Prof. Dr. Christian Baun, Leonard Hußke, Anton Rösler Fachbereich 2 Betriebssysteme und Rechnernetze (SS2021) Frankfurt Univ. of Appl. Sciences

Portfolioprüfung – Werkstück A – Alternative 4

Unsere Gruppe hat sich für die Aufgabenstellung Password-Manager entschieden.
Einer der festgelegten Aufgabenstellungen ist es, dass der Nutzer selbst entscheiden kann, ob er ein eigenes Passwort für eine bestimmte Webseite erstellen möchte oder das Programm eins für ihn bereitstellt.
Jedoch wird hier abgefragt, wie lang sein Passwort sein muss, wie viele Großbuchstaben, Kleinbuchstaben und Sonderzeichen es enthalten solle. Unser Quellcode beginnt mit dem zufälligen Generieren des Passwortes. Dafür wurden Methoden erstellt, die die Länge des Passwortes überprüfen. Falls der Nutzer die Anzahl der Groß-und Kleinbuchstaben, sowie die Anzahl der Sonderzeichen eingibt, müssen diese in der Summe gleich sein, wie die Passwortlänge. welches er wählte.
Wenn es nicht der Fall ist, wird der Nutzer darauf hingewiesen und es kommt zu der erneuten Abfrage, es diesmal in der richtigen Anzahl anzugeben. Als zusätzliche Funktionen geben wir dem Benutzer ein Hinweis, wie sicher sein Passwort ist, welches er erstellte. Dazu haben wir uns online informiert, ab wie viel Zeichen ein Passwort sicher ist.
Wenn der Benutzer also ein Passwort erstellt, erhält er einen hinweis, ob es sicher oder unsicher ist. Falls das Passwort nicht sicher ist, bekommt er die Option es direkt zu ändern, um ein besseres zu generieren. Ansonsten erscheint ein Befehl noch, falls der Nutzer das Programm verlassen will.
Er kann „off“ in der Benutzereingabe schreiben und verlässt das Programm. Bei der nächsten Methode namens def passGenerieren(): werden die einzelnen Variablen angezeigt, die beim Erstellen des Passwortes wichtig sind. Damit das Programm weiß, mit was es das Passwort generieren soll, haben wir drei Variablen erstellt. Für die Groß- und Kleinbuchstaben nutzen wir den Ascii Code. (Für Großbuchstaben = ascii_uppercase und für die Kleinbuchstaben ascii_lowercase. Für die Variabel Zeichen erstellten wir selbst ein ???. Demnächst wurde dann ein Input zu den einzelnen ??? erstellt. Hier kommt die Eingabe des Nutzers. Und im Hintergrund wird ein zufälliges Passwort für die Person nach seinen Wünschen generiert.

Zusätzlich fügten wir die Funktion Pyperclip ein. Diese dient dazu, dass das Passwort in der Zwischenablage gespeichert wird. Dafür muss man zuerst Pyperclip importieren.
Dies allein reicht nicht aus, sondern man muss es beim Terminal noch hinzukommend installieren. Dazu gibt man nur den Befehl: pip Install pyperclip ein. Wir haben uns an der Aufgabenstellung orientiert und haben das Passwort für 30 Sekunden in die Zwischenablage kopiert. Nach 30 Sekunden wird Zwischenablagerung geleert.

Als nächstes beschäftigten wir uns mit dem Masterpasswort. Das Masterpasswort ist eine wichtige Funktion, denn allein durch das Masterpasswort gelangt der Nutzer in seinen Passwort-Manager.
Der Benutzer wird gefragt, ein eigenes Masterpasswort nach seinen eigenen Vorstellungen zu erstellen, ohne die vorherigen Bedingungen mit Länge und Sonderzeichen etc. einzuhalten.
Die Erstellung des Masterpassworts und die spätere Eingabe, um sich anzumelden, werden jedoch nicht sichtbar gemacht, um die Sicherheit des Benutzers zu gewährleisten. Dazu importierten wir "stdiomask und verschleierten die Eingabe auf der Konsole mit *. Dies ist wichtig, da ansonsten Dritte Zugriff auf die Passwörter hätten, falls sie sich mit dem abgeschauten Masterpasswort des Nutzers den Zugriff erschleichen wollen.
Demnächst wird von dem Nutzer wieder verlangt sein erstelltes Masterpasswort einzugeben. Falls er es richtig eingab, fährt das Programm fort.

