Aufgabe 1: Passwörter

Valentin Stephan Zwerschke

22. November 2020

Inhaltsverzeichnis

1 Lösungsidee

Prinzipiell sollen meine Passwörter aus einer zufälligen Folge von Zeichen bestehen. Zwecks leichterer Lesbarkeit und Merkbarkeit des Passworts habe ich einige Regeln definiert, wie die Zeichen aufeinander folgen dürfen. Generell lasse ich auf eine Vokalfolge eine Konsonantenfolge folgen und umgekehrt. Dabei definiere ich Konsonantenfolgen als entweder einen einfachen Konsonanten, einen Doppelkonsonanten (wie 'pp') oder eine im Deutschen gebräuchliche Konsonantenkombination (wie 'sch' oder 'ch'). Als Vokalfolge definiere ich entsprechend alle Vokale, Umlaute (in der Schreibweise mit nachgestelltem 'e', aber auch 'eu' oder 'ei') und gedehnte Vokale (Doppelvokal wie 'ee' oder Vokal mit nachgestelltem 'h'). Zudem sollen im Passwort mit kleiner Wahrscheinlichkeit (ich defieniere diese als 15%) an beleibiger Stelle auch Ziffern auftreten. Am Wortanfang sowie nach Ziffern 1-9 nutze ich mit hoher Wahrscheinlichkeit (75%) Großschreibung, innerhalb des Passworts nur Kleinschreibung. Die Auswahl der verschiedenen Folgen geschieht mit einer definierten Wahrscheinlichkeit, wobei einfache Vokale und Konsonanten am häufigsten gewählt werden, gedehnte Vokale, Umlaute oder Konsonantenfolgen mit niedrigerer Wahrscheinlichkeit. Merkwürdige Anfänge wie einen mit 'Ck'lasse ich nicht zu. Zur Sicherheit sollen meine Passwörter immer mindestens eine Zahl enthalten und sowohl große als auch kleine Buchstaben verwenden. Sonderzeichen schließe ich allerdings aus. Wenn der Zufallsgenerator keine Zahl ins Passwort bringt oder keinen Großbuchstaben im gesamten Wort verwendet, wird als letztes Zeichen eine Zahl gewählt bzw. das erste Zeichen, das keine Zahl ist, groß geschrieben.

2 Umsetzung

Mein Programm habe ich in der Programmiersprache Python umgesetzt. Um zufäliige Zahlen zu bekommen oder eine zufällige Auswahl von Einträgen einer Liste, binde ich die Bilbliothek *random* ein und verwende die Funktionen:

- *choice:* um aus einer Liste einen zufälligen eintrag zu wählen
- *randrange:* um eine zufällige Zahl vom Typ integer zu bekommen

Ich definierne globale Arrays für bestimmte Buchstabengruppen (siehe Kapitel 4), die ich unterscheiden möchte bei der Verwendung:

- Einfache Konsonanten, die in der deutschen Sprache häufig verwendet werden
- Einfache Konsonanten, die seltener verwendet werden, wie z.B. 'x' oder 'y'

- Folgen von Konsonanten, die in der deutschen Sprache vorkommen, wie z.B. 'sp' oder 'ck'
- Vokale
- Umlaute wie z.B. 'ae' oder 'ei'
- Doppelvokale wie 'aa' bzw. gedehnte Vokale 'eh'

Ich habe die Routinen in einer Klasse *PasswordGenerator* defineirt. Die Hauptroutine ist die Methode *create_password(length)* der Klasse *PasswordGenerator*, der man die gewünschte Passwortlänge (muss größer als eine definiertre Zahl sein) als Parameter übergibt. Als Ergebnis liefert diese Methode das Passwort.
Die Methode ruft zunächst eine weitere Methode der Klasse auf (*_first_letter_algorithm*), die das erste Zeichen bzw. die erste Zeichenfolge definiert. Im Anschluss werden mit einer *While*-Schleife die weiteren Zeichen bzw. Zeichenfolgen definiert (beide Umsetzungen beschreibe ich weiter unten noch ausführlicher). Die *While*-Schleife läuft allerdings nur bis 'Passwortlänge minus eins', sodass in der Regel noch ein Zeichen frei bleibt, das ich gezielt als Zahl setzen kann, wenn im Passwort bisher noch keine solche auftaucht. Da ich aber auch Zeichenfolgen (z.B. zwei Zeichen) zulasse, kann es sein, dass nach der *While*-Schleife bereits alle Zeichen gesetzt sind und das Passwort fertig ist. In diesem Fall prüfe ich nur noch, ob eine Zahl enthalten ist und tausche im Fall, dass dem nicht so ist, noch das letzte Zeichen durch eine Zahl aus.
Ganz zum Schluss prüfe ich noch, ob im Passwort ein Großbuchstabe enthalten ist. Wenn dem nicht so ist, wird der erste verwendete Buchstabe des Passworts durch einen Großbuchstaben ersetzt.

