SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE	**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

ProgramName:B. Tech	**Assignment Type: Lab**	**AcademicYear:**2025-2026

CourseCoordinatorName	Venkataramana Veeramsetty		
Instructor(s)Name	Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)		
CourseCode	24CS002PC215	**CourseTitle**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week4 - Wednesday	**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	

AssignmentNumber:9.3(Present assignment number)/**24**(Total number of assignments)

Q.No.	**Question**	***ExpectedTime to complete***
1	Lab 8: Documentation Generation: Automatic documentation and code comments **Lab Objectives:** • To understand the importance of documentation and code comments in software development. • To explore how AI-assisted coding tools can generate meaningful documentation and	Week4 - Wednesday

inline comments.

- To practice generating function-level and module-level docstrings automatically.
- To evaluate the quality, accuracy, and limitations of AI-generated documentation.
- To develop a small automated tool for documentation generation in Python..

Lab Outcomes (LOs):

After completing this lab, students will be able to:

- Apply AI-assisted coding tools to generate docstrings and inline comments for Python code.
- Critically analyze AI-generated documentation for correctness, completeness, and readability.
- Create structured documentation (function-level, module-level) following standard formats.

- Design and implement a mini documentation generator tool to automate code commenting and docstring creation.

Task Description#1 Basic Docstring Generation

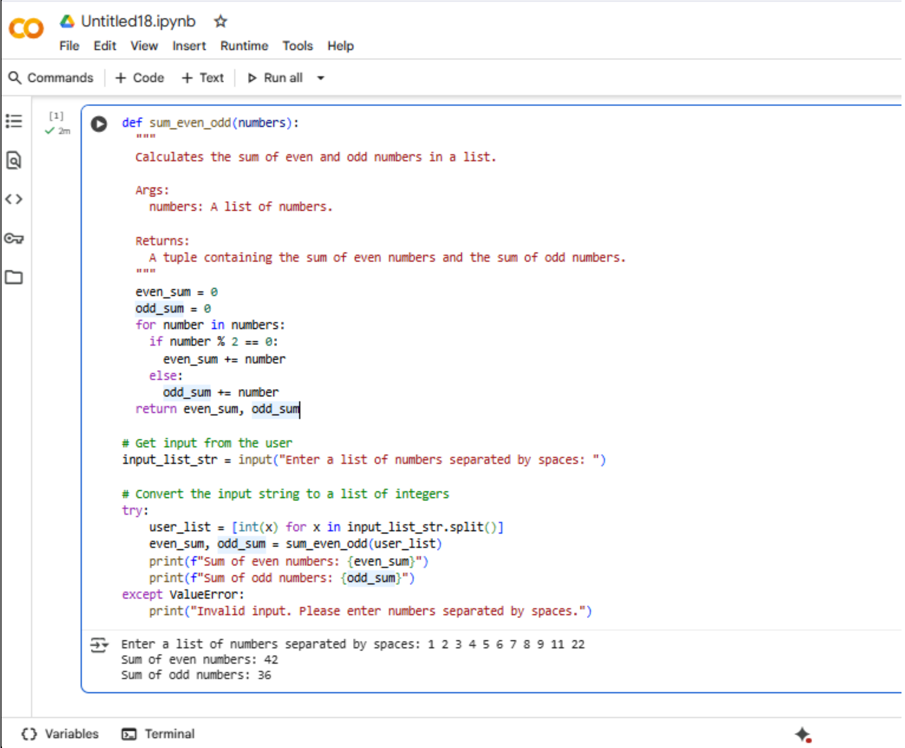
- Write python function to return sum of even and odd numbers in the given list.
- Incorporate manual **docstring** in code with Google Style
- Use an AI-assisted tool (e.g., Copilot, Cursor AI) to generate a docstring describing the function.
- Compare the AI-generated docstring with your manually written one.

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

Prompt#1:

1.Generate a python code to return sum of even and odd numbers in the given list by the user.

