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

```python
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 th
if numbers:
    even_sum, odd_sum =
    print(f"Sum of even numbers: {even_sum}")
    print(f"Sum of odd numbers: {odd_sum}")
```

list: numbers

(8 items) [2, 8, 45, 67, 1, ...]

```
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
```

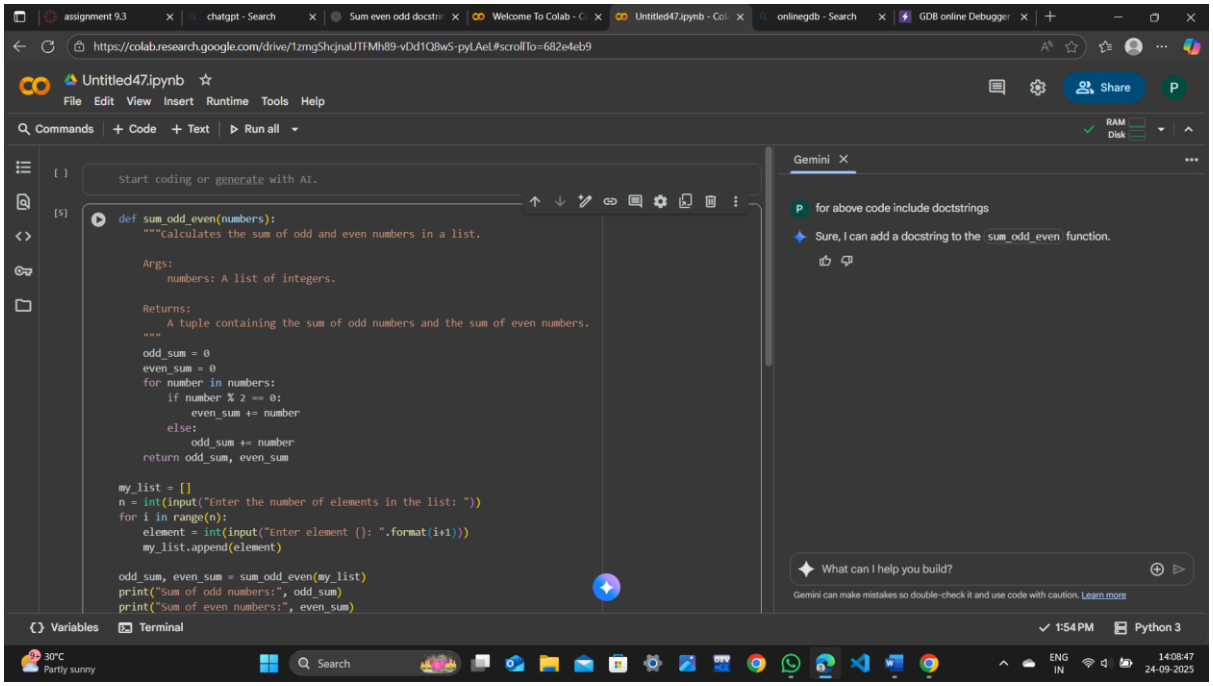
Incorporate manual docstring in code with Google Style

```
    sum_odd_even(numbers) is a method that accepts a list of user defined numbers.

    sum,odd are to store the sum of odd and even at the end
    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 'O
    store the even number in even and odd in odd variable

    At the end return those odd,even variables.
    Out of the Function print them'''

    odd_sum = 0
    even_sum = 0
    for number in numbers:
        if number % 2 == 0:
            even_sum += number
        else:
            odd_sum += number
    return odd_sum, even_sum

my_list = []
n = int(input("Enter the number of elements in the list: "))
for i in range(n):
    element = int(input("Enter element {}: ".format(i+1)))
    my_list.append(element)

odd_sum, even_sum = sum_odd_even(my_list)
print("Sum of odd numbers:", odd_sum)
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 Comparision of the Docstrings of mine and AI tool i.e Gemini.Mine is better and understandable by beginers,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.

