

AI ASSISTED CODING

LAB_Assignment_10.5.

2303A52197

BATCH - 35

Task Description #1 – Variable Naming Issues

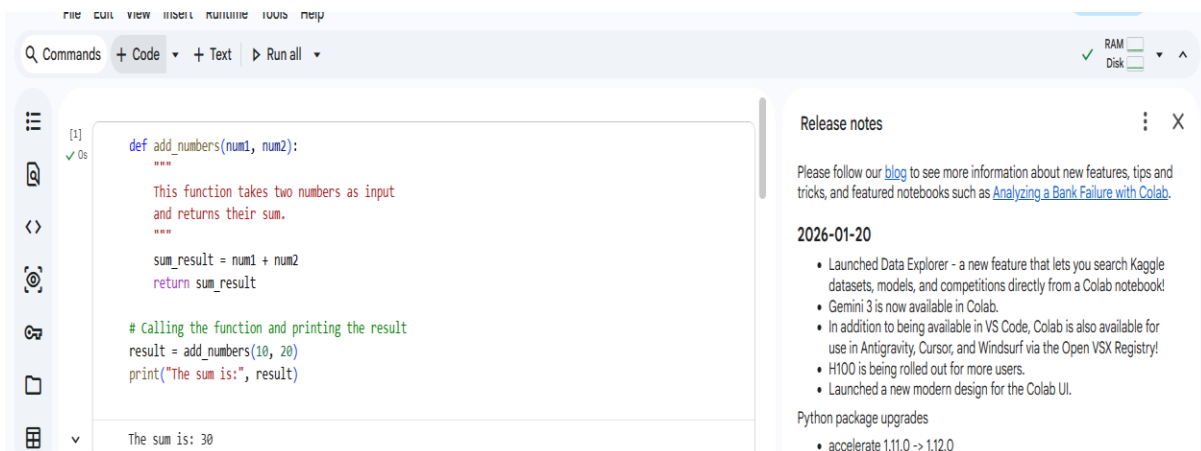
Task: Use AI to improve unclear variable names.

Sample Input Code:

```
def f(a, b):  
  
    return a + b  
  
print(f(10, 20))
```

Expected Output:

- Code rewritten with meaningful function and variable names.



Task Description #2 – Missing Error Handling

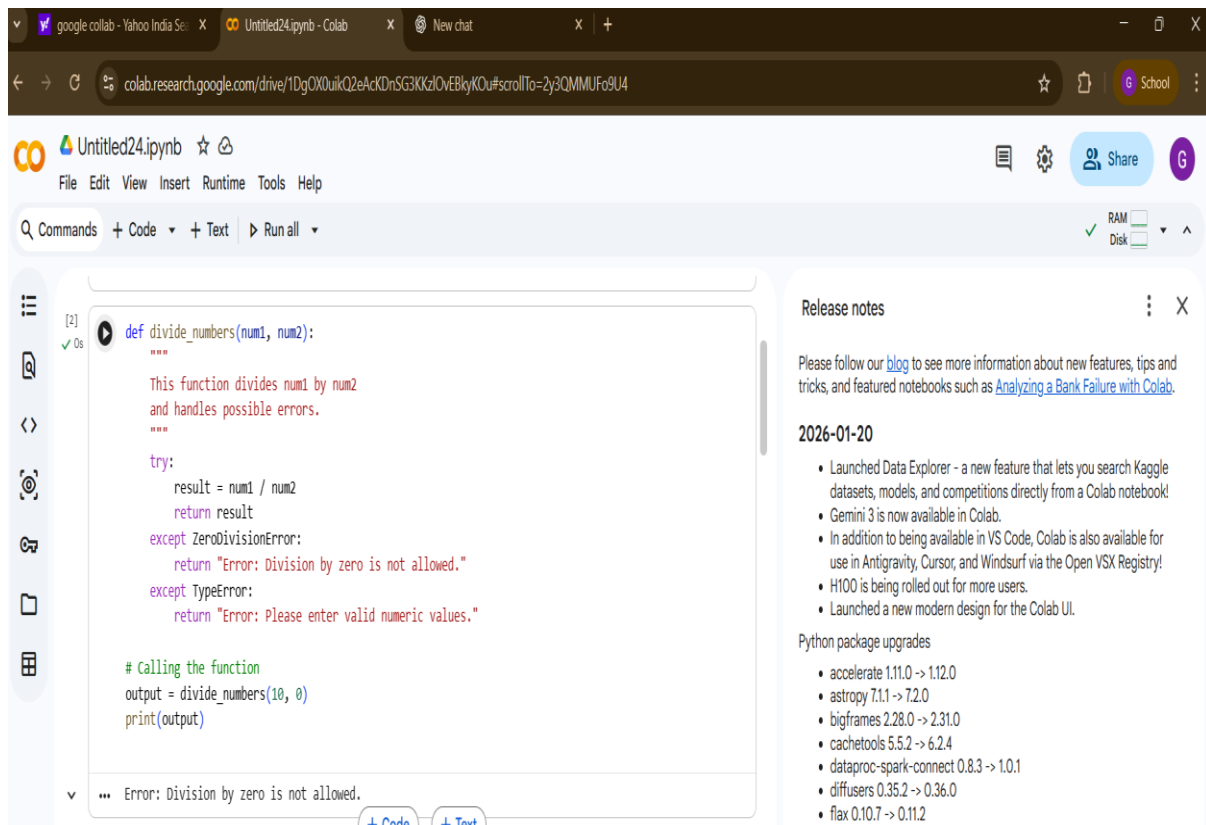
Task: Use AI to add proper error handling.

