

## School of Computer Science and Artificial Intelligence

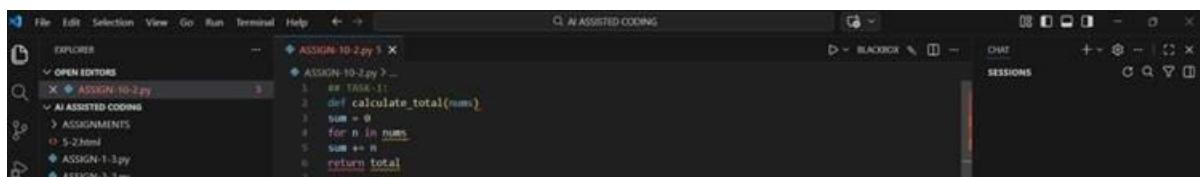
### Lab Assignment # 10.2

<b>Program</b>	: B. Tech (CSE)
<b>Specialization</b>	: -
<b>Course Title</b>	: AI Assisted Coding
<b>Course Code</b>	: 23CS002PC304
<b>Semester</b>	: II
<b>Academic Session</b>	: 2025-2026
<b>Name of Student</b>	: Macharla Naniprasad
<b>Enrollment No.</b>	: 2403A51L11
<b>Batch No.</b>	: 51
<b>Date</b>	: 10/02/26

### Submission Starts here

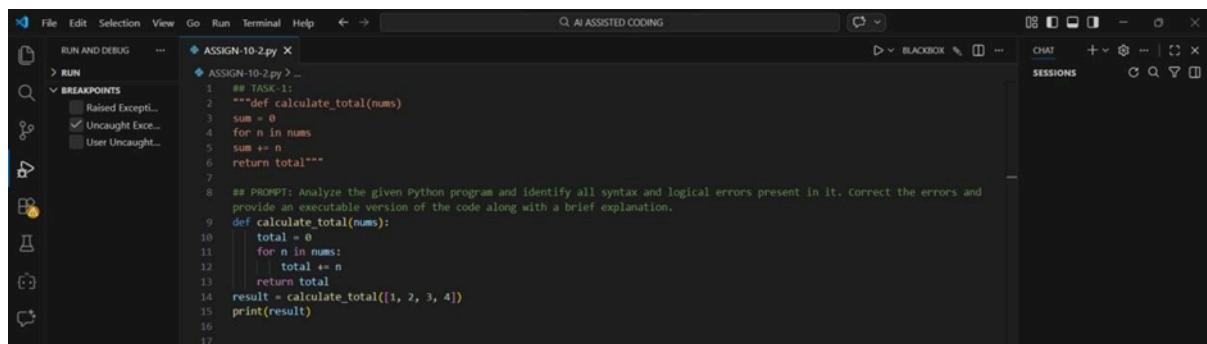
#### **TASK-1: Error Detection and Correction**

##### **Sample Input Code:**



```
ASSIGN-10-2.py
# TASK-1
def calculate_total(nums):
    sum = 0
    for n in nums:
        sum += n
    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:**



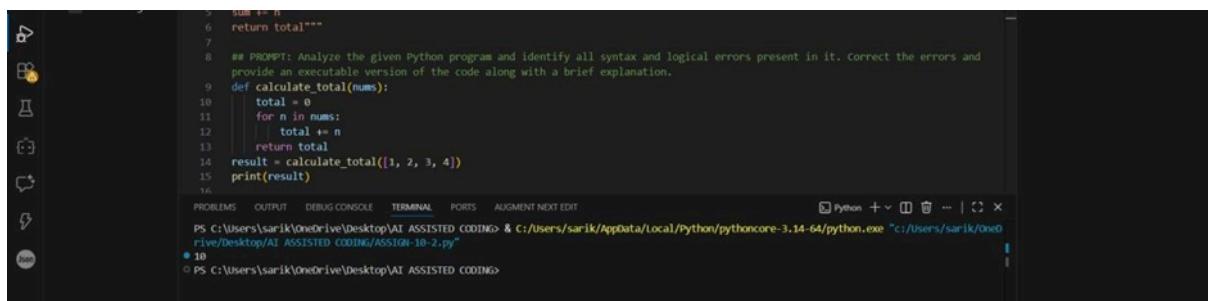
```
ASSIGN-10-2.py
# TASK-1
def calculate_total(nums):
    sum = 0
    for n in nums:
        sum += n
    return total