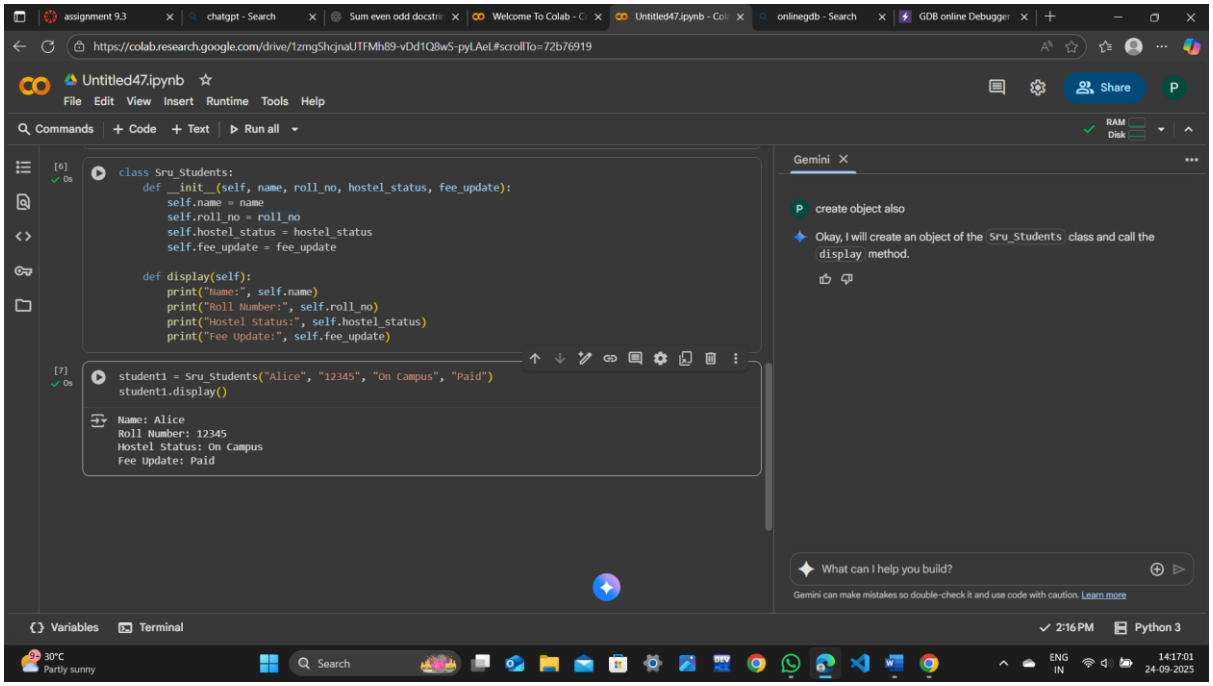
- Write comments manually for each line/code block
- Ask an AI tool to add inline comments explaining each line/step.
- Compare the AI-generated comments with your manually written one.

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:



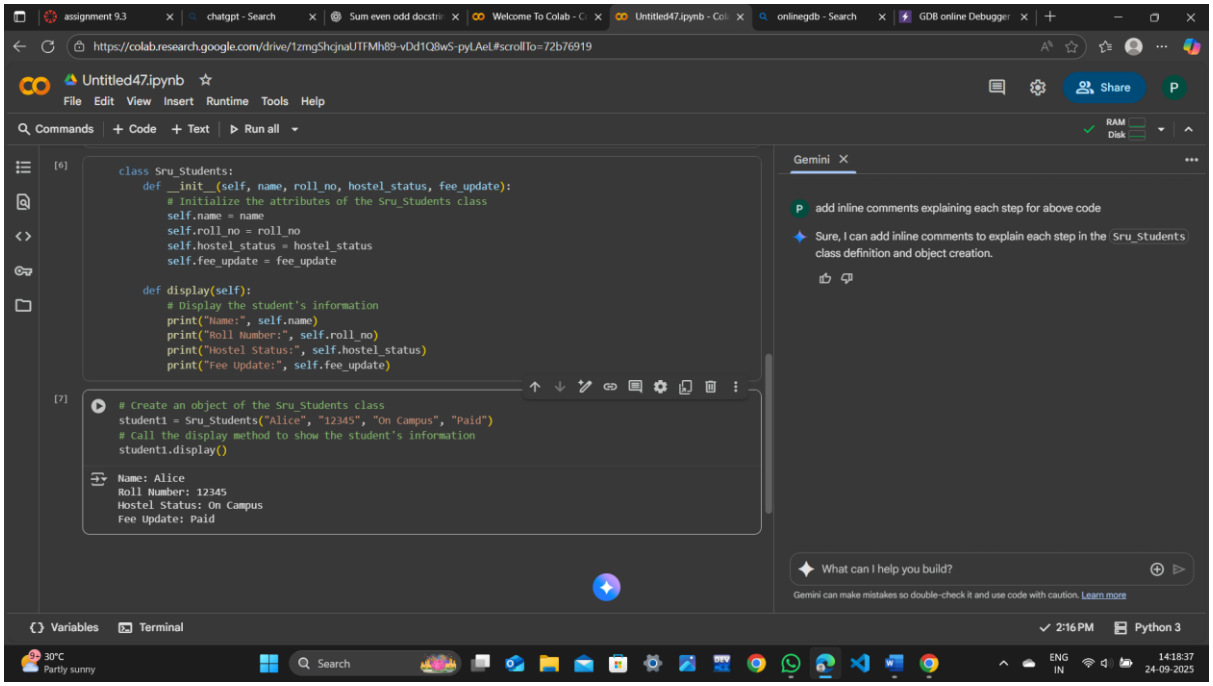
Write comments manually for each line/code block

