AIPP ASSIGNMENT 09

NAME: MOHAMMED NIZAMUDDIN

HALLTICKETNO: 2503B05144(MTECH)

Task Description#1 Basic Docstring Generation

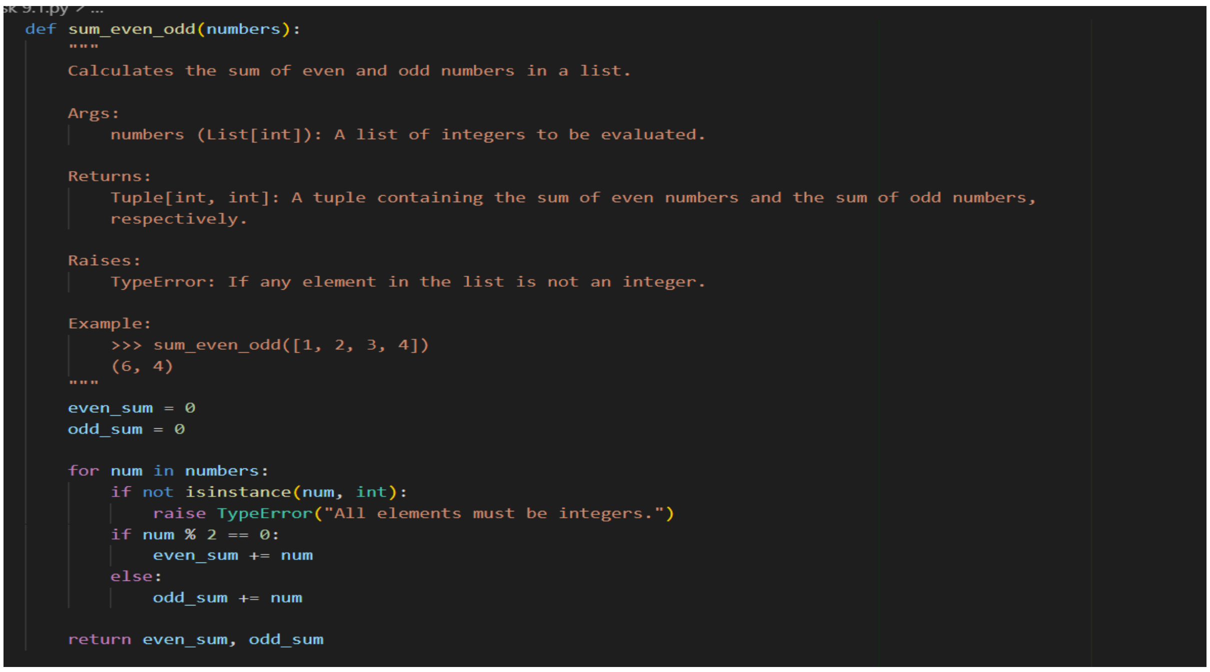
- Write python function to return sum of even and odd numbers in the given list.
- Incorporate manual **docstring** in code with Google Style
- Use an AI-assisted tool (e.g., Copilot, Cursor AI) to generate a docstring describing the function
- Compare the AI-generated docstring with your manually written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

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

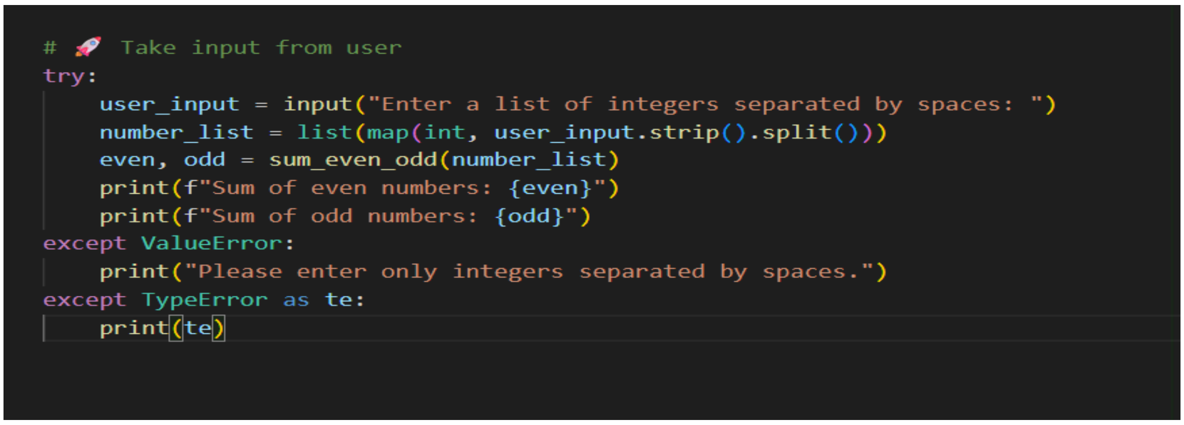
Prompt-1: Write python function to return sum of even and odd numbers in the given list.
Prompt-2: Incorporate manual docstring in code with Google Style

