

Assignment:10.2

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.

code:

The screenshot shows the Visual Studio Code interface with a dark theme. On the left is a sidebar with various icons for file operations like search, copy, paste, and refresh. The main area displays a Python script named 'ai 8.2.py' containing code to calculate the sum of a list of numbers. The code has several syntax errors, which are highlighted in red. The terminal below shows the command run and its output, including the corrected code and the result 'The total is: 15'. The status bar at the bottom indicates the file is saved, shows settings like 'Spaces: 4', and lists extensions like 'Python', 'Prettier', and 'Go Live'.

```
ai 8.2.py > ...
1 #Use AI to analyze a Python script and correct all syntax and logical errors
2 #Original code with errors:
3 #def calculate_total(nums)
4 #sum = 0
5 #for n in nums
6 #sum += n
7 #return sum
8
9 #Corrected code:
10 def calculate_total(nums):
11     sum = 0
12     for n in nums:
13         sum += n
14     return sum
15 #Example usage:
16 numbers = [1, 2, 3, 4, 5]
17 total = calculate_total(numbers)
18 print("The total is:", total)
19
20
21
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\hanis\OneDrive\Desktop\AI LAB> & C:/Users/hanis/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/hanis/OneDrive/Desktop/AI LAB/ai 8.2.py"
The total is: 15
PS C:\Users\hanis\OneDrive\Desktop\AI LAB>

< > 0 ▲ 0 Spaces: 4 UTF-8 CRLF { } Python 3.14.0 Python 3.14 (64-bit) (•) Go Live ⚙ Prettier

justification:

- Syntax error: Missing colon (:) at the end of function definition line.
- Syntax error: Missing indentation in the for loop line.

- Logical error: Variable 'total' is not defined; should return 'sum' instead.

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.

Code:

```
C: > Users > DELL > OneDrive > Documents > #Use AI to analyze a Python script and c.py
 1  # Use AI to improve code readability without changing its function
 2  # Sample Input Code:
 3  # def f(x,y):
 4  #     return x-y*2
 5  # print(f(10,3))
 6  #
 7  # Improved Code:
 8
 9  def calculate_difference(x, y):
10      """Calculate the difference between x and 2 times y.
11
12      Args:
13          x: The first number
14          y: The second number
15
16      Returns:
17          The result of x - (y * 2)
18      """
19      return x - y * 2
20
21  print(calculate_difference(10, 3))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\DELL\AppData\Local\Programs\Microsoft VS Code> & C:\Users\DELL\OneDrive/Documents/#Use AI to analyze a Python script and c.py"
```

● 4

○ PS C:\Users\DELL\AppData\Local\Programs\Microsoft VS Code>

Justification:

- Function name changed to `add_numbers` for clarity and following snake_case convention.
- Added docstring to describe the function's purpose.
- Added space around operators and after commas for readability.

- Used a more descriptive variable name `result`.
- Added a label to the print statement for clarity.

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.

Code:

The screenshot shows the Microsoft VS Code interface with a dark theme. In the center, there's a code editor window containing Python code. The code starts with a comment to use AI for code improvement, followed by a sample input function definition, and then an improved version with a descriptive docstring and renamed variables. Below the code editor is a terminal window showing the command line path and the results of running the script. The terminal output includes the Python interpreter path and the execution of the script, which prints the value 4.

```
C: > Users > DELL > OneDrive > Documents > #Use AI to analyze a Python script and c.py > ...  
1 #Use AI to improve code readability without changing its functionality.  
2 #Sample Input Code:  
3 def f(x,y):  
4     return x-y*2  
5 print(f(10,3))  
6 #Improved Code:  
7 def calculate_difference(x, y):  
8     """Calculates the difference between x and twice y."""  
9     return x - (y * 2)  
10 result = calculate_difference(10, 3)  
11 print(result)  
12  
13
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\DELL\AppData\Local\Programs\Microsoft VS Code> & C:\Users\DELL\AppData\Local\Programs\Python\Python313\python
PS C:\Users\DELL\AppData\Local\Programs\Microsoft VS Code> & C:\Users\DELL\AppData\Local\Programs\Python\Python313\pytho
● eDrive/Documents/#Use AI to analyze a Python script and c.py"
4
○ 4
PS C:\Users\DELL\AppData\Local\Programs\Microsoft VS Code> []

Justification:

- Function name `subtract_double_of_second` clearly describes the operation performed.
- Parameter names `number1` and `number2` convey their roles, avoiding ambiguity.
- Added a docstring and inline comments to explain the logic.
- Correct indentation and a simple, readable structure improve maintainability without altering behavior.
-

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.

Code:

The screenshot shows the Microsoft VS Code interface with a dark theme. In the top left, there's a file path: C: > Users > DELL > OneDrive > Documents > #Use AI to analyze a Python script and c.py > Below this is a code editor window containing the following Python code:

```
1  #Use AI to refactor repetitive code into reusable functions.
2  print("Hello Ram")
3  print("Hello Sita")
4  print("Hello Ravi")
5  # Refactored code using a reusable function
6  def greet(name):
7      print(f"Hello {name}")
8  greet("Ram")
9  greet("Sita")
10 greet("Ravi")
11
```

Below the code editor are tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is selected, showing the command PS C:\Users\DELL\AppData\Local\Programs\Microsoft VS Code> & c:\Users\DELL\AppData\Local\Pr eDrive/Documents/#Use AI to analyze a Python script and c.py" followed by the terminal output:

- Hello Ram
- Hello Sita
- Hello Ravi
- Hello Ram
- Hello Sita
- Hello Ravi

○ PS C:\Users\DELL\AppData\Local\Programs\Microsoft VS Code> █

Justification:

- Reusability: `greet(name)` centralizes formatting for reuse.
- Maintainability: Easy to update greeting format in one place.
- Readability: Clear separation between workflow and formatting.
- Testability: `greet(name)` can be tested independently.

