ASSIGNMENT-9

DOCSTRING AND INLINE COMMENTS

HTNO:2403A51319

BATCH:15

NAME:ANKITHA

**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., Gemini, Copilot, Cursor AI) to generate a docstring describing the function.
* Compare the AI-generated docstring with your manually written one.

**Expected Outcome#1:** Students understand how AI can produce function-level documentation

PROMPT:

write a python program to calculate the sum of even and odd numbers in te list of 1 to 10 numbers without docstring and inline comments.

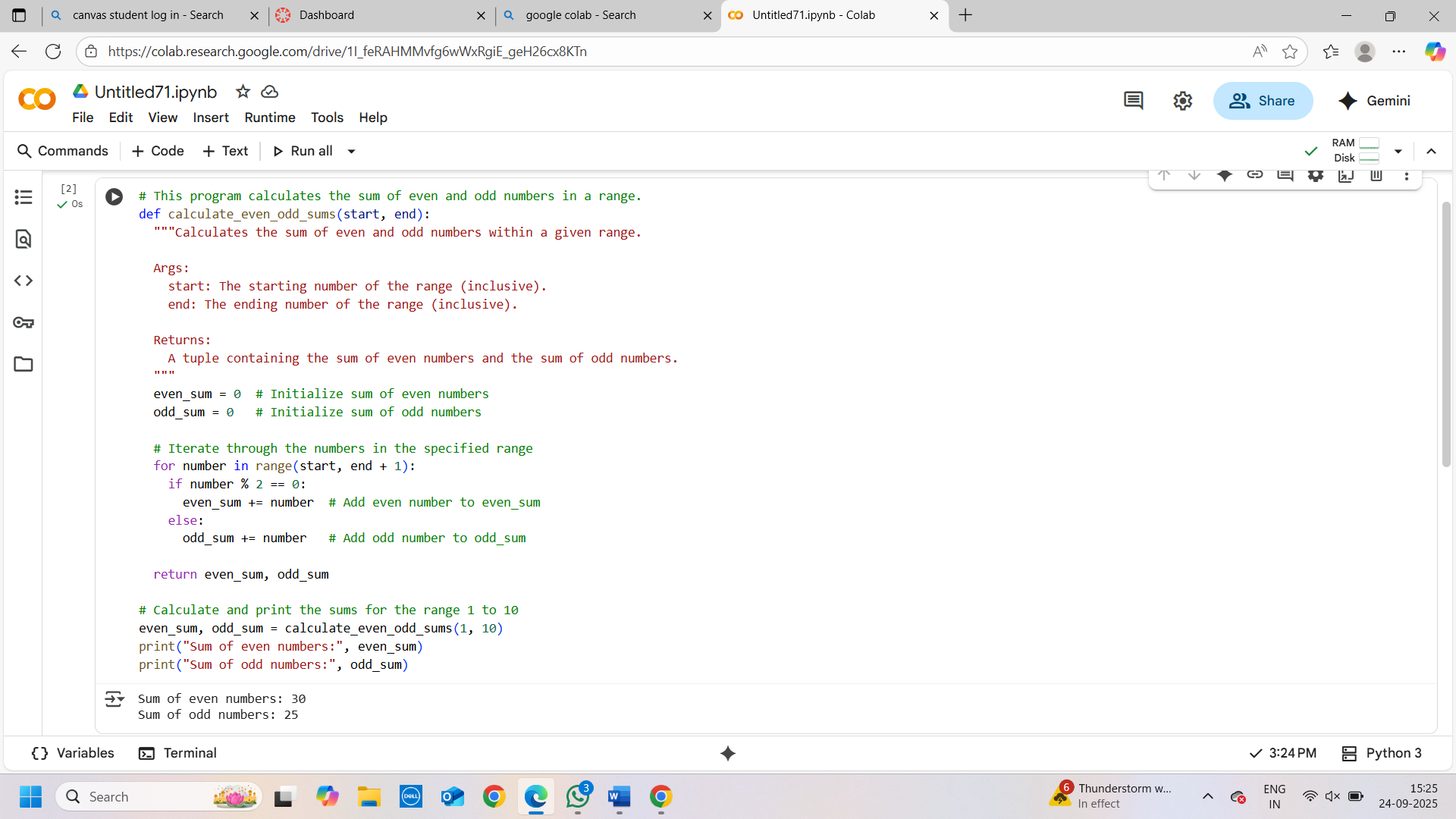
CODE:

A screenshot of a computer code

AI-generated content may be incorrect.

PROMPT:   
please add docstring and inline comments.

