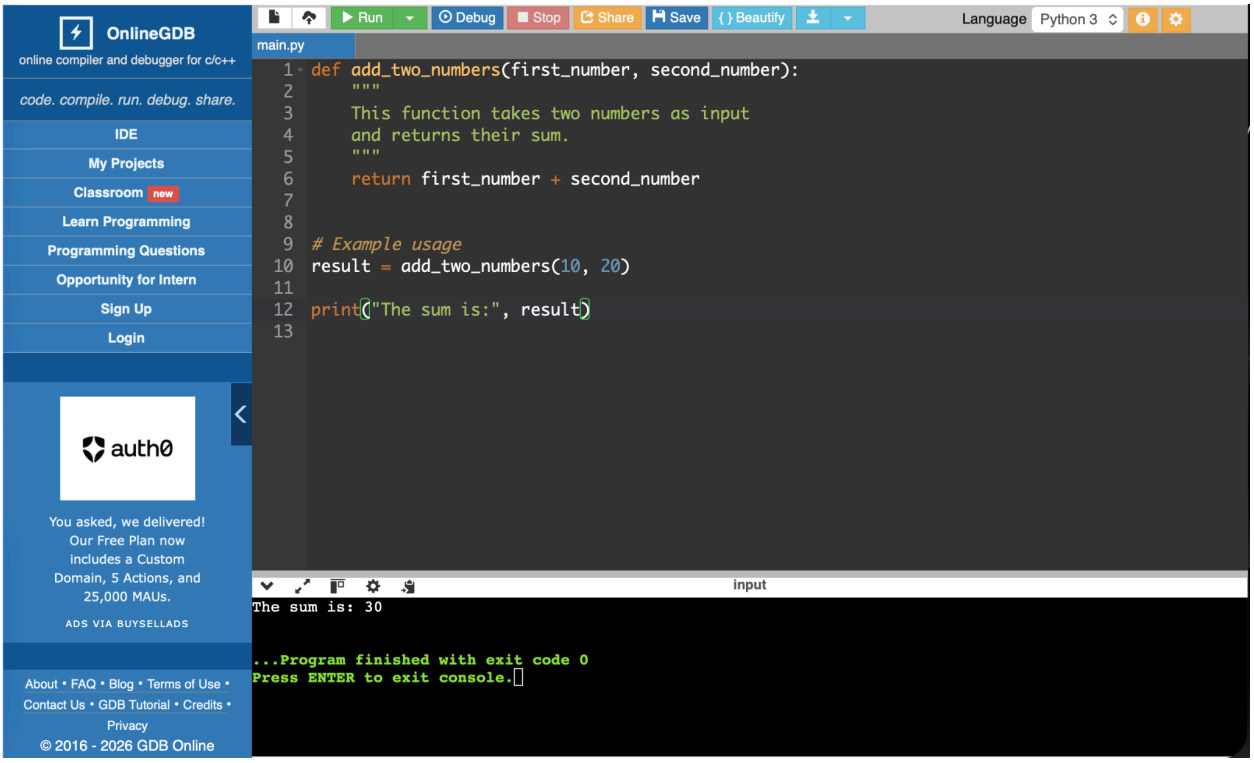
# 2303A52416

# batch=35

Task Description #1 – Variable Naming Issues
Task: Use AI to improve unclear variable names.
Sample Input Code:
def f(a, b):
return a + b
print(f(10, 20))
Expected Output:
• Code rewritten with meaningful function and variable names.

# Code

```
def add_two_numbers(first_number, second_number):
    """
    This function takes two numbers as input
    and returns their sum.
    """
    return first_number + second_number


# Example usage
result = add_two_numbers(10, 20)

print("The sum is:", result)
```

