In [1]:

```
#!/usr/bin/python

import os
import os.path
import time
import base64
import sqlite3
import pyscrypt
from getpass import getpass
from mega import Mega
from argon2 import PasswordHasher
from Crypto.Cipher import ChaCha20_Poly1305


# Functions

def encrypt(pw, mstr, alt):
    header = b"header"
    pw = pw.encode('utf-8')

    # Password Key Derivation Function with master and second factor password
    key = pyscrypt.hash(mstr, alt, 8192, 8, 1, 32)

    ch = ChaCha20_Poly1305.new(key=key)
    ch.update(header)
    chpw, tag = ch.encrypt_and_digest(pw)

    # Encode results from bytes to Base64
    nonce = base64.b64encode(ch.nonce).decode('utf-8')
    header = base64.b64encode(header).decode('utf-8')
    tag = base64.b64encode(tag).decode('utf-8')
    chpw = base64.b64encode(chpw).decode('utf-8')

    # Return nonce, header, tag and encrypted password
    return nonce, header, tag, chpw


def decrypt(nonce, header, tag, chpw, mstr, alt):

    # Decode results from Base64 to bytes
    nonce = base64.b64decode(nonce.encode('utf-8'))
    header = base64.b64decode(header.encode('utf-8'))
    tag = base64.b64decode(tag.encode('utf-8'))
    chpw = base64.b64decode(chpw.encode('utf-8'))

    # Password Key Derivation Function with master and second factor password
    key = pyscrypt.hash(mstr, alt, 8192, 8, 1, 32)

    ch = ChaCha20_Poly1305.new(key=key, nonce=nonce)
    ch.update(header)
    enpw = ch.decrypt_and_verify(chpw, tag)

    return enpw


def clean():
    os.system('cls' if os.name == 'nt' else 'clear')


def login(db1, alt):
    row = db1.cursor()

    row.execute("""SELECT hash1 FROM Login;""")

    data = row.fetchall()
    row.close()

    try:
        PHs = PasswordHasher()
        PHs.verify(data[0][0], alt)
        print("\nLogin successful..")
        tag = input("\n\nPress Enter to continue...")
    except:
        print("\n!!The password does not match the supplied hash!!\n\nGoodbye...")
        time.sleep(1)
        clean()
        exit()


def exist(servn, usern, dbx):
    cursor = dbx.cursor()
    cursor.execute(
        "SELECT ROWID FROM Hashes WHERE(service='%s' AND username='%s');" % (servn, u
    data = cursor.fetchone()
    cursor.close()

    if data == None:
        return False
    return True


# Main


def main():

    op = ""         # option
    master = ""     # master password
    salt = ""       # second factor password used has salt
    db = ""         # database

    print("Python Password Manager\n")

    if not os.path.exists("keystorage.db"):
        db = sqlite3.connect("keystorage.db")

        # Table Hashes - will be used to store the saved passwords
        db.execute("""CREATE TABLE Hashes(
            service  STRING,
            username STRING,
            nonce    STRING,
            header   STRING,
            tag      STRING,
            chpw     STRING);""")

        # Table Login - will be used to store the hashed (argon2-cffi) versions of th
        db.execute("""CREATE TABLE Login(
            hash1    STRING);""")

        db.commit()

        # Login credentials
        print("""\n\nNo suitable database, a new one will be created, set the master p

        master = (getpass("Encryption password: ")).encode('utf-8')
        salt = (getpass("Login password: ")).encode('utf-8')

        PHs = PasswordHasher()
        db.execute("""INSERT INTO Login (hash1) VALUES('%s');""" %
                   (PHs.hash(salt)))

        db.commit()

    else:
        db = sqlite3.connect("keystorage.db")

        # Login credentials
        master = (getpass("Encryption password: ")).encode('utf-8')
        salt = (getpass("Login password: ")).encode('utf-8')

        login(db1, salt)

    while True:
        clean()

        print("Python Password Manager\n")
        op = int(input(
            "> 1 - Add password\n> 2 - Retrive password\n> 3 - Edit password\n> 4 - L

        clean()

        # > 1 - New password to add to the database
        if op == 1:
            print("New password\n")
            servv = input("Service name: ")
            user = input("Username: ")
            pw = getpass("Password to be stored: ")

            nonce, header, tag, chpw = encrypt(pw, master, salt)

            # Inserts new line into the database
            db.execute("""INSERT INTO Hashes (service, username, nonce, header, tag,
                       VALUES('%s', '%s', '%s', '%s', '%s', '%s');""" % (servv, user, nonce,

            db.commit()

            input("\nYour password has been stored !\n\n(Press Enter to continue...)")

