

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: "))

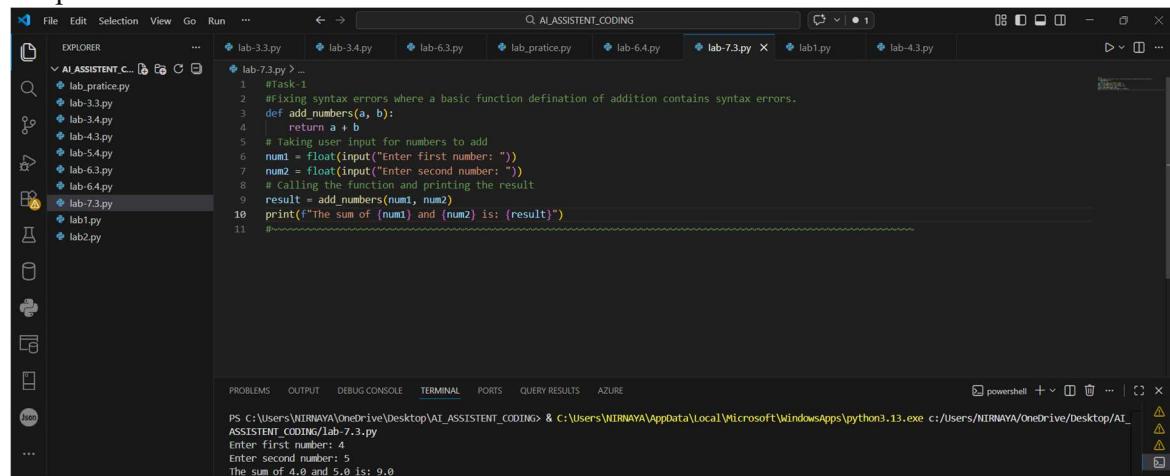
num2 = float(input("Enter second number: "))

# Calling the function and printing the result

result = add_numbers(num1, num2)

print(f"The sum of {num1} and {num2} is: {result}")
```

Output :



The screenshot shows the Visual Studio Code interface. The left sidebar displays a file tree with several Python files. The main editor window shows the code for 'lab-7.3.py'. The terminal at the bottom shows the command to run the file and the resulting output:

```
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 :

The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar: Shows a folder named "AI_ASSISTANT_CODING" containing files: lab-practice.py, lab-3.3.py, lab-3.4.py, lab-6.3.py, lab_practice.py, lab-6.4.py, lab-7.3.py (selected), lab1.py, and lab2.py.
- CODE EDITOR**: The active file is "lab-7.3.py". The code defines a function "update_counter" that takes a counter and an action ("increment" or "decrement"). It initializes a counter to 0, takes user input for action, and prints the result after updating the counter.
- TERMINAL**: Shows the command "PS C:\Users\NIRNAYA\OneDrive\Desktop\AI_ASSISTANT_CODING & C:\Users\NIRNAYA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRNAYA/OneDrive/Desktop/AI_ASSISTANT_CODING/lab-7.3.py" followed by the output: "Enter action (increment/decrement): increment" and "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:

```
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 :

```

File Edit Selection View Go Run ...
AI_ASSISTENT_CODING
EXPLORER
lab-3.3.py lab-3.4.py lab-6.3.py lab_practice.py lab-6.4.py lab-7.3.py X lab1.py lab-4.3.py ...
lab-3.3.py > ...
29 #Task-3
30 #Handling runtime errors which function performs division without validations.
31 def safe_division(a, b):
32     try:
33         return a / b
34     except ZeroDivisionError:
35         return "error: Division by zero is not allowed."
36 # Taking user input for numbers to divide
37 num1 = float(input("Enter numerator: "))
38 num2 = float(input("Enter denominator: "))
39 # Calling the function and printing the result
40 division_result = safe_division(num1, num2)
41 print(f"Result of division: {division_result}")
42 #
43
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS QUERY RESULTS AZURE
powershell + v x
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 :

The screenshot shows the Visual Studio Code interface. The left sidebar has a tree view labeled 'EXPLORER' with several Python files listed under 'AI_ASSISTANT_CODING'. The main code editor window displays the file 'lab-7.3.py' containing the provided Python code. Below the code editor is a terminal window showing the execution of the script and its output:

```

PS C:\Users\NIRWANA\OneDrive\Desktop\AI_ASSISTANT_CODING> & C:\Users\NIRWANA\AppData\Local\Microsoft\WindowsApps\python3.13.exe c:/Users/NIRWANA/OneDrive/Desktop/AI_ASSISTANT_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

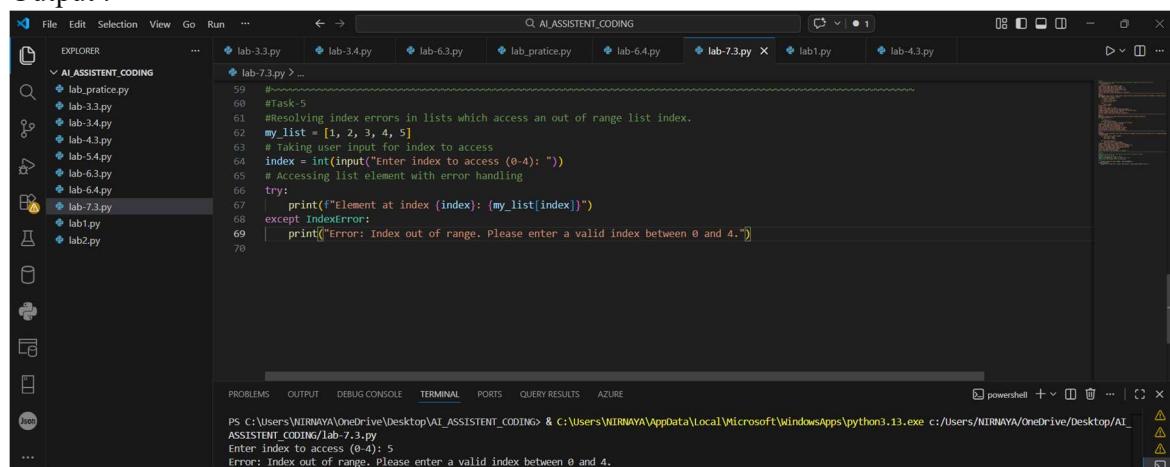
try:

    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 :



```
File Edit Selection View Go Run ... ← → 🔍 AI_ASSISTANT_CODING 🌐
EXPLORER lab-3.3.py ... lab-3.4.py lab-6.3.py lab_practice.py lab-6.4.py lab-7.3.py X lab1.py lab-4.3.py
59 #Task.5
60 #Resolving index errors in lists which access an out of range list index.
61 my_list = [1, 2, 3, 4, 5]
62 # Taking user input for index to access
63 index = int(input("Enter index to access (0-4): "))
64 # Accessing list element with error handling
65 try:
66     print("Element at index {index}: {my_list[index]}")
67 except IndexError:
68     print("Error: Index out of range. Please enter a valid index between 0 and 4.")
```

The screenshot shows a code editor interface with several tabs open. The active tab is 'lab-7.3.py'. The code itself is a simple script that defines a list 'my_list' with elements 1 through 5. It then prompts the user for an index (between 0 and 4) and prints the corresponding list element. If the user enters an index outside this range, it catches the 'IndexError' and prints an error message instead. The terminal below the code editor shows the execution of the script, the user's input of '5', and the resulting error message '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.