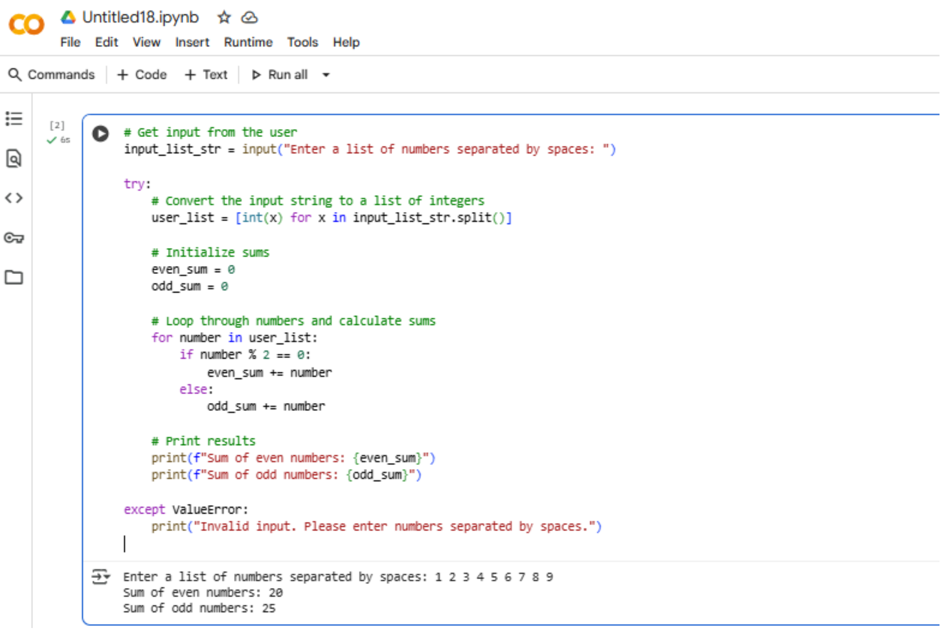
AI-generated code:



```
def sum_even_odd(numbers):
  """
  Calculates the sum of even and odd numbers in a list.

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

# Get input from the user
input_list_str = input("Enter a list of numbers separated by spaces: ")

# Convert the input string to a list of integers
try:
    user_list = [int(x) for x in input_list_str.split()]
    even_sum, odd_sum = sum_even_odd(user_list)
    print(f"Sum of even numbers: {even_sum}")
    print(f"Sum of odd numbers: {odd_sum}")
except ValueError:
    print("Invalid input. Please enter numbers separated by spaces.")
```

```
Enter a list of numbers separated by spaces: 1 2 3 4 5 6 7 8 9 11 22
Sum of even numbers: 42
Sum of odd numbers: 36
```

Manual code:



```python
# Get input from the user
input_list_str = input("Enter a list of numbers separated by spaces: ")

try:
    # Convert the input string to a list of integers
    user_list = [int(x) for x in input_list_str.split()]

    # Initialize sums
    even_sum = 0
    odd_sum = 0

    # Loop through numbers and calculate sums
    for number in user_list:
        if number % 2 == 0:
            even_sum += number
        else:
            odd_sum += number

    # Print results
    print(f"Sum of even numbers: {even_sum}")
    print(f"Sum of odd numbers: {odd_sum}")

except ValueError:
    print("Invalid input. Please enter numbers separated by spaces.")
```

```
Enter a list of numbers separated by spaces: 1 2 3 4 5 6 7 8 9
Sum of even numbers: 20
Sum of odd numbers: 25
```

Comparison Between AI-generated Docstring and Manual

1. Manual code writes everything step by step in the main program.
2. AI-generated code separates logic into a function with docstrings.
3. Manual code is less abstract, easy to understand line by line.
4. AI-generated code introduces functions and list.
5. Both versions use try/except for invalid input.

Code Explanation:

1. A function is defined to calculate the sum of even and odd numbers separately.
2. It takes a list of numbers as input and returns two values: even_sum and odd_sum.
3. If the user enters anything that’s not a number, a ValueError occurs.
4. Program displays the sum of even numbers and the sum of odd numbers separately.

Task Description#2 Automatic Inline Comments

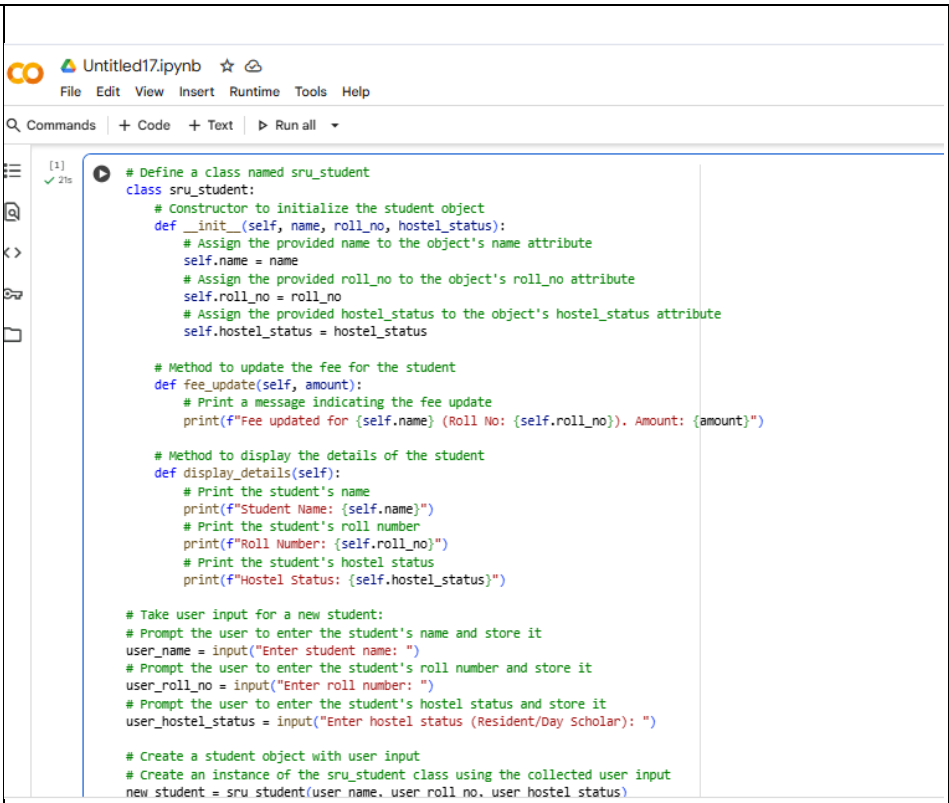
- Write python program for **sru_student** class with attributes like name, roll no., hostel_status and **fee_update** method and **display_details** method.
- Write comments manually for each line/code block
- Ask an AI tool to add inline comments explaining each line/step.
- Compare the AI-generated comments with your manually written one.

