AI LAB-10.3

NAME:A.SREEMANTH REDDY

ROLLNO:2403A510G1

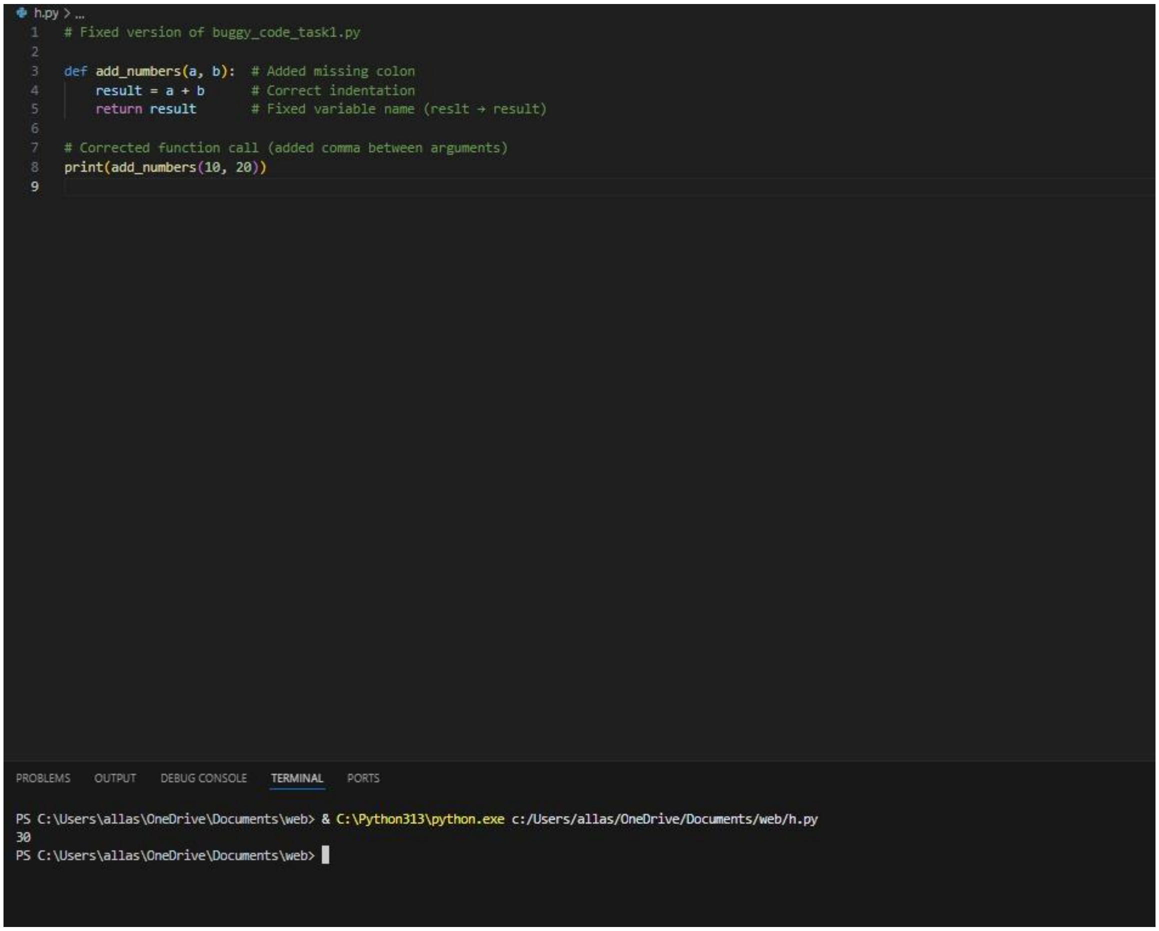
BATCH:06

Task 1: Syntax and Error Detection

Prompt:

Identify and fix syntax, indentation, and variable errors in the following Python script. Explain what changes you made and why.

