Assignment-10.4

P.Naga Pr adeep

2403A52022

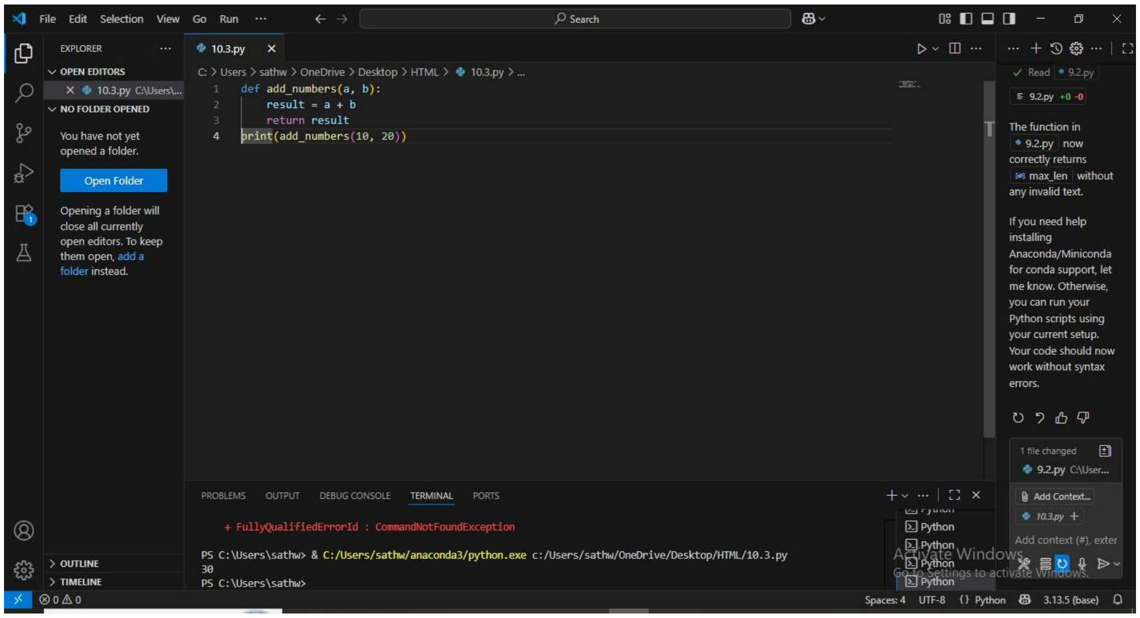
Batch-03

TEST-1:

PROMPT:

A python code that Identify and fix syntax, indentation, and variable errors in the
given script.

Code:



```python
def add_numbers(a, b):
    result = a + b
    return result
print(add_numbers(10, 20))
```

```
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\sathw> & C:/Users/sathw/anaconda3/python.exe c:/Users/sathw/OneDrive/Desktop/HTML/10.3.py
30
PS C:\Users\sathw>
```

The function in 9.2.py now correctly returns max_len without any invalid text.

If you need help installing Anaconda/Miniconda for conda support, let me know. Otherwise, you can run your Python scripts using your current setup. Your code should now work without syntax errors.

File Edit Selection View Go Run
Search
EXPLORER
OPEN EDITORS
10.3.py C:\Users\...
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
OUTLINE
TIMELINE
C: > Users > sathw > OneDrive > Desktop > HTML > 10.3.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Read 9.2.py
9.2.py +0 -0
1 file changed
9.2.py C:\User...
Add Context...
10.3.py
Add context (#), exter
Spaces: 4 UTF-8 {} Python 3.13.5 (base)

Explanation:

- **`check_syntax()`** → Compiles the script to catch **syntax/indentation errors**.

