ASSIGNMENT-10.1

Name: V.Sravani

Ht .No : 2303A51923

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

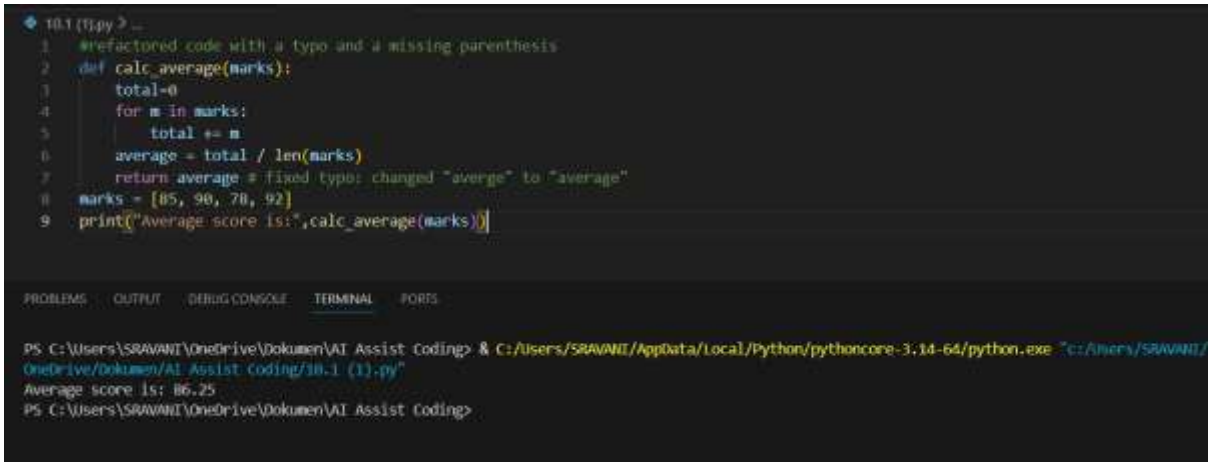
marks = [85, 90, 78, 92]

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

Expected Output:

• Corrected and runnable Python code with explanations of the

fixes.



Task Description #2 – PEP 8 Compliance

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

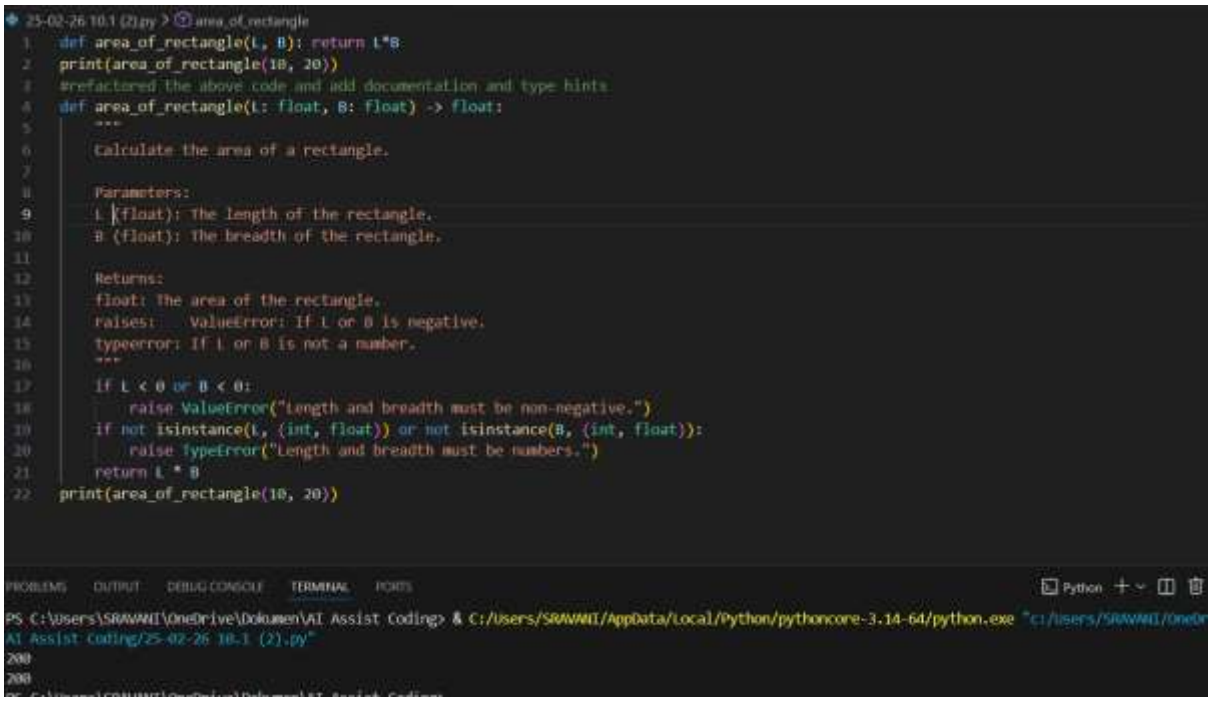
Sample Input Code:

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

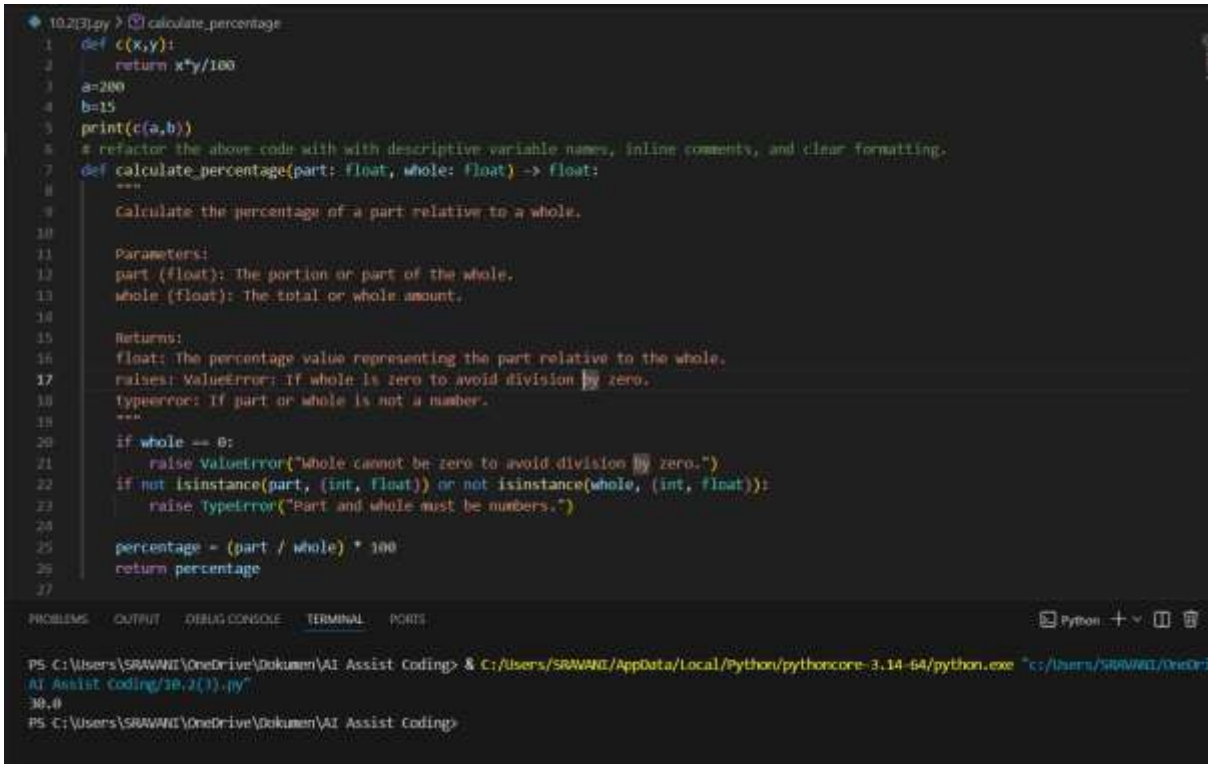
a=200

b=15

print(c(a,b))

Expected Output:

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



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"]

