

GIRUGULA VARSHINI

2403A51L14

## ASSIGNMENT-10.2

### Code Review and Quality: Using AI to Improve Code

#### Quality and Readability

#### Task Description -1(Error Detection and Correction)

**Prompt:** Analyze the given Python code, identify all syntax and logical errors, and provide a corrected executable version with brief explanations

#### AI-Generated Code:

The screenshot shows a VS Code editor with a Python file named 'Untitled-1.py'. The code defines a function 'calculate\_total' and uses it to calculate the sum of a list of numbers. The AI-generated correction is shown in the 'PROBLEMS' panel, which lists the following changes:

- Added missing colons after 'def' and 'for'.
- Replaced 'sum' variable with 'total' to avoid shadowing built-in 'sum'.
- Fixed indentation and returned the correct variable.

The corrected code is as follows:

```
1 """Use AI to analyze a Python script and correct all syntax and logical errors.
2 Sure! Please provide the Python script you would like me to analyze and correct.
3 def calculate_total(nums)
4     sum = 0
5     for n in nums
6     sum += n
7     return total
8 Here is the corrected version of the Python script:'''
9 def calculate_total(nums):
10     total = 0
11     for n in nums:
12         total += n
13     return total
14 # Example usage:
15 numbers = [1, 2, 3, 4, 5]
16 result = calculate_total(numbers)
17 print("The total is:", result)
18 """In this corrected version, I added a colon (:) at the end of the function definition and the for loop. I also changed the variable name from sum to total to avoid shadowing the built-in sum function.
```

#### Output:

The terminal output shows the result of the Python script:

```
PS C:\Users\deep\Downloads\AI> python.exe "C:\Users\deep\Downloads\AI\Untitled-1.py"
The total is: 15
PS C:\Users\deep\Downloads\AI>
```

## Justification:

Missing colon after function definition → Added :

Missing colon in for loop → Added :

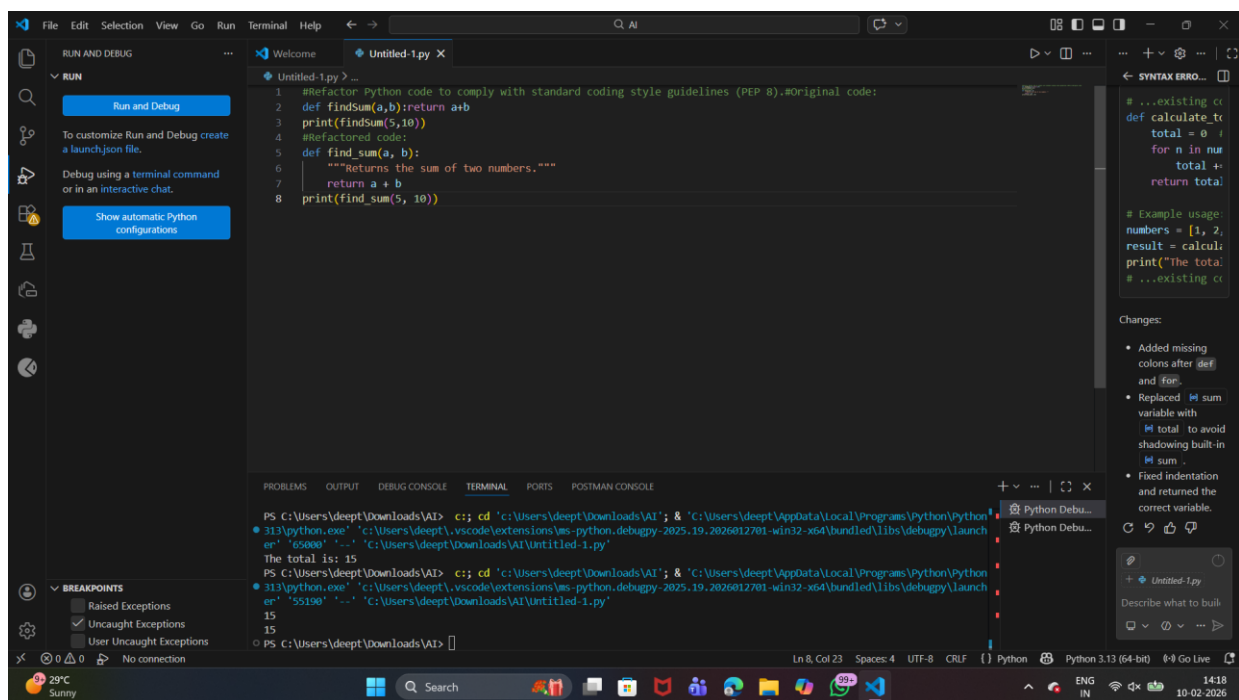
Used sum as variable name (overwrites built-in) → Renamed to total

