ASSIGNMENT – 10.2

NAME : BOPPIDI MANALI REDDY

HALLTICKET NUMBER : 2303A52187

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.

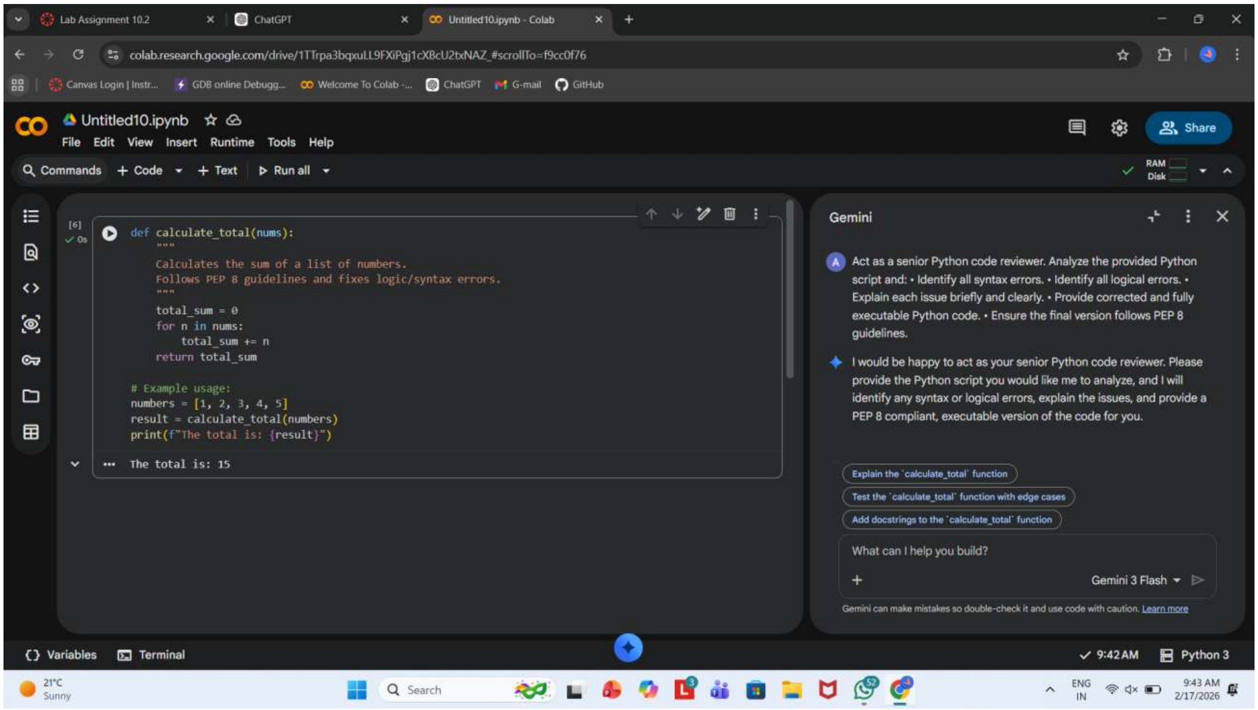
3. Explain each issue briefly.

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
```



```
def calculate_total(nums):
    """
    Calculates the sum of a list of numbers.
    Follows PEP 8 guidelines and fixes logic/syntax errors.
    """
    total_sum = 0
    for n in nums:
        total_sum += n
    return total_sum