Methode *_first_letter_algorithm(length)*: Zunächst wird dabei mit der Methode *_random_num* eine Zuallszahl zwischen 1 und 100 berechnet. Diese Zahl bestimmt, welche Art von Buchstaben gewählt wird. Hintereinander durchlaufene *if/else*-Abfragen verwenden die zufällige Zahl, um verschiedene Fälle zu verwenden. So kann ich prinzipiell die Wahrscheinlichkeiten für das Auftreten von Zahlen, Sonderzeichen etc. steuern.

Methode *create_password*: Sie durchläuft mit einer *While*-Schleife alle Zeichen des Passworts und definiert die zufälligen Zeichen unter berücksichtigung der definierten Regeln. Die Definition eines ersten Buchstabens (direkt am Passwortanfang oder mittendrin nach einer Zahl) wird mit der Methode *_first_letter_algorithm(length)* definiert. Im Unterschied zur Methode *_first_letter_algorithm(length)* wird hier berücksichtigt, welche Art Zeichen vorausging. Direkt hinter Buchstaben folge ich dem Prinzip 'auf Vokal folgt Konsonant und umgekehrt'. Dafür verwende ich das Flag *vokal*. Ansonsten nutze ich auch innerhalb des Passworts die Idee einer Zufallszahl, die die verschiednen Sonderfälle steuert.

3 Beispiele

Hier eine Liste von Ergebnissen von Passwörtern der Länge 10 Zeichen:

708Ot3keff	0Kivehyitu	Ihebedetu8	To7Ohitusp	Abipe6Fewu
Efarettuc2	Aphuschae5	Utonesumu0	Tihiesche7	1Utome3Afu
Woossuf6Sp	Uschivego6	1Id7velleb	Ebibefode7	f2Oetohw2i
4Olonutoph	Spiegeimm5	L4Umehspep	Foerallah5	Ufi18Epaup
Schuttast4	1Og8osp4aw	Sonnainum3	Staxe3Efue	Eissokani6
Leriffiss6	Schig574Po	Heffahspu3	Ecenolada8	Hoffaewie3
Efahttuti8	Faufeemme4	Saduttair8	1Stue8Kahr	7Uchopupuh
oenna4Stow	Molikijoo3	Emmecumuc5	T7Bevamovo	Phuw2Uebak
oli0Iwoett	Omottulug2	Buebiffig3	Vutt1Amaar	soevau6Oqu
A314Gell0e	Afuhittey5	Iffurahdo1	h7Ch3Fo2Wa	2Vidoerahk

Natürlich funktioniert es auch mit weniger oder mehr Zeichen. Allerdings habe ich definiert, dass es nicht weniger als 5 sein sollen. Hier eine Liste von Passwötern mit unterschiedlicher Länge.

8 Zeichen	**10 Zeichen**	**14 Zeichen**
Phasisi0	Bamu4Nivat	Beistaavoscho1
8Ulade0y	Zessib3Pha	Muhopoefehmm7a
Doll8Ira	Stoesito4g	Ahammeune8Ei3f

Ogawelo2	Olubommel1	Ka1Galeb3Ulass
Ipifol8u	Oe8O0Haeff	Estaluhi0Ael7u
ru8Etull	Chipallar7	Gostesabiffos0
Rafol3Oe	I76inauxoy	ph8Ennipurae1r
Immecop6	Badivoesi2	We2Acavowewupi
Chafeki4	Omelophoo1	23Luhf0Ruekuwa
E8Schovi	asto4Paspe	Schuttahdammu3
Baphool8	Mannefust0	Gopeduhulejee8
Eisoxus4	22Rissirof	Upanedo7Avivuh
W55Tisog	Chojerari7	6P68Ataffammap
Uennuff5	Ejuhfopuh2	Olukeihoolles0
Uehogor2	Oevobetak2	Ennemelugulle3
O4Istisp	Apeemmott0	Spovunatiquur6
4Inakira	Levohhopi5	Evobi3Rachespa
Ulallus4	Agikaacho2	Ev4A8Eibeckasu
Ennosun7	Ottolimme4	Uepasidowaven2
Vehte0Ra	Umeuh3stol	6Uteuckalebufo
Dussowo1	Auffad1yof	Ettommeweetto4
Avoyann7	Schiduhph0	Awennoffiwoos7
Be1Suhwe	Eineeg65ke	5I5Iffovidojad
Fauweno7	Taimmulle6	Phommittam27Nu
Phassuh5	Il5Ahischu	Dulletae3Ennut

