AI Assisted Problem Solving Using Python(ass-3)

2505B04108

Savula Akshitha

Task Description-1: -

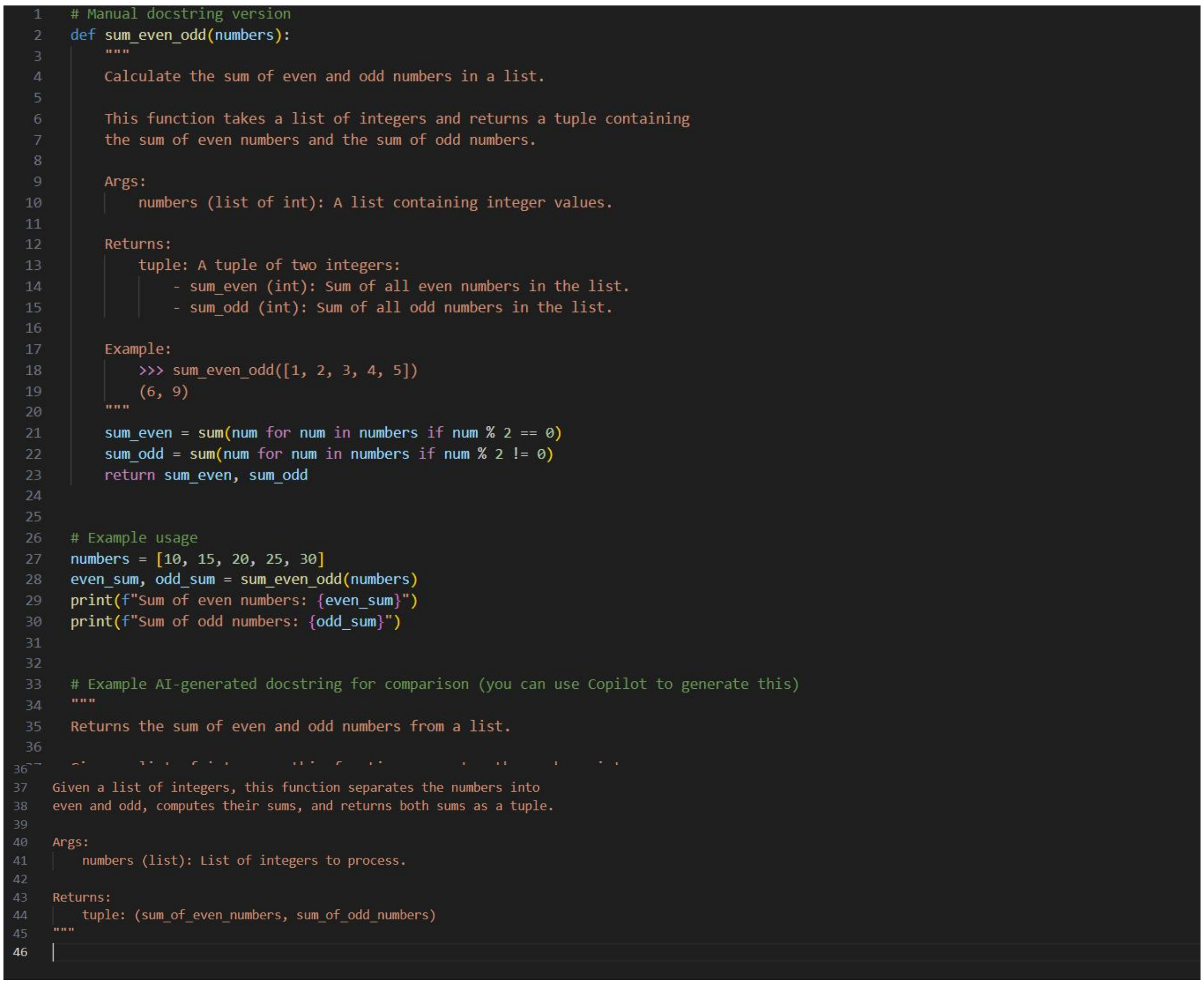
Basic Docstring Generation

• Write python function to return sum of even and odd numbers in the given list.

• Incorporate manual docstring in code with Google Style

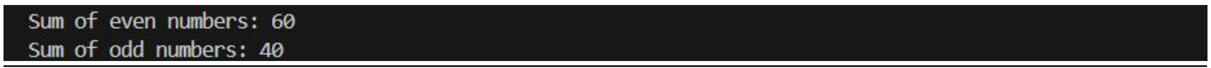
• Use an AI-assisted tool (e.g., Copilot, Cursor AI) to generate a docstring describing the function.

