Βάσεις Δεδομένων και Python

Είδαμε σε προηγούμενα μαθήματα ότι με την χρήση αρχείων μπορούμε να αποθηκεύσουμε μόνιμα στον δίσκο πληροφορίες. Τα αρχεία μπορούν να διαβαστούν, να αλλαχθούν και να συμπληρωθούν από άλλα προγράμματα. Όταν η κλίμακα μεγέθους της πληροφορίας είναι μεγάλη χρειάζεται να γίνει χρήση συστήματος βάσης δεδομένων.

Υπάρχουν πολλοί τύποι βάσεων δεδομένων. Σε αυτό το μάθημα θα δούμε μία σχεσιακή βάση δεδομένων και πως μπορούμε να εκτελέσουμε τις βασικές λειτουργίες με χρήση της γλώσσας Python, όπως: δημιουργία βάσης δεδομένων, κατασκευή πίνακα, προσθήκη στοιχείων, αναζήτηση στοιχείων κλπ.

Μετά την εγκατάσταση της Python έχουμε την δυνατότητα να χρησιμοποιήσουμε χωρίς να χρειάζεται εγκατάσταση μία απλή, αλλά πλήρη σχεσιακή βάση δεδομένων. Λέγεται SQLite και είναι διαθέσιμη προγραμματιστικά με ένα απλό import.

Δημιουργία βάσης

Η SQLite δεν χρειάζεται προ-εγκατάσταση ούτε κάποια υπηρεσία ή server να τρέχει για να μπορεί να χρησιμοποιηθεί. Η βάση δημιουργείται σαν ένα μοναδικό αρχείο, στον φάκελο στον οποίο τρέχουμε το πρόγραμμα που εμφανίζεται στο παράδειγμα 1, εάν δεν υπάρχει ακόμα. Εάν υπάρχει ανοίγει με το connect.

Παράδειγμα 1: Κώδικας για την δημιουργία βάσης:

```
import sqlite3
con=sqlite3.connect('testdb.db')
```

Δημιουργία πίνακα

Παράδειγμα 2: Κώδικας για τη δημιουργία πίνακα στη βάση:

```
import sqlite3
con=sqlite3.connect('testdb.db')
```

```
cursor=con.cursor()
cursor.execute('create table phones'
               '(id integer primary key, name text, phone text)')
```

Εισαγωγή στοιχείων στον πίνακα

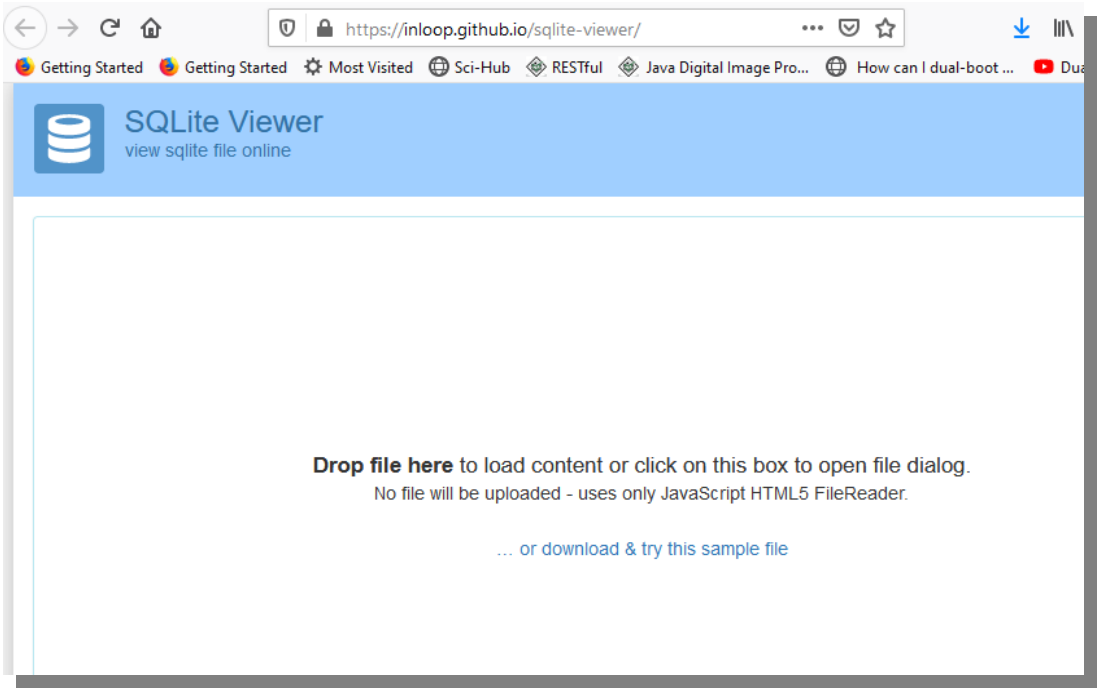
Έχουμε δημιουργήσει μια βάση στο αρχείο 'testdb.db', που περιέχει ένα πίνακα που λέγεται phones.

