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.

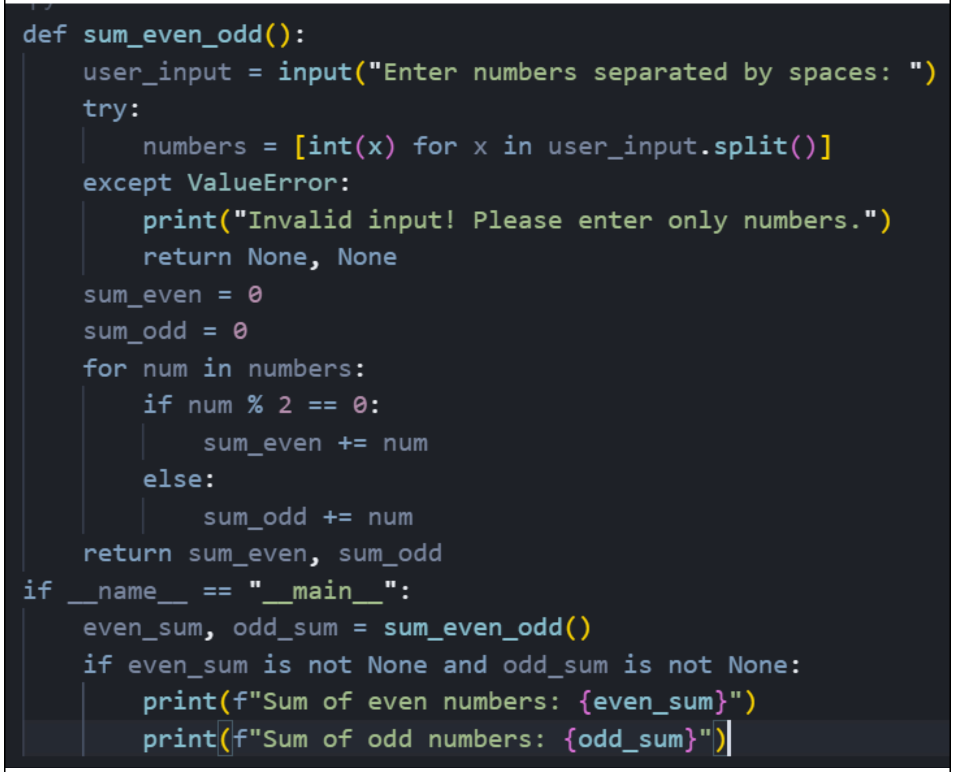
- Design and implement a mini documentation generator tool to automate code commenting and docstring creation.

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.

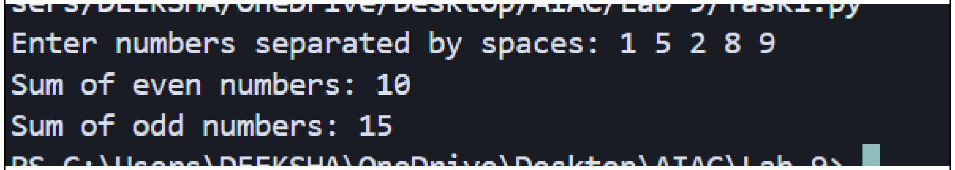
Expected Outcome#1: Students understand how AI can produce function-level documentation.

PROMPT : write the python function that return sum of even and odd numbers in the given list.



```python
def sum_even_odd():
    user_input = input("Enter numbers separated by spaces: ")
    try:
        numbers = [int(x) for x in user_input.split()]
    except ValueError:
        print("Invalid input! Please enter only numbers.")
        return None, None
    sum_even = 0
    sum_odd = 0
    for num in numbers:
        if num % 2 == 0:
            sum_even += num
        else:
            sum_odd += num
    return sum_even, sum_odd
if __name__ == "__main__":
    even_sum, odd_sum = sum_even_odd()
    if even_sum is not None and odd_sum is not None:
        print(f"Sum of even numbers: {even_sum}")
        print(f"Sum of odd numbers: {odd_sum}")
```

OUTPUT :

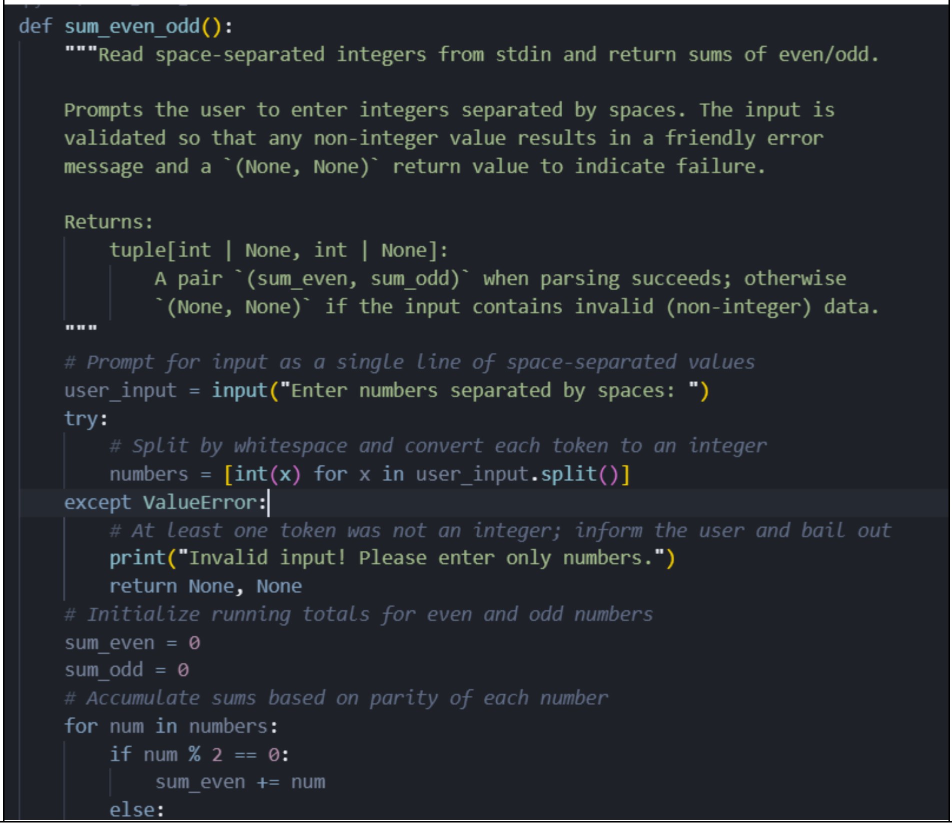


```
sers/DEEKSHA/OneDrive/Desktop/AIAC/Lab 9/Task1.py
Enter numbers separated by spaces: 1 5 2 8 9
Sum of even numbers: 10
Sum of odd numbers: 15
PS C:\Users\DEEKSHA\OneDrive\Desktop\AIAC\Lab 9>
```

