**ASSIGNMENT – 9.3**

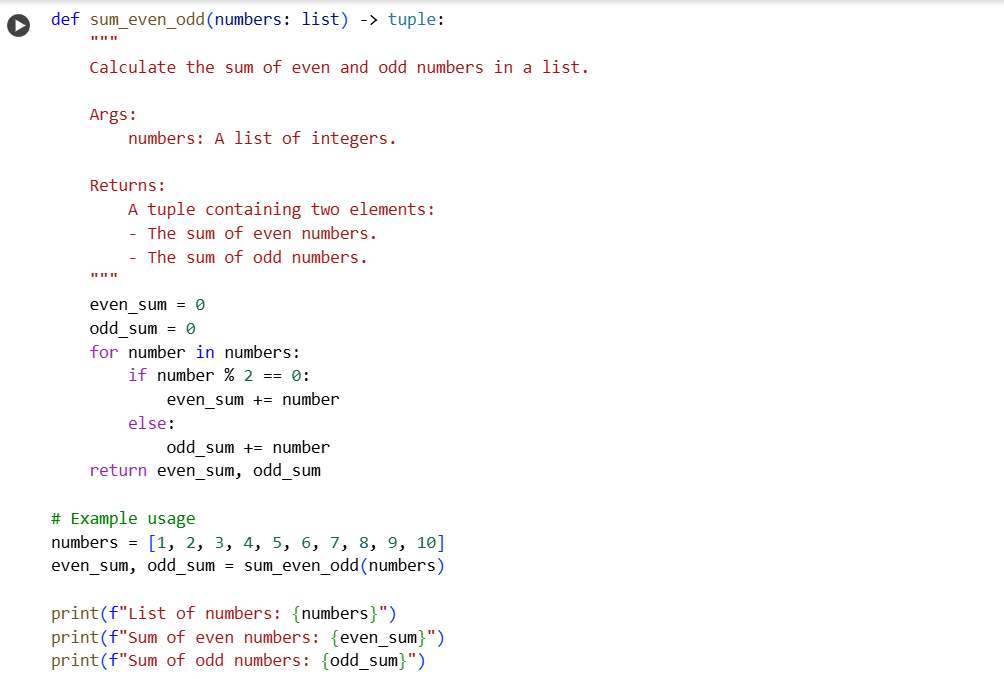
**NAME :** SAI CHARAN .P

**ROLL:NO :** 2403A52343

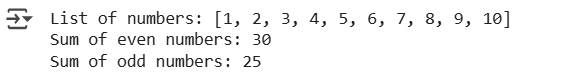
**BATCH :** AI 13

**TASK-1:** Generate a docstring for a Python function that takes a list of integers and returns a tuple containing the sum of even numbers and the sum of odd numbers. Follow a clear and professional style suitable for production code documentation.

**Program in gemini:**

****

**Output in gemini:**

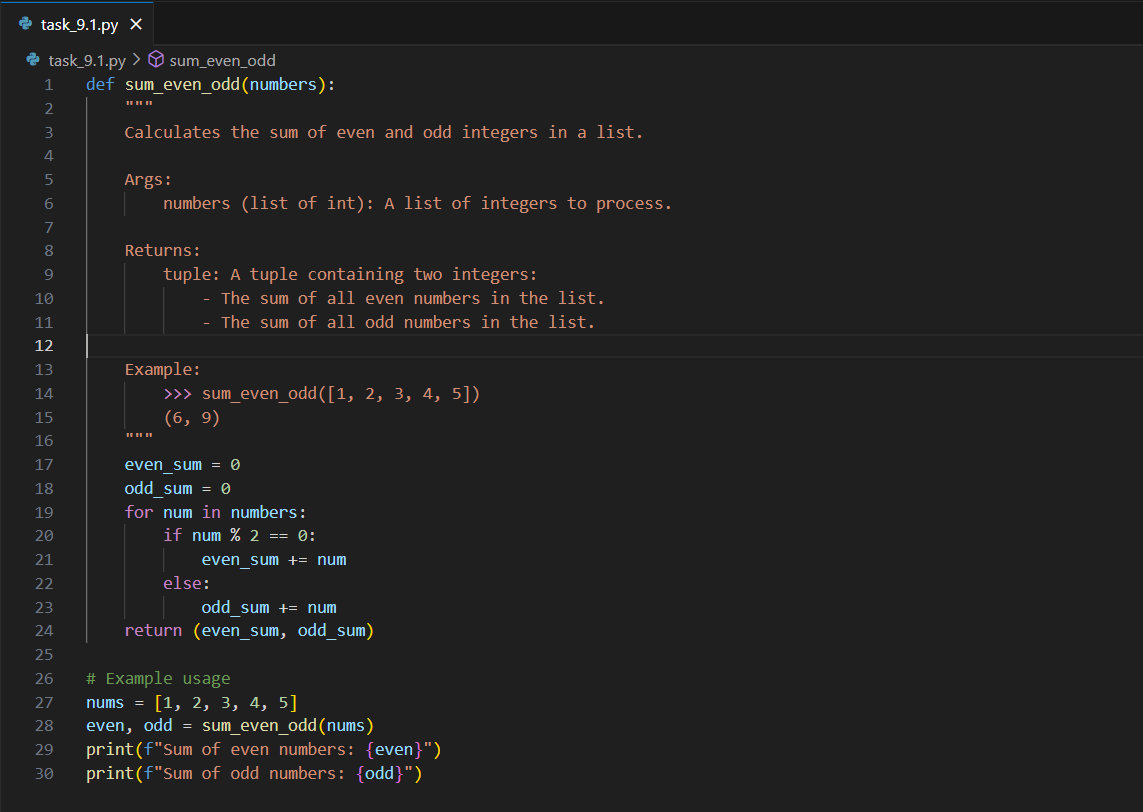
****

**Explanation from gemini :**

The code in the cell you pointed out defines a Python function called sum\_even\_odd that takes a list of integers as input. It then iterates through the list, adding even numbers to even\_sum and odd numbers to odd\_sum. Finally, it returns a tuple containing the calculated even\_sum and odd\_sum. The example usage below the function demonstrates how to call the function with a list of numbers and prints the original list, the sum of even numbers, and the sum of odd numbers.

* **Function Definition:** The code defines a function sum\_even\_odd that takes a list of integers (numbers) and returns a tuple of two integers (the sum of even and odd numbers).
* **Initialization:** It initializes two variables, even\_sum and odd\_sum, to 0.
* **Iteration:** It loops through each number in the input numbers list.
* **Even/Odd Check:** Inside the loop, it checks if a number is even using the modulo operator (%). If the remainder when divided by 2 is 0, the number is even; otherwise, it's odd.
* **Summation:** It adds the number to the respective sum (even\_sum or odd\_sum) based on whether it's even or odd.
* **Return Value:** After the loop finishes, the function returns a tuple containing the final even\_sum and odd\_sum.
* **Example Usage:** The code then shows an example of how to use the function with a sample list [1, 2, 3, 4, 5, 6, 7, 8, 9, 10].
* **Output:** Finally, it prints the original list and the calculated sums of even and odd numbers.

**Program in Copilot:**



**Output in Copilot:**

**A black background with white text

AI-generated content may be incorrect.**

**Explanation from Copilot:**

This code defines a function [sum\_even\_odd](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) that takes a list of integers and returns a tuple with the sum of even numbers and the sum of odd numbers.

How it works:

* + The function initializes two variables, [even\_sum](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) and [odd\_sum](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html), to zero.
  + It loops through each number in the input list:
    - If the number is even ([num % 2 == 0](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ")), it adds it to [even\_sum](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html).
    - If the number is odd, it adds it to [odd\_sum](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html).
  + After the loop, it returns a tuple (even\_sum, odd\_sum).