Hier setzten wir als Bedingung die aktuelle Softwarezeit (datetime.now) ein und subtrahierten das mit der Startzeit (startTime). Das Ergebnis wandelten wir in Sekunden, um mit der Funktion total_seconds(). Dadurch haben wir eine Zeitspanne von 600 Sekunden bzw. 10 Minuten gewählt, in der, der Benutzer wieder sein Masterpasswort korrekt eingeben muss, um mit dem Programm fortzufahren, damit keine Unbefugten das Programm weiter nutzen können, falls er vergessen hat es zu schließen. In der If-Bedingung fragten wir wieder nach dem Masterpastwort und vergleichen wir dies mit der Eingabe des Benutzers.
Wenn die Eingaben übereinstimmen, kommt die Ausgabe „Richtig“ und das Programm wird fortgeführt und bei der falschen Eingabe wird es mit „falsch“ ausgegeben und das Programm wird geschlossen durch exit.
Wir haben das Masterpasswort mit global deklariert, damit das Masterpasswort immer im Programm abgerufen werden kann, wenn die Zeitspanne von 600 Sekunden abläuft.

Weiter gehts im nächsten Schritt. Wir erstellten hier eine While Schleife, denn erst mit der richtigen Eingabe werden ihm die fünf verschiedene Möglichkeiten angezeigt. Falls das falsche Passwort eingegeben wurde, bekommt der Benutzer eine Ausgabe mit „Falsches Passwort“ und das Programm wird auch schon beendet. Wir erstellten fünf verschiedene Optionen in unserem Passwort-Manager. Diese werden ihm angezeigt und ihm steht die freie Wahl und wurden mit if, elif,else Befehlen durchgeführt.

Die erste Möglichkeit ist es, seine Passwörter angezeigt zu bekommen mit der dazugehörigen Webseite und dem Benutzernamen. Wir haben uns dafür entschieden die ganzen Daten in einem File (Textdokument) zu sichern. Es kamen auch Versuche, die Datei in einer Datenbank speichern zu können, aber nach stundenlanger Recherche, kamen wir nicht viel weiter und wählten uns das Speichern in einer File aus, namens password.txt. Zur Anzeige der Dateien vom Nutzer mussten wir die File zuerst öffnen mit dem Befehl: with open (‚password.txt', ‚r') as f:
‚r' dient zum Lesen der File mit f.read(), welches dann mit f_contents gleich gesetzt wurde und der

Befehl aufkam, f_contents auszugeben mit print().
Nachdem die Daten des Nutzers abgebildet worden sind, kann sich die Person seinen weiteren Schritt auswählen. Hierfür werden ihm wieder die fünf Auswahlmöglichkeiten angezeigt.

In der zweiten Option geht es um die Passworterstellung. Zum generieren erstellten wir, wie oben schon detailliert genannt, Methoden, damit es übersichtlicher erstellt wird. Hier werden die einzelnen Dinge vom Nutzer abgefragt.
Zu aller erst nach dem Namen der Webseite. Wir fügten noch nach dem Input den Befehl .lower() hinzu. Dies dient dazu, falls der Benutzer den Namen der Webseite groß schreiben sollte, wird durch .lower() daraus Kleinbuchstaben werden und unser Programm erkennt, um welche Webseite es sich hier handelt. Denn durch die falsche Groß- und Kleinbuchstabe, kann diese Eingabe später mit dem gespeichertem nicht erkannt werden.
Des Weiteren wird nach dem Benutzernamen gefragt. Daraufhin stellt man dem Nutzer die Möglichkeit sein Passwort vom Programm generieren zu lassen oder sich sein eigenes auszuwählen. Dafür muss er entweder den Buchstaben g (für generieren des Passwortes) oder e für (eigenes Passwort). Wir haben auch hierfür eine if, elif, else Befehl eingefügt. Wenn der Nutzer den Buchstaben g eingibt, wird der Befehl passGenerieren() aufgerufen.
Ihm werden dann nacheinander die einzelnen Fragen zum Passwortgenerieren gefragt:

```
laenge = int(input("Bitte geben Sie die Länge des Passwortes ein: "))
grosbuchstaben = int(input("Bitte geben Sie die Anzahl der Großbuchstaben ein: "))
grosbuchstaben = int(pruefGrosBuchstaben(grosbuchstaben, laenge))
pw = pw + (''.join(secrets.choice(buchstabenGross) for _ in range(grosbuchstaben)))
kleinbuchstaben = int(input("Bitte geben Sie die Anzahl der Kleinbuchstaben ein: "))
kleinbuchstaben = int(pruefKleinBuchstaben(kleinbuchstaben, laenge, grosbuchstaben))

pw = pw + (''.join(secrets.choice(buchstabenKlein) for _ in range(kleinbuchstaben)))
sonderzeichen = int(input("Bitte geben Sie die Anzahl der Sonderzeichen ein: "))
sonderzeichen = int(pruefSonderZeichen(sonderzeichen, laenge, grosbuchstaben, kleinbuchstaben))
pw = pw + (''.join(secrets.choice(zeichen) for _ in range(sonderzeichen)))
# file.txt

digit = string.digits
pw = pw + (
    ''.join(secrets.choice(digit) for _ in range(laenge - (grosbuchstaben + kleinbuchstaben + sonderzeichen))))

print(pw)
```

