Projekt für Betriebssystem und Rechnernetz

Passwort Manager

Zusammenarbeit von:

- **AKRITI AKRITI**
- **ANNE EMMANUELLE ATEBA NDOUMOU MISSODI**
- **UMESH DAHAL**

GLIEDERUNG:

Tools:

- Python-IDLE Shell
- MySQL workbench

In unser Projekt geht es darum, ein Programm zu erstellen, das ermöglicht einen Benutzer mehrere Passwörter auf unterschiedliche Webseiten zu verwalten.

WIE KANN MAN AUF DEN PASSWORT MANAGER ZUGREIFEN?

I. Master Passwort erstellen

1. Externaler Sicht

- Der erste Schritt ist es für den Benutzer ein Master Passwort zu erzeugen. Dabei wird gefördert ein Passwort erst mal zu schreiben und dann zu wiederholen.
- Sind die beide Eingaben entsprechend, dann wird auf dem Bildschirm “Master password successfully created” angezeigt und das Master Passwort ist hergestellt und gespeichert.
- Wenn das Passwort nicht exakt wiederholt wurde, bekommt den Benutzer die Meldung “Pass didn’t match” und soll er dann ein neues Passwort erstellen.

2. Konzeptueller Sicht

- Zuerst haben wir das Modul „stdiomask“ importiert. Es setzt das Passwort (Eintrag des Benutzers) in Sternchen (***) um.
- Dann haben wir die Methode getpass() genutzt, sodass es keine Eingabe auf dem Bildschirm gezeigt wird.
- Wir haben die beide in den Variablen “master1” und “master2” folgendermaßen kombiniert:

```
master1 = stdiomask.getpass('make a new Master pass:', '*')
master2 = stdiomask.getpass('Enter again  :', '*')
```

- Eine Variable (master1) für die erste Eingabe und die andere (master2) für die Wiederholung.
- Diese Kombination ermöglicht es dem Benutzer sein Passwort einzugeben, ohne dass es im Klartext angezeigt wird.
- Mit einer If-Schleife stellen wir sicher, dass die beide Eingabe entsprechen. Wenn das der Fall ist, wird auf der Konsole mithilfe der Funktion “print()” (wird für alle Eingaben genutzt, die auf der Konsole angezeigt werden müssen) “Master password successfully created” angezeigt.
- Falls die Eingaben nicht gleich sind, ergibt sich auf der Konsole eine Fehlermeldung, und zwar “Pass didn’t match”. Dann kann den Benutzer ein neues Passwort erstellen.

II. Master Passwort eingeben

1. Externaler Sicht

- Sobald das Master Passwort erstellt ist, wird es noch einmal gefördert, um auf die verschiedenen Funktionalitäten des Passwort Managers zuzugreifen. Das geschieht aber nur, wenn die Eingabe richtig ist.
- Bei falscher Eingabe kriegt den Benutzer die Fehlermeldung “Master password is not correct. You have __ tries left ” und hat ein Versuch weniger.
- Wenn der dritte Versuch erfolglos ist, hat den Benutzer eine letzte Alternative. Es wird eine persönliche Frage gestellt, die nur den Benutzer beantworten kann. Diese Frage wird nach Erstellung des Master Passwort vom Benutzer bereits beantwortet.
- Wäre die Antwort korrekt, dann hat den Benutzer die Möglichkeit sein Master Passwort zu ändern.

2. Konzeptueller Sicht

- Es wird zuerst die Variable “sec_info” hergestellt, die eine persönliche Frage enthält und in der die Eingabe der Benutzer gespeichert wird.
- Dann wird in der Variablen “in_pass”, werden das Modul “stdiomask” und die Methode “getpass()” kombiniert, sodass das Passwort (erzeugte Master Passwort) nicht angezeigt wird. Diese Variable lagert die Eingabe der Benutzer.
- In der Variablen “mp_count” und mithilfe einer while-Schleife wird die Anzahl der Versuche gezählt.
- Ein if-Statement prüft, ob das Passwort korrekt oder nicht ist. Dabei wird die Variable mit dem gespeicherten Passwort mit der neuen Eingabe vergleicht.

☺ Bei richtiger Eingabe

- Mit der Methode “print()” wird “Welcome” und das ganze Menü auf der Konsole angezeigt.
- Eine Variable “Option” wird erstellt, damit der Benutzer seine Optionswahl eingeben kann.

