



AIMS

**African Institute for
Mathematical Sciences
RWANDA**

Python Programming Presentation

Check Password project

Grace KITONYI(Group Leader)

Musonda KATAI

Latifah AKIMANA

Tchandikou OUADJA FARE

October 8, 2025

- Problem Statement
- Project Objectives
- Password Validation rules
- Program Algorithm
- Conclusion
- References

Check a Password Project

The problem is to develop a Python program that validates passwords based on strict security rules. It must check length, character variety, and reject weak or common passwords. The program should provide feedback on why a password fails the rules.

Project Objectives

- The goal of this project was to develop a Python program that:
- Apply multiple security rules:
 - Length and character variety.
 - Avoid common weak passwords.
 - Avoid spaces
 - Prevent consecutive identical characters.
- Give clear feedback to users.

Password Validation rules

- ❶ Must be 8–64 characters long.
- ❷ Must include:
 - At least one uppercase letter
 - At least one lowercase letter
 - At least one digit
 - At least one special character
- ❸ Must not contain spaces.
- ❹ Must not be entirely alphabetic or numeric.
- ❺ Must not contain 3 or more identical consecutive characters.
- ❻ Must not be one of the commonly used insecure passwords such as: password, 123456, qwerty, letmein.

Program Algorithm - Password Length

1. **Start**
2. **Define the function** `check_password(password)`:
 - (a) Create an empty list `comments` to store feedback messages.
 - (b) Create a list `common_passwords` containing commonly used weak passwords.
 - (c) **Check password length:**
 - If the password length is less than 8 or greater than 64, add the comment:
"Password should be 8-64 characters long."

Program Algorithm - Character composition

- (d) Initialize five flags: `has_upper`, `has_lower`, `has_digit`, `has_special`, and `has_space`, all set to `False`.
- (e) Define a string `special_characters` containing symbols such as:
`!@#$%^&*()-_+={}[]:;<.>?/\|`.
- (f) **For each character in the password:**
 - If it is uppercase \rightarrow set `has_upper` = `True`.
 - Else if lowercase \rightarrow set `has_lower` = `True`.
 - Else if a digit \rightarrow set `has_digit` = `True`.
 - Else if in `special_characters` \rightarrow set `has_special` = `True`.
 - Else if it is a space \rightarrow set `has_space` = `True`.
- (g) **Check for missing requirements:**
 - If `has_upper` is `False` \rightarrow add comment: "Add at least one uppercase letter."
 - If `has_lower` is `False` \rightarrow add comment: "Add at least one lowercase letter."
 - If `has_digit` is `False` \rightarrow add comment: "Add at least one number."
 - If `has_special` is `False` \rightarrow add comment: "Add at least one special character."
 - If `has_space` is `True` \rightarrow add comment: "Password must not contain spaces."

(h) **Check if password is entirely alphabetic or numeric:**

- If password has only letters → add comment: "Password must not be entirely alphabetic."
- If password has only digits → add comment: "Password must not be entirely numeric."

(i) **Check for three identical consecutive characters:**

- Loop from the first to the third-last character.
- If `password[i] == password[i+1] == password[i+2]` → add comment: "Password must not contain 3 or more identical consecutive characters." Then stop the loop.

Program Algorithm - Common passwords

(j) **Check if password is common:**

- If the password matches any item in `common_passwords` → add comment:
"Password must not be a common password."

(k) **Decide password strength:**

- If comments is empty → set `is_good` = True.
- Else → set `is_good` = False.

(l) Return the tuple (`is_good`, `comments`).

3. Main Program Loop:

(a) Repeat:

- Ask the user to "Enter a password:".
- Call `check_password(password)` and get (`is_good`, `observations`).
- If `is_good` is True → print "Great, this is a strong password." and stop the loop.
- Otherwise →
 - Print "Oops, your password is weak.".
 - Display all comments from `observations`.
 - Prompt the user to enter another password.

4. End

Conclusion

- The program created successfully detects weak and strong passwords.
- It provides detailed comments to help the user create a strong password.
- This project strengthens understanding of:
 - String manipulation
 - Conditional logic
 - User input validation

- Common Passwords: <https://github.com/danielmiessler/SecLists/blob/master/Passwords/Common-Credentials/10k-most-common.txt>

THANK YOU FOR YOUR ATTENTION!

Any Questions Please?



Figure 1: Scan the QR code to check password.