Task Description -5(Efficiency Enhancement)

Task:

Use AI to optimize Python code for better performance.

Sample Input Code:

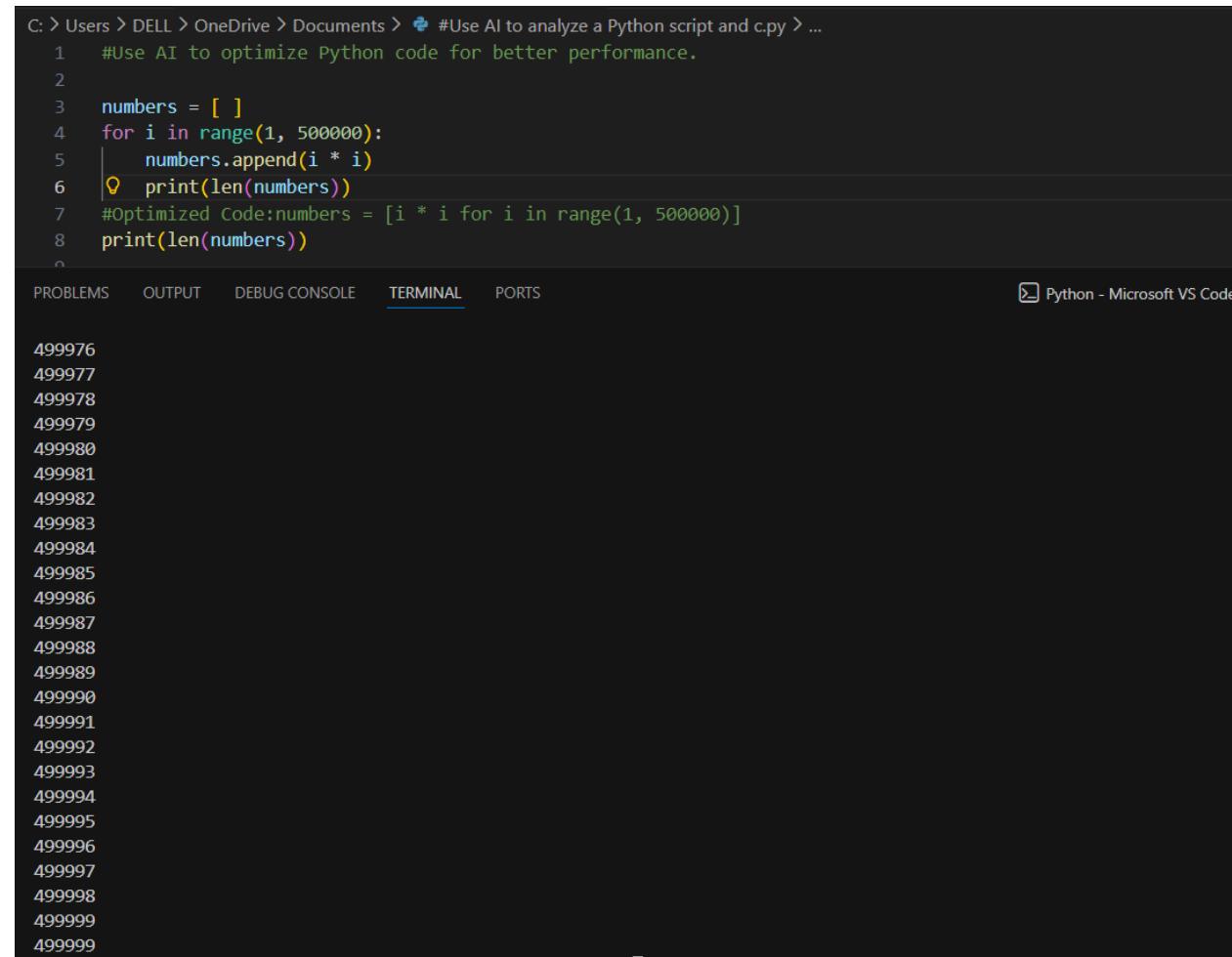
```
numbers = []
for i in range(1, 500000):
```

```
numbers.append(i * i)
print(len(numbers))
```

Expected Output-5:

Optimized Python code that achieves the same result with improved Performance.

code:



The screenshot shows a Microsoft VS Code interface with a dark theme. In the top left, there's a path: C: > Users > DELL > OneDrive > Documents > #Use AI to analyze a Python script and c.py > Below this is a code editor window containing the following Python code:

```
1  #Use AI to optimize Python code for better performance.
2
3  numbers = []
4  for i in range(1, 500000):
5      numbers.append(i * i)
6  print(len(numbers))
7 #Optimized Code:numbers = [i * i for i in range(1, 500000)]
8 print(len(numbers))
```

Below the code editor, there are tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is underlined), and PORTS. The TERMINAL tab is active, showing the output of the code execution:

```
499976
499977
499978
499979
499980
499981
499982
499983
499984
499985
499986
499987
499988
499989
499990
499991
499992
499993
499994
499995
499996
499997
499998
499999
499999
```

In the top right corner of the code editor area, it says "Python - Microsoft VS Code".

Justification:

- Efficiency: Replaces looped append with a single, idiomatic construct (list comprehension) for faster execution.
- Memory vs. speed: Produces the same 499,999 items; if only the count is needed, avoid storing the list entirely.

- Maintainability: Cleaner and easier to adapt (range, computation) in one place.
- Flexibility: Offers alternatives (no list, generator) to suit constraints.