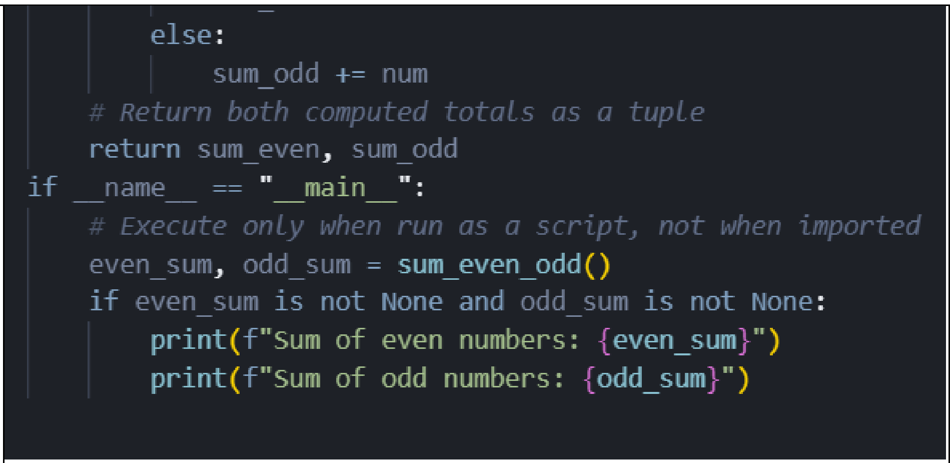
DOCSTRINGD FOR CODE :

The code was generate for the sum of even and odd numbers.
Aggr:
The int was used for list of given numbers.
Return :
It was have Boolean values like (none) that says invalid input.

DOCSTRINGD by cursor CODE :



```python
def sum_even_odd():
    """Read space-separated integers from stdin and return sums of even/odd.

    Prompts the user to enter integers separated by spaces. The input is
    validated so that any non-integer value results in a friendly error
    message and a `(None, None)` return value to indicate failure.

    Returns:
        tuple[int | None, int | None]:
            A pair `(sum_even, sum_odd)` when parsing succeeds; otherwise
            `(None, None)` if the input contains invalid (non-integer) data.
    """
    # Prompt for input as a single line of space-separated values
    user_input = input("Enter numbers separated by spaces: ")
    try:
        # Split by whitespace and convert each token to an integer
        numbers = [int(x) for x in user_input.split()]
    except ValueError:
        # At least one token was not an integer; inform the user and bail out
        print("Invalid input! Please enter only numbers.")
        return None, None
    # Initialize running totals for even and odd numbers
    sum_even = 0
    sum_odd = 0
    # Accumulate sums based on parity of each number
    for num in numbers:
        if num % 2 == 0:
            sum_even += num
        else:
```



```
        else:
            sum_odd += num
    # Return both computed totals as a tuple
    return sum_even, sum_odd
if __name__ == "__main__":
    # Execute only when run as a script, not when imported
    even_sum, odd_sum = sum_even_odd()
    if even_sum is not None and odd_sum is not None:
        print(f"Sum of even numbers: {even_sum}")
        print(f"Sum of odd numbers: {odd_sum}")
```