CODE:



OUTPUT:



EXPLANATION:

1. **Initialization:** Inside the function, two variables, even\_sum and odd\_sum, are initialized to 0. These variables will store the sum of even and odd numbers, respectively.
2. **Iteration:** The code then uses a for loop to iterate through each number in the specified range (from start to end, inclusive).
3. **Checking for even or odd:** For each number in the range, it checks if the number is even or odd using the modulo operator (%). If the remainder when divided by 2 is 0, the number is even; otherwise, it's odd.
4. **Adding to sums:** If the number is even, it's added to even\_sum. If it's odd, it's added to odd\_sum.
5. **Returning sums:** After the loop finishes, the function returns both even\_sum and odd\_sum as a tuple.
6. **Calling the function and printing:** The code then calls the calculate\_even\_odd\_sums function with the range 1 to 10 and stores the returned values in the even\_sum and odd\_sum variables. Finally, it prints the calculated sums of even and odd numbers.

**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.

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

PROMPT:

Write a program in python to create a class name with sru\_student with attributes name, roll no., hostel\_status .without docstring and inline comments

CODE:

A computer screen shot of a number of black text

AI-generated content may be incorrect.

PROMPT:

please add fee\_update method and display\_details method

CODE:

A computer screen shot of a program

AI-generated content may be incorrect.

PROMPT:

please add 5 sample names and attributes details and display it

CODE:

A screenshot of a computer program

AI-generated content may be incorrect.

OUTPUT:

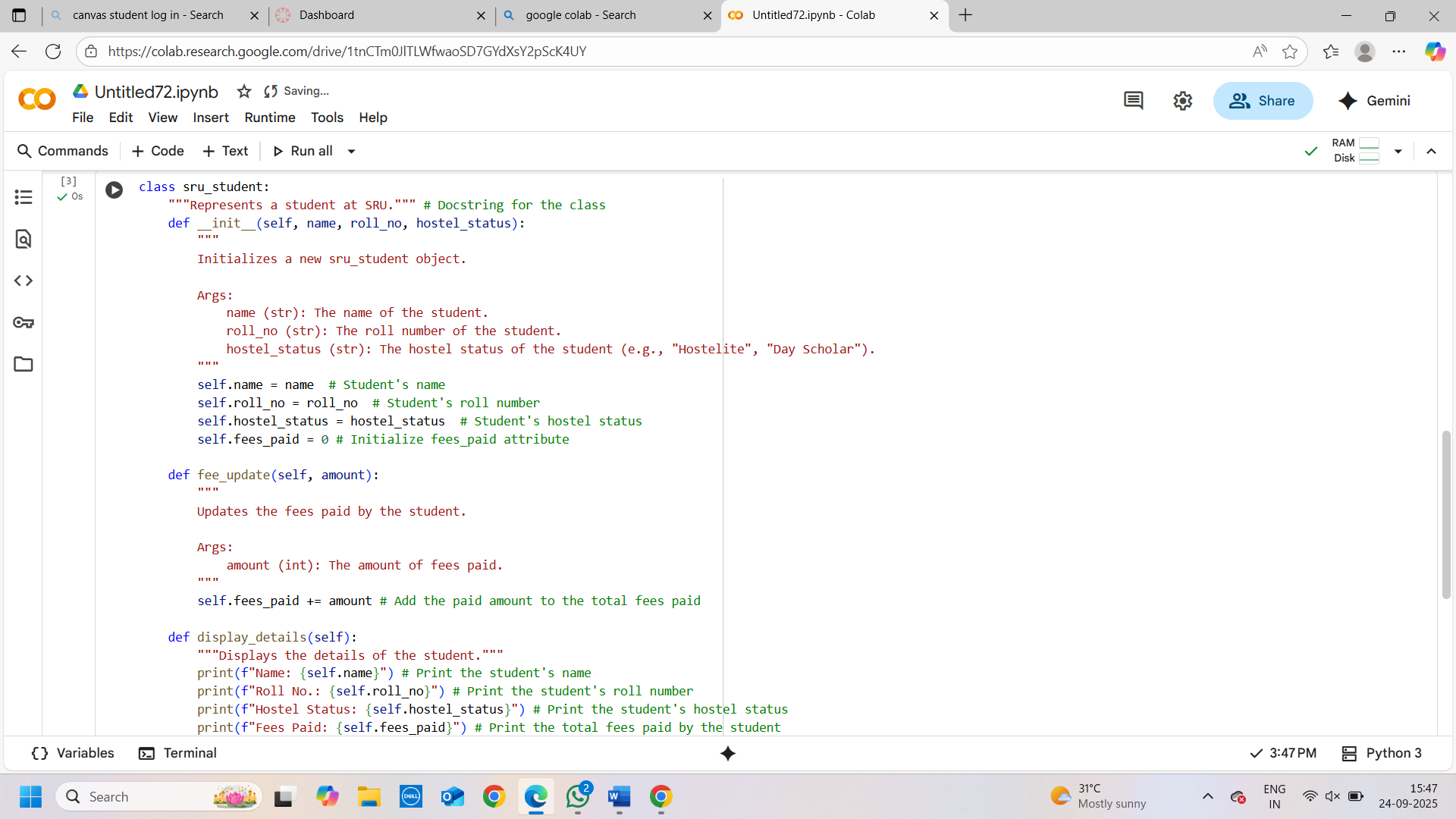
A screenshot of a computer

AI-generated content may be incorrect.