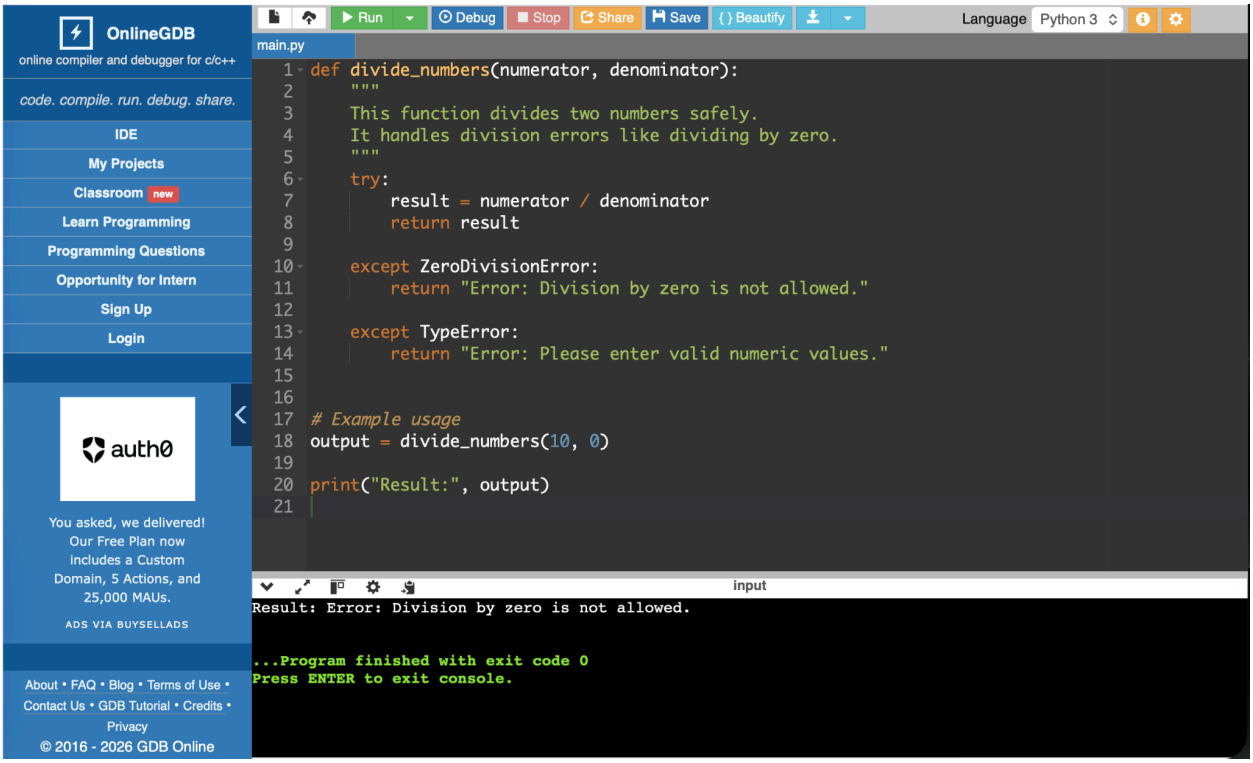



Task Description #2 – Missing Error Handling
Task: Use AI to add proper error handling.
Sample Input Code:
def divide(a, b):
return a / b
print(divide(10, 0))
Expected Output:
• Code with exception handling and clear error messages
Code

```
def divide_numbers(numerator, denominator):
    """
    This function divides two numbers safely.
    It handles division errors like dividing by zero.
    """
    try:
        result = numerator / denominator
        return result

    except ZeroDivisionError:
        return "Error: Division by zero is not allowed."

    except TypeError:
        return "Error: Please enter valid numeric values."
```

# Example usage
output = divide_numbers(10, 0)

print("Result:", output)



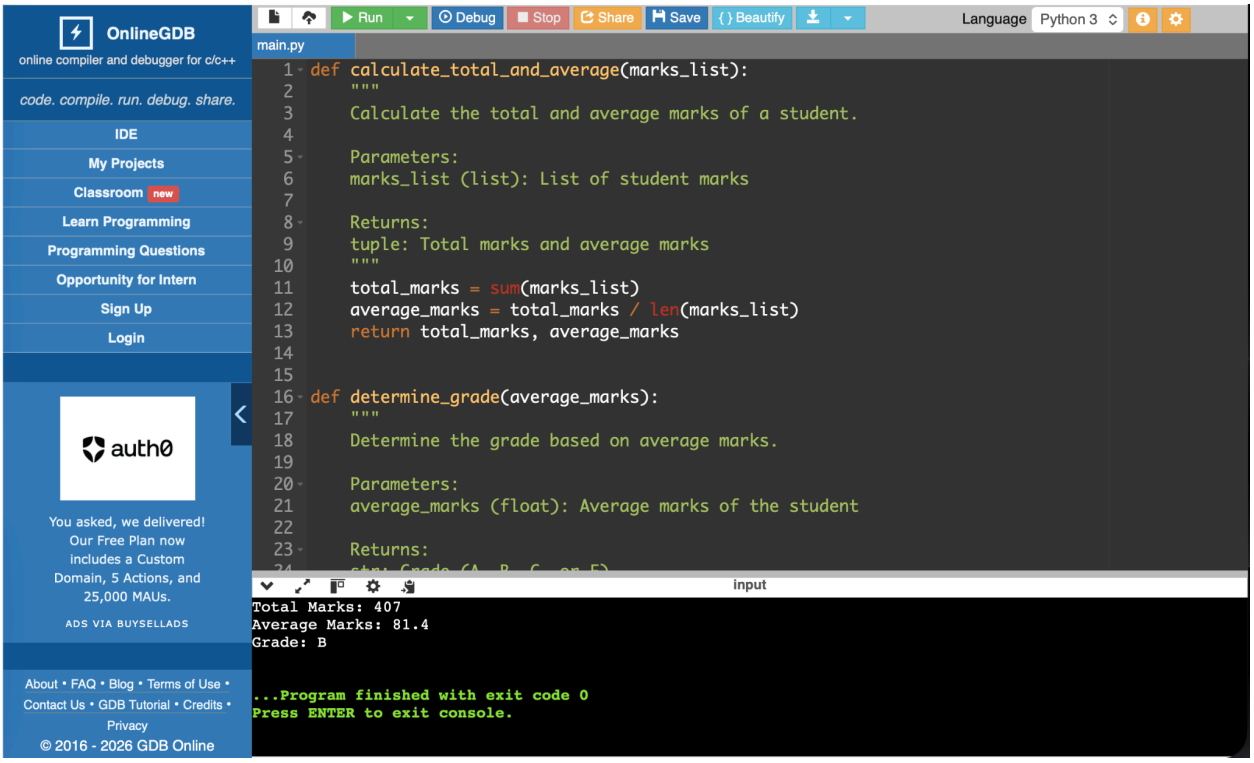

Task Description #3: Student Marks Processing System
The following program calculates total, average, and grade of a student, but it has poor readability, style issues, and no error handling.

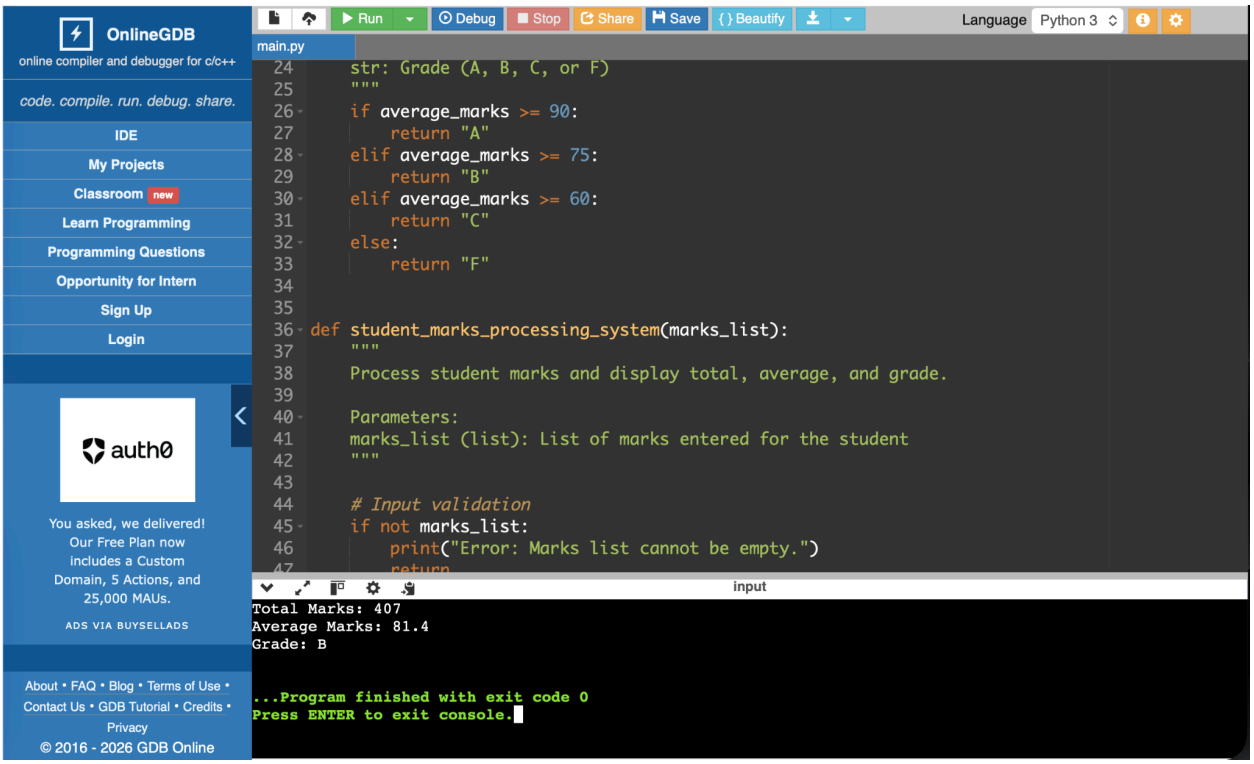
```
marks=[78,85,90,66,88]
t=0
for i in marks:
t=t+i
a=t/len(marks)
if a>=90:
print("A")
elif a>=75:
print("B")
elif a>=60:
print("C")
else:
print("F")
```

Task:
• Use AI to refactor the code to follow PEP 8 standards.
• Add meaningful variable names, functions, and comments.
• Add basic input validation and documentation.

```
def calculate_total_and_average(marks_list):
    """
    Calculate the total and average marks of a student.

    Parameters:
    marks_list (list): List of student marks

    Returns:
    tuple: Total marks and average marks
    """
    total_marks = sum(marks_list)
    average_marks = total_marks / len(marks_list)
    return total_marks, average_marks


def determine_grade(average_marks):
    """
    Determine the grade based on average marks.

    Parameters:
    average_marks (float): Average marks of the student

    Returns:
    str: Grade (A, B, C, or F)
    """
    if average_marks >= 90:
        return "A"
    elif average_marks >= 75:
        return "B"
    elif average_marks >= 60:
        return "C"
    else:
        return "F"


def student_marks_processing_system(marks_list):
    """
    Process student marks and display total, average, and grade.
```

```
    Parameters:
    marks_list (list): List of marks entered for the student
    """

    # Input validation
    if not marks_list:
        print("Error: Marks list cannot be empty.")
        return

    if not all(isinstance(mark, (int, float)) for mark in marks_list):
        print("Error: All marks must be numeric values.")
        return

    # Calculate total and average
    total, average = calculate_total_and_average(marks_list)

    # Determine grade
    grade = determine_grade(average)

    # Display results
    print("Total Marks:", total)
    print("Average Marks:", average)
    print("Grade:", grade)


# Example usage
student_marks = [78, 85, 90, 66, 88]

student_marks_processing_system(student_marks)
```


OnlineGDB
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Opportunity for Intern
Sign Up
Login
auth0
You asked, we delivered! Our Free Plan now includes a Custom Domain, 5 Actions, and 25,000 MAUs.
ADS VIA BUYSELLADS
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2026 GDB Online
Run
Debug
Stop
Share
Save
{ } Beautify
Language
Python 3
main.py
def calculate_total_and_average(marks_list):
"""
Calculate the total and average marks of a student.
Parameters:
marks_list (list): List of student marks
Returns:
tuple: Total marks and average marks
"""
total_marks = sum(marks_list)
average_marks = total_marks / len(marks_list)
return total_marks, average_marks
def determine_grade(average_marks):
"""
Determine the grade based on average marks.
Parameters:
average_marks (float): Average marks of the student
Returns:
input
Total Marks: 407
Average Marks: 81.4
Grade: B
...Program finished with exit code 0
Press ENTER to exit console.

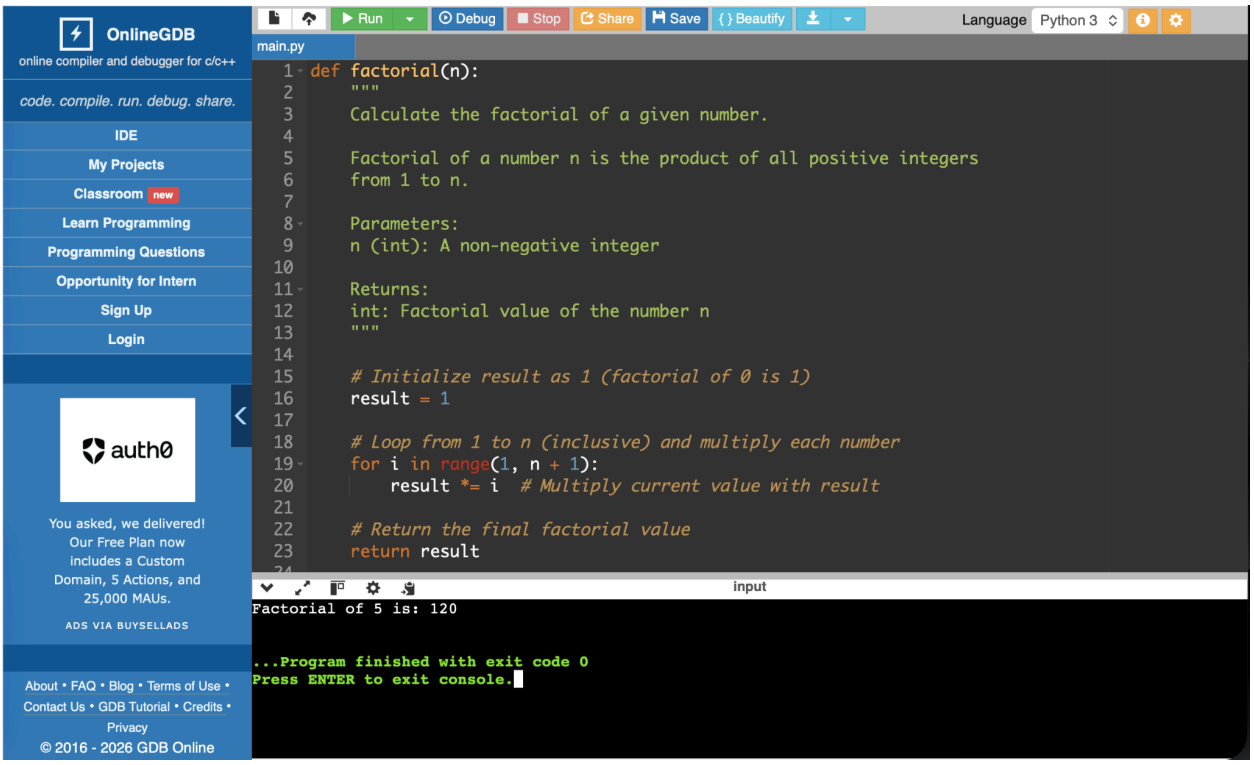

OnlineGDB
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Opportunity for Intern
Sign Up
Login
auth0
You asked, we delivered! Our Free Plan now includes a Custom Domain, 5 Actions, and 25,000 MAUs.
ADS VIA BUYSELLADS
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2026 GDB Online
Run
Debug
Stop
Share
Save
{ } Beautify
Language
Python 3
main.py
str: Grade (A, B, C, or F)
"""
if average_marks >= 90:
return "A"
elif average_marks >= 75:
return "B"
elif average_marks >= 60:
return "C"
else:
return "F"
def student_marks_processing_system(marks_list):
"""
Process student marks and display total, average, and grade.
Parameters:
marks_list (list): List of marks entered for the student
"""
# Input validation
if not marks_list:
print("Error: Marks list cannot be empty.")
input
Total Marks: 407
Average Marks: 81.4
Grade: B
...Program finished with exit code 0
Press ENTER to exit console.

