

AI ASSISTED CODING

ASSIGNMENT-10.1

N. Madhuvani

2303A52117

Batch-40

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.

Code and Output:

```

File Edit Selection View Go Run Terminal Help < > Q AI-ASSIST
Al... 32.py Assignment-6.1.py Assignment-7.1.py exam1.py Assignment-8.1.py Assignment-10.1.py CHAT
SESSIONS
General greeting inquiry Completed in 10 mins. 1 mo ago
Assignment-3.1.py Assignment-3.2.py Assignment-6.1.py Assignment-7.1.py Assignment-8.1.py Assignment-10.1.py
Assignment-10.1.py
1 #Task-1
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 here
9 marks = [85, 90, 78, 92]
10 print("Average Score is ", calc_average(marks))
11 #check the Python program for logical mistakes that may cause incorrect output or runtime errors, and explain why they
12 # The function 'calc_average' has a typo in the return statement where it returns 'gyrage' instead of 'average'. This
13 # Telling me to correct the typo in the function.
14 def calc_average(marks):
15     total = 0
16     for m in marks:
17         total += m
18     average = total / len(marks)
19     return average # Corrected the typo here
20 marks = [85, 90, 78, 92]
21 print("Average Score is ", calc_average(marks)) # Added the missing parenthesis here

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER POSTMAN CONSOLE

Active code page: 65001

C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIST>C:/Users/nalla/AppData/Local/Programs/Python/Python313/python.exe C:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIST/Assignment-10.1.py

Average Score is 86.25

C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIST>

+ ... | X JavaSE... Python

Assignment-10.1.py

Describe what to build next:

In 21, Col 1 Spaces: 4 UTF-8 LF {} Python 3.13.7 13:25 ENG IN 23-02-2026

Launchpad 31°C Mostly cloudy

Analysis:

- The AI detects indentation mistakes, missing brackets, and spelling errors in variables
- It identifies logic flaws such as incorrect variable usage or faulty return values
- The AI corrects the code and produces a runnable program with explanations

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.

Code and Output:

```

Assignment-10.1.py
21 #Task -2
22 """def area_of_rect(l,B) : return L*B
23 print(area_of_rect(10,20))"""
24 #check the given Python code to strictly follow PEP 8 style guidelines, including proper naming conventions, spacing,
25 Tabbing | Edit | Test | Explain | Document
26 def area_of_rectangle(length, breadth):
27     """Calculate the area of a rectangle given its length and breadth."""
28     return length * breadth
29
30 print(area_of_rectangle(10, 20))

```

Analysis:

- The AI enforces proper naming conventions and spacing rules
- It restructures code layout for better readability and consistency
- The program remains functionally unchanged but becomes standard-compliant

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.

Code and Output:

```

29 #Task-3
30 '''def c(x,y):
31     return x*y/100
32 a=200
33 b=15
34 print(c(a,b))
35 #here for the above code don't change the logic but only correct the formatting and i
36 def calculate_percentage(x, y):
37     """Calculate the percentage of x with respect to y."""
38     return x * y / 100
39 a = 200
40 b = 15
41 print(calculate_percentage(a, b))
42
43
44

```

The screenshot shows a code editor interface with multiple tabs open. The active tab is 'Assignment-10.1.py'. The code in this tab has been refactored by AI. It uses descriptive variable names ('x', 'y') instead of 'a', 'b', and includes comments explaining the logic. The code is well-formatted with proper indentation. The sidebar on the left shows other files like 'Assignment-3.1.py', 'Assignment-3.2.py', etc. The bottom status bar shows the file path 'C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS>', line 'Ln 42, Col 1', and other system information.

Analysis:

- The AI replaces unclear variable names with descriptive ones
- It adds meaningful comments explaining each step
- The code is neatly formatted for easy understanding

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.

Code and Output:

```

42 #Task-4
43 '''Students = ["Alice", "Bob", "Charlie"]
44 print("Welcome", students[0])
45 print("Welcome", students[1])
46 print("Welcome", students[2])'''
47 #Make the above code modular and reusable functions to break repetitive
48 def welcome_students(students):
49     """Print a welcome message for each student in the list."""
50     for student in students:
51         print("Welcome", student)
52 students = ["Alice", "Bob", "Charlie"]
53 welcome_students(students)
54
55
56
57
58

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER ... + ... CHAT SESSIONS

/AI-ASSIS/Assignment-10.1.py
Welcome Alice
Welcome Bob
Welcome Charlie

C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS> []

Ln 53, Col 27 Spaces: 4 UTF-8 LF { } Python 3.13.7 (→) Go Live 13:46 23-02-2026

Analysis:

- The AI identifies repeated code blocks
- It converts them into reusable functions
- This makes the program easier to update and maintain

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.

