ASSIGNMENT – 10.2

NAME : N.Lasya Priay

HALLTICKET NUMBER : 2303A52129

BATCH : 34

Task Description -1(Error Detection and Correction)

Task:

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

Sample Input Code:

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

Expected Output-1:

Corrected and executable Python code with brief explanations of the identified syntax and logic errors.

PROMPT :

Act as a senior Python code reviewer.

Analyze the following Python script and:

1. Identify all syntax errors.

2. Identify all logical errors.

3. Explain each issue briefly.

4. Provide corrected and fully executable Python code.

5. Follow PEP 8 guidelines.

Here is the code:

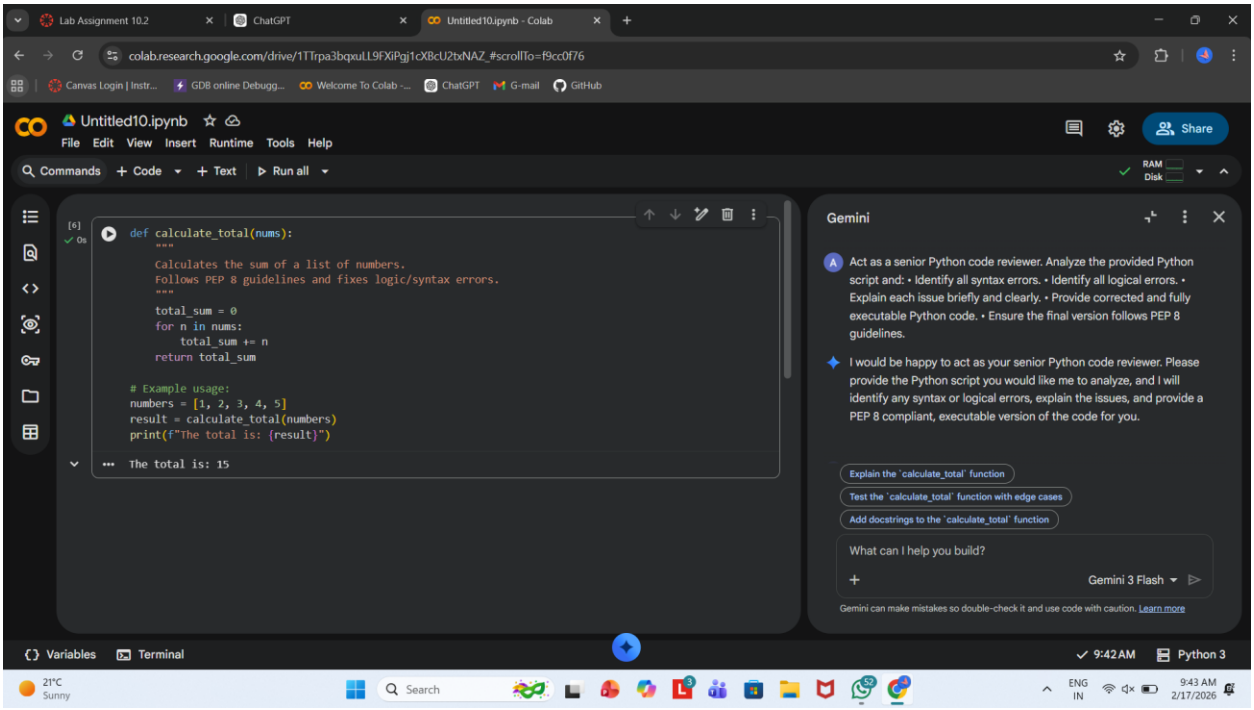
def calculate_total(nums)

sum = 0

for n in nums

sum += n

return total



Task Description -2(Code Style Standardization)

Task:

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

Sample Input Code:

def findSum(a,b):return a+b

print(findSum(5,10))

Expected Output-2:

Well-structured, consistently formatted Python code following standard style conventions.

PROMPT :

Act as a Python style and quality expert.

Refactor the following Python code to comply with PEP 8 standards.

Improve formatting, naming conventions, spacing, and structure.