Returned total though variable was named sum → Corrected consistency

## Task Description -2(Code Style Standardization)

**Prompt :** Refactor the Python code to follow PEP 8 coding standards with proper formatting and readability

## AI-Generated Code:



The screenshot shows the VS Code editor with a Python file named 'Untitled-1.py'. The code is as follows:

```
1 #Refactor Python code to comply with standard coding style guidelines (PEP 8).#Original code:
2 def findSum(a,b):return a+b
3 print(findSum(5,10))
4 #refactored code:
5 def find_sum(a, b):
6     """Returns the sum of two numbers."""
7     return a + b
8 print(find_sum(5, 10))
```

The right sidebar shows the 'SYNTAX ERROR' panel with the following message:

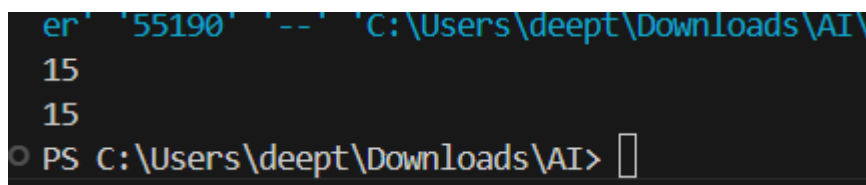
```
...existing c
def calculate t
total = 0
for n in nu
total +
return total

# Example usage:
numbers = [1, 2,
result = calcul
print("The tota
# ...existing c
```

The bottom panel shows the 'TERMINAL' output:

```
PS C:\Users\deept\Downloads\AI> c:; cd 'c:\Users\deept\Downloads\AI'; & 'C:\Users\deept\AppData\Local\Programs\Python\Python
313\python.exe' 'c:\Users\deept\vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundle\libs\debugpy\launch
er' '55190' '--' 'C:\Users\deept\Downloads\AI\Untitled-1.py'
The total is: 15
PS C:\Users\deept\Downloads\AI> c:; cd 'c:\Users\deept\Downloads\AI'; & 'C:\Users\deept\AppData\Local\Programs\Python\Python
313\python.exe' 'c:\Users\deept\vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundle\libs\debugpy\launch
er' '55190' '--' 'C:\Users\deept\Downloads\AI\Untitled-1.py'
15
15
PS C:\Users\deept\Downloads\AI>
```

