

# AI ASSISTED CODING LAB 9.3

Lab 9: Documentation Generation: Automatic  
documentation and code comments

ENROLLMENT NO: 2403A52006

NAME: MOHAMMED MUSTAFA

COURSE: CSM(AI/ML)

BATCH: 01

DATE: 24-09-2025

## TASK DESCRIPTION:

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:

Generate a python code to take a number only list as input from the user and calculate the sum of the even and odd numbers separately

```
def sum_even_odd(numbers_list):  
    """  
    Calculates the sum of even and odd numbers in a list.  
  
    Args:  
        numbers_list: 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_list:  
        if number % 2 == 0:  
            even_sum += number  
        else:  
            odd_sum += number  
    return even_sum, odd_sum  
  
# Get input from the user  
input_str = input("Enter a list of numbers separated by spaces: ")  
  
# Convert the input string to a list of numbers  
try:  
    numbers = [int(x) for x in input_str.split()]  
except ValueError:  
    print("Invalid input. Please enter a list of numbers only.")  
    numbers = []
```

```
# Calculate and print the sum of even and odd numbers in a list
def sum_odd_even(numbers):
    even_sum, odd_sum = 0, 0
    for number in numbers:
        if number % 2 == 0:
            even_sum += number
        else:
            odd_sum += number
    return odd_sum, even_sum

# Example usage
list: numbers = [2, 8, 45, 67, 1, ...]
print(f"Sum of even numbers: {sum_odd_even(numbers)[1]}")
print(f"Sum of odd numbers: {sum_odd_even(numbers)[0]}")
```

Enter a list of numbers separated by spaces: 2 8 45 67 1 87 211 56  
Sum of even numbers: 66  
Sum of odd numbers: 411

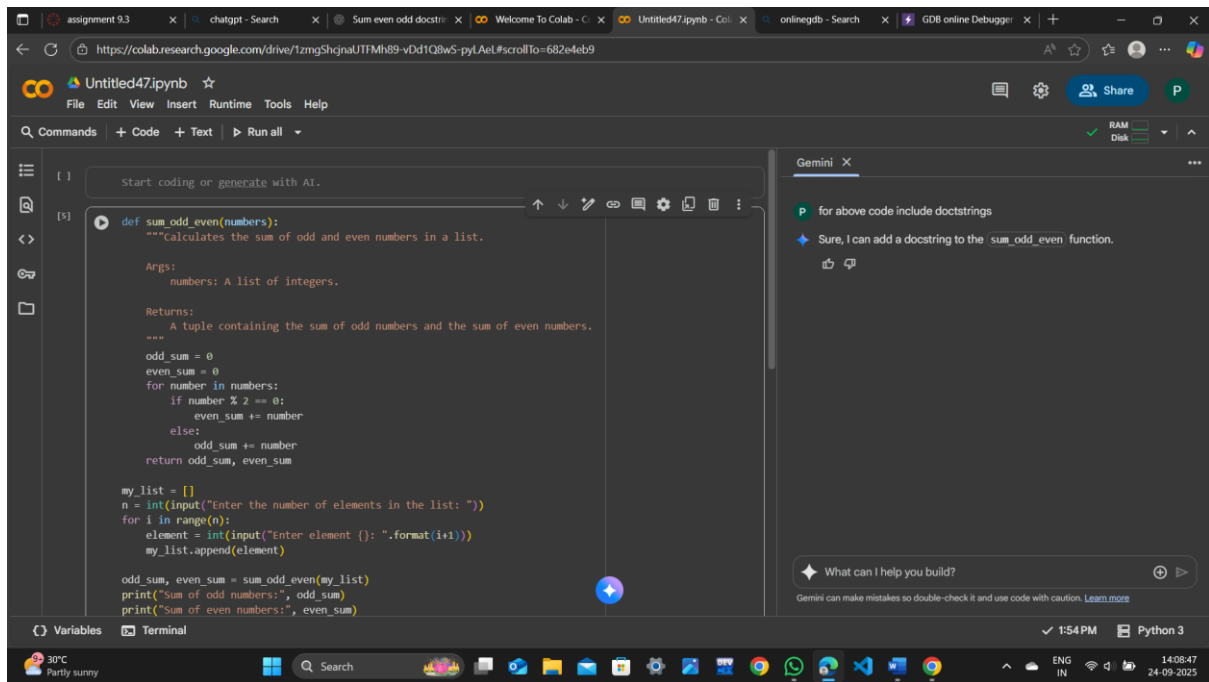
Incorporate manual docstring in code with Google Style

```
6     sum_odd_even(numbers) is a method that accepts a list of user defined numbers.
7
8     sum,odd are to store the sum of odd and even at the end
9     we are looping all the list elements and finds out the odd,even numbers based upon the condition i.e number%2==0 'EVEN' else 'ODD'
10    store the even number in even and odd in odd variable
11
12    At the end return those odd,even variables.
13    Out of the Function print them'''
14
15    odd_sum = 0
16    even_sum = 0
17    for number in numbers:
18        if number % 2 == 0:
19            even_sum += number
20        else:
21            odd_sum += number
22    return odd_sum, even_sum
23
24    my_list = []
25    n = int(input("Enter the number of elements in the list: "))
26    for i in range(n):
27        element = int(input("Enter element {}: ".format(i+1)))
28        my_list.append(element)
29
30    odd_sum, even_sum = sum_odd_even(my_list)
31    print("Sum of odd numbers:", odd_sum)
32    print("Sum of even numbers:", even_sum)
```

