Dokumentation Passwort Manager

Navigation

Reaktion auf Enter Taste:

Durch die Enter-Taste wird an gewünschten Stellen eine Eingabe der Tastatur ermöglicht. Diese lässt sich durch erneutes Enter drücken wieder deaktivieren.

Beim erstmaligen Drücken an einer eingestellten Stelle werden die Pfeiltasten deaktiviert, sie geben zwar noch einen Wert wieder, sind aber unterschiedliche Werte zu davor, durch die sich das Programm ihre Auswahl bewegt oder Ebenen wechselt.

Die Enter-Taste wird auch als Bestätigung z.B. für Erstellen als Account benutzt. Hierbei werden nicht die Pfeiltasten ausgelöst, sondern z.B. die Eingabedaten des erstellten Accounts validiert.

Durch das Drücken der Enter-Taste an einer Eingabe-Stelle wird der TypeMode auf True gesetzt. Nun werden die Tastendrücke in einer Variable gespeichert. Die Auswahl welcher dieser Variablen es ist, wird über layer und chooseRow entschieden. Ein Löschen der Eingabe über die Rücktaste (Backspace) ist möglich.

Reaktion auf Pfeiltasten:

Mit den Links/Rechts Pfeiltasten können die ersten Ebenen nach vorne oder nach Hinten durchlaufen werden.

Mit oben/Unten wird der Zeiger entlang der angezeigten Möglichkeiten verschoben.

Verifikation des Passwortes: Falls eine Änderung einer Variablen für Passwörter geschieht. Setzt dies auch ein Boolean auf True, wodurch dieses Passwort auf die Stärke, Überprüfung der Anzahl von wie oft die 5 ersten Hashwerte gepawned wurden und ob die Kriterien erfüllt sind, überprüft.

Klassen

User

Allgemein:

Die Klasse speichert den Benutzernamen und das Master Password, während der Laufzeit des Programms. Das Passwort wird vor dem Speichern gehashed.

Funktionen:

- *__init__:* setzt das Passwort und den Benutzernamen mit den Funktionen setUsername und hashPassword
- *setUsername:* setter für den Benutzernamen
- *generat* hashPassword*:* hashed das Passwort

Scores:

Mypy: keine Fehler

Pylint: 10/10

Unittest Coverage: 100%

Encryption

Allgemein:

Verschlüsselt und entschlüsselt die Passwörter

Funktionen:

- *__init__:* setzt das Backend und das masterPasswort.
- *deriveKey:* generiert den deriveKey, mithilfe des masterPassword und returned den deriveKey
- *encrypt:* verschlüsselt den übergebenen Parameter mit dem vorhandenem Key.
- *decrypt:* entschlüsselt den übergebenen Parameter mit dem vorhandenem Key

Scores:

Mypy: keine Fehler

Pylint: 10/10

Unittest Coverage: 100%

Storage

Allgemein:

Die Klasse Storage verwaltet die JSON-Datei zur Speicherung der Passwörter. Es können Passwörter erstellt, geändert und gelöscht werden. Es können alle Einträge eines Benutzers eingelesen werden und bei einem einzelnen Passwort alle Informationen übergeben werden.

Funktionen:

- *__init__:* setzt den filepath und ruft die Funktion loadData auf
- *loadData:* Die Funktion liest die JSON-Datei aus und speichert die Informationen in self.data
- *saveData:* Speicher die Datei ab
- *addPasswort:* Bekommt username, name, password, url, notes, category und datetime. Strukturiert die Daten und speichert sie mit saveData ab.
- *getEntry:* Liest einen konkretes Passwort aus und gibt alle Informationen strukturiert zurück
- *getAllEntryes:* Gibt alle Namen der Passwörter des aktuellen Benutzers zurück
- *updatePassword:* Ändert ein Passwort und speichert das alte Passwort in der History
- *deletePassword:* löscht das Passwort, was als Parameter übergeben wurde.

Scores:

Mypy: keine Fehler

Pylint: 10/10

Unittest Coverage: 96%

PasswordManager

Allgemein:

Die Klasse PasswordManager ist die Verwaltungseinheit des Accountsystems. Über sie können alle Account Funktionen bedient werden.