Code and Output:

```

54     #Task-5
55     '''nums = [i for i in range(1,1000000)]
56     squares = []
57     for n in nums:
58         squares.append(n**2)
59     print(len(squares))...
60     #Optimize the above code to run faster
61     #It need to print how much time it takes to run the code
62     import time
63     start_time = time.time()
64     nums = range(1, 1000000) # Using range instead of list comprehension
65     squares = [n**2 for n in nums] # Using list comprehension for faster
66     print(len(squares))
67     end_time = time.time()
68     print(f"Time taken to run the code: {end_time - start_time:.4f} seconds")
69
70
71
72

```

The terminal output shows:

```

/Programs/Python/Python313/python.exe c:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIS/Assignment-10.1.py
999999
Time taken to run the code: 0.0844 seconds

```

Analysis:

- The AI removes unnecessary loops and intermediate lists
- It uses faster techniques like list comprehensions
- The code executes quicker with lower memory usage

Task Description #6 – Complexity Reduction

Task: Use AI to simplify overly complex logic.

Sample Input Code:

```
def grade(score):
```

```
    if score >= 90:
```

```
        return "A"
```

```
    else:
```

```
        if score >= 80:
```

```
            return "B"
```

```
        else:
```

```
            if score >= 70:
```

```
                return "C"
```

```
            else:
```

```
                if score >= 60:
```

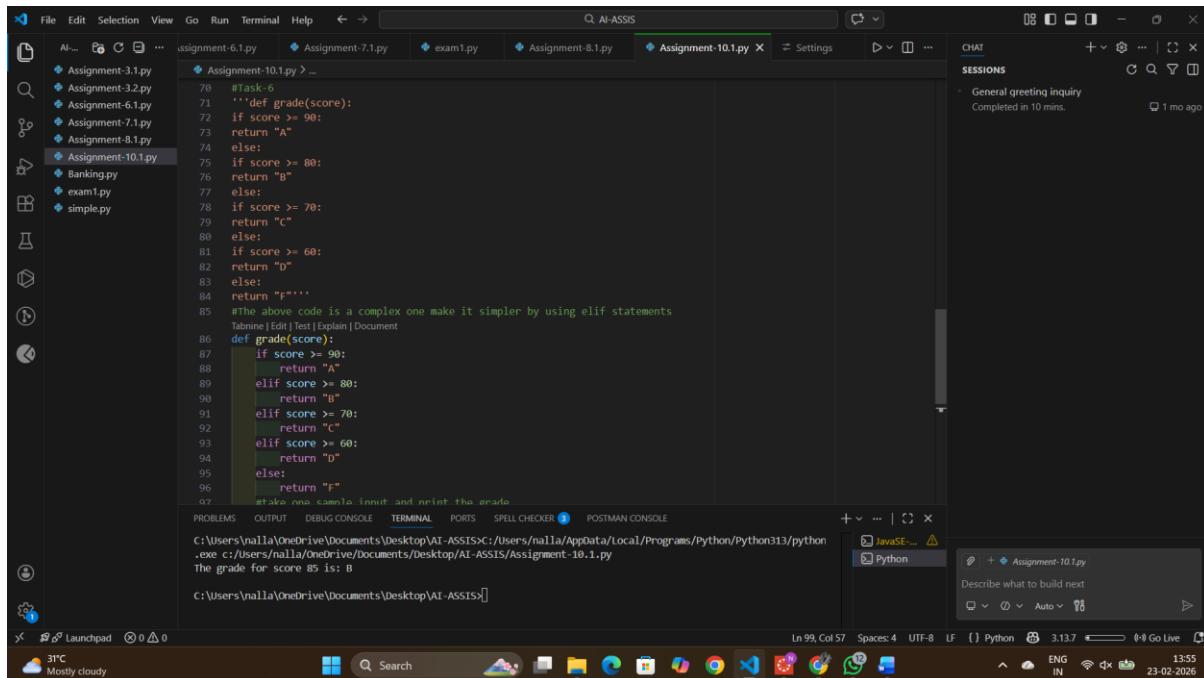
```
                    return "D"
```

```
else:  
    return "F"
```

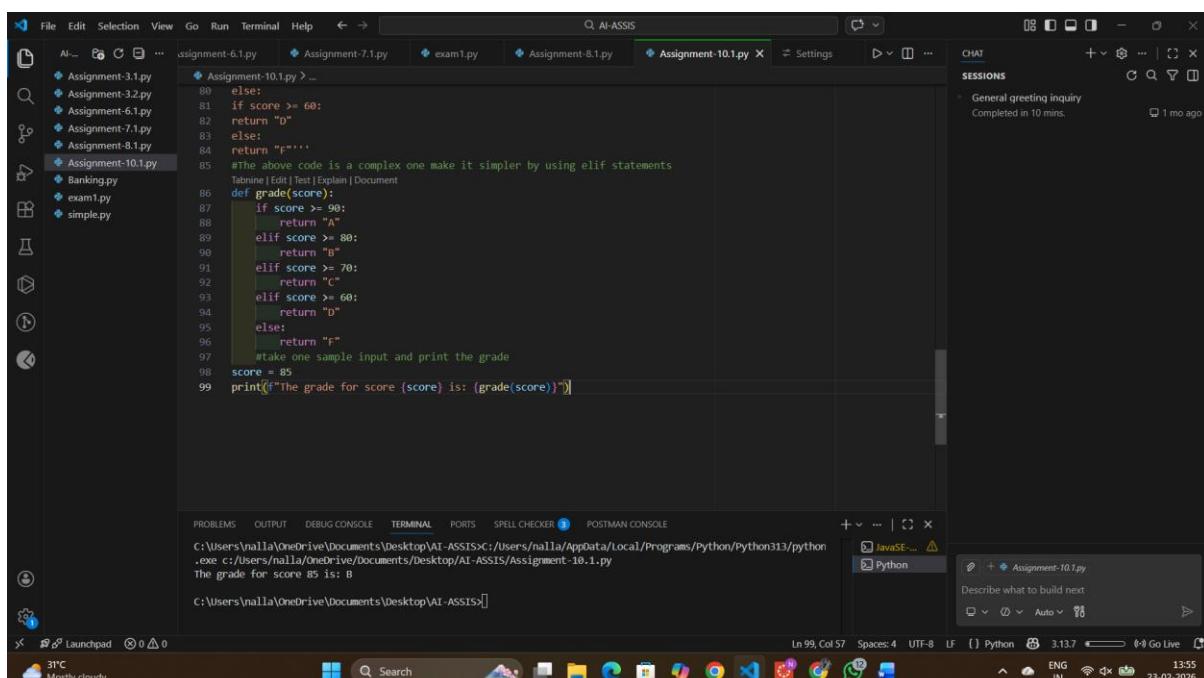
Expected Output:

- Cleaner logic using elif or dictionary mapping.

Code and Output:



```
File Edit Selection View Go Run Terminal Help ⏮ ⏯ 🔍 AI-ASSIS  
assignment-6.1.py Assignment-7.1.py exam1.py Assignment-8.1.py Assignment-10.1.py Settings ⏮ ⏯ ⏯  
Assignment-3.1.py Assignment-3.2.py Assignment-6.1.py Assignment-7.1.py Assignment-8.1.py Assignment-10.1.py > ...  
Assignment-10.1.py  
70     #def grade(score):  
71         if score >= 90:  
72             return "A"  
73         else:  
74             if score >= 80:  
75                 return "B"  
76             else:  
77                 if score >= 70:  
78                     return "C"  
79                 else:  
80                     if score >= 60:  
81                         return "D"  
82                     else:  
83                         return "F"  
84             return "*****"  
85 #The above code is a complex one make it simpler by using elif statements  
Tabnine | Edit | Test | Explain | Document  
def grade(score):  
    if score >= 90:  
        return "A"  
    elif score >= 80:  
        return "B"  
    elif score >= 70:  
        return "C"  
    elif score >= 60:  
        return "D"  
    else:  
        return "F"  
#take one sample input and print the grade  
score = 85  
print(f"The grade for score {score} is: {grade(score)}")  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER POSTMAN CONSOLE  
C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS>C:/Users/nalla/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIS/Assignment-10.1.py  
The grade for score 85 is: B  
C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS>  
Ln 99, Col 57 Spaces: 4 UTF-8 LF Python 3.13.7 13:55 ENG IN 23-02-2026
```



```
File Edit Selection View Go Run Terminal Help ⏮ ⏯ 🔍 AI-ASSIS  
assignment-6.1.py Assignment-7.1.py exam1.py Assignment-8.1.py Assignment-10.1.py Settings ⏮ ⏯ ⏯  
Assignment-3.1.py Assignment-3.2.py Assignment-6.1.py Assignment-7.1.py Assignment-8.1.py Assignment-10.1.py > ...  
Assignment-10.1.py  
80     else:  
81         if score >= 60:  
82             return "D"  
83         else:  
84             return "*****"  
85 #The above code is a complex one make it simpler by using elif statements  
Tabnine | Edit | Test | Explain | Document  
def grade(score):  
    if score >= 90:  
        return "A"  
    elif score >= 80:  
        return "B"  
    elif score >= 70:  
        return "C"  
    elif score >= 60:  
        return "D"  
    else:  
        return "F"  
#take one sample input and print the grade  
score = 85  
print(f"The grade for score {score} is: {grade(score)}")  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SPELL CHECKER POSTMAN CONSOLE  
C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS>C:/Users/nalla/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIS/Assignment-10.1.py  
The grade for score 85 is: B  
C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS>  
Ln 99, Col 57 Spaces: 4 UTF-8 LF Python 3.13.7 13:55 ENG IN 23-02-2026
```

Analysis:

- The AI simplifies nested conditional statements
- It replaces complex logic with cleaner structures like elif
- The result is shorter, clearer, and easier-to-debug code