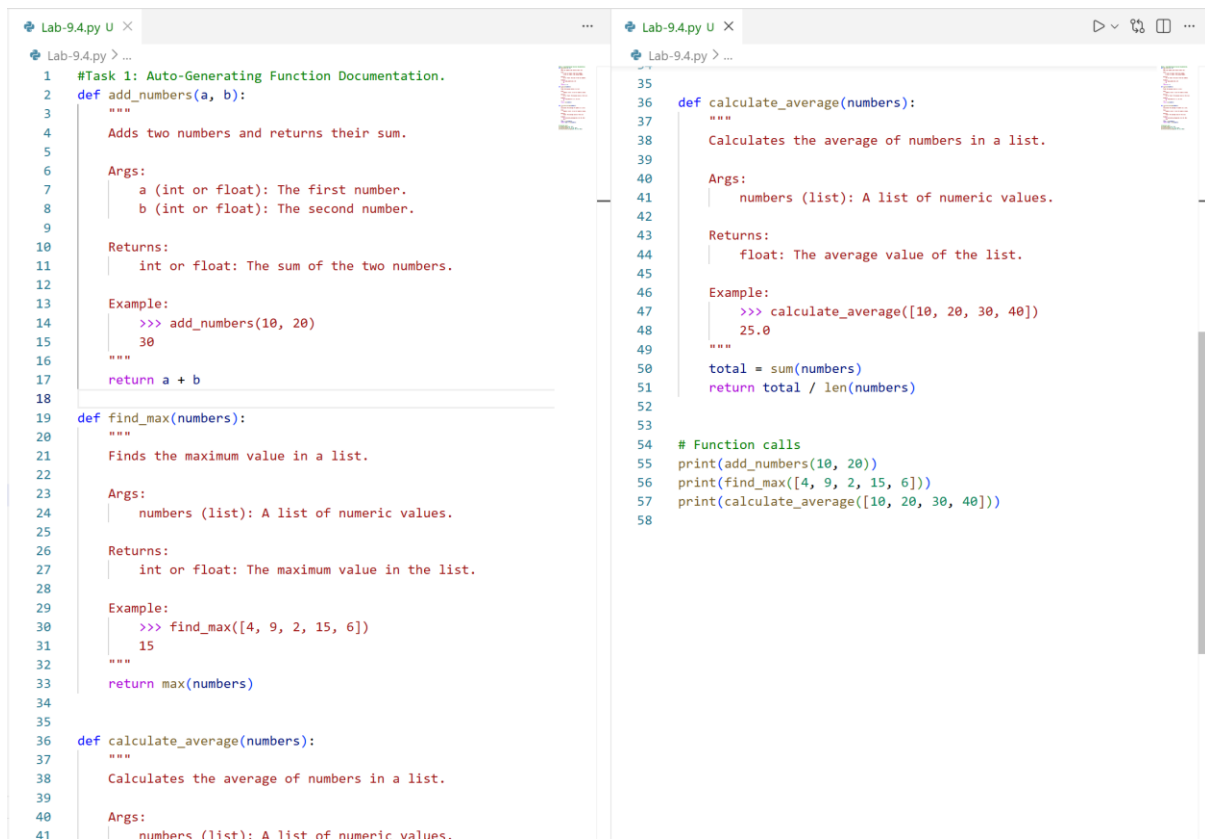


AI ASSISTED CODING LAB-9.4

J.Sai Kumar || 09 || 2303A51562

Task 1: Auto-Generating Function Documentation.



```
Lab-9.4.py U X ... Lab-9.4.py U X
Lab-9.4.py > ... Lab-9.4.py > ...

1 #Task 1: Auto-Generating Function Documentation.
2 def add_numbers(a, b):
3     """
4     Adds two numbers and returns their sum.
5
6     Args:
7     a (int or float): The first number.
8     b (int or float): The second number.
9
10    Returns:
11    int or float: The sum of the two numbers.
12
13    Example:
14    >>> add_numbers(10, 20)
15    30
16    """
17    return a + b
18
19 def find_max(numbers):
20     """
21     Finds the maximum value in a list.
22
23     Args:
24     numbers (list): A list of numeric values.
25
26     Returns:
27     int or float: The maximum value in the list.
28
29     Example:
30     >>> find_max([4, 9, 2, 15, 6])
31     15
32     """
33    return max(numbers)
34
35
36 def calculate_average(numbers):
37     """
38     Calculates the average of numbers in a list.
39
40     Args:
41     numbers (list): A list of numeric values.
42
43     Returns:
44     float: The average value of the list.
45
46     Example:
47     >>> calculate_average([10, 20, 30, 40])
48     25.0
49     """
50    total = sum(numbers)
51    return total / len(numbers)
52
53
54 # Function calls
55 print(add_numbers(10, 20))
56 print(find_max([4, 9, 2, 15, 6]))
57 print(calculate_average([10, 20, 30, 40]))
58
```