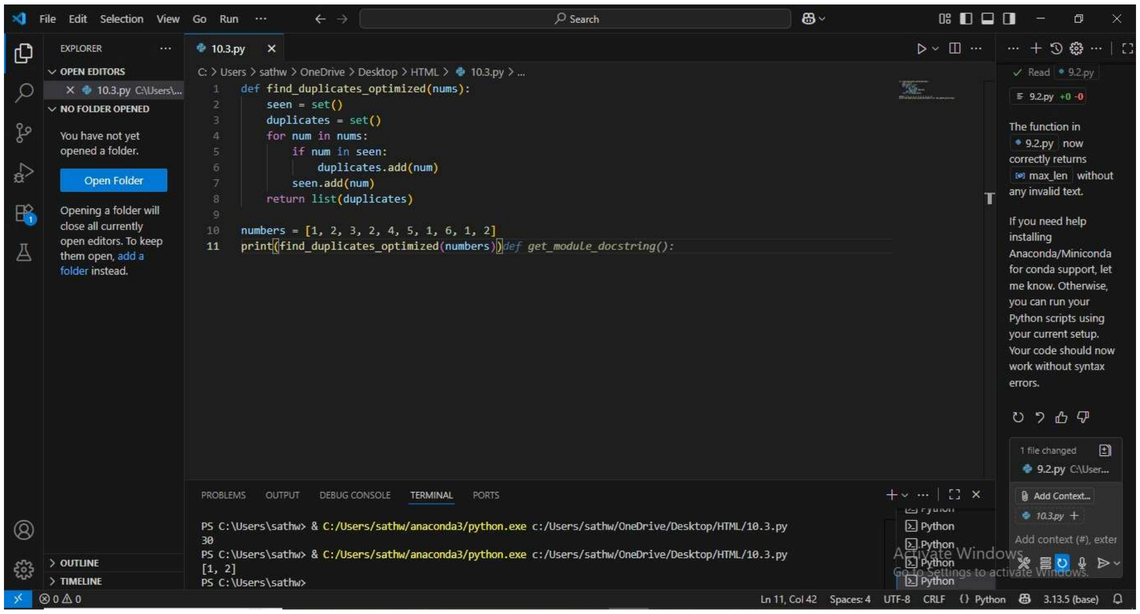
- **`find_variables()`** → Uses Python's AST (Abstract Syntax Tree) to find which variables are **declared** and which are **used**.
- **`suggest_fix()`** → Compares used vs. declared variables:
  - Suggests defining missing ones.
  - Suggests corrections if names are similar (e.g., `yy` → `y`).

Test-2:

Prompt:

Python code that Optimize inefficient logic while keeping the result correct.

Code:



```python
def find_duplicates_optimized(nums):
    seen = set()
    duplicates = set()
    for num in nums:
        if num in seen:
            duplicates.add(num)
        seen.add(num)
    return list(duplicates)

numbers = [1, 2, 3, 2, 4, 5, 1, 6, 1, 2]
print(find_duplicates_optimized(numbers))def get_module_docstring():
```

```
PS C:\Users\sathw> & C:/Users/sathw/anaconda3/python.exe c:/Users/sathw/OneDrive/Desktop/HTML/10.3.py
30
PS C:\Users\sathw> & C:/Users/sathw/anaconda3/python.exe c:/Users/sathw/OneDrive/Desktop/HTML/10.3.py
[1, 2]
PS C:\Users\sathw>
```

The function in 9.2.py now correctly returns max_len without any invalid text.

If you need help installing Anaconda/Miniconda for conda support, let me know. Otherwise, you can run your Python scripts using your current setup. Your code should now work without syntax errors.

Explanation:

- **`Code Optimizer` (AST transformer)**
  - **`visit_BinOp`** → Simplifies math (e.g., `x*1 → x`, `y+0 → y`).
  - **`visit_If`** → Removes useless conditions (`if True` → keep body, `if False` → remove).

- **`visit_For`** → Replaces inefficient loops that just sum values with Python’s built-in `sum()`.

