Password Complexity Checker

By Pallabi Poria

Aim

Build a tool that assesses the strength of a password based on criteria such as length, presence of uppercase and lowercase letters, numbers, and special characters. Provide feedback to users on the password's strength.

Objective

- To create a tool that checks how strong or weak a password is based on multiple security parameters.
- To promote awareness of strong password creation practices among users.
- To use Python's Tkinter library to design an interactive and user-friendly interface.
- To enhance the basic internship task by implementing real-time feedback and GUI visualization.

Description

- This project, developed as part of the Prodigy InfoTech Internship (Cybersecurity Domain), is designed to check the strength of user passwords.
- It analyzes passwords based on various parameters such as length, character diversity, and avoidance of common patterns.
- Unlike basic command-line tools, this version uses Tkinter GUI for real-time password evaluation, making it visually appealing and interactive.
- The user receives a color-coded strength label (Weak, Moderate, or Strong) and detailed feedback on how to improve their password security.
- Additionally, a Show/Hide Password toggle improves usability and privacy.

Features

1. Real-Time Evaluation: Password strength updates automatically as the user types.
2. GUI Interface: Designed using Tkinter for better user interaction.
3. Feedback System: Displays one-line improvement tips for weak passwords.
4. Show/Hide Password Option: Allows users to toggle password visibility securely.
5. Color Indicators:
   - 🟥 Weak (Red)
   - 🟧 Moderate (Orange)
   - 🟩 Strong (Green)
6. Read-Only Feedback Box: Ensures users cannot modify system-generated suggestions.

Arithmetic/Logic

1. Input: The user enters a password.
2. Process:
   - Check length (≥8, ≥12 characters).

- Check for uppercase, lowercase, digits, and special characters.
- Detect common patterns like "password", "admin", "12345", etc.
- Calculate a score based on conditions met.

3. Output:
   - Display overall strength (Weak/Moderate/Strong).
   - Provide feedback tips for missing security elements.
   - Color-code the strength label for visual clarity.

Python Code

```python
import re
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox

# Function to evaluate password strength
def check_password_strength(password):
    score = 0
    feedback = []

    # Length
    if len(password) >= 12:
        score += 2
    elif len(password) >= 8:
        score += 1
    else:
        feedback.append("Use at least 8 characters (12+ is better).")

    # Uppercase
    if re.search(r"[A-Z]", password):
        score += 1
    else:
        feedback.append("Add at least one uppercase letter.")

    # Lowercase
    if re.search(r"[a-z]", password):
        score += 1
    else:
        feedback.append("Add at least one lowercase letter.")

    # Digit
    if re.search(r"[0-9]", password):
        score += 1
    else:
        feedback.append("Add at least one number.")

    # Special character
    if re.search(r"[!@#$%^&*(),.?\{}|<>]", password):
        score += 1
    else:
        feedback.append("Add at least one special character.")

    # Avoid common patterns
    common_patterns = ["password", "admin", "12345", "qwerty", "abc"]
    if any(word in password.lower() for word in common_patterns):
        feedback.append("Avoid common or easily guessed words.")
        score -= 1

    # Determine strength
    if score >= 6:
        strength = "Strong "
        color = "green"
    elif 3 <= score < 6:
        strength = "Moderate "
        color = "orange"
    else:
        strength = "Weak "
        color = "red"

    return strength, score, color, feedback

# Function triggered on typing
def update_strength(event=None):
    password = password_entry.get()
    strength, score, color, feedback = check_password_strength(password)
    strength_label.config(text=f"Strength: {strength} | Score: {max(0, score)}/6",
fg=color)
    feedback_text.config(state="normal")
    feedback_text.delete(1.0, tk.END)
    if feedback:
        for tip in feedback:
            feedback_text.insert(tk.END, f"{tip}\n " , "bold")
    else:
        feedback_text.insert(tk.END, " Great! Your password looks secure.",
"bold")
    feedback_text.config(state="disabled")
```

```python
# Show/Hide toggle function
def toggle_show_password():
    if password_entry.cget('show') == '':
        password_entry.config(show='*')
        show_button.config(text="Show")
    else:
        password_entry.config(show='')
        show_button.config(text="Hide")

# Tkinter Window Setup
root = tk.Tk()
root.title("Password Strength Checker")
root.geometry("500x400")
root.resizable(False, False)

# Title
title_label = tk.Label(root, text=" Secure Password Strength Checker",
font=("Arial", 14, "bold"))
title_label.pack(pady=10)

# Password Entry with show/hide toggle button
frame = tk.Frame(root)
frame.pack(pady=10, padx=20, fill='x')

password_entry = tk.Entry(frame, show="*", font=("Arial", 12))
password_entry.pack(side="left", fill='x', expand=True)
password_entry.bind("<KeyRelease>", update_strength)

show_button = tk.Button(frame, text="Show", command=toggle_show_password)
show_button.pack(side="left", padx=5)

# Strength Label
strength_label = tk.Label(root, text="Strength: ", font=("Arial", 12))
strength_label.pack(pady=10)

# Feedback Box (Read Only)
feedback_text = tk.Text(root, height=8, font=("Arial", 11), state="disabled")
feedback_text.tag_configure("bold", font=("Arial", 11, "bold") )
feedback_text.pack(pady=10, padx=20, fill='both')

root.mainloop()
```