• Compare the AI-generated docstring with your manually written one.



```python
# Manual docstring version
def sum_even_odd(numbers):
    """
    Calculate the sum of even and odd numbers in a list.

    This function takes a list of integers and returns a tuple containing
    the sum of even numbers and the sum of odd numbers.

    Args:
        numbers (list of int): A list containing integer values.

    Returns:
        tuple: A tuple of two integers:
            - sum_even (int): Sum of all even numbers in the list.
            - sum_odd (int): Sum of all odd numbers in the list.

    Example:
        >>> sum_even_odd([1, 2, 3, 4, 5])
        (6, 9)
    """
    sum_even = sum(num for num in numbers if num % 2 == 0)
    sum_odd = sum(num for num in numbers if num % 2 != 0)
    return sum_even, sum_odd


# Example usage
numbers = [10, 15, 20, 25, 30]
even_sum, odd_sum = sum_even_odd(numbers)
print(f"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")


# Example AI-generated docstring for comparison (you can use Copilot to generate this)
"""
Returns the sum of even and odd numbers from a list.

Given a list of integers, this function separates the numbers into
even and odd, computes their sums, and returns both sums as a tuple.

Args:
    numbers (list): List of integers to process.

Returns:
    tuple: (sum_of_even_numbers, sum_of_odd_numbers)
"""
```

Practical Output: -



```
Sum of even numbers: 60
Sum of odd numbers: 40
```

Explanation: -

AI Assisted Problem Solving Using Python(ass-3)
2505B04108

Savula Akshitha

Function Definition
def sum_even_odd(numbers):
This defines a function named sum_even_odd that takes one argument numbers.
numbers is expected to be a list of integers.

---

2. Manual Docstring (Google Style)

```
"""
Calculate the sum of even and odd numbers in a list.

This function takes a list of integers and returns a tuple containing
the sum of even numbers and the sum of odd numbers.

Args:
    numbers (list of int): A list containing integer values.

Returns:
    tuple: A tuple of two integers:
        - sum_even (int): Sum of all even numbers in the list.
        - sum_odd (int): Sum of all odd numbers in the list.

Example:
    >>> sum_even_odd([1, 2, 3, 4, 5])
    (6, 9)
"""
```

Explanation:
Purpose: The first line briefly explains what the function does.
Detailed Description: Explains that it sums even and odd numbers separately and returns them as a tuple.
Args: Specifies the function input type (list of int).
Returns: Explains the output: a tuple with the sum of evens and sum of odds.
Example: Shows how the function works with a real input and what output to expect.
This is very thorough and helpful for someone reading your code for the first time.

AI Assisted Problem Solving Using Python(ass-3)
2505B04108

Savula Akshitha

---

3. Logic of the Function

```
sum_even = sum(num for num in numbers if num % 2 == 0)
sum_odd = sum(num for num in numbers if num % 2 != 0)
return sum_even, sum_odd
```

num % 2 == 0 checks if a number is even.
num % 2 != 0 checks if a number is odd.
sum() adds all numbers that meet the condition.
The function returns a tuple (sum_even, sum_odd).
Example:
For numbers = [10, 15, 20, 25, 30]
Even numbers: 10 + 20 + 30 = 60
Odd numbers: 15 + 25 = 40
Returns: (60, 40)

---

4. Example Usage

```
numbers = [10, 15, 20, 25, 30]
even_sum, odd_sum = sum_even_odd(numbers)
print(f"Sum of even numbers: {even_sum}")
print(f"Sum of odd numbers: {odd_sum}")
```

Calls the function with a sample list.
Stores results in even_sum and odd_sum.
Prints the results in a readable format:
Output:

```
Sum of even numbers: 60
Sum of odd numbers: 40
```

---

5. AI-Generated Docstring

```
"""
Returns the sum of even and odd numbers from a list.

Given a list of integers, this function separates the numbers into
even and odd, computes their sums, and returns both sums as a tuple.
```

AI Assisted Problem Solving Using Python(ass-3)
2505B04108

Savula Akshitha

```
Args:
    numbers (list): List of integers to process.

Returns:
    tuple: (sum_of_even_numbers, sum_of_odd_numbers)
"""
```

Comparison with manual docstring:
Shorter and less detailed.
No example included.
Slightly more general and readable for quick understanding.
Great for speeding up documentation but may need refinement for teaching or detailed explanations.