**Use an AI-assisted tool (e.g., Copilot, Cursor AI) to generate a docstring describing the function.**

**PROMPT:**

For the code that is been generated by You,Include docstrings in it



**OBSERVATION:**  
**Compare the AI-generated docstring with your manually written one.**

On Comparison of the Docstrings of mine and AI tool i.e Gemini. Mine is better and understandable by beginners, where as gemini it is a little bit terminology is involved.

## **TASK DESCRIPTION-2**

**Write python program for sru\_student class with attributes like name, roll no., hostel\_status and fee\_update method and display\_details method.**

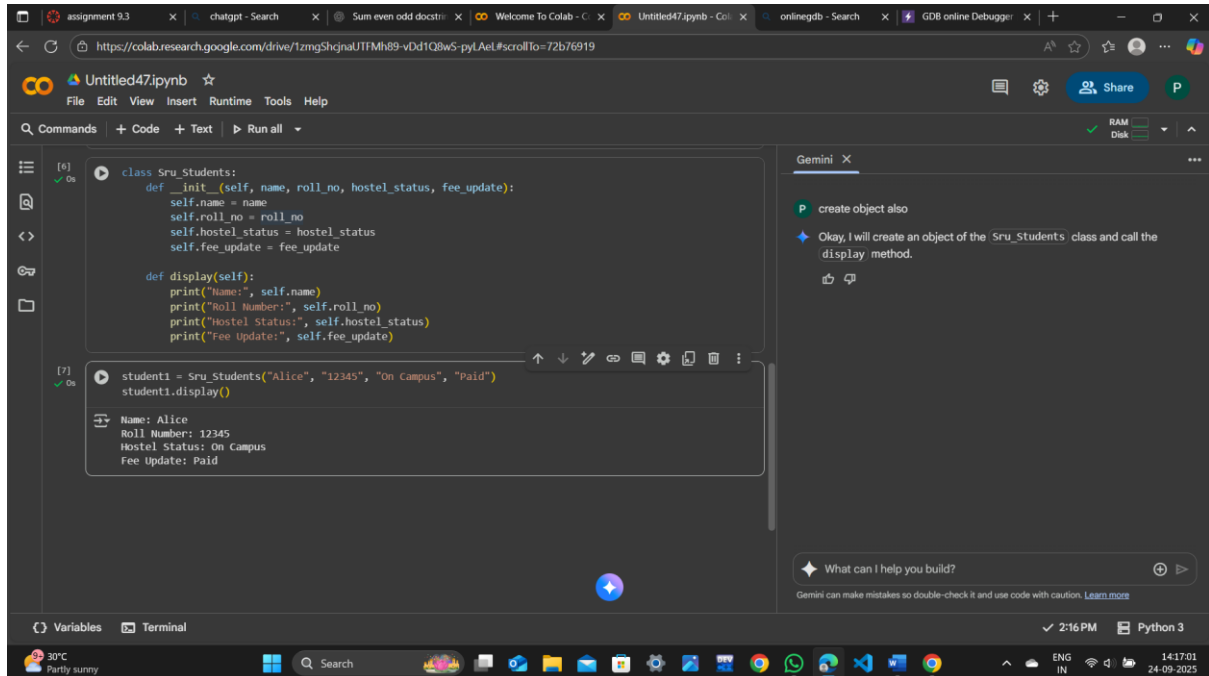
- Write comments manually for each line/code block
  - Ask an AI tool to add inline comments explaining each line/step.
- Compare the AI-generated comments with your manually written one.

