Passwort

> Diese Aufgabe ist eine abgewandelte Version des Password Practice Problems von CS50.
>
> **Disclaimer:** Diese Aufgabe wurde nicht vom Lehrstuhl herausgegeben und kann Fehler enthalten. Sie dient lediglich zu Übungszwecken!

Wie wir inzwischen alle wissen, ist es wichtig, Passwörter zu verwenden, die nicht leicht zu erraten sind! Viele Webanwendungen verlangen inzwischen Passwörter, die nicht nur alphabetische Zeichen, sondern auch Zahlen und Symbole enthalten.

Aufgabe

Diese Übung ist in fünf Teilaufgaben unterteilt, mit denen Sie Ihr Programm Schritt für Schritt aufbauen und verbessern können.

Teilaufgabe 1

In dieser Aufgabe soll der Benutzer mit der Nachricht `Enter your password:` zur Eingabe eines Passwortes aufgefordert werden. Dieses soll dann mit Hilfe der Funktion `valid` validiert werden. Wenn das Passwort mindestens einen Großbuchstaben, einen Kleinbuchstaben, eine Zahl und ein Symbol (d.h. ein druckbares Zeichen, das kein Buchstabe oder eine Zahl ist, wie z.B. `!`, `$` und `#`) enthält, sollte die Funktion `true` zurückgeben. Andernfalls sollte sie `false` zurückgeben.

Der Benutzer sollte natürlich über den Rückgabewert der Funktion `valid` informiert werden. Wenn das Passwort den Kriterien entspricht, sollte die Konsole `Your password is valid!` ausgeben. Entspricht das Passwort nicht den Kriterien, sollte `Your password needs at least one uppercase letter, lowercase letter, number and symbol.` ausgegeben werden.

Teilaufgabe 2

Ändern Sie das Programm aus Teilaufgabe 1 so ab, dass die Eingabe des Benutzers auch als Kommandozeilenargument erfolgen kann. Wird kein Argument übergeben, soll der Benutzer wie in Teilaufgabe 1 zur Eingabe eines Passwortes aufgefordert werden.

Somit sollte folgender Aufruf des Programms auch funktionieren und das Passwort überprüfen.

```
./password 5uper_s3cret
```

Teilaufgabe 3

Auch wenn das Passwort `!1Ab` alle bisherigen Richtlinien erfüllt, ist es nicht besonders sicher. Deshalb sollen Sie in dieser Aufgabe das Programm aus Teilaufgabe 2 so anpassen, dass ein Passwort mindestens `10` Zeichen lang sein muss. Passen Sie auch die Fehlermeldung für den Benutzer entsprechend an.

Teilaufgabe 4

Obwohl das Programm nun Passwörter wie `!1Abc` ablehnt, akzeptiert es immer noch Passwörter wie `!!11AAbbcc`, die nicht viel sicherer sind als das erste. Passen Sie nun die Funktion `valid` so an, dass auch keine Passwörter mehr akzeptiert werden, in denen die gleichen Zeichen aufeinander folgen. Passen Sie auch die Fehlermeldung entsprechend an.

Teilaufgabe 5

Obwohl das Programm auch Passwörter wie `123456` oder `password` ablehnt, ist es besser, den Benutzer darauf hinzuweisen, dass es sich bei seiner Wahl um eines der 100 häufigsten Passwörter handelt. Solche Passwörter sollten schließlich auch nicht erlaubt werden.

Implementieren Sie daher eine Funktion `is_common_password`, die prüft, ob ein übergebenes Passwort in der Liste der *10 häufigsten Passwörter* vorkommt. Auch hier sollte der Benutzer über das Ergebnis der Prüfung informiert werden. Überlegen Sie sich eine sinnvolle Nachricht und finden Sie eine geeignete Stelle, um diese Prüfung in Ihr Programm zu integrieren.

Testen

> ℹ️ Sie können auch `check50` zum Testen verwenden, siehe Befehlszeile weiter unten.