Code:



```python
# Fixed version of buggy_code_task1.py

def add_numbers(a, b):  # Added missing colon
    result = a + b      # Correct indentation
    return result       # Fixed variable name (reslt → result)

# Corrected function call (added comma between arguments)
print(add_numbers(10, 20))
```

```
PS C:\Users\allas\OneDrive\Documents\web> & C:\Python313\python.exe c:/Users/allas/OneDrive/Documents/web/h.py
30
PS C:\Users\allas\OneDrive\Documents\web> 
```

Observation:

- Added missing colon after function definition.
- Fixed variable name (reslt → result).
- Corrected function call (added comma).
- Correct indentation.

Output: 30

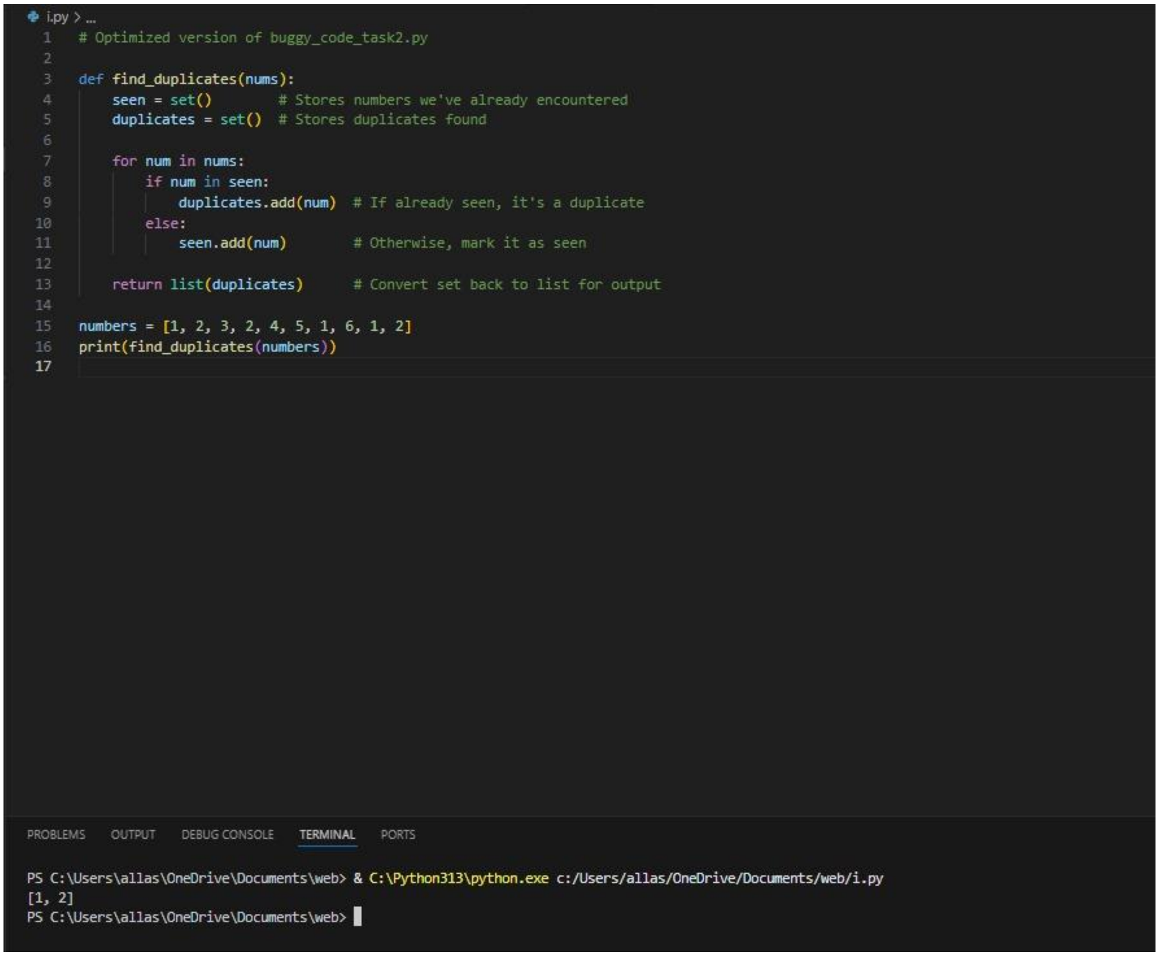
AI LAB-10.3

Task 2: Logical and Performance Issue Review

Prompt:

Optimize the following function to detect duplicates more efficiently while keeping results correct. Explain the optimization.