## OUTPUT:



```
er' '55190' '--' 'C:\Users\deept\Downloads\AI\
15
15
PS C:\Users\deept\Downloads\AI>
```

## Justification:

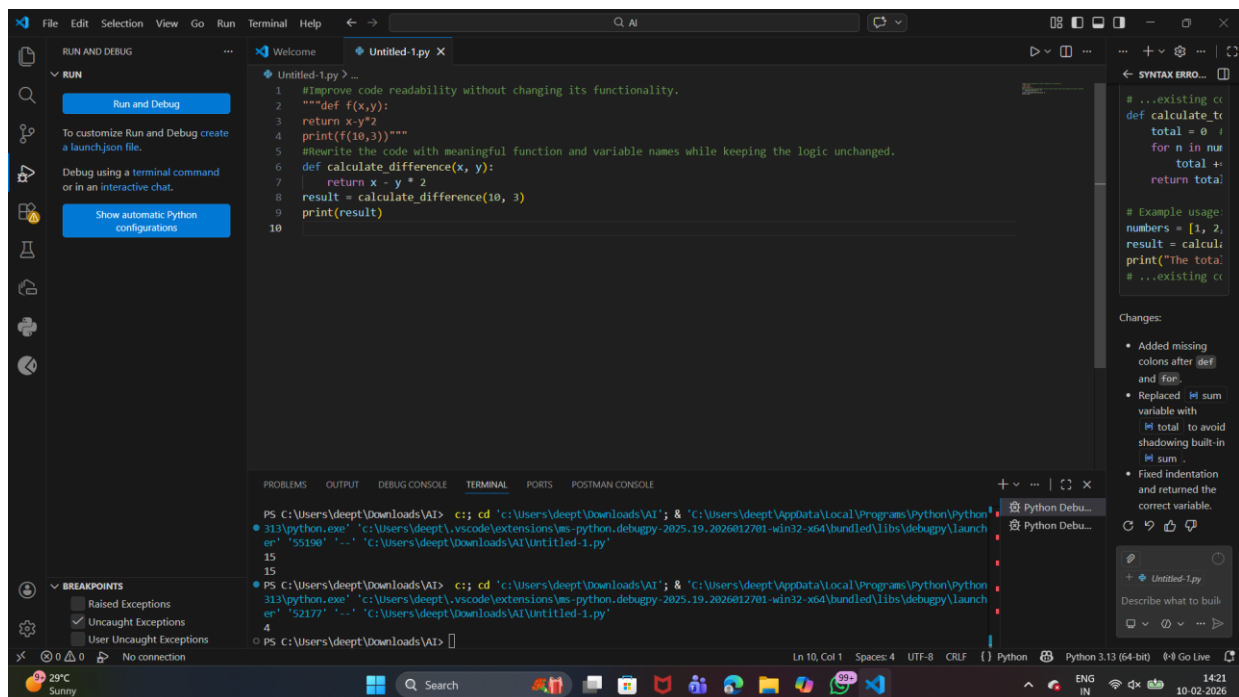
Added proper indentation and spacing

Used snake\_case naming (PEP-8 standard)

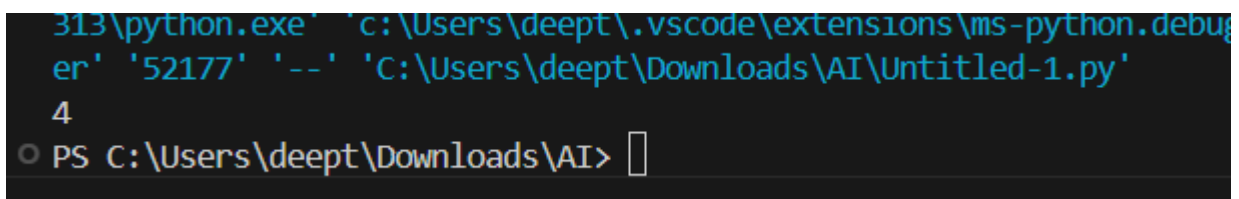
Split single-line function into readable multi-line format

