Assignment-10.1

2303A51272

A.Srivani

Batch-23

Task - 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.

aver C:\Users\HP\Desktop\AI-Lab\login_function.py

```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: 'avrage' to 'average' and added missing paranthesis

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

```
PS C:\Users\HP\Desktop\AI-Lab> & C:\Users\HP\AppData\Local\Programs\Python\Python313\python.exe c:/Users/HP/Desktop/AI-Lab/average.py
Average Score is  86.25
PS C:\Users\HP\Desktop\AI-Lab>
```

Task - 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.

```python
def area_of_rect(_B) : return L*B
print(area_of_rect(10,20))

#refactored the above code and add documentation and type hints
def area_of_rect(length: float, breadth: float) -> float:
    """
    Calculate the area of a rectangle given its length and breadth.

    Parameters:
    length (float): The length of the rectangle.
    breadth (float): The breadth of the rectangle.

    Returns:
    float: The area of the rectangle calculated as length multiplied by breadth.

    Raises:
    ValueError: If length or breadth is negative, as dimensions cannot be negative.
    TypeError: If length or breadth is not a number (int or float).
    """
    if not isinstance(length, (int, float)) or not isinstance(breadth, (int, float)):
        raise TypeError("Length and breadth must be numbers (int or float).")
    if length < 0 or breadth < 0:
        raise ValueError("Length and breadth must be non-negative.")
    return length * breadth

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

```
PS C:\Users\HP\Desktop\AI-Lab> & C:\Users\HP\AppData\Local\Programs\Python\Python313\python.exe c:/Users/HP/Desktop/AI-Lab/rect.py
200
PS C:\Users\HP\Desktop\AI-Lab>
```

Task - 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.

read_file.py > calculate_percentage

```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
```

```
PS C:\Users\HP\Desktop\AI-Lab> & C:\Users\HP\AppData\Local\Programs\Python\Python313\python.exe c:/Users/HP/Desktop/AI-Lab/read_file.py
30.0
PS C:\Users\HP\Desktop\AI-Lab>
```

Task - 4 – Refactoring for Maintainability

Task: Use AI to break repetitive or long code into reusable functions.

Sample Input Code:

students = ["Alice", "Bob", "Charlie"]

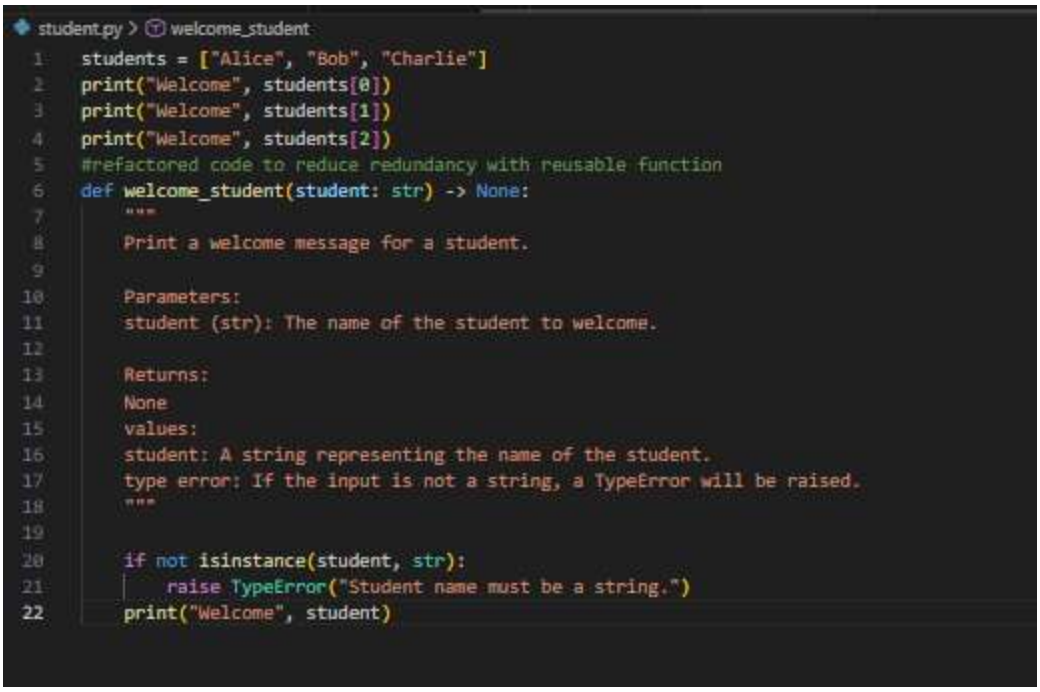
print("Welcome", students[0])

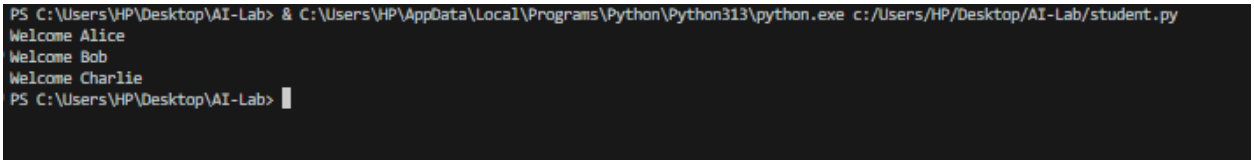
print("Welcome", students[1])

print("Welcome", students[2])

Expected Output:

• Modular code with reusable functions.





Task - 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.

square.py > _ Close (Ctrl+F4)

```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))
```

```
PS C:\Users\HP\Desktop\AI-Lab> & C:\Users\HP\AppData\Local\Programs\Python\Python313\python.exe c:/Users/HP/Desktop/AI-Lab/square.py
999999
999999
PS C:\Users\HP\Desktop\AI-Lab>
```

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)
```

```
PS C:\Users\HP\Desktop\AI-Lab> & C:\Users\HP\AppData\Local\Programs\Python\Python313\python.exe c:/Users/HP/Desktop/AI-Lab/square.py
Time taken: 0.3363194465637207
Time taken: 0.22841691970825195
Time taken: 7.152557373046875e-07
PS C:\Users\HP\Desktop\AI-Lab>
```

Task - 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.

grade.py > grade

```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"
#refactored code to Cleaner logic using elif or dictionary mapping.
def grade(score: int) -> str:
    """
    Return the grade based on the score.
    Parameters:
    score (int): Student score
    Returns:
    str: Grade (A, B, C, D, or F)
    """
    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(95))
def grade(score: int) -> str:
    """
    Return the grade based on the score using dictionary mapping.
    """
    grade_map = {
        90: "A",
        80: "B",
        70: "C",
        60: "D",
        0: "F"
    }
    for cutoff, letter in grade_map.items():
        if score >= cutoff:
            return letter
print (grade(85))
```

```
PS C:\Users\HP\Desktop\AI-Lab>
PS C:\Users\HP\Desktop\AI-Lab> & C:\Users\HP\AppData\Local\Programs\Python\Python313\python.exe c:/Users/HP/Desktop/AI-Lab/grade.py
A
B
PS C:\Users\HP\Desktop\AI-Lab>
```