## 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.

def calculate_total(nums):
    total = 0
    for n in nums:
        total += n
    return total

result = calculate_total([1, 2, 3, 4])
print(result)
```

## Output:



```

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
9     def calculate_total(nums):
10         total = 0
11         for n in nums:
12             total += n
13         return total
14
15 result = calculate_total([1, 2, 3, 4])
16 print(result)
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36

```

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"

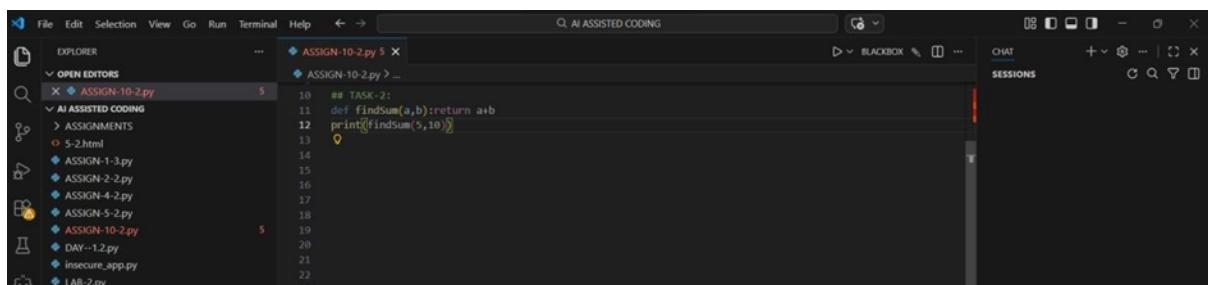
● 10  
○ PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING>

**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:



```

File Edit Selection View Go Run Terminal Help ⏎ → AI ASSISTED CODING
EXPLORER OPEN EDITORS ...
ASSIGN-10-2.py
AI ASSISTED CODING
ASSIGN-1-3.py
ASSIGN-2-2.py
ASSIGN-4-2.py
ASSIGN-5-2.py
ASSIGN-10-2.py
DAY-12.py
insecure_app.py
LAB-2.py

```

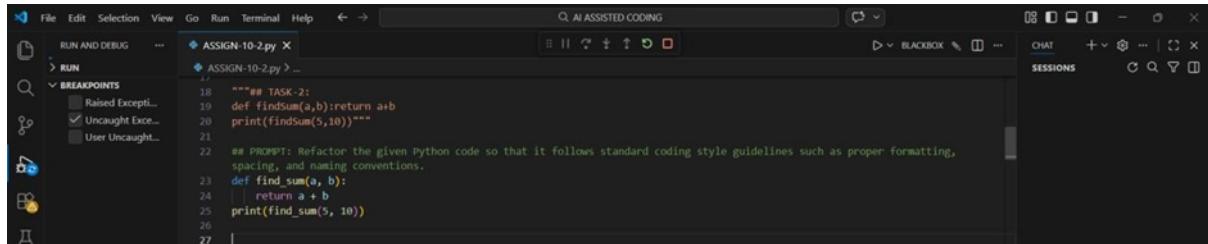
```

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
23
24
25
26
27

```

**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:



```

File Edit Selection View Go Run Terminal Help ⏎ → AI ASSISTED CODING
RUN AND DEBUG ...
RUN
BREAKPOINTS
Raised Except...
Uncaught Except...
User Uncaught...

```

```

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 Visual Studio Code interface. The code editor has the following content:

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

Below the code editor are several icons: file, folder, search, refresh, and others.

The bottom navigation bar includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, PORTS, and AUGMENT NEXT EDIT. The TERMINAL tab is active, showing the command line interface:

```
PS C:\Users\srilk\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/srilk/appData/local/Python/pythoncore-3.14-64/python.exe "c:/users/srilk/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-10-2.py"
● 15
○ PS C:\Users\srilk\OneDrive\Desktop\AI ASSISTED CODING>
```

To the right of the terminal, there is a Python icon followed by a dropdown menu with options like New, Open, Save, and Close, along with other standard window control buttons.

