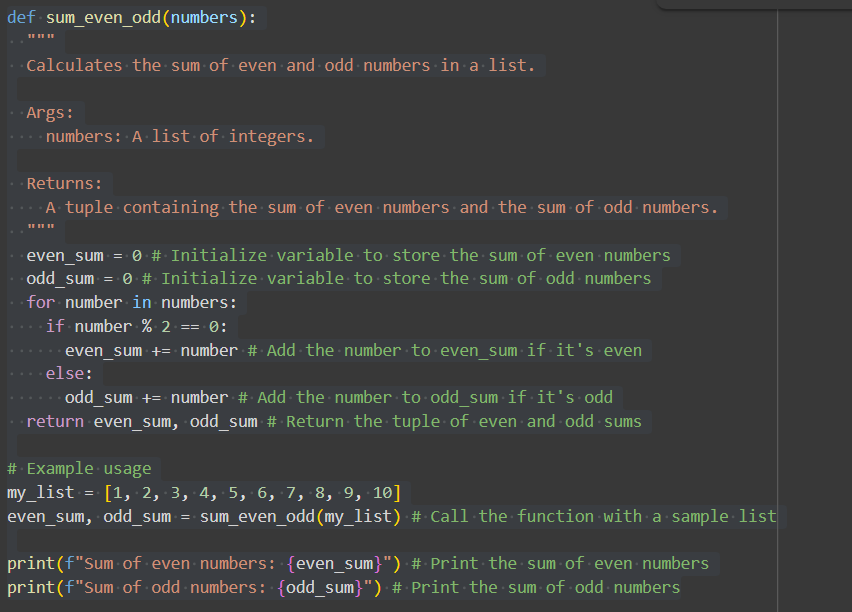
**TASK 1:**

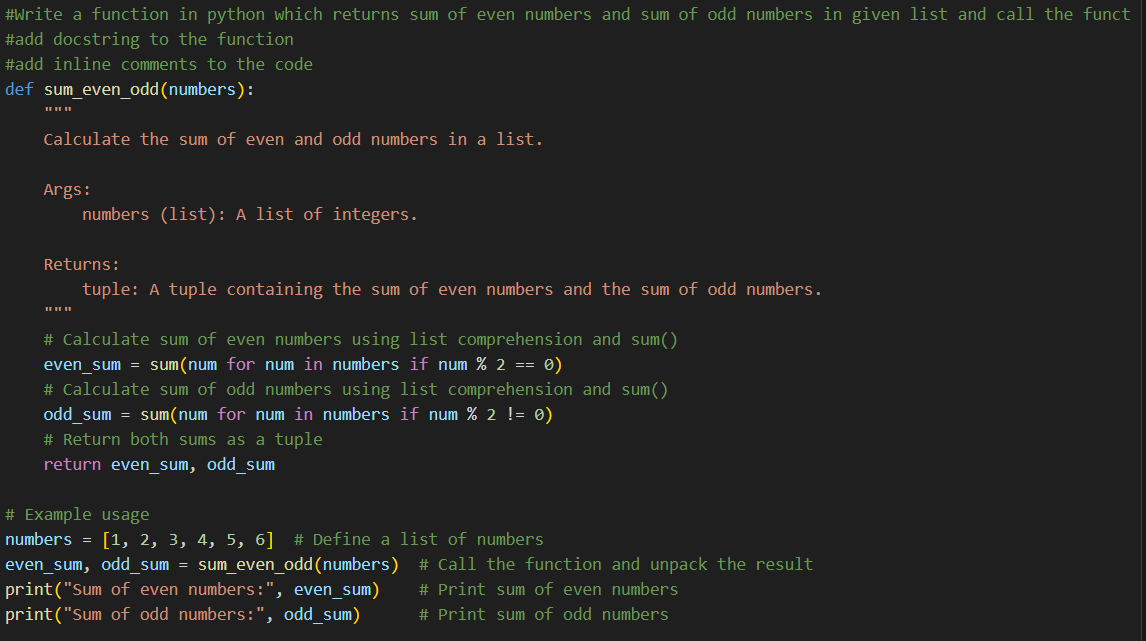
**Task Description#1 Basic Docstring Generation**

* 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.

