Task-03: Password Complexity Checker (Prodigy InfoTech Internship)

Name: Abhinandan Kumar
Role: Cyber Security Intern
Task Objective:
To build a Python tool that evaluates password strength based on length, character types, patterns, and common passwords.

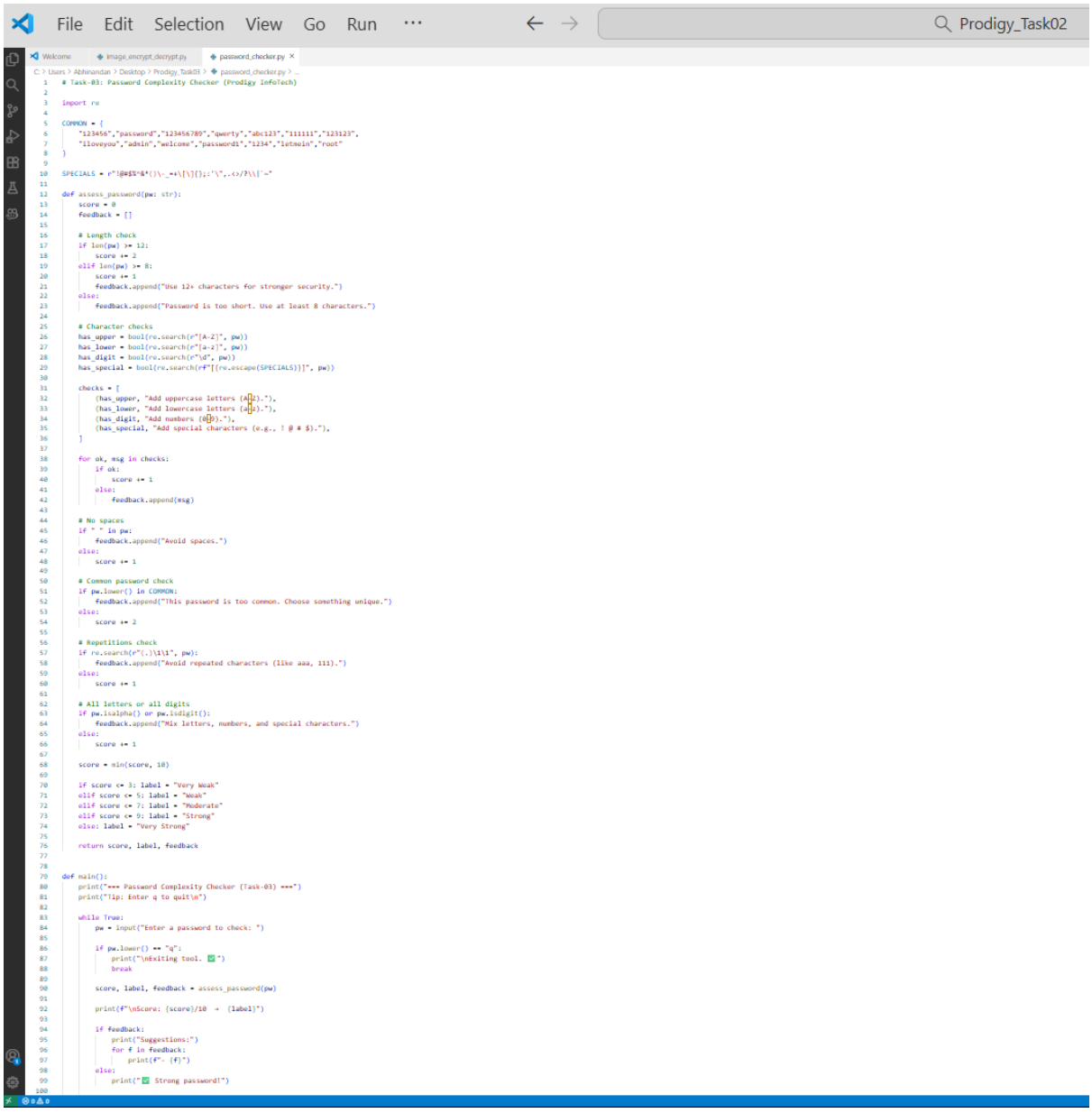
Key Criteria Checked:

