ASSIGNMENT – 7.3

2303A51060

Batch-10

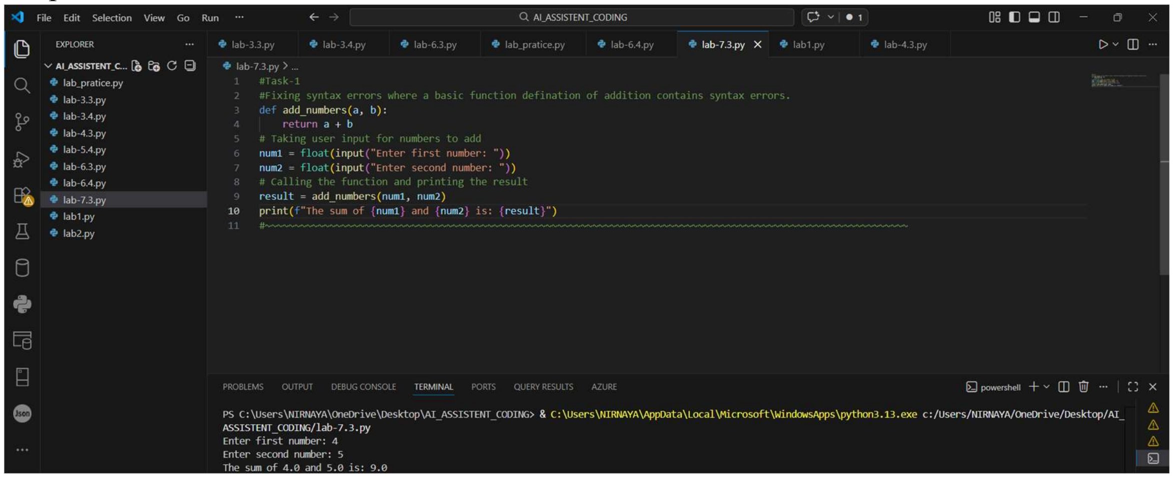
Task-1

Prompt: Fixing syntax errors where a basic function definition of addition contains syntax errors.

Code :

```
def add_numbers(a, b):
    return a + b
# Taking user input for numbers to add
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
# Calling the function and printing the result
result = add_numbers(num1, num2)
print(f"The sum of {num1} and {num2} is: {result}")
```

Output :



```
#Task-1
#Fixing syntax errors where a basic function defination of addition contains syntax errors.
def add_numbers(a, b):
    return a + b
# Taking user input for numbers to add
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
# Calling the function and printing the result
result = add_numbers(num1, num2)
print(f"The sum of {num1} and {num2} is: {result}")
#
```

```
PS C:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTENT_CODING> & C:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRNAYA/OneDrive/Desktop/AI_
ASSISTENT_CODING/lab-7.3.py
Enter first number: 4
Enter second number: 5
The sum of 4.0 and 5.0 is: 9.0
```

Code Analysis :

- A function add_numbers(a, b) is defined to perform addition.

- Syntax issues are corrected to ensure proper function definition and return statement.
- User input is taken and converted to float for numeric calculation.
- Function is called with user inputs to compute the sum.
- Result is displayed using formatted output for clarity.

