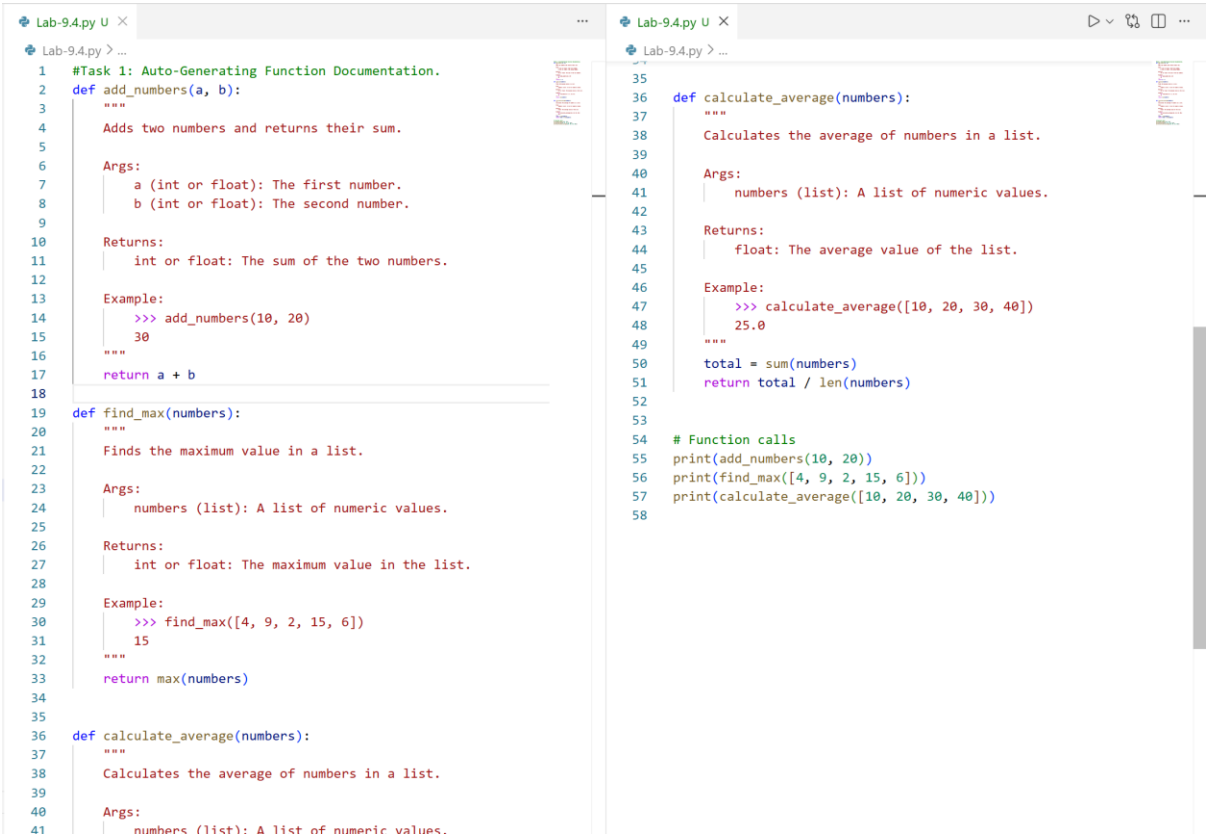
AI ASSISTED CODING LAB-9.4

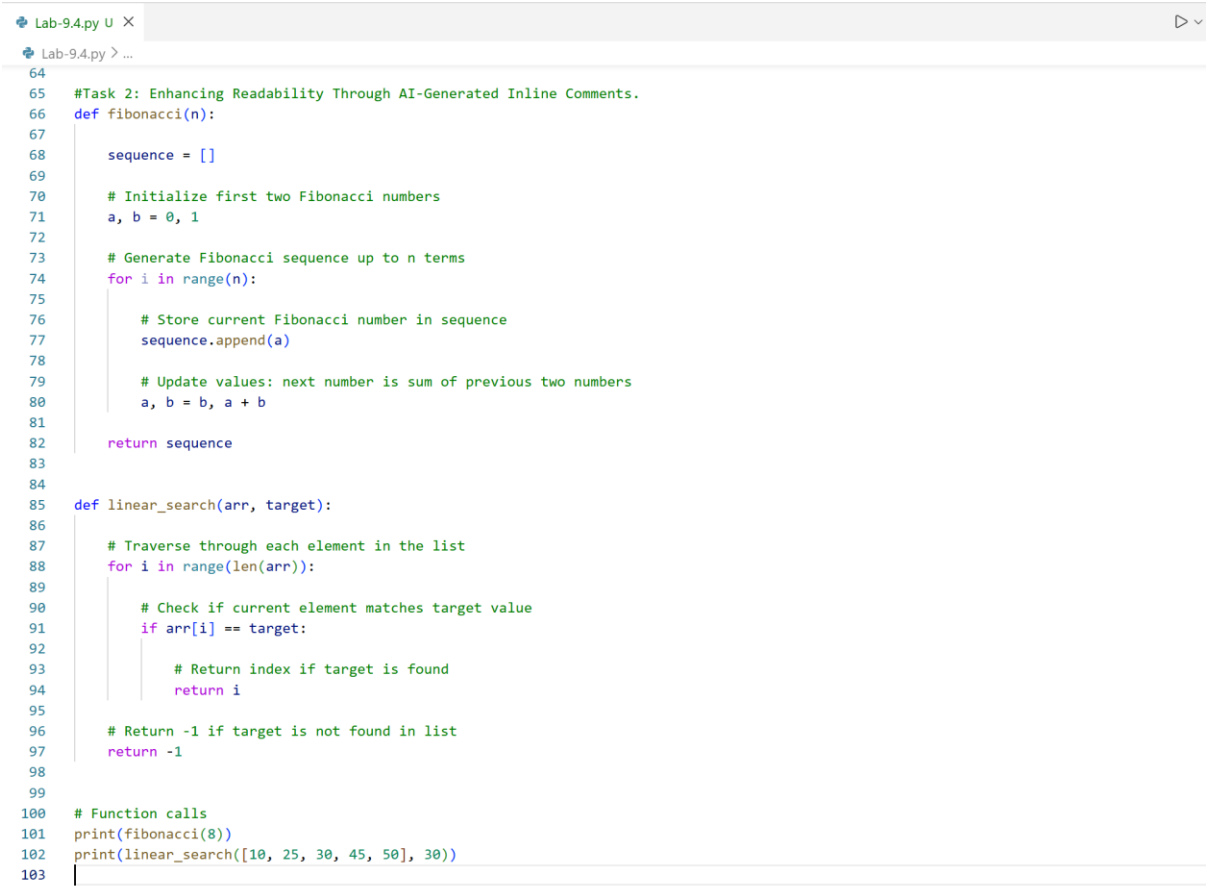
J.Sai Kumar || 09 || 2303A51562

Task 1: Auto-Generating Function Documentation.

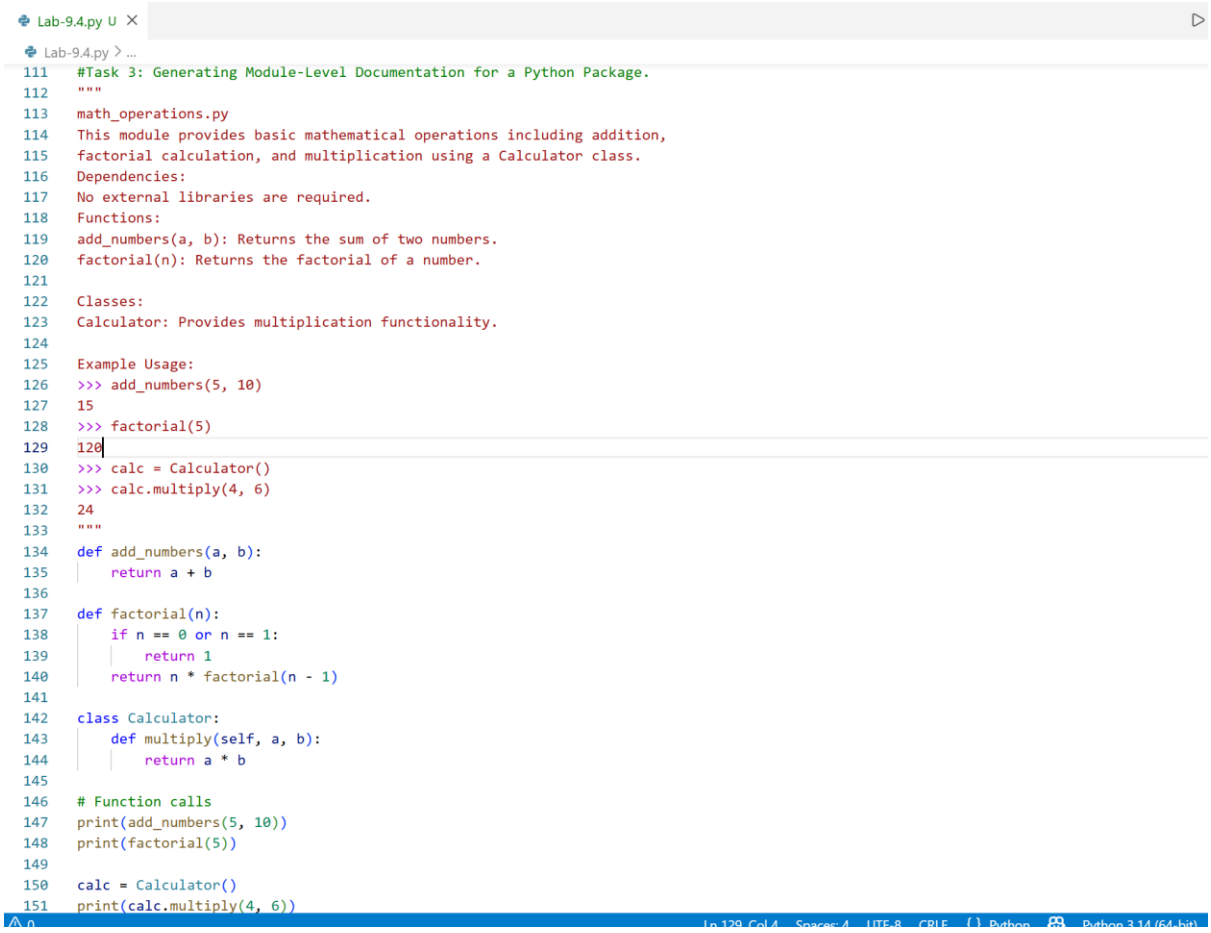


```python
#Task 1: Auto-Generating Function Documentation.
def add_numbers(a, b):
    """
    Adds two numbers and returns their sum.

    Args:
        a (int or float): The first number.
        b (int or float): The second number.

    Returns:
        int or float: The sum of the two numbers.

    Example:
        >>> add_numbers(10, 20)
        30
    """
    return a + b

def find_max(numbers):
    """
    Finds the maximum value in a list.

    Args:
        numbers (list): A list of numeric values.

    Returns:
        int or float: The maximum value in the list.

    Example:
        >>> find_max([4, 9, 2, 15, 6])
        15
    """
    return max(numbers)


def calculate_average(numbers):
    """
    Calculates the average of numbers in a list.

    Args:
        numbers (list): A list of numeric values.

    Returns:
        float: The average value of the list.

    Example:
        >>> calculate_average([10, 20, 30, 40])
        25.0
    """
    total = sum(numbers)
    return total / len(numbers)


# Function calls
print(add_numbers(10, 20))
print(find_max([4, 9, 2, 15, 6]))
print(calculate_average([10, 20, 30, 40]))
```