Example usage:

* + For the list [1, 2, 3, 4, 5], the function returns (6, 9) because:
    - Even numbers: 2 + 4 = 6
    - Odd numbers: 1 + 3 + 5 = 9

The results are printed:

Sum of even numbers: 6

Sum of odd numbers: 9

**Comparison between Gemini and Copilot:**

Here’s is the comparison :

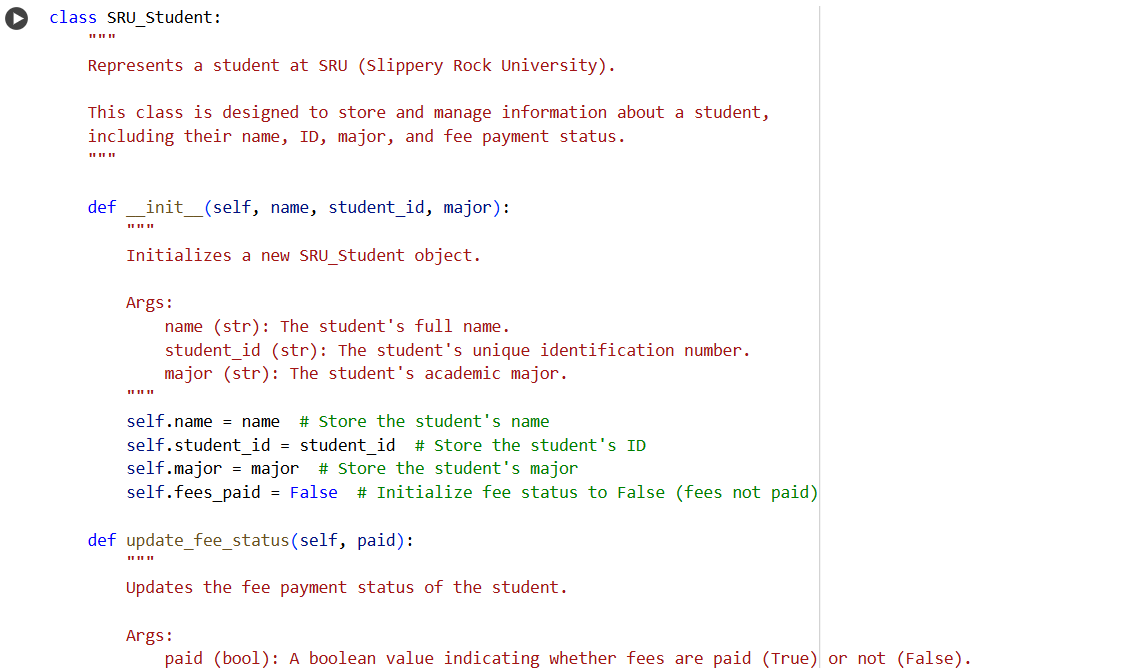
| **Aspect** | **Your Code (**[**task\_9.1.py**](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**)** | **Prompt Code** |
| --- | --- | --- |
| **Function Name** | [**sum\_even\_odd**](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) | [**sum\_even\_odd**](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) |
| **Type Hints** | **No type hints** | **Uses type hints:**[**numbers: list**](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**and -> tuple** |
| **Docstring Style** | **NumPy-style docstring with example** | **Standard docstring, no example** |
| **Parameters** | [**numbers**](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**(list of int)** | [**numbers**](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**(list of int, implied by type hint)** |
| **Return Value** | **Tuple: (sum of even, sum of odd)** | **Tuple: (sum of even, sum of odd)** |
| **Variable Names** | [**even\_sum**](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**,**[**odd\_sum**](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**, [num](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ")** | [**even\_sum**](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**,**[**odd\_sum**](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)**, number** |
| **Example Usage** | **Uses [1, 2, 3, 4, 5] and prints results** | **Uses [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] and prints results, also prints the list itself** |
| **Output** | **Prints sum of even and odd numbers** | **Prints the list, sum of even, and sum of odd numbers** |
| **Extra Features** | **Includes a docstring example with doctest** | **Includes type hints and prints the list before the sums** |