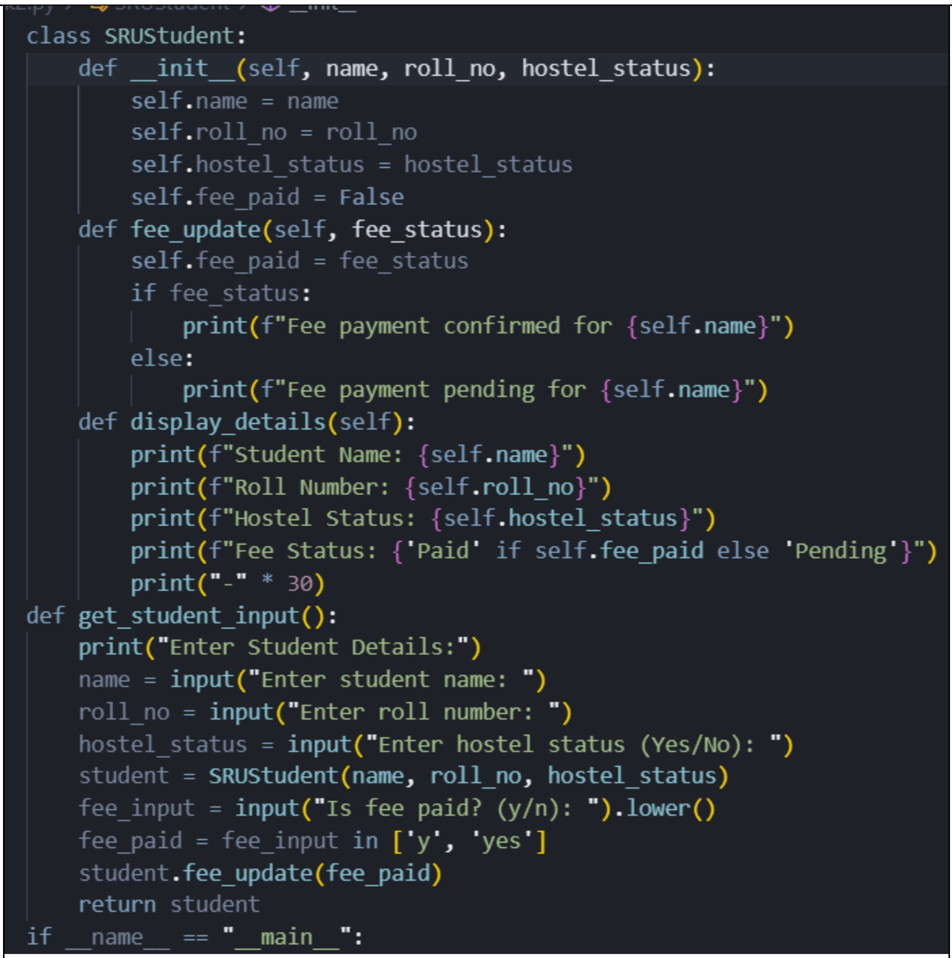
Task Description#2 Automatic Inline Comments

- 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.

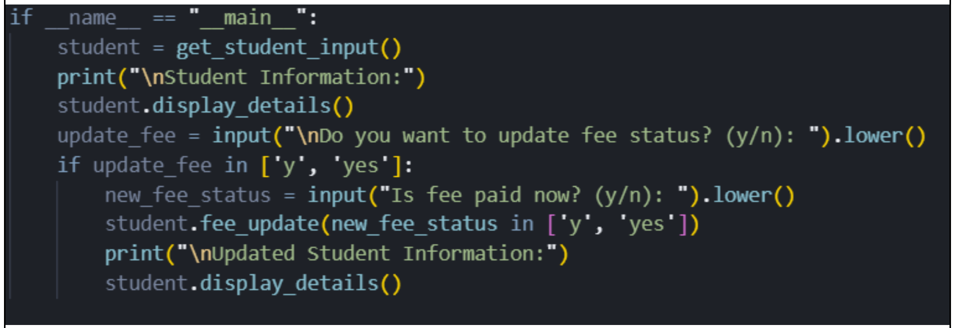
Expected Output#2: Students critically analyze AI-generated code comments.

PROMPT: write a python function that generate for sru_student class with attributes like name,roll no,hostel_status and fee_update method and display_details method with user input.

CODE:

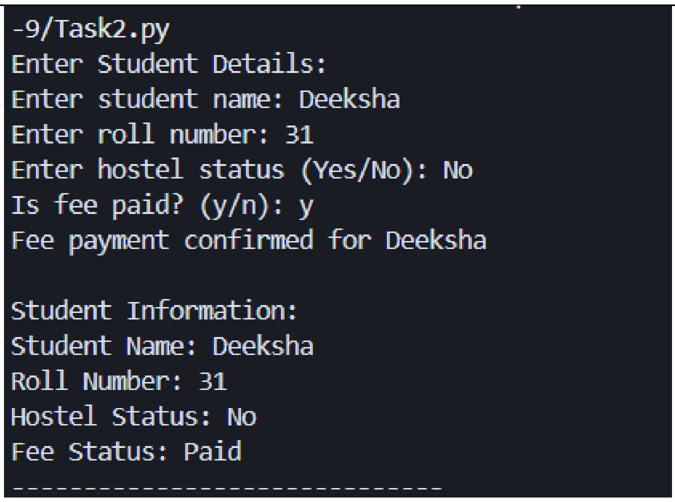


```python
class SRUStudent:
    def __init__(self, name, roll_no, hostel_status):
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee_paid = False
    def fee_update(self, fee_status):
        self.fee_paid = fee_status
        if fee_status:
            print(f"Fee payment confirmed for {self.name}")
        else:
            print(f"Fee payment pending for {self.name}")
    def display_details(self):
        print(f"Student Name: {self.name}")
        print(f"Roll Number: {self.roll_no}")
        print(f"Hostel Status: {self.hostel_status}")
        print(f"Fee Status: {'Paid' if self.fee_paid else 'Pending'}")
        print("-" * 30)
def get_student_input():
    print("Enter Student Details:")
    name = input("Enter student name: ")
    roll_no = input("Enter roll number: ")
    hostel_status = input("Enter hostel status (Yes/No): ")
    student = SRUStudent(name, roll_no, hostel_status)
    fee_input = input("Is fee paid? (y/n): ").lower()
    fee_paid = fee_input in ['y', 'yes']
    student.fee_update(fee_paid)
    return student
if __name__ == "__main__":
```



```python
if __name__ == "__main__":
    student = get_student_input()
    print("\nStudent Information:")
    student.display_details()
    update_fee = input("\nDo you want to update fee status? (y/n): ").lower()
    if update_fee in ['y', 'yes']:
        new_fee_status = input("Is fee paid now? (y/n): ").lower()
        student.fee_update(new_fee_status in ['y', 'yes'])
        print("\nUpdated Student Information:")
        student.display_details()
```