# Example usage:
numbers = [1, 2, 3, 4, 5]
result = calculate_total(numbers)
print(f"The total is: {result}")
```

```
The total is: 15
```

Act as a senior Python code reviewer. Analyze the provided Python script and: • Identify all syntax errors. • Identify all logical errors. • Explain each issue briefly and clearly. • Provide corrected and fully executable Python code. • Ensure the final version follows PEP 8 guidelines.

I would be happy to act as your senior Python code reviewer. Please provide the Python script you would like me to analyze, and I will identify any syntax or logical errors, explain the issues, and provide a PEP 8 compliant, executable version of the code for you.

Explain the `calculate_total` function
Test the `calculate_total` function with edge cases
Add docstrings to the `calculate_total` function
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
9:42 AM
Python 3
21°C
Sunny
Search
ENG
IN
9:43 AM
2/17/2026

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.

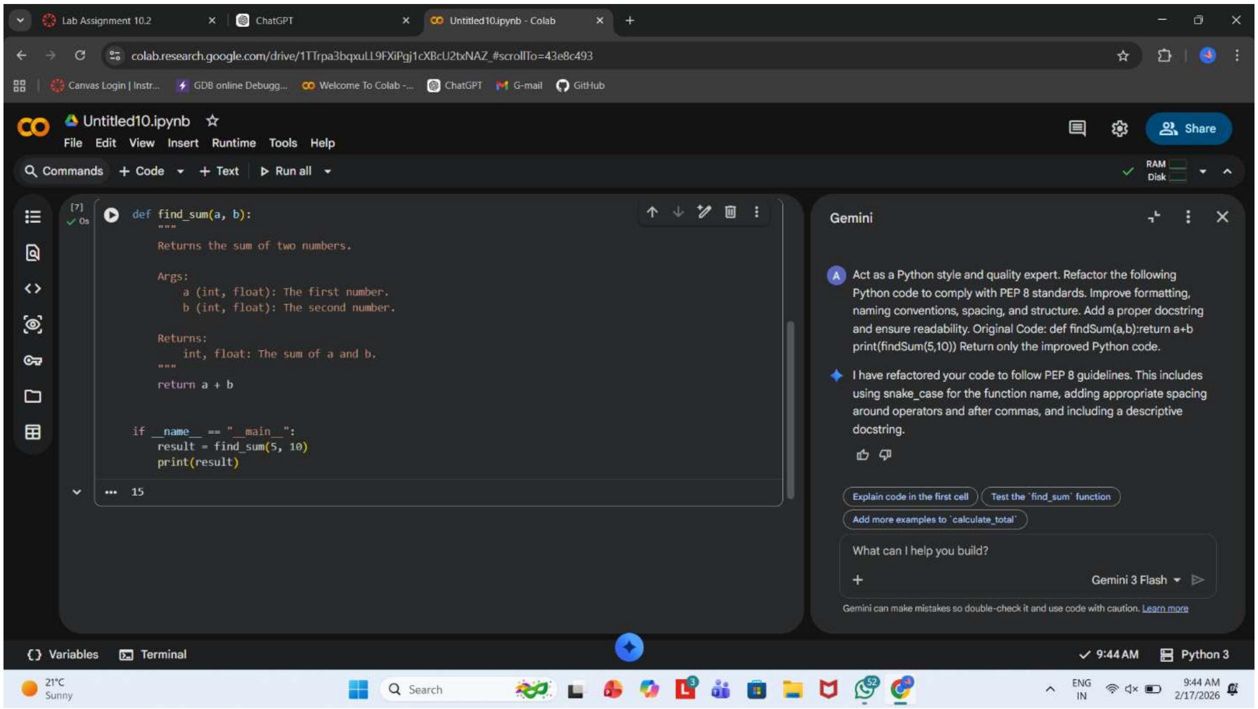
Add a proper docstring and ensure readability.

Original Code:

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

print(findSum(5,10))

Return only the improved Python code.



```
def find_sum(a, b):
    """
    Returns the sum of two numbers.

    Args:
        a (int, float): The first number.
        b (int, float): The second number.