Expected Output#2: Students critically analyze AI-generated code comments.

### **Prompt:**

Write a python class Sru\_Students with attributes like name,roll\_no,hostel\_status,fee\_update and a method 'display' that displays information without any docstrings.

# CODE:



```
[6] ✓ 0s
class Sru_Students:
    def __init__(self, name, roll_no, hostel_status, fee_update):
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee_update = fee_update

    def display(self):
        print("Name:", self.name)
        print("Roll Number:", self.roll_no)
        print("Hostel Status:", self.hostel_status)
        print("Fee Update:", self.fee_update)

[7] ✓ 0s
student1 = Sru_Students("Alice", "12345", "On Campus", "Paid")
student1.display()
```

Name: Alice  
Roll Number: 12345  
Hostel Status: On Campus  
Fee Update: Paid

Gemini ✕

P create object also

✦ Okay, I will create an object of the `Sru_Students` class and call the `display` method.

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Write comments manually for each line/code block

```

1 class Sru_Students:
2     def __init__(self, name, roll_no, hostel_status, fee_update):
3         # Initialize the attributes of the Sru_Students class
4         self.name = name
5         self.roll_no = roll_no
6         self.hostel_status = hostel_status
7         self.fee_update = fee_update
8
9     def display(self):
10        # Display information
11        print("Name:", self.name)
12        print("Roll Number:", self.roll_no)
13        print("Hostel Status:", self.hostel_status)
14        print("Fee Update:", self.fee_update)
15
16 o=Sru_Students('Deep',1,'Yes','paid')
17 o.display()

```

Ask an AI tool to add inline comments explaining each line/step

## PROMPT:

Add inline comments explaining each step for above code

The screenshot shows a Google Colab notebook with the following code and output:

```

class Sru_Students:
    def __init__(self, name, roll_no, hostel_status, fee_update):
        # Initialize the attributes of the Sru_Students class
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee_update = fee_update

    def display(self):
        # Display the student's information
        print("Name:", self.name)
        print("Roll Number:", self.roll_no)
        print("Hostel Status:", self.hostel_status)
        print("Fee Update:", self.fee_update)

# Create an object of the Sru_Students class
student1 = Sru_Students("Alice", "12345", "On Campus", "Paid")
# Call the display method to show the student's information
student1.display()

```

Output:

```

Name: Alice
Roll Number: 12345
Hostel Status: On Campus
Fee Update: Paid

```

Gemini Chat:

P add inline comments explaining each step for above code

Sure, I can add inline comments to explain each step in the Sru\_Students class definition and object creation.

## **OBSERVATION:**

Compare the AI-generated comments with your manually written one

I have Analysed the inline comments that are given by me and Gemini. What I observed is Gemini inline comments are far better than my comments. Since it has involved class ,Attributes in the code. So, it has better approach than me