Zum besseren Verständnis habe ich die Befehle aufgezählt. Wir fügten vor dem input ein integer hinzu, damit Python weiß, dass nur Zahlen als Anwortmöglichkeiten gelten. Falls der Benutzer doch ein String eingibt, wird ihm ein Fehler angezeigt, da dies nicht gültig ist für diesen Input.

Nach dem Input wird hier geprüft, ob die Anzahl der angegebenen Groß- und Kleinbuchstaben oder Sonderzeichen mit der angegeben Länge übereinstimmen.
Daraufhin wird dann ein zufälliger Buchstabe oder Zeichen aus der Variablen ausgewählt und ein Passwort erstellt. Dies geschieht mit der Anforderung(secrets.choice).
Hierzu erstellten wir ein Import Secrets, damit man auf secrets.choice zugreifen kann. Wenn der Nutzer den Buchstaben e eingibt, kommt die Frage mit dem Input "Bitte geben Sie ihr Passwort ein:")
Die Passwörter werden dann durch pyperclip.copy in der Zwischenlage kopiert und der Nutzer bekommt auch dementsprechend ein Hinweis mit "Dein erstelltes Passwort wurde im Clipboard gespeichert.") Wir definierten dann zusätzlich noch eine Methode namens def fe(): .
Hier wurde unser file password.txt wieder aufgerufen, nur diesmal mit ein 'a' , was zu bedeuten hat, dass Informationen in die File hinzugefügt wird.
In diesem Falle werden neue Daten hinzugefügt, wie das neu erstellte Passwort und die dazugehörigen Benutzernamen und der Webseite.
Dies geschieht auch mit dem Befehl .write({} {} {}). In die geschweiften Klammern kommen die deklarierten Variablen ws, bn und pw. ws steht bei uns für die Eingabe der Webseite, bn für die Eingabe des Benutzernamen und pw für das erstellte Passwort. Mit dieser Methode wurden die neu angegebenen Daten in die File Datei gespeichert. Und zuletzt wird die def fe(): Methode mit dem Befehl fe() aufgerufen.

Als nächste Option bieten wir dem Benutzer die Möglichkeit an, sein Passwort löschen zu können. Dazu programmierten wir eine simple Methode:
Wir forderten den Benutzer durch die Input Eingabe auf, die zu gelöschte Webseite zu nennen, welche dann später in der For-Schleife in Kleinschrift umgewandelt wird, damit das Programm die dazugehörige Webseite erkennt.
Der Nutzer wird nochmal aufgefordert, die Bestätigung seiner Wahl zu tätigen.
Wir benutzen hier die Zahl -1, da ein Array nicht ins negative gehen kann und deswegen das Passwort gelöscht wird. Wir erstellen eine For-Schleife, die durch die Tabelle durchgeht und die gesuchte Input Eingabe vergleicht und die gesuchte Webseite und das dazugehörige Passwort entfernt. Falls die Webseite nicht gefunden wird, kommt die Anzeige, dass der Eintrag nicht gefunden wurde.