        # > 2 - Retrive the password the user wants to see
        elif op == 2:
            print("Retrive password\n")
            servv = input("Service name: ")
            user = input("Username: ")

            if exist(servv, user, db):
                # Retrieves data from the database searching for service and username
                row = db.cursor()

                row.execute("""SELECT nonce, header, tag, chpw from Hashes
                    WHERE (service='%s' AND username='%s');""" % (servv, user))

                data = row.fetchall()
                row.close()

                decpw = decrypt(data[0][0], data[0][1], data[0]
                                [2], data[0][3], master, salt)

                print("\nYour password is: %s\n" % decpw.decode('utf-8'))

            else:
                print("\nService non existent")

            input("\n(Press Enter to continue...)")

        # > 3 - Edit Password
        elif op == 3:
            print("Edit password\n")
            servv = input("Service name: ")
            user = input("Username: ")

            if exist(servv, user, db):

                row = db.cursor()
                row.execute("""SELECT nonce, header, tag, chpw from Hashes
                    WHERE (service='%s' AND username='%s');""" % (servv, user)
                data = row.fetchall()
                row.close()

                decpw = decrypt(data[0][0], data[0][1], data[0]
                                [2], data[0][3], master, salt)

                print("\nYour password is: %s\n" % decpw.decode('utf-8'))

                op1 = int(input(
                    "> 1 - Change service name\n> 2 - Change username\n> 3 - Change pa

                str = "\nChanges saved !"

                if op1 == 1:
                    nuserv = input("\nNew service: ")
                    db.execute(
                        "UPDATE Hashes SET service='%s' WHERE (service='%s' AND usern

                elif op1 == 2:
                    nuser = input("\nNew username: ")
                    db.execute(
                        "UPDATE Hashes SET username='%s' WHERE (service='%s' AND user

                elif op1 == 3:
                    npw = getpass("\nNew password: ")

                    nonce, header, tag, chpw = encrypt(npw, master, salt)

                    db.execute(
                        "UPDATE Hashes SET nonce='%s', header='%s', tag='%s', chpw='%

                elif op1 == 4:
                    db.execute(
                        "DELETE from Hashes WHERE (service='%s' AND username='%s');" %

                elif op1 == 5:
                    str = "No changes were applied"

                db.commit()
                print(str)

            else:
                print("\nService non existent")

            input("\n(Press Enter to continue...)")

        # > 4 - List all services
        elif op == 4:
            print("List all services\n")

            row = db.cursor()
            row.execute("SELECT ROWID, service, username FROM Hashes")
            data = row.fetchall()
            row.close()

            for x in data:
                print("ID\t\t%s" % (x[0]))
                print("Service name:\t%s" % (x[1]))
                print("Username:\t%s\n" % (x[2]))

            input("\n(Press Enter to continue...)")

        elif op == 0:
            break

    db.close()


if __name__ == "__main__":
    mega = Mega()
    main()
```

```
Python Password Manager

Encryption password: ·····
Login password: ·····

Login successful...

(Press Enter to continue...)
Python Password Manager

> 1 - Add password
> 2 - Retrive password
> 3 - Edit password
> 4 - List all services
> 0 - Exit
> 1
New password

Service name: Youtube
Username: test_0
Password to be stored: ·····

Your password has been stored !

(Press Enter to continue...)
Python Password Manager

> 1 - Add password
> 2 - Retrive password
> 3 - Edit password
> 4 - List all services
> 0 - Exit
> 2
Retrive password

Service name: Youtube
Username: test_0

Your password is: 12345


(Press Enter to continue...)
Python Password Manager

> 1 - Add password
> 2 - Retrive password
> 3 - Edit password
> 4 - List all services
> 0 - Exit
> 3
Edit password

Service name: Youtube
Username: test_0

Your password is: 12345

> 1 - Change service name
> 2 - Change username
> 3 - Change password
> 4 - Delete service
> 0 - Go back
> 3

New password: ·····

Changes saved !

(Press Enter to continue...)
Python Password Manager

> 1 - Add password
> 2 - Retrive password
> 3 - Edit password
> 4 - List all services
> 0 - Exit
> 4
List all services

ID:             1
Service name:   Youtube
Username:       test_0


(Press Enter to continue...)
Python Password Manager

> 1 - Add password
> 2 - Retrive password
> 3 - Edit password
> 4 - List all services
> 0 - Exit
> 0
```

In [ ]:

In [ ]: