



FRANKFURT UNIVERSITY OF
APPLIED SCIENCES
PORTFOLIOPRÜFUNG
WERKSTÜCK A -
ALTERNATIVE 4
PASSWORD-MANAGER

*von Soufian Zaatani (1126983)
und Nico Coglianini (1268359)*



INHALTSVERZEICHNIS

1. Einleitung
2. Befehl 1 - Das Erstellen eines Masterbenutzers/einer Masterbenutzerin
3. Befehl 2 - Das Hinzufügen von Einträgen
4. Befehl 3 - Das Löschen von Einträgen
5. Befehl 4 - Einträge aus der Ablage herauskopieren
6. Befehl 5 – Das Generieren eines zufälligen Passworts
7. Befehl 6 - Einträge als Tabelle anzeigen
8. Befehl 7 - Das Masterpasswort ändern
9. Befehl 8 - Das Ändern von einem Eintrag
10. Befehl 9 - Das Überprüfen von redundanten Passwörtern
11. Befehl 10 – Abrufen aller Funktionen

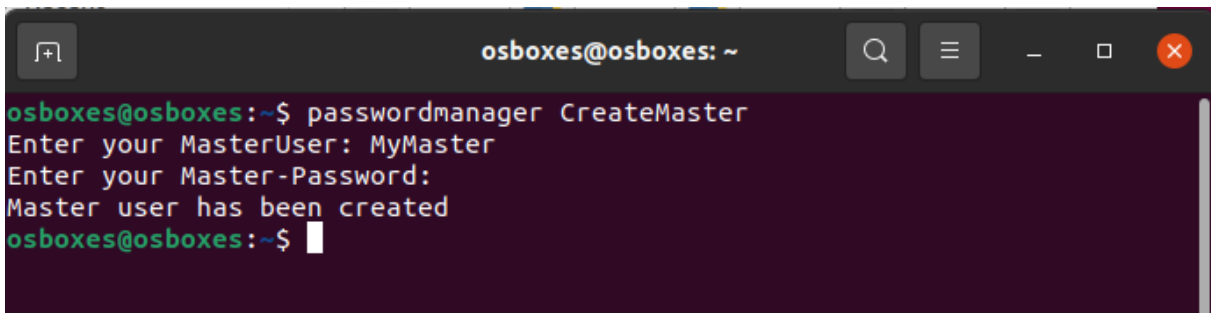
Einleitung

In folgender Ausarbeitung werden die Herangehensweise und das Implementieren von einem Passwort-Manager genaustens dargestellt. Ein Passwort-Manager ist eine Anwendungssoftware, mit welcher ein Nutzer Passwörter oder wichtige Codes sicher speichern kann. Diese können dann wiederverwendet werden, da sie sicher in einer Ablage gespeichert werden. Außerdem werden nicht nur alle erstellten Passwörter sicher aufgehoben, sondern der Passwort-Manager kann auch selbst sichere Passwörter generieren und abrufen.

Nachdem die endgültige Wahl getroffen wurde, die Alternative 4 zu bearbeiten, mussten die gelieferten Anforderungen erneut analysiert werden. Durch das durchforsten der Anforderungen sind wir als Team zum Entschluss gekommen diese in Unterschiedliche Befehle aufzuteilen. Ein nicht leicht zu bewältigendes Hindernis war die Anforderung das Programm direkt aus dem Terminal zu verwenden. In der Regel werden Python Skripte von einem/einer Benutzer/Benutzerin oder einem anderen Skript aufgerufen, deshalb war es für uns als Team eine neue Erfahrung im Bereich der Linux bzw. Python Entwicklung. Eine besondere Herausforderung war es für das Team weitere sinnvolle Funktionen zu ermitteln und zu implementieren, da die ersten Erfahrungen in der Objektorientierten Programmierwelt mit Java stattgefunden haben.

Befehl 1 Das Erstellen eines Masterbenutzers/einer Masterbenutzerin

Mithilfe von diesem Befehl soll es dem/der Benutzer/-in möglich gemacht werden einen Masterbenutzer oder eine Masterbenutzerin zu erstellen. Der/Die Masterbenutzer/-in hat das Recht Einträge in das Programm einzupflegen. Zu den Einträgen gehören der Titel des Eintrags, der Benutzername des Eintrags und das zu speichernden Passwort. Das Aufrufen der Einträge ist nur mit einem/einer Masterbenutzer/-in möglich und stellt somit eine besondere Sicherheit zur Verfügung. Durch die Anforderungsanalyse wurde entschieden das der Befehl wie folgt aussehen soll:



```
osboxes@osboxes: ~  
osboxes@osboxes:~$ passwordmanager CreateMaster  
Enter your MasterUser: MyMaster  
Enter your Master-Password:  
Master user has been created  
osboxes@osboxes:~$
```

Durch das Ausführen des Befehls startet ein Dialog im Terminal, mit dem es möglich ist, ein/eine Masterbenutzer/-in zu erstellen. Im Hintergrund wird dann ein Python Skript ausgeführt, das den eingegebenen String in die Variable „masteruser“ abspeichert. Anschließend wird der/die Benutzer/-in aufgefordert ein Masterpasswort einzugeben. Der eingegebene String wird in die „masterpassword“ Variable abgespeichert. Durch das Python eigene „Getpass“ Module, wird das Passwort beim Eingeben nicht angezeigt.