    Returns:
        int, float: The sum of a and b.
    """
    return a + b


if __name__ == "__main__":
    result = find_sum(5, 10)
    print(result)
```

```
15
```

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.

I have refactored your code to follow PEP 8 guidelines. This includes using snake_case for the function name, adding appropriate spacing around operators and after commas, and including a descriptive docstring.

Explain code in the first cell
Test the `find_sum` function
Add more examples to `calculate_total`
What can I help you build?
Gemini 3 Flash
Gemini can make mistakes so double-check it and use code with caution. Learn more

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.

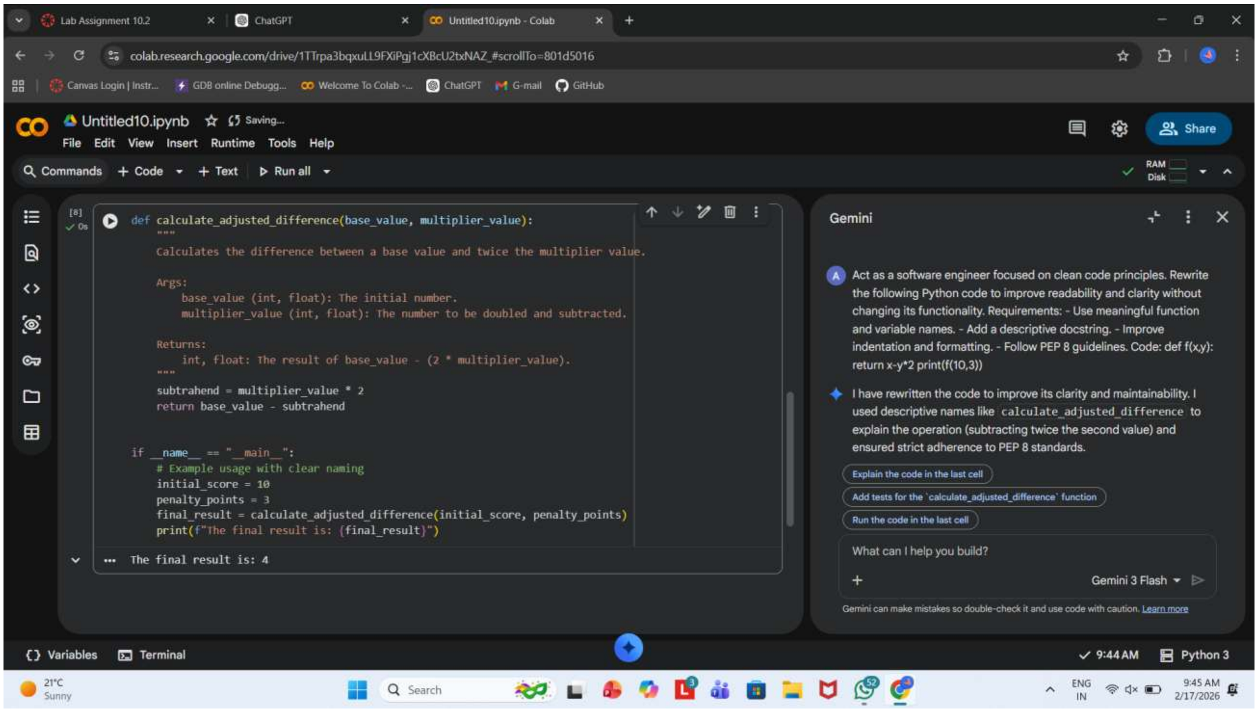
Rewrite the following Python code to improve readability and clarity without changing its functionality.

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))
```



```
def calculate_adjusted_difference(base_value, multiplier_value):
    """
    Calculates the difference between a base value and twice the multiplier value.

    Args:
        base_value (int, float): The initial number.
        multiplier_value (int, float): The number to be doubled and subtracted.

    Returns:
        int, float: The result of base_value - (2 * multiplier_value).
    """
    subtrahend = multiplier_value * 2
    return base_value - subtrahend


if __name__ == "__main__":
    # Example usage with clear naming
    initial_score = 10
    penalty_points = 3
    final_result = calculate_adjusted_difference(initial_score, penalty_points)
    print(f"The final result is: {final_result}")
```

```
The final result is: 4
```

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))

I have rewritten the code to improve its clarity and maintainability. I used descriptive names like `calculate_adjusted_difference` to explain the operation (subtracting twice the second value) and ensured strict adherence to PEP 8 standards.