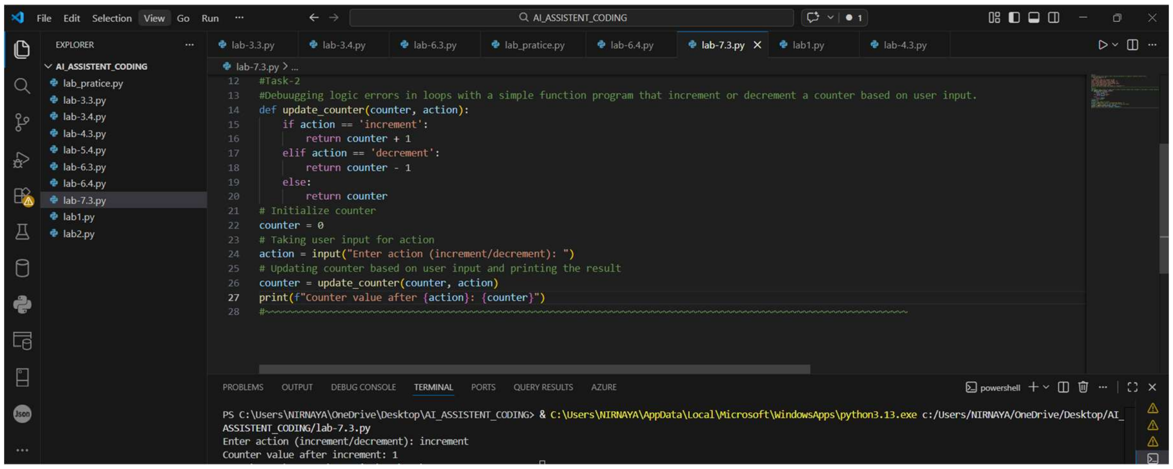
Task-2

Prompt: Debugging logic errors in loops with a simple function program that increments or decrements a counter based on user input.

Code:

```
def update_counter(counter, action):

    if action == 'increment':

        return counter + 1

    elif action == 'decrement':

        return counter - 1

    else:

        return counter

# Initialize counter

counter = 0

# Taking user input for action

action = input("Enter action (increment/decrement): ")

# Updating counter based on user input and printing the result

counter = update_counter(counter, action)

print(f"Counter value after {action}: {counter}")
```

Output :



```python
#Task-2
#Debuugging logic errors in loops with a simple function program that increment or decrement a counter based on user input.
def update_counter(counter, action):
    if action == 'increment':
        return counter + 1
    elif action == 'decrement':
        return counter - 1
    else:
        return counter
# Initialize counter
counter = 0
# Taking user input for action
action = input("Enter action (increment/decrement): ")
# Updating counter based on user input and printing the result
counter = update_counter(counter, action)
print(f"Counter value after {action}: {counter}")
#
```

```
PS C:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTENT_CODING> & C:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRNAYA/OneDrive/Desktop/AI_
ASSISTENT_CODING/lab-7.3.py
Enter action (increment/decrement): increment
Counter value after increment: 1
```

Code Analysis :

 Function update_counter() updates counter based on user action.

 Conditional statements handle increment and decrement operations.

 Default case returns the same counter if action is invalid.

 Counter variable is initialized before processing.

 Demonstrates debugging of logical flow in conditional statements.

Task-3

Prompt : Handling runtime errors which function performs division without validations.

Code:

```python
def safe_division(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed."
# Taking user input for numbers to divide
num1 = float(input("Enter numerator: "))
```

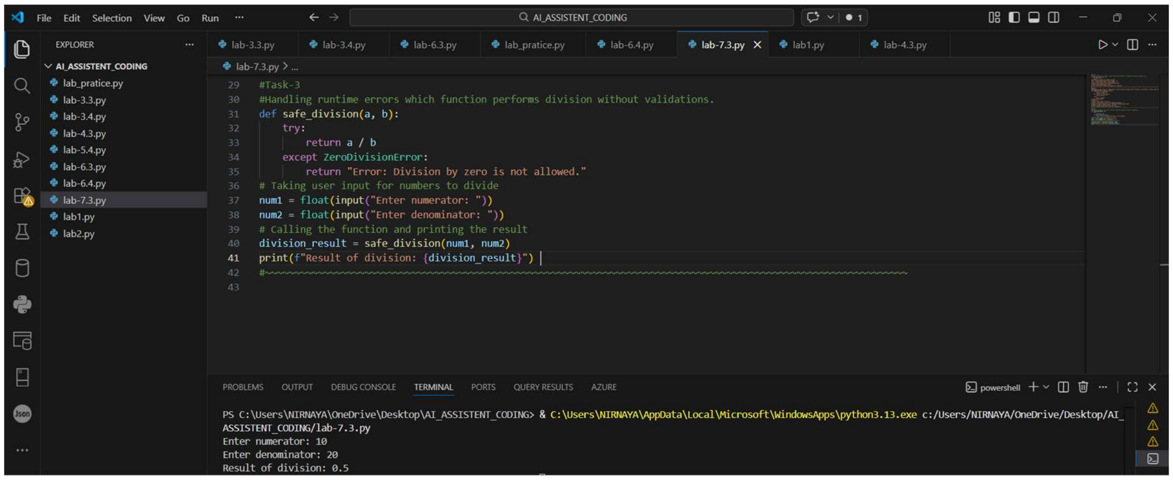
num2 = float(input("Enter denominator: "))