4 Quellcode

Hier die global definierten Listen an Voikal- und Konsonantenfolgen

```
# Konsonanten
KONSONANTEN = ["B", "D", "F", "G", "H", "K", "L", "M", "N", "P", "R", "S", "T", "V", "W"]
KONSONANTEN_SELTEN = ["C", "J", "X", "Y", "Z"]
DOPPELKONSONANTEN = ["NN", "MM", "LL", "SS", "TT", "FF"]

SONDERFOLGEN_2 = ["SP", "ST", "CH", "PH", "QU", "CK"]
SONDERFOLGEN = SONDERFOLGEN_2 + ["SCH"]
SONDERFOLGEN_ANFANG_2 = ["SP", "ST", "CH", "PH", "QU"]
SONDERFOLGEN_ANFANG = SONDERFOLGEN_ANFANG_2 + ["SCH"]

# Vokale
VOKALE = ["A", "E", "I", "O", "U"]
UMLAUTE = ["AE", "UE", "OE", "AU", "EU", "EI", "AI"]
DOPPELVOKALE = ["AA", "AH", "EE", "EH", "UH", "OH", "OO", "IE"]
```

Hier die Klasse *PasswordGenerator* mit allen Methoden.

```
class PasswordGenerator:
    def __init__(self):
        self.password = ""              # Initialisierung des Passwort-Strings
        self.i = 0                      # Laufvariable, Position im Password-String
        self.letter_num = False         # Flag, dass sagt, dass das letzte Zeichen eine Zahl war
        self.jemals_num = False         # Flag, dass ich setze, wenn eine Zahl kommt
        self.jemals_gross = False       # Flag, dass ich setze, wenn ein Grossbuchstabe kommt
        self.vokal = None               # Flag, dass sagt, dass das letze Zeichen ein Vokal war

    def create_password(self, length):
         if length < MINLAENGE:          # Pruefung, ob die minimale Passwortlaenge gegeben ist
            print(f"Length too short, minimum {MINLAENGE} letters!")
            return "error"

        # Erstes Zeichen des Passworts
        letter = self._first_letter_algorithm(length)
        self.password += letter

        # Mittlere Zeichen (meist zweites, ggf. 3./4. je nach Laenge ersten Zeichenfolge)
```

```
20          while self.i < length - 1:
                if self.letter_num:                         # Wenn letztes Zeichen Zahl,
22                                                          # wieder die Fkt des ersten Zeichens
                    self.letter_num = False
24                  letter = self._first_letter_algorithm(length)
                else:
26                  self.letter_num = False
                    random_num = self._random_num()         # Zufallszahl zwischen 1 und 100
28                  if random_num <= 10:                    # Mit 10% Wahrsch eine Zahl
                        letter = str(randrange(0, 9))
30                      self.letter_num = True
                        self.jemals_num = True
32                      self.i += 1
                    elif self.vokal:                        # Vorher Vokal, also nun Konsonant
34                      self.vokal = False
                        if random_num <= 75:                # Einfacher Konsonant mit (75-10)%= 65% Wahrsch
36                          if self._random_konsonant():
                                letter = choice(KONSONANTEN).lower()
38                          else:
                                letter = choice(KONSONANTEN_SELTEN).lower()
40                          self.i += 1
                        elif random_num <= 90:              # Doppelkonsonant mit (90-75)%= 15% Wahrsch
42                          letter = choice(DOPPELKONSONANTEN).lower()
                            self.i += 2
44                      else:                               # Sonderfolge mit 10% Wahrsch.
                            if length-self.i > 3:           # inkl. 3-Zeichen langer Sonderfolgen nur,
46                                                          # wenn es noch passt
                                letter = choice(SONDERFOLGEN).lower()
48                          else:
                                letter = choice(SONDERFOLGEN_2).lower()
50                          self.i += len(letter)

52                  else:                                   # Vorher Konsonant, also nun Vokal
                        self.vokal = True
54                      if random_num <= 80:                # Einfacher Vokal mit 80-15%= 65% Wahrsch
                            letter = choice(VOKALE).lower()
56                          self.i += 1
                        elif random_num <= 90:              # Doppelvokal mit 10% Wahrsch
58                          letter = choice(DOPPELVOKALE).lower()
                            self.i += 2
60                      else:                               # Umlaut mit 10% Wahrsch
                            letter = choice(UMLAUTE).lower()
62                          self.i += 2
            self.password += letter
64
        # Da ich Doppelfolgen von Buchstaben zulasse, kann es bei der vorherigen while-Schleife sein,
66      # dass schon die geforderte Anzahl an Buchstaben erreicht ist.
        # In der Regel ist aber noch eine Stelle frei. Diese wird nun besetzt.
68      # Ich nutze dies, um eine Zahl zu vergeben, wenn noch keine im Wort ist

70      if len(self.password) < length:                     # Noch ein Zeichen Platz am Ende
            if  self.jemals_num == False:
72              letter = str(randrange(0, 9))
                self.letter_num = True
74              self.jemals_num = True

76          else:
                if self.vokal == True:
78                  if self._random_konsonant():
                        letter = choice(KONSONANTEN).lower()
80                  else:
                        letter = choice(KONSONANTEN_SELTEN).lower()
82              else:
                    letter = choice(VOKALE).lower()
84          self.i += 1
            self.password += letter
86
        else:                                       # Alle zeichen schon voll
88          if self.jemals_num == False:            # Sonderfall keine Zahl --> Zahl ans Ende
                # self.password = str(randrange(0, 9)) + self.password[1:]
90              self.password = self.password[:-1] + str(randrange(0, 9))
                # print("Letzten Buchstaben zur Zahl gemacht")
92
```

```
        if self.jemals_gross == False:          # Sonderfall kein grosser Buchstabe -> 1. ersetzen
94          if not self.password[0].isnumeric():
                self.password = self.password.capitalize()
96          else:
                str1 = ""
98              cap = False
                for x in self.password:
100                 if not cap and not x.isnumeric():
                        x = x.upper()
102                     cap = True
                    str1 += x
104             self.password = str1

106     return self.password

108 # Liefert ein zufaelliges erstes Zeichen bzw. eine Zeichenfolge;
    # Wird auch aufgerufen nach Zahl im Passwort
110
    def _first_letter_algorithm(self, length):
112     random_num = self._random_num()             # Zufallszahl zwischen 1 und 100
        letter = ""
114     self.letter_num = False
        self.vokal = False
116
        if random_num <= 30:                        # Einfacher Vokal mit 30% Wahrsch
118         if self._random_capital():              # Grossbuchstabe mit gewisser Wahrsch. (aktuell 75%)
                letter = choice(VOKALE)
120             self.jemals_gross = True
            else:                                   # Kleinbuschstabe mit gewisser Wahrsch
122             letter = choice(VOKALE).lower()
            self.vokal = True
124         self.i += 1

126     elif random_num <= 40:                      # Umlaut mit 10% (am Anfang kein Doppelvokal)
            if self._random_capital():              # Grossbuchstabe am Anfang
128             letter = choice(UMLAUTE).lower().capitalize()
                self.jemals_gross = True
130         else:                                   # Kleinbuchstabe am Anfang
                letter = choice(UMLAUTE).lower()
132         self.vokal = True
            self.i += 2
134
        elif random_num <= 75:                      # Einfacher Konsonant mit 35%
136         if self._random_konsonant():
                kons = choice(KONSONANTEN)
138         else:
                kons = choice(KONSONANTEN_SELTEN)
140         if self._random_capital():              # Grossbuchstabe
                letter = kons
142             self.jemals_gross = True
            else:                                   # Kleinbuchstabe
144             letter = kons.lower()
            self.i += 1
146
        elif random_num <= 85:                      # Sonderfolge mit 10%
148         if length-self.i < 3:                   # Diese Abfrage braucht es eigentlich nicht mehr
                if self._random_capital():
150                 letter = choice(SONDERFOLGEN_ANFANG_2).lower().capitalize()
                    self.jemals_gross = True
152             else:
                    letter = choice(SONDERFOLGEN_ANFANG_2).lower()
154         else:
                if self._random_capital():
156                 letter = choice(SONDERFOLGEN_ANFANG).lower().capitalize()
                    self.jemals_gross = True
158             else:
                    letter = choice(SONDERFOLGEN_ANFANG).lower()
160         self.i+= len(letter)

162     else:                                       # also 15% Wahrsch. --> Zahl
            letter = str(randrange(0, 9))
164         self.letter_num = True
            self.jemals_num = True
```

```
166             self.i += 1

168         return letter

170     def _random_capital(self):
            # Liefert mit 75% Wahrscheinlichkeit True, sonst False
172         return self._random_num() > 25

174     def _random_konsonant(self):
            # Liefert mit 90% Wahrscheinlichkeit True, sonst False
176         return self._random_num() > 10

178     def _random_num(self):
            # Zufallszahl zwischen 1 und 100
180         return randrange(1, 100)

182 # Hier noch fder Aufruf zur Genierierung von Beispielen
    for x in range(25):
184     client = PasswordGenerator()
        pw = client.create_password(8)
186     print(pw, len(pw))
        client = PasswordGenerator()
188     pw = client.create_password(10)
        print(pw, len(pw))
190     client = PasswordGenerator()
        pw = client.create_password(14)
192     print(pw, len(pw))
```