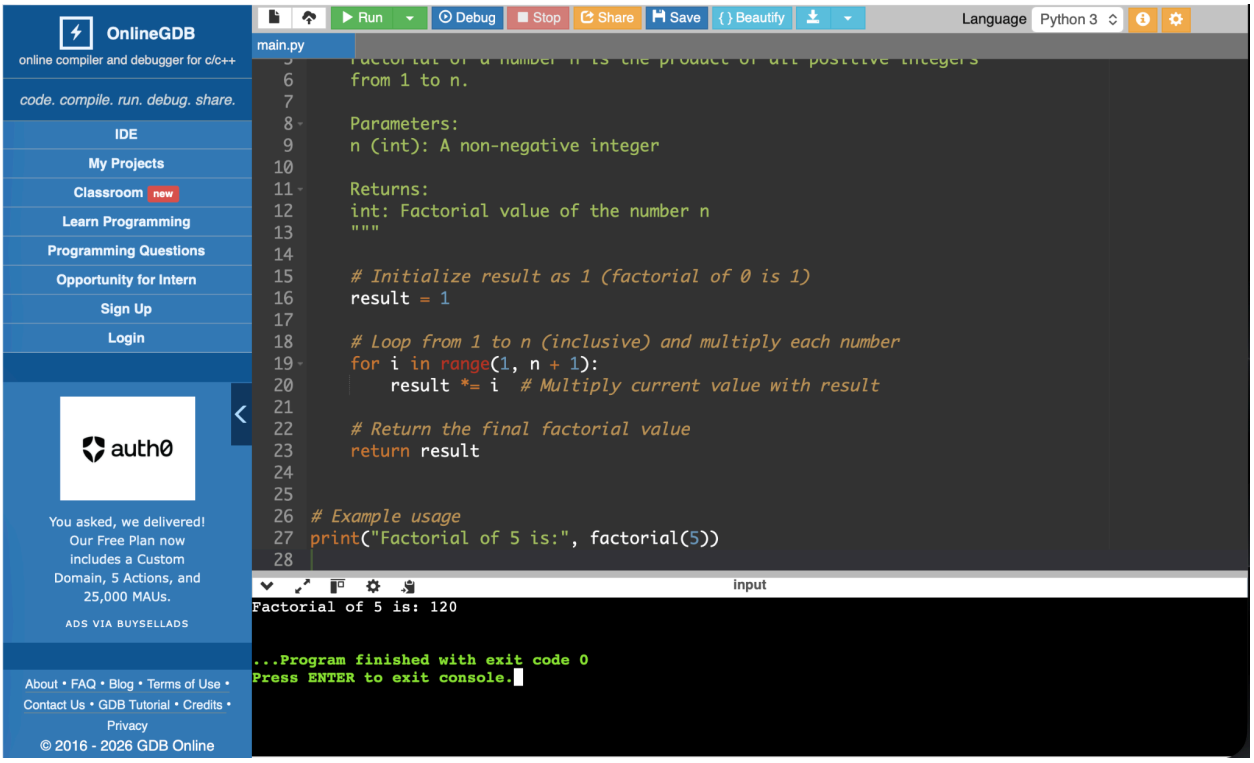

Task Description #4: Use AI to add docstrings and inline comments to the following function.

```
def factorial(n):
result = 1
for i in range(1,n+1):
result *= i
return result
```

```
def factorial(n):
    """
    Calculate the factorial of a given number.

    Factorial of a number n is the product of all positive integers
    from 1 to n.

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

    Returns:
    int: Factorial value of the number n
    """

    # Initialize result as 1 (factorial of 0 is 1)
    result = 1
```

```
    # Loop from 1 to n (inclusive) and multiply each number
    for i in range(1, n + 1):
        result *= i  # Multiply current value with result

    # Return the final factorial value
    return result


# Example usage
print("Factorial of 5 is:", factorial(5))
```






Task Description #5: Password Validation System (Enhanced)
The following Python program validates a password using only a minimum length check, which is insufficient for real-world security requirements.

```
pwd = input("Enter password: ")
if len(pwd) >= 8:
print("Strong")
else:
print("Weak")
```

Task:

1. Enhance password validation using AI assistance to include multiple security rules such as:
o Minimum length requirement
o Presence of at least one uppercase letter
o Presence of at least one lowercase letter
o Presence of at least one digit
o Presence of at least one special character
2. Refactor the program to:
o Use meaningful variable and function names
o Follow PEP 8 coding standards
o Include inline comments and a docstring
3. Analyze the improvements by comparing the original and AI-enhanced versions in terms of:
o Code readability and structure

o Maintainability and reusability
o 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.

```
import re
```

Task Description #5: Password Validation System (Enhanced)
The following Python program validates a password using only a minimum length check, which is insufficient for real-world security requirements.

```
pwd = input("Enter password: ")
if len(pwd) >= 8:
print("Strong")
else:
print("Weak")
```

Task:
1. Enhance password validation using AI assistance to include multiple security rules such as:
o Minimum length requirement
o Presence of at least one uppercase letter
o Presence of at least one lowercase letter
o Presence of at least one digit
o Presence of at least one special character
2. Refactor the program to:
o Use meaningful variable and function names
o Follow PEP 8 coding standards
o Include inline comments and a docstring
3. Analyze the improvements by comparing the original and AI-enhanced versions in terms of:
o Code readability and structure
o Maintainability and reusability
o 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.

```
import re

pwd = input("Enter password: ")

if (len(pwd) >= 8 and
```

```
    re.search(r"[A-Z]", pwd) and
    re.search(r"[a-z]", pwd) and
    re.search(r"[0-9]", pwd) and
    re.search(r"[!@#$%^&*]", pwd)):

    print("Strong Password ✅")
else:
    print("Weak Password ❌")
```