Task Description-2: -

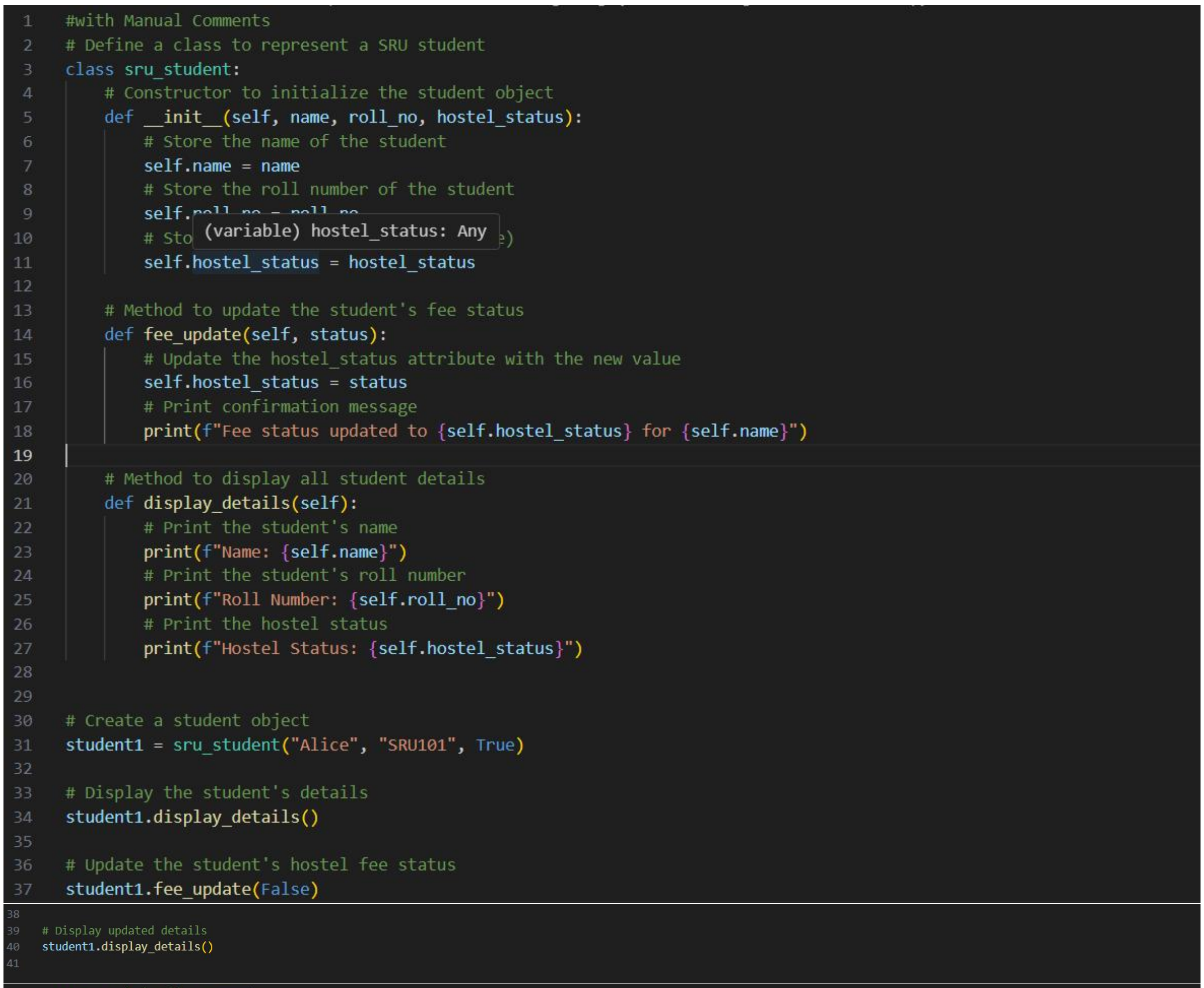
Automatic Inline Comments

• 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.

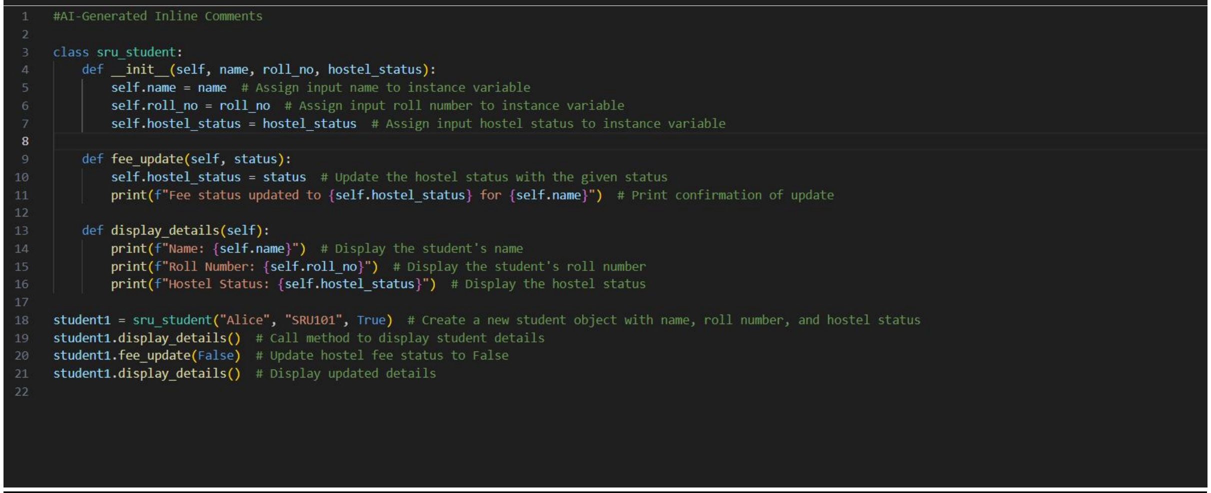
AI Assisted Problem Solving Using Python(ass-3)

2505B04108

Savula Akshitha



```python
#with Manual Comments
# Define a class to represent a SRU student
class sru_student:
    # Constructor to initialize the student object
    def __init__(self, name, roll_no, hostel_status):
        # Store the name of the student
        self.name = name
        # Store the roll number of the student
        self.roll_no = roll_no
        # Sto (variable) hostel_status: Any e)
        self.hostel_status = hostel_status

    # Method to update the student's fee status
    def fee_update(self, status):
        # Update the hostel_status attribute with the new value
        self.hostel_status = status
        # Print confirmation message
        print(f"Fee status updated to {self.hostel_status} for {self.name}")

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


# Create a student object
student1 = sru_student("Alice", "SRU101", True)

# Display the student's details
student1.display_details()

# Update the student's hostel fee status
student1.fee_update(False)

# Display updated details
student1.display_details()

```



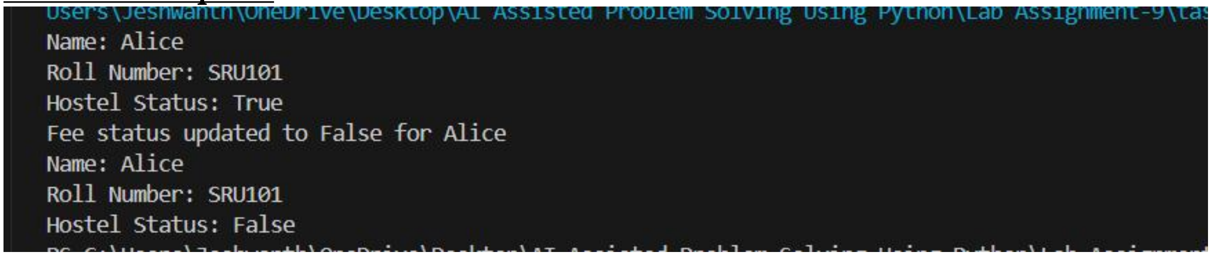
```python
#AI-Generated Inline Comments

class sru_student:
    def __init__(self, name, roll_no, hostel_status):
        self.name = name  # Assign input name to instance variable
        self.roll_no = roll_no  # Assign input roll number to instance variable
        self.hostel_status = hostel_status  # Assign input hostel status to instance variable

    def fee_update(self, status):
        self.hostel_status = status  # Update the hostel status with the given status
        print(f"Fee status updated to {self.hostel_status} for {self.name}")  # Print confirmation of update

    def display_details(self):
        print(f"Name: {self.name}")  # Display the student's name
        print(f"Roll Number: {self.roll_no}")  # Display the student's roll number
        print(f"Hostel Status: {self.hostel_status}")  # Display the hostel status

student1 = sru_student("Alice", "SRU101", True)  # Create a new student object with name, roll number, and hostel status
student1.display_details()  # Call method to display student details
student1.fee_update(False)  # Update hostel fee status to False
student1.display_details()  # Display updated details

```

AI Assisted Problem Solving Using Python(ass-3)

2505B04108

Savula Akshitha

<u>Practical Output: -</u>



```
Users\Jeshwanth\OneDrive\Desktop\AI Assisted Problem Solving Using Python\Lab Assignment-9\tas
Name: Alice
Roll Number: SRU101
Hostel Status: True
Fee status updated to False for Alice
Name: Alice
Roll Number: SRU101
Hostel Status: False
```

<u>Explanation: -</u>

Comparison

Aspect	Manual Comments	AI-Generated Comments
Detail	More descriptive, explains reasoning behind steps	More concise, explains what each line does
Readability	Easy for beginners to understand	Clear and straightforward, good for quick reading
Usefulness	Good for teaching or documentation purposes	Good for code review or quick understanding
Effort	Requires manual writing	Automatically generated by AI tool

Class Definition

class sru_student:

This line defines a new class called sru_student.

Classes are used to model objects—in this case, a student at SRU with attributes and methods.

---

2. Constructor Method (__init__)

def __init__(self, name, roll_no, hostel_status):

The __init__ method is called when a new student object is created.

AI Assisted Problem Solving Using Python(ass-3)

2505B04108

Savula Akshitha

It takes three arguments besides self: name, roll_no, and hostel_status.

```
self.name = name
self.roll_no = roll_no
self.hostel_status = hostel_status
```

These lines store the input values as attributes of the student object so they can be used later.

self refers to the instance of the object itself.

Manual comments: Explain the purpose of each line clearly.

AI-generated comments: Usually just say what is happening, e.g., "Assign input name to instance variable."

---

3. Fee Update Method

```
def fee_update(self, status):
    self.hostel_status = status
    print(f"Fee status updated to {self.hostel_status} for {self.name}")
```

fee_update changes the student's hostel fee status.

status is the new fee status (True or False).

First line updates the hostel_status attribute.

Second line prints a confirmation message.

Manual comments: Explain why we do this (update the status and confirm).

AI-generated comments: Explain what is done, often very straightforward.

---

4. Display Details Method

```
def display_details(self):
    print(f"Name: {self.name}")
    print(f"Roll Number: {self.roll_no}")
    print(f"Hostel Status: {self.hostel_status}")
```

This method prints all attributes of the student.

Easy way to check the current state of the object.

Manual comments: Can describe each attribute's meaning.

AI-generated comments: Usually repeat the obvious, e.g., "Display the student's name."