**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:

The screenshot shows the Visual Studio Code interface with the 'AI ASSISTED CODING' extension active. The 'OPEN EDITORS' sidebar highlights 'ASSIGN-10-2.py'. The main editor pane displays the following Python code:

```
def f(x,y):  
    return x-y**2  
print(f(10,3))
```

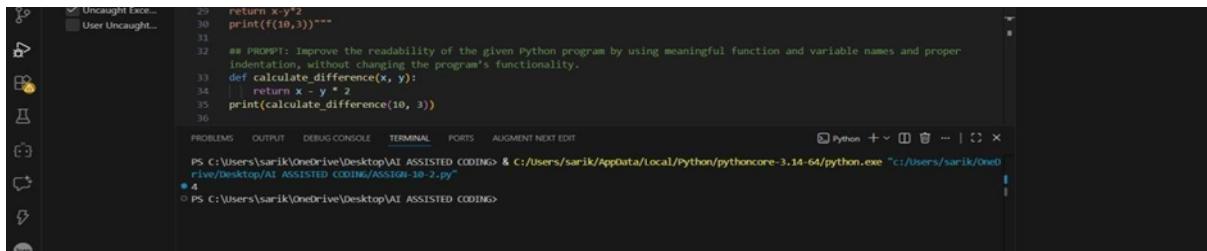
**Prompt:** Improve the readability of the given Python program by using meaningful function and variable names and proper indentation, without changing the program's functionality.

## **Corrected Input Code:**

The screenshot shows the PyCharm IDE interface. The top bar includes 'File', 'Edit', 'Selection', 'View', 'Go', 'Run', 'Terminal', and 'Help' menus. A central search bar says 'AI ASSISTED CODING'. On the left, there's a 'RUN AND DEBUG' section with a 'RUN' button and a 'BREAKPOINTS' section listing 'Raised Except...', 'Uncaught Except...', and 'User Uncaught...'. The main code editor displays Python code for 'ASSIGN-10-2.py'. The code defines a function 'f(x,y)' that returns the product of x and y. It then defines a function 'calculate\_difference(x, y)' that returns the absolute difference between x and y. Finally, it prints the result of calling 'calculate\_difference(10, 3)'. A status bar at the bottom indicates 'BLACKBOX %'.

```
27     """## TASK-3:  
28     def f(x,y):  
29         return x*y  
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:



```

29     return x-y**2
30     print(f'({x},{y})'''')
31
32 ## PROMPT: Improve the readability of the given Python program by using meaningful function names and proper
33 def calculate_difference(x, y):
34     return x - y ** 2
35     print(calculate_difference(10, 3))
36

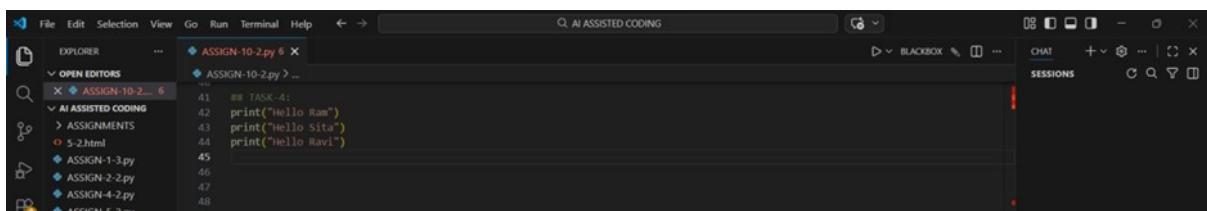
```

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\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:



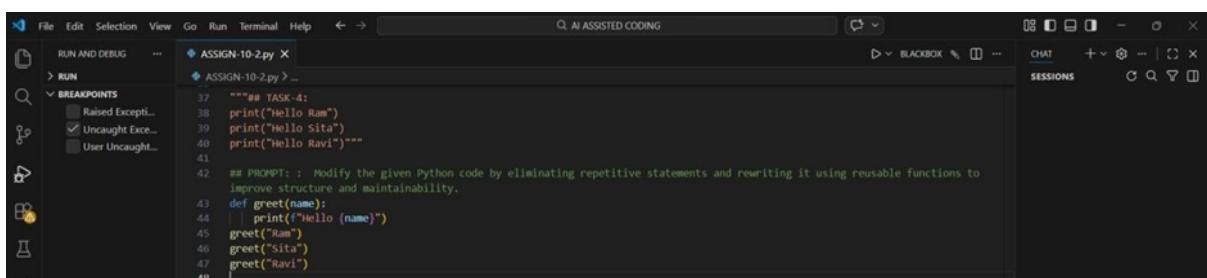
```

