

ASSIGNMENT-10.1

Name: Sandeep

Roll No: 2303A51665

Batch-23

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.

```
1  # refactored code with a typo and a missing parenthesis
2
3 def calc_average(marks):
4     total = 0
5     for m in marks:
6         total += m
7     average = total / len(marks)
8     return average # Fixed typo : 'avrge' to 'average' and added missing parenthesis
9
10 marks = [85, 90, 78, 92]
11 print("Average Score is ", calc_average(marks))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\AI> & "C:/Users/sande/miniconda3/bin/python.exe" d:/AI/lab10.py
Average Score is 86.25
PS D:\AI>

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.

```
lab10.py > ...
1 def area_of_rect(L, B):
2     return L * B
3
4
5 print(area_of_rect(10, 20))
6
7
8 # refactored the above code and add documentation and type hints
9 def area_of_rect(length: float, breadth: float) -> float:
10    """
11        Calculate the area of a rectangle given its length and breadth.
12
13    Parameters:
14        length (float): The length of the rectangle.
15        breadth (float): The breadth of the rectangle.
16
17    Returns:
18        float: The area of the rectangle calculated as length multiplied by breadth.
19
20    Raises:
21        ValueError: If length or breadth is negative, as dimensions cannot be negative.
22        TypeError: If length or breadth is not a number (int or float).
23    """
24
25    if not isinstance(length, (int, float)) or not isinstance(breadth, (int, float)):
26        raise TypeError("Length and breadth must be numbers (int or float).")
27
28    if length < 0 or breadth < 0:
29        raise ValueError("Length and breadth must be non-negative.")
30
31    return length * breadth
...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\AI> & "C:/Users/sande/miniconda3/sr univ/python.exe" d:/AI/lab10.py
200
200
PS D:\AI>
```

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.

```

36 def c(x, y):
37     return x * y / 100
38
39 a = 200
40 b = 15
41 print(c(a, b))
42
43 # refactored the above code with descriptive variable names, inline comments, and clear formatting
44 def calculate_percentage(part: float, whole: float) -> float:
45     """
46     Calculate the percentage of a part relative to a whole.
47
48     Parameters:
49     part (float): The portion or part value.
50     whole (float): The total or whole value.
51
52     Returns:
53     float: The percentage calculated as (part / whole) * 100.
54
55     Raises:
56     ValueError: If the whole is zero, as division by zero is not allowed.
57     TypeError: If part or whole is not a number (int or float).
58     """
59
60     if not isinstance(part, (int, float)) or not isinstance(whole, (int, float)):
61         raise TypeError("Both part and whole must be numbers (int or float).")
62
63     if whole == 0:
64         raise ValueError("Whole cannot be zero to avoid division by zero.")
65
66     return (part / whole) * 100

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS D:\AI> & "c:/Users/sande/miniconda3/python.exe" d:/AI/lab10.py
30.0
PS D:\AI>

```

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.

```

68 students = ["Alice", "Bob", "Charlie"]
69 print("Welcome", students[0])
70 print("Welcome", students[1])
71 print("Welcome", students[2])
72
73 # refactored code to reduce redundancy with reusable function
74 def welcome_student(student: str) -> None:
75     """
76     Print a welcome message for a student.
77
78     Parameters:
79     student (str): The name of the student to welcome.
80
81     Returns:
82     None
83
84     values:
85     student: A string representing the name of the student.
86     type error: If the input is not a string, a TypeError will be raised.
87     """
88
89     if not isinstance(student, str):
90         raise TypeError("Student name must be a string.")
91
92     print("Welcome", student)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS D:\AI> & "C:/Users/sande/miniconda3/bin/univ/python.exe" d:/AI/lab10.py
Welcome Alice
Welcome Bob
Welcome Charlie
PS D:\AI>

```

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.

```

93     nums = [i for i in range(1, 1000000)]
94     squares = []
95     for n in nums:
96         |     squares.append(n**2)
97     print(len(squares))
98
99
100    # refactored the above code to reduce time complexity
101    nums = [i for i in range(1, 1000000)]
102    squares = [n**2 for n in nums]
103    print(len(squares))

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS D:\AI> & "C:/Users/sande/miniconda3/sr univ/python.exe" d:/AI/lab10.py
999999
999999
PS D:\AI>

```

```

105   import time
106
107   time1 = time.time()
108   nums = [i for i in range(1, 1000000)]
109   squares = []
110   for n in nums:
111       |     squares.append(n**2)
112   #print(len(squares))
113   time2 = time.time()
114   print("Time taken: ", time2 - time1)
115
116   # refactor the above code to reduce time complexity
117   time3 = time.time()
118   nums = [i for i in range(1, 1000000)]
119   squares = [n**2 for n in nums]
120   #print(len(squares))
121   time4 = time.time()
122   print("Time taken:", time4 - time3)
123
124   time5 = time.time()
125   #print(len([n**2 for n in range(1, 1000000)]))
126   time6 = time.time()
127   print("Time taken:", time6 - time5)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

PS D:\AI> & "C:/Users/sande/miniconda3/sr univ/python.exe" d:/AI/lab10.py
Time taken: 0.22025132179260254
Time taken: 0.20869088172912598
Time taken: 7.152557373046875e-07
PS D:\AI>

```

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.

```

156     if score >= 90:
157         return "A"
158     elif score >= 80:
159         return "B"
160     elif score >= 70:
161         return "C"
162     elif score >= 60:
163         return "D"
164     else:
165         return "F"
166
167 print(grade(95))
168 def grade(score: int) -> str:
169     """
170     Return the grade based on the score using dictionary mapping.
171     """
172     grade_map = {
173         90: "A",
174         80: "B",
175         70: "C",
176         60: "D",
177         0: "F"
178     }
179
180     for cutoff, letter in grade_map.items():
181         if score >= cutoff:
182             return letter
183     print(grade(85))

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\AI> & "C:/Users/sande/miniconda3/python.exe" d:/AI/lab10.py

A
B
PS D:\AI> []