SQLITE3

SQLite3 הוא מנוע sql פשוט מאוד לשימוש. הוא מסד נתונים עצמאי, חסר שרת, בלי צורך באיזשהי קונפיגורציה מיוחדת ונייד. הוא מנוע מהיר וקל מאוד, וכל תכולת המסד נשמרת בקובץ בודד. הוא משמש בהרבה מאוד אפליקציות כמסד אחסון פנימי. ה-sqlite בנוי על השפה פייתון והוא חלק מהספרייה הסטנדארטית שלה כך שאין צורך להוריד שום מודול מיוחד כדי להשתמש בו. המודול מבוסס sql אז כל השאילתות שניתן לבצע ב-sql יהיו זהות במסד זה.
במסמך הבא נתמקד האופן השימוש ב-sqlite ולא בפקודות sql, לעוד מידע על שאילתות sql מומלץ להתסכל באתר w3schools בנושא sql בקישור הבא.

יצירת מסד חדש-

שימוש במודול יהיה ע"י ייבוא של הספרייה sqlite3.
בשביל להתחבר נצטרך להשתמש בפונקציה connect() שאמורה לקבל את שם המסד כארגומנט.
הפונקציה יכולה לשמור את כל הנתונים בזיכרון (ב-ram) לאורך כל התוכנית, או ממש לשמור את זה בקובץ '.db' , בשביל לשמור בזיכרון נשלח לפונקציה את המחרוזת ':memory:' ,ואם נרצה לשמור במסד נתונים ספציפי נצטרך לתת את ה-path לאותו מסד, אם הוא לא קיים הפונקציה תיצור אותו. בסוף כל שימוש במסד נתונים נצטרך גם לסגור את המסד.

```
import sqlite3
db = sqlite3.connect('my_database.db')
db.close()
```

כדי לבצע איזושהי פקודה במסד נתונים נצטרך אובייקט שנקרא cursor.
ה-cursor מבצע פקודות sql על המסד עם הפונקציה execute() שמקבלת מחרוזת עם הפקודת sql שאנחנו רוצים לבצע. הפקודה הראשונה שנבצע תהיה ליצור טבלה חדשה נקרא לה users עם העמודות id, name, phone, email ו-password אחרי הפקודה נבצע את הפונקציה commit() של המסד שמעדכנת אותו.
בשביל לזרוק את הטבלה נפצע את פקודות ה-drop sql ונעדכן את המסד :

```
# Get a cursor
cursor = db.cursor()
# Creates new table
cursor.execute('''
    CREATE TABLE users(id INTEGER PRIMARY KEY, name TEXT,
                       phone TEXT, email TEXT unique, password TEXT)
''')
db.commit()
# To drop the table
cursor.execute("DROP TABLE users")
db.commit()
```

הכנסה של משתנים חדשים תעשה ע"י הcursor ואם נרצה להוסיף משתנים מתוך משתנים שהגדרנו בתוכנית נשתמש המחרוזת '???' כדי להכניס אותם (ולא בכל מיני אופרציות של פייתון):

```
name1 = 'Tom'
phone1 = '3366858'
```

```
email1 = 'Tom.Pythonovitz@example.com'
# A very secure password
password1 = '12345'

name2 = 'Tammi'
phone2 = '5557241'
email2 = 'TammiPaythonovitz@example.com'
password2 = 'TammiLoveTom'

# Insert user 1
cursor.execute('''INSERT INTO users(name, phone, email, password)
                  VALUES(?,?,?,?)''', (name1,phone1, email1, password1))
print('First user inserted')

# Insert user 2
cursor.execute('''INSERT INTO users(name, phone, email, password)
                  VALUES(?,?,?,?)''', (name2,phone2, email2, password2))
print('Second user inserted')
db.commit()
```

דרך נוספת להעביר נתונים היא עם מילון, ואז בשאילתא נכניס את המפתח עם נקודתיים לפני:

```
name3 = 'George'
phone3 = 'Rustniovsky'
email3 = 'GRust@example.com'
password3 = 'Rust forever'

# Insert user 3
cursor.execute('''INSERT INTO users(name, phone, email, password)
                  VALUES(:name,:phone, :email, :password)''',
                  {'name':name3, 'phone':phone3, 'email':email3, 'password':password3})
print('Third user inserted')
db.commit()
```

