Assignment-10.1

2303A51663

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

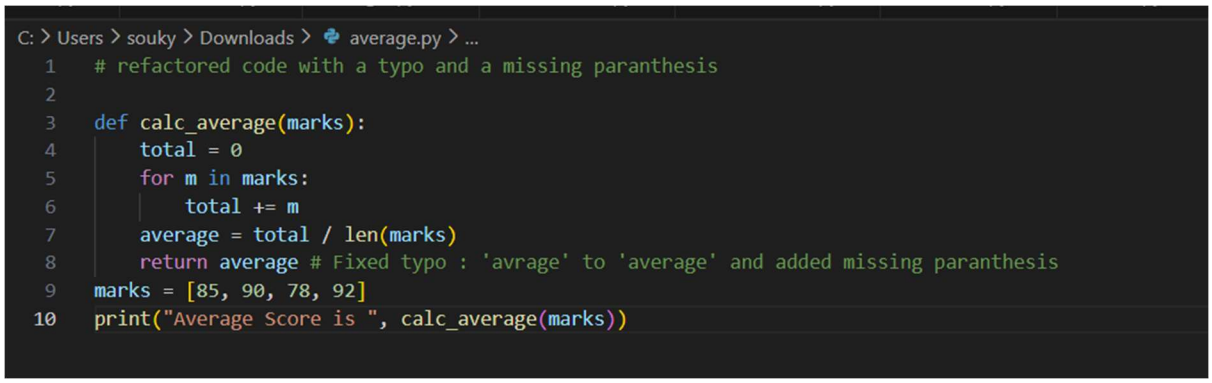
marks = [85, 90, 78, 92]

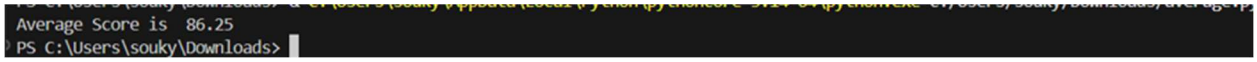
print("Average Score is ", calc_average(marks)

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.

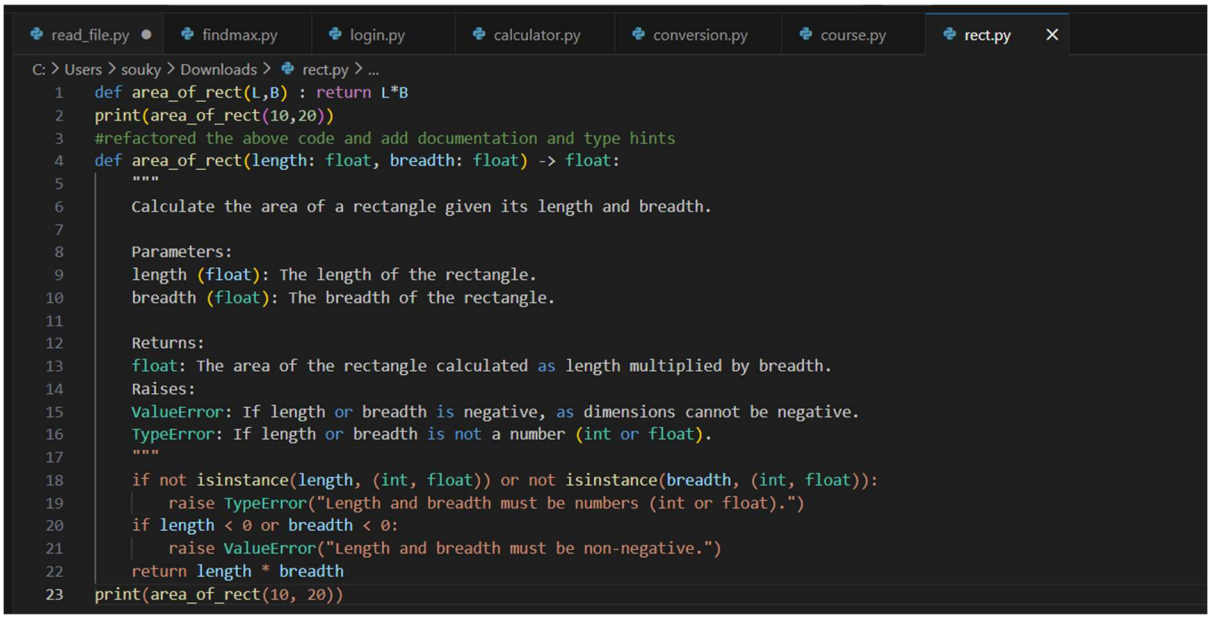
Sample Input Code:

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

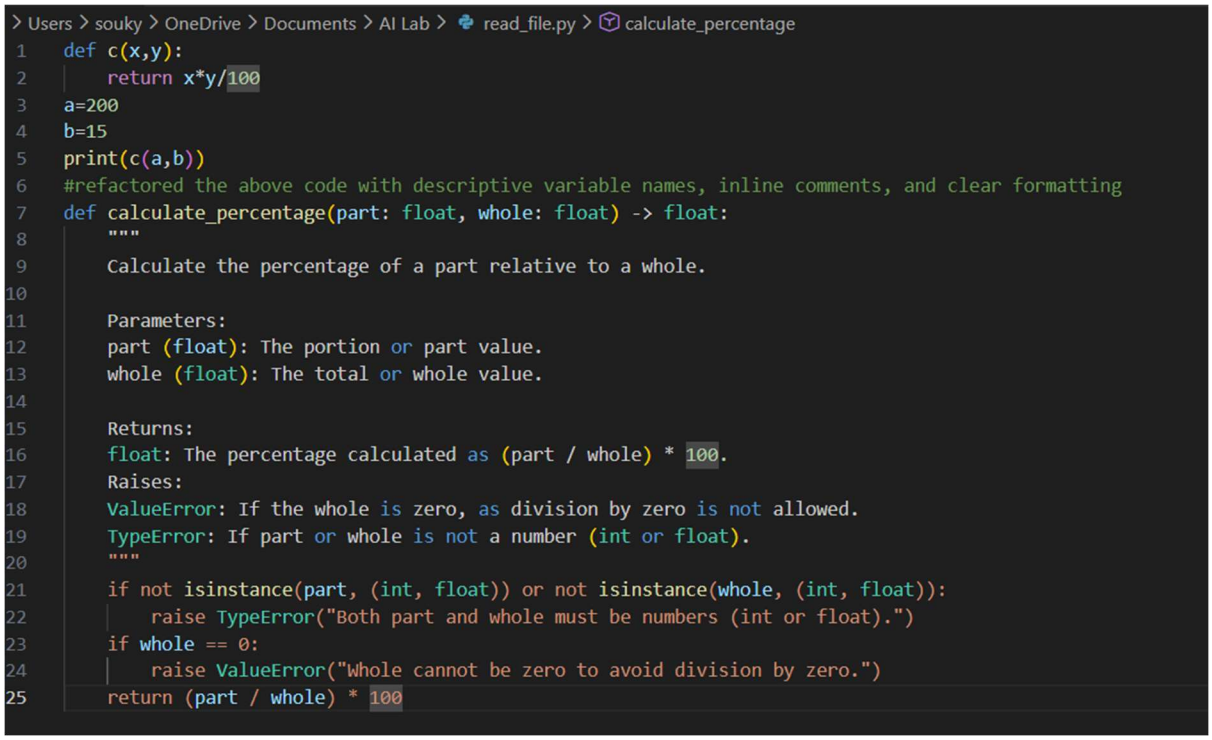
b=15

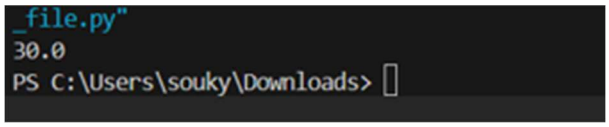
print(c(a,b))

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"]