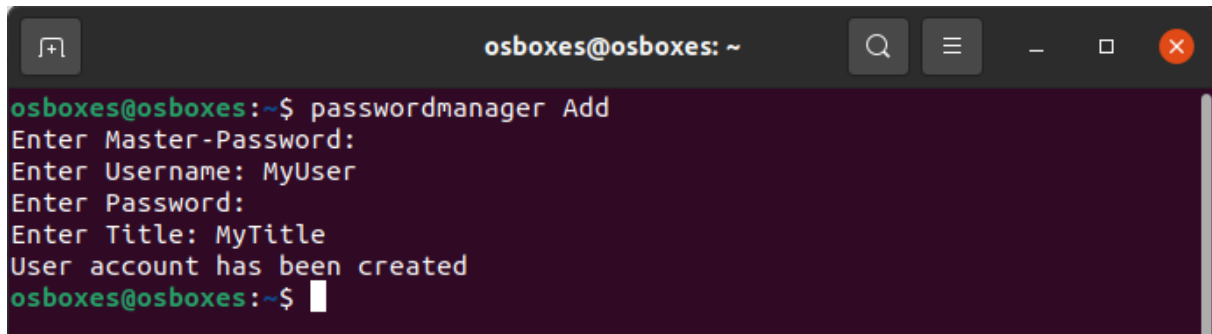
```
if cmd == "CreateMaster":  
    masteruser = input("Enter your MasterUser: ")  
    masterpassword = getpass.getpass("Enter your Master-Password: ")  
    add_new_master (masteruser, masterpassword)  
    print("Master user has been created")
```

Sobald beide Variablen initialisiert wurden, wird die Funktion „add_new_master“ ausgeführt. Mit dieser Funktion werden die beiden Variablen zusammengesetzt, in ein allgemeines Format und in einer neuen Zeile abgespeichert.

```
def add_new_master (username, password):  
    storage_format = f"{username}{{password}}"  
    append_new_line(masters_credencials, storage_format)
```

Befehl 2 Das Hinzufügen von Einträgen

Mithilfe von diesem Befehl soll es dem Benutzer oder der Benutzerin möglich gemacht werden einen Eintrag für den Passwort-Manager zu erstellen. Ein Eintrag besteht aus einem Titel, einen Benutzernamen und einem Passwort. Der Titel soll beschreiben wofür die Zugangsdaten benutzt werden zum Beispiel Moodle, Facebook etc. Durch die Anforderungsanalyse wurde der Entschluss getroffen das der Befehl wie folgt aussieht:

A terminal window with a dark purple background. The title bar shows 'osboxes@osboxes: ~' and standard window controls. The terminal text shows the command 'passwordmanager Add' being executed, followed by prompts for Master-Password, Username (MyUser), Password, and Title (MyTitle). A confirmation message 'User account has been created' is displayed, followed by the shell prompt 'osboxes@osboxes:~\$' and a cursor.

```
osboxes@osboxes:~$ passwordmanager Add
Enter Master-Password:
Enter Username: MyUser
Enter Password:
Enter Title: MyTitle
User account has been created
osboxes@osboxes:~$
```

Durch das Ausführen des Befehls wird vom Terminal ein Dialog gestartet. Zunächst muss der Benutzer oder die Benutzerin das Masterpasswort eingeben, daraufhin ist es ihm/ihr möglich einen Eintrag zu erstellen. Der Benutzer oder die Benutzerin erhält bei erfolgreichem Erstellen eine Bestätigung. Um erfolgreich den Eintrag zu hinterlegen, muss der Benutzer oder die Benutzerin einen Benutzernamen, ein Passwort und den dazugehörigen Titel eingeben. Bei erfolgreichen hinzufügen eines Eintrags erhält der Benutzer oder die Benutzerin eine Bestätigung.

```
elif cmd == "Add":
    password_master = getpass.getpass("Enter Master-Password: ")
    verification_password = master_pass_exists(password_master)

    if verification_password:

        username = input("Enter Username: ")
        password = getpass.getpass("Enter Password: ")
        title = input("Enter Title: ")
        add_new_user (username, password, title)
        print("User account has been created")
    else:
        print("Wrong Master-Password")
```

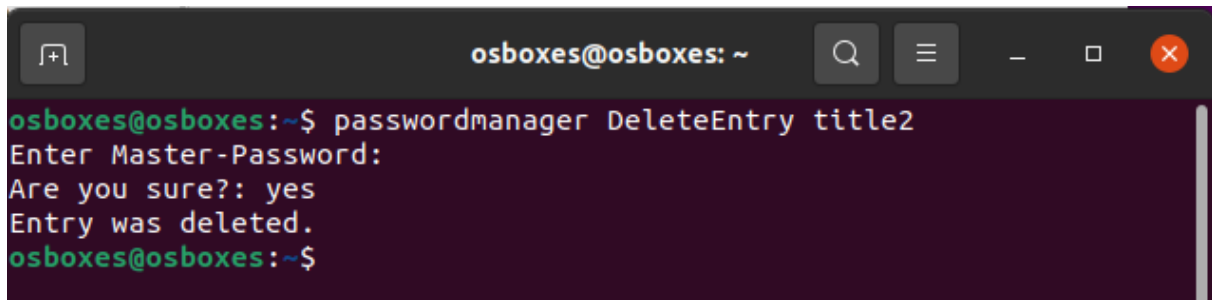
Im Hintergrund wird geprüft, ob das Masterpasswort richtig ist. Dies geschieht mit der Funktion „master_pass_exist()“. Die genauere Funktion wird im folgenden Quellcode angegeben.

```
def master_pass_exists(password):  
    return any((masters_pass == password) for _, masters_pass in  
get_existing_masters())
```

Mithilfe einer for-Schleife wird das angegebene Passwort, mit dem abgelegten Passwort verglichen.