CODE:

# Python Function with Manual Docstring

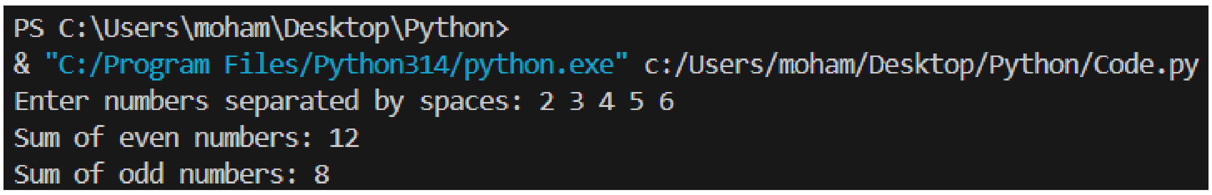


```python
def sum_even_odd(numbers):
    """
    Calculates the sum of even and odd numbers in a list.

    Args:
        numbers (List[int]): A list of integers to be evaluated.

    Returns:
        Tuple[int, int]: A tuple containing the sum of even numbers and the sum of odd numbers,
        respectively.

    Raises:
        TypeError: If any element in the list is not an integer.

    Example:
        >>> sum_even_odd([1, 2, 3, 4])
        (6, 4)
    """
    even_sum = 0
    odd_sum = 0

    for num in numbers:
        if not isinstance(num, int):
            raise TypeError("All elements must be integers.")
        if num % 2 == 0:
            even_sum += num
        else:
            odd_sum += num

    return even_sum, odd_sum
```



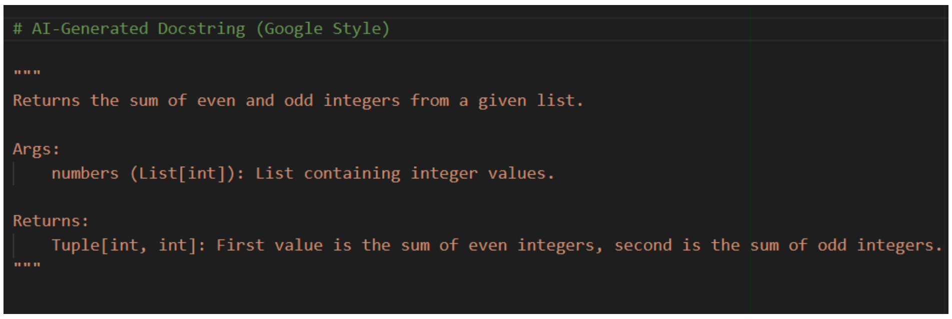
```python
# 🚀 Take input from user
try:
    user_input = input("Enter a list of integers separated by spaces: ")
    number_list = list(map(int, user_input.strip().split()))
    even, odd = sum_even_odd(number_list)
    print(f"Sum of even numbers: {even}")
    print(f"Sum of odd numbers: {odd}")
except ValueError:
    print("Please enter only integers separated by spaces.")
except TypeError as te:
    print(te)
```

OUTPUT:



```
PS C:\Users\moham\Desktop\Python>
& "C:/Program Files/Python314/python.exe" c:/Users/moham/Desktop/Python/Code.py
Enter numbers separated by spaces: 2 3 4 5 6
Sum of even numbers: 12
Sum of odd numbers: 8
```

# AI-Generated Docstring (Google Style)



```
# AI-Generated Docstring (Google Style)

"""
Returns the sum of even and odd integers from a given list.

Args:
    numbers (List[int]): List containing integer values.

Returns:
    Tuple[int, int]: First value is the sum of even integers, second is the sum of odd integers.
"""
```

Comparison: Manual vs AI-Generated Docstrin

Feature	**Manual Docstring**	**AI-Generated Docstring**
Clarity	Very clear and descriptive	Clear but more concise
Structure	Includes Args, Returns, Raises, Example	Includes only Args and Returns
Error Handling	Mentions TypeError explicitly	No mention of exceptions
Example Usage	Provides a usage example	No example provided
Tone	Slightly more formal and instructional	More minimalistic and utilitarian