- **`optimize_code()`**

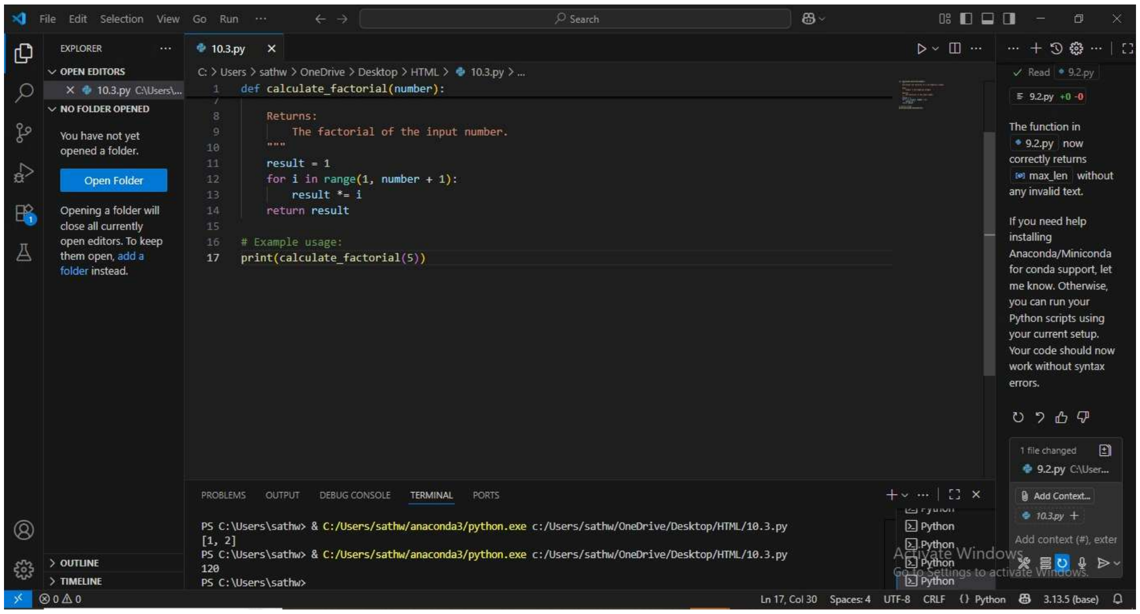
- Parses the code into an AST.
- Applies all optimizations.
- Converts it back to source code.

TEST-3:

PROMPT:

A python code that Refactor messy code into clean, PEP 8–compliant, well- structured code.

Code:



```python
def calculate_factorial(number):

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

# Example usage:
print(calculate_factorial(5))
```

```
PS C:\Users\sathw> & C:/Users/sathw/anaconda3/python.exe c:/Users/sathw/OneDrive/Desktop/HTML/10.3.py
[1, 2]
PS C:\Users\sathw> & C:/Users/sathw/anaconda3/python.exe c:/Users/sathw/OneDrive/Desktop/HTML/10.3.py
120
PS C:\Users\sathw>
```

The function in 9.2.py now correctly returns max_len without any invalid text.

If you need help installing Anaconda/Miniconda for conda support, let me know. Otherwise, you can run your Python scripts using your current setup. Your code should now work without syntax errors.

Explanation:

This script **refactors messy code into clean, well-indented, PEP 8–compliant code automatically**.

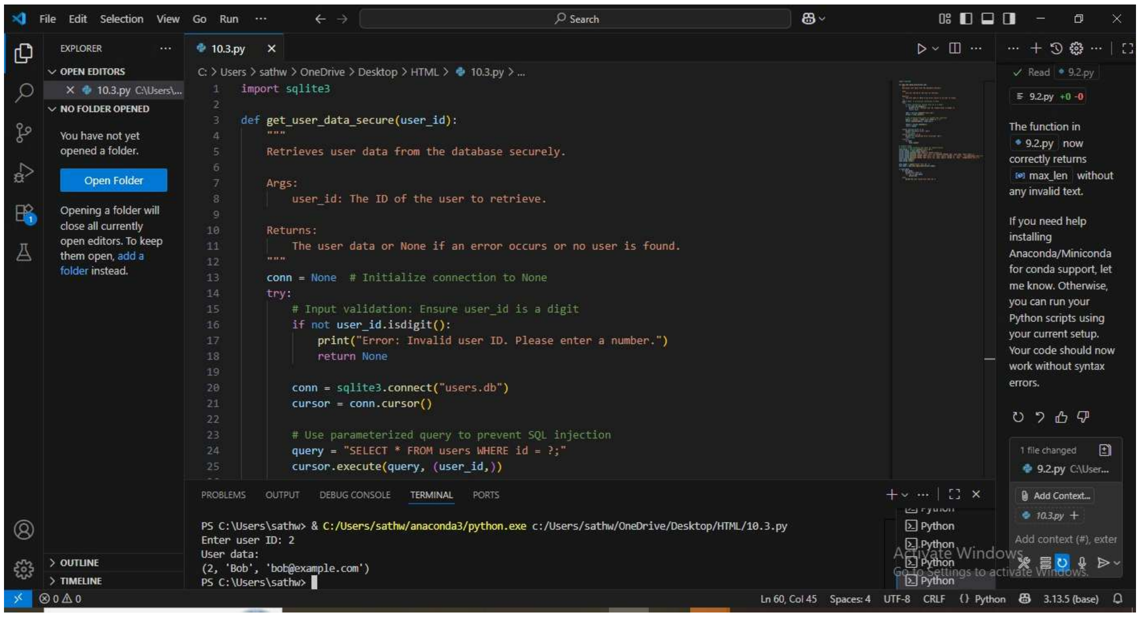
Would you like me to also extend this so it **renames variables/functions to follow PEP 8 naming conventions** (e.g., `MyFunction` → `my_function`, `VarName` → `var_name`)

