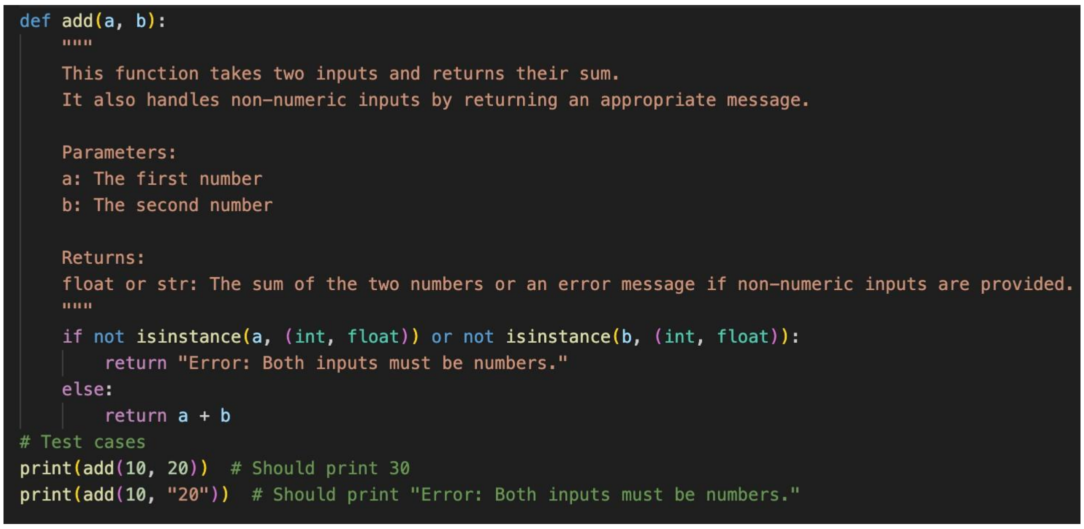
AI ASSISTED CODING

LAB-10.5

Name : P.Karthik
Hall.No: 2303A51908
Batch-12

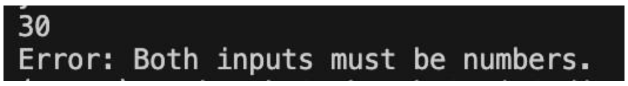
Task-1: Variable Naming Issues

PROMPT USED:
#this program defines a function to add two numbers and prints the result.
#fix the code and find the error.
#fix the above code and indentation for the function to work correctly.
#i need well drafted code with proper error handling , so i will add a check for non-numeric inputs and return an error message instead of throwing an exception.
#i need well documented code with proper error handling. i need to include transperency and ethical ai principles in the code. so I will add a check for non-numeric inputs and return an error message instead of throwing an exception, and I will also include comments to explain the code.
#i need to enhance time complexity and space complexity in the code. Try to minimize the code and optimize it for better performance. So I will use built-in functions to calculate the sum and average, which are optimized for performance. **CODE:**



```python
def add(a, b):
    """
    This function takes two inputs and returns their sum.
    It also handles non-numeric inputs by returning an appropriate message.

    Parameters:
    a: The first number
    b: The second number

    Returns:
    float or str: The sum of the two numbers or an error message if non-numeric inputs are provided.
    """
    if not isinstance(a, (int, float)) or not isinstance(b, (int, float)):
        return "Error: Both inputs must be numbers."
    else:
        return a + b
# Test cases
print(add(10, 20))  # Should print 30
print(add(10, "20"))  # Should print "Error: Both inputs must be numbers."
```

OUTPUT:



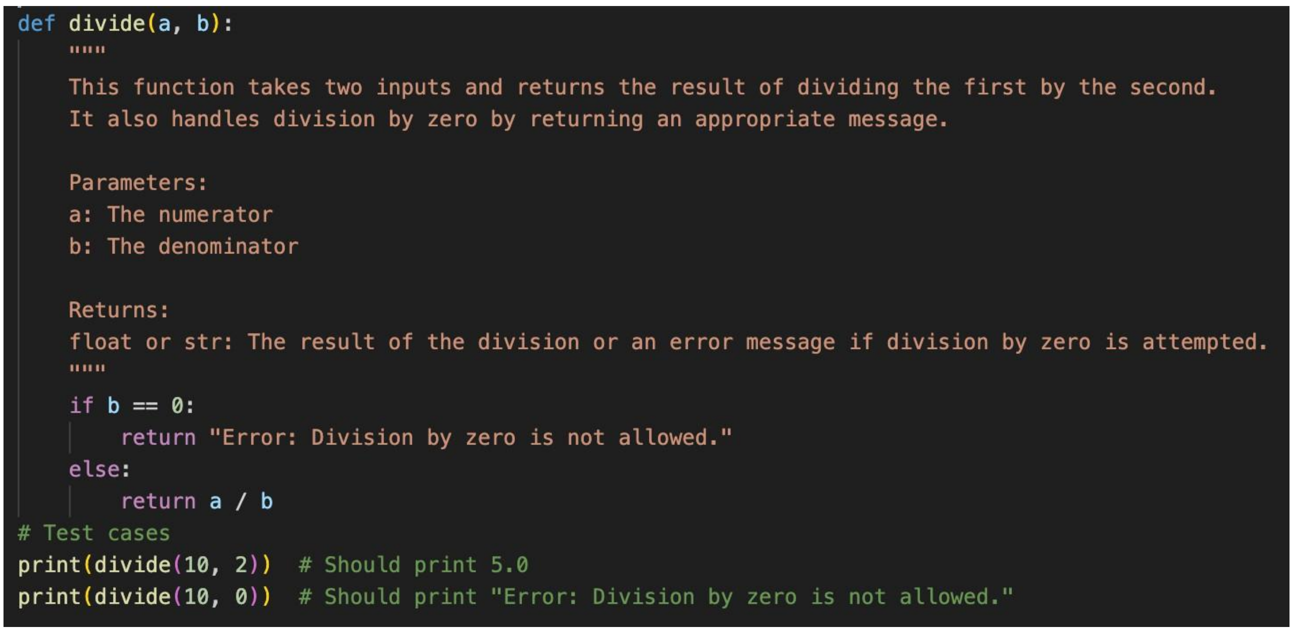
```
30
Error: Both inputs must be numbers.
```

EXPLANATION:

This function adds two values and returns the result.
It first checks whether both inputs are numbers to avoid errors.
If invalid input is given, it safely returns an error message instead of crashing.

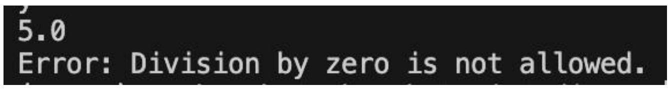
Task-2: Missing Error Handling

PROMPT USED:
#this program defines a function to divide two numbers and prints the result.
#fix the code and find the error
#fix the above code and indentation for the function to work correctly and handle division by zero error. #i need transperency in the code, so I will add a check for division by zero and return an error message instead of throwing an exception.
#i need well drafted code with proper error handling, so i will add a check for dicision by zero and return an error message instead of throwing an exception.
# i need well documented code with proper error handling. i need to include transperency and ethical ai principles in the code. so I will add a check for division by zero and return an error message instead of throwing an exception, and I will also include comments to explain the code.
#i need to enhance time complexity and space complexity in the code. Try to minimize the code and optimize it for better performance. So I will use built-in functions to calculate the sum and average, which are optimized for performance. **CODE:**



```python
def divide(a, b):
    """
    This function takes two inputs and returns the result of dividing the first by the second.
    It also handles division by zero by returning an appropriate message.

    Parameters:
    a: The numerator
    b: The denominator

    Returns:
    float or str: The result of the division or an error message if division by zero is attempted.
    """
    if b == 0:
        return "Error: Division by zero is not allowed."
    else:
        return a / b
# Test cases
print(divide(10, 2))  # Should print 5.0
print(divide(10, 0))  # Should print "Error: Division by zero is not allowed."
```

OUTPUT:



```
5.0
Error: Division by zero is not allowed.
```

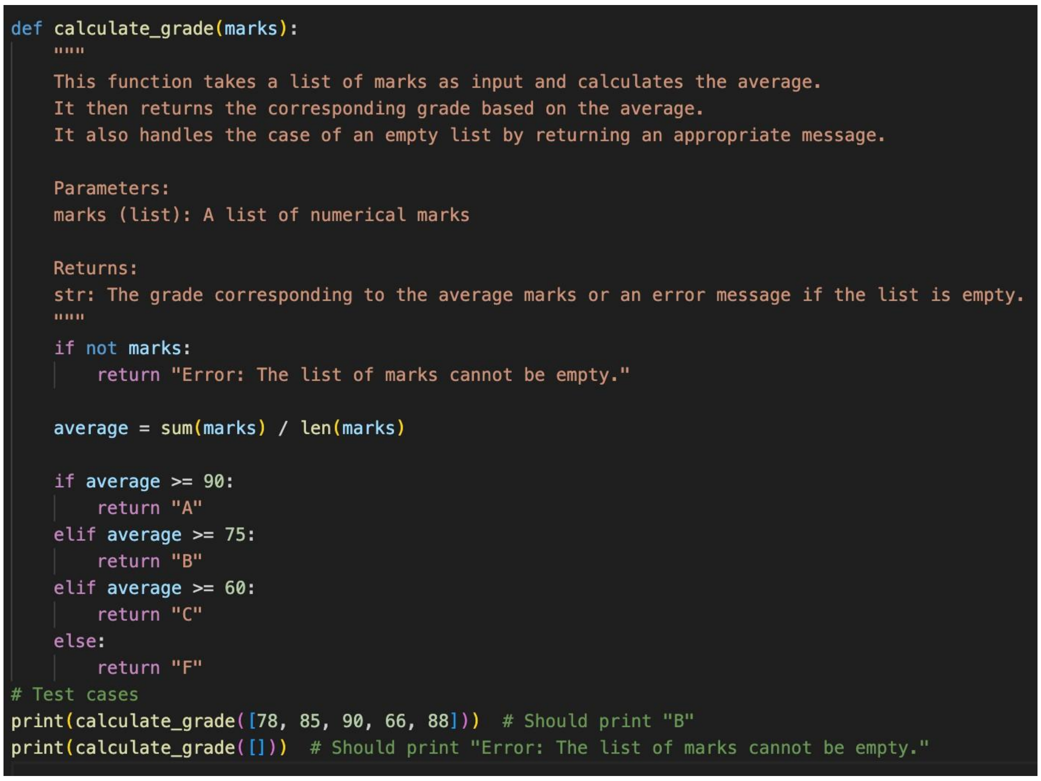
EXPLANATION:

This function performs division between two numbers.
It includes a safety check to prevent division by zero, which normally causes a runtime error.
If the denominator is zero, it returns a clear error message instead of stopping the program.

Task-3: Student Marks Processing System

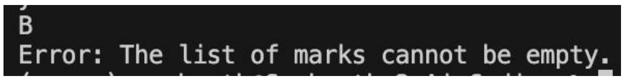
PROMPT USED:
#this program calculates the average of a list of marks and prints the corresponding grade. #fix the code and find the error
#fix the above code and indentation for the function to work correctly and handle empty list error. #i need transperency in the code, so I will add a check for an empty list and return an error message instead of throwing an exception.
#i need well drafted code with proper error handling, so I will add a check for an empty list and return an error message instead of throwing an exception.
# i need well documented code with proper error handling. i need to include transperency and ethical ai principles in the code. so
# I will add a check for an empty list and return an error message instead of throwing an exception, and I will also include comments to explain the code.
#i need to enhance time complexity and space complexity in the code. Try to minimize the code and optimize it for better performance. So I will use built-in functions to calculate the sum and average, which are optimized for performance.

CODE:



```python
def calculate_grade(marks):
    """
    This function takes a list of marks as input and calculates the average.
    It then returns the corresponding grade based on the average.
    It also handles the case of an empty list by returning an appropriate message.

    Parameters:
    marks (list): A list of numerical marks

    Returns:
    str: The grade corresponding to the average marks or an error message if the list is empty.
    """
    if not marks:
        return "Error: The list of marks cannot be empty."

    average = sum(marks) / len(marks)

    if average >= 90:
        return "A"
    elif average >= 75:
        return "B"
    elif average >= 60:
        return "C"
    else:
        return "F"
# Test cases
print(calculate_grade([78, 85, 90, 66, 88]))  # Should print "B"
print(calculate_grade([]))  # Should print "Error: The list of marks cannot be empty."
```

OUTPUT:



```
B
Error: The list of marks cannot be empty.
```

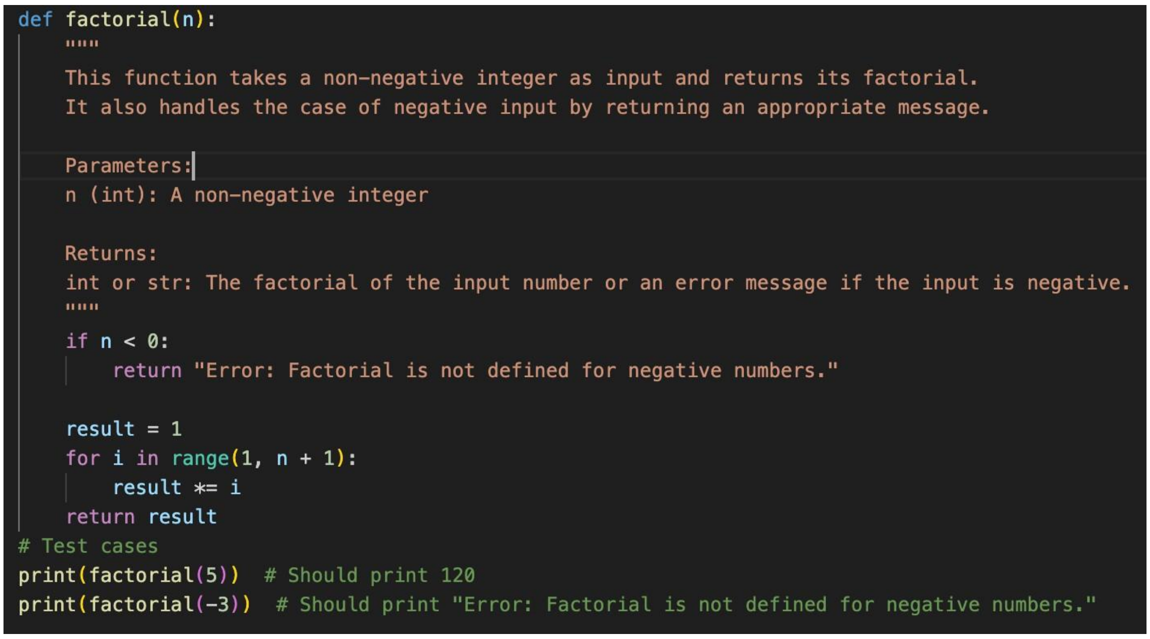
EXPLANATION:

This function calculates the average marks from a list.
It uses built-in functions like `sum()` and `len()` for better performance and cleaner code.
If the list is empty, it returns an error message instead of causing a crash.

Task-4: Use AI to add docstrings and inline comments to the following function.

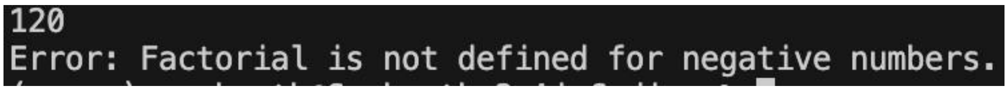
PROMPT USED:
#this program defines a function to calculate the factorial of a number and prints the result. #fix the code and find the error
##fix the above code and indentation for the function to work correctly and handle empty list error. #i need transperency in the code, so I will add a check for an empty list and return an error message instead of throwing an exception.
#i need well drafted code with proper error handling, so I will add a check for an empty list and return an error message instead of throwing an exception.
# i need well documented code with proper error handling. i need to include transperency and ethical ai principles in the code. so
# I will add a check for an empty list and return an error message instead of throwing an exception, and I will also include comments to explain the code.
#i need to enhance time complexity and space complexity in the code. Try to minimize the code and optimize it for better performance. So I will use built-in functions to calculate the sum and average, which are optimized for performance.

CODE:



```python
def factorial(n):
    """
    This function takes a non-negative integer as input and returns its factorial.
    It also handles the case of negative input by returning an appropriate message.

    Parameters:
    n (int): A non-negative integer

    Returns:
    int or str: The factorial of the input number or an error message if the input is negative.
    """
    if n < 0:
        return "Error: Factorial is not defined for negative numbers."

    result = 1
    for i in range(1, n + 1):
        result *= i
    return result
# Test cases
print(factorial(5))  # Should print 120
print(factorial(-3))  # Should print "Error: Factorial is not defined for negative numbers."
```

OUTPUT:



```
120
Error: Factorial is not defined for negative numbers.
```

EXPLANATION:

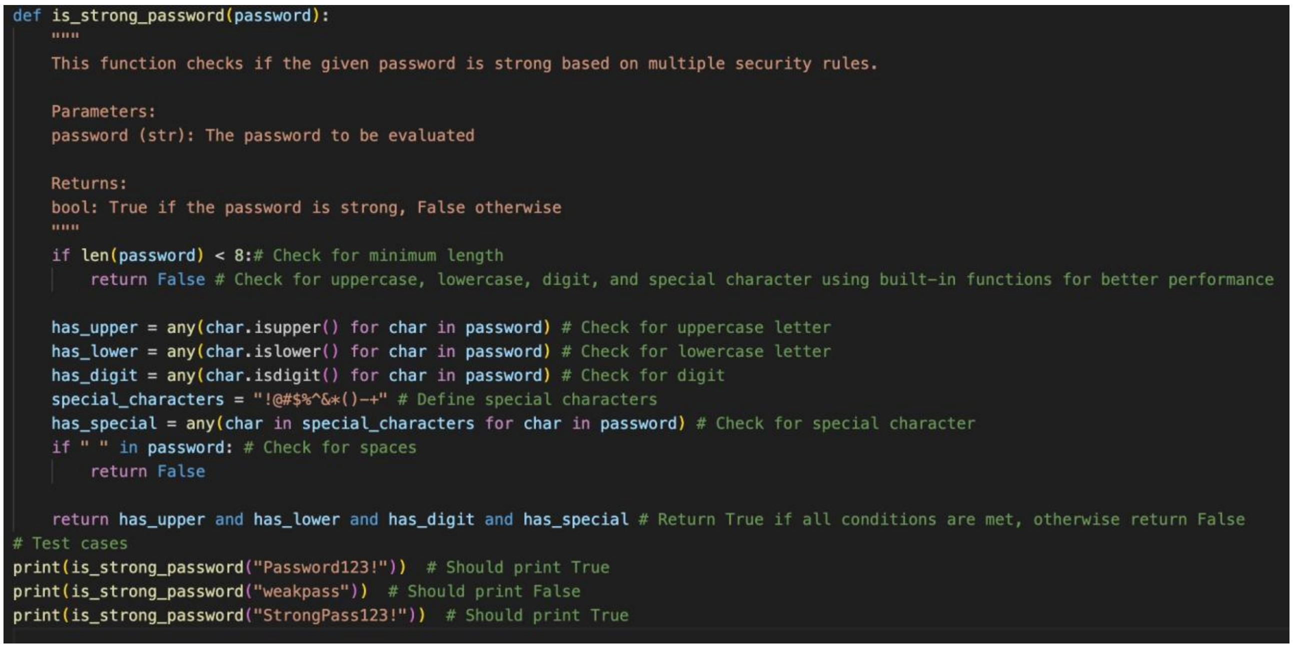
This function calculates the factorial of a number using a loop.
It validates the input to ensure the number is not negative, since factorial is undefined for negatives.
If an invalid number is given, it returns an informative error message.