Sample Input Code:

```
def divide(a, b):  
  
    return a / b  
  
print(divide(10, 0))
```

Expected Output:

- Code with exception handling and clear error messages



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.

google colab - Yahoo India Se... x Untitled24.ipynb - Colab x New chat x | +

colab.research.google.com/drive/1DgOXDuikQ2eAcKdN5G3KKzIOvEBlyKOu#scrollTo=2y3QMMUF9U4

Untitled24.ipynb ☆

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all

```
[4] ✓ Os
def calculate_total_and_average(marks_list):
    """
    This function calculates the total and average
    of a list of student marks.
    """
    total_marks = sum(marks_list)
    average_marks = total_marks / len(marks_list)
    return total_marks, average_marks

def calculate_grade(average_marks):
    """
    This function returns the grade
    based on the average marks.
    """
    if average_marks >= 90:
        return "A"
    elif average_marks >= 75:
        return "B"
    elif average_marks >= 60:
        return "C"
    else:
        return "F"

def validate_marks(marks_list):
    """
    This function checks whether the marks list
    contains valid numeric values between 0 and 100.
    """
```

Release notes

Please follow our [blog](#) to see more information about new features, tips and tricks, and featured notebooks such as [Analyzing a Bank Failure with Colab](#).

2026-01-20

- Launched Data Explorer - a new feature that lets you search Kaggle datasets, models, and competitions directly from a Colab notebook!
- Gemini 3 is now available in Colab.
- In addition to being available in VS Code, Colab is also available for use in Antigravity, Cursor, and Windsurf via the Open VSX Registry!
- H100 is being rolled out for more users.
- Launched a new modern design for the Colab UI.

Python package upgrades

- accelerate 1.11.0 -> 1.12.0
- astropy 7.1.1 -> 7.2.0
- bigframes 2.28.0 -> 2.31.0
- cachetools 5.5.2 -> 6.2.4
- dataproc-spark-connect 0.8.3 -> 1.0.1
- diffusers 0.35.2 -> 0.36.0
- flax 0.10.7 -> 0.11.2
- google 2.0.3 -> 3.0.0
- google-adk 1.17.0 -> 1.21.0
- google-auth 2.38.0 -> 2.43.0
- google-genai 1.49.0 -> 1.55.0
- gradio 5.49.1 -> 5.50.0
- holidays 0.84 -> 0.88
- humanize 4.14.0 -> 4.15.0
- langchain 0.3.27 -> 1.2.4
- langsmith 0.4.42 -> 0.6.4

google colab - Yahoo India Se... x Untitled24.ipynb - Colab x New chat x | +

colab.research.google.com/drive/1DgOXDuikQ2eAcKdN5G3KKzIOvEBlyKOu#scrollTo=2y3QMMUF9U4