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

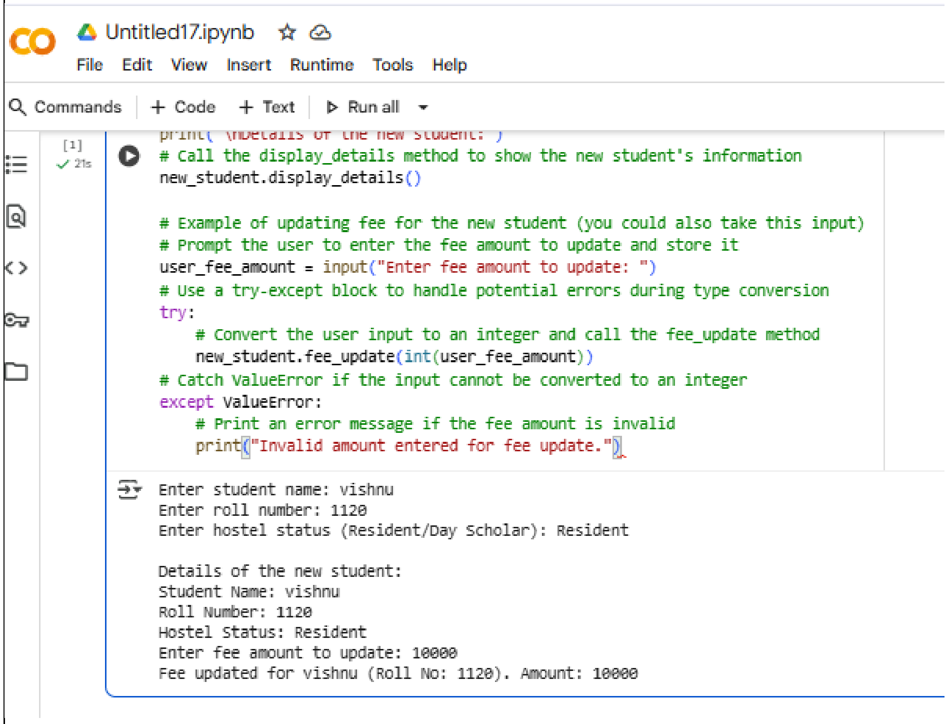
Prompt#2:

Generate a python program for sru_student class with attributes like name, roll no., hostel_status and fee_update method and display_details method.

AI-generated code:



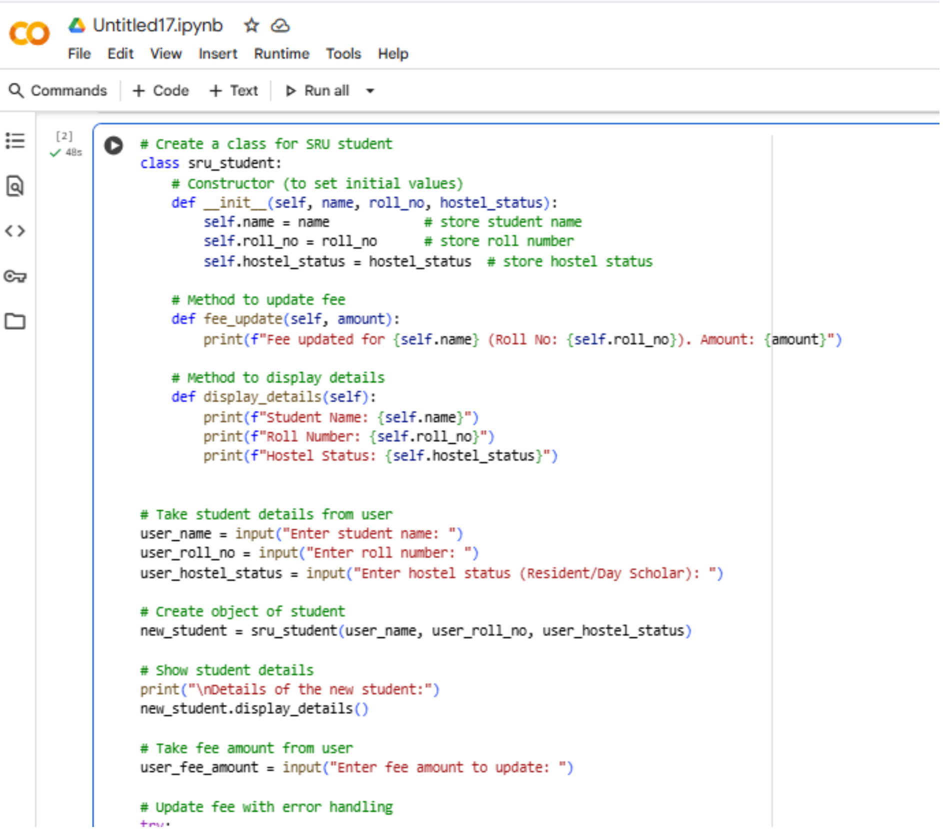
```
# Define a class named sru_student
class sru_student:
    # Constructor to initialize the student object
    def __init__(self, name, roll_no, hostel_status):
        # Assign the provided name to the object's name attribute
        self.name = name
        # Assign the provided roll_no to the object's roll_no attribute
        self.roll_no = roll_no
        # Assign the provided hostel_status to the object's hostel_status attribute
        self.hostel_status = hostel_status

    # Method to update the fee for the student
    def fee_update(self, amount):
        # Print a message indicating the fee update
        print(f"Fee updated for {self.name} (Roll No: {self.roll_no}). Amount: {amount}")

    # Method to display the details of the student
    def display_details(self):
        # Print the student's name
        print(f"Student Name: {self.name}")
        # Print the student's roll number
        print(f"Roll Number: {self.roll_no}")
        # Print the student's hostel status
        print(f"Hostel Status: {self.hostel_status}")

# Take user input for a new student:
# Prompt the user to enter the student's name and store it
user_name = input("Enter student name: ")
# Prompt the user to enter the student's roll number and store it
user_roll_no = input("Enter roll number: ")
# Prompt the user to enter the student's hostel status and store it
user_hostel_status = input("Enter hostel status (Resident/Day Scholar): ")

# Create a student object with user input
# Create an instance of the sru_student class using the collected user input
new_student = sru_student(user_name, user_roll_no, user_hostel_status)
```



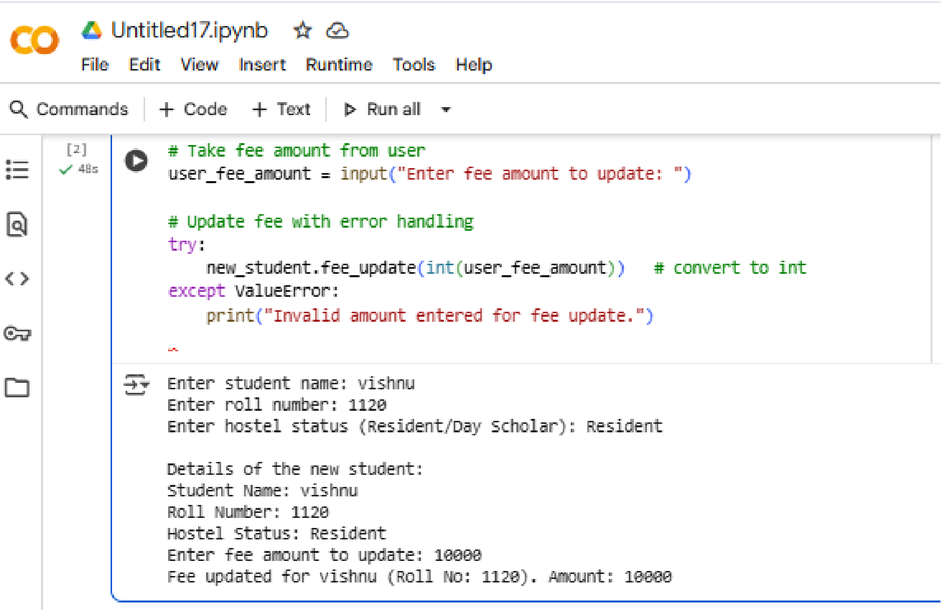
```
print("\nDetails of the new student:")
# Call the display_details method to show the new student's information
new_student.display_details()

# Example of updating fee for the new student (you could also take this input)
# Prompt the user to enter the fee amount to update and store it
user_fee_amount = input("Enter fee amount to update: ")
# Use a try-except block to handle potential errors during type conversion
try:
    # Convert the user input to an integer and call the fee_update method
    new_student.fee_update(int(user_fee_amount))
# Catch ValueError if the input cannot be converted to an integer
except ValueError:
    # Print an error message if the fee amount is invalid
    print("Invalid amount entered for fee update.")
```

```
Enter student name: vishnu
Enter roll number: 1120
Enter hostel status (Resident/Day Scholar): Resident

Details of the new student:
Student Name: vishnu
Roll Number: 1120
Hostel Status: Resident
Enter fee amount to update: 10000
Fee updated for vishnu (Roll No: 1120). Amount: 10000
```

Manual Code:



```python
# Create a class for SRU student
class sru_student:
    # Constructor (to set initial values)
    def __init__(self, name, roll_no, hostel_status):
        self.name = name            # store student name
        self.roll_no = roll_no      # store roll number
        self.hostel_status = hostel_status  # store hostel status

    # Method to update fee
    def fee_update(self, amount):
        print(f"Fee updated for {self.name} (Roll No: {self.roll_no}). Amount: {amount}")

    # Method to display details
    def display_details(self):
        print(f"Student Name: {self.name}")
        print(f"Roll Number: {self.roll_no}")
        print(f"Hostel Status: {self.hostel_status}")


# Take student details from user
user_name = input("Enter student name: ")
user_roll_no = input("Enter roll number: ")
user_hostel_status = input("Enter hostel status (Resident/Day Scholar): ")

# Create object of student
new_student = sru_student(user_name, user_roll_no, user_hostel_status)

# Show student details
print("\nDetails of the new student:")
new_student.display_details()

# Take fee amount from user
user_fee_amount = input("Enter fee amount to update: ")

# Update fee with error handling
```



```python
# Take fee amount from user
user_fee_amount = input("Enter fee amount to update: ")

# Update fee with error handling
try:
    new_student.fee_update(int(user_fee_amount))   # convert to int
except ValueError:
    print("Invalid amount entered for fee update.")
```

```
Enter student name: vishnu
Enter roll number: 1120
Enter hostel status (Resident/Day Scholar): Resident

Details of the new student:
Student Name: vishnu
Roll Number: 1120
Hostel Status: Resident
Enter fee amount to update: 10000
Fee updated for vishnu (Roll No: 1120). Amount: 10000
```

Comparison Between AI-generated comments and Manual Comments

1. AI comments are long and descriptive, while manual comments are short and direct.
2. AI uses formal, full-sentence explanations; manual uses simple keywords or phrases.
3. AI comments are best for learning/teaching, while manual comments are best for real coding projects.
4. AI comments suit beginners and documentation; manual comments suit developers reading

code quickly.

Code Explanation:

1. A class sru_student is created to represent a student.
2. Initializes the student object with name, roll_no, and hostel_status.
3.Prints a message showing that the student's fee has been updated with the given amount.
4. Prints the student's name, roll number, and hostel status

Task Description#3

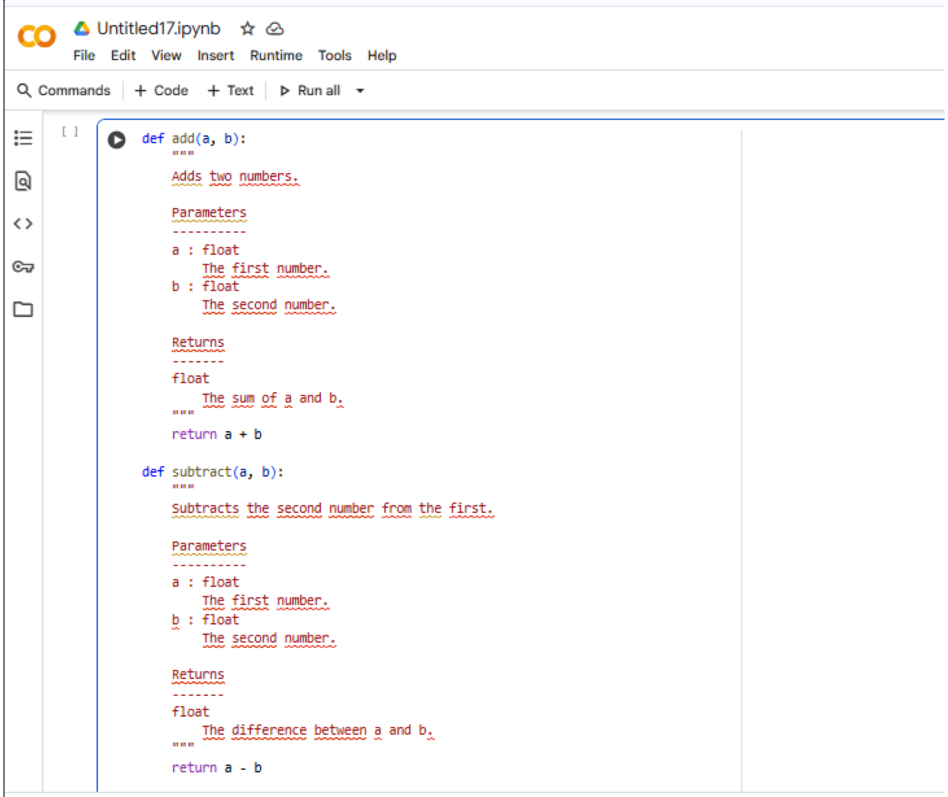
- Write a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).
- Incorporate manual **docstring** in code with NumPy Style
- Use AI assistance to generate a module-level docstring + individual function docstrings.
- Compare the AI-generated docstring with your manually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