Befehl 3 Das Löschen von Einträgen

Um den Benutzer oder der Benutzerin die Möglichkeit zu geben seine/ihre Einträge wieder zu löschen, kann der Benutzer oder die Benutzerin den Befehl „DeleteEntry“ benutzen. Der Befehl sieht wie folgt aus:

A terminal window titled 'osboxes@osboxes: ~' with search, menu, and window control icons. The terminal shows the command 'passwordmanager DeleteEntry title2' being executed. The output is: 'Enter Master-Password:', 'Are you sure?: yes', and 'Entry was deleted.' followed by the prompt 'osboxes@osboxes:~\$'.

Der Befehl wird mit dem Parameter „Titel“ ergänzt, damit der/die Benutzer/-in den genauen Eintrag löschen kann. Um dem Benutzer oder der Benutzerin eine Möglichkeit zu geben seine/ihre Entscheidung zu bestätigen, wird vor dem endgültigen Löschen gefragt, ob der Benutzer oder die Benutzerin sich sicher ist.

Die Parameterübergabe wurde mit den „sys.argv“ implementiert. Mit der ersten IF Abfrage wird überprüft, ob das Masterpassword verifiziert ist. Tritt dies ein, wird mit einer weiteren IF Abfrage überprüft ob der Benutzer oder die Benutzerin sich sicher ist, den Eintrag zu löschen. Willig der Benutzer oder die Benutzerin ein den Eintrag zu löschen, wird die Funktion „del_existing_users()“ abgerufen.

```
elif cmd == "DeleteEntry":# -title
    title = sys.argv[2]
    password_master = getpass.getpass("Enter Master-Password: ")
    verification_password = master_pass_exists(password_master)
    if verification_password:
        confirmation = input("Are you sure?: ")
        if confirmation.lower() == "yes":
            del_existing_users(title)
            print("Entry was deleted.")
        elif confirmation.lower() == "no":
            print("Entry was not deleted.")
    else:
        print("Wrong Master-Password")
```

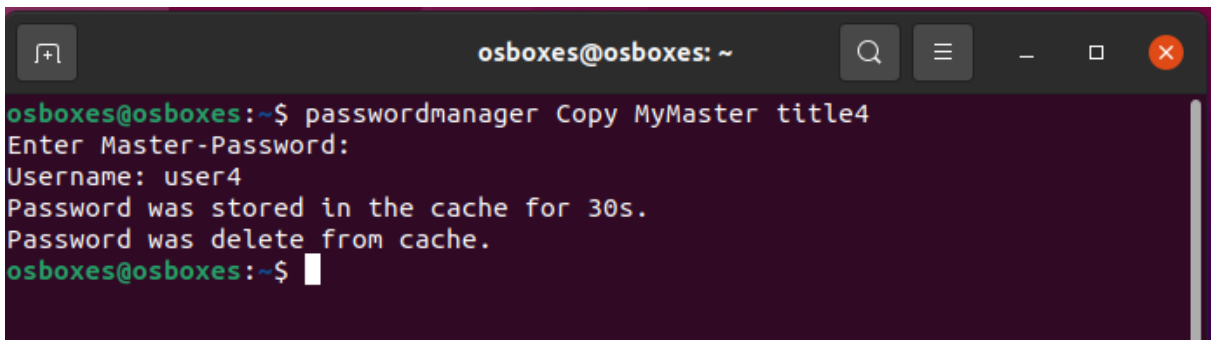
Die Funktion del_existing_users() speichert als erstes die richtige Zeile des Eintrags. Die bereits existierenden Einträge werden in eine Variable zwischengespeichert.

Durch das Python eigene Statement „del“ können Zeilen aus einer Liste oder auch Text entfernt werden. Durch die zuvor gespeicherte Zeile kann nun die ausgewählte Zeile entfernt und die Ablage Datei überschrieben werden.

```
def del_existing_users(user,title):
    match_line = get_line_number(users_credencials, user, title)
    old_file = open(users_credencials, "r")
    lines = old_file.readlines()
    old_file.close()
    del lines[match_line]
    new_file = open(users_credencials, "w+")
    for line in lines:
        new_file.write(line)
    new_file.close()
```


Befehl 4 Einträge aus der Ablage herauskopieren

Mithilfe von diesem Befehl ist dem Benutzer oder der Benutzerin die Möglichkeit gegeben ein Eintrag für 30 Sekunden zwischenspeichern ohne das Passwörter im Klartext sichtbar werden. Nach den 30 Sekunden wird das Passwort aus dem Zwischenspeichern gelöscht. Um die Sicherheit zu gewährleisten, wird das Passwort nur nach erfolgreicher Verifizierung ausgegeben. Somit kann der Benutzer oder die Benutzerin das Passwort auf eine Internetseite oder einem Programm einfügen. Der Befehl für das Kopieren eines Passworts sieht wie folgt aus.