OUTPUT:

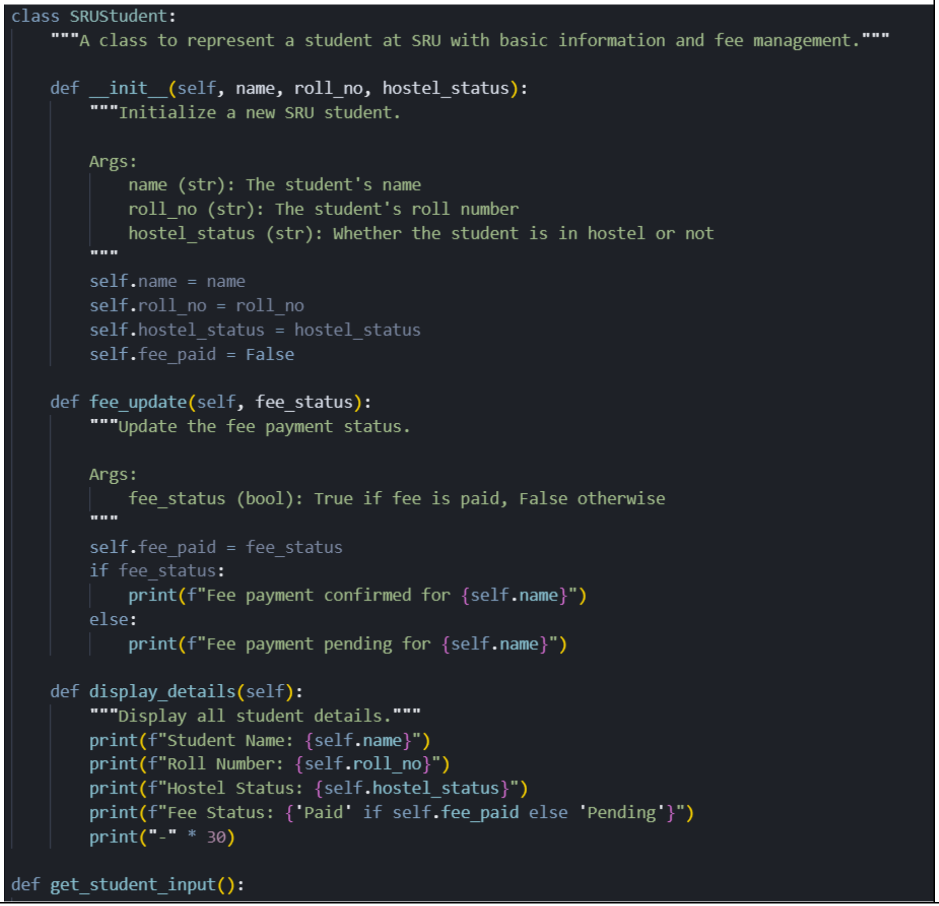


```
-9/Task2.py
Enter Student Details:
Enter student name: Deeksha
Enter roll number: 31
Enter hostel status (Yes/No): No
Is fee paid? (y/n): y
Fee payment confirmed for Deeksha

Student Information:
Student Name: Deeksha
Roll Number: 31
Hostel Status: No
Fee Status: Paid
------------------------------
```

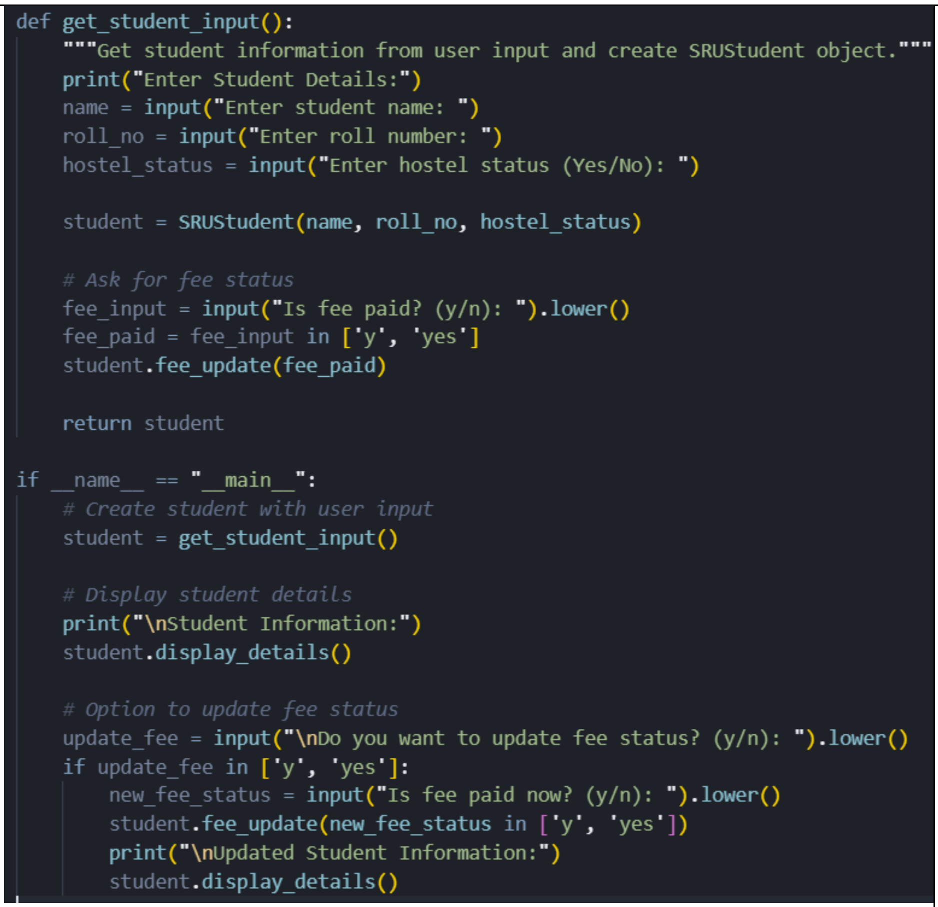
DOCSTRING FOR CODE:

This code was generated for sru student_details regarding to college details
AGGR:
def class was used for details from
return :
main function that prints the question to fill.

