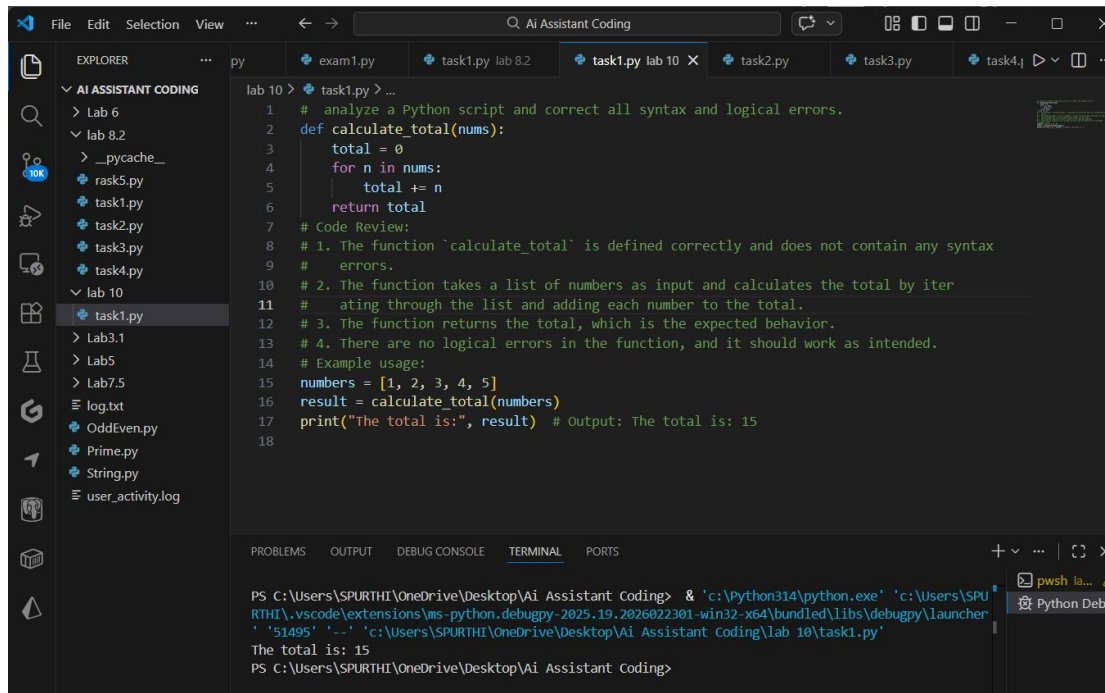


AIAC LAB 10.2
Billa Spurthi
2303A51802
Batch 29

Task:

Use AI to analyze a Python script and correct all syntax and logical errors.



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a project structure with folders 'Lab 6', 'lab 8.2', and 'lab 10'. The 'lab 10' folder is expanded, showing files like 'task1.py'. The main editor window displays the content of 'task1.py' for 'lab 10'. The code includes a function definition, a code review comment, and an example usage. The terminal at the bottom shows the command to run the script and its output.

```
lab 10 > task1.py > ...
1 # analyze a Python script and correct all syntax and logical errors.
2 def calculate_total(nums):
3     total = 0
4     for n in nums:
5         total += n
6     return total
7 # Code Review:
8 # 1. The function 'calculate_total' is defined correctly and does not contain any syntax
9 # errors.
10 # 2. The function takes a list of numbers as input and calculates the total by iter
11 # ating through the list and adding each number to the total.
12 # 3. The function returns the total, which is the expected behavior.
13 # 4. There are no logical errors in the function, and it should work as intended.
14 # Example usage:
15 numbers = [1, 2, 3, 4, 5]
16 result = calculate_total(numbers)
17 print("The total is:", result) # Output: The total is: 15
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> & 'c:\Python314\python.exe' 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\lab 10\task1.py'
The total is: 15
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>
```

Task Description -2(Code Style Standardization)

Task:

Use AI to refactor Python code to comply with standard coding style guidelines.