Code:



```python
# Optimized version of buggy_code_task2.py

def find_duplicates(nums):
    seen = set()        # Stores numbers we've already encountered
    duplicates = set()  # Stores duplicates found

    for num in nums:
        if num in seen:
            duplicates.add(num)  # If already seen, it's a duplicate
        else:
            seen.add(num)        # Otherwise, mark it as seen

    return list(duplicates)      # Convert set back to list for output

numbers = [1, 2, 3, 2, 4, 5, 1, 6, 1, 2]
print(find_duplicates(numbers))
```

```
PS C:\Users\allas\OneDrive\Documents\web> & C:\Python313\python.exe c:/Users/allas/OneDrive/Documents/web/i.py
[1, 2]
PS C:\Users\allas\OneDrive\Documents\web>
```

Observation:

- Nested loop replaced with set-based approach (O(n) instead of O(n^2)).
- Returns unique duplicates.

Output: [1, 2]

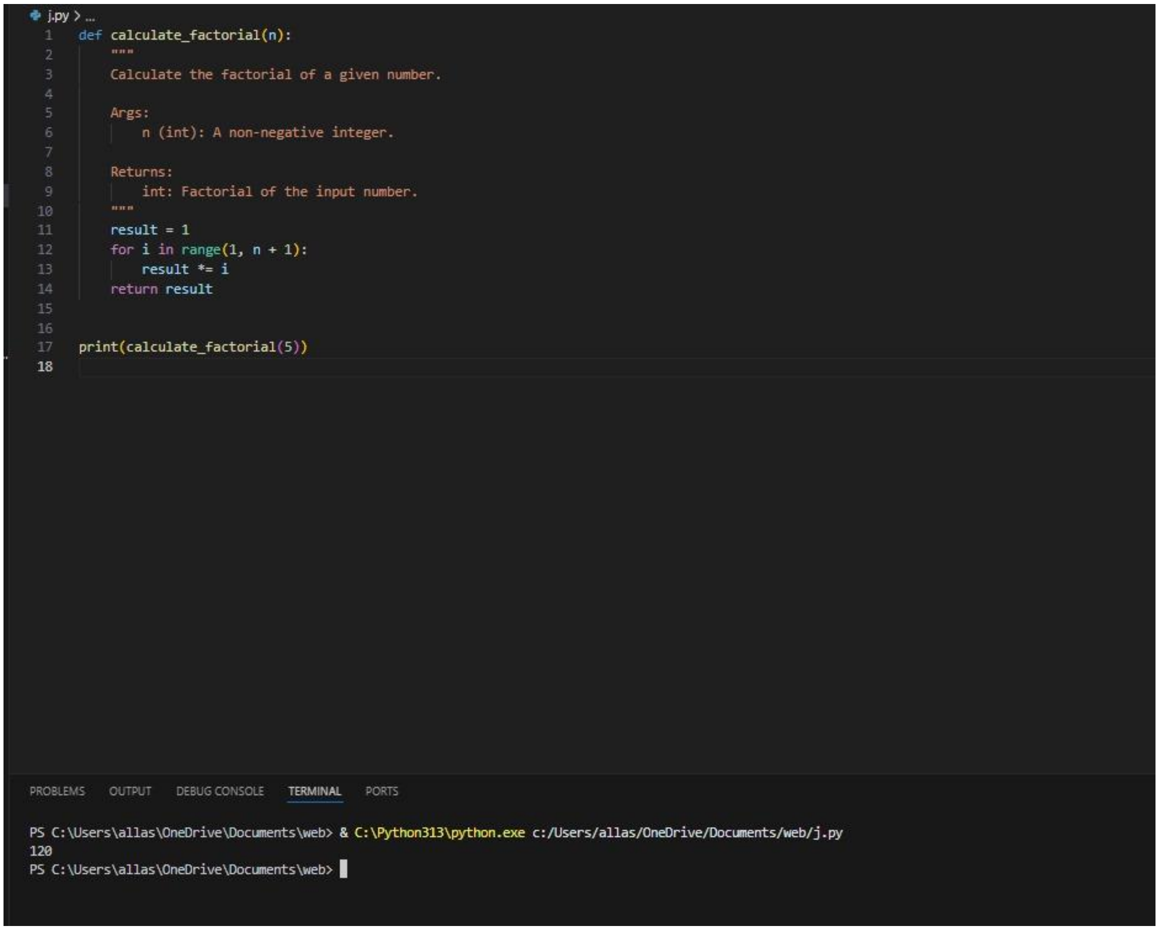
Task 3: Code Refactoring for Readability

