Password Strength Analyzer (GUI Version)

Author: Krish Patel

Date: September 2025

# Abstract

The Password Strength Analyzer is a Python-based GUI application built using Tkinter. It evaluates the strength of a password against multiple security criteria including length, use of uppercase letters, lowercase letters, digits, and special characters. The tool provides immediate feedback and suggestions to improve weak or moderate passwords, helping users adopt stronger password practices for better cybersecurity.

# Tools & Technologies Used

- Python 3.x

- Tkinter (for GUI)

- Regular Expressions (re library)

# System Description

The application consists of a Tkinter-based interface with an input field for entering a password, an 'Analyze' button, and a text area to display results. The password is evaluated on five main 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 (@, #, $, !, %, \*, ?, &)  
  
Based on the rules satisfied, the password is classified as Weak, Moderate, or Strong, with suggestions provided for improvement.

# Source Code

import re  
import tkinter as tk  
from tkinter import messagebox  
  
def check\_password\_strength(password):  
 strength = 0  
 remarks = []  
  
 if len(password) >= 8:  
 strength += 1  
 else:  
 remarks.append("❌ Password should be at least 8 characters long.")  
  
 if re.search(r"[A-Z]", password):  
 strength += 1  
 else:  
 remarks.append("❌ Add at least one uppercase letter.")  
  
 if re.search(r"[a-z]", password):  
 strength += 1  
 else:  
 remarks.append("❌ Add at least one lowercase letter.")  
  
 if re.search(r"[0-9]", password):  
 strength += 1  
 else:  
 remarks.append("❌ Add at least one number.")  
  
 if re.search(r"[@#$!%\*?&]", password):  
 strength += 1  
 else:  
 remarks.append("❌ Add at least one special character (@,#, $, !, %, \*, ?, &).")  
  
 if strength == 5:  
 return "✅ Strong Password", remarks  
 elif 3 <= strength < 5:  
 return "⚠️ Moderate Password", remarks  
 else:  
 return "❌ Weak Password", remarks  
  
def analyze\_password():  
 password = entry.get()  
 if not password:  
 messagebox.showwarning("Warning", "Please enter a password.")  
 return  
  
 result, feedback = check\_password\_strength(password)  
  
 output\_text.delete("1.0", tk.END)  
 output\_text.insert(tk.END, f"Password Strength: {result}\n\n")  
  
 if feedback:  
 output\_text.insert(tk.END, "Suggestions:\n")  
 for item in feedback:  
 output\_text.insert(tk.END, f"- {item}\n")  
  
root = tk.Tk()  
root.title("Password Strength Analyzer")  
root.geometry("500x400")  
root.config(bg="#f0f4f7")  
  
heading = tk.Label(root, text="🔐 Password Strength Analyzer",  
 font=("Arial", 16, "bold"), bg="#f0f4f7", fg="#2c3e50")  
heading.pack(pady=10)  
  
entry\_label = tk.Label(root, text="Enter Password:", font=("Arial", 12), bg="#f0f4f7")  
entry\_label.pack(pady=5)  
  
entry = tk.Entry(root, show="\*", width=30, font=("Arial", 12))  
entry.pack(pady=5)  
  
analyze\_btn = tk.Button(root, text="Analyze", command=analyze\_password,  
 font=("Arial", 12, "bold"), bg="#3498db", fg="white", padx=10, pady=5)  
analyze\_btn.pack(pady=10)  
  
output\_text = tk.Text(root, height=10, width=55, font=("Arial", 10))  
output\_text.pack(pady=10)  
  
root.mainloop()

# Sample Output

Example 1: Password = 'abc' → Weak Password (Suggestions: add uppercase, digit, special character).

Example 2: Password = 'Abc1234' → Moderate Password (Suggestion: add special character).

Example 3: Password = 'Abc123!@' → Strong Password.

# Conclusion

This project demonstrates the importance of password security and helps users understand how to create stronger passwords. The tool is lightweight, user-friendly, and can be extended with advanced features such as color coding, show/hide password, and saving results to a file. It serves as an excellent mini-project for learning both cybersecurity basics and Python GUI development.