DOCSTRING FROM CURSOR:



```python
class SRUStudent:
    """A class to represent a student at SRU with basic information and fee management."""

    def __init__(self, name, roll_no, hostel_status):
        """Initialize a new SRU student.

        Args:
            name (str): The student's name
            roll_no (str): The student's roll number
            hostel_status (str): Whether the student is in hostel or not
        """
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee_paid = False

    def fee_update(self, fee_status):
        """Update the fee payment status.

        Args:
            fee_status (bool): True if fee is paid, False otherwise
        """
        self.fee_paid = fee_status
        if fee_status:
            print(f"Fee payment confirmed for {self.name}")
        else:
            print(f"Fee payment pending for {self.name}")

    def display_details(self):
        """Display all student details."""
        print(f"Student Name: {self.name}")
        print(f"Roll Number: {self.roll_no}")
        print(f"Hostel Status: {self.hostel_status}")
        print(f"Fee Status: {'Paid' if self.fee_paid else 'Pending'}")
        print("-" * 30)

def get_student_input():
```



```python
def get_student_input():
    """Get student information from user input and create SRUStudent object."""
    print("Enter Student Details:")
    name = input("Enter student name: ")
    roll_no = input("Enter roll number: ")
    hostel_status = input("Enter hostel status (Yes/No): ")

    student = SRUStudent(name, roll_no, hostel_status)

    # Ask for fee status
    fee_input = input("Is fee paid? (y/n): ").lower()
    fee_paid = fee_input in ['y', 'yes']
    student.fee_update(fee_paid)

    return student

if __name__ == "__main__":
    # Create student with user input
    student = get_student_input()

    # Display student details
    print("\nStudent Information:")
    student.display_details()

    # Option to update fee status
    update_fee = input("\nDo you want to update fee status? (y/n): ").lower()
    if update_fee in ['y', 'yes']:
        new_fee_status = input("Is fee paid now? (y/n): ").lower()
        student.fee_update(new_fee_status in ['y', 'yes'])
        print("\nUpdated Student Information:")
        student.display_details()
```