Explain the code in the last cell
Add tests for the `calculate_adjusted_difference` function
Run the code in the last cell
What can I help you build?
Gemini 3 Flash
Gemini can make mistakes so double-check it and use code with caution. Learn more
21°C
Sunny
Search
ENG
IN
9:45 AM
2/17/2026

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.

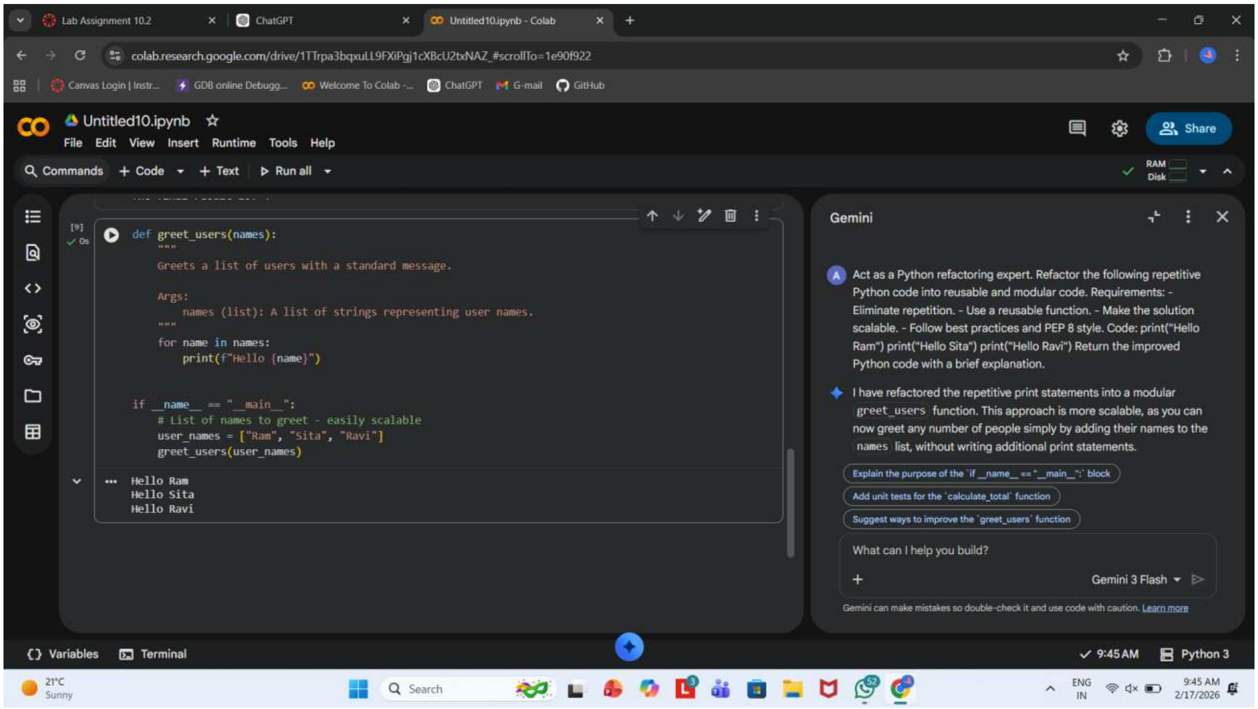
Code:

print("Hello Ram")

print("Hello Sita")

print("Hello Ravi")

Return the improved Python code with a brief explanation.



```
def greet_users(names):
    """
    Greets a list of users with a standard message.

    Args:
        names (list): A list of strings representing user names.
    """
    for name in names:
        print(f"Hello {name}")


if __name__ == "__main__":
    # List of names to greet - easily scalable
    user_names = ["Ram", "Sita", "Ravi"]
    greet_users(user_names)
```

```
Hello Ram
Hello Sita
Hello Ravi
```

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.