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

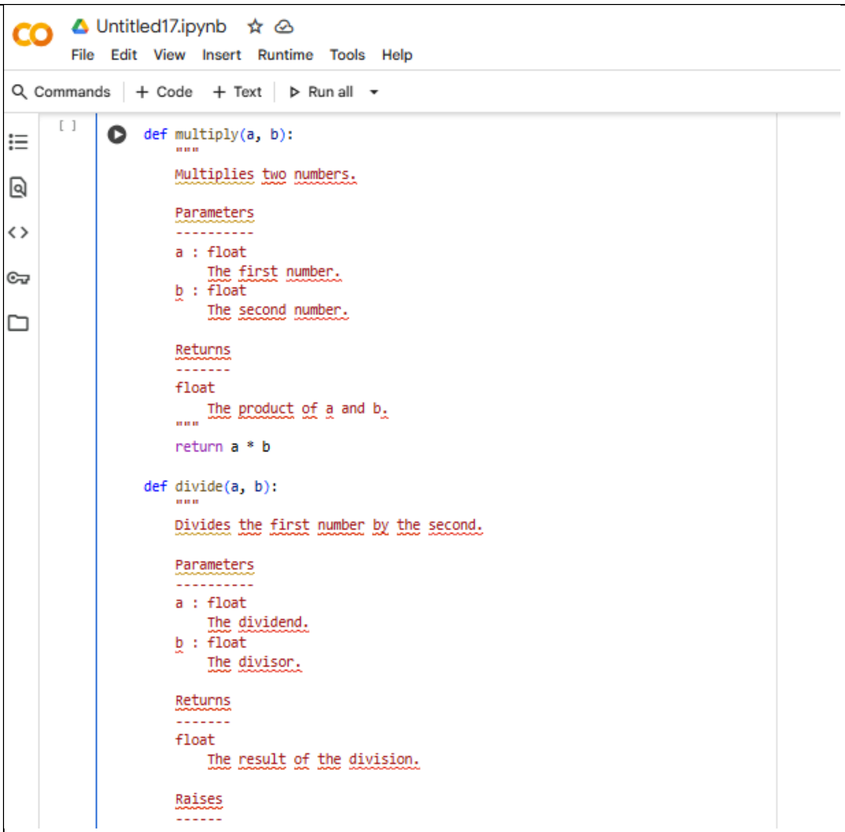
Prompt#3:

Generate a Python script with 3–4 functions (e.g., calculator: add, subtract, multiply, divide).with user input.