Prompt:

Refactor the following function into clean, PEP 8–compliant, well-structured code with meaningful names and a docstring.

AI LAB-10.3

Code:



```python
def calculate_factorial(n):
    """
    Calculate the factorial of a given number.

    Args:
        n (int): A non-negative integer.

    Returns:
        int: Factorial of the input number.
    """
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result


print(calculate_factorial(5))

```

```
PS C:\Users\allas\OneDrive\Documents\web> & C:\Python313\python.exe c:/Users/allas/OneDrive/Documents/web/j.py
120
PS C:\Users\allas\OneDrive\Documents\web> 
```

Observation:

- Function renamed to calculate_factorial.
- Added docstring and meaningful variable names.
- PEP 8 formatting applied.

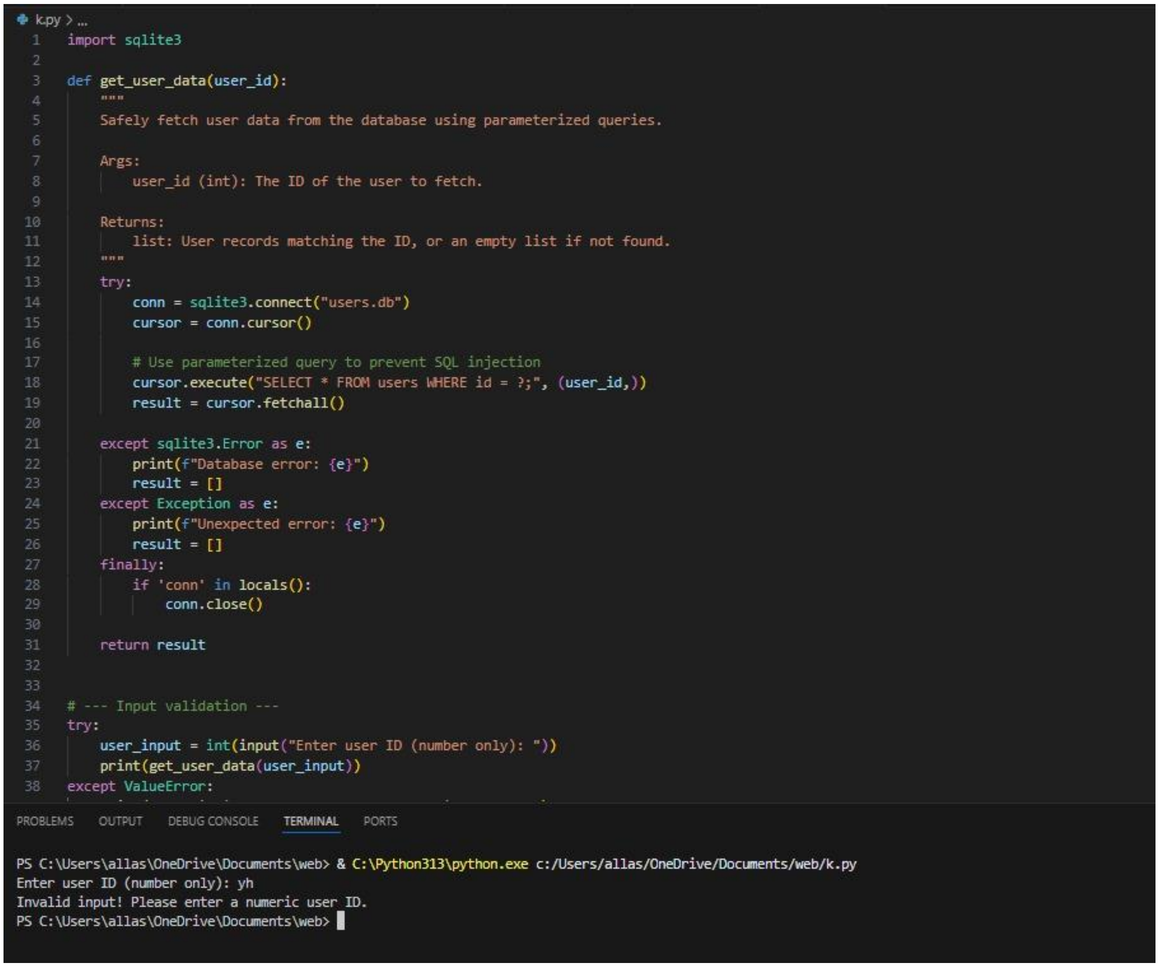
Output: 120

Task 4: Security and Error Handling Enhancement

Prompt:

Improve the following function by removing SQL injection risks, adding exception handling, and validating input before query execution.

Code:



```python
import sqlite3

def get_user_data(user_id):
    """
    Safely fetch user data from the database using parameterized queries.

    Args:
        user_id (int): The ID of the user to fetch.

    Returns:
        list: User records matching the ID, or an empty list if not found.
    """
    try:
        conn = sqlite3.connect("users.db")
        cursor = conn.cursor()

        # Use parameterized query to prevent SQL injection
        cursor.execute("SELECT * FROM users WHERE id = ?;", (user_id,))
        result = cursor.fetchall()

    except sqlite3.Error as e:
        print(f"Database error: {e}")
        result = []
    except Exception as e:
        print(f"Unexpected error: {e}")
        result = []
    finally:
        if 'conn' in locals():
            conn.close()

    return result


# --- Input validation ---
try:
    user_input = int(input("Enter user ID (number only): "))
    print(get_user_data(user_input))
except ValueError:
```

```
PS C:\Users\allas\OneDrive\Documents\web> & C:\Python313\python.exe c:/Users/allas/OneDrive/Documents/web/k.py
Enter user ID (number only): yh
Invalid input! Please enter a numeric user ID.
PS C:\Users\allas\OneDrive\Documents\web>
```

Observation:

- Used parameterized SQL query (? placeholders).
- Added try-except for sqlite3.Error.
- Validates input as integer.

Output: Invalid input handled / returns DB rows if valid.