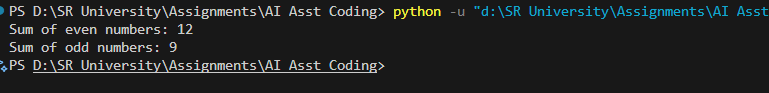
**PROMPT:** **Write a function in python which returns sum of even numbers and sum of odd numbers in given list and call the function**

**CODE:**

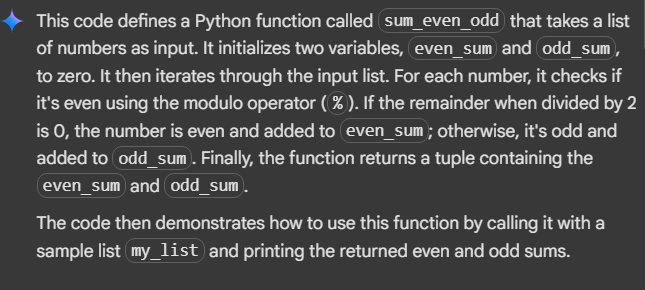
**** ****

****

**OUTPUT:**

**** 

**EXPLANATION:**

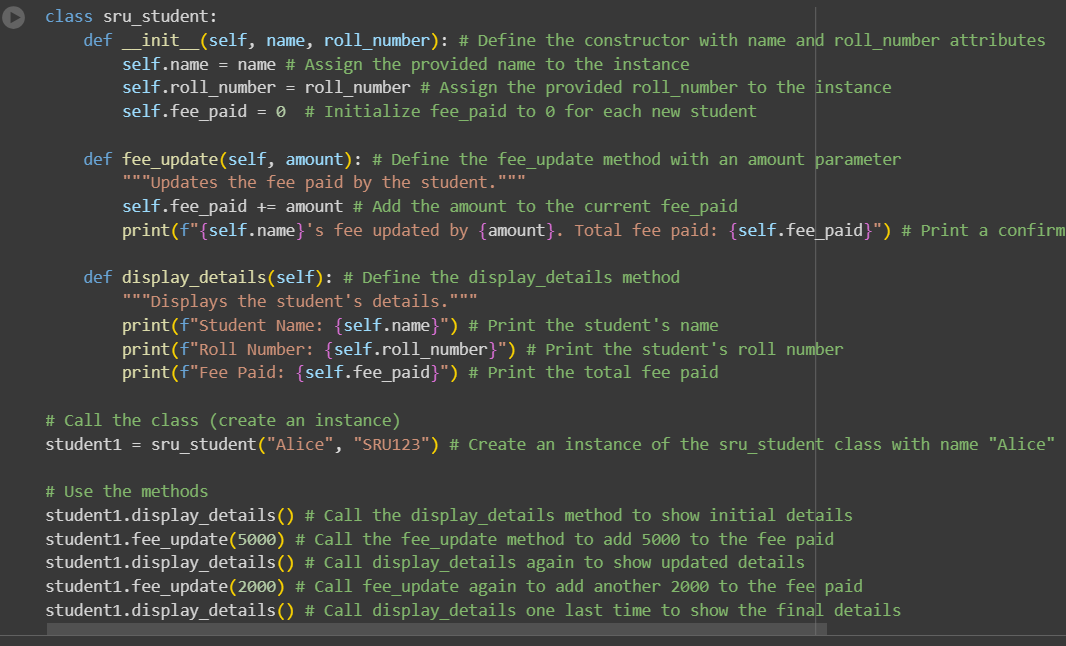
****

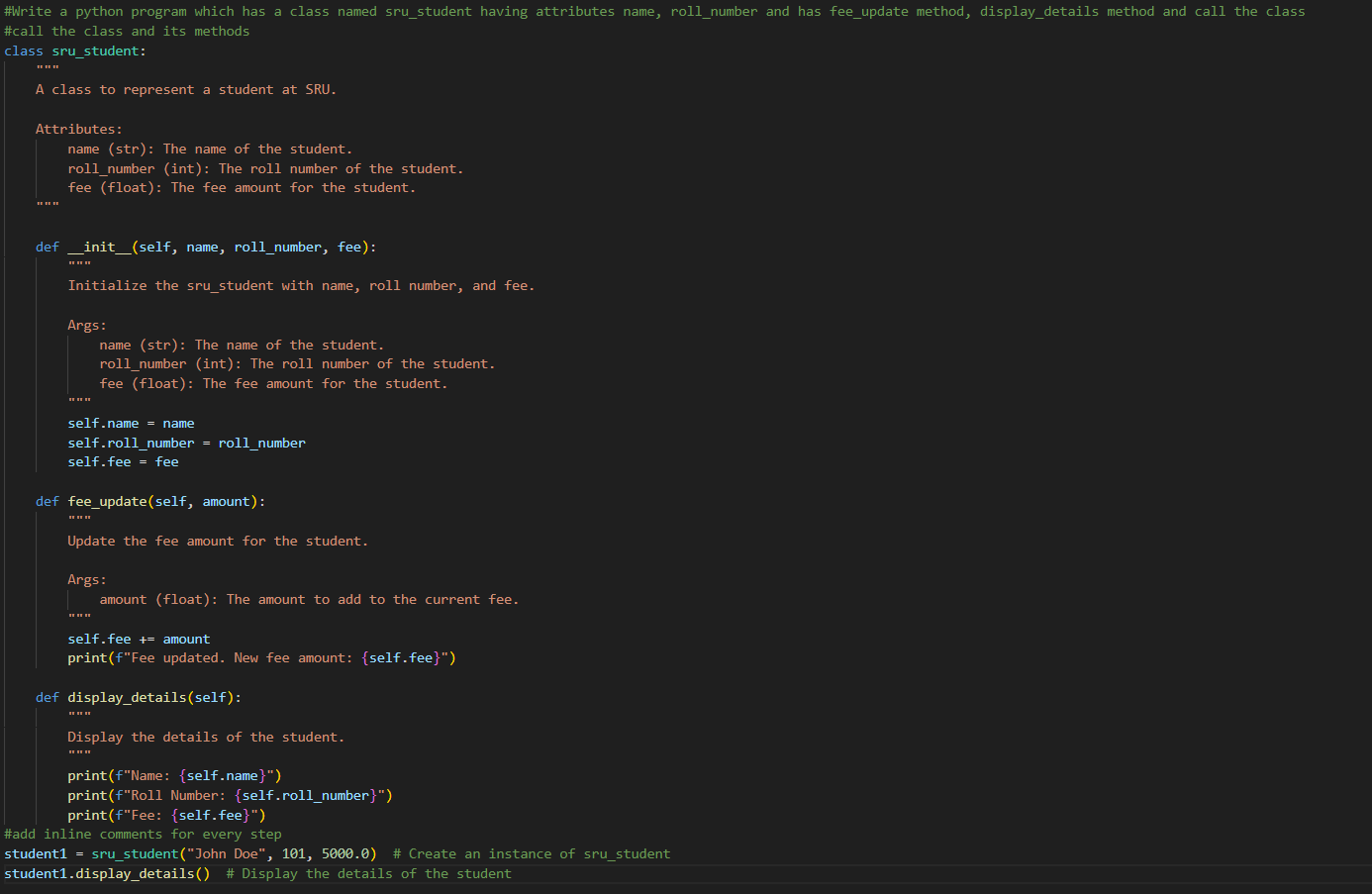
**Task Description#2 Automatic Inline Comments**

* 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.

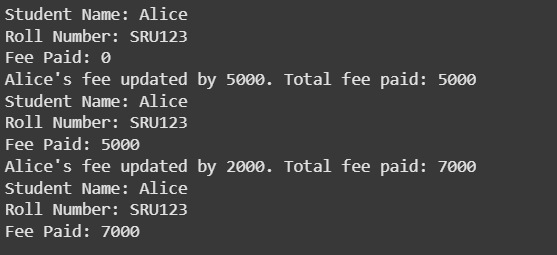
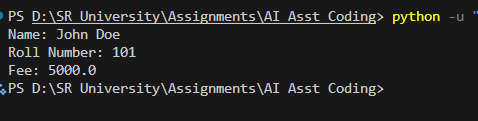
**PROMPT:** **Write a python program which has a class named sru\_student having attributes name, roll\_number and has fee\_update method, display\_details method and call the class**

**CODE:**

****

****

**OUTPUT:**

**** ****

**EXPLANATION:**

This code defines a Python class called sru\_student.