☹ Bei falscher Eingabe

- Der Benutzer bekommt eine Fehlermeldung wie folgt

```
print ("Master password is not correct. You have "+ mp_count +" tries left")
```

- Wenn der Benutzer kein Versuch mehr hat, dann wird in der Variablen “sec_info_check” die Antwort auf die persönliche Frage gespeichert. Diese Antwort wird mithilfe einer If-Schleife mit der ursprünglichen Antwort vergleicht.
- Wenn die Eingaben entsprechen, dann kann der Benutzer ein neues Master Passwort erstellen. Sonst wird er immer wieder nach der richtigen Antwort gefragt. So gescheht das Ganze:

```
if mp_count < 0:
  in_menu = 0
else:
  sec_info_input = 11


  while sec_info_input == 11:

    sec_info_check = Input ("enter your first crush name: ")

    if sec_info_check != sec_info:
       sec_info_input = 11

        else:
     sec_info_check = ""
     in_menu = ""
     check_in_mas = 1
     break
```

WAS KANN MAN MIT DEM PASSWORT MANAGER TUN?

III. Auswahl von Optionen

Sobald das Passwort korrekt ist, werden die verschiedene Funktionalitäten unseres Passwort Managers vorgestellt. Dabei soll der Benutzer nach seinen Bedürfnissen, eine davon auswählen. Im Folgenden werden Sie die verschiedenen Funktionalitäten unseres Passwort-Managers entdecken.

→ Eintrag speichern

1. Externaler Sicht

- Wenn der Benutzer einen neuen Eintrag speichern möchtet, soll er die Option 1 auswählen
- Innerhalb dieser Option wird die Erstellung von Benutzername, Titel (Name der Website) und Passwort gefördert.
- Außerdem gibt es für den Benutzer die Möglichkeit ein Passwort selbst zu erstellen beziehungsweise es vom Programm generieren zu lassen.
- Wenn der Benutzer bevorzugt es, sein Passwort selbst zu erzeugen, kann er einfach die Option “Manual” auswählen. Er wird dann sein Passwort eingeben und es speichern.
- Bei selbst Erstellung des Passwortes weist dem Benutzer das Programm darauf hin, ob das Passwort schwach, normal oder stark ist.
- Wenn das Passwort enthält nur alphabetische Zeichen, ist es als schwach betrachtet. Der Benutzer bekommt die Meldung “Password level: WEAK”.
- Wenn es alphabetische und numerische Zeichen enthält, die Meldung ist “Password level: NORMAL”.

- Wenn es alphabetische und numerische Zeichen sowie Sonderzeichen enthält, die Meldung ist “Password level: STRONG”
- Wenn der Benutzer möchtet sein Passwort automatisch generieren lassen, dann soll er einfach “auto-generate” auswählen.
- Der Benutzer wird eine Länge für sein Passwort vorschreiben. Er kann auch die Groß/-Kleinschreibung, Sonderzeichen und Symbole angeben. Das Passwort wird danach generiert und gespeichert.
- Hier kann das Passwort nicht mehr als 12 Zeichen und nicht weniger als 8 Zeichen enthalten
- Nach der Speicherung entscheidet den Benutzer, ob er noch einmal einen Eintrag speichern oder zurück nach Menü umgeleitet werden will.

2. Konzeptueller Sicht

- In der Variable Option gibt den Benutzer seine Wahl aus. Danach stellen wir mit einem If-Statement sicher, dass der Benutzer nach einem Benutzername, Titel und Passwort gefragt wird.
- Es wird die Klasse “activity_list” erzeugt mit den Attributen “title”, “website”, “username” und “password”
- Mithilfe der Funktion “input()”, werden in Variablen “title”, “website” und “user” Eingabe des Benutzers gelagert. Was das Passwort betrifft, kann der Benutzer entscheiden: In der Variablen “pass_op” kann er 1 für eine manuelle Erstellung und 2 für eine automatische Generierung eingeben.
- Bei der manuellen Erstellung des Passwortes, wird in der Variablen “password” das Modul “stdiomask” und die Funktion “getpass()” kombiniert. Diese Variable enthält die Eingabe der Benutzer. Nach der Erstellung erhält des Benutzers die Meldung “Your password is saved”.
- Mit einer for-/ und if-Schleife wird den Stand des Passwortes geprüft. Dazu haben das Modul “string” importiert.
- Wenn “password.isalpha()” ist TRUE, dann ist das Passwort als “WEAK” betrachtet. Wenn “password.isalnum” ist TRUE, dann ist es als “NORMAL” betrachtet. Und wenn beide FALSE sind, dann ist das Passwort “STRONG”.
- Bei der automatischen Generierung des Passwortes:

*Es wird das Modul “random” importiert

*Der Benutzer soll in der Variablen “length” die gewünschte Länge für sein Passwort eintragen

*Es wird mithilfe einer While-Schleife sichergestellt, dass das Passwort kleiner gleich 12 und größer gleich 8 ist. Sonst wird auf der Konsole die Fehlermeldung “***Invalid length***” angezeigt.