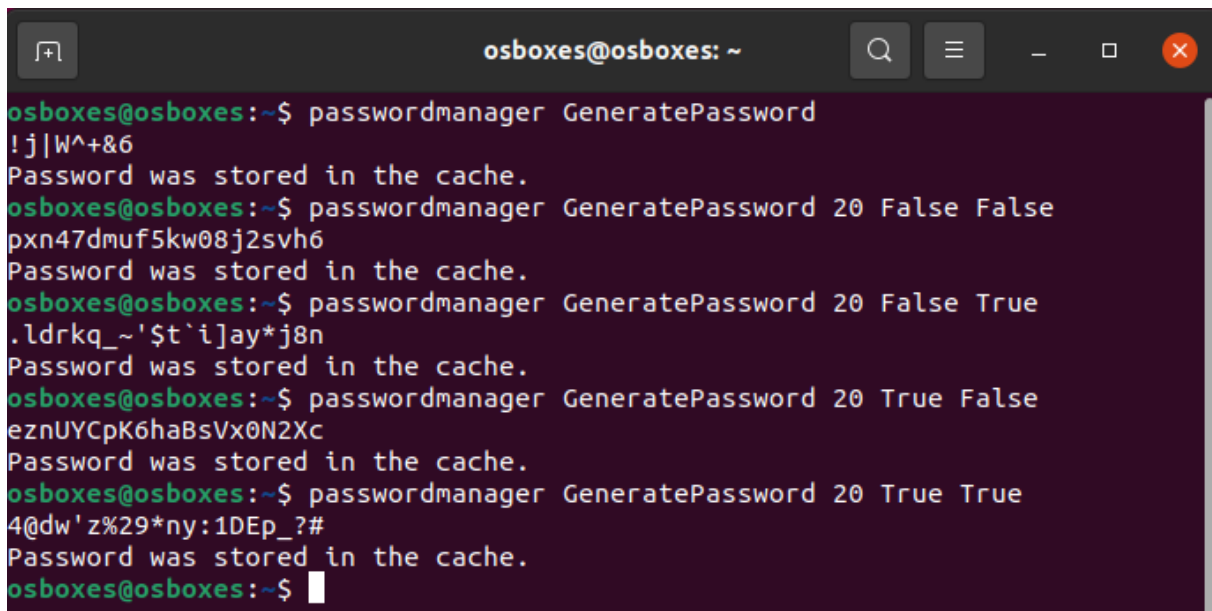
```
osboxes@osboxes: ~  
osboxes@osboxes:~$ passwordmanager Copy MyMaster title4  
Enter Master-Password:  
Username: user4  
Password was stored in the cache for 30s.  
Password was delete from cache.  
osboxes@osboxes:~$
```

Im Hintergrund läuft folgendes Python Skript. Durch das Python Module „sys.argv“ können Argumente aus dem Terminal entnommen werden. Da das Python Skript in der Terminal Umgebung Parameter erhält, müssen diese Parameter durch das „sys.argv“-Module aufgefangen um im weiteren Schritten verarbeitet zu werden. Damit das Passwort genau 30 Sekunden lang im Cache gespeichert ist wird es mit dem Python Time Module pausiert.

```
elif cmd == "Copy":#### -MasterUser -title  
    master_user = sys.argv[2]  
    title = sys.argv[3]  
    password_master = getpass.getpass("Enter Master-Password: ")  
    verification_password = get_pass_master(master_user)  
    if verification_password == password_master:  
        username, password = get_user_pass_title(title)  
        print(f"Usuario: {username}")  
        pyperclip.copy(password)  
        print("Password was stored in the cache for 30s.")  
        time.sleep(30)  
        pyperclip.copy("")  
        print("Password was delete from cache.")  
    else:  
        print("Wrong Master-Password")
```

Befehl 5 Das Generieren eines zufälligen Passworts

Mit diesem Befehl soll der Benutzer oder die Benutzerin ein neues Passwort generieren, welches auch komplett zufällig ist. Dazu kann der Benutzer oder die Benutzerin die Länge des Passwortes, ob Groß und Klein Buchstaben verwendet werden sollen und ob Sonderzeichen verwendet werden sollen entscheiden. Wenn der Benutzer oder die Benutzerin keine Angaben macht, wird immer ein acht Zeichen langes Passwort generiert, welches Groß- und Kleinschreibung und Sonderzeichenerhalten kann. Danach wird das generierte Passwort im Zwischenspeicher gelagert und kann wieder benutzt werden. Um diesen Befehl erfolgreich zu nutzen müssen die Eingaben wie folgt aussehen.

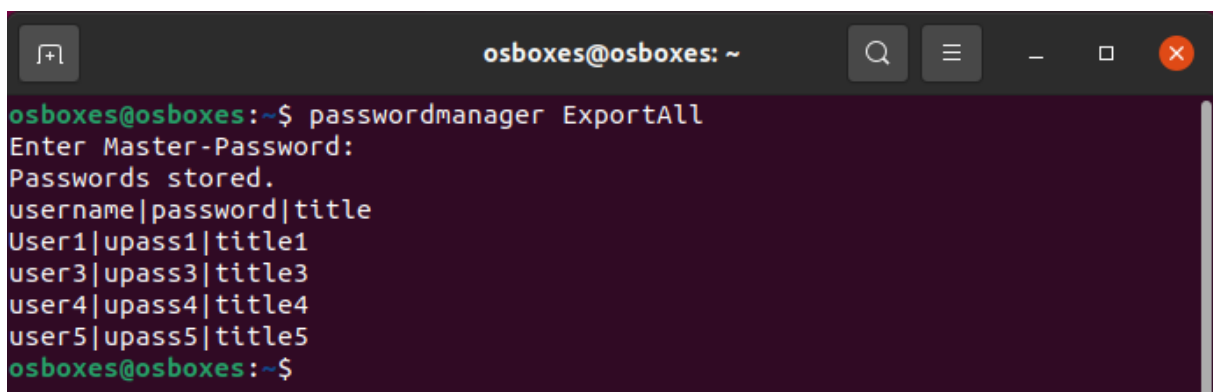
A terminal window titled 'osboxes@osboxes: ~' with search, menu, and window control icons. It shows five sequential uses of the 'passwordmanager GeneratePassword' command with varying parameters (length, uppercase, lowercase, special characters). Each command outputs a random password and a confirmation message: 'Password was stored in the cache.'

```
osboxes@osboxes:~$ passwordmanager GeneratePassword
!j|W^+&6
Password was stored in the cache.
osboxes@osboxes:~$ passwordmanager GeneratePassword 20 False False
pxn47dmuf5kw08j2svh6
Password was stored in the cache.
osboxes@osboxes:~$ passwordmanager GeneratePassword 20 False True
.ldrkq_~'$t`i]ay*j8n
Password was stored in the cache.
osboxes@osboxes:~$ passwordmanager GeneratePassword 20 True False
eznUYCpK6haBsVx0N2Xc
Password was stored in the cache.
osboxes@osboxes:~$ passwordmanager GeneratePassword 20 True True
4@dw'z%29*ny:1DEp_?#
Password was stored in the cache.
osboxes@osboxes:~$
```