Hinzukommend stellen wir dem Nutzer auch die Möglichkeit sein Passwort ändern zu können.
Im Prinzip läuft es hier wie bei der dritten Möglichkeit ab.
Hier wird der Benutzer wieder angefordert die dazugehörige Webseite zu nennen.
Wir erstellten hier auch eine For-Schleife, damit das Programm das gesuchte Element an der i- Stelle ändert. Der Nutzer kommt zu der Aufforderung sein neues Passwort wählen zu können.

Die letzte Option, die auf der Liste steht, ist das Beenden des Passwort-Managers mit exit().

Auf Github haben wir bewusst all unsere Files nicht gelöscht, um zu zeigen wie wir anfingen und wie unser Endprogramm am Ende aussieht.

Kommentare haben wir an manchen Stellen ausführlich geschrieben und an anderen Stellen bewusst weggelassen, da meistens die Bezeichnungen im Code für sich sprechen und keine weiteren Kommentare nötig gewesen sind und wir Wiederholungen im Text vermeiden wollten, damit das Programm nicht mit unnötigen Texten beschriftet wird.

```
from datetime import datetime
import getopt
import secrets
import string
import time
from threading import Thread
import pyperclip
import sys
import os
```

Wir mussten bei Python zahlreiche Hilfsdateien importieren, damit das Programm in vollen Umfang programmieren kann.
From datetime import datetime = dient dazu, dass wir auf die Echtzeit zugreifen können.
Wir verwendeten diese beim Abfragen des Masterpasswortes nach einer Zeit
Import secrets = dient dazu, dass wir zufällige Zeichen auswählen können
Wir verwendeten diese, um zufällige Zeichen für das Passwort generieren zu können
Import string = für Zeichenketten
From threading import Thread = benutzen wir dafür, dass die Zwischenablage geleert wird nach einer Zeit
Import pyperclip = dient dazu, dass wir das Passwort speichern konnten in der Zwischenablage
Dafür mussten wir beim Terminal: Pip Install pyperclip installieren
Import sys = Zugriff
Import os = wenn man Dateien verschieben und verändern möchte

```
    else:
        # Eingabestrings normieren zum vergleich
        if argv[0].lower() == "add":
            if len(argv) != 6:
                # 6 Parameter=> damit man sich an add -title moodle -username -generatepassword orientiert
                print("Parameter Fehlt")
                print("Bitte folgendes Schema benutzen:")
                print("password_manager.py Add -title <Title> -username <Username> -generatepassword")
                sys.exit(0)
            title = ""
            username = ""
            generatedpassword = False
            for i in range(1, len(argv)):
                if argv[i] == "-title":
                    try:
                        title = argv[i + 1]
                    except:
                        pass
                elif argv[i] == "-username":
                    try:
                        username = argv[i + 1]
                    except:
                        pass
                elif argv[i] == "-generatepassword":
                    generatedpassword = True

            if title == "" or username == "" or generatedpassword == False:
                print("Falsche eingabe")
                sys.exit(0)
            else:
                print("Ausgewaehlte Title: " + title)
                print("Ausgewaehlte Username: " + username)
                fe(title, username, autoPassGenerieren())
```

```
            else:
                print("Ausgewaehlte Title: " + title)
                print("Ausgewaehlte Username: " + username)
                fe(title, username, autoPassGenerieren())
                print("OK password created and copied to clipboard")

        if argv[0].lower() == "copy":

            # 3 wegen title, username, generatepassword

            if len(argv) == 3:
                if argv[1] == "-title":
                    title = argv[2]
                    for i in range(len(websites)):
                        if websites[i].lower() == title.lower():
                            print("Username:" + usernames[i])
                            print("Password:" + passwords[i])
                            pyperclip.copy(passwords[i])
                            print("Password copied to clipboard")

            else:
                print("Parameter Fehlt")
                print("Bitte folgendes Schema benutzen:")
                print("password_manager.py copy -title <Title> ")
                sys.exit(0)


if __name__ == "__main__":
    readPasswords()
    main(sys.argv[1:])
```

3 Beispielhafte Befehle

Neues Passwort für Moodle generieren:

```
$ passman add -title moodle -username uas123456 -generatepassword
--> OK password created and copied to clipboard
```

Passwort für Moodle abrufen:

```
$ passman copy -title moodle
--> Master-Password:
--> Master ok
--> Username: uas123456 | password copied to clipboard
```

Das hier ist die Shell Funktion.
Die 2 Bilder, die oben abgebildet sind, sind so programmiert worden, wie in der Aufgabenstellung gefordert wurden.