*Was der Inhalt des Passwortes betrifft, hat der Benutzer die Wahl zwischen Option 1 für Groß-/Kleinschreibung sowie Symbole und Ziffern; Option 2 für Klein- und Großschreibung; und Option 3 für Symbolen und Ziffern. Es sieht wie folgt aus:

Option 1:

```
password_option == 1:
    password_characters = string.ascii_letters + string.digits +
string.punctuation
    password = ''.join(random.choice(password_characters) for i in
range(length))
```

- Option 2:

```
password_option == 2:
    password_characters = string.ascii_letters
    password = ''.join(random.choice(password_characters) for i in
range(length))
```

Option 3:

```
password_option == 3:
    password_characters = string.digits + string.punctuation
    password = ''.join(random.choice(password_characters) for i in
range(length))
```

- Mit einer while-Schleife haben bieten wir die Möglichkeit zu entscheiden, ob man einen Eintrag noch einmal speichern will, indem man “yes” eingibt oder zu dem Hauptmenü gehen will, indem man irgendwelchen Eintrag eingibt

❖ Einträge Lagerung

1. Externaler Sicht

Als der Benutzer seine Einträge eingegeben hat, wird er durch verschiedene Wege (Dazu kommen wir noch später) in der Lage sein, auf seine Daten reibungslos zuzugreifen. Diese Einträge sind in einer Tabelle gesammelt und die Anzahl der Einträge wird ständig gezählt.

2. Konzeptueller Sicht

- In unser Programm wird eine Datenbank verbunden, in der wir alle Daten abgelegt haben. Dazu haben wir die Module “mysql.connector” und “PrettyTable” importiert.
- Wir haben die Funktion “TableList.append()” mit der Klasse “activity_list” wie folgt verbunden:

```
tableList.append(activity_list (title, website, user, password)).
```

- In der Variablen “tab1.field_names” steht die Liste von Attributen (Spaltenüberschrift) in der Tabelle und in der Variablen “tab1.add_row” steht die Liste der Variablen, deren Werte in jede Spalte gespeichert werden.
- Mit der Syntax “mycursor=mydb.cursor” wird eine Datenbank erstellt und so kann man auch darauf zugreifen.
- In der Variablen “sql” steht den SQL-Befehl, der ermöglicht das Programm die Einträge in der Tabelle hinzuzufügen. In der Variablen “val” stehen die Variablen, deren Werte gespeichert werden müssen. Mit der Funktion “mycursor.execute()” wird den Befehl ausgeführt.
- Mit der Funktion commit(), werden die Änderung in der Datenbank gespeichert.
- Die Variable “count” speichert die Summe alle Einträge.

→ Eintrag aktualisieren

Die gespeicherten Daten werden direkt von der Datenbank aktualisiert

1. Externaler Sicht

- Der Benutzer hat die Möglichkeit die Website, der Benutzername, das Passwort oder das Titel zu ändern, indem er die Option 2 auswählt.
- Er soll auch spezifizieren was genau er verändert möchtet zwischen Website, Titel, Benutzername und Passwort.

2. Konzeptueller Sicht

- Für jede mögliche Änderung haben wir in einer Variablen eine SQL-Anfrage zur Aktualisierung gespeichert in der Form

```
sql = "UPDATE activity SET website = %s WHERE website = %s"

sql = "UPDATE activity SET title = %s WHERE title = %s"

sql = "UPDATE activity SET username = %s WHERE username = %s"

sql = "UPDATE activity SET password = %s WHERE password = %s"
```

- Für jede mögliche Änderung wird es eine Variable geben, die die neue und alte Werte der Variablen zu verändern trägt.
- Es wird danach mithilfe der jeweiligen Funktionen “mycursor.execute()” und “mydb.commit()” die Befehle in der Datenbank ausgeführt und gespeichert.

→ Eintrag löschen

Genauso wie bei den anderen Optionen, wird die Option 3 direkt in der Datenbank durchgeführt.

