

## ASSIGNMENT - 10.1

---

Name : E mahanth arya

date :- 16/02/2026

Hall ticket no :- 2303A54028

Course :-AI ASSISTANT CODING

---

### Task Description #1 – Syntax and Logic Errors

Task: Use AI to identify and fix syntax and logic errors in a faulty Python script.

Sample Input Code:

```
# Calculate average score of a student
```

```
def calc_average(marks):
```

```
total = 0
```

```
for m in marks:
```

```
total += m
```

```
average = total / len(marks)
```

```
return avrage # Typo here
```

```
marks = [85, 90, 78, 92]
```

```
print("Average Score is ", calc_average(marks))
```

Expected Output:

- Corrected and runnable Python code with explanations of the Fixes
- 

Prompt :

# Identify syntax and logic errors, correct the code, and explain fixes briefly.

```

HELLO WORLD.py > ...
1  ## Identify syntax and logic errors, correct the code, and explain fixes briefly.
2  # Calculate average score of a student
3  #def calc_average(marks):
4  #total = 0
5  #for m in marks:
6  #    total += m
7  #average = total / len(marks)
8  #return average # Typo fixed here
9  #marks = [85, 90, 78, 92]
10 #print("Average Score is ", calc_average(marks))
11 # The original code had indentation issues and a typo in the return statement.
12 # The corrected code properly indents the function and fixes the typo in the return statement to ensure
13 def calc_average(marks):
14     total = 0
15     for m in marks:
16         total += m
17     average = total / len(marks)
18     return average
19 marks = [85, 90, 78, 92]
20 print("Average Score is ", calc_average(marks))
21
22

```

OUTPUT :

```

4\bundled\libs\debugpy\launcher' '52874' '--' 'c:\Users\
Average Score is  86.25
PS C:\Users\aryae\OneDrive\Desktop\AI ASSISTANT CODING>

```

## Task Description #2 – PEP 8 Compliance

Task: Use AI to refactor Python code to follow PEP 8 style guidelines.

Sample Input Code:

```

def area_of_rect(L,B) : return L*B
print(area_of_rect(10,20))

```

Expected Output:

- Well-formatted PEP 8-compliant Python code

PROMPT :-

Refactor the following Python code to be fully PEP 8 compliant and return the well-formatted version.

```

#TASK2 :Refactor the following Python code to be fully **PEP 8 compliant** and return the well-for
#Sample Input Code: def area_of_rect(L,B)
# return L*B print(area_of_rect(10,20))
def area_of_rect(L, B):
    return L * B
print(area_of_rect(10, 20))

```

OUTPUT :

```
PS C:\Users\aryae\OneDrive\Desktop\AI ASSISTANT CODING> ^C
```

---

### Task Description #3 – Readability Enhancement

Task: Use AI to make code more readable without changing its logic.

Sample Input Code:

```
def c(x,y):  
    return x*y/100  
a=200  
b=15  
print(c(a,b))
```

Expected Output:

- Python code with descriptive variable names, inline comments, and clear formatting.
- 

### SCREEN SHOT

:

```
# Improve the readability of this Python code  
# Use descriptive variable names, proper formatting, and inline comments  
# Do NOT change the logic  
  
#def c(x, y):  
#return x * y / 100  
  
#a = 200  
#b = 15  
#print(c(a, b))  
def calculate_percentage(value, percentage):  
    """  
    Calculate the percentage of a given value.  
  
    Parameters:  
    value (float): The total value from which the percentage will be calculated.  
    percentage (float): The percentage to be calculated.  
  
    Returns:  
    float: The calculated percentage of the value.  
    """  
    return value * percentage / 100
```

OUTPUT :-

```
30.0
```

---

## Task Description #4 – Refactoring for Maintainability

Task: Use AI to break repetitive or long code into reusable functions.

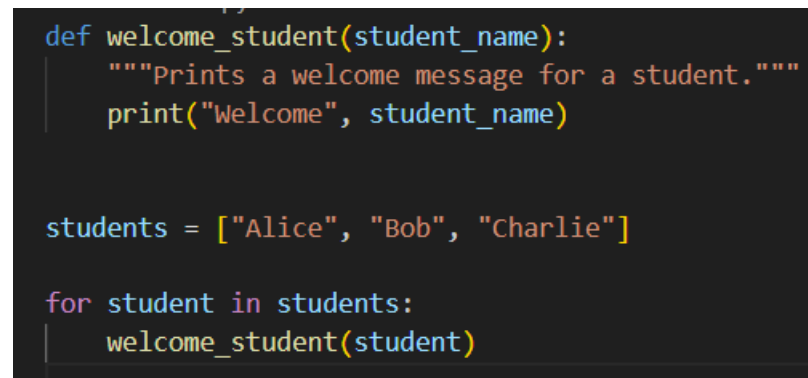
Sample Input Code:

```
students = ["Alice", "Bob", "Charlie"]
print("Welcome", students[0])
print("Welcome", students[1])
print("Welcome", students[2])
```

Expected Output:

- Modular code with reusable functions.
- 

SCREENSHOT :-

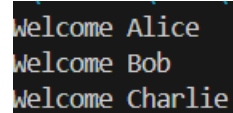


```
def welcome_student(student_name):
    """Prints a welcome message for a student."""
    print("Welcome", student_name)

students = ["Alice", "Bob", "Charlie"]

for student in students:
    welcome_student(student)
```

OUTPUT :



```
Welcome Alice
Welcome Bob
Welcome Charlie
```

## TASK -5

### Task Description #5 – Performance Optimization

Task: Use AI to make the code run faster.

Sample Input Code:

```
# Find squares of numbers
nums = [i for i in range(1,1000000)]
squares = []
for n in nums:
    squares.append(n**2)
print(len(squares))
```

Expected Output:

- Optimized code using list comprehensions or vectorized

operations.

```
Task Description #5 - Performance Optimization
Task: Use AI to make the code run faster.
Sample Input Code:
# Find squares of numbers
nums = [i for i in range(1,1000000)]
squares = []
for n in nums:
    squares.append(n**2)
print(len(squares))
Expected Output:
• Optimized code using list comprehensions or vectorized
operations.

# Optimized code using list comprehensions
nums = [i for i in range(1, 1000000)]
squares = [n**2 for n in nums]
print(len(squares))

# Optimized code using vectorized operations with NumPy
import numpy as np
nums = np.arange(1, 1000000)
squares = nums ** 2
print(len(squares))
```

OUTPUT :-

```
999999
```