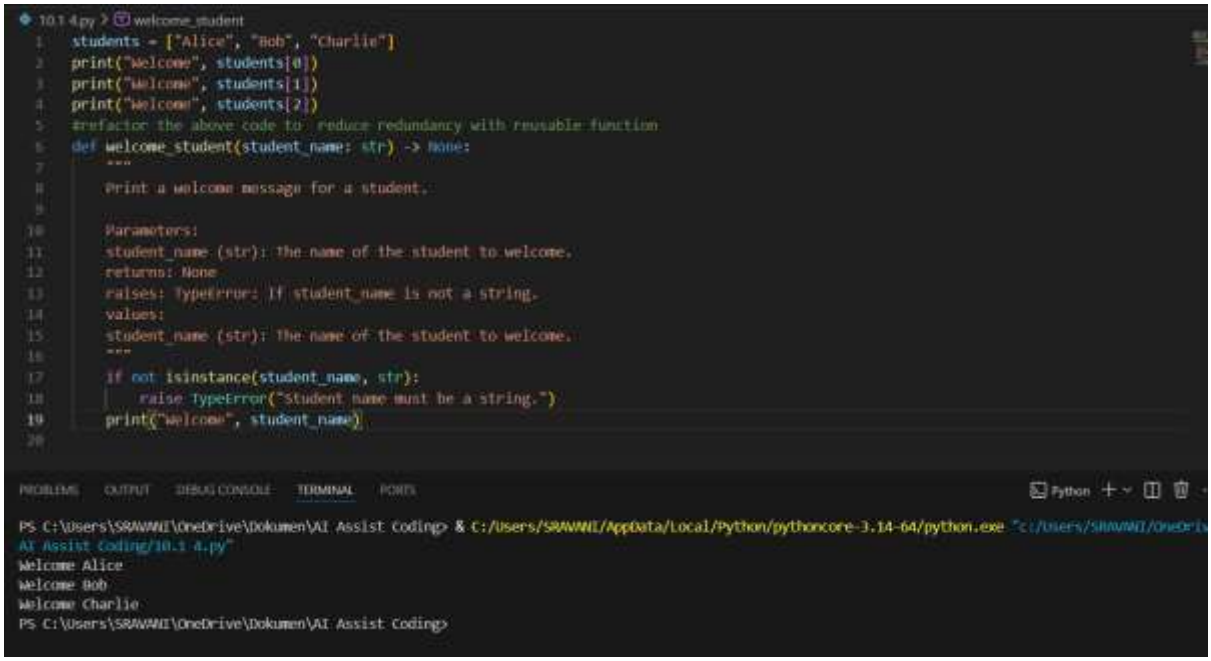
print("Welcome", students[0])

print("Welcome", students[1])

print("Welcome", students[2])

Expected Output:

• Modular code with reusable functions.



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.

10.1 (5).py > ...

```python
nums = [i for i in range(1,1000000)]
squares = []
for n in nums:
  squares.append(n**2)
print(len(squares))
#refactor 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 C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3.14
AI Assist Coding/10.1 (5).py"
999999
999999
PS C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding>
```

10.1 (5).py > ...

```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 C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding> & C:/Users/SRAVANI/AppData/Local/Py
OneDrive/Dokumen/AI Assist Coding/10.1 (5).py"
Time taken:  0.1295607089996338
Time taken: 0.11092209815979004
Time taken: 2.384185791015625e-07
PS C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding>
```

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.

```python
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"
#refactor code to cleaner logic using elif or dictionary maping
def grade(score: float) -> str:
    """
    return the score based on the score
    parameters:
    score (float): The score to evaluate.
    returns:
    str: grade(A, B, C, D, F) based on the score."""
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 70:
        return "C"
    elif score >= 60:
        return "D"
    else:
        return "F"
print(grade(85))
```

```python
def grade(score: float) -> str:
    """
    return the grade based on score using dictionary mapping"""
    grade_map = {
        90: "A",
        80: "B",
        70: "C",
        60: "D"
    }
    for threshold, grade in grade_map.items():
        if score >= threshold:
            return grade
    return "F"
print(grade(95))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding> & C:/Users/SRAVANI/AppData/Local/Python/pythoncore-3.14-64/python.exe "
AI Assist Coding/10.1 6.py"
B
A
PS C:\Users\SRAVANI\OneDrive\Dokumen\AI Assist Coding>
```