---

5. Creating and Using the Student Object

AI Assisted Problem Solving Using Python(ass-3)
2505B04108

Savula Akshitha

student1 = sru_student("Alice", "SRU101", True)
Creates a new student named "Alice" with roll number SRU101 and hostel_status = True.
student1.display_details()
student1.fee_update(False)
student1.display_details()
First call prints initial details.
Second call updates the fee status to False and prints confirmation.
Third call prints updated details.

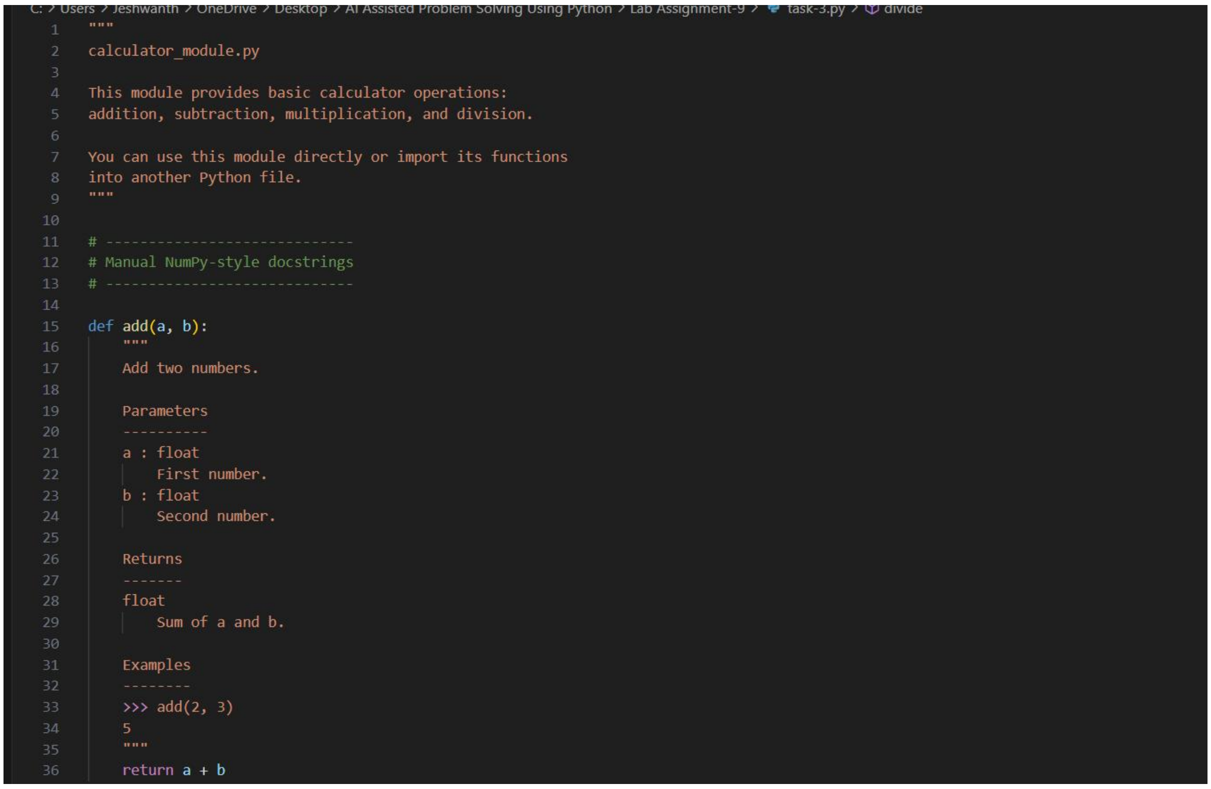
Task Description-3: -

• 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.

AI Assisted Problem Solving Using Python(ass-3)