Here's a breakdown of the code:

* class sru\_student:: This line defines a new class named sru\_student.
* \_\_init\_\_(self, name, roll\_number):: This 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 itself.
  + name and roll\_number are parameters that you pass when creating a new student object.
  + Inside the constructor, self.name = name and self.roll\_number = roll\_number assign the provided values to the name and roll\_number attributes of the student object.
  + self.fee\_paid = 0 initializes the fee\_paid attribute to 0 for every new student.
* fee\_update(self, amount):: This method is used to update the fee paid by the student.
  + self refers to the instance of the class.
  + amount is the amount of fee to add.
  + self.fee\_paid += amount adds the specified amount to the current fee\_paid.
  + The print statement displays a confirmation message.
* display\_details(self):: This method displays the student's details.
  + self refers to the instance of the class.
  + The print statements display the student's name, roll number, and fee paid.
* student1 = sru\_student("Alice", "SRU123"): This line creates an instance of the sru\_student class named student1, passing "Alice" as the name and "SRU123" as the roll number.
* student1.display\_details(): This line calls the display\_details method on the student1 object to show its initial details.
* student1.fee\_update(5000): This line calls the fee\_update method on the student1 object to add 5000 to the fee paid.
* student1.display\_details(): This line calls display\_details again to show the updated details.
* student1.fee\_update(2000): This line calls fee\_update again to add another 2000 to the fee paid.
* student1.display\_details(): This line calls display\_details one last time to show the final details.

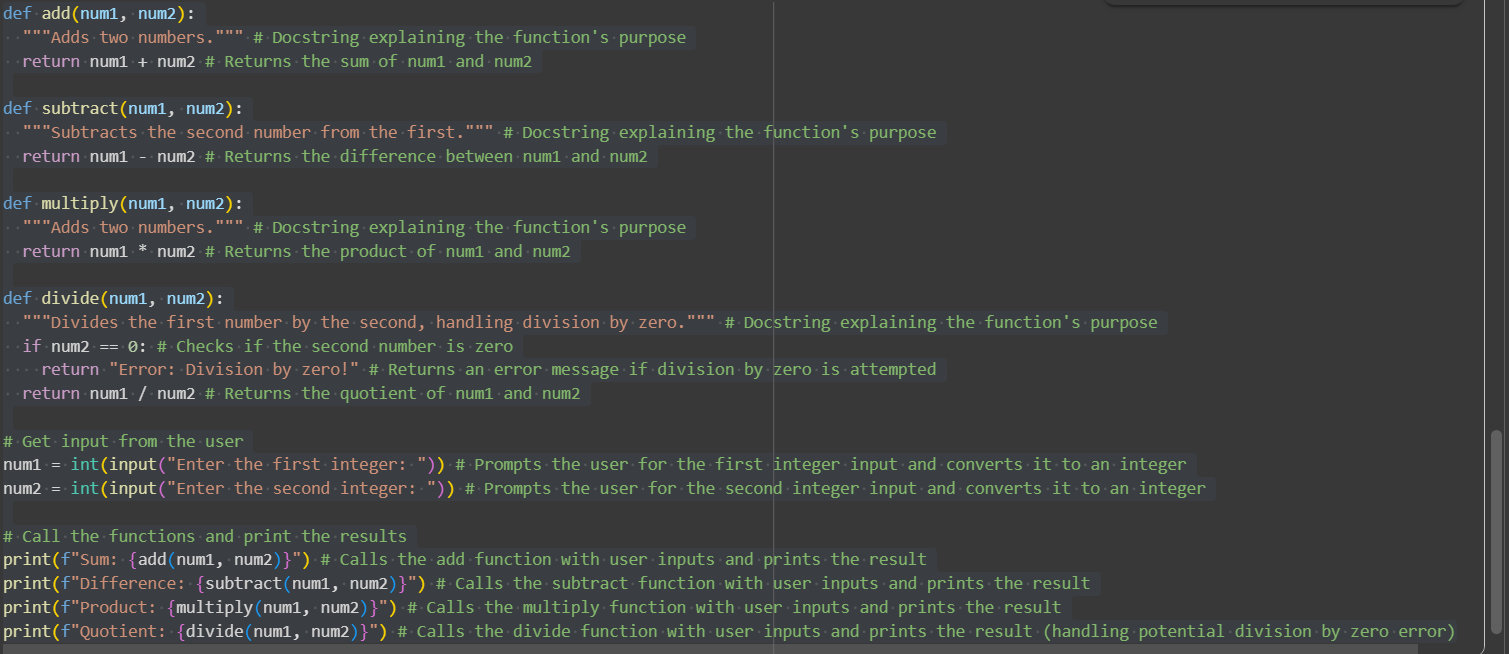
In essence, this code defines a blueprint for creating student objects, each with a name, roll number, and a way to track and display their fee payments.