Ihr Programm sollte sich, je nach Teilaufgabe, wie in den folgenden Beispielen verhalten.

Eingabe	Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5
hello	❌	❌	❌	❌	❌
H3!lo	✅	✅	❌	❌	❌
Pas123456!	✅	✅	✅	✅	✅
P@ssw0rd	✅	✅	❌	❌	❌
1234abcd	❌	❌	❌	❌	❌
!@#ABC123def	✅	✅	✅	✅	✅
1111aAaa!!!!	✅	✅	✅	❌	❌
QwErTy123!@	✅	✅	✅	✅	✅
!!AAaa11bb	✅	✅	✅	❌	❌

Eingabe	Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5
AbC!123xyz@	✅	✅	✅	✅	✅
admin	❌	❌	❌	❌	❌
letMein123!	✅	✅	✅	✅	✅
pas5word!23A	✅	✅	✅	✅	✅
abcDE!ghi1234	✅	✅	✅	✅	✅
P@$W0rD12345	✅	✅	✅	✅	✅
ABCabc123	❌	❌	❌	❌	❌
Abc@1233Abc_	✅	✅	✅	❌	❌
12abc!XYZ	✅	✅	❌	❌	❌
mySecret2021!	✅	✅	✅	✅	✅
qwerty!@123ABC	✅	✅	✅	✅	✅
dragon!@123ABC	✅	✅	✅	✅	✅
Hello123!!	✅	✅	✅	❌	❌
Zyx!9876lmNOP	✅	✅	✅	✅	✅
Test@123	✅	✅	❌	❌	❌
R@nd0mPasw0rd	✅	✅	✅	✅	✅
$up3r$trongP@s5	✅	✅	✅	✅	✅
abc123def!	❌	❌	❌	❌	❌
123456	🚫	🚫	🚫	🚫	❌
password	🚫	🚫	🚫	🚫	❌
12345678	🚫	🚫	🚫	🚫	❌
qwerty	🚫	🚫	🚫	🚫	❌
123456789	🚫	🚫	🚫	🚫	❌
12345	🚫	🚫	🚫	🚫	❌
1234	🚫	🚫	🚫	🚫	❌
111111	🚫	🚫	🚫	🚫	❌
1234567	🚫	🚫	🚫	🚫	❌
dragon	🚫	🚫	🚫	🚫	❌

✅ = Passwort akzeptiert; ❌ = Passwort nicht akzeptiert; 🚫 = Test nicht verfügbar

Befehlszeile

Verwenden Sie dazu je nach Teilaufgabe die folgenden Befehle, um Ihr Programm mit `check50` zu überprüfen.

- **Teilaufgabe 1:** `check50 -l Kuschel-Swein/inf-einf-labs/main/password/check/a1`
- **Teilaufgabe 2:** `check50 -l Kuschel-Swein/inf-einf-labs/main/password/check/a2`
- **Teilaufgabe 3:** `check50 -l Kuschel-Swein/inf-einf-labs/main/password/check/a3`
- **Teilaufgabe 4:** `check50 -l Kuschel-Swein/inf-einf-labs/main/password/check/a4`
- **Teilaufgabe 5:** `check50 -l Kuschel-Swein/inf-einf-labs/main/password/check/a5`

Webbrowser

Um das Testen interaktiver zu gestalten, können Sie das mitgelieferte Skript `server.py` verwenden. Verwenden Sie dazu `python3 server.py` nachdem Sie Ihr Programm mit `make password` kompiliert haben. Es sollte sich nun automatisch eine Webseite öffnen, auf der Sie verschiedene Passwörter zum Testen in das Eingabefeld eingeben können. Um das Programm wieder zu verlassen, drücken Sie `Strg` + `C` in dem Terminal, in dem Sie das Skript gestartet haben.

> ℹ️ Werfen Sie gerne einen Blick in das Python Skript nachdem wir in der Vorlesung Python erreicht haben!

Style

Führen Sie den folgenden Befehl aus, um den Stil Ihres Codes mit `style50` zu analysieren:

```
style50 password.c
```