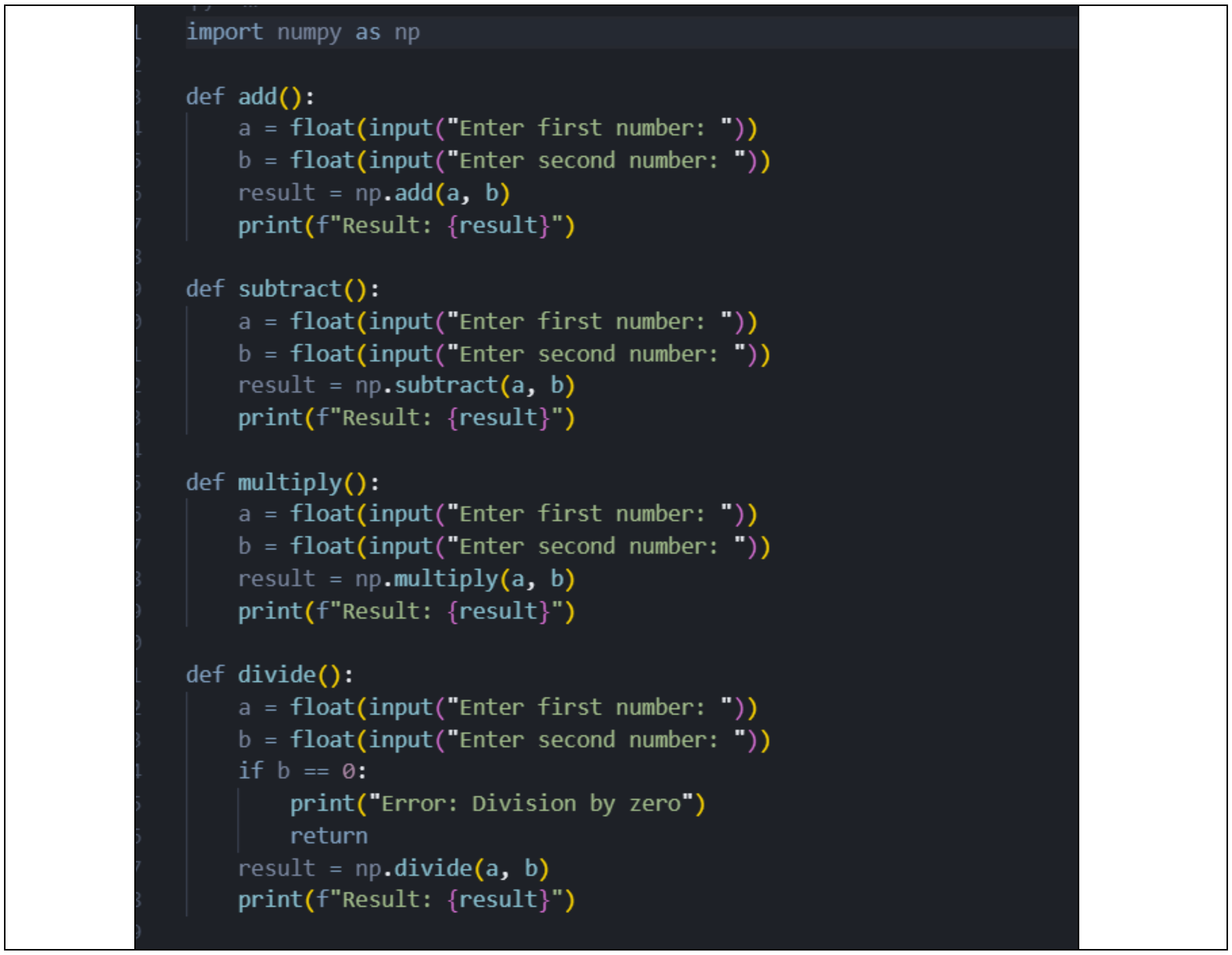
Task Description#3

- 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.

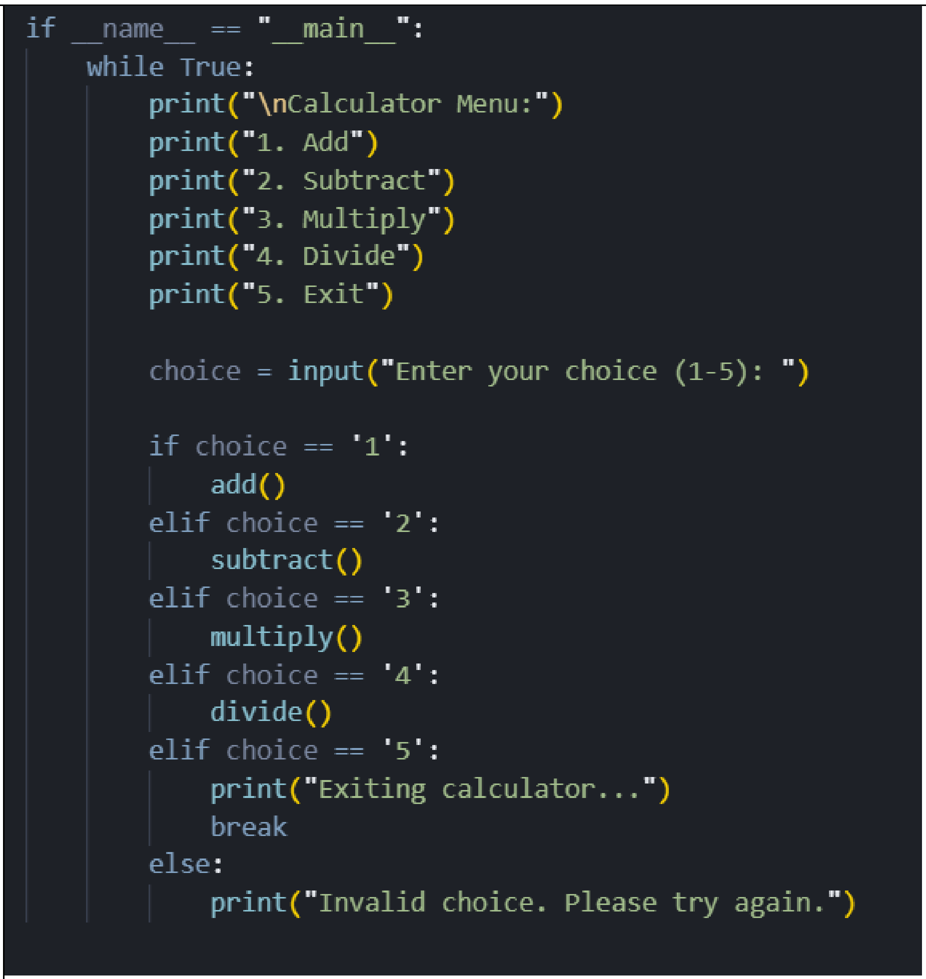
Expected Output#3: Students learn structured documentation for multi-function scripts

PROMPT :

Write a python function that script with 3-4 functions(calculator: add,subtract,multiply,divide) by using numpy with user input.

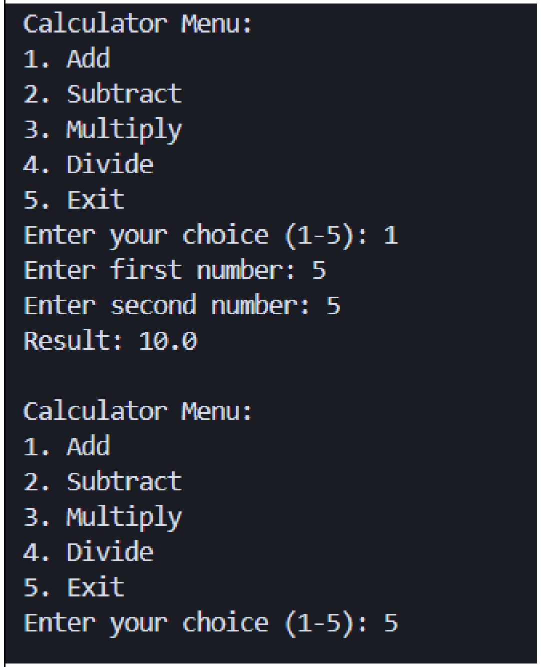


```python
import numpy as np

def add():
    a = float(input("Enter first number: "))
    b = float(input("Enter second number: "))
    result = np.add(a, b)
    print(f"Result: {result}")

def subtract():
    a = float(input("Enter first number: "))
    b = float(input("Enter second number: "))
    result = np.subtract(a, b)
    print(f"Result: {result}")

def multiply():
    a = float(input("Enter first number: "))
    b = float(input("Enter second number: "))
    result = np.multiply(a, b)
    print(f"Result: {result}")

def divide():
    a = float(input("Enter first number: "))
    b = float(input("Enter second number: "))
    if b == 0:
        print("Error: Division by zero")
        return
    result = np.divide(a, b)
    print(f"Result: {result}")
```