Um die verschiedenen Möglichkeiten das Passwort zu generieren abzudecken, wurden verschiedene Try/Except Ausgänge implementiert. Nachdem alle Try/Except abfragen bearbeitet wurden, wird mithilfe von IF/Else Anweisungen überprüft ob die jeweiligen Ereignisse zutreffen. Nachdem der Befehl geschrieben wurde kann der Benutzer oder die Benutzerin die Länge des Passworts angeben. Der Zweite Parameter, ein Bool Parameter, gibt an ob die Groß und Kleinschreibung für das zu generierende Passwort verwendet werden soll. Der Dritte Parameter, ebenfalls ein Bool Parameter, gibt an ob für das zu generierende Passwort Sonderzeichen verwendet werden soll.

Befehl 6 Einträge als Tabelle anzeigen

Mit diesem Befehl ist es dem Benutzer oder der Benutzerin möglich alle Einträge, welche vorher im Passwort-Manager gespeichert wurden, als Tabelle auszugeben. Um hierfür zugriff zu erhalten ist eine Verifizierung durch das Masterpasswort zwingend notwendig. Der Grund hierfür ist die Ausgabe sämtlicher Zugangsdaten, die mithilfe einer Tabelle im Klartext angezeigt werden. Um diesen nutzen zu können muss der Benutzer oder die Benutzerin folgenden Befehl in den Terminal eingeben.



```
osboxes@osboxes: ~$ passwordmanager ExportAll
Enter Master-Password:
Passwords stored.
username|password|title
User1|upass1|title1
user3|upass3|title3
user4|upass4|title4
user5|upass5|title5
osboxes@osboxes:~$
```

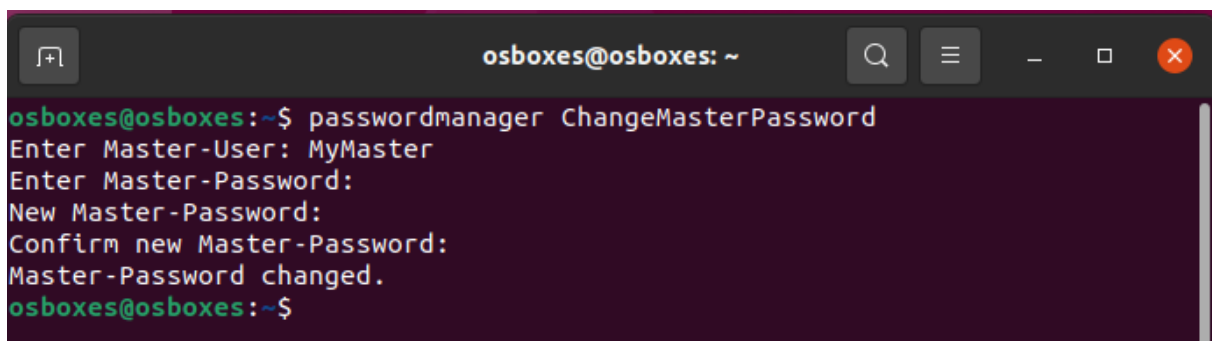
Das Python Skript für das Erstellen der Tabelle benutzt ebenfalls die Methode „master_pass_exist“ um zu überprüfen um das Masterpasswort übereinstimmt. Mithilfe der Funktion get_table_users() werden die Einträge aufbereitet und in der Terminalkonsole dargestellt.

```
elif cmd == "ExportAll":
    password = getpass.getpass("Enter Master-Password: ")
    verification_password = master_pass_exists(password)

    if verification_password:
        print("Passwords stored.")
        get_table_users()
    else:
        print("Wrong Master-Password")
```

Befehl 7 Das Masterpassword ändern

Mithilfe dieses Befehls kann der Benutzer oder die Benutzerin das Masterpassword beliebig ändern. Um das Passwort zu ändern, muss sich der Benutzer oder die Benutzerin mit dem Masterpassword verifizieren. Wird das Passwort bestätigt, so kann der Benutzer oder die Benutzerin sein eigenes Masterpassword nach belieben ändern. Beim Ändern ist es wichtig das der Benutzer oder die Benutzerin das Passwort richtig eingibt, denn sonst gibt es keine Möglichkeit das alte Passwort zu erneuern. Daraufhin muss der Benutzer oder die Benutzerin ein neues Passwort eingeben. Dieses wird noch einmal von der Funktion abgefragt um sicher zu gehen, dass das Passwort richtig ist. Wie es in den Anforderungen beschrieben wurde, werden die Passwörter beim Eingeben nicht im Klartext angezeigt. Laut den Anforderungen und der Analyse wurde entschlossen das der Befehl für das Ändern des Passwortes wie folgt aussieht:

A terminal window with a dark purple background. The title bar shows 'osboxes@osboxes: ~' and standard window controls. The terminal text is as follows:

```
osboxes@osboxes:~$ passwordmanager ChangeMasterPassword
Enter Master-User: MyMaster
Enter Master-Password:
New Master-Password:
Confirm new Master-Password:
Master-Password changed.
osboxes@osboxes:~$
```

Nachdem das alte Masterpassword eingeben wurde, wird mithilfe einer IF Abfrage kontrolliert, ob der Benutzer oder die Benutzerin ein neues Passwort eingibt. Tritt der Fall ein wird mit einer weiteren IF Abfrage untersucht ob die Passwörter miteinander übereinstimmen. Sobald dies der Fall ist, wird die Funktion `change_pass_master()` abgerufen. Die Verifizierung erfolgt wie in allen Fällen mit der Funktion `master_pass_exist()`, deshalb wird sie nicht im folgenden Quellcode angezeigt:

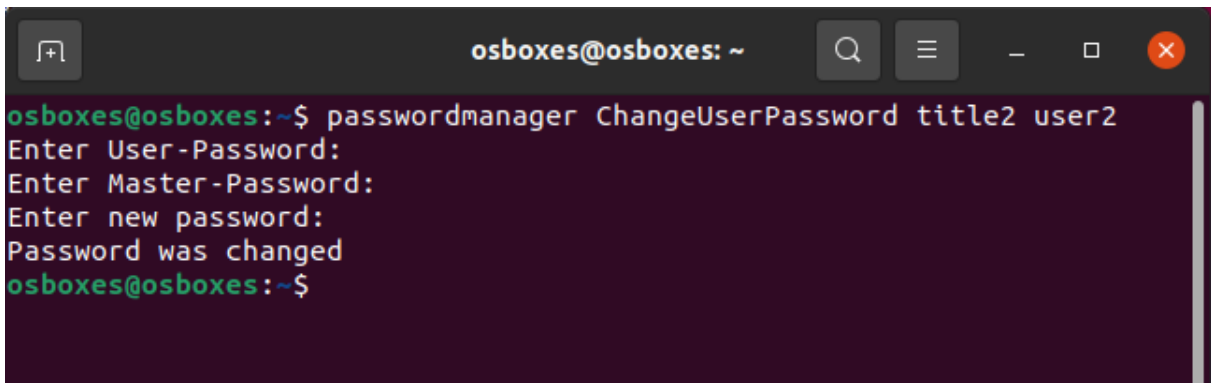
```
if verification_password:
    new_password = getpass.getpass("New Master-Password: ")
    confirm_new_password = getpass.getpass("Confirm new Master-Password: ")
    if new_password == confirm_new_password:
        change_pass_master(password, new_password)
        print("Master-Password changed.")
```

Mit dem Filereader, hier als fin deklariert, wird die Ablagedatei „Master_credentials“ geöffnet und in der Variable „data“ abgespeichert. Durch die Funktion „data.replace“, ist es in Python möglich Texte zu ersetzen. Der ersetzte Text wird daraufhin mit der fin.write überschrieben und gespeichert.

```
def change_pass_master(password_old, password_new):  
    fin = open(masters_credentials, "rt")  
    data = fin.read()  
  
    data = data.replace(password_old, password_new)  
    fin.close()  
    fin = open(masters_credentials, "wt")  
    fin.write(data)  
    fin.close()
```

Befehl 8 Das Ändern eines Eintrags

Mit diesem Befehl wird es dem Benutzer oder der Benutzerin möglich gemacht einen Eintrag im Passwort-Manager zu ändern. Nachdem der Benutzer oder die Benutzerin das Masterpasswort verifiziert hat, ist es ihm/ihr möglich einen Eintrag zu ändern. Mit dem Befehl „ChangeUserPassword“ muss der Benutzer oder die Benutzerin weitere Parameter eingeben. Der erste Parameter ist der Titel, der Zweite der Benutzername. Nach der Übergabe der Parameter, führt das Skript ein Dialog aus und überprüft die Zugangsdaten.



```
osboxes@osboxes:~$ passwordmanager ChangeUserPassword title2 user2
Enter User-Password:
Enter Master-Password:
Enter new password:
Password was changed
osboxes@osboxes:~$
```

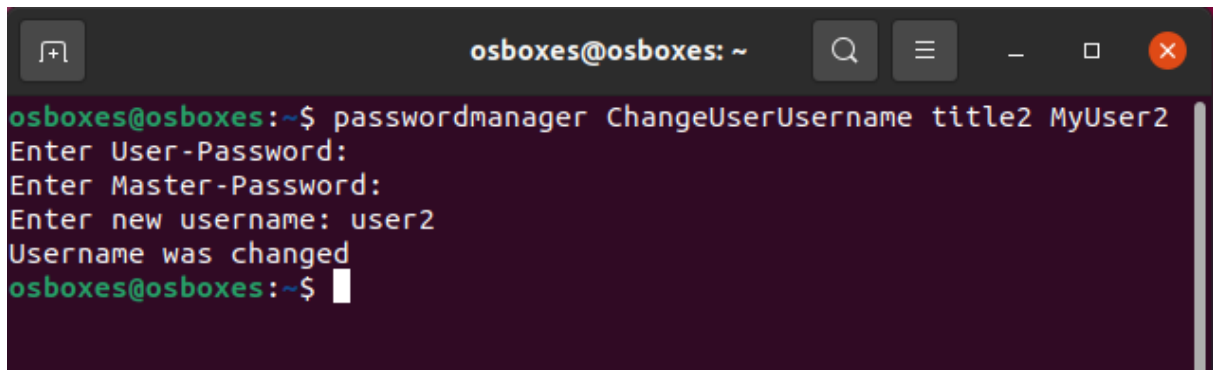
Die „ChangeUserPassword“ Funktion wurde ebenfalls mit dem Sys.argv Module implementiert. In diesem Fall sind die eingegebenen Parameter aus der Konsole entnommen, um die Daten mit den Zugangsdaten zu vergleichen.

```
elif cmd == "ChangeUserPassword": # -title -user
    title_user = sys.argv[2]
    username_user= sys.argv[3]
    password_old = getpass.getpass("Enter User-Password: ")
    password_master = getpass.getpass("Enter Master-Password: ")
    verification_password = master_pass_exists(password_master)

    if verification_password:
        password_new = getpass.getpass("Enter new password: ")
        if user_pass_exists(password_old):
            change_password_user (username_user, title_user, password_old,
password_new)
            print("Password was changed")
        else:
            print("Incorrect user information entered || User not found")
```