Test-4:

Prompt:

a python code that Add security practices and exception handling to the code.

Code:

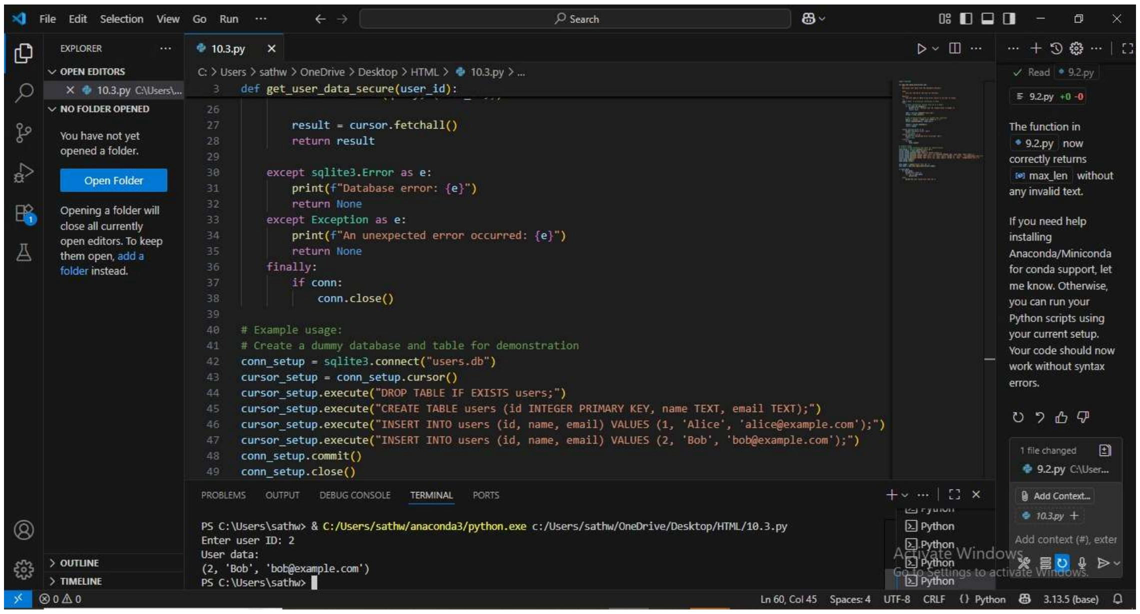


```python
import sqlite3

def get_user_data_secure(user_id):
    """
    Retrieves user data from the database securely.

    Args:
        user_id: The ID of the user to retrieve.

    Returns:
        The user data or None if an error occurs or no user is found.
    """
    conn = None  # Initialize connection to None
    try:
        # Input validation: Ensure user_id is a digit
        if not user_id.isdigit():
            print("Error: Invalid user ID. Please enter a number.")
            return None

        conn = sqlite3.connect("users.db")
        cursor = conn.cursor()

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