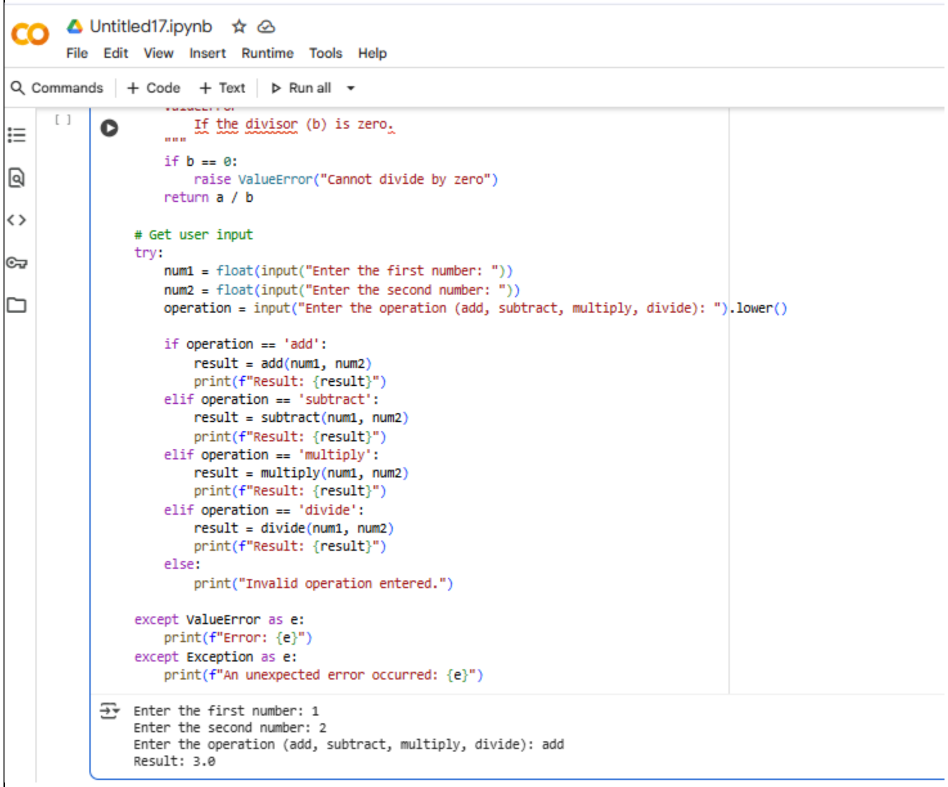
AI-generated code:



```
def add(a, b):
    """
    Adds two numbers.

    Parameters
    ----------
    a : float
        The first number.
    b : float
        The second number.

    Returns
    -------
    float
        The sum of a and b.
    """
    return a + b

def subtract(a, b):
    """
    Subtracts the second number from the first.

    Parameters
    ----------
    a : float
        The first number.
    b : float
        The second number.

    Returns
    -------
    float
        The difference between a and b.
    """
    return a - b
```


Untitled17.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
[ ]

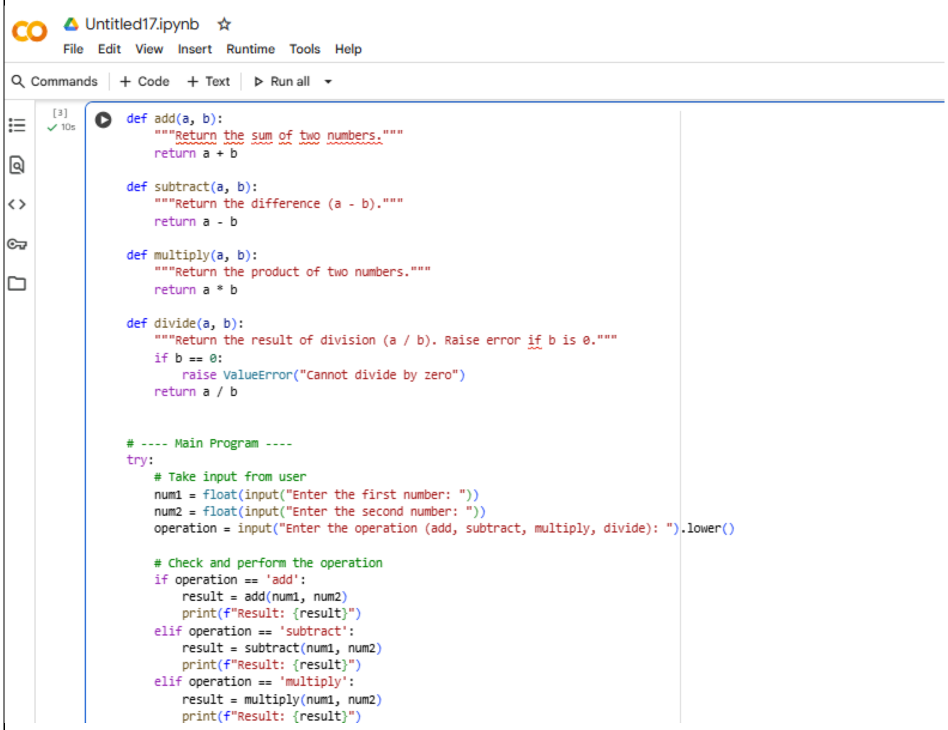
```
def multiply(a, b):
    """
    Multiplies two numbers.

    Parameters
    ----------
    a : float
        The first number.
    b : float
        The second number.

    Returns
    -------
    float
        The product of a and b.
    """
    return a * b

def divide(a, b):
    """
    Divides the first number by the second.

    Parameters
    ----------
    a : float
        The dividend.
    b : float
        The divisor.

    Returns
    -------
    float
        The result of the division.

    Raises
    ------
```


Untitled17.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
[ ]