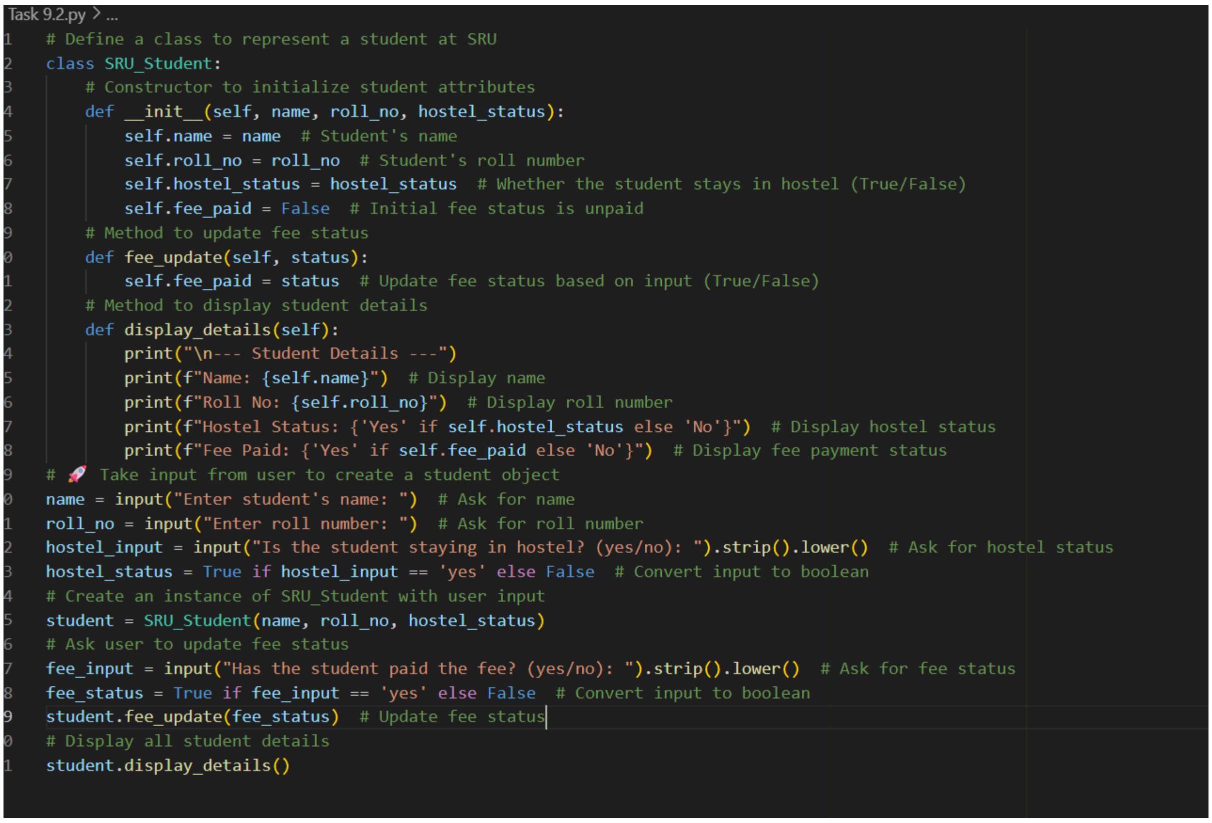
Task Description#2 Automatic Inline Comments

- 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.

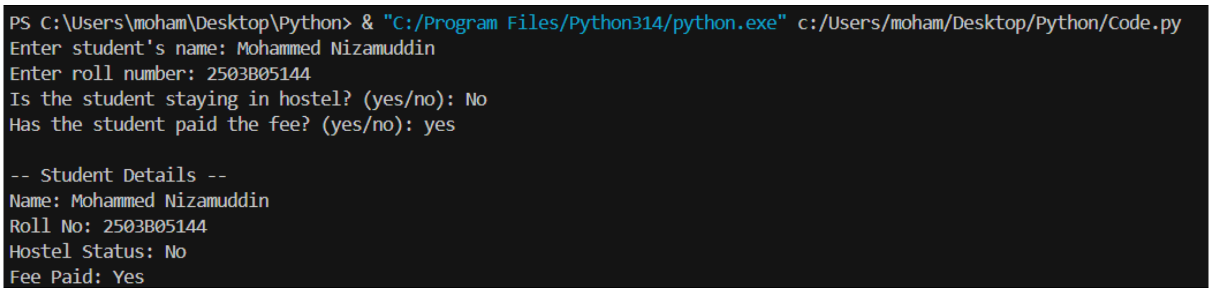
CODE:
#Manually Commented Python Program with User Input



Task 9.2.py > ...