```python
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 information
        print("Name:", self.name)
        print("Roll Number:", self.roll_no)
        print("Hostel Status:", self.hostel_status)
        print("Fee Update:", self.fee_update)

o=Sru_Students('Deep',1,'Yes','paid')
o.display()
```

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

PROMPT:

Add inline comments explaining each step for above code



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.

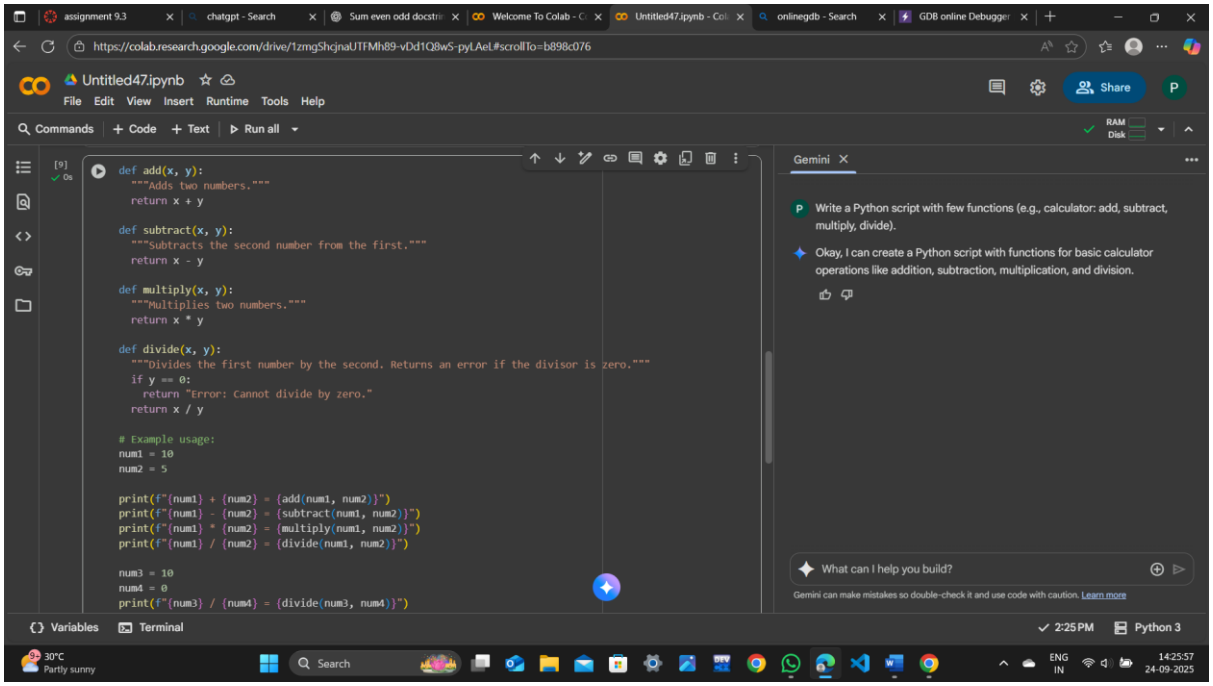
• Compare the AI-generated docstring with your

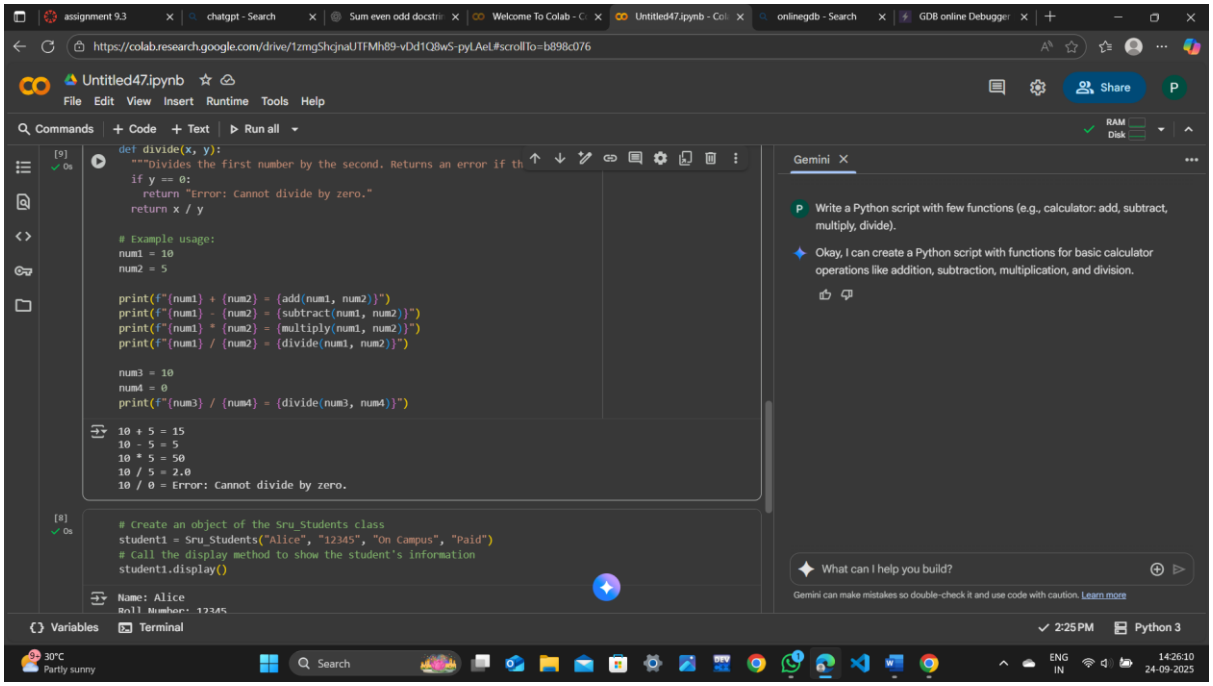
manually written one.

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).