The screenshot shows the Visual Studio Code interface with the AI Assistant coding tool active. The AI Assistant is providing a code review and refactoring suggestions for a Python file named `task2.py`. The code being reviewed is a simple function `find_sum(a, b)` that returns the sum of two numbers. The AI Assistant's suggestions include: 1. Refactoring the code to comply with standard Python coding style guidelines (PEP-8). 2. Improving formatting, naming conventions, spacing, and structure without changing the functionality of the program. 3. The function is defined correctly and follows PEP-8 naming conventions. 4. The function takes two parameters, 'a' and 'b', and returns their sum, which is the expected behavior. 5. The function is called with the arguments 5 and 10, and the result is printed, which is correct. 6. The code is properly formatted with appropriate spacing and indentation. 7. Example usage: `result = find_sum(5, 10)` and `print("The sum is:", result)` # Output: The sum is: 15. The terminal output shows the command `python task2.py` being executed, and the output is `The sum is: 15`.

```
lab 10 > task2.py > ...
1 # Refactor the given Python code to comply with standard Python coding
2 # style guidelines (PEP-8).
3 # Improve formatting, naming conventions, spacing, and structure without
4 # changing the functionality of the program.
5 def find_sum(a, b):
6     return a + b
7
8
9 print(find_sum(5, 10))
10
11 # Code Review:
12 # 1. The function 'find_sum' is defined correctly and follows PEP-8 naming conventions.
13 # 2. The function takes two parameters, 'a' and 'b', and returns
14 # their sum, which is the expected behavior.
15 # 3. The function is called with the arguments 5 and 10, and the
16 # result is printed, which is correct.
17 # 4. The code is properly formatted with appropriate spacing and indentation.
18 # Example usage:
19 result = find_sum(5, 10)
20 print("The sum is:", result) # Output: The sum is: 15
21
```

Debug Console (Ctrl+Shift+Y)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> & 'c:\Python314\python.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026022301-win32-x64\bundle\libs\debugpy\launcher' '55561' '--' 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\lab 10\task2.py'

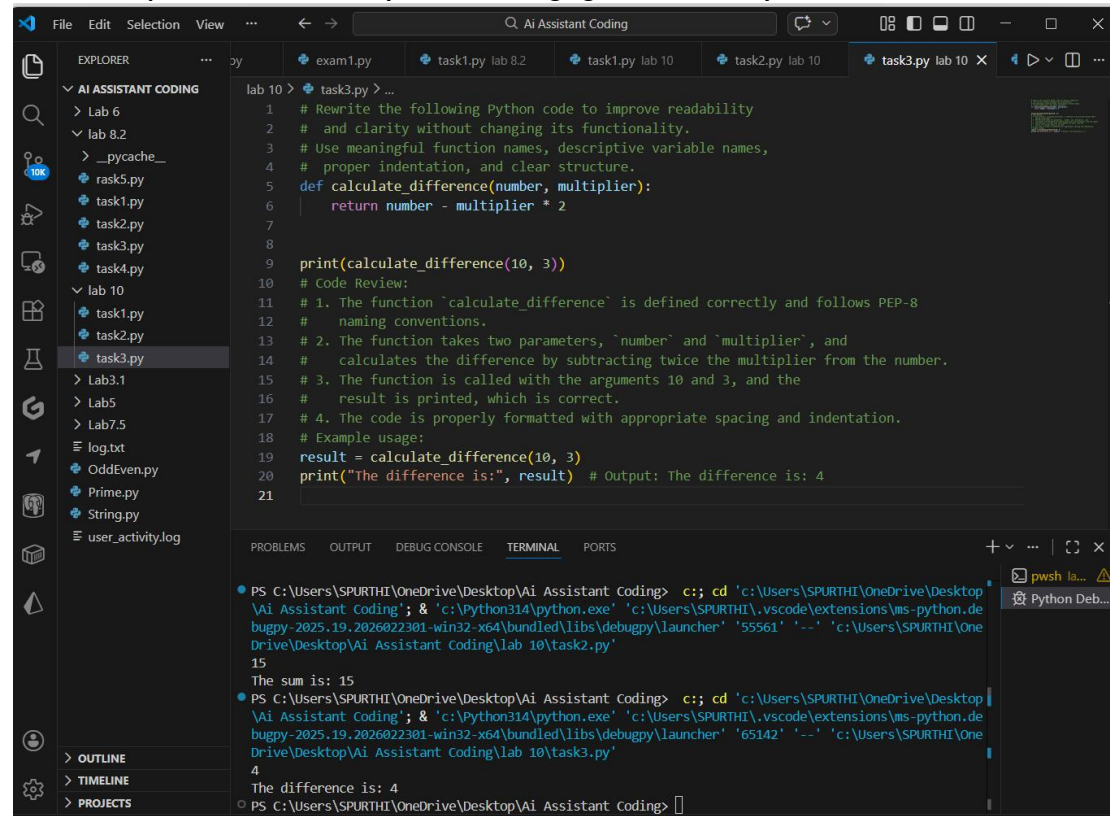
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> c;; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding'; & 'c:\Python314\python.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026022301-win32-x64\bundle\libs\debugpy\launcher' '55561' '--' 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\lab 10\task2.py'

