

# AIAC-LAB ASSIGNMENT

Lab 10.2 – Code Review and Quality: Using AI to Improve Code Quality and Readability

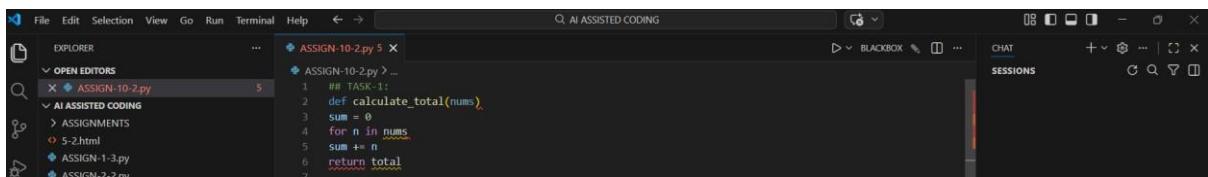
Name: P.Vyshnavi

2403a51I34

B-52

## TASK-1: Error Detection and Correction

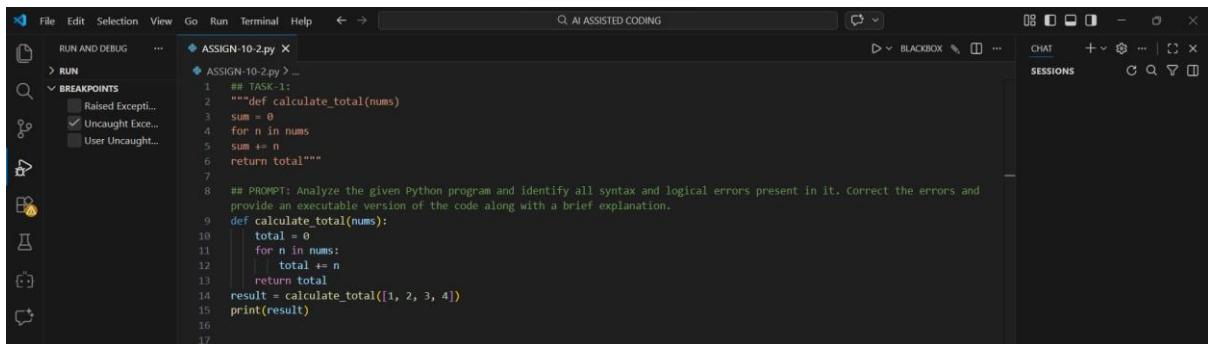
### Sample Input Code:



```
File Edit Selection View Go Run Terminal Help < > Q AI ASSISTED CODING
EXPLORER OPEN EDITORS ASSIGN-10-2.py 5
AI ASSISTED CODING ASSIGNMENTS 5-2.html
ASSIGN-1-3.py ASSIGN-2-2.py
```

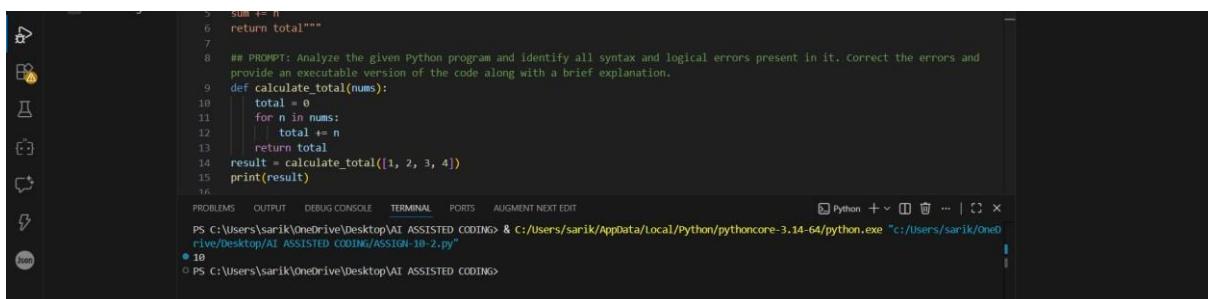
```
ASSIGN-10-2.py > ...
1 ## TASK-1:
2 def calculate_total(nums):
3     sum = 0
4     for n in nums
5         sum += n
6     return total
```