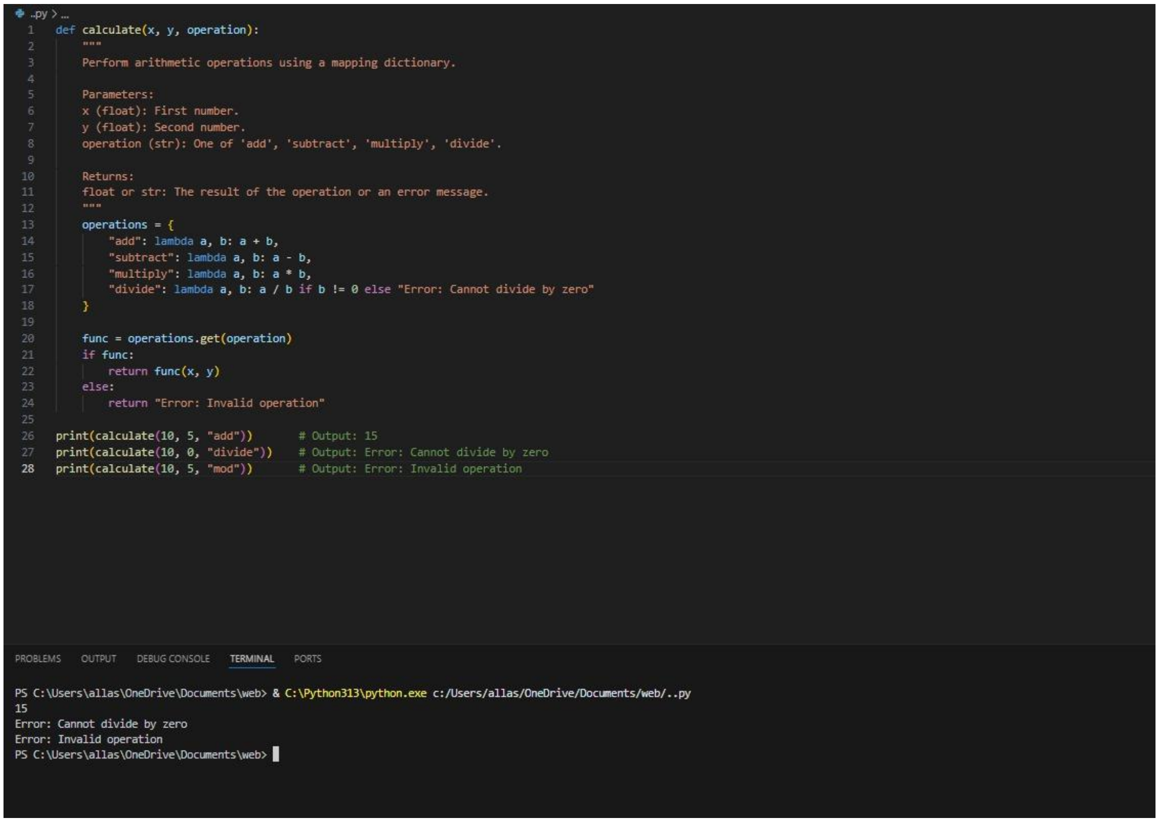
Task 5: Automated Code Review Report Generation

Prompt:

Generate a code review report for the following function. Identify issues with readability, maintainability, error handling, and security, and suggest improvements.

AI LAB-10.3

Code:



```python
def calculate(x, y, operation):
    """
    Perform arithmetic operations using a mapping dictionary.

    Parameters:
    x (float): First number.
    y (float): Second number.
    operation (str): One of 'add', 'subtract', 'multiply', 'divide'.

    Returns:
    float or str: The result of the operation or an error message.
    """
    operations = {
        "add": lambda a, b: a + b,
        "subtract": lambda a, b: a - b,
        "multiply": lambda a, b: a * b,
        "divide": lambda a, b: a / b if b != 0 else "Error: Cannot divide by zero"
    }

    func = operations.get(operation)
    if func:
        return func(x, y)
    else:
        return "Error: Invalid operation"

print(calculate(10, 5, "add"))       # Output: 15
print(calculate(10, 0, "divide"))    # Output: Error: Cannot divide by zero
print(calculate(10, 5, "mod"))       # Output: Error: Invalid operation
```

```
PS C:\Users\allas\OneDrive\Documents\web> & C:\Python313\python.exe c:/Users/allas/OneDrive/Documents/web/..py
15
Error: Cannot divide by zero
Error: Invalid operation
PS C:\Users\allas\OneDrive\Documents\web>
```

Observation:

- Missing docstrings.
- Function name and variable names not descriptive.
- Inconsistent formatting (inline vs multiline).
- No error handling for division by zero.
- Suggested improvements: descriptive names, error handling, docstrings, and PEP 8 compliance.