```
PS C:\Users\sathw> & C:/Users/sathw/anaconda3/python.exe c:/Users/sathw/OneDrive/Desktop/HTML/10.3.py
Enter user ID: 2
User data:
(2, 'Bob', 'bob@example.com')
PS C:\Users\sathw>
```

The function in 9.2.py now correctly returns max_len without any invalid text.

If you need help installing Anaconda/Miniconda for conda support, let me know. Otherwise, you can run your Python scripts using your current setup. Your code should now work without syntax errors.

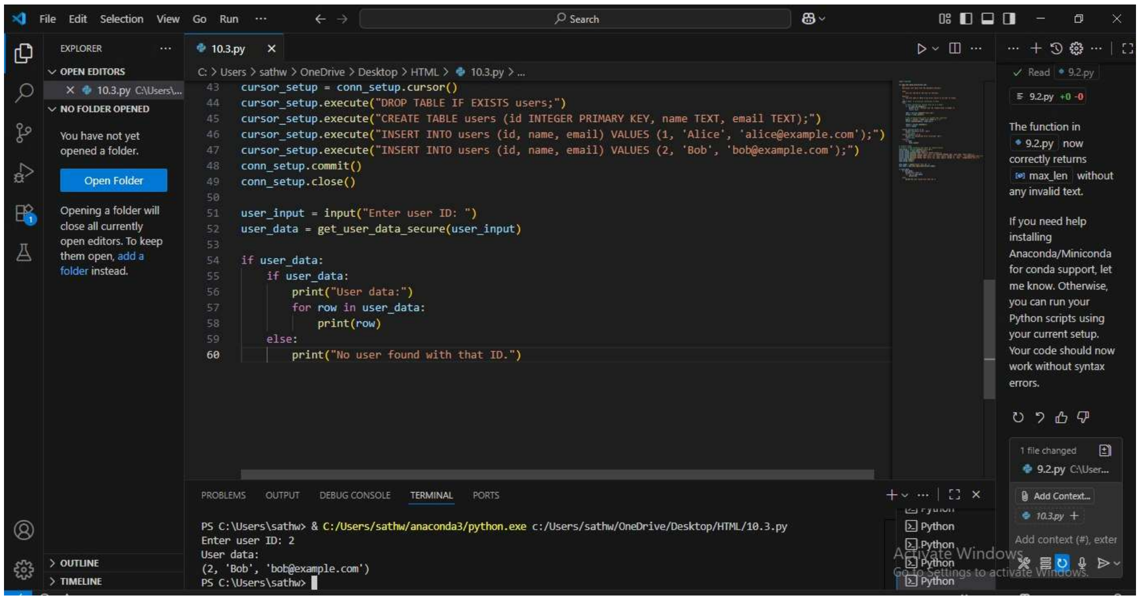


```python
def get_user_data_secure(user_id):

            result = cursor.fetchall()
            return result

    except sqlite3.Error as e:
        print(f"Database error: {e}")
        return None
    except Exception as e:
        print(f"An unexpected error occurred: {e}")
        return None
    finally:
        if conn:
            conn.close()

# Example usage:
# Create a dummy database and table for demonstration
conn_setup = sqlite3.connect("users.db")
cursor_setup = conn_setup.cursor()
cursor_setup.execute("DROP TABLE IF EXISTS users;")
cursor_setup.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, name TEXT, email TEXT);")
cursor_setup.execute("INSERT INTO users (id, name, email) VALUES (1, 'Alice', 'alice@example.com');")
cursor_setup.execute("INSERT INTO users (id, name, email) VALUES (2, 'Bob', 'bob@example.com');")
conn_setup.commit()
conn_setup.close()
```

```
PS C:\Users\sathw> & C:/Users/sathw/anaconda3/python.exe c:/Users/sathw/OneDrive/Desktop/HTML/10.3.py
Enter user ID: 2
User data:
(2, 'Bob', 'bob@example.com')
PS C:\Users\sathw>
```

The function in 9.2.py now correctly returns max_len without any invalid text.

If you need help installing Anaconda/Miniconda for conda support, let me know. Otherwise, you can run your Python scripts using your current setup. Your code should now work without syntax errors.