Untitled24.ipynb ☆

File Edit View Insert Runtime Tools Help

Q Commands + Code + Text ▶ Run all

```
[4] ✓ Os
def validate_marks(marks_list):
    """
    This function checks whether the marks list
    contains valid numeric values between 0 and 100.
    """
    if not marks_list:
        return False

    for mark in marks_list:
        if not isinstance(mark, (int, float)) or mark < 0 or mark > 100:
            return False

    return True

# Main Program
student_marks = [78, 85, 90, 66, 88]

if validate_marks(student_marks):
    total, average = calculate_total_and_average(student_marks)
    grade = calculate_grade(average)

    print("Total Marks:", total)
    print("Average Marks:", average)
    print("Grade:", grade)
else:
    print("Error: Invalid marks data provided.")
```

Release notes

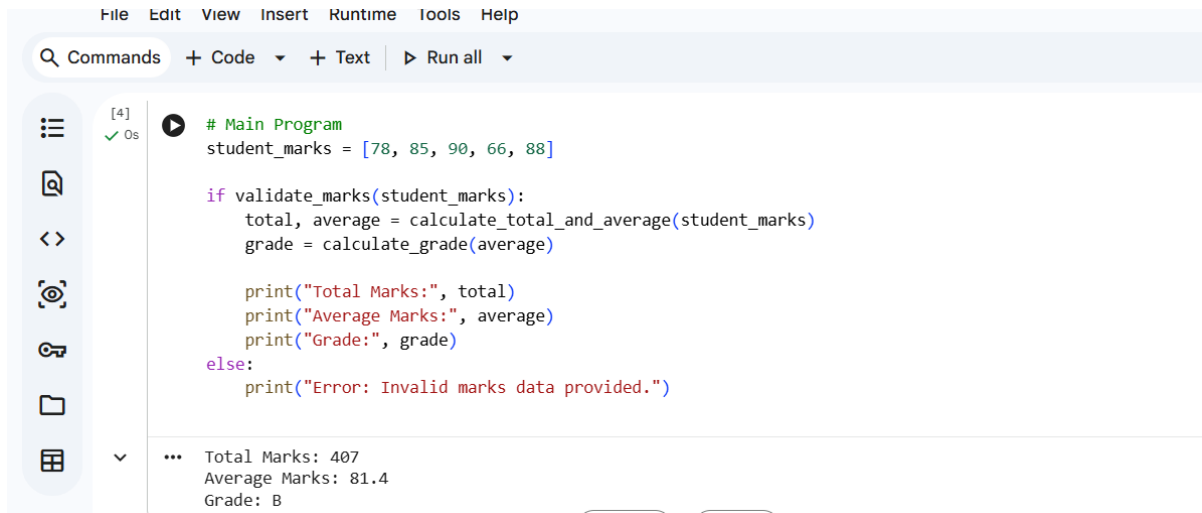
Please follow our [blog](#) to see more information about new features, tips and tricks, and featured notebooks such as [Analyzing a Bank Failure with Colab](#).

2026-01-20

- Launched Data Explorer - a new feature that lets you search Kaggle datasets, models, and competitions directly from a Colab notebook!
- Gemini 3 is now available in Colab.
- In addition to being available in VS Code, Colab is also available for use in Antigravity, Cursor, and Windsurf via the Open VSX Registry!
- H100 is being rolled out for more users.
- Launched a new modern design for the Colab UI.