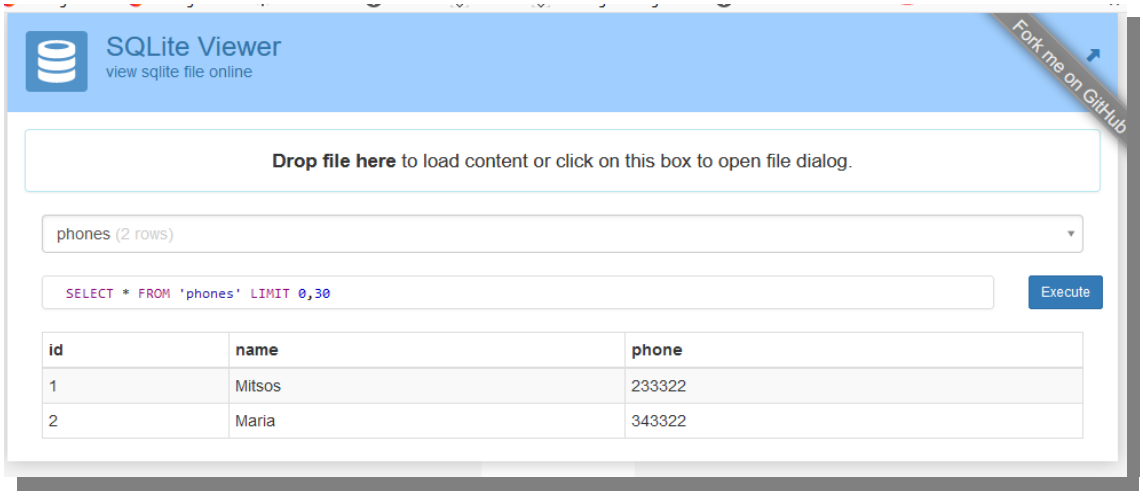
Παράδειγμα 3: Κώδικας για τη εισαγωγή στοιχείων στη βάση:

```
import sqlite3
con=sqlite3.connect('testdb.db')

cursor=con.cursor()
cursor.execute('insert into phones values(?, ?, ?)',
                                  (1, 'Mitsos', '233322'))
cursor.execute('insert into phones values(?, ?, ?)',
                                  (2, 'Maria', '343322'))
con.commit()
```

Ένας εύκολος τρόπος για να δούμε τα περιεχόμενα της βάσης είναι να κάνουμε drop το αρχείο που την περιέχει στην εφαρμογή που βρίσκεται στο link sqlite-viewer.


https://inloop.github.io/sqlite-viewer/
Getting Started
Getting Started
Most Visited
Sci-Hub
RESTful
Java Digital Image Pro...
How can I dual-boot ...
SQLite Viewer
view sqlite file online
Drop file here to load content or click on this box to open file dialog.
No file will be uploaded - uses only JavaScript HTML5 FileReader.
... or download & try this sample file


