

## ASSIGNMENT-10.1

Name: B.Bhargava Chary

H.No: 2303A51747

Batch-24

### 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:

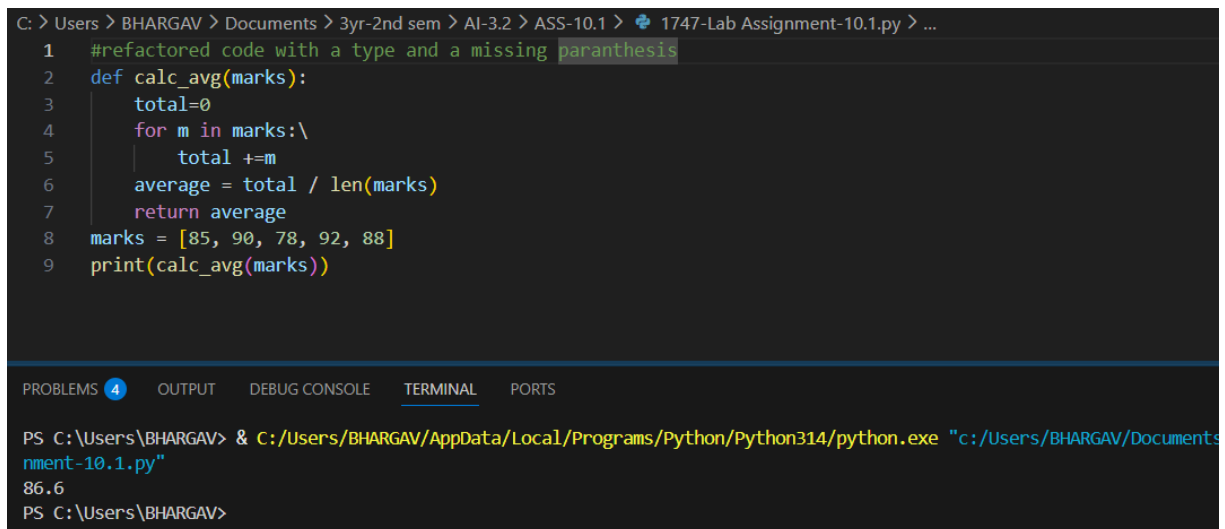
```
# Calculate average score of a student def
calc_average(marks):

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.



The screenshot shows a Python IDE with a file named '1747-Lab Assignment-10.1.py'. The code is as follows:

```
1 #refactored code with a type and a missing paranthesis
2 def calc_avg(marks):
3     total=0
4     for m in marks:\
5         total +=m
6     average = total / len(marks)
7     return average
8 marks = [85, 90, 78, 92, 88]
9 print(calc_avg(marks))
```

The IDE interface includes tabs for PROBLEMS (4), OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, showing the command prompt execution:

```
PS C:\Users\BHARGAV> & C:/Users/BHARGAV/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/BHARGAV/Documents/Assignment-10.1.py"
86.6
PS C:\Users\BHARGAV>
```

### 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.

```
13
14 def area_of_rect(L, B):
15     return L * B
16
17 print(area_of_rect(10, 20))
18
19 # refactored the above code and add documentation and type hints
20 def area_of_rect(length: float, breadth: float) -> float:
21     """
22     Calculate the area of a rectangle given its length and breadth.
23
24     Parameters:
25     length (float): The length of the rectangle.
26     breadth (float): The breadth of the rectangle.
27
28     Returns:
29     float: The area of the rectangle calculated as length multiplied by breadth.
30
31     Raises:
32     ValueError: If length or breadth is negative, as dimensions cannot be negative.
33     TypeError: If length or breadth is not a number (int or float).
34     """
35
36     if not isinstance(length, (int, float)) or not isinstance(breadth, (int, float)):
37         raise TypeError("Length and breadth must be numbers (int or float).")
38
39     if length < 0 or breadth < 0:
40         raise ValueError("Length and breadth must be non-negative.")
41
42     return length * breadth
43 print(area_of_rect(10, 20))
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS Python

PS C:\Users\BHARGAV> & C:/Users/BHARGAV/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/BHARGAV/Documents/3yr-2nd sem/AI-3.2/ASS-10.1/1747-Lab  
200  
200  
PS C:\Users\BHARGAV>

### 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.

```
C:\Users\BHARGAV> Documents\3yr-2nd sem\AI-3.2\ASS-10.1> 1747-Lab Assignment-10.1.py calculate_percentage

47 def c(x, y):
48     return x * y / 100
49
50 a = 200
51 b = 15
52 print(c(a, b))
53
54 # refactored the above code with descriptive variable names, inline comments, and clear formatting
55 def calculate_percentage(part: float, whole: float) -> float:
56     """
57     Calculate the percentage of a part relative to a whole.
58
59     Parameters:
60     part (float): The portion or part value.
61     whole (float): The total or whole value.
62
63     Returns:
64     float: The percentage calculated as (part / whole) * 100.
65
66     Raises:
67     ValueError: If the whole is zero, as division by zero is not allowed.
68     TypeError: If part or whole is not a number (int or float).
69     """
70
71     if not isinstance(part, (int, float)) or not isinstance(whole, (int, float)):
72         raise TypeError("Both part and whole must be numbers (int or float).")
73
74     if whole == 0:
75         raise ValueError("Whole cannot be zero to avoid division by zero.")
76
77     return (part / whole) * 100

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\BHARGAV> & C:\Users\BHARGAV\AppData\Local\Programs\Python\Python314\python.exe "c:/Users/BHARGAV/Documents/3yr-2nd sem/AI-3.2/ASS-10.1.py"
30.0
PS C:\Users\BHARGAV>
```