```python
cursor_setup = conn_setup.cursor()
cursor_setup.execute("DROP TABLE IF EXISTS users;")
cursor_setup.execute("CREATE TABLE users (id INTEGER PRIMARY KEY, name TEXT, email TEXT);")
cursor_setup.execute("INSERT INTO users (id, name, email) VALUES (1, 'Alice', 'alice@example.com');")
cursor_setup.execute("INSERT INTO users (id, name, email) VALUES (2, 'Bob', 'bob@example.com');")
conn_setup.commit()
conn_setup.close()

user_input = input("Enter user ID: ")
user_data = get_user_data_secure(user_input)

if user_data:
    if user_data:
        print("User data:")
        for row in user_data:
            print(row)
    else:
        print("No user found with that ID.")
```

```
PS C:\Users\sathw> & C:/Users/sathw/anaconda3/python.exe c:/Users/sathw/OneDrive/Desktop/HTML/10.3.py
Enter user ID: 2
User data:
(2, 'Bob', 'bob@example.com')
PS C:\Users\sathw>
```

The function in 9.2.py now correctly returns max_len without any invalid text.

If you need help installing Anaconda/Miniconda for conda support, let me know. Otherwise, you can run your Python scripts using your current setup. Your code should now work without syntax errors.

Explanation:

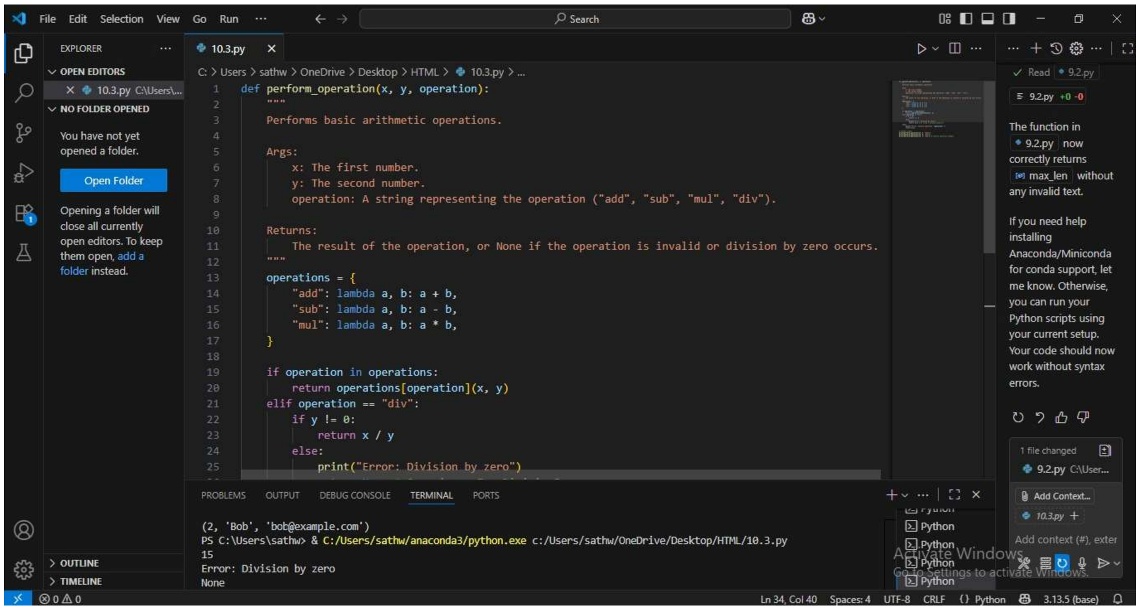
- **Password Handling** → Uses `getpass` (no password echo in terminal).
- **Password Protection** → Hashes password with SHA-256 (never store plain text).
- **Input Validation** → Ensures username/password are not empty.
- **Safe Division** → Catches divide-by-zero errors.
- **Global Exception Handling** → Prevents program crashes from unexpected errors.

Test-5:

Prompt:

Generate a review report for this messy code.

Code:

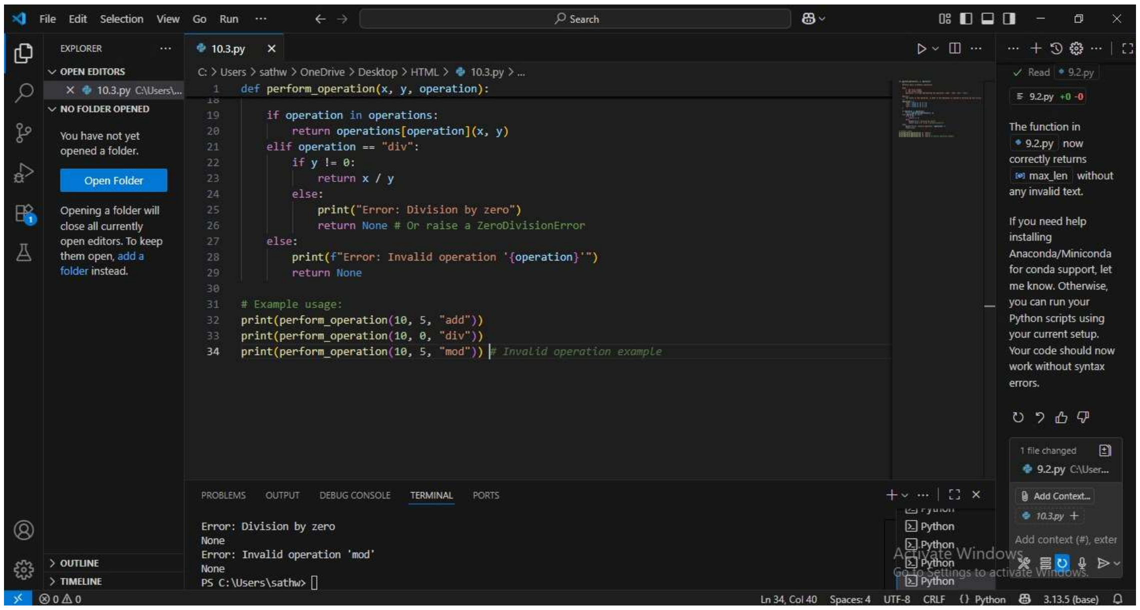