Python package upgrades

- accelerate 1.11.0 -> 1.12.0
- astropy 7.1.1 -> 7.2.0
- bigframes 2.28.0 -> 2.31.0
- cachetools 5.5.2 -> 6.2.4
- dataproc-spark-connect 0.8.3 -> 1.0.1
- diffusers 0.35.2 -> 0.36.0
- flax 0.10.7 -> 0.11.2
- google 2.0.3 -> 3.0.0
- google-adk 1.17.0 -> 1.21.0
- google-auth 2.38.0 -> 2.43.0
- google-genai 1.49.0 -> 1.55.0
- gradio 5.49.1 -> 5.50.0
- holidays 0.84 -> 0.88
- humanize 4.14.0 -> 4.15.0
- langchain 0.3.27 -> 1.2.4
- langsmith 0.4.42 -> 0.6.4



```
File Edit View Insert Runtime Tools Help
Q Commands + Code + Text ▶ Run all
[4]
✓ 0s
# Main Program
student_marks = [78, 85, 90, 66, 88]

if validate_marks(student_marks):
    total, average = calculate_total_and_average(student_marks)
    grade = calculate_grade(average)

    print("Total Marks:", total)
    print("Average Marks:", average)
    print("Grade:", grade)
else:
    print("Error: Invalid marks data provided.")

... Total Marks: 407
Average Marks: 81.4
Grade: B
```

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

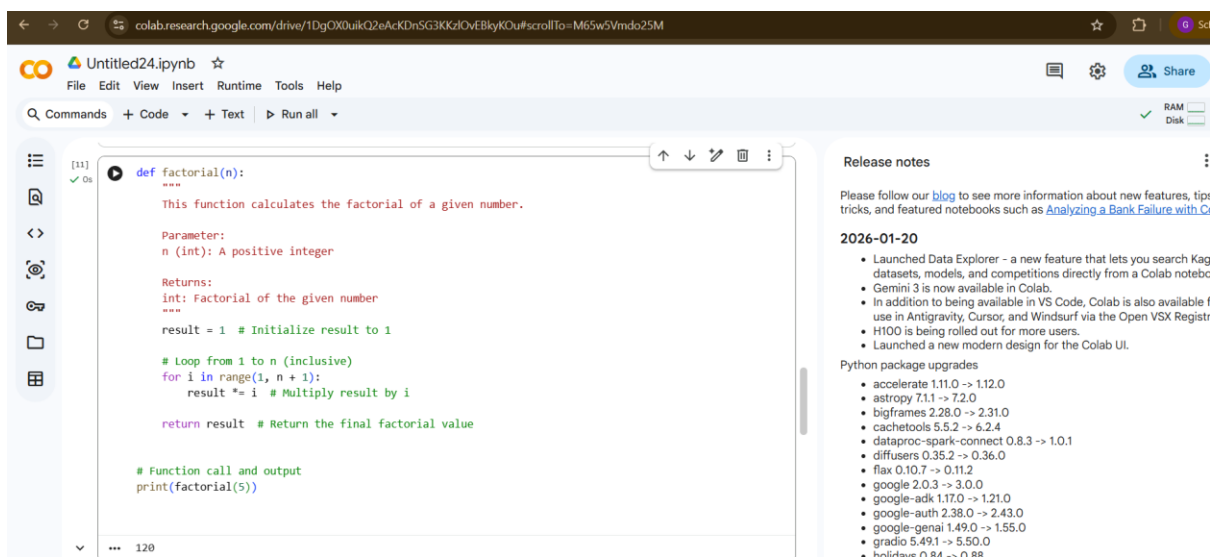
```
def factorial(n):
```

```
    result = 1
```

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

```
        result *= i
```

```
    return result
```



```
colab.research.google.com/drive/1DgOX0uikQ2eAckDnSG3KKzOvEBkyK0u#scrollTo=M65w5Vmdo25M
Untitled24.ipynb
File Edit View Insert Runtime Tools Help
Q Commands + Code + Text ▶ Run all
[11]
✓ 0s
def factorial(n):
    """
    This function calculates the factorial of a given number.

    Parameter:
    n (int): A positive integer

    Returns:
    int: Factorial of the given number
    """
    result = 1 # Initialize result to 1

    # Loop from 1 to n (inclusive)
    for i in range(1, n + 1):
        result *= i # Multiply result by i

    return result # Return the final factorial value

# Function call and output
print(factorial(5))

... 120
```

Release notes

Please follow our [blog](#) to see more information about new features, tips tricks, and featured notebooks such as [Analyzing a Bank Failure with Co](#)

2026-01-20

- Launched Data Explorer - a new feature that lets you search Kag datasets, models, and competitions directly from a Colab notebo
- Gemini 3 is now available in Colab.
- In addition to being available in VS Code, Colab is also available f use in Antigravity, Cursor, and Windsurf via the Open VSX Registr
- H100 is being rolled out for more users.
- Launched a new modern design for the Colab UI.