15
The sum is: 15
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>

Task Description -3(Code Clarity Improvement)

Task:

Use AI to improve code readability without changing its functionality.



```
lab 10 > task3.py > ...
1 # Rewrite the following Python code to improve readability
2 # and clarity without changing its functionality.
3 # Use meaningful function names, descriptive variable names,
4 # proper indentation, and clear structure.
5 def calculate_difference(number, multiplier):
6     return number - multiplier * 2
7
8
9 print(calculate_difference(10, 3))
10 # Code Review:
11 # 1. The function 'calculate_difference' is defined correctly and follows PEP-8
12 # naming conventions.
13 # 2. The function takes two parameters, 'number' and 'multiplier', and
14 # calculates the difference by subtracting twice the multiplier from the number.
15 # 3. The function is called with the arguments 10 and 3, and the
16 # result is printed, which is correct.
17 # 4. The code is properly formatted with appropriate spacing and indentation.
18 # Example usage:
19 result = calculate_difference(10, 3)
20 print("The difference is:", result) # Output: The difference is: 4
21
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> c;; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding'; & 'c:\Python314\python.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026022301-win32-x64\bundled\libs\debugpy\launcher' '55561' '--' 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\lab 10\task2.py'

15
The sum is: 15

PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> c;; cd 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding'; & 'c:\Python314\python.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.debugpy-2025.19.2026022301-win32-x64\bundled\libs\debugpy\launcher' '65142' '--' 'c:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding\lab 10\task3.py'

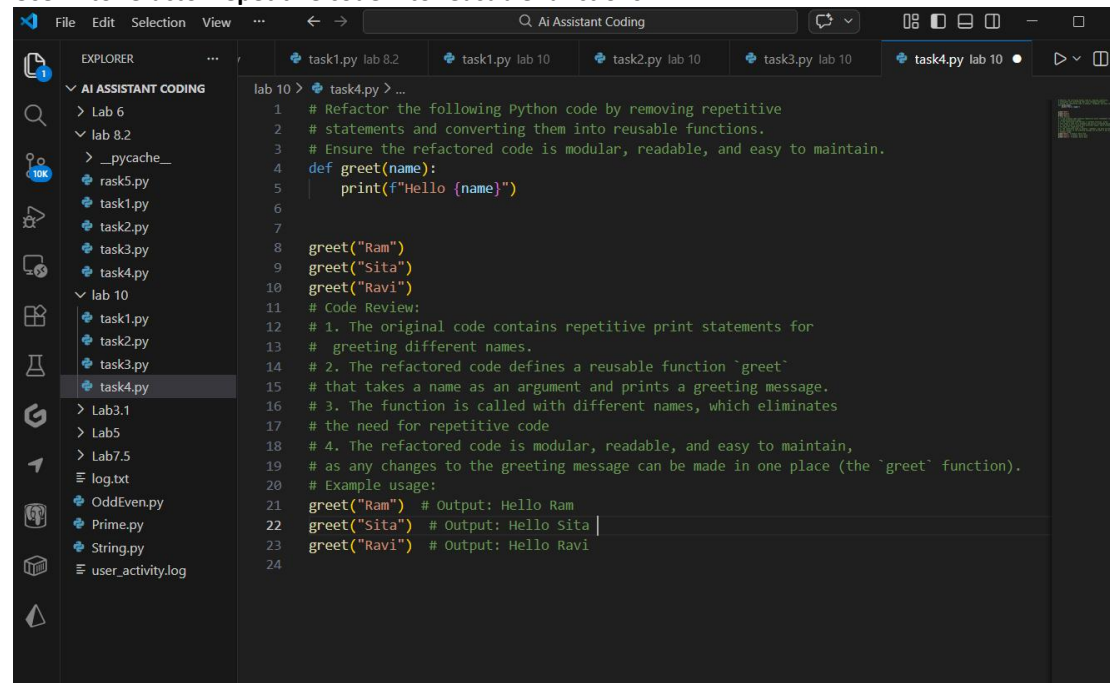
4
The difference is: 4

PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding>

Task Description -4(Structural Refactoring)

Task:

Use AI to refactor repetitive code into reusable functions.



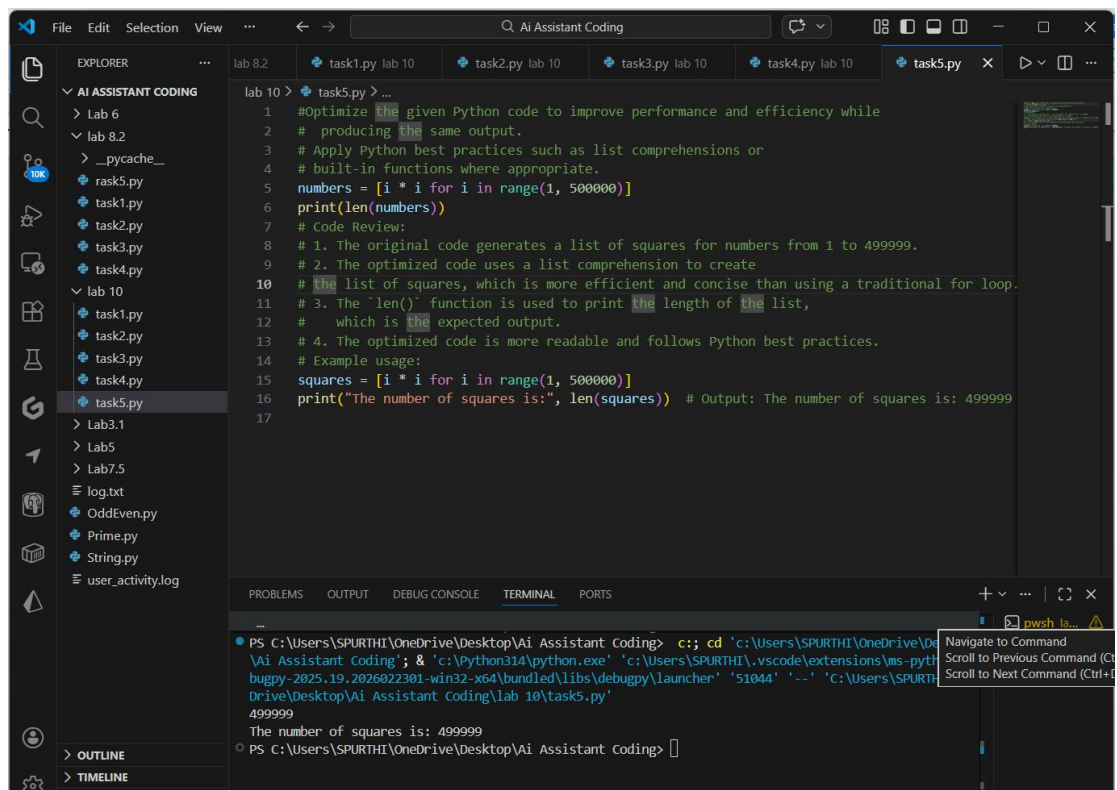
```
lab 10 > task4.py > ...
1 # Refactor the following Python code by removing repetitive
2 # statements and converting them into reusable functions.
3 # Ensure the refactored code is modular, readable, and easy to maintain.
4 def greet(name):
5     print(f"Hello {name}")
6
7
8 greet("Ram")
9 greet("Sita")
10 greet("Ravi")
11 # Code Review:
12 # 1. The original code contains repetitive print statements for
13 # greeting different names.
14 # 2. The refactored code defines a reusable function 'greet'
15 # that takes a name as an argument and prints a greeting message.
16 # 3. The function is called with different names, which eliminates
17 # the need for repetitive code
18 # 4. The refactored code is modular, readable, and easy to maintain,
19 # as any changes to the greeting message can be made in one place (the 'greet' function).
20 # Example usage:
21 greet("Ram") # Output: Hello Ram
22 greet("Sita") # Output: Hello Sita
23 greet("Ravi") # Output: Hello Ravi
24
```