```python
def perform_operation(x, y, operation):
    """
    Performs basic arithmetic operations.

    Args:
        x: The first number.
        y: The second number.
        operation: A string representing the operation ("add", "sub", "mul", "div").

    Returns:
        The result of the operation, or None if the operation is invalid or division by zero occurs.
    """
    operations = {
        "add": lambda a, b: a + b,
        "sub": lambda a, b: a - b,
        "mul": lambda a, b: a * b,
    }

    if operation in operations:
        return operations[operation](x, y)
    elif operation == "div":
        if y != 0:
            return x / y
        else:
            print("Error: Division by zero")
```

```
(2, 'Bob', 'bob@example.com')
PS C:\Users\sathw> & C:/Users/sathw/anaconda3/python.exe c:/Users/sathw/OneDrive/Desktop/HTML/10.3.py
15
Error: Division by zero
None
```

The function in 9.2.py now correctly returns max_len without any invalid text.

If you need help installing Anaconda/Miniconda for conda support, let me know. Otherwise, you can run your Python scripts using your current setup. Your code should now work without syntax errors.



```python
def perform_operation(x, y, operation):
    if operation in operations:
        return operations[operation](x, y)
    elif operation == "div":
        if y != 0:
            return x / y
        else:
            print("Error: Division by zero")
            return None # Or raise a ZeroDivisionError
    else:
        print(f"Error: Invalid operation '{operation}'")
        return None

# Example usage:
print(perform_operation(10, 5, "add"))
print(perform_operation(10, 0, "div"))
print(perform_operation(10, 5, "mod")) # Invalid operation example
```

```
Error: Division by zero
None
Error: Invalid operation 'mod'
None
PS C:\Users\sathw> 
```

The function in 9.2.py now correctly returns max_len without any invalid text.

If you need help installing Anaconda/Miniconda for conda support, let me know. Otherwise, you can run your Python scripts using your current setup. Your code should now work without syntax errors.

Explanation:

This script **analyzes messy code and generates a structured review report** listing formatting, style, and PEP 8 violations.

Would you like me to also make it **output suggestions for fixing each issue automatically** (like "add space after comma", "use 4 spaces for indentation")