PROMPT:

please add docstring and inline comments

CODE:



EXPLANATION:

* **class sru\_student:**: This line defines the beginning of the class named sru\_student.
* **"""Represents a student at SRU."""**: This is a docstring, providing a brief description of what the class does.
* **def \_\_init\_\_(self, name, roll\_no, hostel\_status):**: This is the constructor method. It's called when you create a new sru\_student object.
  + **""" Initializes a new sru\_student object. ... """**: This is the docstring for the \_\_init\_\_ method, explaining its purpose and arguments.
  + **self.name = name**, **self.roll\_no = roll\_no**, **self.hostel\_status = hostel\_status**: These lines assign the values passed when creating an object to the object's attributes (properties).
  + **self.fees\_paid = 0**: This initializes the fees\_paid attribute to 0 for every new student object.
* **def fee\_update(self, amount):**: This method is used to update the fees paid by a student.
  + **""" Updates the fees paid by the student. ... """**: This is the docstring for the fee\_update method.
  + **self.fees\_paid += amount**: This line adds the amount passed to the method to the existing fees\_paid attribute of the student object.
* **def display\_details(self):**: This method is used to print the details of a student.
  + **"""Displays the details of the student."""**: This is the docstring for the display\_details method.
  + **print(f"Name: {self.name}")**, etc.: These lines use f-strings to print the values of the student's attributes.

The second code cell demonstrates how to use the sru\_student class:

* **student1 = sru\_student("Alice", "101", "Hostelite")**, etc.: These lines create 5 different sru\_student objects with specific names, roll numbers, and hostel statuses.
* **student1.fee\_update(50000)**, etc.: These lines call the fee\_update method for specific student objects to update their fees paid.
* **student1.display\_details()**, etc.: These lines call the display\_details method for each student object to print their information to the console.
* **print("-" \* 20)**: This line simply prints a separator line to make the output more readable.

**Task Description#3**

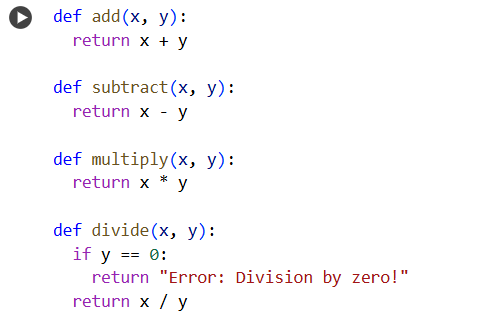
* Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).
* Incorporate manual **docstring** in code with NumPy Style
* Use AI assistance to generate a module-level docstring + individual function docstrings.
* Compare the AI-generated docstring with your manually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

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

PROMPT:

write a python code for calculator to find addition,subtraction,multiolication,and division using function without docstring and inline commands.

CODE:



PROMPT:

Generate manual docstring in code with numpy style

CODE:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

A screenshot of a computer error message

AI-generated content may be incorrect.

PROMPT:

GENERATE a module level docstring and individual function docstring and inline comments

CODE:

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

EXPLANATION:

* **""" This module provides basic calculator functions. ... """**: This is a **module-level docstring**. It explains the overall purpose of the Python script or module. In this case, it tells us that the script provides functions for basic calculations like addition, subtraction, multiplication, and division.
* **def add(x, y):**: This line defines a function named add that takes two arguments, x and y.
  + **""" Adds two numbers. ... """**: This is the **function docstring** for add. It explains what the function does, describes its input parameters (Parameters), and specifies what it returns (Returns). The docstring follows the NumPy style, which is a common convention for documenting scientific code in Python.
  + **return x + y # Return the sum of x and y**: This line performs the addition of x and y and returns the result. The text after the # is an **inline comment**, which provides a brief explanation of that specific line of code.
* **def subtract(x, y):**: This defines the subtract function, which takes two arguments and returns their difference. It also has a docstring explaining its purpose, parameters, and return value, as well as an inline comment.
* **def multiply(x, y):**: This defines the multiply function, which takes two arguments and returns their product. It also has a docstring and an inline comment.
* **def divide(x, y):**: This defines the divide function, which takes two arguments and returns their quotient.
  + **if y == 0:**: This line is a condition that checks if the second argument y is equal to 0.
  + **return "Error: Division by zero!" # Return error message if division by zero**: If y is 0, this line returns an error message as a string. The inline comment explains why this message is returned.
  + **return x / y # Return the result of the division**: If y is not 0, this line performs the division and returns the result. The inline comment explains this action.