Befehl 8 Das ändern von einem Eintrag

Mit diesem Befehl wird es dem Benutzer oder der Benutzerin möglich gemacht einen Eintrag im Passwort-Manager zu ändern. Nachdem der Benutzer oder die Benutzerin das Masterpasswort verifiziert hat, ist es ihm/ihr möglich einen Eintrag zu ändern. Mit dem Befehl „ChangeUserUsername“ muss der Benutzer oder die Benutzerin weitere Parameter eingeben. Der erste Parameter ist der Titel, der Zweite der Benutzername. Nach der Übergabe der Parameter, führt das Skript ein Dialog aus und überprüft die Zugangsdaten.

A terminal window titled 'osboxes@osboxes: ~' with search, menu, and window control icons. The command 'passwordmanager ChangeUserUsername title2 MyUser2' is entered. The terminal shows a series of prompts: 'Enter User-Password:', 'Enter Master-Password:', and 'Enter new username: user2'. After the third prompt, the message 'Username was changed' is displayed, followed by the shell prompt 'osboxes@osboxes:~\$' and a cursor.

```
osboxes@osboxes:~$ passwordmanager ChangeUserUsername title2 MyUser2
Enter User-Password:
Enter Master-Password:
Enter new username: user2
Username was changed
osboxes@osboxes:~$
```

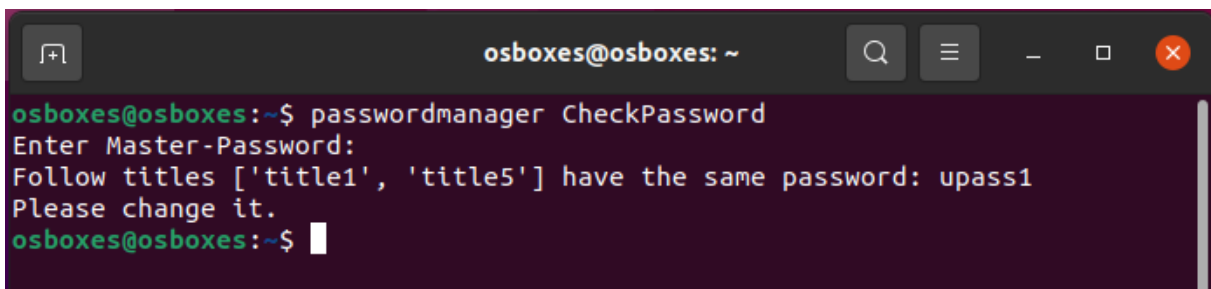
Die gleiche Herangehensweise verwendete man für das ändern von Benutzernamen.

```
elif cmd == "ChangeUserUsername": # -title -user
    title_user = sys.argv[2]
    username_user_old= sys.argv[3]
    password_user = getpass.getpass("Enter User-Password: ")
    password = getpass.getpass("Enter Master-Password: ")
    verification_password = master_pass_exists(password)

    if verification_password:
        username_user_new = input("Enter new username: ")
        if user_exists(username_user_old):
            change_username_user (username_user_old, title_user, password_user,
            username_user_new)
            print("Username was changed")
        else:
            print("Incorrect user information entered || User not found")
    else:
        print("Wrong Master-Password")
```

Befehl 9 Das Überprüfen von redundanten Passwörtern

Als Nützliche Zusatz Funktion war die Überprüfungen der Passwörter, ob diese sich wiederholen. Mithilfe von diesem Befehl soll der Benutzer oder die Benutzerin überprüfen können, ob mehrere Einträge mit demselben Passwort genutzt werden. Nachdem der Benutzer oder die Benutzerin den Befehl zum Überprüfen bestätigt, wird er/sie dazu aufgefordert das Masterpasswort anzugeben. Dies ist erforderlich, da die Passwörter im Klartext ausgegeben werden. Falls sich Passwörter wiederholen, werden die jeweiligen Titel angezeigt und der Benutzer oder die Benutzerin erhält die Benachrichtigung, die Passwörter zu wechseln.



```
osboxes@osboxes: ~  
osboxes@osboxes:~$ passwordmanager CheckPassword  
Enter Master-Password:  
Follow titles ['title1', 'title5'] have the same password: upass1  
Please change it.  
osboxes@osboxes:~$
```

