AI Assisted Coding Assignment 10.5

Name: V.Harivamsh

Hall ticket no: 2303A51266

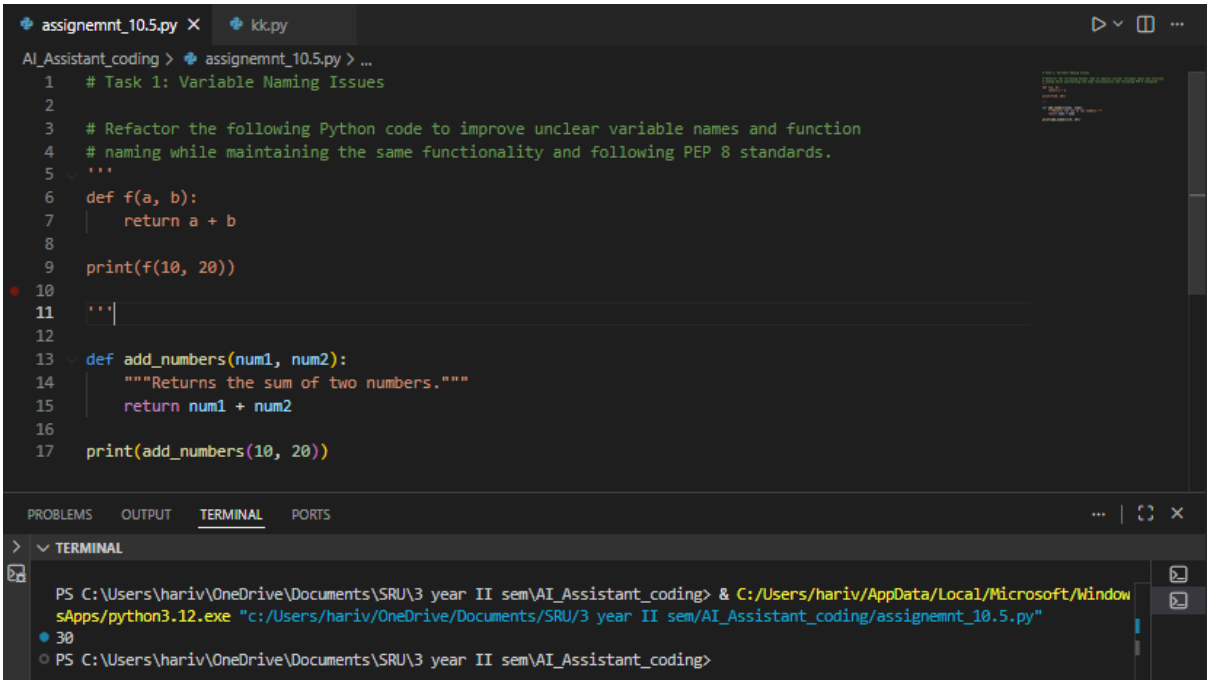
Batch no: 19

Task 1: Variable Naming Issues

Prompt:

Refactor the following Python code to improve unclear variable names and function naming while maintaining the same functionality and following PEP 8 standards. def f(a, b): return a + b print(f(10, 20))

Code & Output:



Explanation:

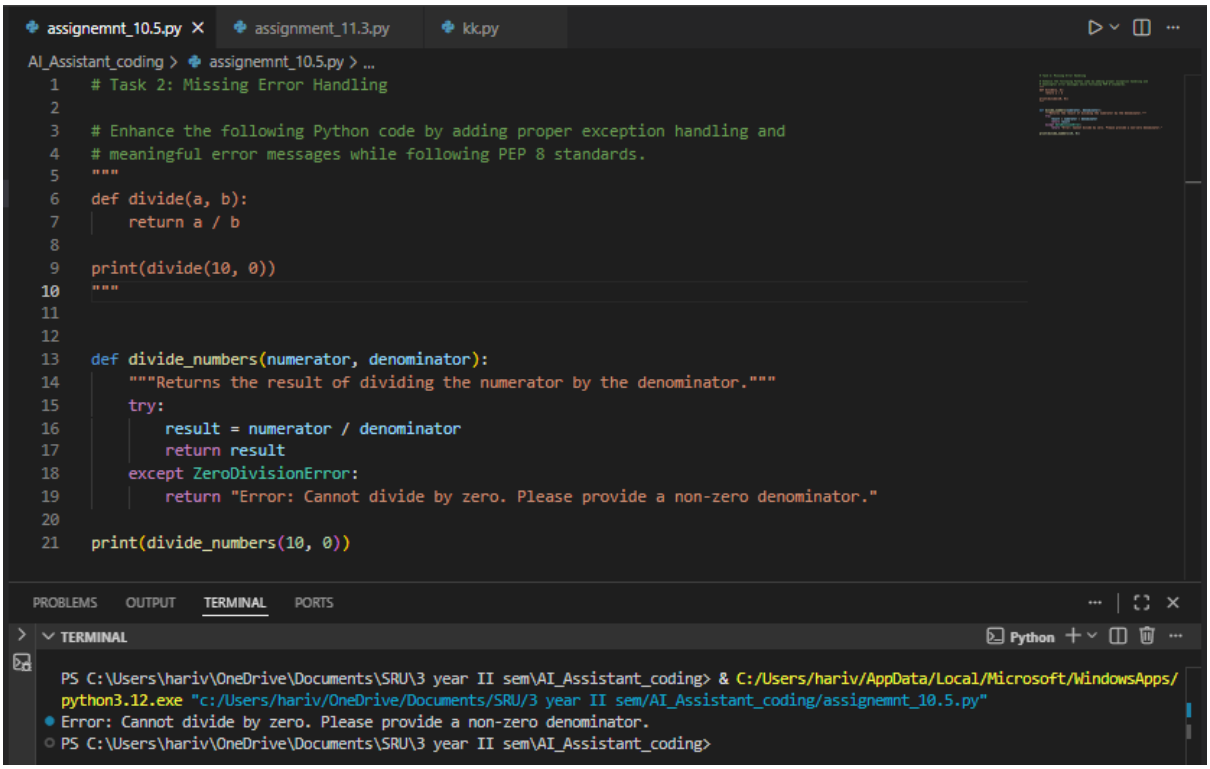
The original code uses short and unclear names like **f**, **a**, and **b**, which makes it hard to understand the purpose of the function. In the improved version, these are replaced with clear names like add_numbers**,** first_number, and second_number. This makes the code easier to read and understand at first glance. It also follows proper naming conventions, making the code cleaner and easier to maintain without changing how it works.

Task 2: Missing Error Handling

Prompt:

Enhance the following Python code by adding proper exception handling and meaningful error messages while following PEP 8 standards. def divide(a, b): return a / b print(divide(10, 0))

Code & Output:



Explanation:

The original code does not check for division by zero, which can cause the program to crash at runtime. The improved version adds a try-except block to handle this situation safely. It also uses clear variable names, making the code easier to understand. This makes the

program more reliable, avoids unexpected crashes, and provides a better user experience. Proper error handling is important for writing real-world, production-quality software.