## 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.

```
83 |
84 | students = ["Alice", "Bob", "Charlie"]
85 | print("Welcome", students[0])
86 | print("Welcome", students[1])
87 | print("Welcome", students[2])
88 |
89 | # refactored code to reduce redundancy with reusable function
90 | def welcome_student(student: str) -> None:
91 |     """
92 |     Print a welcome message for a student.
93 |
94 |     Parameters:
95 |     student (str): The name of the student to welcome.
96 |
97 |     Returns:
98 |     None
99 |
100 |     values:
101 |     student: A string representing the name of the student.
102 |     type error: If the input is not a string, a TypeError will be raised.
103 |     """
104 |
105 |     if not isinstance(student, str):
106 |         raise TypeError("Student name must be a string.")
107 |
108 |     print("Welcome", student)
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + v [ ] [ ]

```
PS C:\Users\BHARGAV> & C:/Users/BHARGAV/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/BHARGAV/Documents/3yr-2nd sem/AI-3.2/ASS-10.1/1747-Lab Assignment-
Welcome Alice
Welcome Bob
Welcome Charlie
PS C:\Users\BHARGAV>
```

## 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.

```
111     nums = [i for i in range(1, 1000000)]
112     squares = []
113     for n in nums:
114         squares.append(n**2)
115     print(len(squares))
116
117     # refactored the above code to reduce time complexity
118     nums = [i for i in range(1, 1000000)]
119     squares = [n**2 for n in nums]
120     print(len(squares))
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS Python +

PS C:\Users\BHARGAV> C:/Users/BHARGAV/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/BHARGAV/Documents/3yr-2nd sem/AI-3.2/ASS-10.1/1747-Lab Assignment-10.1.py"
999999
999999
PS C:\Users\BHARGAV>

```
C:\Users\BHARGAV> Documents\3yr-2nd sem> AI-3.2> ASS-10.1> 1747-Lab Assignment-10.1.py > ...
122     import time
123
124     time1 = time.time()
125     nums = [i for i in range(1, 1000000)]
126     squares = []
127     for n in nums:
128         squares.append(n**2)
129     #print(len(squares))
130     time2 = time.time()
131     print("Time taken: ", time2 - time1)
132
133     # refactor the above code to reduce time complexity
134     time3 = time.time()
135     nums = [i for i in range(1, 1000000)]
136     squares = [n**2 for n in nums]
137     #print(len(squares))
138     time4 = time.time()
139     print("Time taken:", time4 - time3)
140
141     time5 = time.time()
142     #print(len([n**2 for n in range(1, 1000000)]))
143     time6 = time.time()
144     print("Time taken:", time6 - time5)
```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS Python +

PS C:\Users\BHARGAV> & C:/Users/BHARGAV/AppData/Local/Programs/Python/Python314/python.exe "c:/Users/BHARGAV/Documents/3yr-2nd sem/AI-3.2/ASS-10.1/1747-Lab Assignment-10.1.py"
Time taken: 0.11278772354125977
Time taken: 0.09960126876831055
Time taken: 2.384185791015625e-07
PS C:\Users\BHARGAV>

## 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.

```
C:\Users\BHARGAV> Documents > 3yr-2nd sem > AI-3.2 > ASS-10.1 > 1747-Lab Assignment-10.1.py > ...
148 def grade(score):
149     if score >= 90:
150         return "A"
151     else:
152         if score >= 80:
153             return "B"
154         else:
155             if score >= 70:
156                 return "C"
157             else:
158                 if score >= 60:
159                     return "D"
160                 else:
161                     return "F"
162 # refactored code to Cleaner logic using elif or dictionary mapping.
163 def grade(score: int) -> str:
164     """
165     Return the grade based on the score.
166
167     Parameters:
168     score (int): Student score
169
170     Returns:
171     str: Grade (A, B, C, D, or F)
172     """
173     if score >= 90:
174         return "A"
175     elif score >= 80:
176         return "B"
177     elif score >= 70:
178         return "C"
179     elif score >= 60:
180         return "D"
181     else:
182         return "F"
183 print(grade(95))
184 def grade(score: int) -> str:
185     """
186     Return the grade based on the score using dictionary mapping.
187     """
188     grade_map = {
189         90: "A",
190         80: "B",
191         70: "C",
192         60: "D",
193         0: "F"
194     }
195     for cutoff, letter in grade_map.items():
196         if score >= cutoff:
197             return letter
198 print(grade(85))

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\BHARGAV> & C:\Users\BHARGAV\AppData\Local\Programs\Python\Python314\python.exe "c:/Users/BHARGAV/Documents/3yr-2nd sem/AI-3.2/ASS-10.1/1747-Lab Assignment-10.1.py"
A
B
PS C:\Users\BHARGAV>
```