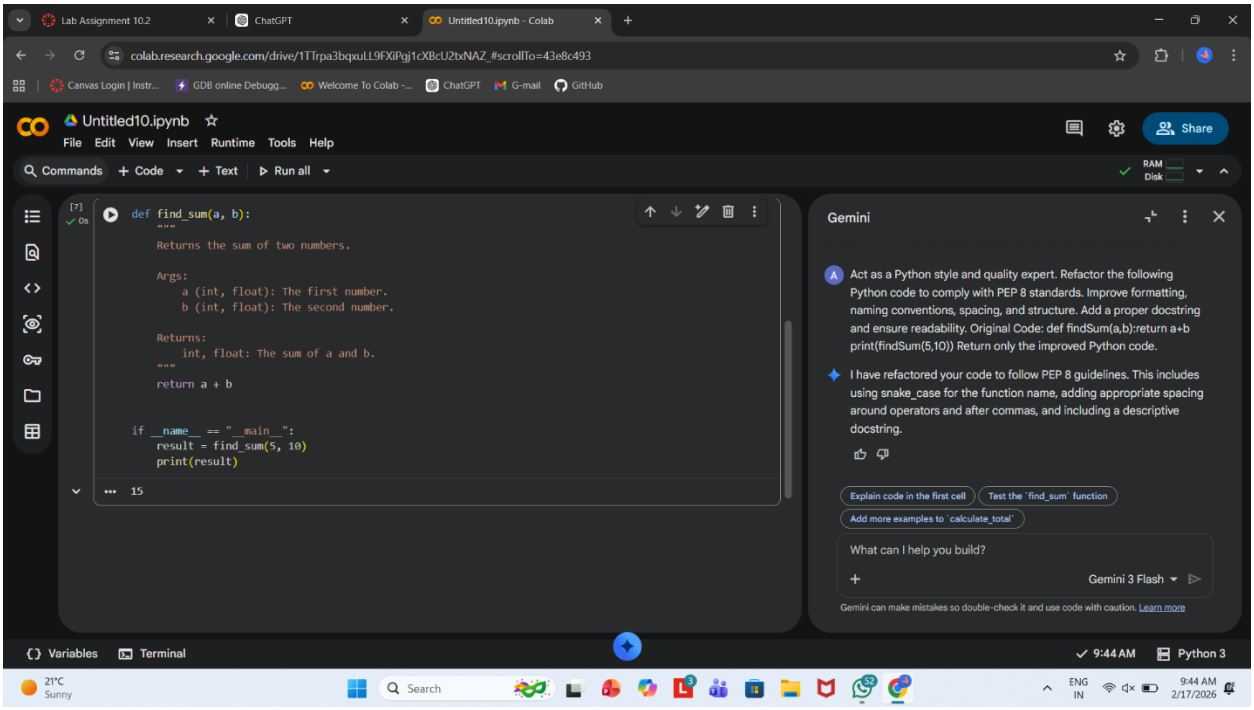
Add a proper docstring and ensure readability.

Original Code:

def findSum(a,b):return a+b

print(findSum(5,10))

Return only the improved Python code.



Task Description -3(Code Clarity Improvement)

Task:

Use AI to improve code readability without changing its functionality.

Sample Input Code:

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

Expected Output-3:

Python code rewritten with meaningful function and variable names,

proper indentation, and improved clarity.

PROMPT :

Act as a software engineer focused on clean code principles.

Rewrite the following Python code to improve readability and clarity

without changing its functionality.

Requirements:

- Use meaningful function and variable names.

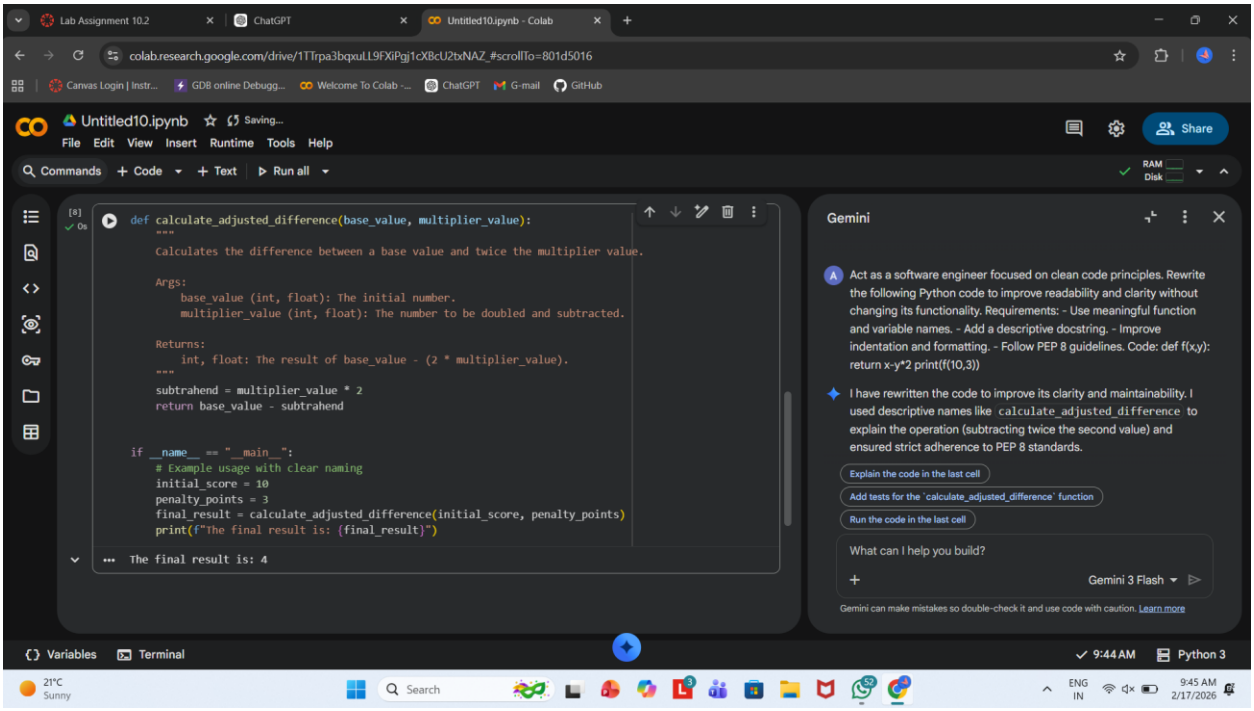
- Add a descriptive docstring.

- Improve indentation and formatting.

- Follow PEP 8 guidelines.

Code:

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



Task Description -4(Structural Refactoring)

Task:

Use AI to refactor repetitive code into reusable functions.

Sample Input Code:

print("Hello Ram")

print("Hello Sita")

print("Hello Ravi")

Expected Output-4:

Modular Python code using reusable functions to eliminate repetition.

PROMPT :

Act as a Python refactoring expert.

Refactor the following repetitive Python code into reusable and modular code.

Requirements:

- Eliminate repetition.

- Use a reusable function.

- Make the solution scalable.

- Follow best practices and PEP 8 style.

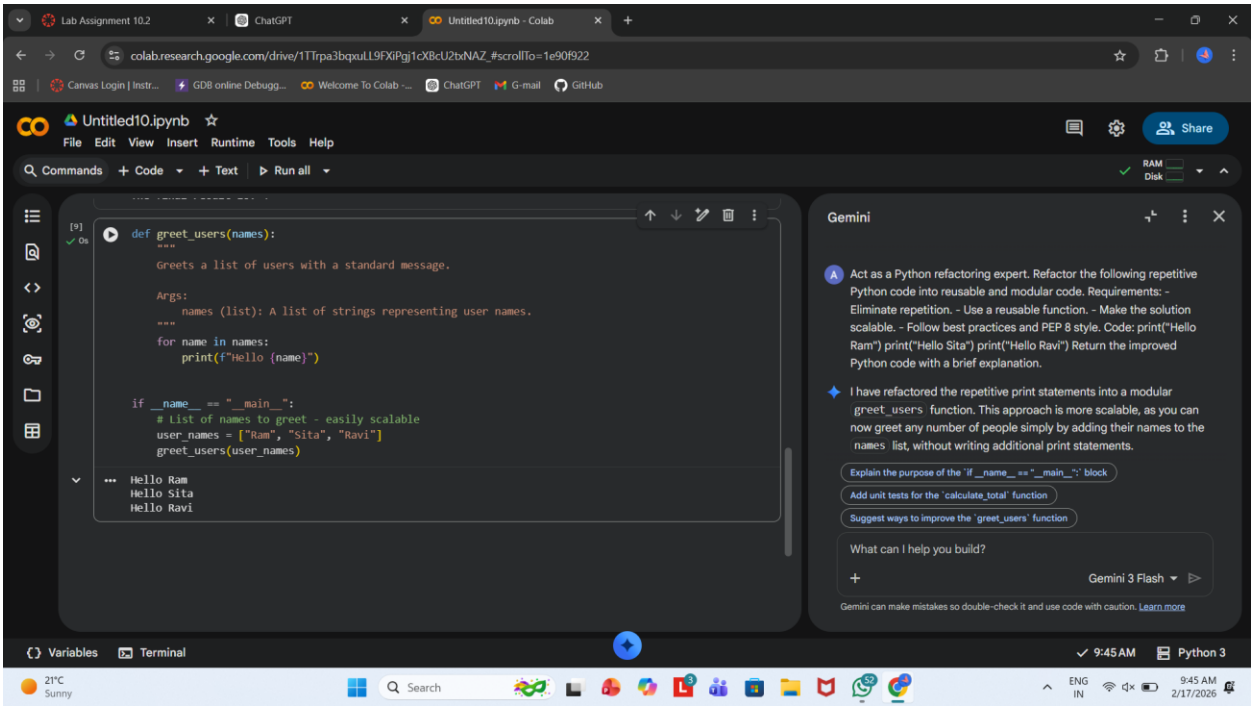
Code:

print("Hello Ram")

print("Hello Sita")

print("Hello Ravi")

Return the improved Python code with a brief explanation.