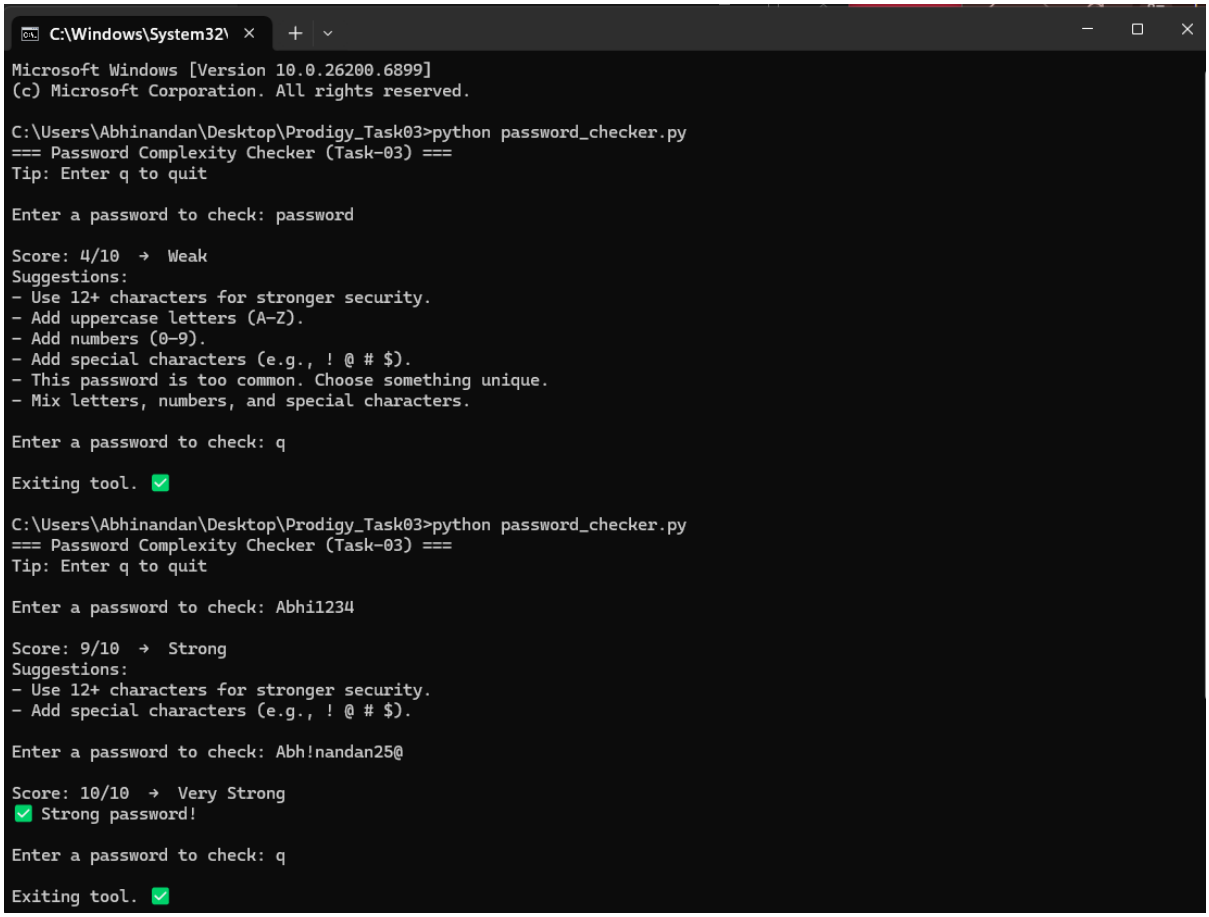
- Minimum length
- Uppercase letters
- Lowercase letters
- Numbers
- Special characters
- No spaces
- Avoid common passwords
- Avoid repeated characters
- Must mix character types

Tools Used:

- Python 3

Output:


File Edit Selection View Go Run ···
Prodigy_Task02
Welcome
image_encrypt_decrypt.py
password_checker.py
# Task-03: Password Complexity Checker (Prodigy InfoTech)
import re
COMMON = {
"123456","password","123456789","qwerty","abc123","111111","123123",
"iloveyou","admin","welcome","password1","1234","letmein","root"
}
SPECIALS = r"!@#$%^&*()\-_=+\[\]{};:'\",.<>/?\\|`~"
def assess_password(pw: str):
score = 0
feedback = []
# Length check
if len(pw) >= 12:
score += 2
elif len(pw) >= 8:
score += 1
feedback.append("Use 12+ characters for stronger security.")
else:
feedback.append("Password is too short. Use at least 8 characters.")
# Character checks
has_upper = bool(re.search(r"[A-Z]", pw))
has_lower = bool(re.search(r"[a-z]", pw))
has_digit = bool(re.search(r"\d", pw))
has_special = bool(re.search(rf"[{re.escape(SPECIALS)}]", pw))
checks = [
(has_upper, "Add uppercase letters (A–Z)."),
(has_lower, "Add lowercase letters (a–z)."),
(has_digit, "Add numbers (0–9)."),
(has_special, "Add special characters (e.g., ! @ # $)."),
]
for ok, msg in checks:
if ok:
score += 1
else:
feedback.append(msg)
# No spaces
if " " in pw:
feedback.append("Avoid spaces.")
else:
score += 1
# Common password check
if pw.lower() in COMMON:
feedback.append("This password is too common. Choose something unique.")
else:
score += 2
# Repetitions check
if re.search(r"(.)\1\1", pw):
feedback.append("Avoid repeated characters (like aaa, 111).")
else:
score += 1
# All letters or all digits
if pw.isalpha() or pw.isdigit():
feedback.append("Mix letters, numbers, and special characters.")
else:
score += 1
score = min(score, 10)
if score <= 3: label = "Very Weak"
elif score <= 5: label = "Weak"
elif score <= 7: label = "Moderate"
elif score <= 9: label = "Strong"
else: label = "Very Strong"
return score, label, feedback
def main():
print("=== Password Complexity Checker (Task-03) ===")
print("Tip: Enter q to quit\n")
while True:
pw = input("Enter a password to check: ")
if pw.lower() == "q":
print("\nExiting tool. ✅")
break
score, label, feedback = assess_password(pw)
print(f"\nScore: {score}/10 → {label}")
if feedback:
print("Suggestions:")
for f in feedback:
print(f"- {f}")
else:
print("✅ Strong password!")



Conclusion:

Successfully developed a password complexity tool that rates security and gives improvement tips. This tool helps users create strong passwords for better security.