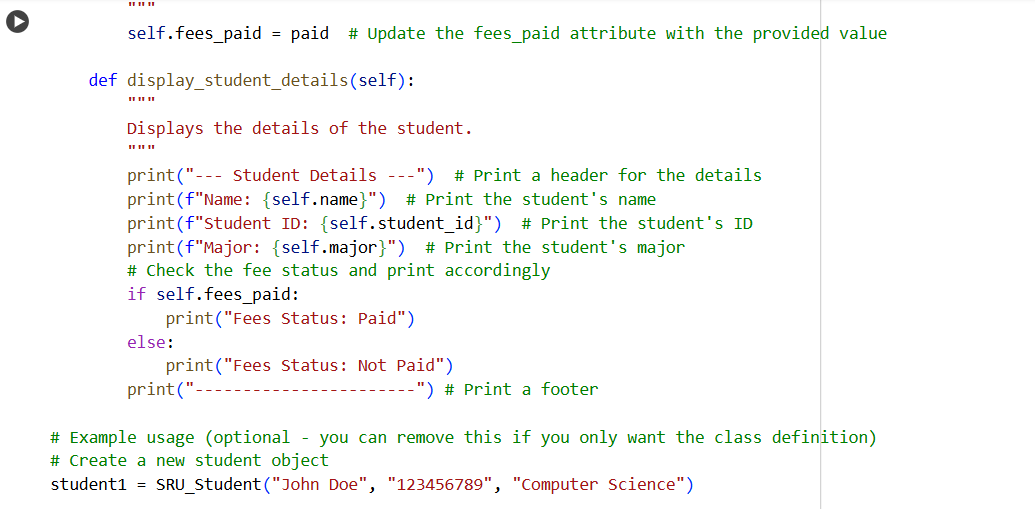
**Summary:**

* Both codes implement the same logic and return the same type of result.
* The prompt code uses type hints and prints the input list.
* Your code uses a more detailed docstring with an example (doctest style).
* Both are correct and clear; the differences are mainly in style and presentation.

**TASK-2:** Add detailed inline comments to this Python class that defines an SRU\_Student with methods to update fee status and display student details. Explain each line or logical block clearly for beginners.

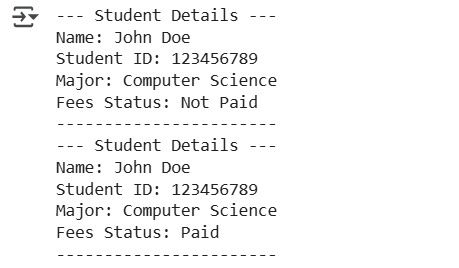
**Program in gemini:**

****

****

****

**Output in gemini:**

****

**Explanation from gemini:**

The code defines a class named SRU\_Student that represents a student at Slippery Rock University (SRU). This class stores information about the student such as their name, student ID, major, and whether they have paid their fees.

1. The class starts with a description explaining its purpose.
2. The \_\_init\_\_ method is called when a new student object is created. It takes three pieces of information: the student’s name, their student ID, and their major. It also sets a fees\_paid attribute to False by default, meaning the student hasn't paid their fees yet.
3. The update\_fee\_status method is used to change the fee payment status of the student. It accepts a Boolean value (True or False). If it’s True, the student has paid their fees; if False, they haven’t.
4. The display\_student\_details method prints the student’s information in a readable format. It shows the student’s name, ID, major, and whether their fees are paid or not.
5. At the bottom, there is an example of how to create a student object named student1 with specific details. It first displays the student’s information (which will show fees as unpaid), then updates the fee status to paid, and finally displays the updated information.

In summary, this class is a simple way to store and manage student information, especially their fee payment status, and print their details clearly.

**Program in Copilot:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Output in Copilot:**

A black screen with white text

AI-generated content may be incorrect.

**Explanation from Copilot:**

* The [SRU\_Student](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) class is a blueprint for creating student objects with specific attributes and behaviors.
* The [\_\_init\_\_](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) method (constructor) initializes each student with a name, roll number, course, and fee paid.
* The attributes [self.name](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html), [self.roll\_no](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html), [self.course](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html), and [self.fee\_paid](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) store the student's information.
* The [update\_fee](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) method allows you to add a specified amount to the student's current fee payment and prints the updated total.
* The [display\_details](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) method prints all the stored details of the student (name, roll number, course, and fee paid).
* Example usage at the end shows how to create a student, display their details, update their fee, and display the updated details again.
* The code uses clear comments to explain each logical block, making it beginner-friendly.