I have refactored the repetitive print statements into a modular `greet_users` function. This approach is more scalable, as you can now greet any number of people simply by adding their names to the `names` list, without writing additional print statements.

Explain the purpose of the `if __name__ == "__main__":` block

Add unit tests for the `calculate_total` function

Suggest ways to improve the `greet_users` function

What can I help you build?

Gemini 3 Flash

Gemini can make mistakes so double-check it and use code with caution. Learn more

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.**

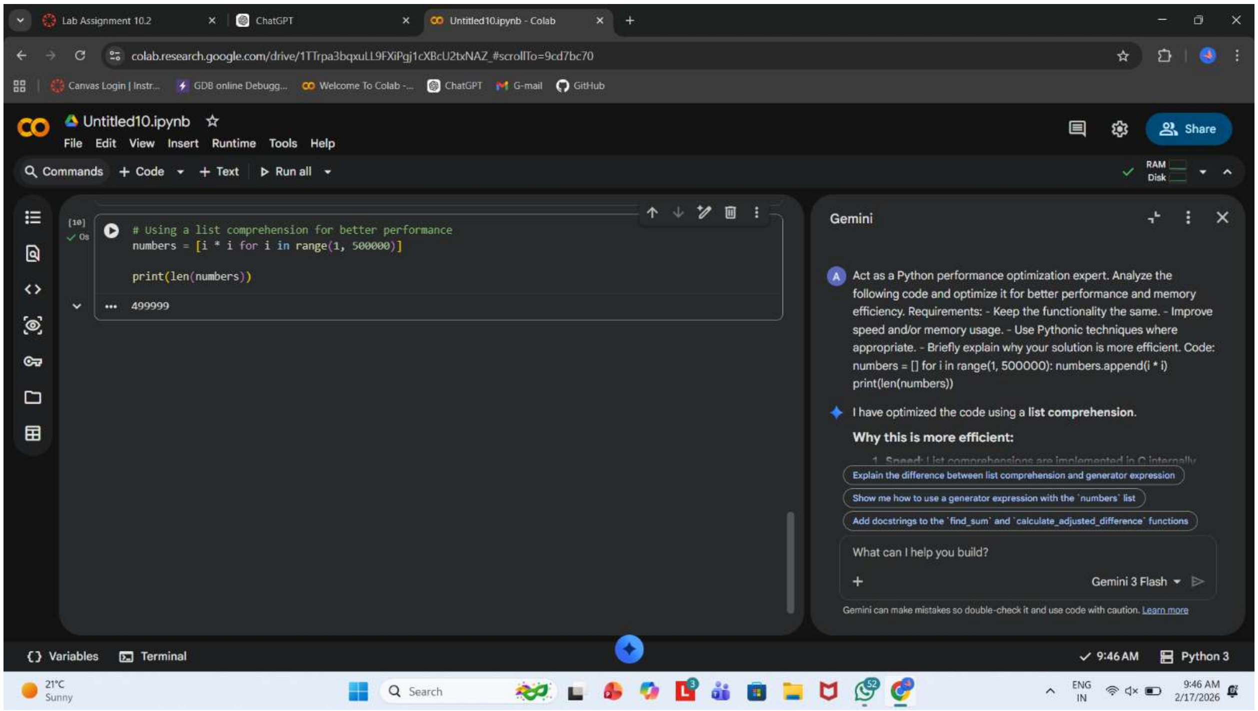
Code:

```
numbers = []

for i in range(1, 500000):

    numbers.append(i * i)

print(len(numbers))
```



```
# Using a list comprehension for better performance
numbers = [i * i for i in range(1, 500000)]

print(len(numbers))
```

```
499999
```

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:

1. **Speed:** List comprehensions are implemented in C internally

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