Code Explanation

1. **Importing Required Libraries**
   The project begins by importing two key libraries:
   - re: Used for regular expressions to check if the password meets specific security patterns (uppercase, lowercase, digits, special characters).
   - tkinter: Used to create the graphical user interface (GUI) for an interactive user experience.
2. **Function: check_password_strength(password)**
   - This function takes a user-entered password as input and evaluates it based on five main parameters:
     - Length Check: Ensures the password is at least 8 characters long.
     - Uppercase Letter Check: Confirms if the password contains at least one uppercase letter (A–Z).
     - Lowercase Letter Check: Ensures presence of at least one lowercase letter (a–z).
     - Digit Check: Validates if there's at least one numeric character (0–9).
     - Special Character Check: Verifies if there's at least one symbol (like @, #, $, etc.).
   - For each fulfilled condition, a score is incremented.
   - Based on the final score, the function categorizes the password as:
     - Weak: If it fails to meet basic conditions.
     - Moderate: If it meets most conditions but lacks certain strength factors.
     - Strong: If all conditions are met, ensuring robust security.
   - Along with the strength level, the function provides suggestions to help users improve weak passwords.
3. **Function: update_strength(event)**
   - This function is bound to the password input field and triggers automatically whenever the user types something.
   - It retrieves the text entered by the user and passes it to the check_password_strength() function.
   - The evaluated strength and feedback are then displayed dynamically on the GUI:
     - The Strength Label changes color (Red, Orange, or Green) depending on password strength.
     - The Feedback Box gives actionable tips to make the password stronger.
   - This real-time feedback mechanism helps users learn password hygiene interactively.
4. **Function: toggle_show_password()**
   - Provides a button that allows users to switch between showing and hiding their entered password.

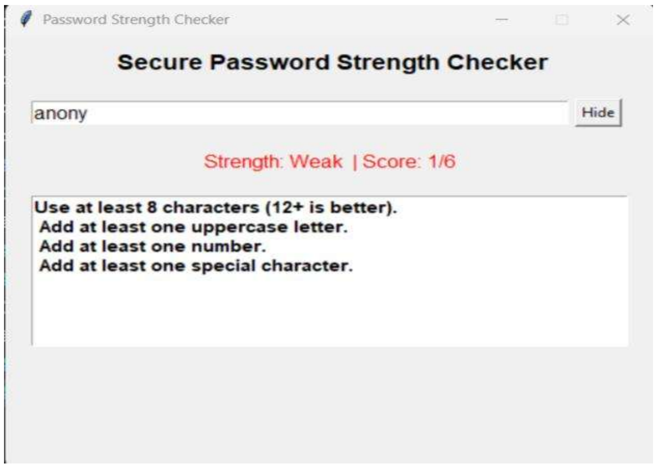
- If the password is hidden (using show="*"), clicking the button makes it visible — and vice versa.
- This feature enhances usability while maintaining user privacy.

5. **GUI Layout & Design:**
   - The GUI is created using Tkinter widgets arranged neatly for clarity and ease of use.
   - Key components include:
     - Label: Displays the application title.
     - Entry Field: For password input.
     - Button: For toggling password visibility.
     - Strength Label: Shows the evaluated strength (Weak/Moderate/Strong).
     - Text Widget: Displays suggestions to improve the password.
   - The interface is designed with intuitive color coding — Red for Weak, Orange for Moderate, and Green for Strong — providing instant visual feedback.

6. **Execution:**
   - Finally, the program calls root.mainloop() to start the Tkinter event loop.
   - This ensures the GUI remains responsive, allowing users to interact in real time as they test different passwords.

Sample Output:

Weak password:



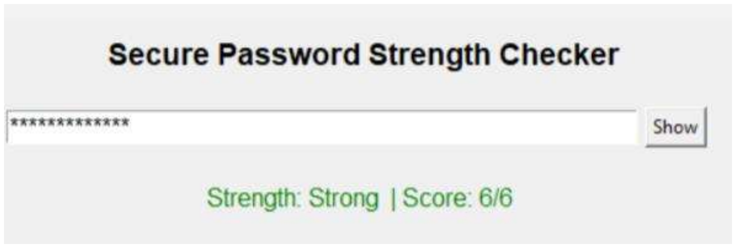
Secure Password Strength Checker

anony

Strength: Weak | Score: 1/6

```
Use at least 8 characters (12+ is better).
 Add at least one uppercase letter.
 Add at least one number.
 Add at least one special character.
```

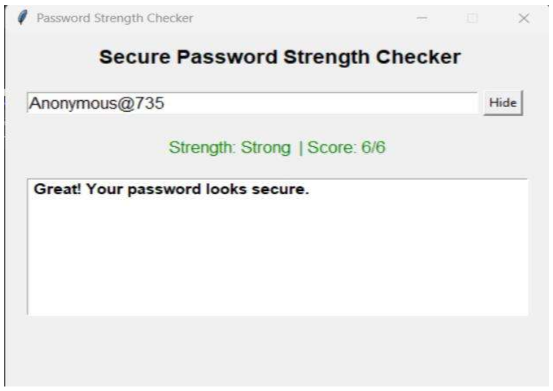
You can hide the password


Secure Password Strength Checker

****

Strength: Strong | Score: 6/6

Strong Password



Secure Password Strength Checker

Anonymous@735

Strength: Strong | Score: 6/6

Great! Your password looks secure.

Project Justification

- This project fulfills the internship requirement of developing a cybersecurity-related tool that helps users enhance their password hygiene.
- By converting it into a GUI version, it becomes more interactive and practical for real-world usage.
- It demonstrates strong understanding of Python programming, regex, and UI design principles — aligning perfectly with cybersecurity best practices.

Conclusion

- The project successfully evaluates password strength based on multiple security parameters and educates users about creating secure passwords.
- It showcases how Python and Tkinter can be used to develop interactive security tools, making password safety easy and accessible for everyone.

Scope

- Integrate a password generator feature to suggest strong passwords automatically.
- Add machine learning-based strength prediction using password datasets.
- Store and visualize user strength statistics for analysis.
- Convert the GUI tool into a web-based application for broader accessibility.