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

```
Lab-9.4.py U X
Lab-9.4.py > ...
64
65 #Task 2: Enhancing Readability Through AI-Generated Inline Comments.
66 def fibonacci(n):
67
68     sequence = []
69
70     # Initialize first two Fibonacci numbers
71     a, b = 0, 1
72
73     # Generate Fibonacci sequence up to n terms
74     for i in range(n):
75
76         # Store current Fibonacci number in sequence
77         sequence.append(a)
78
79         # Update values: next number is sum of previous two numbers
80         a, b = b, a + b
81
82     return sequence
83
84
85 def linear_search(arr, target):
86
87     # Traverse through each element in the list
88     for i in range(len(arr)):
89
90         # Check if current element matches target value
91         if arr[i] == target:
92
93             # Return index if target is found
94             return i
95
96     # Return -1 if target is not found in list
97     return -1
98
99
100 # Function calls
101 print(fibonacci(8))
102 print(linear_search([10, 25, 30, 45, 50], 30))
103
```

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

```
Lab-9.4.py U X
Lab-9.4.py > ...
111 #Task 3: Generating Module-Level Documentation for a Python Package.
112 """
113 math_operations.py
114 This module provides basic mathematical operations including addition,
115 factorial calculation, and multiplication using a Calculator class.
116 Dependencies:
117 No external libraries are required.
118 Functions:
119 add_numbers(a, b): Returns the sum of two numbers.
120 factorial(n): Returns the factorial of a number.
121
122 Classes:
123 Calculator: Provides multiplication functionality.
124
125 Example Usage:
126 >>> add_numbers(5, 10)
127 15
128 >>> factorial(5)
129 120
130 >>> calc = Calculator()
131 >>> calc.multiply(4, 6)
132 24
133 """
134 def add_numbers(a, b):
135     return a + b
136
137 def factorial(n):
138     if n == 0 or n == 1:
139         return 1
140     return n * factorial(n - 1)
141
142 class Calculator:
143     def multiply(self, a, b):
144         return a * b
145
146 # Function calls
147 print(add_numbers(5, 10))
148 print(factorial(5))
149
150 calc = Calculator()
151 print(calc.multiply(4, 6))
```

Task 4: Converting Developer Comments into Structured Docstrings.

```
Lab-9.4.py U X
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 X  sample.py U  output.py U
Lab-9.4.py > ...

191 #Task 5: Building a Mini Automatic Documentation Generator
192
193 import re
194
195 def generate_docstrings(file_name):
196
197     with open(file_name, "r") as file:
198         lines = file.readlines()
199
200     new_lines = []
201
202     for line in lines:
203
204         new_lines.append(line)
205
206         # Detect function definitions
207         if re.match(r"\s*def\s+\w+", line):
208
209             new_lines.append('        """\n')
210             new_lines.append('        Description: TODO\n')
211             new_lines.append('        Args: TODO\n')
212             (variable) new_lines: list s: TODO\n')
213             new_lines.append('        """\n')
214
215         # Detect class definitions
216         if re.match(r"\s*class\s+\w+", line):
217
218             new_lines.append('        """\n')
219             new_lines.append('        Class Description: TODO\n')
220             new_lines.append('        """\n')
221
222     with open("output.py", "w") as file:
223         file.writelines(new_lines)
224
```

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 X

output.py U

sample.py > ...

```
1 def sample_function(a, b):  
2     return a + b  
3
```

Lab-9.4.py U

sample.py U

output.py U X

output.py > sample_function

```
1 def sample_function(a, b):  
2     """  
3     Description: TODO  
4     Args: TODO  
5     Returns: TODO  
6     """  
7     return a + b  
8
```