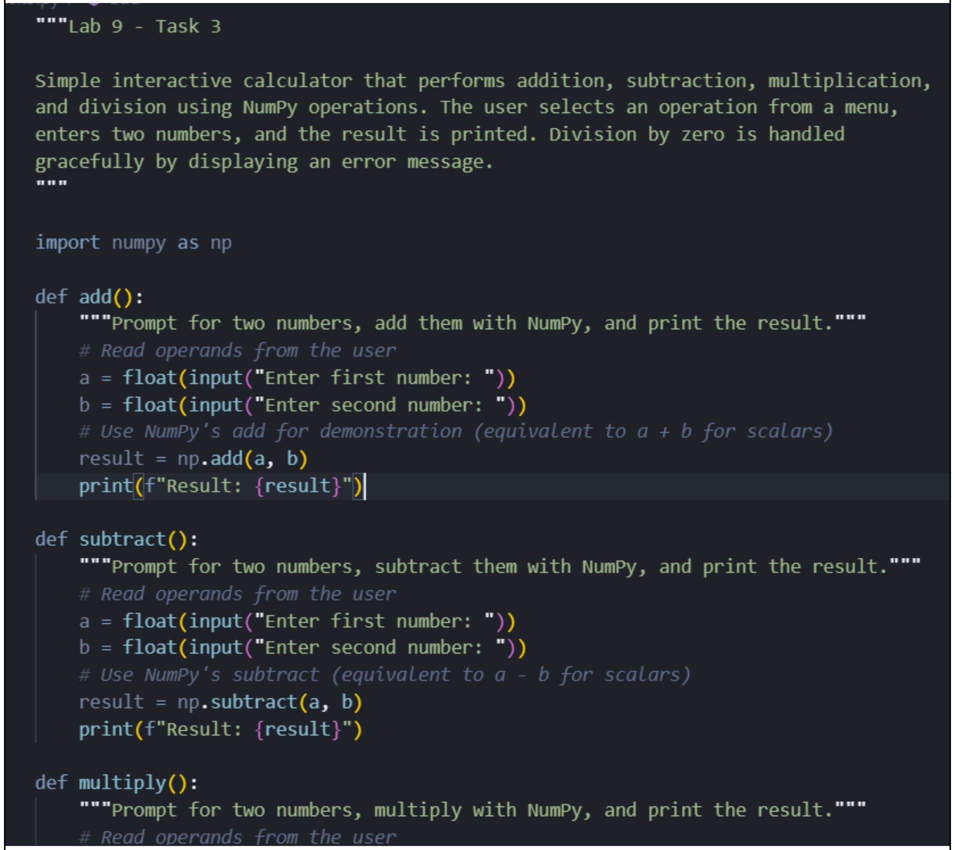
```python
if __name__ == "__main__":
    while True:
        print("\nCalculator Menu:")
        print("1. Add")
        print("2. Subtract")
        print("3. Multiply")
        print("4. Divide")
        print("5. Exit")

        choice = input("Enter your choice (1-5): ")

        if choice == '1':
            add()
        elif choice == '2':
            subtract()
        elif choice == '3':
            multiply()
        elif choice == '4':
            divide()
        elif choice == '5':
            print("Exiting calculator...")
            break
        else:
            print("Invalid choice. Please try again.")
```

OUTPUT:

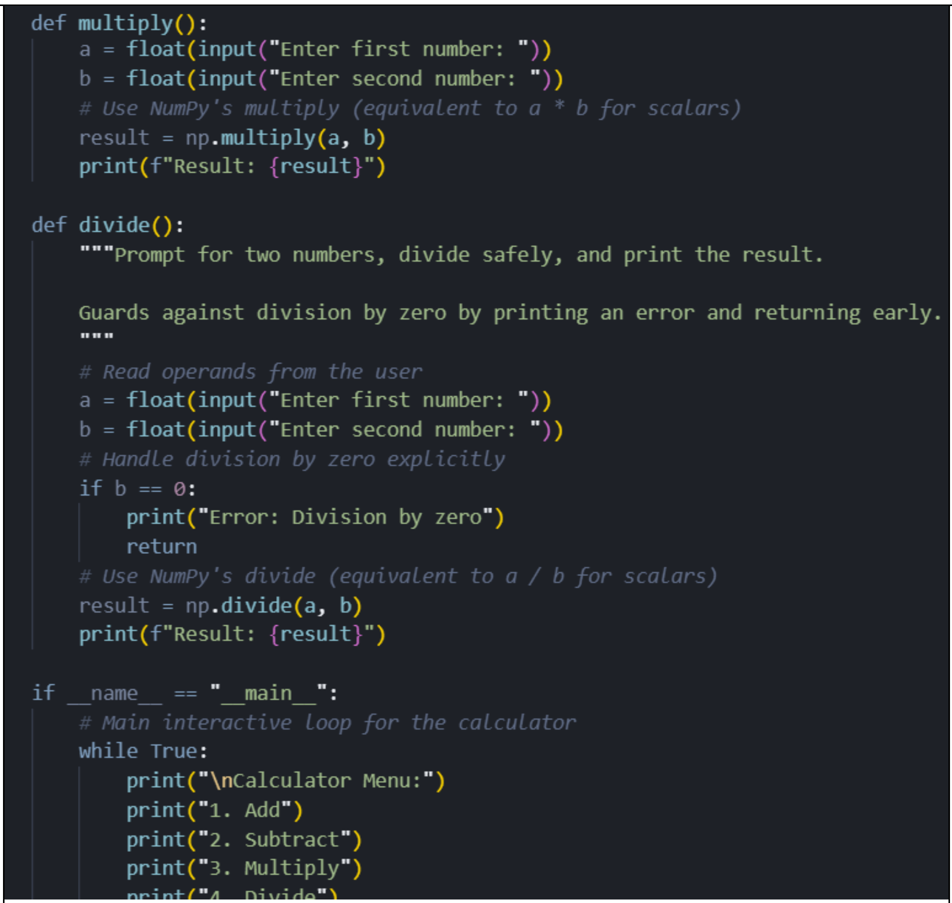


```
Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Enter your choice (1-5): 1
Enter first number: 5
Enter second number: 5
Result: 10.0

Calculator Menu:
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Enter your choice (1-5): 5
```