```
Hello Sita
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> c;; cd 'c:\Users\SPURTHI\OneDrive\Desktop\
o \Ai Assistant Coding'; & 'c:\Python314\python.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-python.de
bugpy-2025.19.2026022301-win32-x64\bundled\libs\debugpy\launcher' '54774' '--' 'c:\Users\SPURTHI\One
Drive\Desktop\Ai Assistant Coding\lab 10\task4.py'
Hello Ram
Hello Sita
Hello Ravi
Hello Ram
Hello Sita
Hello Sita
Hello Ravi
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> []
```

Task Description -5(Efficiency Enhancement)

Task:

Use AI to optimize Python code for better performance.



```
lab 10 > task5.py > ...
1 #Optimize the given Python code to improve performance and efficiency while
2 # producing the same output.
3 # Apply Python best practices such as list comprehensions or
4 # built-in functions where appropriate.
5 numbers = [i * i for i in range(1, 500000)]
6 print(len(numbers))
7 # Code Review:
8 # 1. The original code generates a list of squares for numbers from 1 to 499999.
9 # 2. The optimized code uses a list comprehension to create
10 # the list of squares, which is more efficient and concise than using a traditional for loop.
11 # 3. The 'len()' function is used to print the length of the list,
12 # which is the expected output.
13 # 4. The optimized code is more readable and follows Python best practices.
14 # Example usage:
15 squares = [i * i for i in range(1, 500000)]
16 print("The number of squares is:", len(squares)) # Output: The number of squares is: 499999
17
```

```
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> c;; cd 'c:\Users\SPURTHI\OneDrive\De
\Ai Assistant Coding'; & 'c:\Python314\python.exe' 'c:\Users\SPURTHI\.vscode\extensions\ms-pyth
bugpy-2025.19.2026022301-win32-x64\bundled\libs\debugpy\launcher' '51044' '--' 'c:\Users\SPURTHI
Drive\Desktop\Ai Assistant Coding\lab 10\task5.py'
499999
The number of squares is: 499999
PS C:\Users\SPURTHI\OneDrive\Desktop\Ai Assistant Coding> []
```