**Prompt:** Analyse the given Python program and identify all syntax and logical errors present in it. Correct the errors and provide an executable version of the code along with a brief explanation. **Corrected Input Code:**



```
File Edit Selection View Go Run Terminal Help < > Q AI ASSISTED CODING
RUN AND DEBUG RUN
BREAKPOINTS Raised Except... Uncaught Except... User Uncaught...
ASSIGN-10-2.py X
ASSIGN-10-2.py > ...
1 ## TASK-1:
2 """def calculate_total(nums):
3     sum = 0
4     for n in nums
5         sum += n
6     return total"""
7
8 ## PROMPT: Analyze the given Python program and identify all syntax and logical errors present in it. Correct the errors and provide an executable version of the code along with a brief explanation.
9 def calculate_total(nums):
10     total = 0
11     for n in nums:
12         total += n
13     return total
14 result = calculate_total([1, 2, 3, 4])
15 print(result)
16
```

### Output:



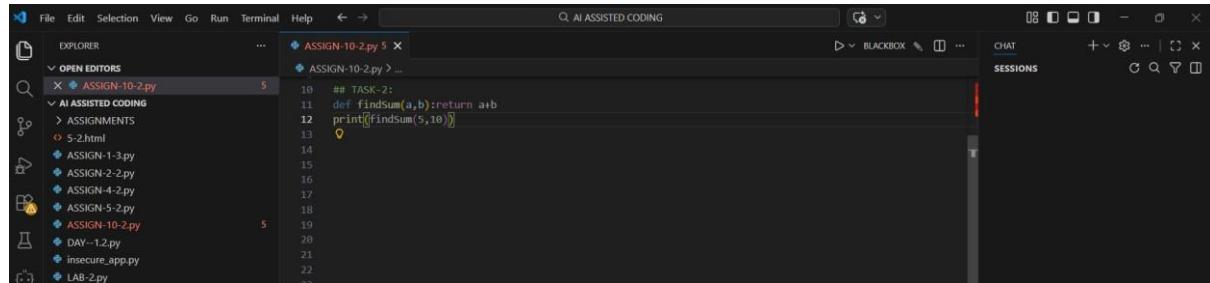
```
File Edit Selection View Go Run Terminal Help < > Q AI ASSISTED CODING
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ALIGNMENT EDIT
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-10-2.py"
● 10
○ PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

**Explanation:** The original code contained syntax errors such as missing colons and improper indentation, which prevented execution. A logical error was also

present where an undefined variable was returned. These issues were corrected to make the function executable and reliable.

## TASK-2: Code Style Standardization Sample

### Input Code:

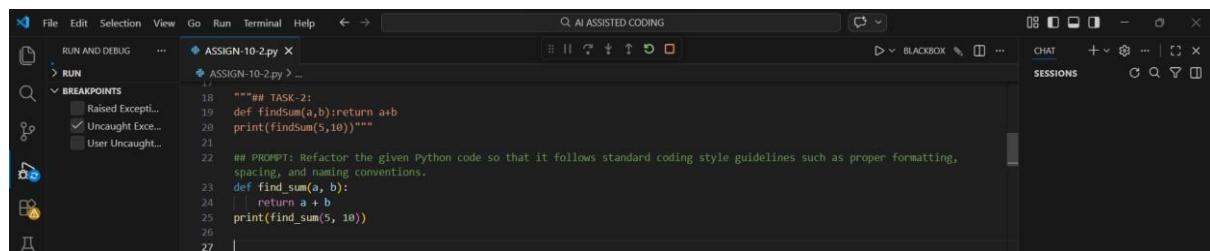


The screenshot shows the VS Code interface with the Explorer sidebar open, displaying several Python files. The current file is 'ASSIGN-10-2.py'. The code contains a function definition:

```
10  ## TASK-2:
11  def findSum(a,b):return a+b
12  print([findSum(5,10)])
13
14
15
16
17
18
19
20
21
22
```

**Prompt:** Refactor the given Python code so that it follows standard coding style guidelines such as proper formatting, spacing, and naming conventions.

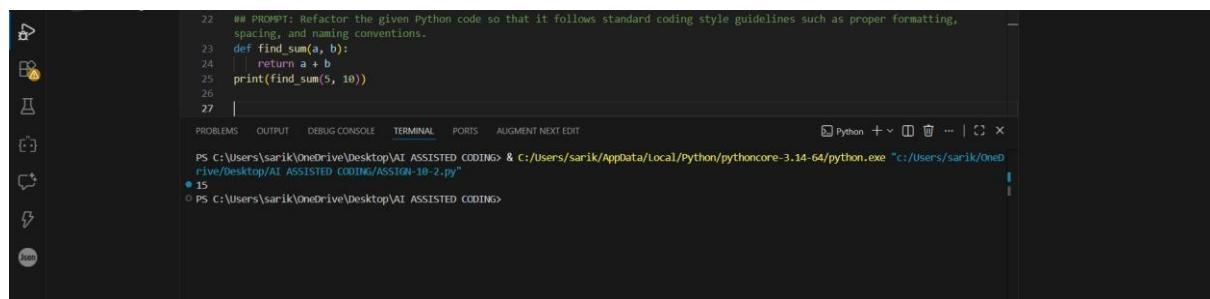
### Corrected Input Code:



The screenshot shows the VS Code interface with the Explorer sidebar open, displaying several Python files. The current file is 'ASSIGN-10-2.py'. The code has been refactored:

```
18  """## TASK-2:
19  def findSum(a,b):return a+b
20  print(findSum(5,10))"""
21
22  ## PROMPT: Refactor the given Python code so that it follows standard coding style guidelines such as proper formatting,
23  # spacing, and naming conventions.
24  def find_sum(a, b):
25      return a + b
26  print(find_sum(5, 10))
27
```

### Output:



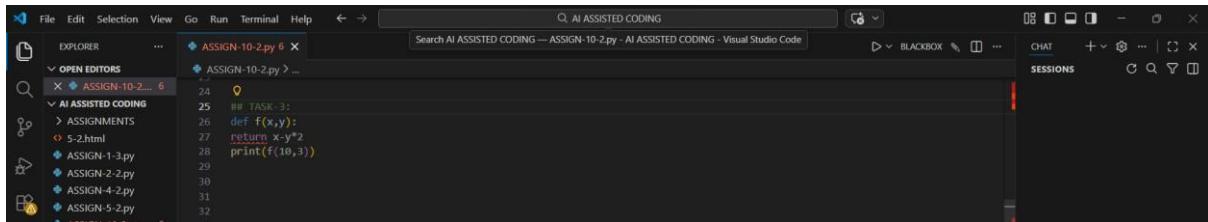
The screenshot shows the VS Code interface with the Terminal tab active. The terminal window displays the output of the refactored code:

```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneD
rive/Desktop/AI ASSISTED CODING/ASSIGN-10-2.py"
● 15
○ PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

**Explanation:** The code was reformatted to follow Python's PEP 8 style guidelines by improving spacing, naming conventions, and structure. These changes enhance readability and make the code easier to maintain without altering its functionality.

## TASK-3: Code Clarity Improvement Sample

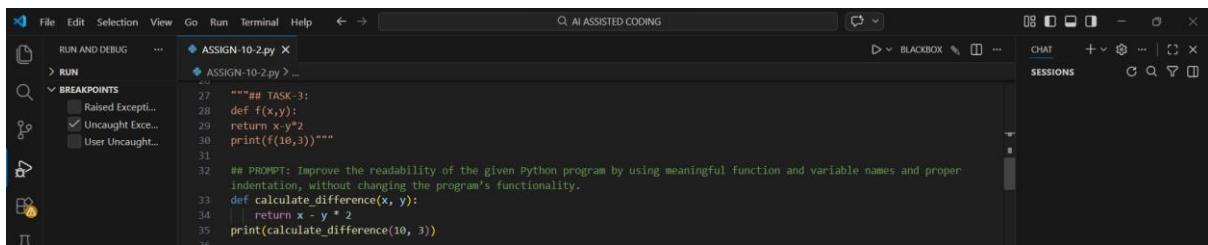
### Input Code:



A screenshot of the Visual Studio Code interface. The title bar says "AI ASSISTED CODING". The Explorer sidebar shows several files under "OPEN EDITORS" and "AI ASSISTED CODING" categories. The main editor window contains the following Python code:

```
ASSIGN-10-2.py
24     Q
25     ## TASK-3:
26     def f(x,y):
27         return x-y**2
28     print(f(10,3))
29
30
31
32
33
34
35
36
```

