

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

image_encrypt_decrypt.py password_checker.py x
C:\Users\Abhinandan > Desktop > Prodigy_Task02 > password_checker.py > ...
1 # Task-03: Password Complexity Checker (Prodigy InfoTech)
2
3 import re
4
5 COMMON = [
6     "123456", "password", "123456789", "qwerty", "abc123", "111111", "123123",
7     "iloveyou", "admin", "welcome", "password1", "1234", "1qaz!@WSX", "root"
8 ]
9
10 SPECIALS = r"[!@#$%^&*(){}~`-+=~\|;:'",.</P>\|`~]"
11
12 def assess_password(pw: str):
13     score = 0
14     feedback = []
15
16     # Length check
17     if len(pw) >= 12:
18         score += 2
19     elif len(pw) >= 8:
20         score += 1
21     feedback.append("Use 12+ characters for stronger security.")
22     else:
23         feedback.append("Password is too short. Use at least 8 characters.")
24
25     # Character checks
26     has_upper = bool(re.search(r"[A-Z]", pw))
27     has_lower = bool(re.search(r"[a-z]", pw))
28     has_digit = bool(re.search(r"[0-9]", pw))
29     has_special = bool(re.search(r"[^a-zA-Z0-9]", pw))
30
31     checks = [
32         (has_upper, "Add uppercase letters (A-Z)."),
33         (has_lower, "Add lowercase letters (a-z)."),
34         (has_digit, "Add numbers (0-9)."),
35         (has_special, "Add special characters (e.g., ! @ # $ %)."),
36     ]
37
38     for ok, msg in checks:
39         if ok:
40             score += 1
41         else:
42             feedback.append(msg)
43
44     # No spaces
45     if " " in pw:
46         feedback.append("Avoid spaces.")
47     else:
48         score += 1
49
50     # Common password check
51     if pw.lower() in COMMON:
52         feedback.append("This password is too common. Choose something unique.")
53     else:
54         score += 2
55
56     # Repetitions check
57     if re.search(r"(\w)\1+", pw):
58         feedback.append("Avoid repeated characters (like aaa, 111).")
59     else:
60         score += 1
61
62     # All letters or all digits
63     if pw.isalpha() or pw.isdigit():
64         feedback.append("Mix letters, numbers, and special characters.")
65     else:
66         score += 1
67
68     score = min(score, 10)
69
70     if score <= 3: label = "Very Weak"
71     elif score <= 5: label = "Weak"
72     elif score <= 7: label = "Moderate"
73     elif score <= 9: label = "Strong"
74     else: label = "Very Strong"
75
76     return score, label, feedback
77
78 def main():
79     print("=== Password Complexity Checker (Task-03) ===")
80     print("Tip: Enter q to quit\n")
81     while True:
82         pw = input("Enter a password to check: ")
83
84         if pw.lower() == "q":
85             print("Exiting tool. 🚪")
86             break
87
88         score, label, feedback = assess_password(pw)
89
90         print(f"Score: {score}/10 - {label}")
91
92         if feedback:
93             print("Suggestions:")
94             for f in feedback:
95                 print(f"  {f}")
96         else:
97             print("👍 Strong password!")
98
99
100
```

```
C:\Windows\System32\ x + v
Microsoft Windows [Version 10.0.26200.6899]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Abhinandan\Desktop\Prodigy_Task03>python password_checker.py
=== Password Complexity Checker (Task-03) ===
Tip: Enter q to quit

Enter a password to check: password

Score: 4/10 → Weak
Suggestions:
- Use 12+ characters for stronger security.
- Add uppercase letters (A-Z).
- Add numbers (0-9).
- Add special characters (e.g., ! @ # $).
- This password is too common. Choose something unique.
- Mix letters, numbers, and special characters.

Enter a password to check: q

Exiting tool. ✔

C:\Users\Abhinandan\Desktop\Prodigy_Task03>python password_checker.py
=== Password Complexity Checker (Task-03) ===
Tip: Enter q to quit

Enter a password to check: Abhi1234

Score: 9/10 → Strong
Suggestions:
- Use 12+ characters for stronger security.
- Add special characters (e.g., ! @ # $).

Enter a password to check: Abh!nandan25@

Score: 10/10 → Very Strong
✔ Strong password!

Enter a password to check: q

Exiting tool. ✔
```

Conclusion:

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