1. Externaler Sicht

- Der Benutzer wird gefördert die Website, mit der Daten zu löschen, einzutragen.
- Es wird danach eine Bestätigung gefördert, indem der Benutzer das Master-Passwort eingibt. Dann werden diese Daten von der Datenbank gelöscht.

2. Konzeptueller Sicht

- Es wird in der Variablen “sql” eine SQL-Anfrage nach Löschung der Daten wie folgt gespeichert

```
sql = "DELETE FROM activity WHERE website = %s"
```

- In der Variablen “del_wb” wird die Eingabe der Benutzer gespeichert.
- In der Variablen “conform” gibt den Benutzer das Master Passwort ein.
- Wenn das Passwort korrekt ist, wird die Anfrage durchgeführt und gespeichert. Sonst bekommt den Benutzer die Meldung “Wrong password” und soll er es noch eingeben.

→ Eintrag durch Website suchen

1. Externaler Sicht

Mit der Option 4 hat den Benutzer die Möglichkeit seine Einträge zu suchen, indem er der Name der Website eingibt. Es wird sich das Passwort ergeben, dass die eingegebene Website entspricht.

2. Konzeptueller Sicht

- In der Variablen “sql” wird die Anfrage an der Datenbank wie folgt gespeichert

```
sql = "SELECT * FROM activity WHERE website = %s "
```

- In den Variablen “wb” wird die Eingabe des Benutzers gespeichert
- Mit der Funktion “mycursor.execute” werden die Variablen “sql” und “wb” kombiniert und die Anweisung durchgeführt.
- Mit der Funktion “mycursor.fetchall()” werden alle Tupel von der Tabelle selektiert und ausgegeben.

→ Eintrag durch Benutzername suchen

1. Externaler Sicht

Mit der Option 5 kann der Benutzer sein Passwort durch sein Benutzername. Er wird nur gefördert der Benutzername einzutragen, den dieses Passwort entspricht.

2. Konzeptueller Sicht

- In der Variablen “sql” wird die Anfrage an der Datenbank wie folgt gespeichert

```
sql = "SELECT * FROM activity WHERE title = %s "
```

- In den Variablen “tl” wird die Eingabe des Benutzers gespeichert
- Mit der Funktion “mycursor.execute” werden die Variablen “sql” und “tl” kombiniert und die Anweisung durchgeführt.
- Mit der Funktion “mycursor.fetchall()” werden alle Tupel von der Tabelle selektiert und ausgegeben.

→ Alle Einträge ansehen

1. Externaler Sicht

Wenn der Benutzer Option 6 auswählt, wird eine Tabelle mit allen gespeicherten Daten auf der Konsole ausgegeben. In dieser Tabelle hat auch den Benutzer die Möglichkeit, seine Passwörter im Klartext anzusehen. Es ist besonders sinnvoll bei den Passwörtern die automatisch generiert wurden. Die Anzahl der Einträge kann auch der Benutzer immer sehen.

2. Konzeptueller Sicht

- In der Variablen Option gibt den Benutzer ein, sein ausgewählte Optionsnummer (6 in diesem Fall)
- In der Methode “mycursor.execute()” wird die Anfrage an der Datenbank wie folgt gegeben:

```
mycursor.execute("SELECT * FROM activity")
```

- Mit der Methode “fetchall()” wird alle Tupel von der Datenbank selektiert und der Variable “myresult” gespeichert
- Für jede neue Eintrag wird es in der Tabelle hinzugefügt wie folgt:

```
for x in myresult:
    tab.add_row(x)
print(tab.get_string(title="Activities"))
```

- Die Funktion “add_row()” ermöglicht das Programm neue Tupel in der Tabelle hinzuzufügen
- Die Methode get_string() ermöglicht die Tabelle “Activities” zu nennen.
- Zur Vermeidung von Wiederholungen bei Veränderung der Tabelle wird die Methode “tab.clear_rows()” genutzt.
- Mithilfe der Methode “rowcount()” können wir prüfen, ob die Tabelle nicht leer ist:

```
if mycursor.rowcount == 0:
    print("there are no records")
```

- Wenn es leer ist, dann bekommt den Benutzer die Meldung “there are no records”. Sonst wird auf der Konsole gezeigt wie viel Einträge in der Tabelle gespeichert sind.

