

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.

```
C: > Users > souky > Downloads > average.py > ...
1  # refactored code with a typo and a missing paranthesis
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 : 'avrage' to 'average' and added missing paranthesis
9  marks = [85, 90, 78, 92]
10 print("Average Score is ", calc_average(marks))
```

```
Average Score is 86.25
PS C:\Users\souky\Downloads>
```

Task Description #2 – PEP 8 Compliance

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

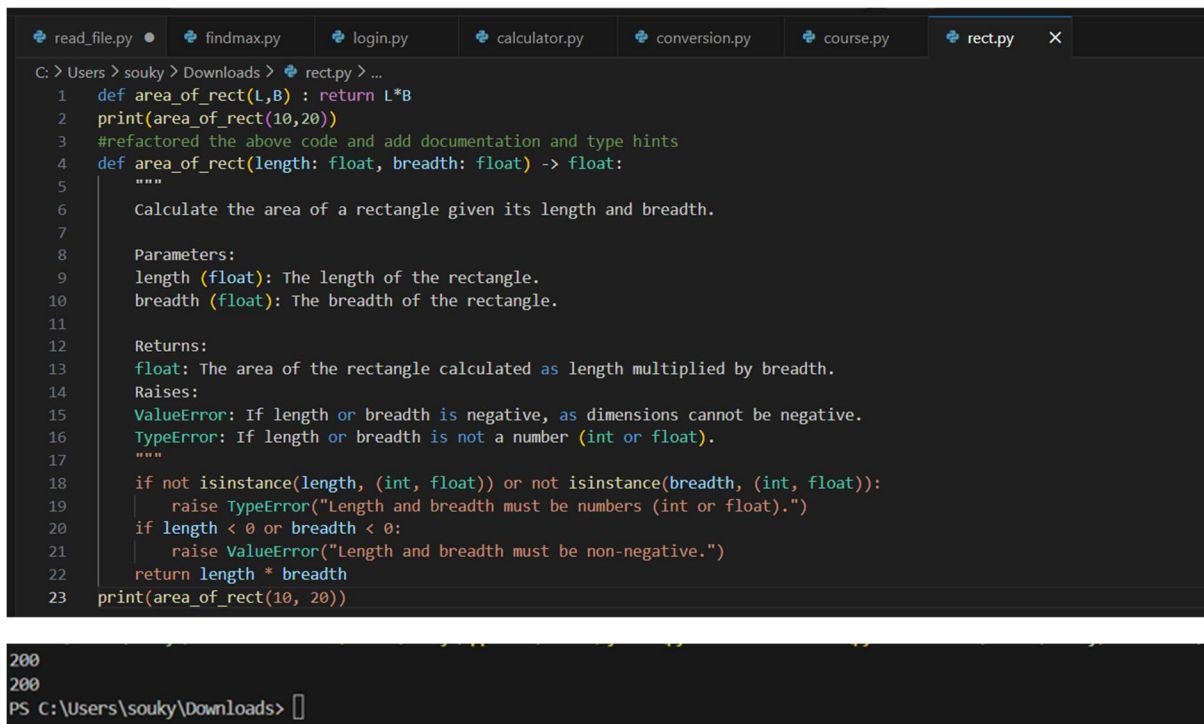
Sample Input Code:

```
def area_of_rect(L,B) : return L*B

print(area_of_rect(10,20))
```

Expected Output:

- Well-formatted PEP 8-compliant Python code.



```
read_file.py • findmax.py login.py calculator.py conversion.py course.py rect.py X
C: > Users > souky > Downloads > rect.py > ...
1  def area_of_rect(L,B) : return L*B
2  print(area_of_rect(10,20))
3  #refactored the above code and add documentation and type hints
4  def area_of_rect(length: float, breadth: float) -> float:
5      """
6      Calculate the area of a rectangle given its length and breadth.
7
8      Parameters:
9      length (float): The length of the rectangle.
10     breadth (float): The breadth of the rectangle.
11
12     Returns:
13     float: The area of the rectangle calculated as length multiplied by breadth.
14     Raises:
15     ValueError: If length or breadth is negative, as dimensions cannot be negative.
16     TypeError: If length or breadth is not a number (int or float).
17     """
18     if not isinstance(length, (int, float)) or not isinstance(breadth, (int, float)):
19         raise TypeError("Length and breadth must be numbers (int or float).")
20     if length < 0 or breadth < 0:
21         raise ValueError("Length and breadth must be non-negative.")
22     return length * breadth
23 print(area_of_rect(10, 20))

200
200
PS C:\Users\souky\Downloads>
```

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.

```
> Users > souky > OneDrive > Documents > AI Lab > read_file.py > calculate_percentage
1  def c(x,y):
2      return x*y/100
3
4  a=200
5  b=15
6  print(c(a,b))
7  #refactored the above code with descriptive variable names, inline comments, and clear formatting
8  def calculate_percentage(part: float, whole: float) -> float:
9      """
10         Calculate the percentage of a part relative to a whole.
11
12         Parameters:
13         part (float): The portion or part value.
14         whole (float): The total or whole value.
15
16         Returns:
17         float: The percentage calculated as (part / whole) * 100.
18
19         Raises:
20         ValueError: If the whole is zero, as division by zero is not allowed.
21         TypeError: If part or whole is not a number (int or float).
22         """
23         if not isinstance(part, (int, float)) or not isinstance(whole, (int, float)):
24             raise TypeError("Both part and whole must be numbers (int or float).")
25         if whole == 0:
26             raise ValueError("whole cannot be zero to avoid division by zero.")
27         return (part / whole) * 100
```

```
_file.py"
30.0
PS C:\Users\souky\Downloads> 
```

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.

```
C: > Users > souky > OneDrive > Documents > AI Lab > student.py > welcome_student
1  students = ["Alice", "Bob", "Charlie"]
2  print("Welcome", students[0])
3  print("Welcome", students[1])
4  print("Welcome", students[2])
5  #refactored code to reduce redundancy with reusable function
6  def welcome_student(student: str) -> None:
7      """
8      Print a welcome message for a student.
9
10     Parameters:
11     student (str): The name of the student to welcome.
12
13     Returns:
14     None
15     values:
16     student: A string representing the name of the student.
17     type error: If the input is not a string, a TypeError will be raised.
18
19     """
20     if not isinstance(student, str):
21         raise TypeError("Student name must be a string.")
22     print("Welcome", student)
```

```
ent.py"
Welcome Alice
Welcome Bob
Welcome Charlie
PS C:\Users\souky\Downloads>
```

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.

```
read_file.py • findmax.py login.py calculator.py conversion.  
C: > Users > souky > Downloads > square.py > ...  
1  nums = [i for i in range(1,1000000)]  
2  squares = []  
3  for n in nums:  
4      squares.append(n**2)  
5  print(len(squares))  
6  #refactored the above code to reduce time complexity  
7  nums = [i for i in range(1,1000000)]  
8  squares = [n**2 for n in nums]  
9  print(len(squares))
```

```
999999  
999999  
PS C:\Users\souky\Downloads >
```

```
id_me.py • findmax.py • login.py • calculator.py • conversi
C: > Users > souky > Downloads > square.py > ...
1 import time
2 time1 = time.time()
3 nums = [i for i in range(1,1000000)]
4 squares = []
5 for n in nums:
6     squares.append(n**2)
7 #print(len(squares))
8 time2 = time.time()
9 print("Time taken:", time2 - time1)
10 # refactor the above code to reduce time complexity
11 time3 = time.time()
12 nums = [i for i in range(1,1000000)]
13 squares = [n**2 for n in nums]
14 #print(len(squares))
15 time4=time.time()
16 print("Time taken:", time4 - time3)
17 time5 = time.time()
18 #print(len([n**2 for n in range(1,1000000)]))
19 time6 = time.time()
20 print("Time taken:", time6 - time5)
```

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