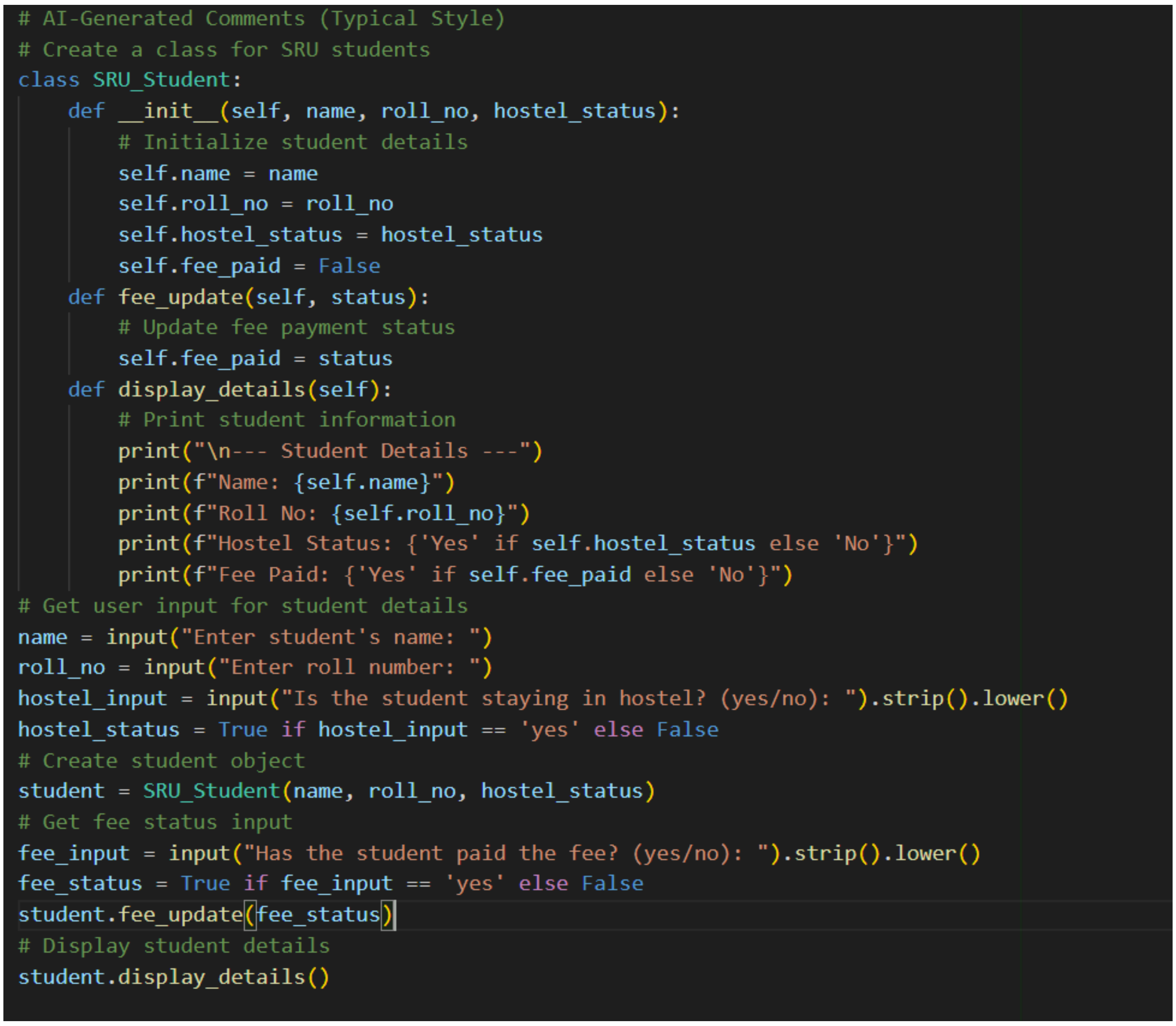
```python
# Define a class to represent a student at SRU
class SRU_Student:
    # Constructor to initialize student attributes
    def __init__(self, name, roll_no, hostel_status):
        self.name = name  # Student's name
        self.roll_no = roll_no  # Student's roll number
        self.hostel_status = hostel_status  # Whether the student stays in hostel (True/False)
        self.fee_paid = False  # Initial fee status is unpaid
    # Method to update fee status
    def fee_update(self, status):
        self.fee_paid = status  # Update fee status based on input (True/False)
    # Method to display student details
    def display_details(self):
        print("\n--- Student Details ---")
        print(f"Name: {self.name}")  # Display name
        print(f"Roll No: {self.roll_no}")  # Display roll number
        print(f"Hostel Status: {'Yes' if self.hostel_status else 'No'}")  # Display hostel status
        print(f"Fee Paid: {'Yes' if self.fee_paid else 'No'}")  # Display fee payment status
# 🚀 Take input from user to create a student object
name = input("Enter student's name: ")  # Ask for name
roll_no = input("Enter roll number: ")  # Ask for roll number
hostel_input = input("Is the student staying in hostel? (yes/no): ").strip().lower()  # Ask for hostel status
hostel_status = True if hostel_input == 'yes' else False  # Convert input to boolean
# Create an instance of SRU_Student with user input
student = SRU_Student(name, roll_no, hostel_status)
# Ask user to update fee status
fee_input = input("Has the student paid the fee? (yes/no): ").strip().lower()  # Ask for fee status
fee_status = True if fee_input == 'yes' else False  # Convert input to boolean
student.fee_update(fee_status)  # Update fee status
# Display all student details
student.display_details()
```