Calling the function and printing the result

division_result = safe_division(num1, num2)

print(f"Result of division: {division_result}")

Output :



```python
#Task-3
#Handling runtime errors which function performs division without validations.
def safe_division(a, b):
    try:
        return a / b
    except ZeroDivisionError:
        return "Error: Division by zero is not allowed."
# Taking user input for numbers to divide
num1 = float(input("Enter numerator: "))
num2 = float(input("Enter denominator: "))
# Calling the function and printing the result
division_result = safe_division(num1, num2)
print(f"Result of division: {division_result}")
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

```
PS C:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTENT_CODING> & C:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRNAYA/OneDrive/Desktop/AI_
ASSISTENT_CODING/lab-7.3.py
Enter numerator: 10
Enter denominator: 20
Result of division: 0.5
```

Code Analysis :

- Function safe_division() performs division inside a try block.
- ZeroDivisionError is handled using except to avoid program crash.
- User inputs are converted to float for accurate division.
- Function returns either result or error message.
- Demonstrates runtime error handling using exception handling.

Task-4

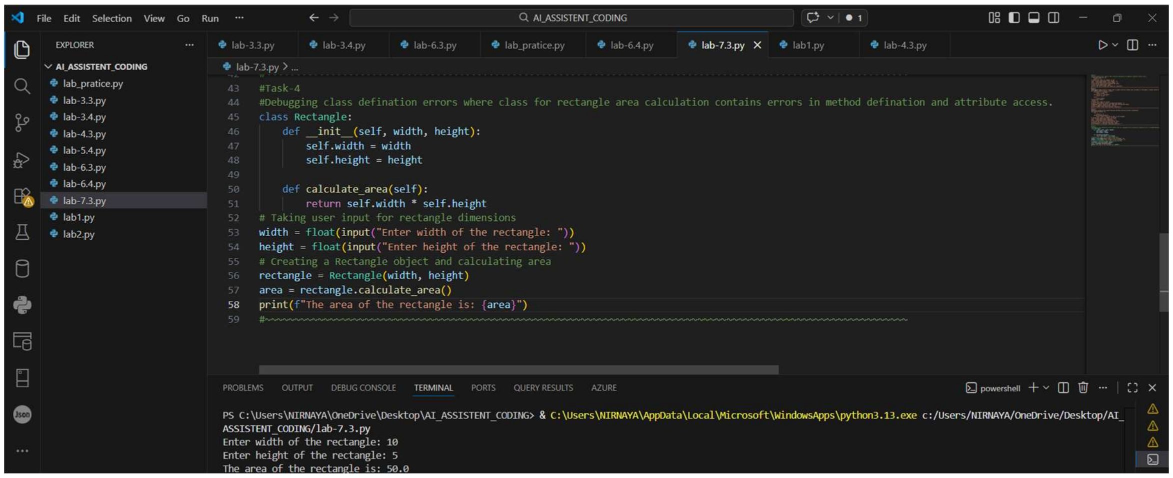
Prompt: Debugging class definition errors where the class for rectangle area calculation contains errors in method definition and attribute access.

Code:

class Rectangle:

```
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def calculate_area(self):
        return self.width * self.height
# Taking user input for rectangle dimensions
width = float(input("Enter width of the rectangle: "))
height = float(input("Enter height of the rectangle: "))
# Creating a Rectangle object and calculating area
rectangle = Rectangle(width, height)
area = rectangle.calculate_area()
print(f"The area of the rectangle is: {area}")
```

Output :