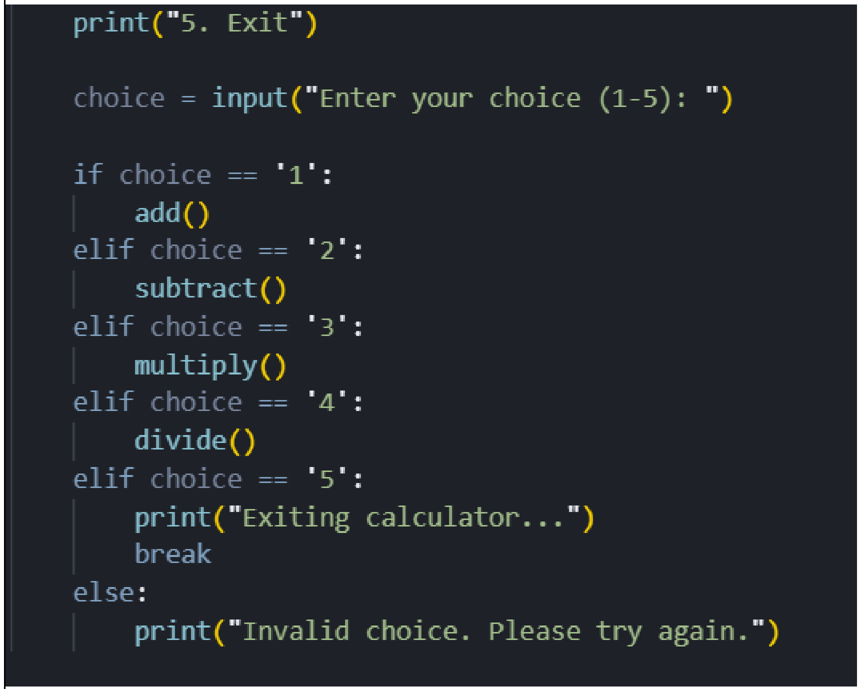
DOCSTRING FOR CODE:

	This code was generate for calculator that performs addition,subtravtion,multiplication,division **Aggr:** **Uses the float function** **And numpy def** **And if else statements** **User:** **given user inputs values.** **DOCSTRING FROM CURSOR:** 	

```
"""Lab 9 - Task 3

Simple interactive calculator that performs addition, subtraction, multiplication,
and division using NumPy operations. The user selects an operation from a menu,
enters two numbers, and the result is printed. Division by zero is handled
gracefully by displaying an error message.
"""

import numpy as np

def add():
    """Prompt for two numbers, add them with NumPy, and print the result."""
    # Read operands from the user
    a = float(input("Enter first number: "))
    b = float(input("Enter second number: "))
    # Use NumPy's add for demonstration (equivalent to a + b for scalars)
    result = np.add(a, b)
    print(f"Result: {result}")

def subtract():
    """Prompt for two numbers, subtract them with NumPy, and print the result."""
    # Read operands from the user
    a = float(input("Enter first number: "))
    b = float(input("Enter second number: "))
    # Use NumPy's subtract (equivalent to a - b for scalars)
    result = np.subtract(a, b)
    print(f"Result: {result}")

def multiply():
    """Prompt for two numbers, multiply with NumPy, and print the result."""
    # Read operands from the user
```



```python
def multiply():
    a = float(input("Enter first number: "))
    b = float(input("Enter second number: "))
    # Use NumPy's multiply (equivalent to a * b for scalars)
    result = np.multiply(a, b)
    print(f"Result: {result}")

def divide():
    """Prompt for two numbers, divide safely, and print the result.

    Guards against division by zero by printing an error and returning early.
    """
    # Read operands from the user
    a = float(input("Enter first number: "))
    b = float(input("Enter second number: "))
    # Handle division by zero explicitly
    if b == 0:
        print("Error: Division by zero")
        return
    # Use NumPy's divide (equivalent to a / b for scalars)
    result = np.divide(a, b)
    print(f"Result: {result}")

if __name__ == "__main__":
    # Main interactive loop for the calculator
    while True:
        print("\nCalculator Menu:")
        print("1. Add")
        print("2. Subtract")
        print("3. Multiply")
        print("4. Divide")
```



```python
    print("5. Exit")

    choice = input("Enter your choice (1-5): ")

    if choice == '1':
        add()
    elif choice == '2':
        subtract()
    elif choice == '3':
        multiply()
    elif choice == '4':
        divide()
    elif choice == '5':
        print("Exiting calculator...")
        break
    else:
        print("Invalid choice. Please try again.")
```

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