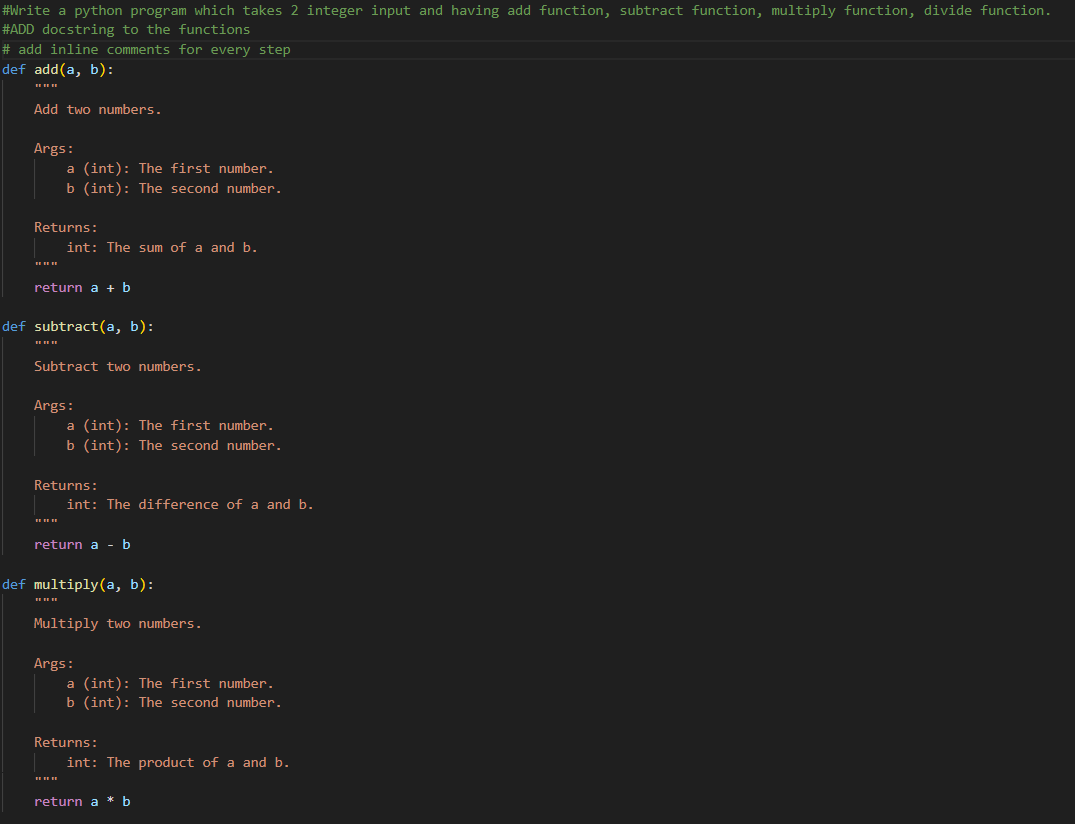
**Task Description#3**

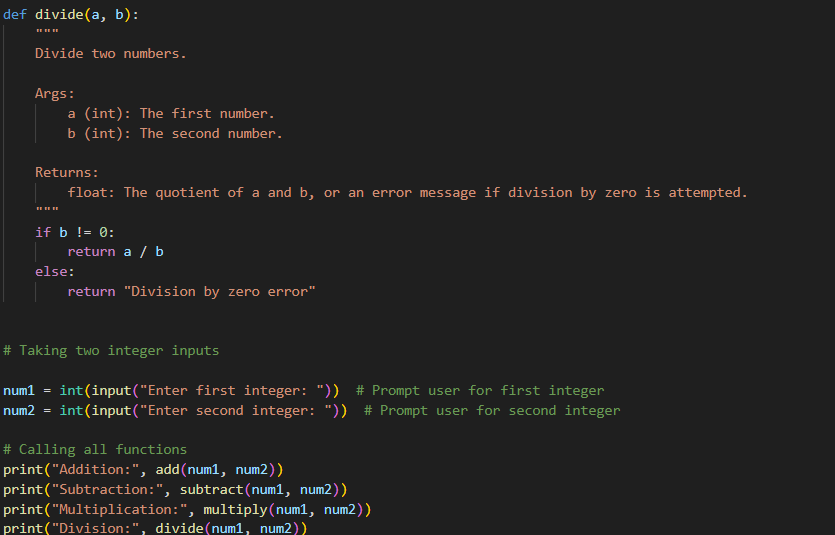
* Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).
* Incorporate manual **docstring** in code with NumPy Style
* Use AI assistance to generate a module-level docstring + individual function docstrings.
* Compare the AI-generated docstring with your manually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

**PROMPT:** **Write a python program which takes 2 integer input and having add fucntion, subtract function, multiply function, divide function.Finally call all function**

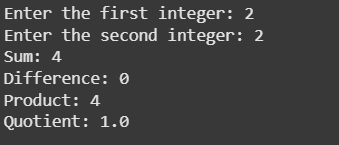
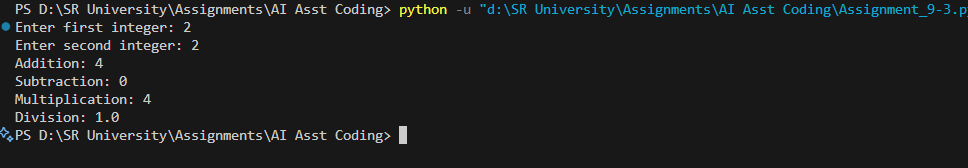
**CODE:**







**OUTPUT:**

**** ****

**EXPLANATION:**

The selected code defines four functions for basic arithmetic operations (addition, subtraction, multiplication, and division) and then takes two integer inputs from the user to demonstrate how these functions are used and print their results.

Here's a breakdown of each part:

* def add(num1, num2):: This defines a function called add that takes two arguments, num1 and num2. The """Adds two numbers.""" is a docstring, which explains what the function does. The line return num1 + num2 calculates the sum of the two numbers and returns the result.
* def subtract(num1, num2):: Similar to add, this defines a function subtract that takes two numbers and returns their difference. The docstring explains its purpose, and return num1 - num2 performs the subtraction.
* def multiply(num1, num2):: This defines the multiply function, which takes two numbers and returns their product using return num1 \* num2. The docstring explains its purpose.