→ Master Passwort ändern

1. Externaler Sicht

Der Benutzer hat mit der Option 7 die Möglichkeit beliebig häufig sein Master-Passwort zu ändern. Dazu soll er sein aktuelles Passwort zuerst eingeben und dann sein neues Passwort eintragen und wiederholen. Wenn die beiden Eingaben entsprechen, wird das Passwort erfolgreich geändert werden, sonst muss der Benutzer den Prozess wiederholen. Sonst kann er den Prozess wiederholen.

2. Konzeptueller Sicht

- Es soll in der Variablen “ask7” das aktuelle Passwort eingegeben. Die Eingabe muss korrekt sein, sonst wird dem Benutzer immer noch nach dem aktuellen Master Passwort gefragt.
- Es wird dieselbe Kombination gemacht für die Variablen, die das neue Passwort und die Wiederholung dieses Passwort tragen
- Mithilfe eins If-Statements wird geprüft, ob die Passwörter entsprechen. Wenn ja wird es sich mit “print()” eine Bestätigung auf der Konsole ergeben.
- In den Variablen “mas_chg1” und “mas_chg2” werden jeweilig das neue Passwort und die Wiederholung gespeichert. Wenn die Eingaben nicht gleich sind, dann muss den Benutzer ein neues Passwort noch mal erstellen. Ansonsten bekommt den Benutzer die Meldung “Masterpass changed successfully”

→ Aktivität beenden

1. Externaler Sicht

Mit der Option 8 kann den Benutzer dem Passwort Manager beenden. Falls er noch darauf zugreifen möchtet, dann solltet er wieder sein Master Passwort eingeben.

2. Konzeptueller Sicht

- In der Variablen “Option” wird Nummer 8 eingetragen. Das führt dazu, dass der Benutzer außerhalb das Passwort Manager geschickt wird. Da soll er sich wieder anmelden, indem er die Zugangsdaten eingibt.

AUS WELCHEN BESONDEREN EIGENSCHAFTEN BESTEHT DAS PASSWORT MANAGER?

IV. Häufigkeit der Abfrage des Passwortes

1. Externaler Sicht

In unserer Anwendung wird das Master Passwort nicht nur einmal angefordert, sondern mit einer Benutzerdefinierte Frequenz.

Das heißt am Anfang des Programms wird dem Benutzer gefragt, wie oft er wünscht, dass das Master Passwort angefordert wird. Abhängig von der Eingabe des Benutzers wird das Master-Passwort repetitiv angefordert.

2. Konzeptueller Sicht

- Es wird zuerst das Modul “datetime” importiert.
- Es wird in der Variablen “zeit_festlegen” das Input des Benutzers gespeichert.
- In der Variablen “date_now” ist mithilfe der Funktion “datetime.now()” die aktuelle Zeit angegeben.
- In der Variablen “date_change" wird die Funktion “timedelta()” um die vom Benutzer eingegebene Zeit in Minuten zu konvertieren.
- In der Variablen “date_after” wird die Summe der Variablen “date_now” und ”date_change” berechnet. Und auf der Konsole werden die Variablen “date_now” und “date_after” angezeigt.
- Mit einer If-Schleife wird geprüft, ob die Variable “date_now” größer als die Variable “date_after” ist. Wenn das der Fall ist, dann wird auf der Konsole “Time over” angezeigt und der Benutzer sollte das Master-Passwort nochmal eingeben.

V. Speicherung des Passwortes im Zwischenablage

Es werden die Passwörter, die im Programm gespeichert sind in einem Zwischenablage gelagert

1. Externaler Sicht

Sobald das Passwort im Programm gespeichert ist, wird es im Zwischenablage automatisch kopiert. Der Benutzer kann dann auf der Login-Seite der gewünschten Website das Passwort einfach einfügen.

2. Konzeptueller Sicht

- Zuerst importieren wir das Modul “pyperclip”. Es ist ein Modul, das als Plattform zum Kopieren und Einfügen im Zwischenablage gilt.
- Die Variable “myresult” dank der Funktion “fetchall()” enthält alle Einträge der Datenbank
- Mit einer for-Schleife werden alle Passwörter in dem Zwischenablage gespeichert. Es geschieht wie folgt:

```
for x in myresult:
print ("Password is copied to clipboard!!!")
pyperclip.copy(x[4])
```

- In unserer Datenbank gibt es 4 Attributen und Passwort steht auf der 4. Stelle
- Mit der Funktion “pyperclip.copy()” werden alle Einträge, die sich auf der 4. Stelle befinden, im Clipboard gespeichert.