SQLite Viewer
view sqlite file online
Fork me on GitHub
Drop file here to load content or click on this box to open file dialog.
phones (2 rows)
SELECT * FROM 'phones' LIMIT 0,30
Execute
id
name
phone
1
Mitsos
233322
2
Maria
343322

Εισαγωγή στοιχείων στον πίνακα με auto-increment id

Μπορούμε να προσθέσουμε στοιχεία στη βάση χωρίς να δηλώνουμε εμείς το id. Αυτό γίνεται όταν χρησιμοποιήσουμε insert με όνομα στήλης:

Παράδειγμα 4: Κώδικας για τη εισαγωγή στοιχείων στη βάση με auto-increment κλειδί:

```
import sqlite3
con=sqlite3.connect('testDB.db')

cursor=con.cursor()
cursor.execute('insert into phones(name, phone) values(?, ?)',
                                            ('Stelios', '434662'))
cursor.execute('insert into phones(name, phone) values(?, ?)',
                                            ('Antonia', '66722'))
con.commit()
```

Το αντίθετο του commit() είναι το rollback().

Παράδειγμα 5: Κώδικας για τη εισαγωγή στοιχείων στη βάση με autoincrement κλειδί. Το insert με το rollback() ακυρώνεται:

```
import sqlite3
con=sqlite3.connect('testDB.db')

cursor=con.cursor()
cursor.execute('insert into phones(name, phone) values(?, ?)',
                                            ('Takis', '202992'))
con.rollback()
cursor.execute('insert into phones(name, phone) values(?, ?)',
                                            ('Eleni', '43212'))
con.commit()

cursor.close()
con.close()
```

Το rollback δεν επέτρεψε την εγγραφή του Τάκη.

```
SELECT * FROM 'phones' LIMIT 0,30
```

id	name	phone
1	Mitsos	233322
2	Maria	343322
3	Stelios	434662
4	Antonia	66722
5	Eleni	43212

Ανάγνωση στοιχείων από τον πίνακα

Παράδειγμα 6: Κώδικας που χρησιμοποιεί την **fetchone()** για πρόσβαση στη βάση σε όλα τα στοιχεία γραμμή-γραμμή. Το tuple με όνομα row περιέχει όλα τα στοιχεία κάθε γραμμής:

```
import sqlite3

con=sqlite3.connect('testDB.db')
cursor = con.cursor()

cursor.execute('SELECT * FROM phones')

row = cursor.fetchone()
while row!=None:
    id=str(row[0])
    name=row[1]
    phone=row[2]
    print("{}, {}, {} ".format(id, name, phone))
    row = cursor.fetchone()

cursor.close()
con.close()
```

Παράδειγμα 7: Κώδικας για να διαβάσουμε όλα τα στοιχεία από τη βάση:

```
import sqlite3
con=sqlite3.connect('testDB.db')

cursor = con.cursor()
query = ('SELECT id, name, phone FROM phones')
cursor.execute(query)

for (id, name, phone) in cursor:
  print("{}, {}, {} ".format(id, name, phone))

cursor.close()
con.close()
```

=== RESTART: C:\Users\costi\Dropbox\TEI\AdvancedSoftEng\Py\databases\read2.py ==
1, Mitsos, 233322
2, Maria, 343322
3, Stelios, 434662
4, Antonia, 66722
5, Eleni, 43212
>>>