וכמובן אם יש לנו כמה משתמשים ונרצה להכניס את כולם, נוכל בצורה דומה להכניס רשימה של אובייקטים, אבל בשביל זה צריך להשתמש בפונקציה executemany() :

```
users = [('Joee', 'Javany','joo@example.mail','password'),
        ('Shirel', 'Cplustik','cpp@example.mail','cppass'),
        ('Adam', 'Kotlinernberg','Adam_Kotlin@example.mail','JustPassword')
        ]
# Insert many
cursor.executemany('''INSERT INTO users(name, phone, email, password)
                  VALUES(?,?,?,?)''', users)
print('Many users inserted')
db.commit()
```

שאילתות-

בשביל לבצע שאילתות נשתמש באותו cursor , רק כדי לקבל את התוצאה של השאילתא נצטרך להשתמש במתודת fetch כלשהי של ה-cursor. יש כמה סוגים של פונקציות fetch יש fetchone שמחזירה רק את השורה הראשונה מהשאילתא; fetchmany(number) שמקבלת כארגומנט כמה שורות ומחזירה אותם; ויש את fetchall() שמחזירה את כל השורות של המסד. המתודות מחזירות רשימה של פייתון כשכל אובייקט ברשימה הוא tuple עם התכונות של אותו משתמש שהכנסנו. לכן נוכל להשתמש בלולאת for כדי לסגנן את הפלט:

```
cursor.execute('''SELECT name, email, phone FROM users''')
all_rows = cursor.fetchall()
print()
for row in all_rows:
    # row[0] returns the first column in the query (name), row[1] returns email column.
    print('{0} : {1}, {2}'.format(row[0], row[1], row[2]))
```

```
_____________________________________________
Tom Pythonovitz : Tom.Pythonovitz@example.com, 3366858
Tammi Pythonovitz : Tammi@example.com, 5557241
George Rustniovsky : GRust@example.com, 33333
Joee Javany : joo@example.mail, 2222
Shirel Cplustik : cpp@example.mail, 3333
Adam Kotlinernberg : Adam_Kotlin@example.mail, 4444
```

ה-cursor עובד גם כאיטרטור ואפשר לגשת אליו ישירות:

```
cursor.execute('''SELECT name, password FROM users''')
for row in cursor:
    print(f'{row[0]} : {row[1]}')
```

```
_____________________________
Tom Pythonovitz : 12345
Tammi Pythonovitz : TammiLoveTom
George Rustniovsky : Rust for ever
Joee Javany : password
Shirel Cplustik : cppass
Adam Kotlinernberg : JustPassword
```

גם כדי לקבל מידע נשתמש ב-'?' וניתן כארגומט את המשתנה:

```
cursor.execute('''SELECT name, email, phone
                  FROM users WHERE name=?''',(name1,) )
print(cursor.fetchall())
```

ובאותו אופן גם עדכון ומחיקה של נתונים מהטבלה:

```
# Update user with id 1
newphone = '3113093164'
userid = 1
cursor.execute('''UPDATE users SET phone = ? WHERE id = ? ''',
 (newphone, userid))
# Delete user with id 2
delete_userid = 2
```

```
cursor.execute('''DELETE FROM users WHERE id = ? ''', (delete_userid,))
db.commit()

print()
cursor.execute('''SELECT name,phone  FROM users''')
for index ,row in enumerate(cursor):
    print(f'{index} ) {row[0]} : {row[1]}')
```

```
0 ) Tom Pythonovitz : 3113093164
1 ) George Rustniovsky : 33333
2 ) Joee Javany : 2222
3 ) Shirel Cplustik : 3333
4 ) Adam Kotlinernberg : 4444
```

ה-commit() היא פקודה מאוד חשובה, היא שומרת על האטומיות של המסד נתונים, היא מורה לבצע את כל הפקודות האחרונות שהתבצעו ע"י ה-cursor על המסד. אם לא ביצענו commit() אחרי רצף פקודות execute() השינויים לא ישמור, אבל לאחר ה-commit() כל השינויים האחרונים ישמרו, לכן במקרה שביצענו פקודה ואנחנו לא רוצים שהיא תתבצע בסוף במסד, למשל מחקנו עמודה ובדיעבד הבנו שלא רצינו , אם התמזל מזלנו ולא ביצענו commit() עדיין, נוכל להשתמש בפקודה rollback() והיא תאפס את ה-cursor לנקודה של ה-commit האחרון שהתבצע:

```
cursor.execute('''UPDATE users SET phone = ? WHERE id = ? ''',
("121212", userid))
# The user's phone is not updated
db.rollback()
```

רשימת טיפוסי הנתונים של sqlite :

- `None` type is converted to `NULL`
- `int` type is converted to `INTEGER`
- `float` type is converted to `REAL`
- `str` type is converted to `TEXT`
- `bytes` type is converted to `BLOB`