Damit der Passwort-Manager die Passwörter überprüfen kann, wird die Funktion „get_user_repeat_pass()“ aufgerufen. Das Verifizieren wurde bereits in anderen Funktionen erwähnt.

```
elif cmd == "CheckPassword":  
    password = getpass.getpass("Enter Master-Password: ")  
    verification_password = master_pass_exists(password)  
  
    if verification_password:  
        get_user_repeat_pass()  
        print("Please change it.")
```

Die „get_user_repeat_pass()“ ist wie folgt aufgebaut:

```
def get_user_repeat_pass():  
  
    with open(users_credenciales, "r") as f:  
        next(f)  
        users = []  
        passwords = []  
        titles = []  
        for line in f:  
            line = line.strip()  
  
            user, password, title = line.split("|")  
            users.append(user)
```



```

        passwords.append(password)
        titles.append(title)

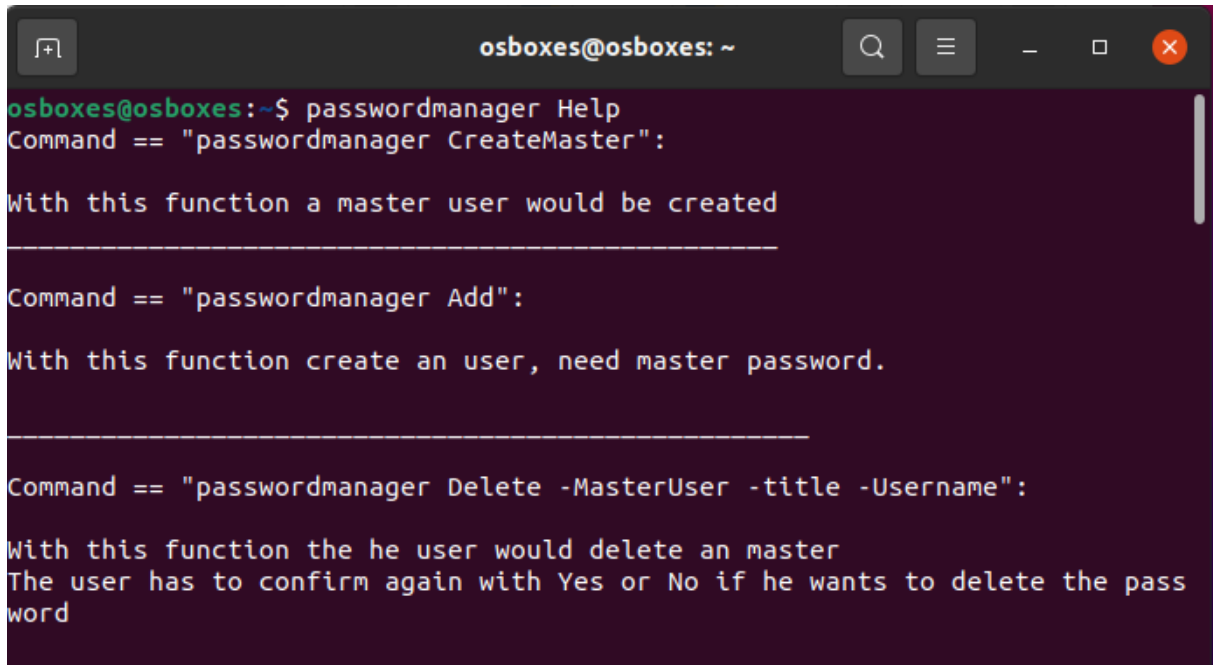
f.close()
list_set = set(passwords)
# convert the set to the list
unique_pass_list = (list(list_set))
for password_unique in unique_pass_list:
    index_pos_list = []
    for i in range(len(passwords)):
        if passwords[i] == password_unique:
            title_repeat = titles[i]
            index_pos_list.append(title_repeat)
    if len(index_pos_list) > 1:
        print(f"Follow titles {index_pos_list} have the same password:
{password_unique}")
    else:
        pass

```

Durch die Verwendung eines Filereaders werden die Zugangsdaten eingelesen und in einem Array gespeichert. Daraufhin werden die Zugangsdaten mithilfe zweier for-Schleifen in einem eigenen Array abgespeichert. Die Titel der sich wiederholenden Passwörter und das Passwort werden auf der Konsole aufgezeigt.

Befehl 9 Das Abrufen aller Funktionen

Als nützliche Zusatz Funktion kam die Überlegung dem Benutzer oder der Benutzerin eine Hilfestellung bereitzustellen. Die Idee basiert auf den typischen „Help“ Befehl, der in sämtlichen Windows und Linux Betriebssystemen vorhanden ist.

A terminal window titled 'osboxes@osboxes: ~' with standard window controls. The terminal shows the command 'passwordmanager Help' and its output. The output lists three commands: 'CreateMaster', 'Add', and 'Delete', each with a brief description of their function. The text is displayed in a light green monospace font on a dark purple background.

```
osboxes@osboxes:~$ passwordmanager Help
Command == "passwordmanager CreateMaster":

With this function a master user would be created

-----

Command == "passwordmanager Add":

With this function create an user, need master password.

-----

Command == "passwordmanager Delete -MasterUser -title -Username":

With this function the he user would delete an master
The user has to confirm again with Yes or No if he wants to delete the pass
word
```

Technisch realisiert wurde der „Help“ Befehl mit dem Python eigenen File-Reader. Der FileReader öffnet eine Bereits erstelle Text Datei. In der Text Datei befinden sich alle Befehle und eine Kurzbeschreibung. Im folgenden Quellcode befindet sich die Funktion, die hinter dem Help Befehl steckt.

```
def get_commands_information():

    with open(commands_information) as f:
        contents = f.read()
        print(contents)
    f.close()
```