## **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

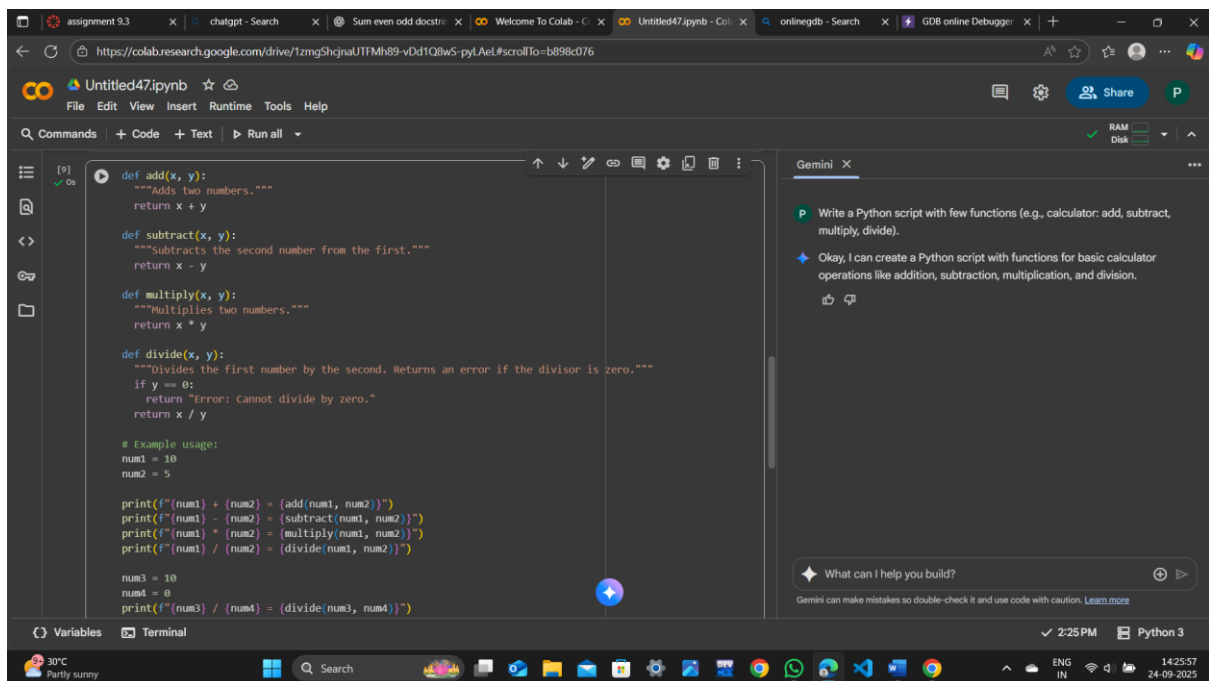


manually written one.

Expected Output#3: Students learn structured documentation for multi-function scripts

## PROMPT:

Write a Python script with few functions (e.g., calculator: add, subtract, multiply, divide).



The screenshot displays a Google Colab notebook environment. The main code cell contains a Python script with four functions: `add`, `subtract`, `multiply`, and `divide`. Each function is documented with a docstring. The `divide` function includes an error handling for division by zero. Below the functions, there is an example usage section that defines `num1` and `num2`, and prints the results of the four operations. The script also includes a final example using `num3` and `num4`. On the right side of the notebook, the Gemini chat interface is visible, showing a prompt to write a Python script with few functions and a response confirming the task.

```
def add(x, y):  
    """Adds two numbers."""  
    return x + y  
  
def subtract(x, y):  
    """Subtracts the second number from the first."""  
    return x - y  
  
def multiply(x, y):  
    """Multiplies two numbers."""  
    return x * y  
  
def divide(x, y):  
    """Divides the first number by the second. Returns an error if the divisor is zero."""  
    if y == 0:  
        return "Error: 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)}")  
  
num3 = 10  
num4 = 0  
print(f"{num3} / {num4} = {divide(num3, num4)}")
```