Funktionen:

- *__init__:* Setzt den Benutzer. Erstellt Instanzen von Encryption und Storage und speichert diese lokal ab.
- *existingAccountValid:* Kontrolliert, ob Benutzername und Masterpassword richtig eingegeben wurden. Falls die Daten korrekt sind, wird ein True zurückgegeben, sonst ein False.
- *newAccountValid:* schaut ob bereits ein Account mit diesem Username existiert. Falls bereits ein Account existiert, wird ein False zurück gegeben sonst ein True.
- *addEntry:* Verschlüsselt alle übergebenen Daten und gibt diese ans storage zur speicherung.
- *getEntry:* Liest aus dem Storage das gesuchte Passwort aus und entschlüsselt sie mit der Klasse Encrypton. Am Ende wird das Dictionary retuned.
- *updateEntry:* Ändert ein vorhandenes Passwort.
- *deleteEntry:* Löscht ein vorhandenes Passwort.
- *getAllEntryes:* gibt alle Passwörter eines Benutzers zurück. Nur der Name der Passwörter.
- *getAllEntryesByUrl: gibt alle Entryes von einer URL zurück*
- *getAllEntryesByUsername: gibt alle Entryes mit einem bestimmten Namen zurück.*

Scores:

Mypy: keine Fehler

Pylint: 10/10

Unittest Coverage: 3% (Import Fehler verhindert Testen)

PasswordGenerator

Allgemein:

Die Klasse generiert die Passwörter. Man kann anfangs entscheiden, ob man Großbuchstaben, Zahlen und Sonderzeichen miteinbeziehen kann. Außerdem kann die Länge des Passworts bestimmt werden. Es gibt die Möglichkeit, bestimmte Zeichen auszuschließen. Durch die HaveIBeenPwned API kann bestimmt werden, ob das Passwort schon einmal gehackt wurde, wenn nicht wird durch die Passwortlänge und der Anzahl der verschiedenen Zeichen eine Passwortstärke ermittelt. Das Passwort wird bei der Erstellung so überprüft, dass es auch die gewünschten Zeichen enthält.

Funktionen:

- *__init__:* setzt die klasseninternen Variablen auf die übergebenen Werte .
- *initialSettings:* erstellt die Zeichenkette, anhand welcher das Passwort generiert wird.
- *excludeChars:* schneidet die gewünschten Zeichen aus der bei initialSettings erstellten Zeichenkette aus.
- *haveIBeenPwned:* prüft, ob das Passwort gehackt wurde. Dazu wird das Passwort als SHA1-Hash verschlüsselt und die ersten fünf Zeichen des Hashs an die HaveIBeenPwned API gesendet. Diese gibt eine Liste mit den Resten der Passwörter zurück, die mit denselben fünf Zeichen anfangen. Es wird überprüft, ob die das gegebene Passwort dort vorhanden ist und wie oft es gehackt wurde. Falls es nicht in der Liste auftaucht, wurde das Passwort noch nicht gehackt.
- *passwordSafety:* berechnet die Stärke des Passworts anhand der Länge des Passworts und der Anzahl der verschiedenen Zeichen (Formel: Passwortlänge * ln(Anzahl der verschiedenen Zeichen)). Daran wird ermittelt, ob das Passwort schwach („weak“), mittel („OK“) oder stark („strong“) ist. Wurde das Passwort aber schon mal gehackt wird dies nicht gemacht.
- *containsEverything:* prüft, ob die gewüschten Zeichen (Großbuchtstaben, Zahlen, Sonderzeichen) auch in dem generierten Passwort enthalten sind.
- *generate:* erstellt das Passwort zufällig. Es wird so lange ausgeführt, bis das Passwort alle. gewünschten Anforderungen entspricht.

Scores

Mypy: Keine Fehler

Pylint: 10/10

Unittest Coverage: 96%

Interface:

Allgemein:
Diese Klasse dient der Eingabe über Tastatur sowie die Ausgabe über das Terminal. Um dies zu verwirklichen, benutzt es die Klasse Curses. Um die Reihenfolge zu behalten, besitzt Interface zwei Variable. Einmal layer, welches die Ebene/Seite der gerade dargestellten Seite beschreibt und chooseRow, welches die ausgewählte Zeilennummer verwirklicht.

Funktionen

- **__init__**: Die Seitenreihenfolge und Zeilen werden fest geladen und die Klasse Passwordtgenerator wird initialisiert.
- **showList**: Diese Funktion erhält die nächste Seite als Integer und gibt diese gewünschte Seite auf dem Terminal wieder. Dafür liest es die Liste für diese Seite ein und gibt zudem noch die Länge der Einträge wieder, um die Auswahl über Pfeiltasten im sinnvollen Bereich zu halten
- **extractData:** Um eine ausgewählte Plattform mit ihren Daten wie Kategorie oder Notizen darzustellen, benötigt es diese Funktion, welche aus dem erhaltenen Dictionary die Einträge an der richtigen Stelle in der Liste für die Seite einfügt.

Scores

Mypy: 1 Fehler

Pylint: 10/10

Unittest Coverage: 1% (Import Fehler verhindert Testen)