2505B04108

Savula Akshitha



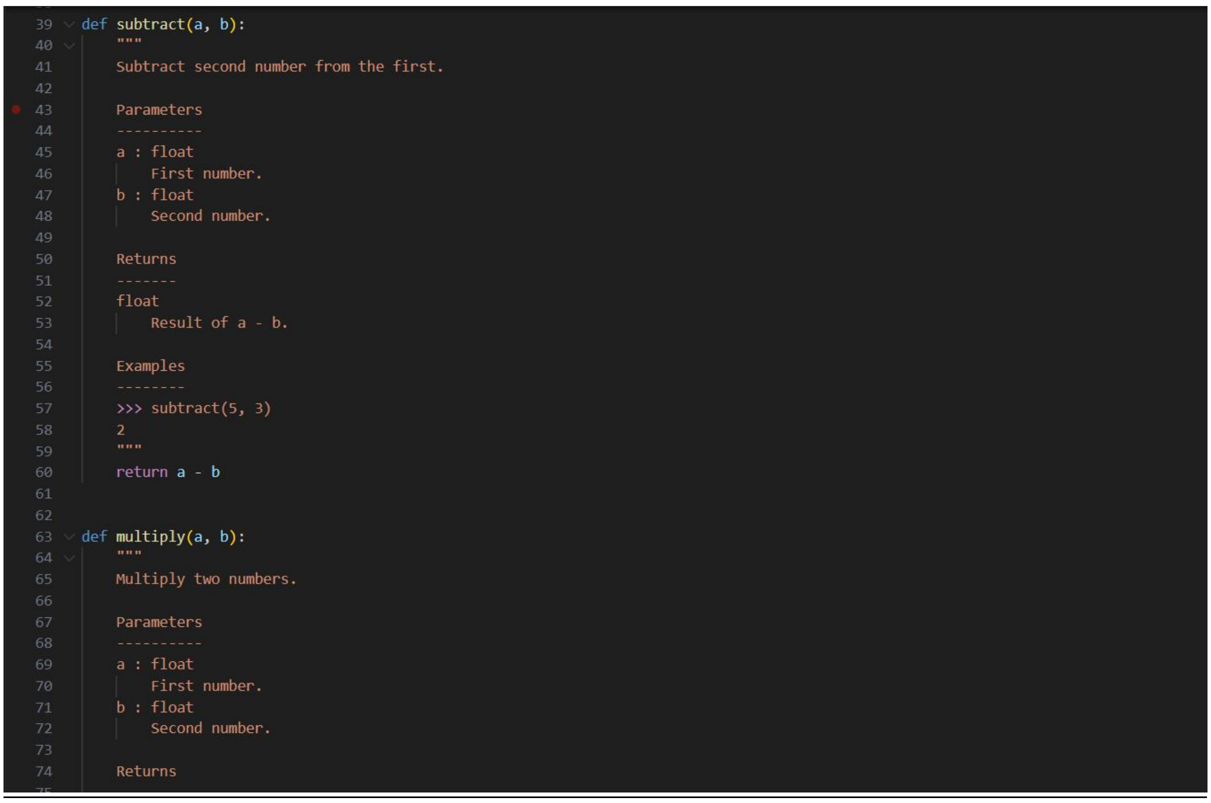
C: > Users > Jeshwanth > OneDrive > Desktop > AI Assisted Problem Solving Using Python > Lab Assignment-9 > task-3.py > divide

```python
"""
calculator_module.py

This module provides basic calculator operations:
addition, subtraction, multiplication, and division.

You can use this module directly or import its functions
into another Python file.
"""

# ----------------------------
# Manual NumPy-style docstrings
# ----------------------------

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

    Parameters
    ----------
    a : float
        First number.
    b : float
        Second number.

    Returns
    -------
    float
        Sum of a and b.

    Examples
    --------
    >>> add(2, 3)
    5
    """
    return a + b
```

AI Assisted Problem Solving Using Python(ass-3)

2505B04108

Savula Akshitha

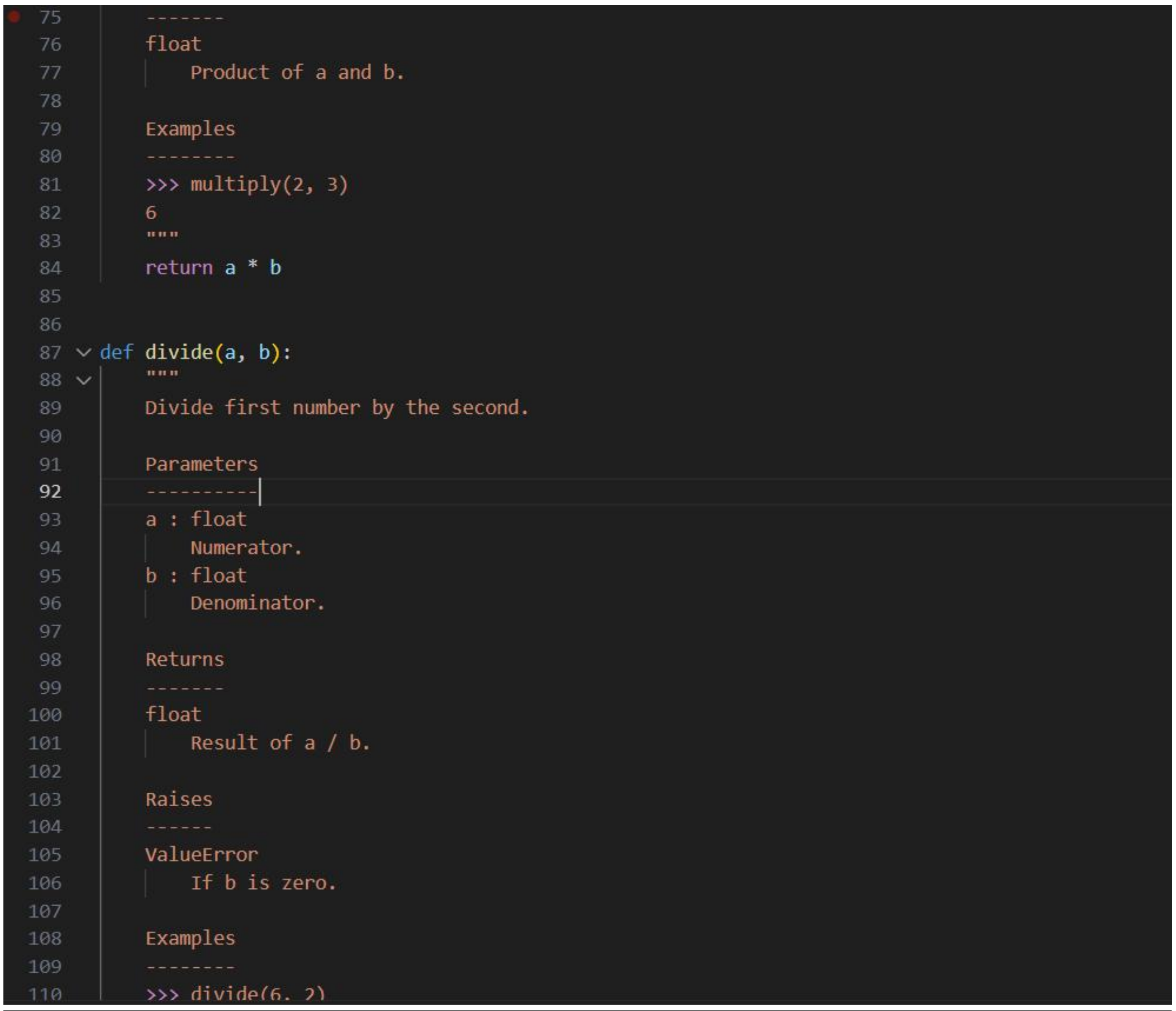


```python
def subtract(a, b):
    """
    Subtract second number from the first.

    Parameters
    ----------
    a : float
        First number.
    b : float
        Second number.

    Returns
    -------
    float
        Result of a - b.

    Examples
    --------
    >>> subtract(5, 3)
    2
    """
    return a - b


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

    Parameters
    ----------
    a : float
        First number.
    b : float
        Second number.

    Returns
```