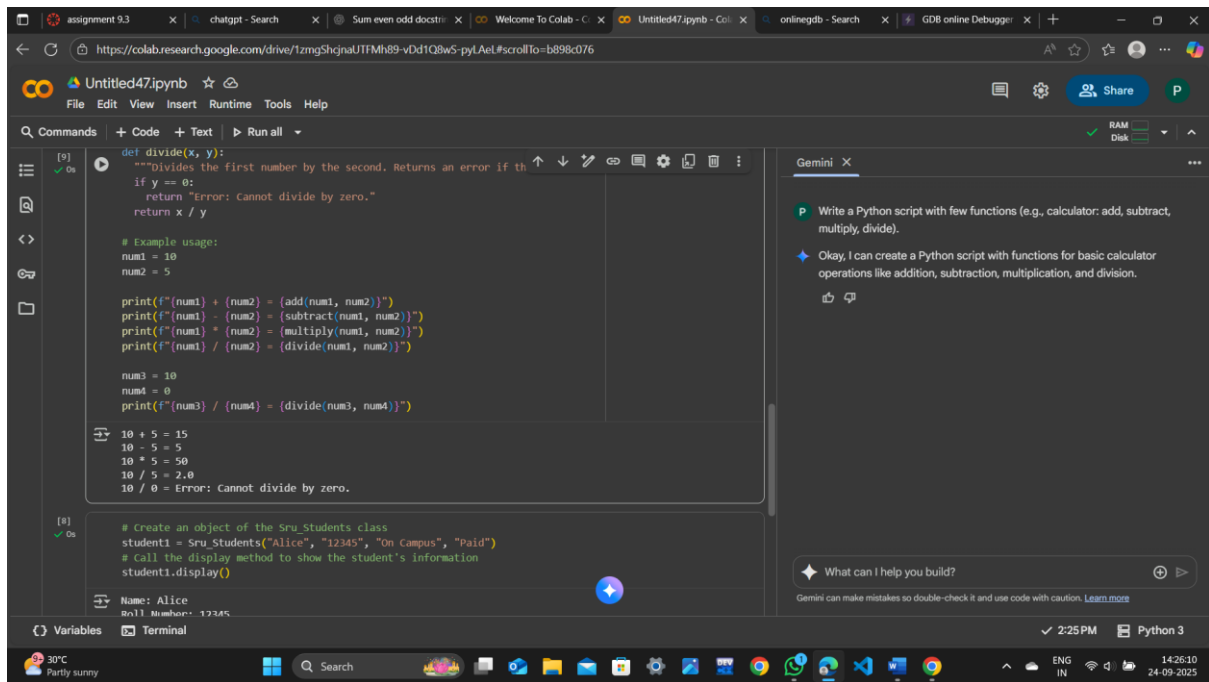
Gemini X

Write a Python script with few functions (e.g., calculator: add, subtract, multiply, divide).

Okay, I can create a Python script with functions for basic calculator operations like addition, subtraction, multiplication, and division.

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)



# Incorporate manual docstring in code with NumPy Style

## PROMPT:

For Above code incorporate manual docstring in code with NumPy Style

assignment 9.3 x chatgpt - Search x Sum even odd docstr x Welcome To Colab - C x Untitled47.ipynb - Col x onlinegdb - Search x GDB online Debugger x

https://colab.research.google.com/drive/1zm9Shj9naU7FMh89-vDd1Q8wS-pyLAeL#scrollTo=b898c076

Untitled47.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
[9] def add(x, y):  
    """Adds two numbers.  
  
    Parameters  
    -----  
    x : float or int  
        The first number.  
    y : float or int  
        The second number.  
  
    Returns  
    -----  
    float or int  
        The sum of the two numbers.  
    """  
    return x + y  
  
def subtract(x, y):  
    """Subtracts the second number from the first.  
  
    Parameters  
    -----  
    x : float or int  
        The first number.  
    y : float or int  
        The second number.  
  
    Returns  
    -----  
    float or int  
        The difference between the two numbers.  
    """  
    return x - y
```

Gemini

For above code incorporate manual docstring in code with NumPy Style

Sure, I can add docstrings to the calculator functions in NumPy style.

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Variables Terminal

30°C Partly sunny

Search

2:25 PM Python 3

14:27:49 24-09-2025

assignment 9.3 x chatgpt - Search x Sum even odd docstr x Welcome To Colab - C x Untitled47.ipynb - Col x onlinegdb - Search x GDB online Debugger x

https://colab.research.google.com/drive/1zm9Shj9naU7FMh89-vDd1Q8wS-pyLAeL#scrollTo=b898c076

Untitled47.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
[9] def subtract(x, y):  
    """Subtracts the second number from the first.  
  
    Parameters  
    -----  
    x : float or int  
        The first number.  
    y : float or int  
        The second number.  
  
    Returns  
    -----  
    float or int  
        The difference between the two numbers.  
    """  
    return x - y  
  
def multiply(x, y):  
    """Multiplies two numbers.  
  
    Parameters  
    -----  
    x : float or int  
        The first number.  
    y : float or int  
        The second number.  
  
    Returns  
    -----  
    float or int  
        The product of the two numbers.  
    """  
    return x * y
```

Gemini

For above code incorporate manual docstring in code with NumPy Style

Sure, I can add docstrings to the calculator functions in NumPy style.

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Variables Terminal

30°C Partly sunny

Search

2:25 PM Python 3

14:28:01 24-09-2025

```
def divide(x, y):  
    """Divides the first number by the second.  
  
    Parameters  
    -----  
    x : float or int  
        The numerator.  
    y : float or int  
        The denominator.  
  
    Returns  
    -----  
    float or int or str  
        The quotient of the two numbers, or an error message if the denominator is zero.  
    """  
    if y == 0:  
        return "Error: 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)}")  
  
num3 = 10  
num4 = 0  
print(f"{num3} / {num4} = {divide(num3, num4)}")
```

Gemini X

P For above code incorporate manual docstring in code with NumPy Style

Sure, I can add docstrings to the calculator functions in NumPy style.