OUTPUT:



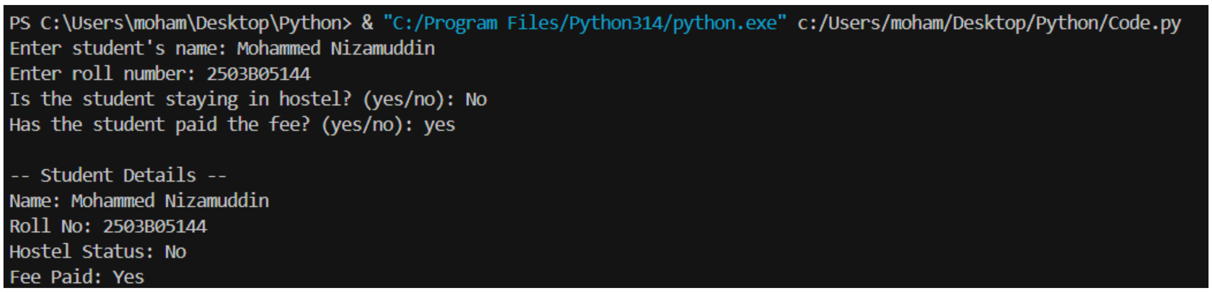
```
PS C:\Users\moham\Desktop\Python> & "C:/Program Files/Python314/python.exe" c:/Users/moham/Desktop/Python/Code.py
Enter student's name: Mohammed Nizamuddin
Enter roll number: 2503B05144
Is the student staying in hostel? (yes/no): No
Has the student paid the fee? (yes/no): yes

-- Student Details --
Name: Mohammed Nizamuddin
Roll No: 2503B05144
Hostel Status: No
Fee Paid: Yes
```

AI Generated Comments



```python
# AI-Generated Comments (Typical Style)
# Create a class for SRU students
class SRU_Student:
    def __init__(self, name, roll_no, hostel_status):
        # Initialize student details
        self.name = name
        self.roll_no = roll_no
        self.hostel_status = hostel_status
        self.fee_paid = False
    def fee_update(self, status):
        # Update fee payment status
        self.fee_paid = status
    def display_details(self):
        # Print student information
        print("\n--- Student Details ---")
        print(f"Name: {self.name}")
        print(f"Roll No: {self.roll_no}")
        print(f"Hostel Status: {'Yes' if self.hostel_status else 'No'}")
        print(f"Fee Paid: {'Yes' if self.fee_paid else 'No'}")
# Get user input for student details
name = input("Enter student's name: ")
roll_no = input("Enter roll number: ")
hostel_input = input("Is the student staying in hostel? (yes/no): ").strip().lower()
hostel_status = True if hostel_input == 'yes' else False
# Create student object
student = SRU_Student(name, roll_no, hostel_status)
# Get fee status input
fee_input = input("Has the student paid the fee? (yes/no): ").strip().lower()
fee_status = True if fee_input == 'yes' else False
student.fee_update(fee_status)
# Display student details
student.display_details()
```

OUTPUT:



```
PS C:\Users\moham\Desktop\Python> & "C:/Program Files/Python314/python.exe" c:/Users/moham/Desktop/Python/Code.py
Enter student's name: Mohammed Nizamuddin
Enter roll number: 2503B05144
Is the student staying in hostel? (yes/no): No
Has the student paid the fee? (yes/no): yes

-- Student Details --
Name: Mohammed Nizamuddin
Roll No: 2503B05144
Hostel Status: No
Fee Paid: Yes
```

“The AI comments are technically correct but lack depth. For example, # Initialize student details doesn’t explain what each attribute represents or why fee_paid defaults to False.
Also, the comment # Create student object is vague—it could be improved by saying # Create an SRU_Student instance using user-provided data.
The AI doesn’t use inline comments, which makes it harder to follow for be

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.