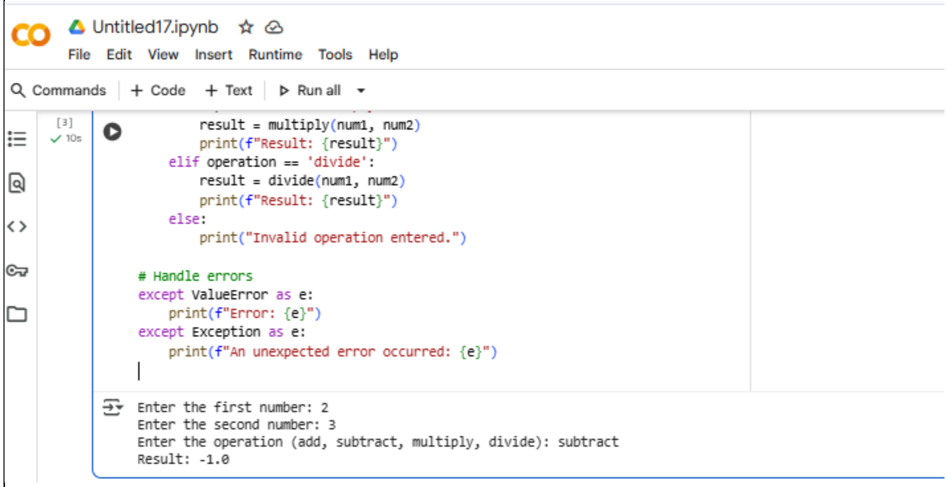
```
        If the divisor (b) is zero.
    """
    if b == 0:
        raise ValueError("Cannot divide by zero")
    return a / b

# Get user input
try:
    num1 = float(input("Enter the first number: "))
    num2 = float(input("Enter the second number: "))
    operation = input("Enter the operation (add, subtract, multiply, divide): ").lower()

    if operation == 'add':
        result = add(num1, num2)
        print(f"Result: {result}")
    elif operation == 'subtract':
        result = subtract(num1, num2)
        print(f"Result: {result}")
    elif operation == 'multiply':
        result = multiply(num1, num2)
        print(f"Result: {result}")
    elif operation == 'divide':
        result = divide(num1, num2)
        print(f"Result: {result}")
    else:
        print("Invalid operation entered.")

except ValueError as e:
    print(f"Error: {e}")
except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

```
Enter the first number: 1
Enter the second number: 2
Enter the operation (add, subtract, multiply, divide): add
Result: 3.0
```

Manual Code:


Untitled17.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
[3]
10s

```
def add(a, b):
    """Return the sum of two numbers."""
    return a + b

def subtract(a, b):
    """Return the difference (a - b)."""
    return a - b

def multiply(a, b):
    """Return the product of two numbers."""
    return a * b

def divide(a, b):
    """Return the result of division (a / b). Raise error if b is 0."""
    if b == 0:
        raise ValueError("Cannot divide by zero")
    return a / b


# ---- Main Program ----
try:
    # Take input from user
    num1 = float(input("Enter the first number: "))
    num2 = float(input("Enter the second number: "))
    operation = input("Enter the operation (add, subtract, multiply, divide): ").lower()

    # Check and perform the operation
    if operation == 'add':
        result = add(num1, num2)
        print(f"Result: {result}")
    elif operation == 'subtract':
        result = subtract(num1, num2)
        print(f"Result: {result}")
    elif operation == 'multiply':
        result = multiply(num1, num2)
        print(f"Result: {result}")
```


Untitled17.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
[3]
10s

```
        result = multiply(num1, num2)
        print(f"Result: {result}")
    elif operation == 'divide':
        result = divide(num1, num2)
        print(f"Result: {result}")
    else:
        print("Invalid operation entered.")

# Handle errors
except ValueError as e:
    print(f"Error: {e}")
except Exception as e:
    print(f"An unexpected error occurred: {e}")
```

```
Enter the first number: 2
Enter the second number: 3
Enter the operation (add, subtract, multiply, divide): subtract
Result: -1.0
```

Comparison:

1.AI docstring explains the function in a formal, structured way (with Parameters, Returns, Raises).
2. Manual docstring explains it in a short and direct way (one-liner focusing only on what the function does).
3. AI docstrings are best for documentation, while Manual docstrings are best for real projects .
4. AI docstrings focus on explaining *how inputs/outputs work*, while Manual docstrings focus on *what the function does*.

Code Explanation:

1.Four functions are defined → add, subtract, multiply, divide.
2.The program asks the user to enter two numbers.

	3.The user chooses which operation to perform based on that choice ,the result is printed. 4.If the user enters wrong input the program shows an error message. **Push documentation whole workspace as .md file in GitHub Repository** **Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots**	