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

```
    """  
    The quotient of the two numbers, or an error message if the d  
    if y == 0:  
        return "Error: 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)}")  
  
num3 = 10  
num4 = 0  
print(f"{num3} / {num4} = {divide(num3, num4)}")  
  
10 + 5 = 15  
10 - 5 = 5  
10 * 5 = 50  
10 / 5 = 2.0  
10 / 0 = Error: Cannot divide by zero.  
  
[8]  
# Create an object of the Sru_Students class  
student1 = Sru_Students("Alice", "12345", "On Campus", "Paid")  
# Call the display method to show the student's information  
student1.display()  
  
Name: Alice
```

Gemini X

P For above code incorporate manual docstring in code with NumPy Style

Sure, I can add docstrings to the calculator functions in NumPy style.

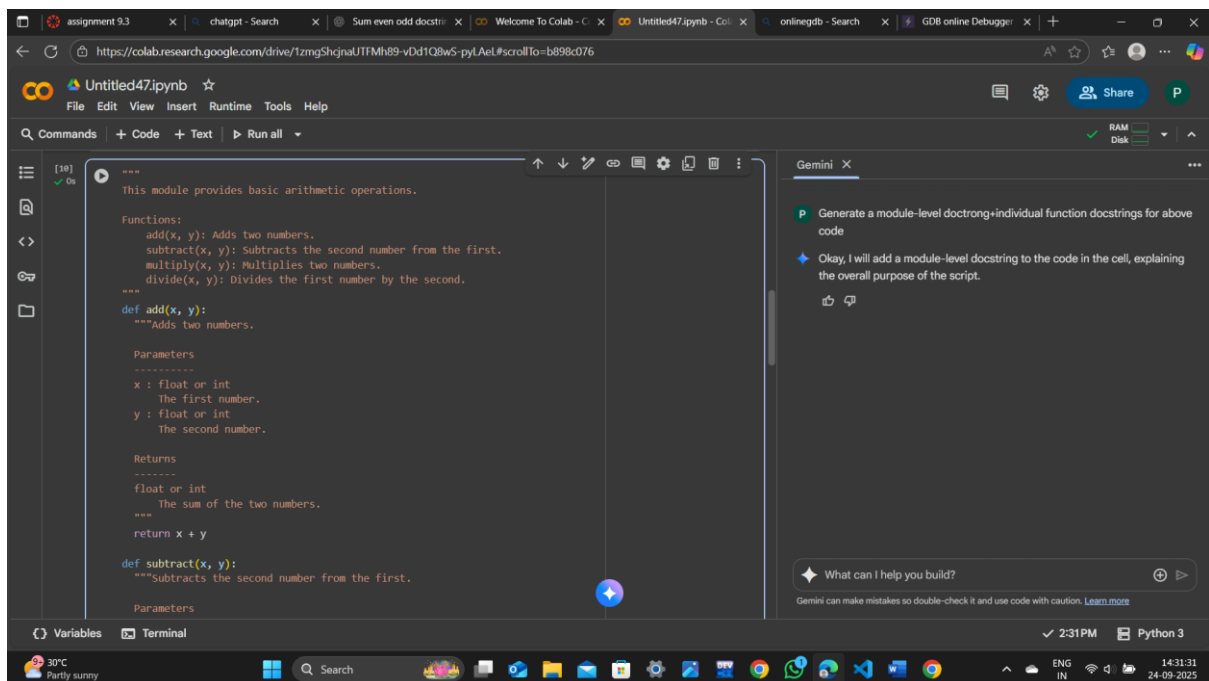
What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

**Use AI assistance to generate a module-level docstring + individual function docstrings.**

# PROMPT:

**Generate a module-level  
doctring+individual function docstrings for  
above code**



assignment 9.3 | chatgpt - Search | Sum even odd doctri... | Welcome To Colab - C... | Untitled47.ipynb - Col... | onlinedb - Search | GDB online Debugger | +

https://colab.research.google.com/drive/1zmgShjnaUTFMh89-vDd1Q8wS-pyLAeL#scrollTo=b898c076

Untitled47.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
[18] ✓ 0s
def subtract(x, y):
    """Subtracts the second number from the first.

    Parameters
    -----
    x : float or int
        The first number.
    y : float or int
        The second number.

    Returns
    -----
    float or int
        The difference between the two numbers.
    """
    return x - y

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

    Parameters
    -----
    x : float or int
        The first number.
    y : float or int
        The second number.

