import re

def calculate\_password\_strength(password, common\_passwords=[]):

"""

Evaluate the strength of a password and return a score with detailed feedback.

"""

score = 0

feedback = []

# Length check

if len(password) < 6:

feedback.append("Password is too short (less than 6 characters).")

return score, feedback

elif len(password) >= 6 and len(password) < 8:

score += 1

feedback.append("Password length is okay, but longer passwords are stronger.")

elif len(password) >= 8 and len(password) < 12:

score += 2

feedback.append("Password length is good.")

elif len(password) >= 12:

score += 3

feedback.append("Password length is excellent.")

# Lowercase letters check

if re.search(r'[a-z]', password):

score += 1

feedback.append("Includes lowercase letters."

else:

feedback.append("Consider adding lowercase letters.")

# Uppercase letters check

if re.search(r'[A-Z]', password):

score += 1

feedback.append("Includes uppercase letters.")

else:

feedback.append("Consider adding uppercase letters.")

# Numbers check

if re.search(r'\d', password):

score += 1

feedback.append("Includes numbers.")

else:

feedback.append("Consider adding numbers.")

# Special characters check

if re.search(r'[!@#$%^&\*(),.?":{}|<>]', password):

score += 1

feedback.append("Includes special characters.")

else:

feedback.append("Consider adding special characters."

# Consecutive character penalties (like "aaaa", "1111")

if re.search(r'(.)\1{2,}', password):

score -= 1

feedback.append("Avoid repeated consecutive characters.")

# Sequential character penalties (like "123", "abc")

If re.search(r'(012|123|234|345|456|567|678|789|890|abc|bcd|cde|def|efg|fgh|ghi|hij|ijk|jkl|klm|lmn|mno|nop|opq|pqr|qrs|rst|stu|tuv|uvw|vwx|wxy|xyz)', password, re.IGNORECASE):

score -= 1

feedback.append("Avoid sequential characters.")

# Common passwords check

if password.lower() in common\_passwords:

feedback.append("This password is too common and easily guessable.")

return 0, feedback

return score, feedback

def categorize\_password\_strength(score):

"""

Categorize password strength based on its score.

"""

if score <= 1:

return "Very Weak"

elif score <= 3:

return "Weak"

elif score <= 5:

return "Good"

elif score <= 7:

return "Strong"

else:

return "Very Strong"

def load\_common\_passwords(file\_path):

"""

Load a list of common passwords from a file.

"""

try:

with open(file\_path, 'r') as file:

return [line.strip() for line in file]

except FileNotFoundError:

print(f"Warning: {file\_path} not found. Continuing without custom common passwords.")

return []

def log\_results(password, strength\_category, feedback):

"""

Log the results of the password strength check to a file.

"""

with open("password\_strength\_log.txt", "a") as log\_file:

log\_file.write(f"Password: {password}\n")

log\_file.write(f"Strength: {strength\_category}\n")

log\_file.write("Feedback:\n")

for suggestion in feedback:

log\_file.write(f"- {suggestion}\n")

log\_file.write("\n")

def main():

"""

Main function to interact with the user and provide password strength feedback.

"""

print("Welcome to the Password Strength Checker!")

print("Type 'exit' to quit the program.")

# Load common passwords from a file

common\_passwords = load\_common\_passwords("most-common-pass.txt")

while True:

password = input("\nEnter a password to check its strength: ")

if password.lower() == "exit":

print("Exiting the Password Strength Checker. Stay safe online!")

break

# Evaluate password strength

score, feedback = calculate\_password\_strength(password, common\_passwords)

strength\_category = categorize\_password\_strength(score)

# Display results

print(f"\nPassword Strength: {strength\_category}")

print("Feedback:")

for suggestion in feedback:

print(f"- {suggestion}")

# Log the results

log\_results(password, strength\_category, feedback)

if \_\_name\_\_ == "\_\_main\_\_":

main()