Python package upgrades

- accelerate 1.11.0 -> 1.12.0
- astropy 7.1.1 -> 7.2.0
- bigframes 2.28.0 -> 2.31.0
- cachetools 5.5.2 -> 6.2.4
- dataproc-spark-connect 0.8.3 -> 1.0.1
- diffusers 0.35.2 -> 0.36.0
- flax 0.10.7 -> 0.11.2
- google 2.0.3 -> 3.0.0
- google-adk 1.17.0 -> 1.21.0
- google-auth 2.38.0 -> 2.43.0
- google-gemini 1.49.0 -> 1.55.0
- gradio 5.49.1 -> 5.50.0
- holivis: 0.8.4 -> 0.88

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")
```

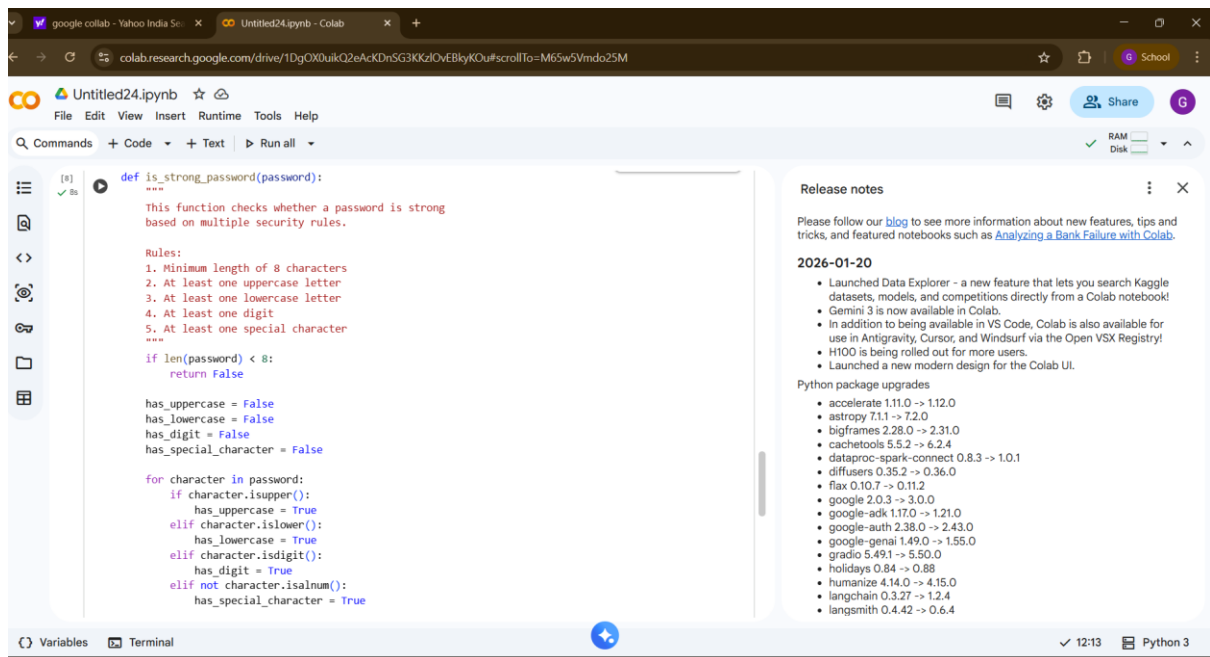
Task:

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

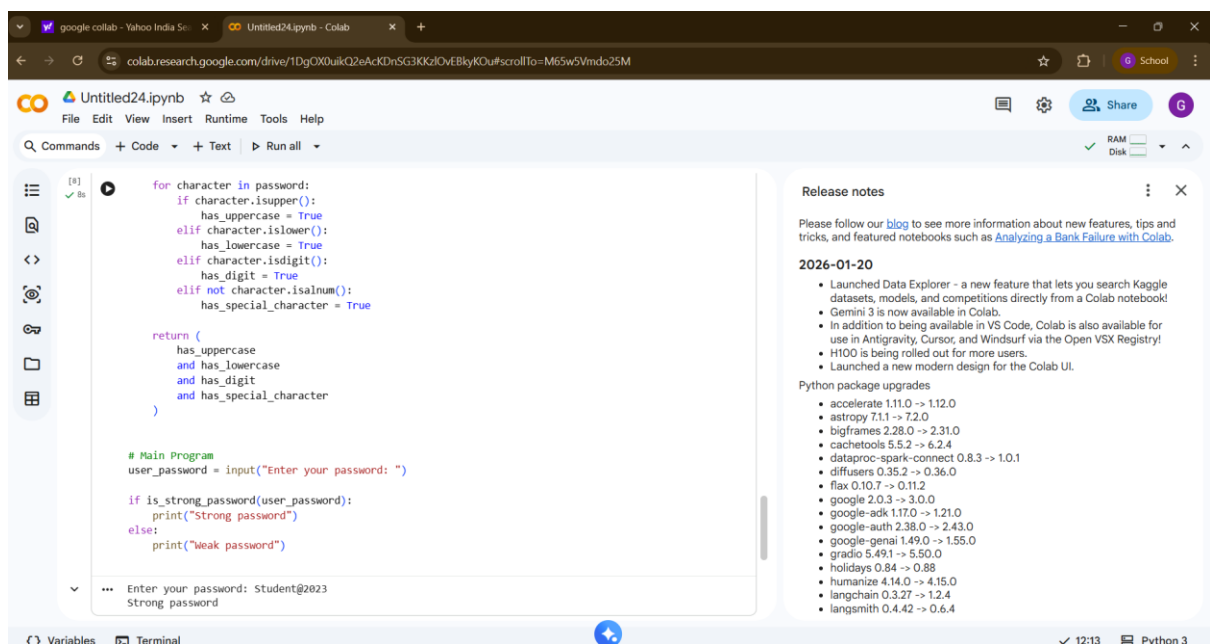
o Security strength and robustness

4. Justify the AI-generated changes, explaining why each added rule and refactoring decision improves the overall quality of the program.



The screenshot shows a Google Colab notebook titled 'Untitled24.ipynb'. The code defines a function `is_strong_password(password)` that checks a password against five rules: minimum length of 8 characters, at least one uppercase letter, at least one lowercase letter, at least one digit, and at least one special character. The function uses boolean flags (`has_uppercase`, `has_lowercase`, `has_digit`, `has_special_character`) and a loop to verify each rule. The right sidebar displays 'Release notes' and 'Python package upgrades'.