AI Assisted Problem Solving Using Python(ass-3)

2505B04108

Savula Akshitha

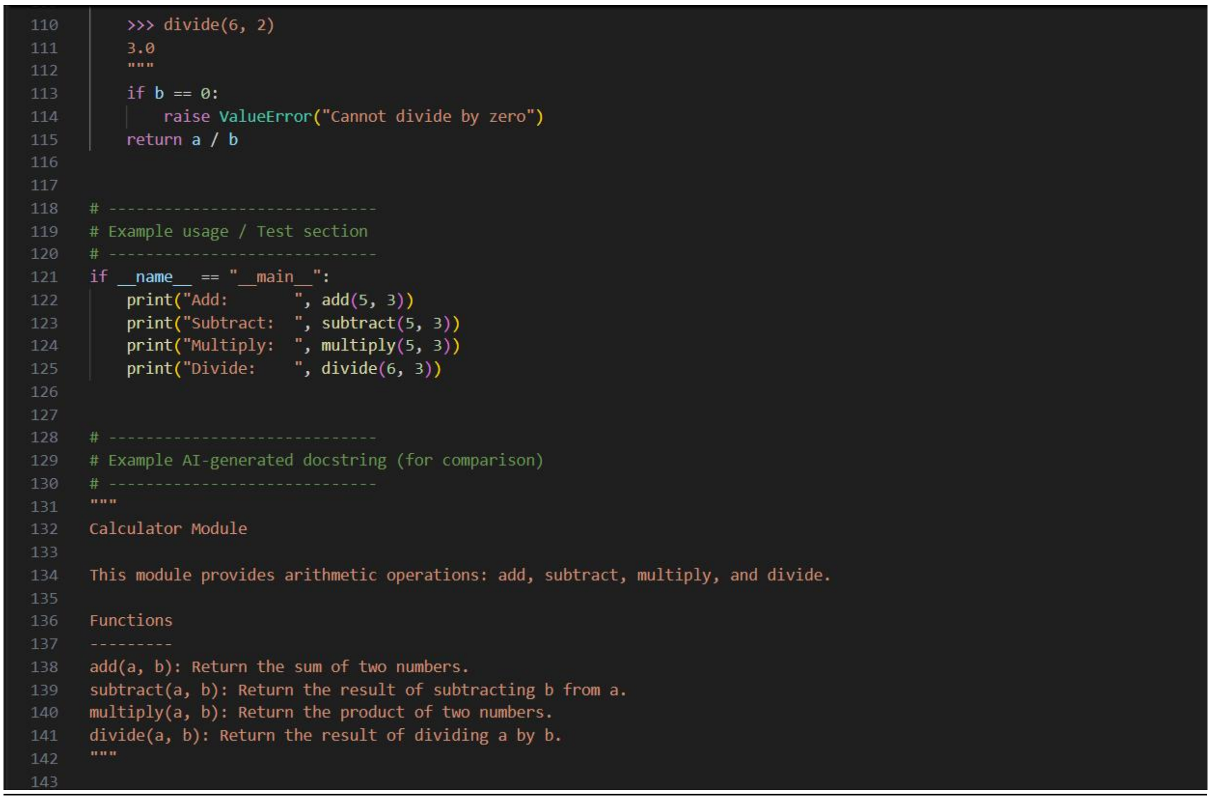


```
    -------
    float
        Product of a and b.

    Examples
    --------
    >>> multiply(2, 3)
    6
    """
    return a * b


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

    Parameters
    ----------
    a : float
        Numerator.
    b : float
        Denominator.

    Returns
    -------
    float
        Result of a / b.

    Raises
    ------
    ValueError
        If b is zero.

    Examples
    --------
    >>> divide(6, 2)
```