## Task Description -3(Code Clarity Improvement)

**Prompt:** Rewrite the code with meaningful function and variable names while keeping the logic unchanged.



## OUTPUT:



## Justification:

Meaningful function and variable names

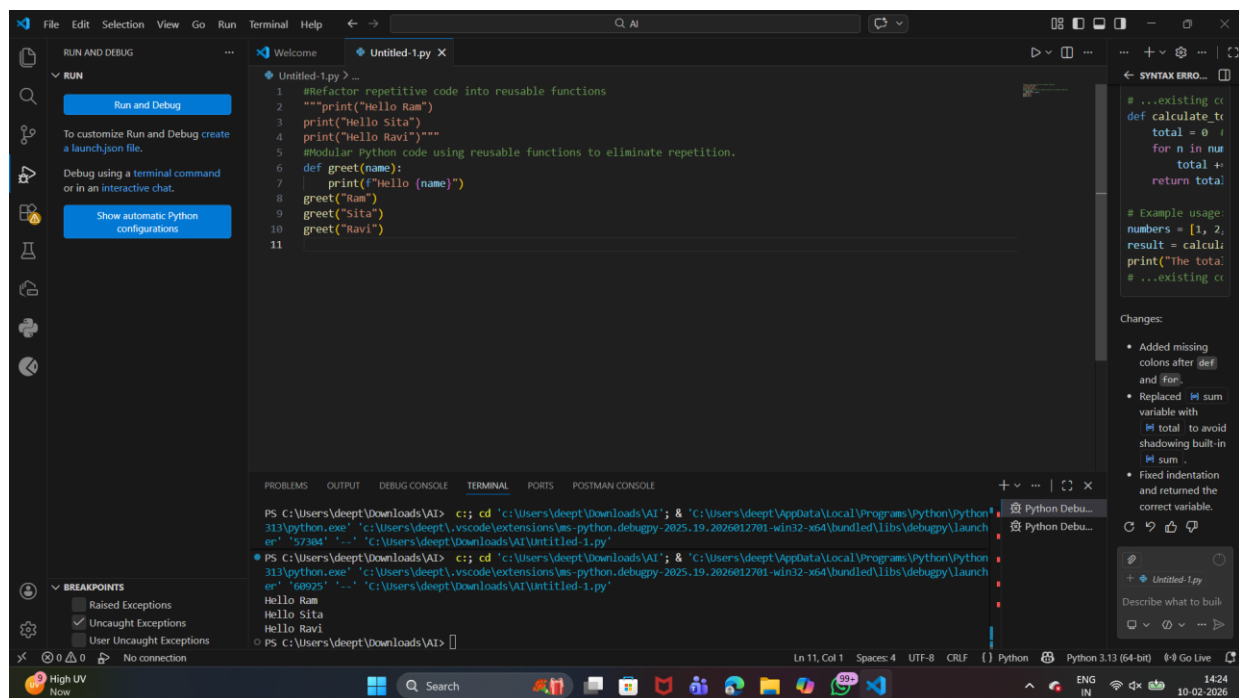
Docstring for clarity

Proper indentation and spacing

## Task Description -4(Structural Refactoring)

**Prompt:** Refactor the repeated print statements into a reusable function.

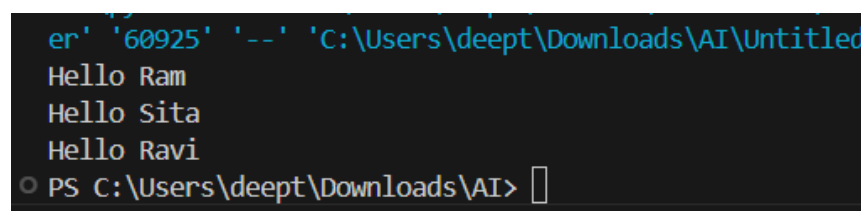
## AI-Generated Code:



```
1 #Refactor repetitive code into reusable functions
2 """print("Hello Ram")
3 print("Hello Sita")
4 print("Hello Ravi)"""
5 #Modular Python code using reusable functions to eliminate repetition.
6 def greet(name):
7     print(f"Hello {name}")
8     greet("Ram")
9     greet("Sita")
10    greet("Ravi")
11
```

```
PS C:\Users\deept\Downloads\AI> cd 'c:\Users\deept\Downloads\AI'; & 'C:\Users\deept\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\deept\.vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundle\libs\debugpy\launcher' '57384' '--' 'C:\Users\deept\Downloads\AI\Untitled-1.py'
Hello Ram
Hello Sita
Hello Ravi
PS C:\Users\deept\Downloads\AI>
```

## OUTPUT:



```
er' '60925' '--' 'C:\Users\deept\Downloads\AI\Untitled-1.py'
Hello Ram
Hello Sita
Hello Ravi
PS C:\Users\deept\Downloads\AI>
```