```
def is_strong_password(password):  
    """  
    This function checks whether a password is strong  
    based on multiple security rules.  
  
    Rules:  
    1. Minimum length of 8 characters  
    2. At least one uppercase letter  
    3. At least one lowercase letter  
    4. At least one digit  
    5. At least one special character  
    """  
    if len(password) < 8:  
        return False  
  
    has_uppercase = False  
    has_lowercase = False  
    has_digit = False  
    has_special_character = False  
  
    for character in password:  
        if character.isupper():  
            has_uppercase = True  
        elif character.islower():  
            has_lowercase = True  
        elif character.isdigit():  
            has_digit = True  
        elif not character.isalnum():  
            has_special_character = True
```



The screenshot shows the same Google Colab notebook after refactoring. The `is_strong_password` function is now more concise, using a single loop with `if`, `elif`, and `elif not` statements to set the boolean flags. Below the function, a main program prompts the user for a password and prints 'Strong password' or 'Weak password' based on the function's return value. The right sidebar remains the same.

```
for character in password:  
    if character.isupper():  
        has_uppercase = True  
    elif character.islower():  
        has_lowercase = True  
    elif character.isdigit():  
        has_digit = True  
    elif not character.isalnum():  
        has_special_character = True  
  
return (  
    has_uppercase  
    and has_lowercase  
    and has_digit  
    and has_special_character  
)  
  
# Main Program  
user_password = input("Enter your password: ")  
  
if is_strong_password(user_password):  
    print("Strong password")  
else:  
    print("Weak password")
```

Enter your password: Student@2023
Strong password