* def divide(num1, num2):: This defines the divide function. It takes two numbers and returns their quotient. It includes an if num2 == 0: check to see if the second number is zero. If it is, it returns the string "Error: Division by zero!" to prevent a division-by-zero error. Otherwise, it returns the result of the division num1 / num2. The docstring explains its purpose and error handling.
* num1 = int(input("Enter the first integer: ")): This line prompts the user to enter the first integer using the input() function. The entered value is then converted to an integer using int() and stored in the variable num1.
* num2 = int(input("Enter the second integer: ")): This line does the same as the previous one, but for the second integer input, storing it in the variable num2.
* print(f"Sum: {add(num1, num2)}"): This line calls the add function with num1 and num2 as arguments. The result returned by the function is then embedded in an f-string and printed to the console, along with the label "Sum: ".
* print(f"Difference: {subtract(num1, num2)}"): This line calls the subtract function with the user inputs and prints the result with the label "Difference: ".
* print(f"Product: {multiply(num1, num2)}"): This line calls the multiply function with the user inputs and prints the result with the label "Product: ".
* print(f"Quotient: {divide(num1, num2)}"): This line calls the divide function with the user inputs and prints the result with the label "Quotient: ". Due to the error handling in the divide function, this line will either print the quotient or the division-by-zero error message.

This code demonstrates how to create modular code using functions to perform specific tasks and how to interact with the user to get input.