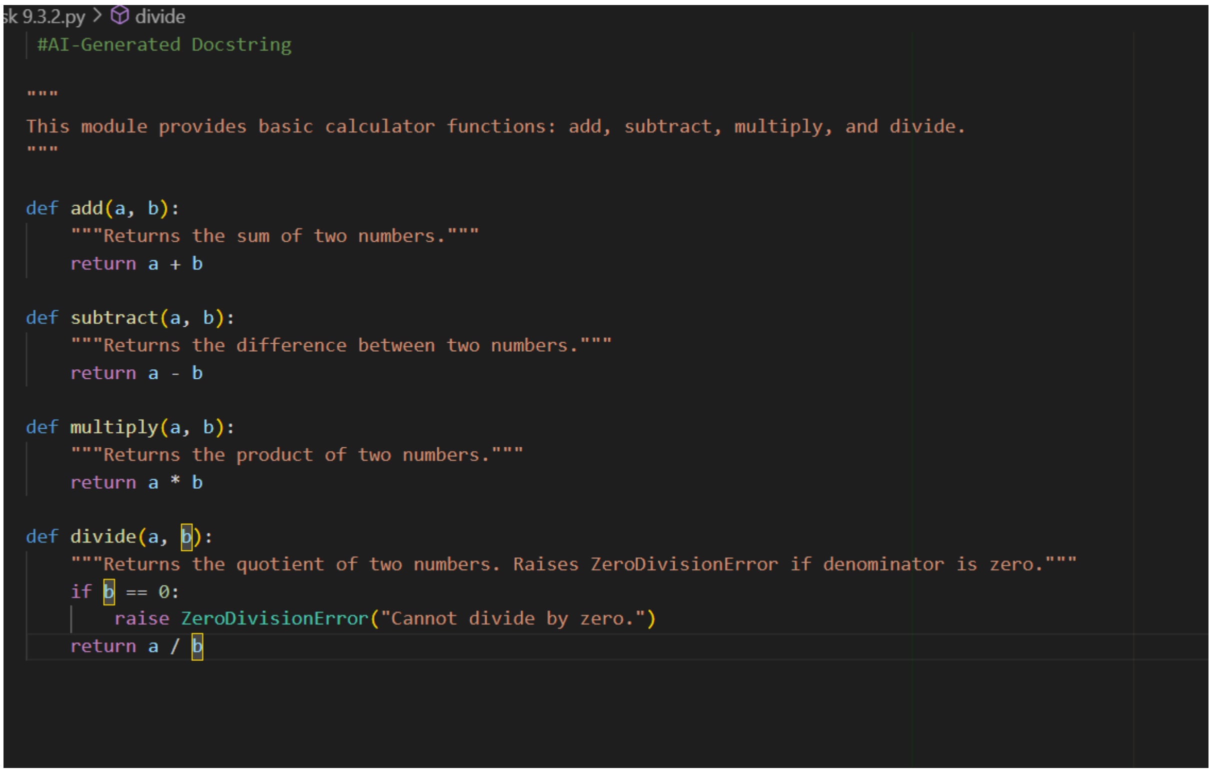
Expected Output#3: Students learn structured documentation for multi-function scripts

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

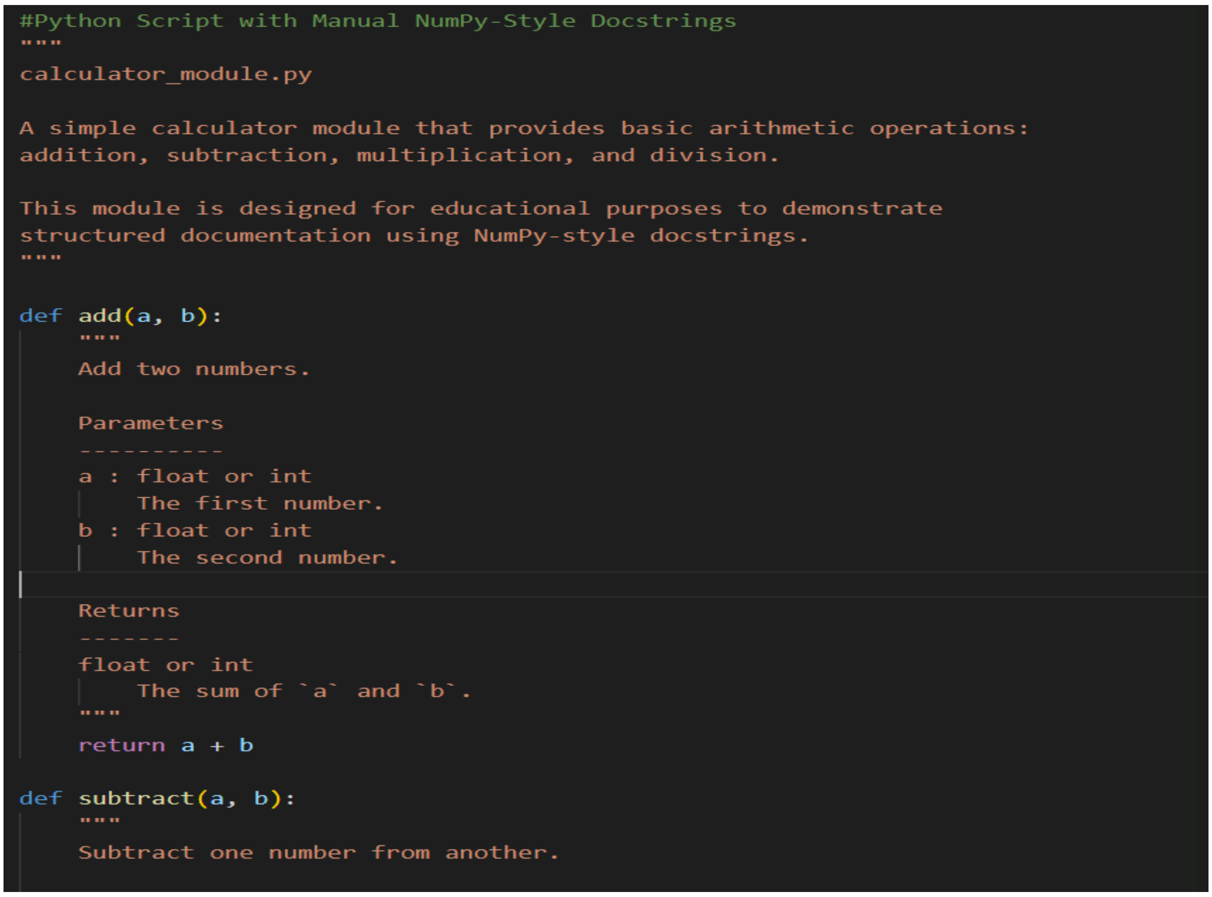
CODE:

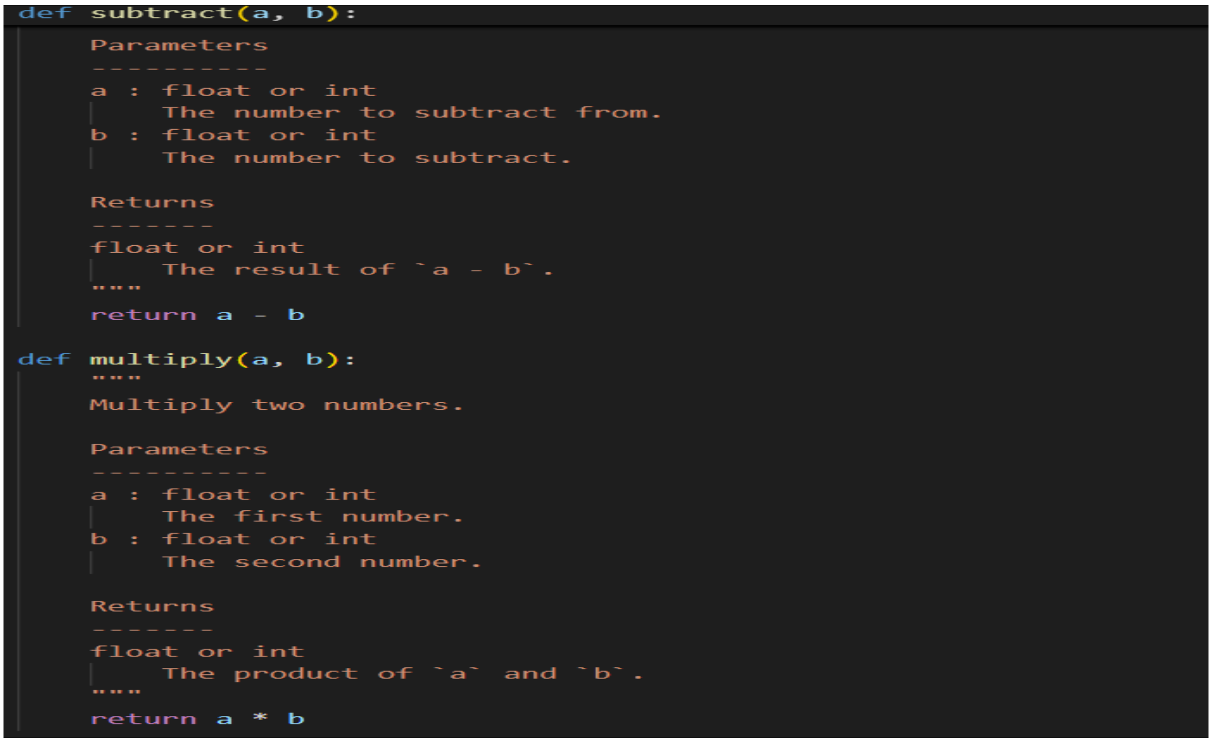
#AI-Generated Docstring



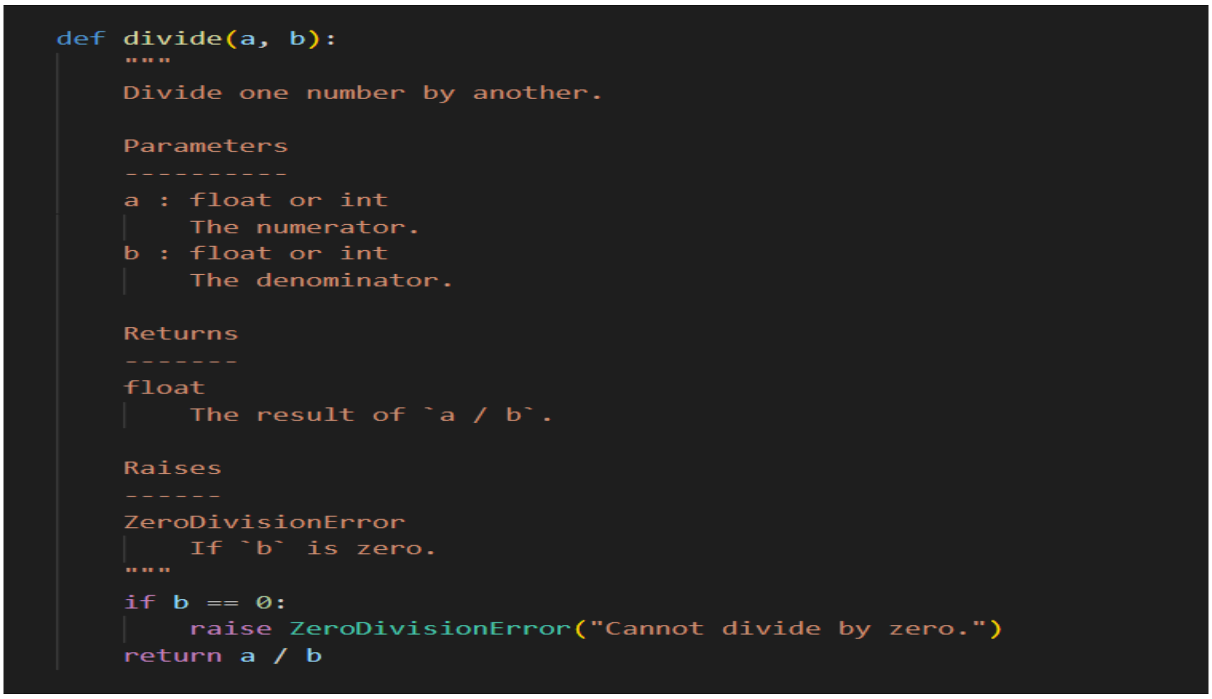
```
sk 9.3.2.py > divide
#AI-Generated Docstring

"""
This module provides basic calculator functions: add, subtract, multiply, and divide.
"""

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

def subtract(a, b):
    """Returns the difference between two numbers."""
    return a - b

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

def divide(a, b):
    """Returns the quotient of two numbers. Raises ZeroDivisionError if denominator is zero."""
    if b == 0:
        raise ZeroDivisionError("Cannot divide by zero.")
    return a / b
```

#Python Script with Manual NumPy-Style Docstrings


```
#Python Script with Manual NumPy-Style Docstrings
"""
calculator_module.py

A simple calculator module that provides basic arithmetic operations:
addition, subtraction, multiplication, and division.

This module is designed for educational purposes to demonstrate
structured documentation using NumPy-style docstrings.
"""

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

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

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

def subtract(a, b):
    """
    Subtract one number from another.
```



```
def subtract(a, b):
    Parameters
    ----------
    a : float or int
        The number to subtract from.
    b : float or int
        The number to subtract.

    Returns
    -------
    float or int
        The result of `a - b`.
    """
    return a - b

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

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

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



```
def divide(a, b):
    """
    Divide one number by another.

    Parameters
    ----------
    a : float or int
        The numerator.
    b : float or int
        The denominator.

    Returns
    -------
    float
        The result of `a / b`.

    Raises
    ------
    ZeroDivisionError
        If `b` is zero.
    """
    if b == 0:
        raise ZeroDivisionError("Cannot divide by zero.")
    return a / b
```

Comparison: Manual vs AI-Generated Docstrings

Feature	Manual Docstrings (NumPy Style)	AI-Generated Docstrings
Structure	Follows NumPy format with Parameters, Returns, etc.	Simple one-liners without formal structure

Detail Level	Specifies types, behavior, and exceptions	Minimal detail, no parameter typing
Educational Value	High—teaches documentation standards	Moderate—good for quick reference
Error Handling	Explicitly documents exceptions	Mentions error but lacks formal Raises section
Audience Suitability	Ideal for learners and collaborators	Suitable for quick solo use