**Comparison between Gemini and Copilot:**

Here are the main differences:

**1. Attributes and Naming**

* **Your code:**
  + Uses [name](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html), student\_id, major, and fees\_paid (a boolean for fee status).
* **Active file:**
  + Uses [name](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html), [roll\_no](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html), [course](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html), and [fee\_paid](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) (an integer for the amount paid).

**2. Fee Handling**

* **Your code:**
  + Tracks only whether the fees are paid (fees\_paid: bool).
  + Updates fee status with a boolean (update\_fee\_status).
* **Active file:**
  + Tracks the actual amount paid ([fee\_paid: int](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)).
  + Updates the fee by adding an amount ([update\_fee](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)).

**3. Method Names**

* **Your code:**
  + update\_fee\_status (sets paid/not paid)
  + display\_student\_details
* **Active file:**
  + [update\_fee](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) (adds to fee)
  + [display\_details](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)

**4. Output**

* **Your code:**
  + Prints a formatted header and footer, and shows "Fees Status: Paid" or "Not Paid".
* **Active file:**
  + Prints the amount of fee paid.

**5. Student Identification**

* **Your code:**
  + Uses student\_id (string).
* **Active file:**
  + Uses [roll\_no](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) (integer).

**6. Course/Major**

* **Your code:**
  + Uses major.
* **Active file:**
  + Uses [course](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html).

**Summary:**  
Your code is focused on whether the fee is paid (yes/no), while the active file tracks the actual amount paid. Attribute names and method names are also different, reflecting these design choices.

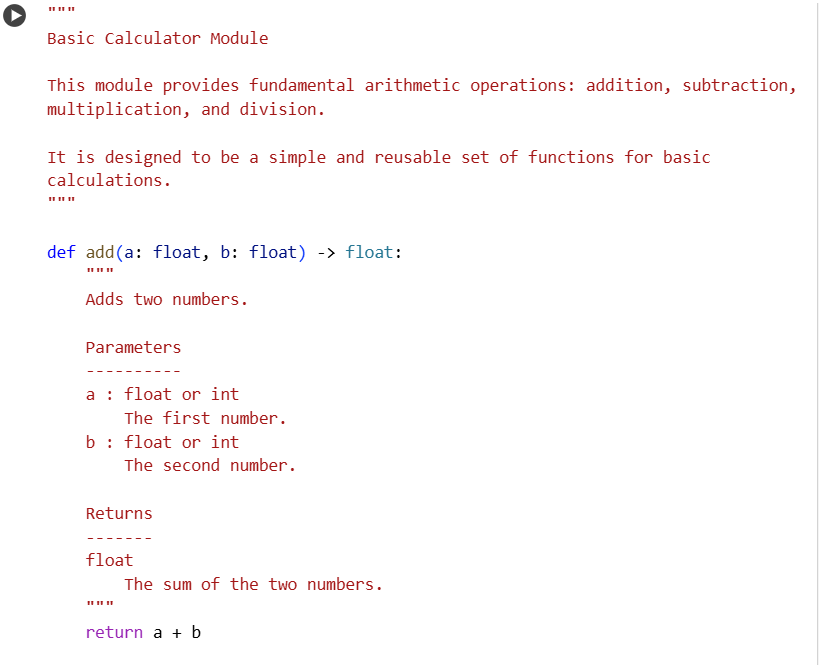
**TASK-3:** I have written a Python script with 3–4 basic calculator functions (add, subtract, multiply, divide).