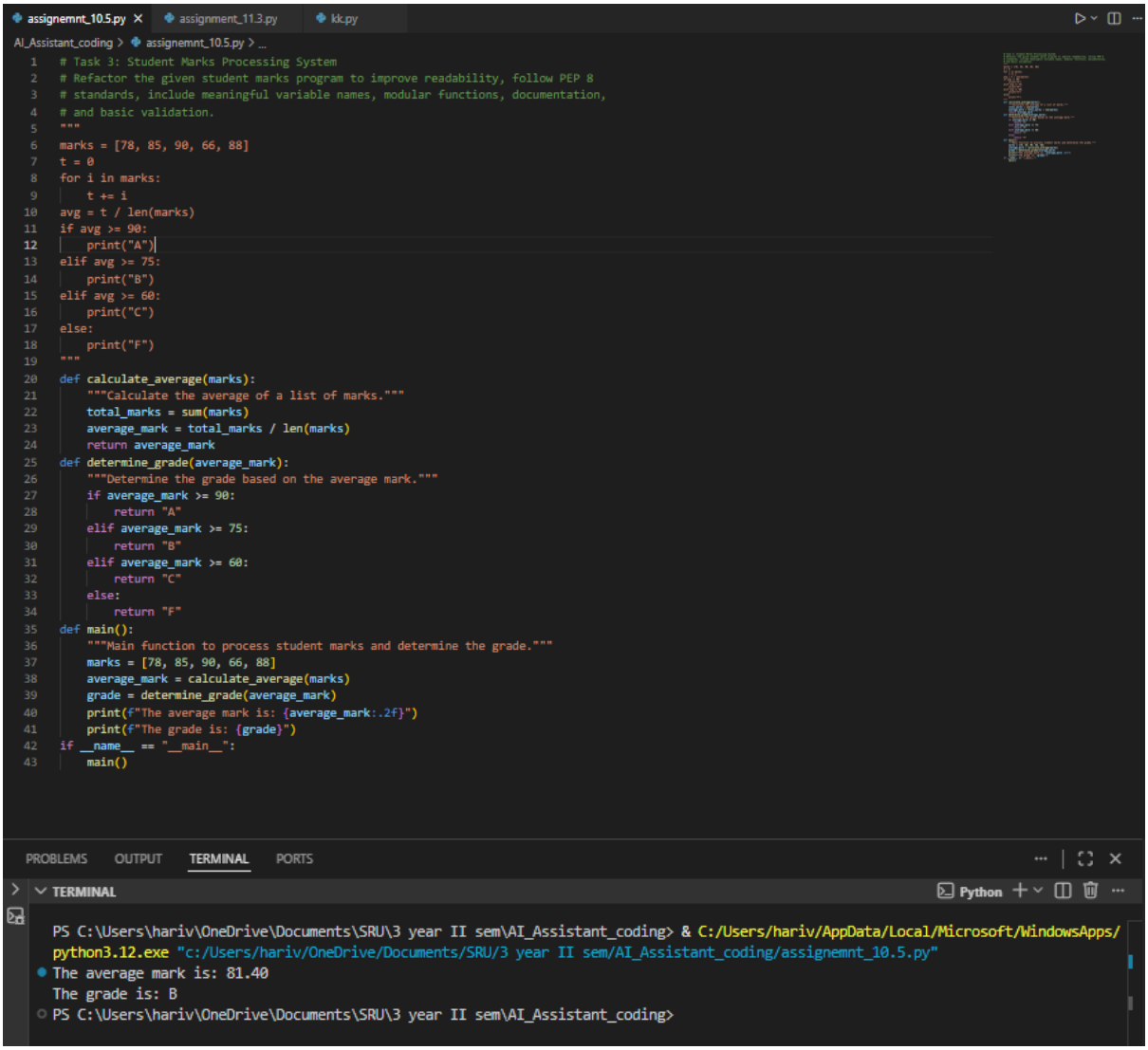
Task 3: Student Marks Processing System

Prompt:

Refactor the given student marks program to improve readability, follow PEP 8 standards, include meaningful variable names, modular functions, documentation, and basic validation.
marks=[78,85,90,66,88]

```
t=0
for i in marks:
    t=t+i
a=t/len(marks)
if a>=90:
    print("A")
elif a>=75:
    print("B")
elif a>=60:
    print("C")
else:
    print("F")
```

Code & Output:



Explanation:

The original code is not well structured and uses unclear variable names, and it also does not include proper documentation. The improved version organizes the logic into a separate function and uses meaningful variable names. It also follows PEP 8 formatting rules, making the code cleaner. Using the built-in sum() function makes the code shorter and easier to read. Overall, this version is more efficient, maintainable, and reusable.

