Lösungsidee

Für diese Aufgabe müssen mehrere zufällig generierte Zeichen aneinander gereiht werden, um ein einzigartiges Passwort zu erzeugen. Dass das Programm einfach bedient werden kann, soll es Anfangs die gewünschte Länge des Passwortes aufnehmen und ein Passwort dieser Länge generieren.

Zudem soll das generierte Passwort mehreren Regeln befolgen, dass es einfacher zu merken ist. Außer der vorgegebenen Regel, dass es keine Doppelt-Buchstaben enthalten soll, befolgt es auch vier weiteren Regeln.

Die erste Regel ist, dass der erste Buchstabe des Passworts die Chance hat, ein Großbuchstabe zu sein. Das muss aber nicht zwingend der Fall sein, da es im deutschen auch sowohl groß- als auch kleingeschriebene Wörter gibt. Außerdem ist die Sicherheit erhöht, da es anstatt den jeweils 26 Möglichkeiten, wenn es nur groß- der Kleinbuchstaben wären, mit dem Mix 52 Möglichkeiten gibt.

Die zweite Regel ist simpel und orientiert sich auch an dem Wortbau. Alle Buchstaben des Passworts, ausgeschlossen dem Ersten, sind Kleinbuchstaben. Da man sich hier am deutschen Wortbau orientieren kann, lässt es sich das Passwort einfacher merken.

Die dritte Regel orientiert sich an den Vorschlägen von Nicknamen von verschiedenen Webseiten. Bspw. „maxmuster13“. In diesem Format bestehen die letzten Zeichen aus Zahlen, was in dem Fall von sozialen Netzwerken mehrere Nutzer mit ähnlichem Nutzernamen ermöglicht. Solch eine ähnliche Regel wurde ebenso implementiert. Wegen der implementierten Regel bestehen das 3. und 2. letzte Zeichen nur aus Zahlen.

Die vierte Regel orientiert sich am deutschen Satzbau. Das letzte Zeichen des Passworts ist entweder ein Punkt, Fragezeichen, Ausrufezeichen oder auch ein Hashtag. Auch das sorgt für ein einfaches aber sicheres Passwort.

Umsetzung

Die Lösungsidee wird in Python umgesetzt. Die random-Library ist der Kernbestandteil des Generators. Diese Bibliothek wird zuerst importiert.

Anfangs soll es eine Eingabeaufforderung geben, wobei man die Länge des Passworts eingeben kann. Der Wert muss mindestens 8 sein, sollte es weniger als 8 sein, erscheint die Eingabeaufforderung aufgrund der While-Schleife erneut. Der Wert wird dann in noch zu generierenden Zeichen gespeichert und in der Generator-schleife Schrittweise verringert.

Der Zeichengenerator selbst wird zu einer eigenen Funktion zusammengefasst. Die Inputs sind boolean Argumente für Großbuchstaben, Zahlen und Sonderzeichen. Anhand der Argumente wird ein passendes Passwort generiert. Der Standard String aus möglichen Zeichen beinhaltet alle Buchstaben von a bis z. Wenn das Großbuchstaben Argument True ist, dann werden an diesen String alle Buchstaben von A bis Z angefügt. Wenn aber das Zahlen- oder Sonderzeichenargument True ist, so wird der komplette String durch Zahlen von 0 bis 9 oder den Zeichen „!?.#“ ersetzt. Der Zahlengenerator erzeugt dann eine Zahl zwischen 0 und der Anzahl an Zeichen im String -1, sodass dann aus dem String das Zeichen mit dem Index des zufällig generierten Wertes ausgegeben wird:

```
def generatepwd(caps,numbers,extra):
    given_chars = "abcdefghijklmnopqrstuvwxyz"
    if(caps):
        given_chars += "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    if(numbers):
        given_chars = "1234567890"
    if(extra):
        given_chars = "!?.#"
    random = randint(0,len(given_chars)-1)
    return(given_chars[random])
```

Der Hauptteil des Programms ist eine While-Schleife, welche nach jedem erfolgreich generiertem Zeichen, die Zahl an noch zu generierenden Zeichen um 1 verringert, also bis auf 0 runterzählt. Innerhalb der Schleife werden alle Regeln angewendet und entsprechend Argumente zu dem Zeichengenerator übergeben. Sobald alle Regeln angewendet wurden und ein Zeichen generiert wurde, wird überprüft ob das neu generierte Zeichen dem letzten generierten Zeichen entspricht, womit man Zeichendopplungen vermeidet. Sollte das nicht der Fall sein, wird das neu generierte Zeichen zum vorherigen Zeichen, das Zeichen wird dem Passwort angehängt und der „noch zu generieren“ Zähler wird um 1 verringert. Sollte ein Zeichen bereits zuvor verwendet worden sein, wird der Zähler nicht verringert und so lange ein Zeichen generiert, bis es nicht mehr dasselbe ist:

```
while not (wished_chars==0):
    if(wished_chars<=3 and wished_chars>1):
        char = generatepwd(False,True,False)
    elif(last_char == 0):
        char = generatepwd(True,False,False)
    elif(wished_chars==1):
        char = generatepwd(False,False,True)
    else:
        char = generatepwd(False,False,False)
    if(last_char != char):
        last_char = char
        password = password + char
        wished_chars -= 1
```

Am Ende des Programms wird das Passwort an den Nutzer ausgegeben und zuvor in als „Generiertes Passwort:“ gekennzeichnet.

Beispiele

Um ein Passwort zu generieren, welches den Regeln befolgt, starten wir nun das Programm.

python Junioraufgabe1.py

Es erscheint dann eine Eingabeaufforderung:

Anzahl an Zeichen des Passworts (min.8):

Nun gibt man eine Zahl von mindestens 8 ein und es wird zufällig ein Passwort generiert, welches allen Regeln befolgt. Hier ein Beispiel für ein achtstelliges Passwort:

Generiertes Passwort:

Kplrk56!

Das Passwort ist sicher und lässt sich einfach merken.

Beispiel für ein 12 stelliges Passwort:

Generiertes Passwort:

kfmscuwjm62.

Hier ist das erste Zeichen ein Kleinbuchstabe, wie es durch Regel 1 vorgesehen war.