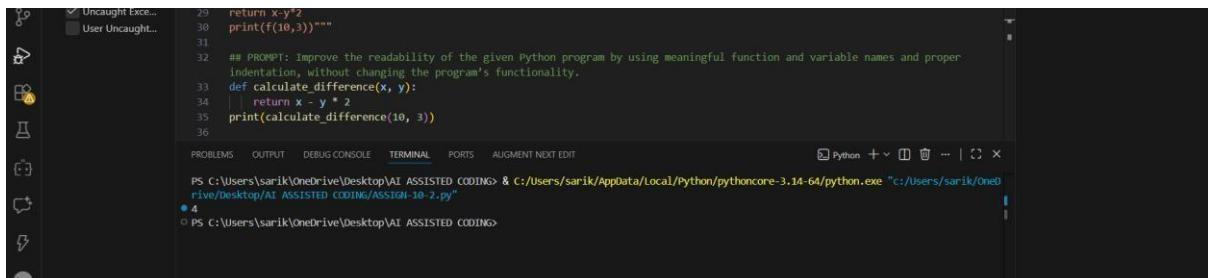
**Prompt:** Improve the readability of the given Python program by using meaningful function and variable names and proper indentation, without changing the program's functionality. **Corrected Input Code:**



A screenshot of the Visual Studio Code interface. The title bar says "AI ASSISTED CODING". The Explorer sidebar shows "RUN AND DEBUG" and "BREAKPOINTS" sections. The main editor window contains the following Python code:

```
ASSIGN-10-2.py
27     """## TASK-3:
28     def f(x,y):
29         return x-y**2
30     print(f(10,3))"""
31
32     ## PROMPT: Improve the readability of the given Python program by using meaningful function and variable names and proper
33     # indentation, without changing the program's functionality.
34     def calculate_difference(x, y):
35         return x - y ** 2
36     print(calculate_difference(10, 3))
```

### Output:



A screenshot of the Visual Studio Code interface. The title bar says "Python". The Explorer sidebar shows "PROBLEMS", "OUTPUT", "DEBUG CONSOLE", "TERMINAL", "PORTS", and "AUGMENT NEXT EDIT" sections. The Terminal tab is active, showing the command line output:

```
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-10-2.py"
● 4
○ PS C:\Users\sarik\OneDrive\Desktop\Desktop\AI ASSISTED CODING>
```

**Explanation:** The function and variable names were updated to clearly describe their purpose, making the code easier to understand. Proper indentation and clearer expressions were also applied while preserving the original logic.

## TASK-4: Structural Refactoring Sample Input

### Code:

**Prompt:** Modify the given Python code by eliminating repetitive statements and rewriting it using reusable functions to improve structure and maintainability.

### Corrected Input Code:

```
37     """## TASK-4:
38     print("Hello Ram")
39     print("Hello Sita")
40     print("Hello Ravi")"""
41
42     ## PROMPT: : Modify the given Python code by eliminating repetitive statements and rewriting it using reusable functions to
43     # improve structure and maintainability.
44     def greet(name):
45         print("Hello " + name)
46     greet("Ram")
47     greet("Sita")
48     greet("Ravi")
```

### Output:

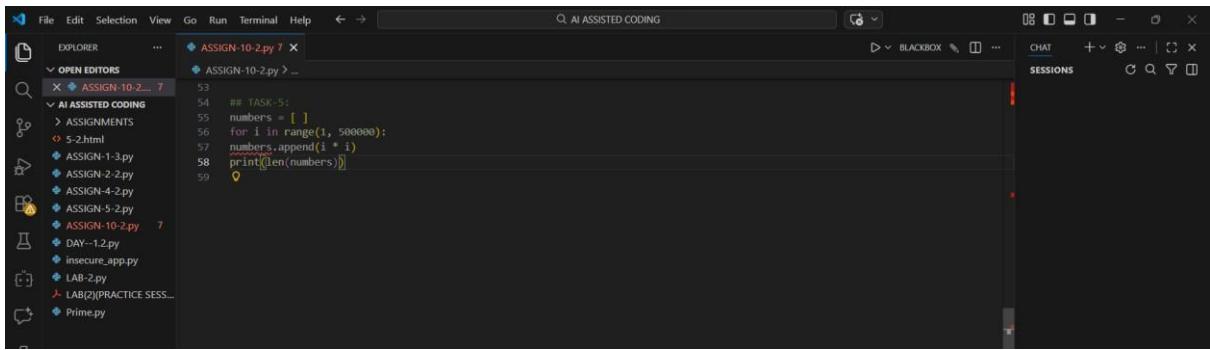
```
40     print("Hello " + name)
41
42     ## PROMPT: : Modify the given Python code by eliminating repetitive statements and rewriting it using reusable functions to
43     # improve structure and maintainability.
44     def greet(name):
45         print("Hello " + name)
46     greet("Ram")
47     greet("Sita")
48     greet("Ravi")
```

PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-10-2.py"
Hello Ram
Hello Sita
Hello Ravi

**Explanation:** The repetitive print statements were replaced with a reusable function that accepts a name as input. This approach reduces redundancy and makes the code more scalable and easier to modify.