AI Assisted Problem Solving Using Python(ass-3)

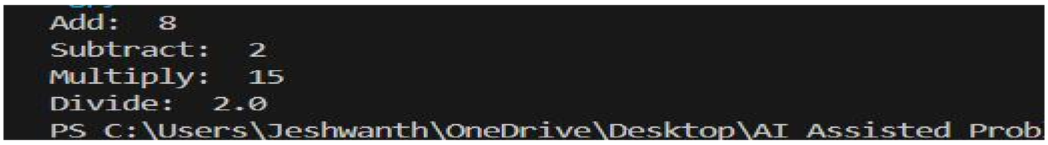
2505B04108

Savula Akshitha



```
    >>> divide(6, 2)
    3.0
    """
    if b == 0:
        raise ValueError("Cannot divide by zero")
    return a / b


# ----------------------------
# Example usage / Test section
# ----------------------------
if __name__ == "__main__":
    print("Add:       ", add(5, 3))
    print("Subtract:  ", subtract(5, 3))
    print("Multiply:  ", multiply(5, 3))
    print("Divide:    ", divide(6, 3))


# ----------------------------
# Example AI-generated docstring (for comparison)
# ----------------------------
"""
Calculator Module

This module provides arithmetic operations: add, subtract, multiply, and divide.

Functions
---------
add(a, b): Return the sum of two numbers.
subtract(a, b): Return the result of subtracting b from a.
multiply(a, b): Return the product of two numbers.
divide(a, b): Return the result of dividing a by b.
"""
```

Practical Output: -



```
Add:    8
Subtract:   2
Multiply:   15
Divide:   2.0
PS C:\Users\Jeshwanth\OneDrive\Desktop\AI Assisted Prob
```

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

Explanation: -

Module Overview

"""

calculator_module.py

This module provides basic calculator operations:

AI Assisted Problem Solving Using Python(ass-3)
2505B04108

Savula Akshitha

addition, subtraction, multiplication, and division.
"""

- This is a **module-level docstring**.
- Explains what the module does overall: it provides four basic arithmetic operations.

Manual Docstring: Structured, clear, educational.
AI-Generated: Similar, concise, may skip extra details.

---

2. Function Definitions
Add Function

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

    Parameters
    ----------
    a : float
        First number.
    b : float
        Second number.

    Returns
    -------
    float
        Sum of a and b.

    Examples
    --------
    >>> add(2, 3)
    5
    """
    return a + b
```

Explanation:

AI Assisted Problem Solving Using Python(ass-3)

2505B04108

Savula Akshitha

- **Parameters:** Lists the inputs with type and description.
- **Returns:** Specifies the type and meaning of output.
- **Examples:** Shows usage.

AI-generated version might look like:

```
"""
Return the sum of two numbers.

Parameters:
    a (float): First number.
    b (float): Second number.

Returns:
    float: Sum of a and b.
"""
```

Difference:

- Manual docstring includes **Examples** section.
- AI docstring is shorter, explains **what it does** but no usage example.

Subtract, Multiply, Divide Functions

- Same structure:
    - **Manual:** Parameters, Returns, Examples, and for divide a **Raises** section if dividing by zero.
    - **AI:** Mostly Parameters and Returns, sometimes includes Raises.

Divide Example with Error Handling:

```
def divide(a, b):
    if b == 0:
        raise ValueError("Cannot divide by zero")
    return a / b
```

- Manual docstring explains **why the ValueError is raised**.
- AI docstring usually mentions the exception briefly.

---

3. Example Usage

```
if __name__ == "__main__":
```

AI Assisted Problem Solving Using Python(ass-3)
2505B04108

Savula Akshitha

```
print("Add: ", add(5, 3))
print("Subtract: ", subtract(5, 3))
print("Multiply: ", multiply(5, 3))
print("Divide: ", divide(6, 3))
```

- Tests all four functions.
- Demonstrates **how the module works when run directly**.
- Output will be:

```
Add:  8
Subtract:  2
Multiply:  15
Divide:  2.0
```

---

4. Comparison of Manual vs AI-Generated Docstrings

Feature	**Manual Docstring (NumPy)**	**AI-Generated Docstring**
Style	NumPy style: Parameters, Returns, Examples, Raises	Simple, generic documentation
Examples	Included for clarity	Usually missing
Detail	Describes reasoning, error handling, and output	Describes what function does only
Usefulness	Good for learning, teaching, or official docs	Quick, saves time, good for code review
Effort	Manual writing required	Auto-generated

Key Insight:

- **Manual docstrings** are **more structured and educational**, perfect for formal documentation.
- **AI docstrings** are **faster to generate**, enough for quick understanding but less detailed.

☑ **Takeaway for Students:**

- Structured documentation (NumPy style) makes multi-function scripts **easy to read, maintain, and use**.

AI Assisted Problem Solving Using Python(ass-3)
2505B04108

Savula Akshitha

- *AI tools can help save time but* **may require manual refinement** *for full clarity and examples.*