```
#Task-4
#Debugging class defination errors where class for rectangle area calculation contains errors in method defination and attribute access.
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    def calculate_area(self):
        return self.width * self.height
# Taking user input for rectangle dimensions
width = float(input("Enter width of the rectangle: "))
height = float(input("Enter height of the rectangle: "))
# Creating a Rectangle object and calculating area
rectangle = Rectangle(width, height)
area = rectangle.calculate_area()
print(f"The area of the rectangle is: {area}")
#
```

```
PS C:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTENT_CODING> & C:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRNAYA/OneDrive/Desktop/AI_
ASSISTENT_CODING/lab-7.3.py
Enter width of the rectangle: 10
Enter height of the rectangle: 5
The area of the rectangle is: 50.0
```

Code Analysis :

- Rectangle class is created with width and height attributes.
- Constructor initializes object properties correctly.
- calculate_area() method returns area using instance variables.

- Object is created using user input values.

- Demonstrates debugging of method definition and attribute usage.

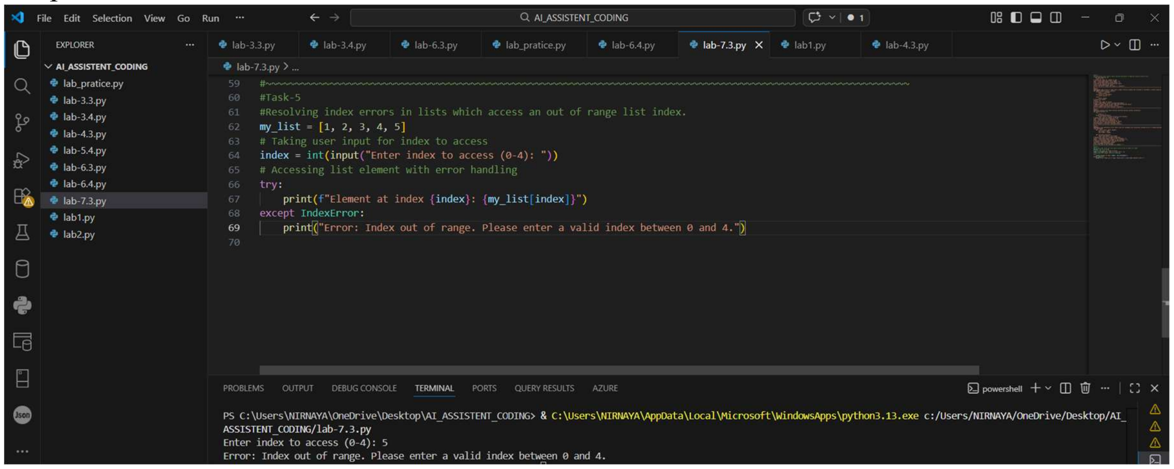
Task 5

Prompt: Resolving index errors in lists that access an out-of-range list index.

Code:

```
my_list = [1, 2, 3, 4, 5]
# Taking user input for index to access
index = int(input("Enter index to access (0-4): "))
# Accessing list element with error handling
try:
    print(f"Element at index {index}: {my_list[index]}")
except IndexError:
    print("Error: Index out of range. Please enter a valid index between 0 and 4.")
```

Output :



```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#Task-5
#Resolving index errors in lists which access an out of range list index.
my_list = [1, 2, 3, 4, 5]
# Taking user input for index to access
index = int(input("Enter index to access (0-4): "))
# Accessing list element with error handling
try:
    print(f"Element at index {index}: {my_list[index]}")
except IndexError:
    print("Error: Index out of range. Please enter a valid index between 0 and 4.")
```

```
PS C:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTENT_CODING> & C:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRNAYA/OneDrive/Desktop/AI_
ASSISTENT_CODING/lab-7.3.py
Enter index to access (0-4): 5
Error: Index out of range. Please enter a valid index between 0 and 4.
```

Code Analysis :

- A list is defined with fixed elements.

- User provides an index to access list elements.

- Access operation is placed inside a try block.

- IndexError is handled using except to prevent crash.
- Program ensures safe list access and user-friendly error message.