    Returns
    -----
    float or int
        The product of the two numbers.
    """
```

Gemini X

P Generate a module-level doctstring+individual function doctstrings for above code

Okay, I will add a module-level doctstring to the code in the cell, explaining the overall purpose of the script.

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Variables Terminal

2:31 PM Python 3

30°C Partly sunny

Search

14:32:23 24-09-2025

assignment 9.3 | chatgpt - Search | Sum even odd doctri... | Welcome To Colab - C... | Untitled47.ipynb - Col... | onlinedb - Search | GDB online Debugger | +

https://colab.research.google.com/drive/1zmgShjnaUTFMh89-vDd1Q8wS-pyLAeL#scrollTo=b898c076

Untitled47.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
[18] ✓ 0s
def divide(x, y):
    """Divides the first number by the second.

    Parameters
    -----
    x : float or int
        The numerator.
    y : float or int
        The denominator.

    Returns
    -----
    float or int or str
        The quotient of the two numbers, or an error message if the denominator is zero.
    """
    if y == 0:
        return "Error: 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)}")

num3 = 10
num4 = 0
print(f"{num3} / {num4} = {divide(num3, num4)}")
```

Gemini X

P Generate a module-level doctstring+individual function doctstrings for above code

Okay, I will add a module-level doctstring to the code in the cell, explaining the overall purpose of the script.

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

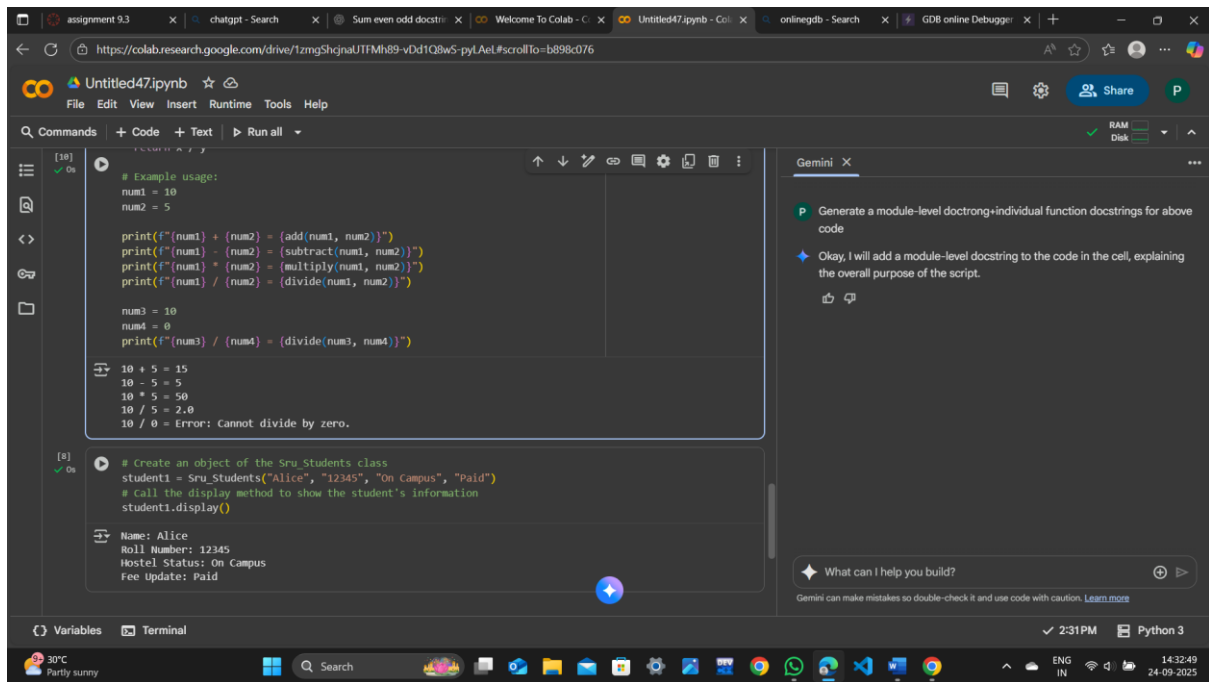
Variables Terminal

2:31 PM Python 3

30°C Partly sunny

Search

14:32:36 24-09-2025



# Compare the AI-generated docstring with your manually written one

## OBSERVATION:

Both docstrings look good and have meaning as per the code. In my docstring it has simple English letters and words where as in Gemini, it has used some hard Terminology.

Apart from that Everything is good when we compare the docstrings .Externally We took help to get module level docstring from GEMINI.