### TASK-5: Efficiency Enhancement Sample

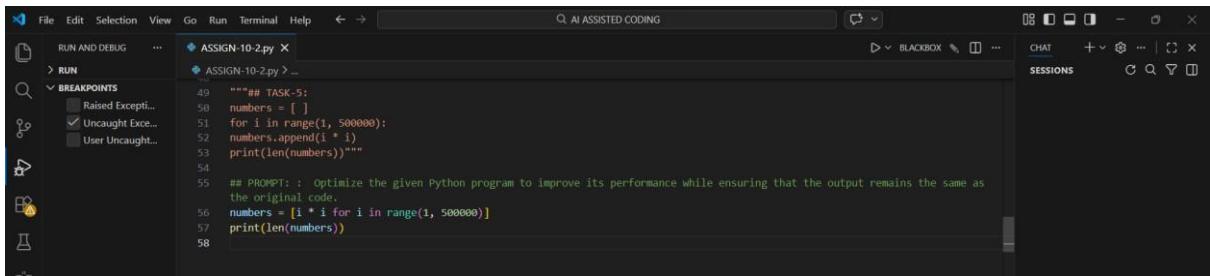
#### Input Code:



```
ASSIGN-10-2.py
53
54     ## TASK-5:
55     numbers = []
56     for i in range(1, 500000):
57         numbers.append(i * i)
58     print(len(numbers))
59 
```

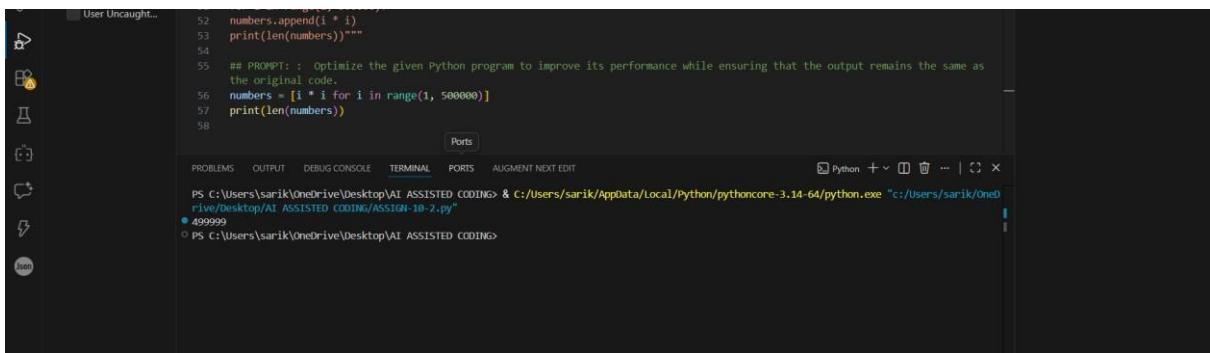
**Prompt:** Optimize the given Python program to improve its performance while ensuring that the output remains the same as the original code.

### Corrected Input Code:



```
ASSIGN-10-2.py
49     """## TASK-5:
50     numbers = []
51     for i in range(1, 500000):
52         numbers.append(i * i)
53     print(len(numbers))"""
54
55     ## PROMPT: : Optimize the given Python program to improve its performance while ensuring that the output remains the same as the original code.
56     numbers = [i * i for i in range(1, 500000)]
57     print(len(numbers))
58 
```

### Output:



```
User Uncaught...
52     numbers.append(i * i)
53     print(len(numbers))"""
54
55     ## PROMPT: : Optimize the given Python program to improve its performance while ensuring that the output remains the same as the original code.
56     numbers = [i * i for i in range(1, 500000)]
57     print(len(numbers))
58
Ports
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS AUGMENT NEXT EDIT
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & c:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-10-2.py"
● 499999
○ PS c:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>
```

**Explanation:** The loop-based implementation was optimized using a list comprehension, which is faster and more concise in Python. This improves performance while producing the same output as the original code.