Παράδειγμα 8: Κώδικας για τη αναζήτηση στοιχείων στη βάση με χρήση συγκεκριμένου κλειδιού. Ζητήσαμε την εγγραφή με κλειδί 4. Η fetchone() επιστρέφει μία γραμμή, τα στοιχεία της οποίας βρίσκονται σε tuple και είναι προσβάσιμα βάσει του δείκτη του tuple με όνομα row:

```
import sqlite3

con=sqlite3.connect('testDB.db')
cursor = con.cursor()

cursor.execute('SELECT * FROM phones where id=?', (4,))

row = cursor.fetchone()

id=str(row[0])
name=row[1]
```

```
phone=row[2]


print("{}, {}, {} ".format(id, name, phone))

con.commit()
```

Παράδειγμα 9: Κώδικας για τη αναζήτηση όλων των στοιχείων της βάσης με χρήση του fetchone(). Ζητήσαμε όλες τις εγγραφές. Η fetchone() επιστρέφει μία γραμμή, τα στοιχεία της οποίας βρίσκονται στο tuple row και επαναλαμβάνοντας την fetchone διαβάζουμε την επόμενη μέχρι να τελειώσουν οι εγγραφές:

```
import sqlite3

con=sqlite3.connect('testDB.db')
cursor = con.cursor()

cursor.execute('SELECT * FROM phones')

row = cursor.fetchone()
while row!=None:
    id=str(row[0])
    name=row[1]
    phone=row[2]
    print("{}, {}, {} ".format(id, name, phone))
    row = cursor.fetchone()

cursor.close()
con.close()
```

Παράδειγμα 10: Κώδικας για τη αναζήτηση όλων των στοιχείων της βάσης με χρήση του fetchall(). Ζητήσαμε όλες τις εγγραφές. Η **fetchall()** επιστρέφει μία λίστα με όλες τις γραμμές που λέγεται allrows, από την οποία ζητάμε κάθε row και τυπώνουμε τα στοιχεία κάθε γραμμής:

```
import sqlite3

con=sqlite3.connect('testDB.db')
cursor = con.cursor()
```

```
cursor.execute('SELECT * FROM phones')

allrows = cursor.fetchall()
for row in allrows:
    print("{}, {}, {} ".format(row[0], row[1], row[2]))

cursor.close()
con.close()
```

Παράδειγμα 11: Εάν δηλώσουμε con.row_factory=sqlite3.Row, τότε το κάθε row αντί για tuple γίνεται αντικείμενο sqlite3.Row. Στο αντικείμενο αυτό μπορούμε αντί για τον δείκτη που είχαμε πριν να χρησιμοποιήσουμε το όνομα της κάθε στήλης:

```
import sqlite3

con=sqlite3.connect('testDB.db')
con.row_factory = sqlite3.Row

cursor = con.cursor()

cursor.execute('SELECT * FROM phones')

allrows = cursor.fetchall()
for row in allrows:
    print("{}, {}, {} ".format(row['id'], row['name'], row['phone']))

cursor.close()
con.close()
```

Πρόσβαση στη MySQL

Για να μπορέσουμε να συνδεθούμε στην MySQL χρειάζεται να έχουμε εγκαταστήσει την εφαρμογή MySQL-Connection-Python. Επειδή η MySQL λειτουργεί με users, passwords και URL πρέπει να τα ορίσουμε. Αυτό γίνεται στο connect. Εκτός από τις δύο πρώτες γραμμές που διαφέρουν από τα προγράμματα που χρησιμοποιήσαμε για να δουλέψουμε με την SQLite υπάρχει και διαφορά στον τρόπο που δημιουργούνται οι βάσεις δεδομένων.

Παράδειγμα 12: Κώδικας που διαβάζει στοιχεία πίνακα από βάση δεδομένων MySQL.

```
import mysql.connector
con = mysql.connector.connect(user='root', password='zoot',
                              host='127.0.0.1',
                              database='tax2011')

cursor = con.cursor()
query = ('SELECT idexpenses, eamount, supplier FROM expenses'
         ' WHERE supplier=101')

cursor.execute(query)

for (idexpenses, eamount, supplier) in cursor:
  print("{}, {}, {} ".format(idexpenses, eamount, supplier))

cursor.close()
con.close()
```