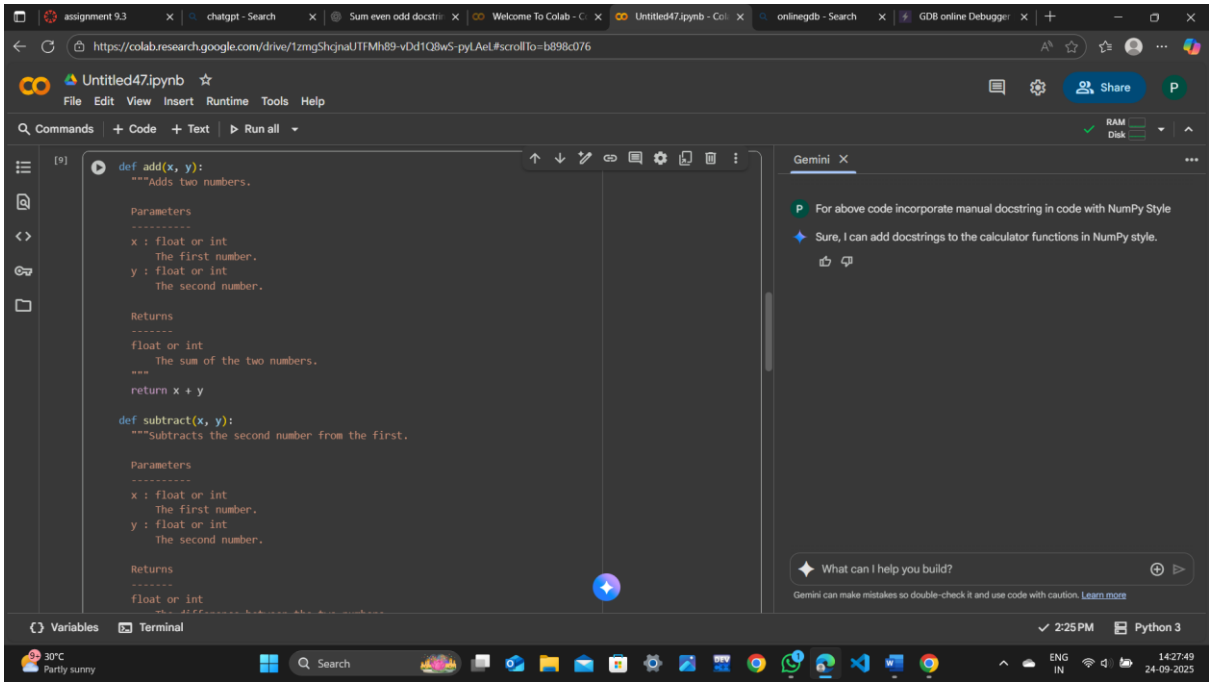


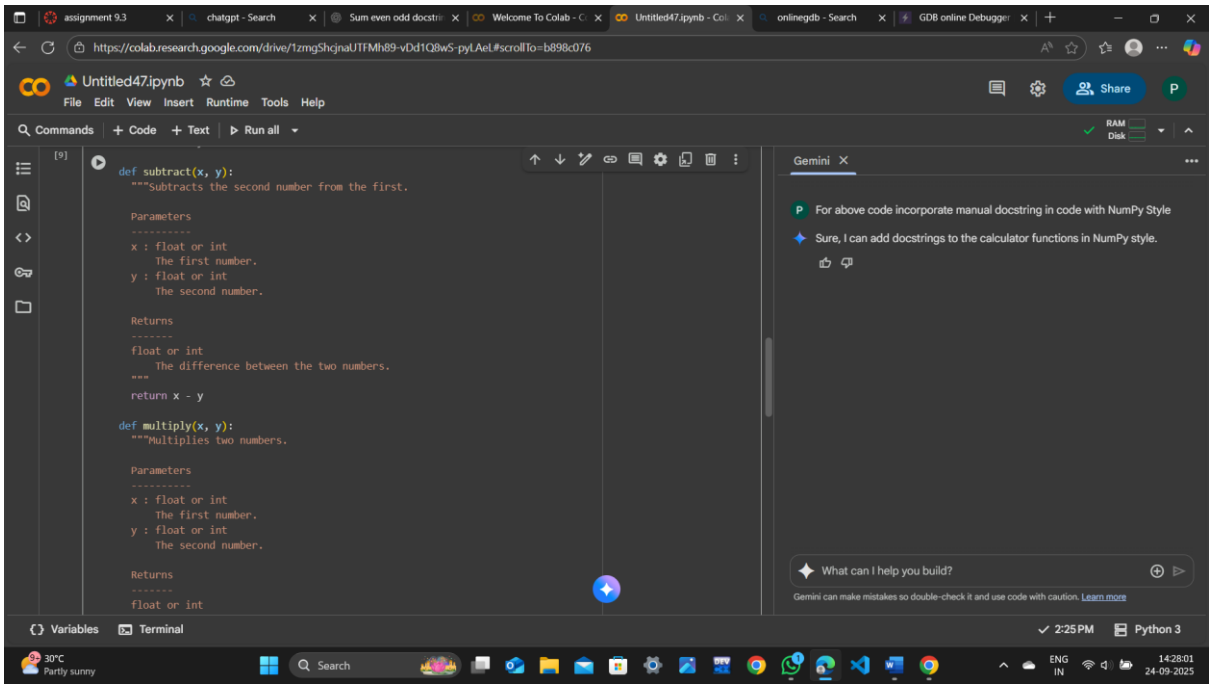


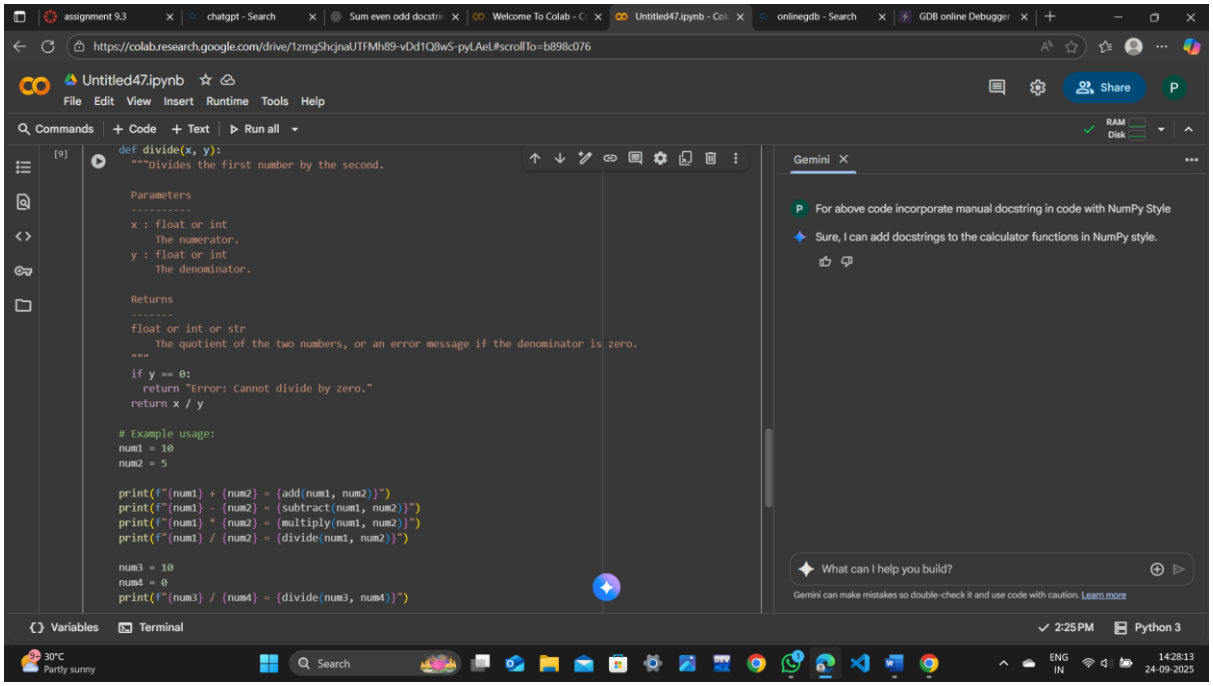
Incorporate manual docstring in code with NumPy Style

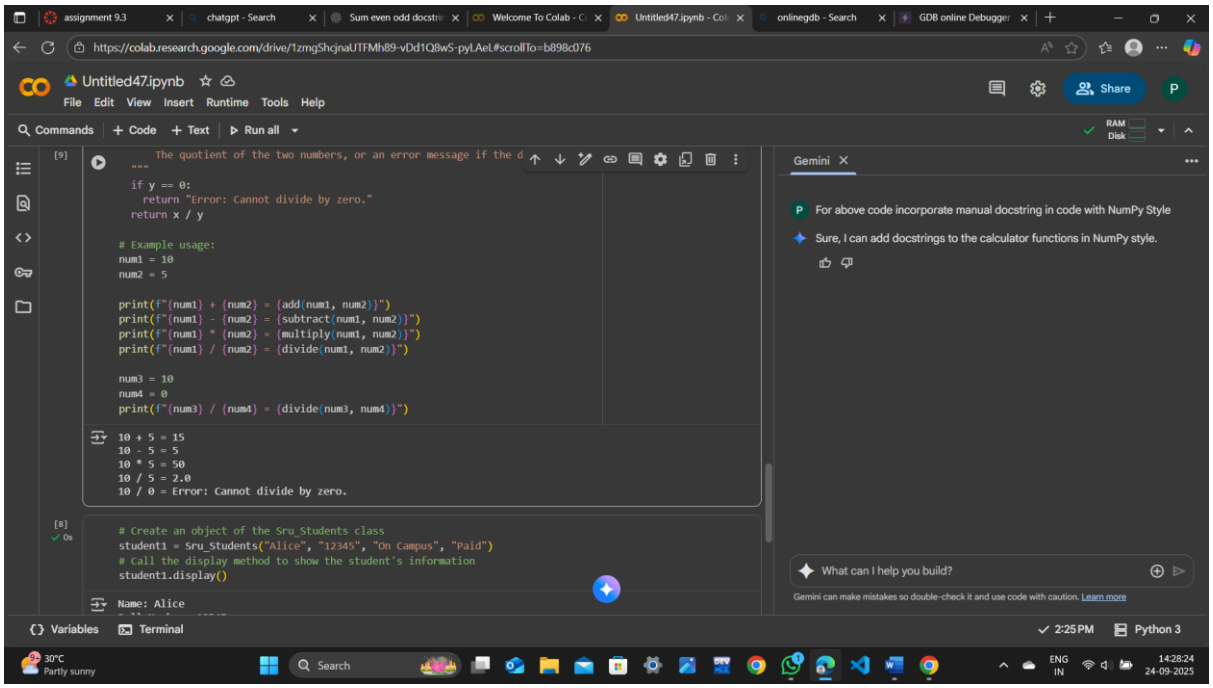
PROMPT:

For Above code incorporate manual docstring in code with NumPy Style


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
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
2:25 PM Python 3


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
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
2:25 PM Python 3

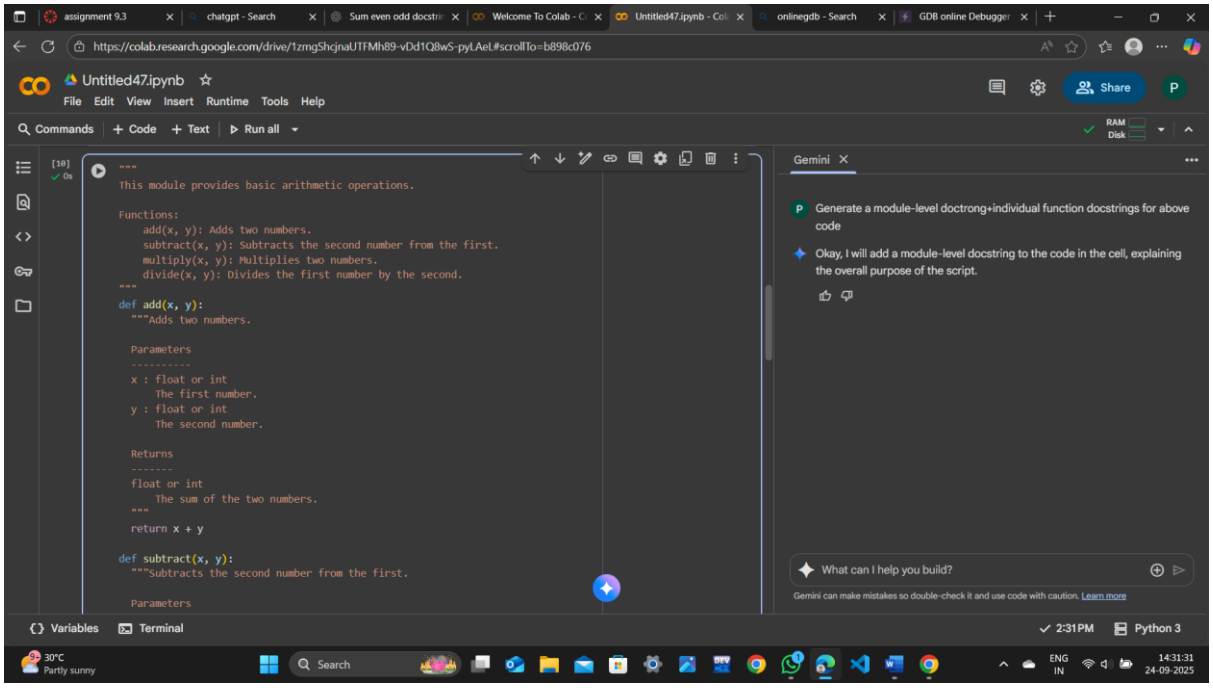




Use AI assistance to generate a module-level docstring + individual function docstrings.