Task 4: Adding Docstrings and Inline Comments

Prompt:

Enhance the factorial function by adding a proper docstring and meaningful inline comments.
def factorial(n):

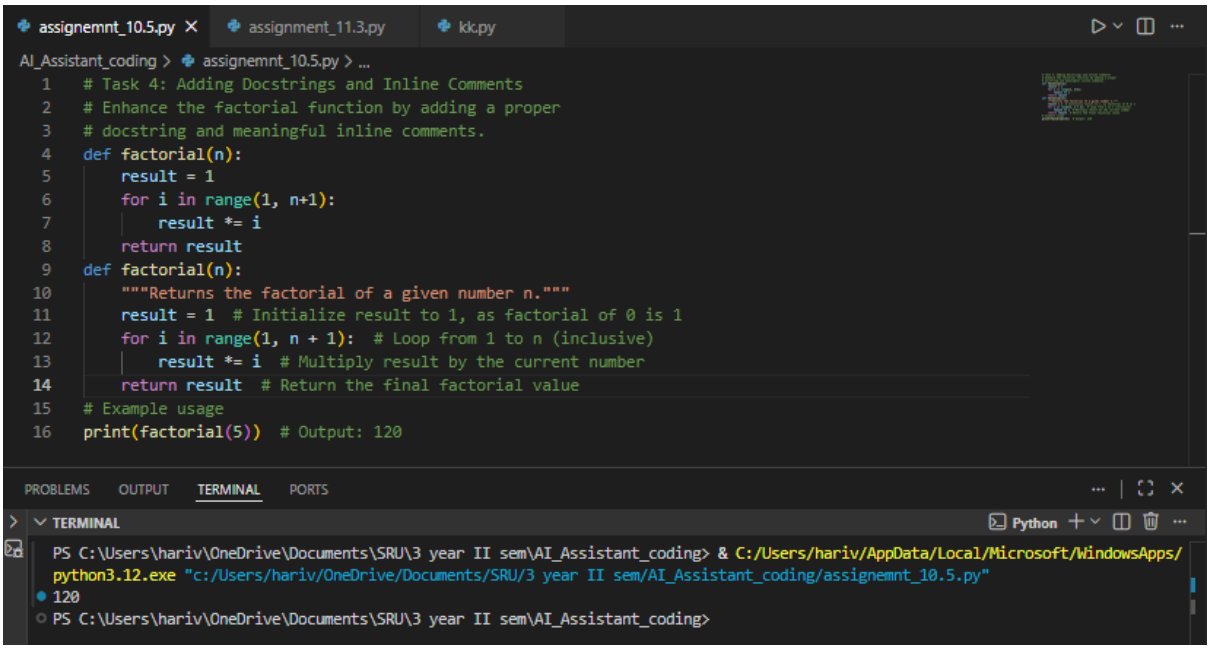
    result = 1

    for i in range(1,n+1):

        result *= i

    return result

Code & Output:



Explanation:

The AI-enhanced function includes a clear docstring that explains the purpose of the function and its parameters. It also adds simple inline comments to describe each step in the logic. The variable names are improved to make their meaning clear. All these changes make the code easier to read and better documented.