41     ## TASK-4:
42     print("Hello Ram")
43     print("Hello Sita")
44     print("Hello Ravi")
45
46
47
48

```

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

### Corrected Input Code:

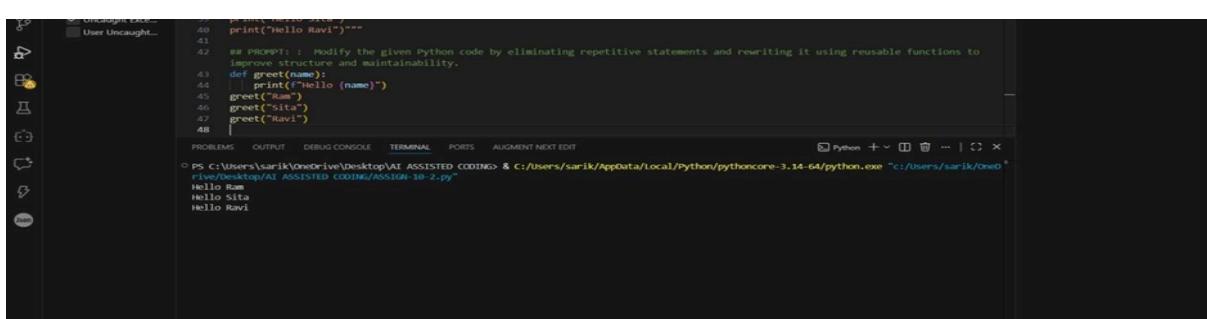


```

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(f"Hello {name}")
46     greet("Ram")
47     greet("Sita")
48     greet("Ravi")
49

```

## Output:



```

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

```

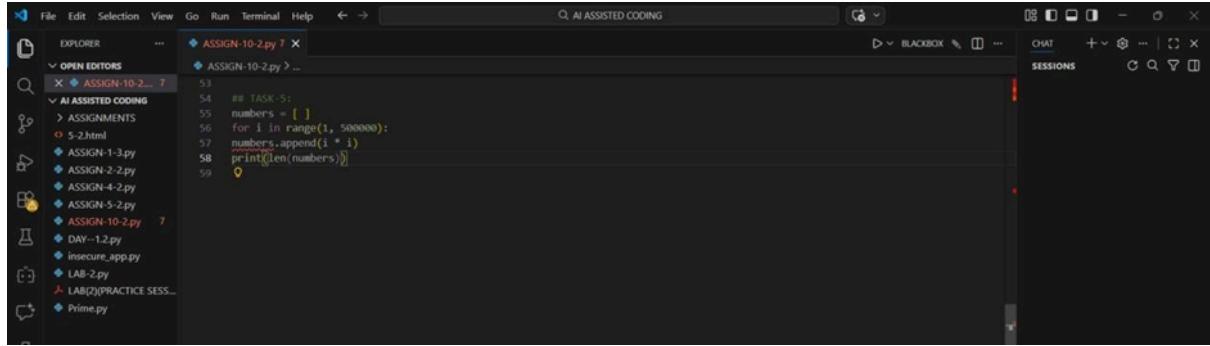
PS C:\Users\sarik\OneDrive\Desktop\AI ASSISTED CODING> & C:/Users/sarik/AppData/Local/Python/pythoncore-3.14-64/python.exe "c:/Users/sarik/OneDrive/Desktop/AI ASSISTED CODING/ASSIGN-10-2.py"
Hello Ram
Hello Sita
Hello Ravi

**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:**