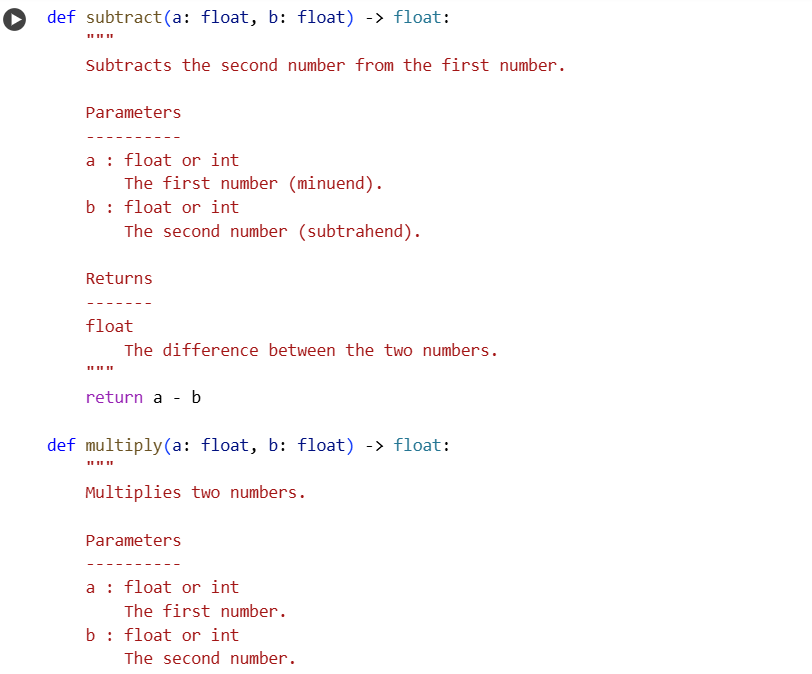
Please generate:

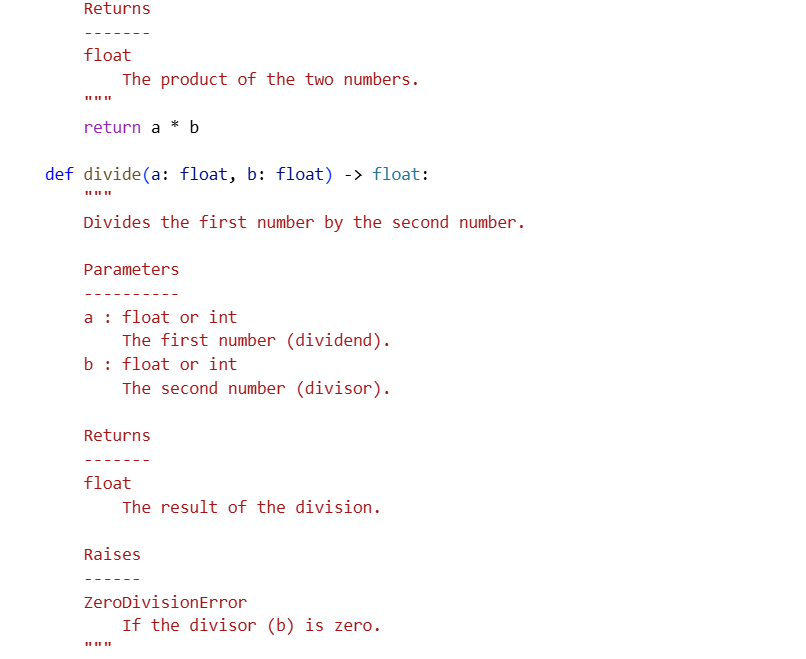
1. A module-level docstring
2. Individual function docstrings

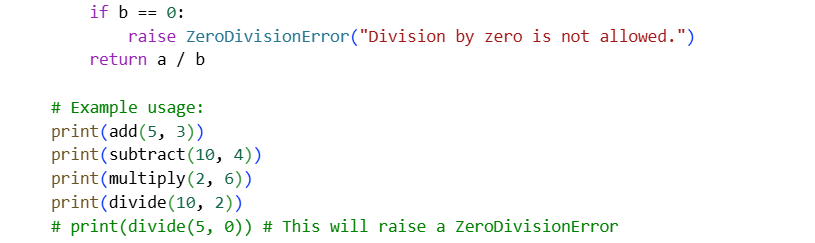
Use the NumPy docstring style.  
Each function takes two numerical inputs (int or float) and returns the result of the operation.  
Also include any relevant exceptions like ZeroDivisionError for division.