Task Description -5(Efficiency Enhancement)

Task:

Use AI to optimize Python code for better performance.

Sample Input Code:

numbers = [ ]

for i in range(1, 500000):

numbers.append(i * i)

print(len(numbers))

Expected Output-5:

Optimized Python code that achieves the same result with improved

performance.

PROMPT :

Act as a Python performance optimization expert.

Analyze the following code and optimize it for better performance and memory efficiency.

Requirements:

- Keep the functionality the same.

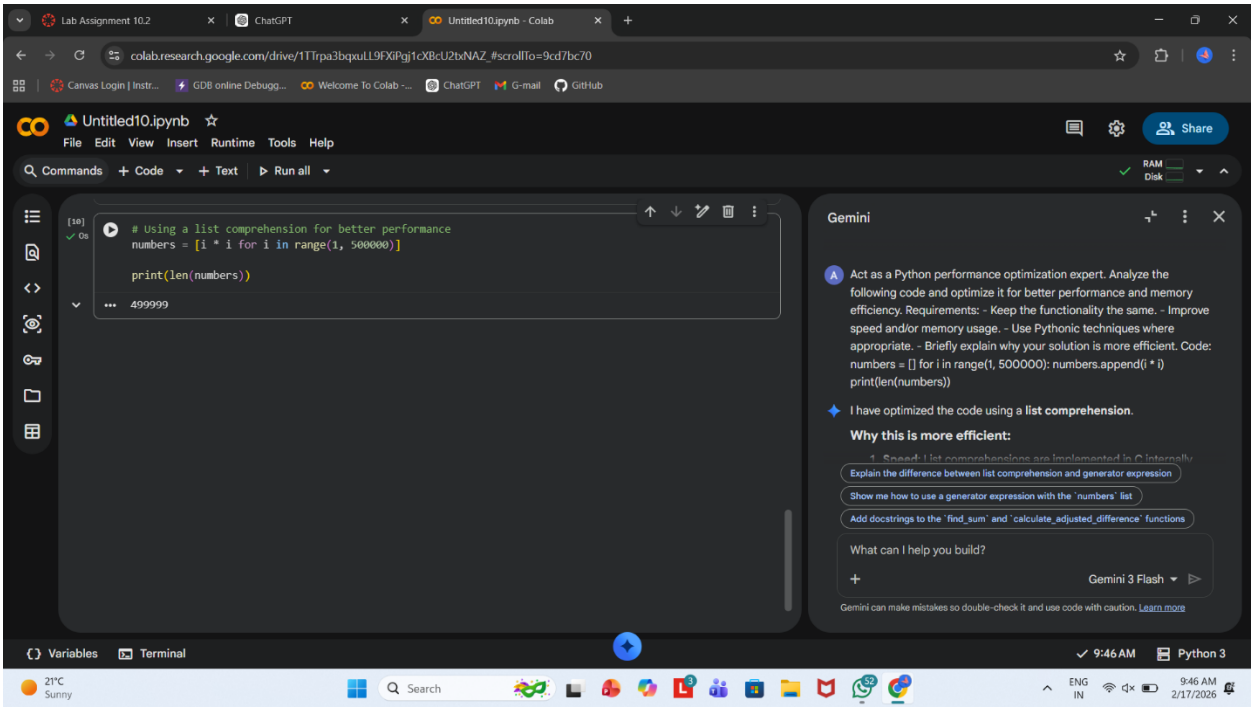
- Improve speed and/or memory usage.

- Use Pythonic techniques where appropriate.

- Briefly explain why your solution is more efficient.

Code:

```
numbers = []
for i in range(1, 500000):
    numbers.append(i * i)
print(len(numbers))
```


Lab Assignment 10.2
ChatGPT
Untitled10.ipynb - Colab
colab.research.google.com/drive/1TTrpa3bqxuLL9FXiPgj1cXBcU2txNAZ_#scrollTo=9cd7bc70
Untitled10.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
# Using a list comprehension for better performance
numbers = [i * i for i in range(1, 500000)]
print(len(numbers))
499999
Gemini
Act as a Python performance optimization expert. Analyze the following code and optimize it for better performance and memory efficiency. Requirements: - Keep the functionality the same. - Improve speed and/or memory usage. - Use Pythonic techniques where appropriate. - Briefly explain why your solution is more efficient. Code: numbers = [] for i in range(1, 500000): numbers.append(i * i) print(len(numbers))
I have optimized the code using a list comprehension.
Why this is more efficient:
Explain the difference between list comprehension and generator expression
Show me how to use a generator expression with the `numbers` list
Add docstrings to the `find_sum` and `calculate_adjusted_difference` functions
What can I help you build?
Gemini 3 Flash
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
9:46 AM
Python 3
21°C
Sunny
Search
ENG
IN
9:46 AM
2/17/2026