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.

```python
# refactored code with a typo and a missing paranthesis

def calc_average(marks):
    total = 0
    for m in marks:
        total += m
    average = total / len(marks)
    return average  # Fixed typo : 'avrge' to 'average' and added missing paranthesis

marks = [85, 90, 78, 92]
print("Average Score is ", calc_average(marks))
```

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

Task Description #2 – PEP 8 Compliance

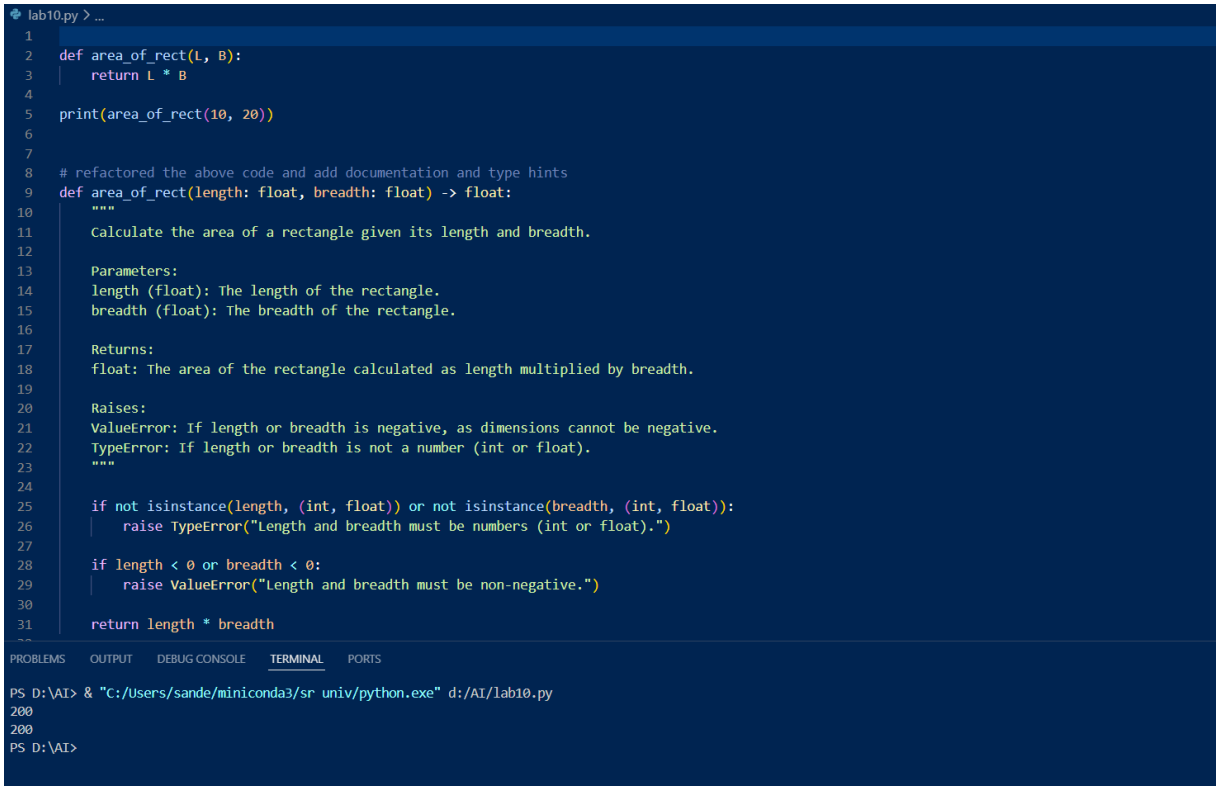
Task: Use AI to refactor Python code to follow PEP 8 style guidelines.

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.



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.

```python
def c(x, y):
    return x * y / 100

a = 200
b = 15
print(c(a, b))

# refactored the above code with descriptive variable names, inline comments, and clear formatting
def calculate_percentage(part: float, whole: float) -> float:
    """
    Calculate the percentage of a part relative to a whole.

    Parameters:
    part (float): The portion or part value.
    whole (float): The total or whole value.

    Returns:
    float: The percentage calculated as (part / whole) * 100.

    Raises:
    ValueError: If the whole is zero, as division by zero is not allowed.
    TypeError: If part or whole is not a number (int or float).
    """

    if not isinstance(part, (int, float)) or not isinstance(whole, (int, float)):
        raise TypeError("Both part and whole must be numbers (int or float).")

    if whole == 0:
        raise ValueError("Whole cannot be zero to avoid division by zero.")

    return (part / whole) * 100
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\AI> & "C:/Users/sande/miniconda3/sr univ/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.

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

# refactored code to reduce redundancy with reusable function
def welcome_student(student: str) -> None:
    """
    Print a welcome message for a student.

    Parameters:
    student (str): The name of the student to welcome.

    Returns:
    None

    values:
    student: A string representing the name of the student.
    type error: If the input is not a string, a TypeError will be raised.
    """

    if not isinstance(student, str):
        raise TypeError("Student name must be a string.")

    print("Welcome", student)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\AI> & "C:/Users/sande/miniconda3/sr 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.

```python

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

# refactored the above code to reduce time complexity
nums = [i for i in range(1, 1000000)]
squares = [n**2 for n in nums]
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>
```

```python
import time

time1 = time.time()
nums = [i for i in range(1, 1000000)]
squares = []
for n in nums:
    squares.append(n**2)
#print(len(squares))
time2 = time.time()
print("Time taken: ", time2 - time1)

# refactor the above code to reduce time complexity
time3 = time.time()
nums = [i for i in range(1, 1000000)]
squares = [n**2 for n in nums]
#print(len(squares))
time4 = time.time()
print("Time taken:", time4 - time3)

time5 = time.time()
#print(len([n**2 for n in range(1, 1000000)]))
time6 = time.time()
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.