**Program in gemini:**

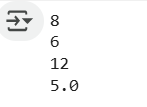
****

****

****

****

**Output in gemini:**

****

**Explanation from gemini:**

Here is an explanation of the code:

* **Module Docstring:** The code starts with a triple-quoted string, which is a module-level docstring. It provides a brief overview of the module's purpose: to provide basic arithmetic operations.
* **add(a, b) function:**
  + This function takes two arguments, a and b.
  + It returns the sum of a and b.
  + The docstring explains its purpose, parameters (a and b), and what it returns.
* **subtract(a, b) function:**
  + This function takes two arguments, a and b.
  + It returns the result of subtracting b from a.
  + The docstring explains its purpose, parameters (a and b), and what it returns.
* **multiply(a, b) function:**
  + This function takes two arguments, a and b.
  + It returns the product of a and b.
  + The docstring explains its purpose, parameters (a and b), and what it returns.
* **divide(a, b) function:**
  + This function takes two arguments, a and b.
  + It returns the result of dividing a by b.
  + It includes a check to see if b is zero. If it is, it raises a ZeroDivisionError to prevent division by zero.
  + The docstring explains its purpose, parameters (a and b), what it returns, and the Raises section highlights the ZeroDivisionError.
* **Example Usage:** The code includes commented-out lines demonstrating how to call each of the functions with example values and print their results. The line that would cause a ZeroDivisionError is also included but commented out.

**Program in Copilot:**

****

**Output in Copilot:**

**A black screen with white text

AI-generated content may be incorrect.**

**Explanation from Copilot:**

The code script is a simple calculator module with four arithmetic functions: [add](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html), [subtract](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html), [multiply](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html), and [divide](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html). Here’s a breakdown:

Module Docstring

* Describes the module’s purpose and lists the available functions.
* Follows the NumPy docstring style for clarity and documentation.

Functions

Each function:

* Takes two arguments ([a](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) and [b](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)), both can be int or float.
* Returns the result of the arithmetic operation.
* Has a detailed docstring explaining parameters, return type, and (for [divide](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)) possible exceptions.

Main Block

* Demonstrates usage with [x = 10](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) and [y = 5](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html).
* Prints the result of each operation.
* Shows how [divide](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) raises a [ZeroDivisionError](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) if the second argument is zero, and catches this error to print a friendly message.

Example Output

add(10, 5) = 15

subtract(10, 5) = 5

multiply(10, 5) = 50

divide(10, 5) = 2.0

Error: division by zero

Summary:  
This script is well-documented, easy to read, and demonstrates both normal operation and exception handling for division by zero.

**Comparison between Gemini and Copilot:**

Here are the main differences:

**1. Type Annotations**

* **Gemini code:** Uses type annotations ([a: float, b: float -> float](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)) for function parameters and return types.
* **Copilot code:** No type annotations.

**2. Division by Zero Handling**

* **Gemini code:** Explicitly checks for [b == 0](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) in [divide](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) and raises a custom error message.
* **Copilot code:** Relies on Python's default behavior, which raises [ZeroDivisionError](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) automatically.

**3. Docstrings**

* **Gemini code:** Docstrings are slightly reworded and more concise.
* **Copilot code:** Docstrings are more detailed and use a consistent format.

**4. Return Types**

* **Gemini code:** Always returns float (by annotation and docstring).
* **Copilot code:** Returns int or float depending on input.

**5. Example Usage**

* **Gemini code:** Only prints function results directly.
* **Copilot code:** Includes an [if \_\_name\_\_ == "\_\_main\_\_":](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) block with formatted output and error handling for division by zero.

**6. Module Docstring**

* **Gemini code:** Shorter, more concise.
* **Copilot code:** More detailed, includes a list of functions.

**Summary:**  
Your code is more type-annotated and explicit about division by zero, while [task\_9.3.py](vscode-file://vscode-app/c:/Users/sweety/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)(**Copilot code**) is more general, with richer documentation and example usage. Both provide the same basic calculator functionality.