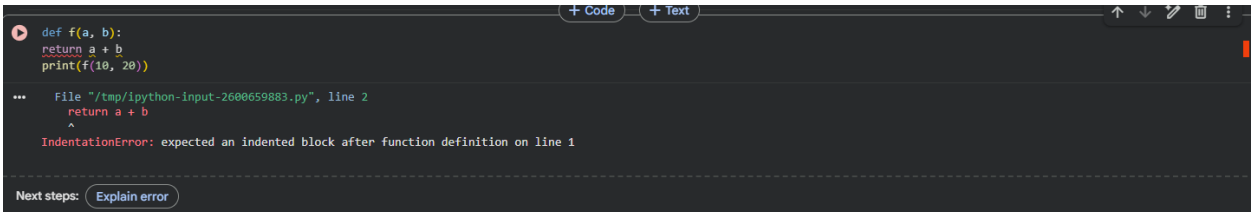
ASSIGNMENT-10.5

2303A52462

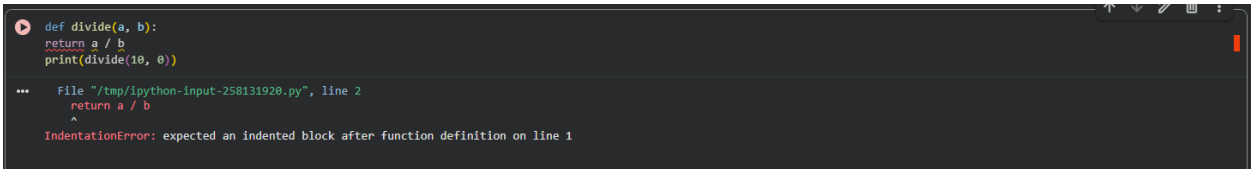
B-35

Task 1:



Corrected code with AI:

Task 2:



Corrected code with AI:

Task 3:

```
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")
```

```
  File "/tmp/ipython-input-150655628.py", line 4
    t=t+i
    ^
IndentationError: expected an indented block after 'for' statement on line 3
```

Corrected code:

```
def calculate_letter_grade(marks):
    """
    Calculates the average of a list of marks and returns the corresponding letter grade.

    Args:
        marks (list): A list of numerical scores.

    Returns:
        str: The letter grade (A, B, C, or F) or an error message.
    """
    # Basic input validation
    if not isinstance(marks, list):
        return "Error: Input must be a list."
    if not marks:
        return "Error: List of marks is empty."
    if not all(isinstance(m, (int, float)) for m in marks):
        return "Error: All marks must be numeric."

    # Calculate the average score
    total_score = sum(marks)
    average_score = total_score / len(marks)

    # Determine the letter grade based on predefined thresholds
    if average_score >= 90:
        return "A"
    elif average_score >= 75:
        return "B"
    elif average_score >= 60:
        return "C"
    else:
        return "F"

# Define student scores
student_marks = [78, 85, 90, 66, 88]

# Calculate and display the result
grade = calculate_letter_grade(student_marks)
print(f"The student's final grade is: {grade}")
```

```
The student's final grade is: B
```

Task 4:

def factorial(n):

result = 1

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

result *= i

return result

Using Ai to add docstrings and inline comments

```
def factorial(n):
    """
    Calculates the factorial of a given non-negative integer n.

    Args:
        n (int): A non-negative integer.

    Returns:
        int: The factorial of n (n!). Returns 1 for n=0.
    """
    # Initialize the result to 1 (factorial of 0 is 1)
    result = 1

    # Iterate from 1 up to n (inclusive)
    for i in range(1, n + 1):
        # Multiply current result by the loop counter
        result *= i

    # Return the final product
    return result

# Example usage
number = 5
print(f"The factorial of {number} is {factorial(number)}")
```

```
The factorial of 5 is 120
```

Task 5:

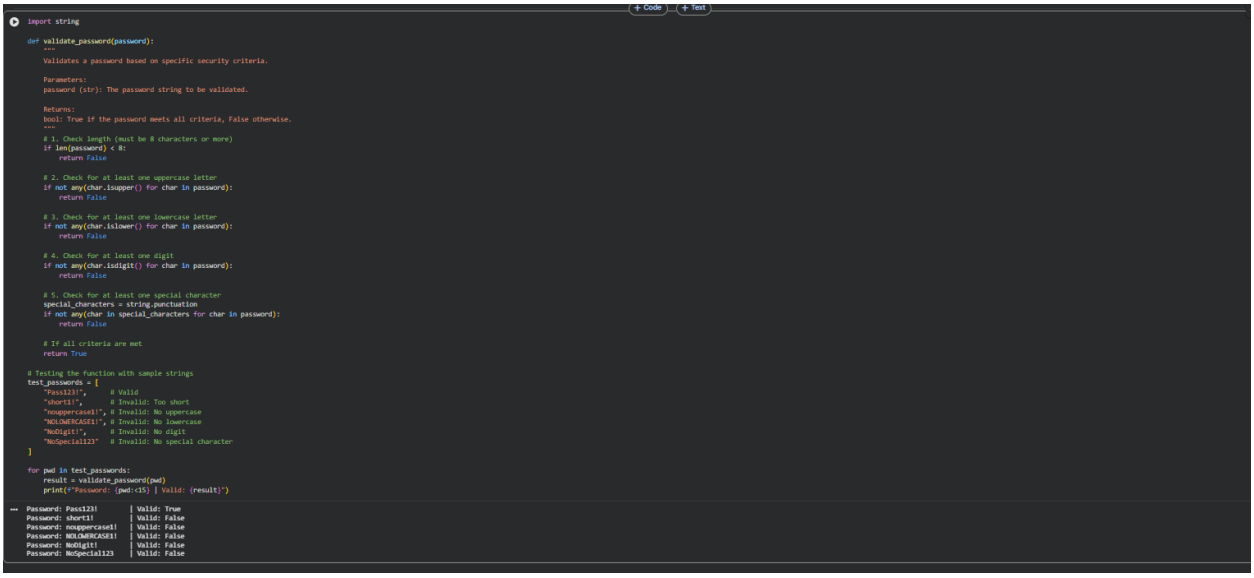
pwd = input("Enter password: ")

if len(pwd) >= 8:

print("Strong")

else:

print("Weak")



```
import string

def validate_password(password):
    """
    Validates a password based on specific security criteria.

    Parameters:
    password (str): The password string to be validated.

    Returns:
    bool: True if the password meets all criteria, False otherwise.
    """
    # 1. Check length (must be 8 characters or more)
    if len(password) < 8:
        return False

    # 2. Check for at least one uppercase letter
    if not any(char.isupper() for char in password):
        return False

    # 3. Check for at least one lowercase letter
    if not any(char.islower() for char in password):
        return False

    # 4. Check for at least one digit
    if not any(char.isdigit() for char in password):
        return False

    # 5. Check for at least one special character
    special_characters = string.punctuation
    if not any(char in special_characters for char in password):
        return False

    # If all criteria are met
    return True

# Testing the function with sample strings
test_passwords = [
    "Pass123!",     # Valid
    "short1!",      # Invalid: Too short
    "nouppercase1!", # Invalid: No uppercase
    "NOLOWERCASE1!", # Invalid: No lowercase
    "NoDigit!",     # Invalid: No digit
    "NoSpecial123"  # Invalid: No special character
]

for pwd in test_passwords:
    result = validate_password(pwd)
    print(f"Password: {pwd:<15} | Valid: {result}")
```

```
Password: Pass123!        | Valid: True
Password: short1!         | Valid: False
Password: nouppercase1!   | Valid: False
Password: NOLOWERCASE1!   | Valid: False
Password: NoDigit!        | Valid: False
Password: NoSpecial123    | Valid: False
```

Analysis: Original vs AI-Enhanced Version

1️⃣ Readability & Structure

Original	AI-Enhanced
One-line logic	Well-structured function
No explanation	Docstring + comments
Hard to extend	Easy to modify

2️⃣ Maintainability & Reusability

- Logic moved into a **function**
- Can be reused across applications
- Easy to add new rules later

3️⃣ Security Strength

Rule	Why It Matters
Length ≥ 8	Resists brute force
Uppercase	Increases complexity

Rule	Why It Matters
Lowercase	Avoids pattern predictability
Digit	Expands search space
Special character	Defends against dictionary attacks