```

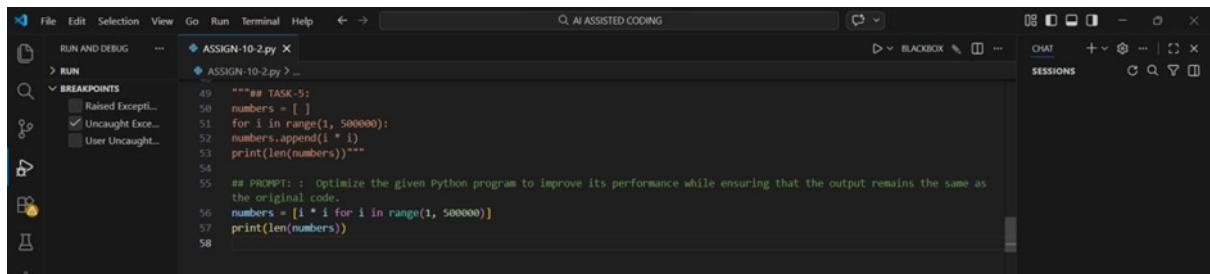
File Edit Selection View Go Run Terminal Help ⏎ → AI ASSISTED CODING
OPEN EDITORS ASSIGN-10-2.py ...
EXPLORER ASSIGNMENTS 5-2.html ASSIGN-1-3.py ASSIGN-2-2.py ASSIGN-4-2.py ASSIGN-5-2.py ASSIGN-10-2.py DAY-1-2.py insecure_app.py LAB-2.py LAB2/PRACTICE SESSIONS Prime.py

53     ## TASK-5:
54     numbers = []
55     for i in range(1, 50000):
56         numbers.append(i * i)
57     print(len(numbers))
58
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:**



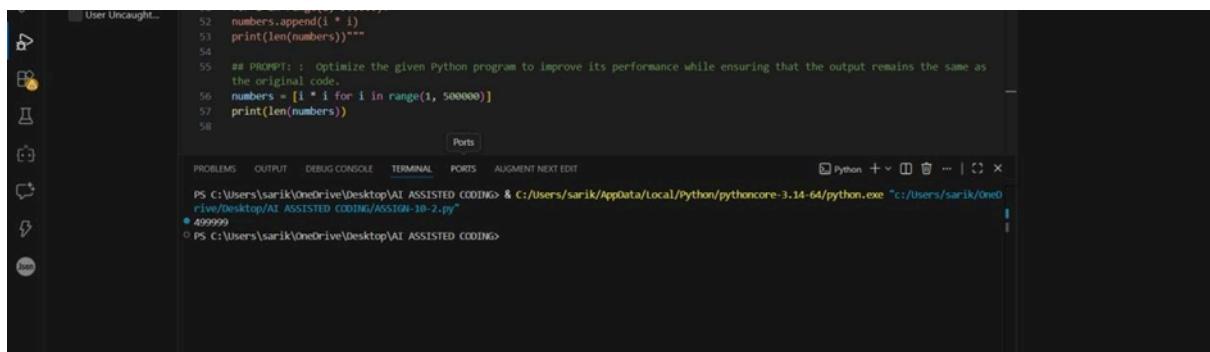
```

File Edit Selection View Go Run Terminal Help ⏎ → AI ASSISTED CODING
RUN AND DEBUG RUN ASSIGN-10-2.py ...
BREAKPOINTS Raised Except... Uncaught Except... User Uncaught...
EXPLORER RUN ASSIGNMENTS 5-2.html ASSIGN-1-3.py ASSIGN-2-2.py ASSIGN-4-2.py ASSIGN-5-2.py ASSIGN-10-2.py DAY-1-2.py insecure_app.py LAB-2.py LAB2/PRACTICE SESSIONS Prime.py

49     """# TASK-5:
50     numbers = []
51     for i in range(1, 50000):
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, 50000)]
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, 50000)]
57     print(len(numbers))
58

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.