## Justification:

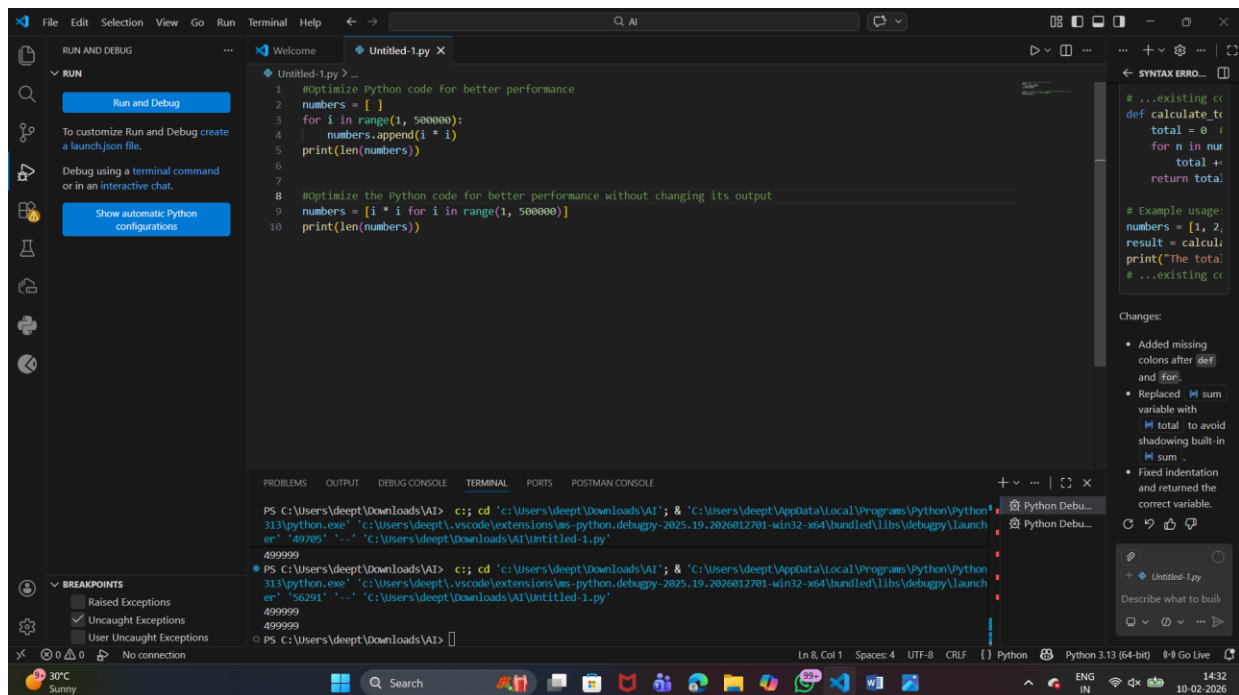
Eliminated repetition

Created reusable function greet()

## Task Description -5(Efficiency Enhancement):

**Prompt:** Optimize the Python code for better performance without changing its output.

## AI-Generated Code:



The screenshot shows the Visual Studio Code interface with a Python file named 'Untitled-1.py'. The code is as follows:

```
1 #Optimize Python code for better performance
2 numbers = []
3 for i in range(1, 500000):
4     numbers.append(i * i)
5 print(len(numbers))
6
7
8 #Optimize the Python code for better performance without changing its output
9 numbers = [i * i for i in range(1, 500000)]
10 print(len(numbers))
```

The right sidebar shows a 'SYNTAX ERROR' panel with the following message:

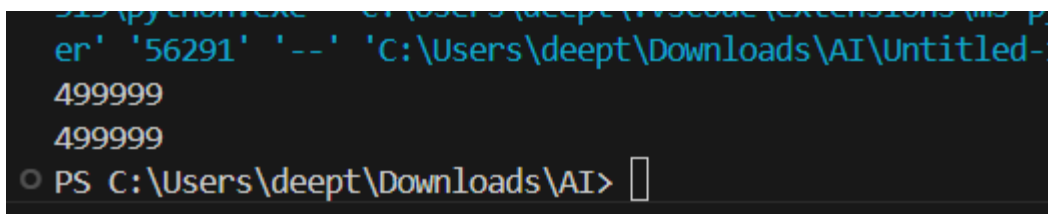
```
< SYNTAX ERROR...
# ...existing co
def calculate_t
total = 0
for n in num
total +=
return total

# Example usage:
numbers = [1, 2,
result = calcul
print("The total
# ...existing co
```

The bottom panel shows the 'TERMINAL' with the following command and output:

```
PS C:\Users\deept\Downloads\AI> c:\; cd 'C:\Users\deept\Downloads\AI'; & 'C:\Users\deept\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\deept\vscode\extensions\ms-python.debugpy-2025.19.2026012701-win32-x64\bundle\libs\debugpy\launcher' '56291' '--' 'C:\Users\deept\Downloads\AI\Untitled-1.py'
499999
499999
PS C:\Users\deept\Downloads\AI>
```

## OUTPUT:



```
499999
499999
PS C:\Users\deept\Downloads\AI>
```

**Justification:**

- Performance Improvements
- Used list comprehension instead of loop
- Faster execution
- Cleaner and more Pythonic code