Task 2: Enhancing Readability Through AI-Generated Inline Comments.



Task 3: Generating Module-Level Documentation for a Python Package.



Lab-9.4.py U

Lab-9.4.py > ...

```
#Task 3: Generating Module-Level Documentation for a Python Package.
"""
math_operations.py
This module provides basic mathematical operations including addition,
factorial calculation, and multiplication using a Calculator class.
Dependencies:
No external libraries are required.
Functions:
add_numbers(a, b): Returns the sum of two numbers.
factorial(n): Returns the factorial of a number.

Classes:
Calculator: Provides multiplication functionality.

Example Usage:
>>> add_numbers(5, 10)
15
>>> factorial(5)
120
>>> calc = Calculator()
>>> calc.multiply(4, 6)
24
"""
def add_numbers(a, b):
    return a + b

def factorial(n):
    if n == 0 or n == 1:
        return 1
    return n * factorial(n - 1)

class Calculator:
    def multiply(self, a, b):
        return a * b

# Function calls
print(add_numbers(5, 10))
print(factorial(5))

calc = Calculator()
print(calc.multiply(4, 6))
```

Task 4: Converting Developer Comments into Structured Docstrings.

Lab-9.4.py U

Lab-9.4.py > ...

```
155
156
157
158 #Task 4: Converting Developer Comments into Structured Docstrings.
159 def calculate_average(numbers):
160     """
161     Calculates the average of numbers in a list.
162
163     Args:
164         numbers (list): A list of numeric values.
165
166     Returns:
167         float: The average value of the numbers.
168
169     Example:
170         >>> calculate_average([10, 20, 30, 40])
171         25.0
172     """
173
174     total = sum(numbers)
175
176     count = len(numbers)
177
178     average = total / count
179
180     return average
181
182
183 print(calculate_average([10, 20, 30, 40]))
184
```

Task 5: Building a Mini Automatic Documentation Generator.

Lab-9.4.py U | sample.py U | output.py U

Lab-9.4.py > ...

```python
#Task 5: Building a Mini Automatic Documentation Generator

import re

def generate_docstrings(file_name):

    with open(file_name, "r") as file:
        lines = file.readlines()

    new_lines = []

    for line in lines:

        new_lines.append(line)

        # Detect function definitions
        if re.match(r"\s*def\s+\w+", line):

            new_lines.append('    """\n')
            new_lines.append('    Description: TODO\n')
            new_lines.append('    Args: TODO\n')
            new_lines.append('    Returns: TODO\n')
            new_lines.append('    """\n')

        # Detect class definitions
        if re.match(r"\s*class\s+\w+", line):

            new_lines.append('    """\n')
            new_lines.append('    Class Description: TODO\n')
            new_lines.append('    """\n')

    with open("output.py", "w") as file:
        file.writelines(new_lines)
```

(variable) new_lines: list

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
ive/Desktop/AI ASSISTANT CODING/Lab-9.4.py"
Docstrings generated successfully.
PS C:\Users\saiku\OneDrive\Desktop\AI ASSISTANT CODING>
```

Lab-9.4.py U | sample.py U | output.py U

sample.py > ...

```python
def sample_function(a, b):
    return a + b

```

Lab-9.4.py U | sample.py U | output.py U

output.py > sample_function

```python
def sample_function(a, b):
    """
    Description: TODO
    Args: TODO
    Returns: TODO
    """
    return a + b

```