PROMPT:

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



```python
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

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

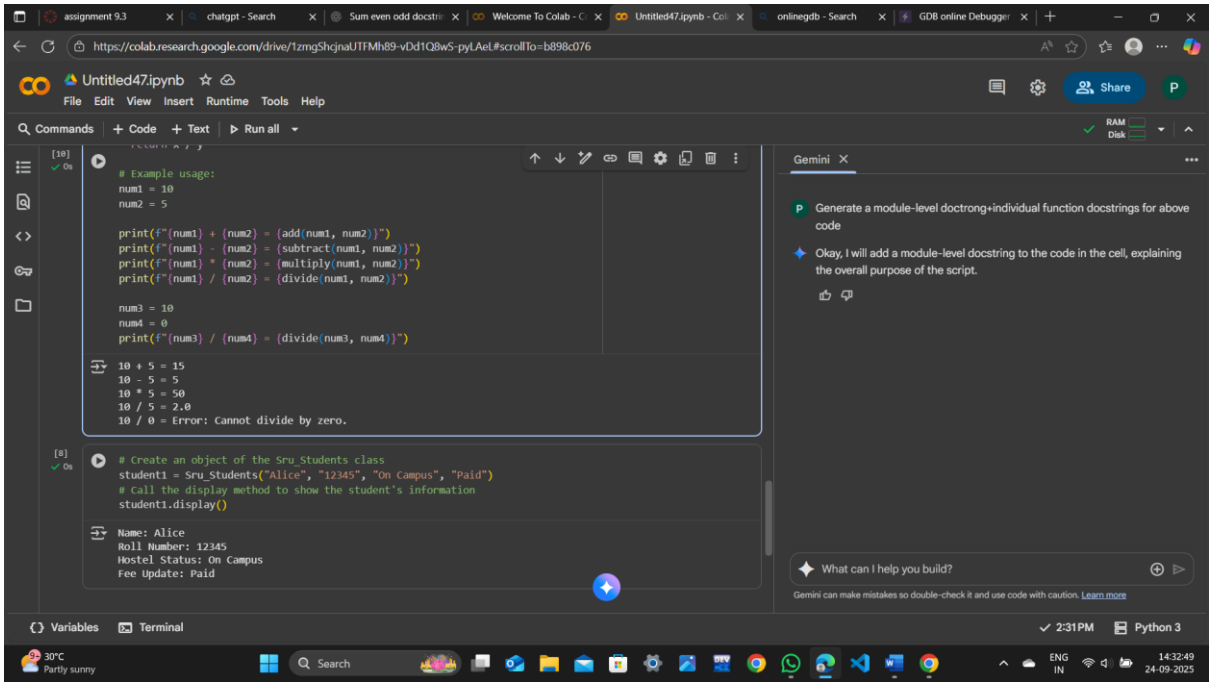
Okay, I will add a module-level docstring to the code in the cell, explaining the overall purpose of the script.

```python
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

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

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



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.