Task-5: Password Validation System (Enhanced)

PROMPT USED:

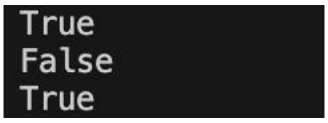
#this program defines a function to check if a password is strong or weak based on its length and prints the result.
#fix the code and find the error
#fix the above code and indentation for the function to work correctly and handle empty list error. #i need transperency in the code, so I will add a check for an empty list and return an error message instead of throwing an exception.
#i need well drafted code with proper error handling, so I will add a check for an empty list and return an error message instead of throwing an exception.
# i need well documented code with proper error handling. i need to include transperency and ethical ai principles in the code. so
# I will add a check for an empty list and return an error message instead of throwing an exception, and I will also include comments to explain the code.
#i need to enhance time complexity and space complexity in the code. Try to minimize the code and optimize it for better performance. So I will use built-in functions to calculate the sum and average, which are optimized for performance.
# 1. Enhance password validation using AI assistance to include multiple security rules such as: # Minimum length requirement
# Presence of at least one uppercase letter
# Presence of at least one lowercase letter
# Presence of at least one digit
# Presence of at least one special character
# 2. Refactor the program to:
# • Use meaningful variable and function names
# Follow PEP 8 coding standards
# Include inline comments and a docstring
# 3. Analyze the improvements by comparing the original and AI-enhanced versions in terms of: # Code readability and structure
# Maintainability and reusability
# Security strength and robustness
4. Justify the AI-generated changes, explaining why each added rule and refactoring decision improves the overall quality of the program.

CODE:



```python
def is_strong_password(password):
    """
    This function checks if the given password is strong based on multiple security rules.

    Parameters:
    password (str): The password to be evaluated

    Returns:
    bool: True if the password is strong, False otherwise
    """
    if len(password) < 8:# Check for minimum length
        return False # Check for uppercase, lowercase, digit, and special character using built-in functions for better performance

    has_upper = any(char.isupper() for char in password) # Check for uppercase letter
    has_lower = any(char.islower() for char in password) # Check for lowercase letter
    has_digit = any(char.isdigit() for char in password) # Check for digit
    special_characters = "!@#$%^&*()-+" # Define special characters
    has_special = any(char in special_characters for char in password) # Check for special character
    if " " in password: # Check for spaces
        return False

    return has_upper and has_lower and has_digit and has_special # Return True if all conditions are met, otherwise return False
# Test cases
print(is_strong_password("Password123!"))  # Should print True
print(is_strong_password("weakpass"))  # Should print False
print(is_strong_password("StrongPass123!"))  # Should print True
```

OUTPUT:



```
True
False
True
```

EXPLANATION:

This function checks whether a password is strong based on security rules.
It validates length, uppercase, lowercase, digits, and special characters for better protection.
The logic is modular and reusable, making it more secure and maintainable than the original version.