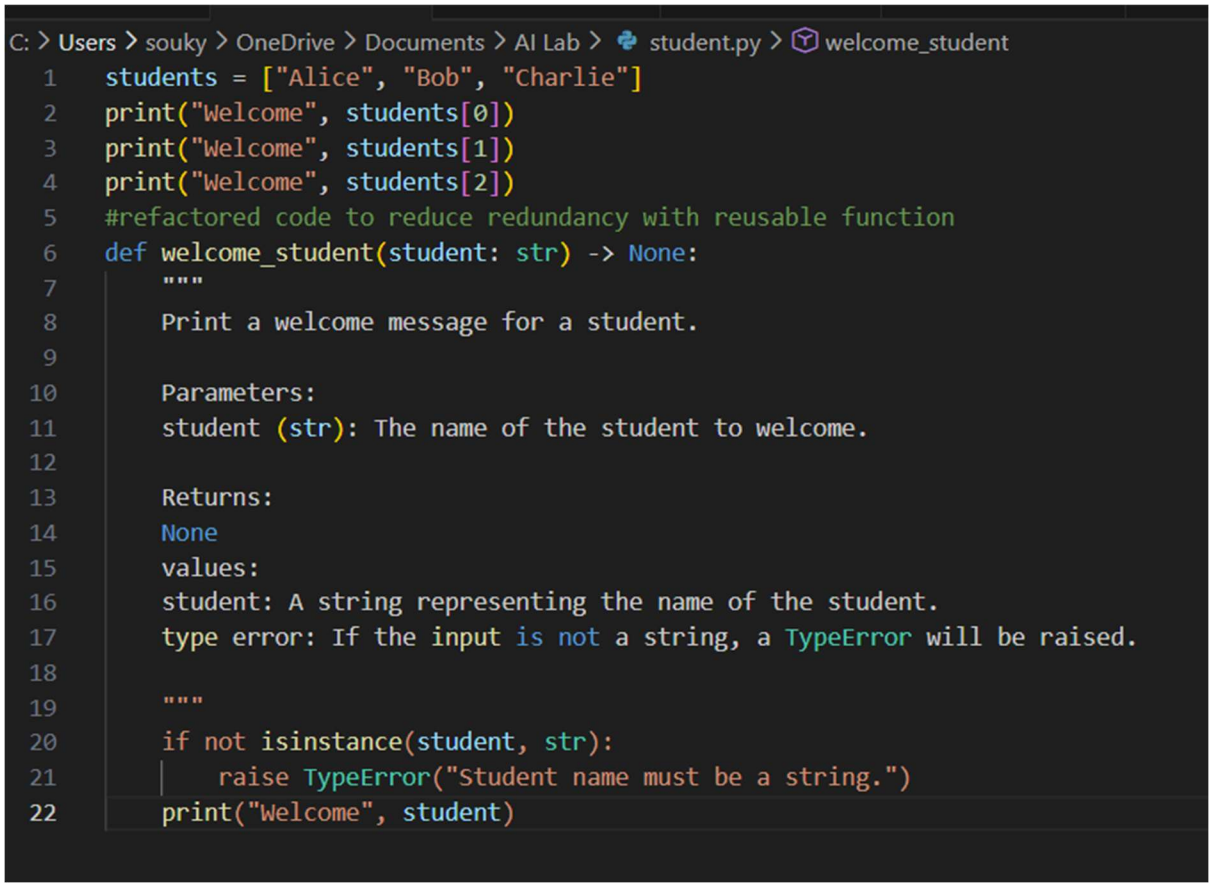
print("Welcome", students[0])

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:

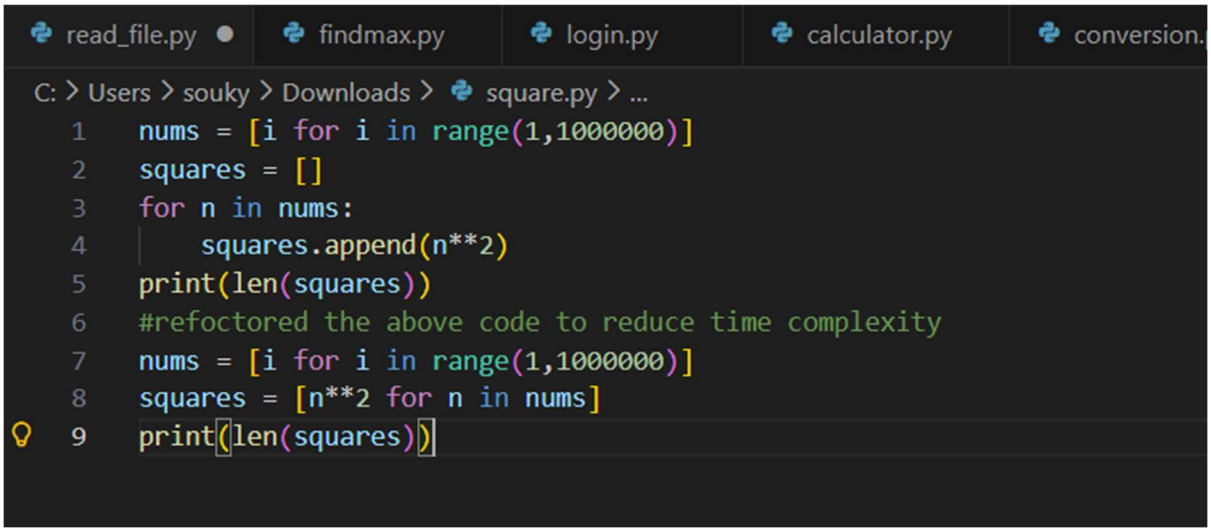
squares.append(n**2)

print(len(squares))

Expected Output:

• Optimized code using list comprehensions or vectorized

operations.



C: > Users > souky > Downloads > square.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)
```

```
Time taken: 0.36050939559936523
Time taken: 0.3215620517730713
Time taken: 9.5367431640625e-07
PS C:\Users\souky\Downloads>
```