Task 5: Password Validation System (Enhanced)

Prompt:

Enhance and refactor the password validation system by adding multiple security checks and improving structure according to PEP 8 standards.

pwd = input("Enter password: ")

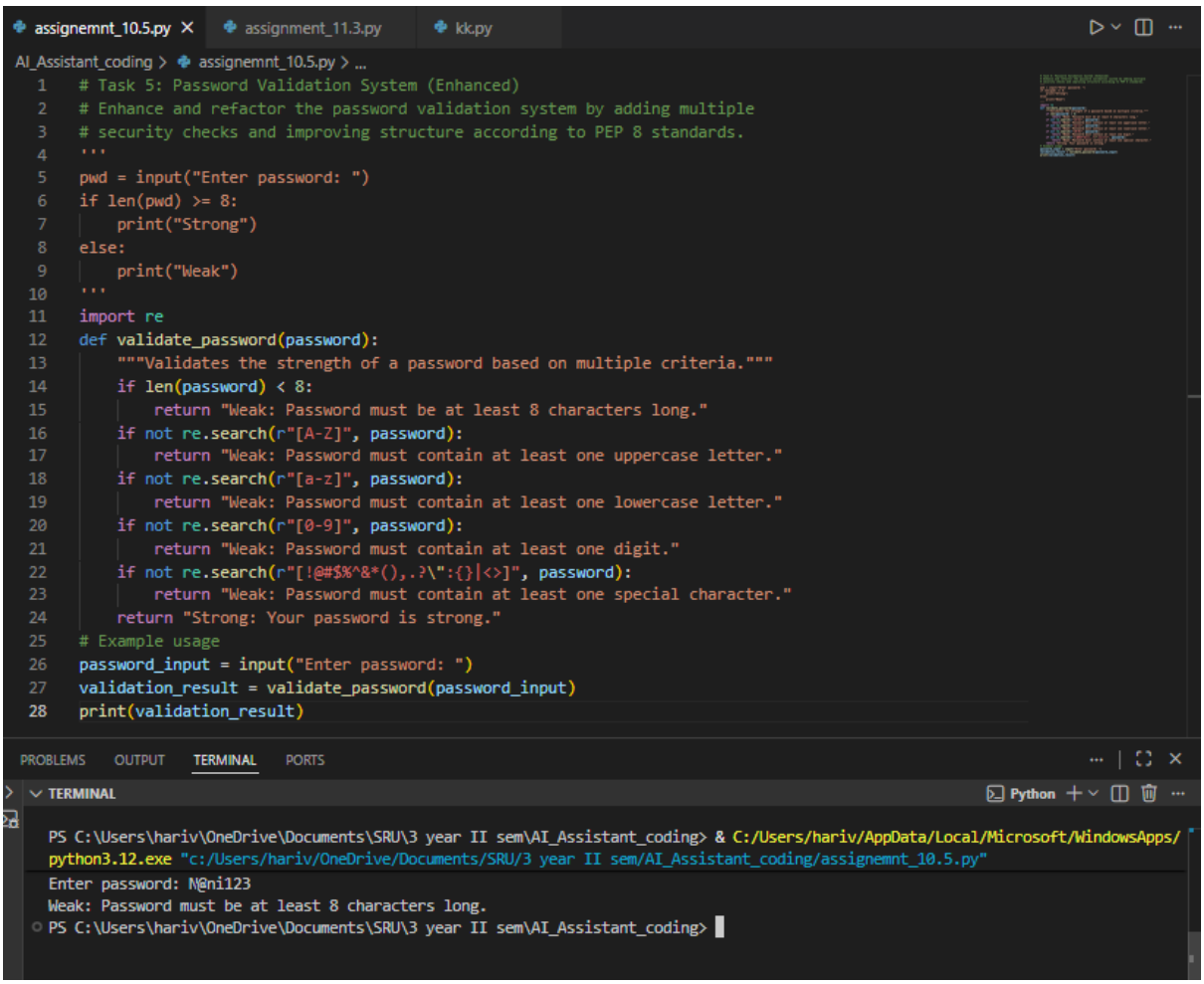
if len(pwd) >= 8:

print("Strong")

else:

print("Weak")

Code & Output:



Explanation:

The improved program adds several password security rules, such as checking for uppercase letters, lowercase letters, numbers, and special characters. The logic is organized into a separate function with proper documentation. Compared to the original version, this one is more secure and easier to understand. It is also better structured, making it easier to maintain and update in the future.