Python – Data Persistence

Αρχές Γλωσσών Προγραμματισμού και Μεταφραστών

Data Persistence

- Αρχεία csv – ανάγνωση και εγγραφή
- Pickle – shelves Lilbraries
- Η κωδικοποίηση JSON
- Σχεσιακές βάσεις δεδομένων
- NoSQL βάσεις δεδομένων

Αρχεία CSV 1/2

- Τα αρχεία CSV χρησιμοποιούνται για την αποθήκευση μεγάλου αριθμού μεταβλητών ή δεδομένων - το περιεχόμενο αποθηκεύεται σε απλό κείμενο.

- Το module CSV είναι μια ενσωματωμένη λειτουργία που επιτρέπει στην Python να αναλύει αυτούς τους τύπους αρχείων.

Αρχεία CSV 2/2

- Το κείμενο μέσα σε ένα αρχείο CSV παρατίθεται σε σειρές και κάθε ένα από αυτά έχει στήλες, όλες διαχωρισμένες με κόμματα.

- Κάθε γραμμή του αρχείου είναι μια σειρά ενώ τα κόμματα χρησιμοποιούνται για τον ορισμό και τη διαίρεση κελιών.

Ανάκτηση από αρχείο κειμένου

CSV

```
name,department,birthday month
John Smith,Accounting,November
Erica Meyers,IT,March
```

```
import csv

with open('employee_birthday.txt', mode='r') as csv_file:
        csv_reader = csv.DictReader(csv_file)
         line_count = 0
         for row in csv_reader:
                 if line_count == 0:
                         print(f'Column names are {", ".join(row)}')
                         line_count += 1
                 print(f'\t{row["name"]} works in the {row["department"]}
department, and was born in {row["birthday month"]}.')
                  line_count += 1
        print(f'Processed {line_count} lines.')
```

```
Column names are name, department, birthday month
John Smith works in the Accounting department, and was born in November. Erica Meyers works in the IT department, and was born in March.
Processed 3 lines.
```

Εγγραφή σε αρχείο κειμένου

CSV

```
name,address,date joined
john smith,1132 Anywhere,Jan 4
 erica meyers,1234 Smith,March 2
```

```
import csv

with open('employee_file.csv', mode='w') as employee_file:
employee_writer = csv.writer(employee_file, delimiter=',', quotechar='"',
quoting=csv.QUOTE_MINIMAL)

employee_writer.writerow(['John Smith', ‘Anywhere', ‘January'])
employee_writer.writerow(['Erica Meyers', ‘1234 Smith', 'March'])
```

Quoting CSV files

- ***csv.QUOTE_ALL*** - Quote everything, regardless of type.
- ***csv.QUOTE_MINIMAL*** - Quote fields with special characters
- ***csv.QUOTE_NONNUMERIC*** - Quote all fields that are not integers or floats
- ***csv.QUOTE_NONE*** - Do not quote anything on output

Pickle library

- χρήσιμο για εφαρμογές όπου χρειάζεται μόνιμη αποθήκευση στα δεδομένα.
- Τα δεδομένα μπορούν να αποθηκευτούν στο δίσκο

Τύποι δεδομένων

- Booleans
- Integers
- Floats
- Complex numbers
- (normal and Unicode) Strings
- Tuples
- Lists
- Sets
- Dictionaries with picklable objects

Παράδειγμα pickle σε λίστα 1/2

```
import pickle

a = ['test value','test value 2','test value 3’]

file_Name = "testfile"
# open the file for writing
fileObject = open(file_Name,'wb')
```

Παράδειγμα pickle σε λίστα 2/2

```
# this writes the object a to the # file named 'testfile'
pickle.dump(a,fileObject)

# here we close the fileObject
 fileObject.close()
# we open the file for reading
fileObject = open(file_Name,'r’)
# load the object from the file into var b
b = pickle.load(fileObject)
```

Παράδειγμα pickle σε λεξικό

```
import pickle

emp = {1:"A",2:"B",3:"C",4:"D",5:"E"}
pickling_on = open("Emp.pickle","wb")
pickle.dump(emp, pickling_on)
pickling_on.close()
```

Παράδειγμα unpickle σε λεξικό

```
pickle_off = open("Emp.pickle","rb")

emp = pickle.load(pickle_off)

print(emp)
```

Πλεονεκτήματα- Μειονεκτήματα

+

- Βοηθά στην αποθήκευση περίπλοκων δεδομένων.
- Πολύ εύκολο στη χρήση, δεν απαιτεί πολλές γραμμές κώδικα

-

- Δυσκολία με μη python προγράμματα στο χειρισμό αντικειμένων Pickle
- Κίνδυνοι ασφαλείας από δεδομένα που προέρχονται από κακόβουλες πηγές.

Shelve library

- Η βιβλιοθήκη shelve χρησιμεύει για αποθήκευση περισσότερων από ένα αντικείμενα στα οποία θέλουμε να έχουμε άμεση πρόσβαση, αν δεν επιθυμούμε να αποθηκεύσουμε τον container.
- Με τον τρόπο αυτό μπορούμε να ανασύρουμε επί μέρους αντικείμενα, ανεξάρτητα της σειράς που τα αποθηκεύσαμε.

Δημιουργία shelve

```
import shelve

with shelve.open('test_shelf.db') as
s: s['key1'] = {
            'int': 10,
            'float': 9.5,
            'string': 'Sample data’,
}
```

Ανάγνωση δεδομένων

```
import shelve

with shelve.open('test_shelf.db') as s:
        existing = s['key1']

print(existing)
```

Εγγραφή δεδομένων

```
import shelve
with shelve.open('test_shelf.db') as s:
        print(s['key1'])
        s['key1']['new_value'] = 'test write'
with shelve.open('test_shelf.db', writeback=True)
as s:
        print(s['key1'])
```

Data Persistence

JSON

JSON

Το JSON αποτελεί ένα ανοικτό πρότυπο το οποίο χρησιμοποιεί κείμενο αναγνώσιμο από τον άνθρωπο με σκοπό τη μετάδοση πληροφοριακών αντικειμένων δεδομένων. Η συντομογραφία του JavaScript Object Notation είναι μια μορφή κειμένου που βασίζεται σε ένα υποσύνολο της γλώσσας προγραμματισμού JavaScript, αλλά είναι εντελώς ανεξάρτητη από την οποιαδήποτε γλώσσα προγραμματισμού. Χρησιμοποιείται ως μια εναλλακτική λύση της XML για τη μετάδοση δεδομένων μεταξύ του εξυπηρετητή και των εφαρμογών διαδικτύου.

Το JSON αποτελείται από δυο δομές. Η πρώτη δομή είναι μια συλλογή από ζεύγη ονομάτων /τιμών, στις περισσότερες γλώσσες προγραμματισμού αυτό μπορεί να γίνει αντιληπτό ως ένα αντικείμενο, μια δομή ένα λεξικό ή μια λίστα κλειδιών.

Η δεύτερη δομή είναι μια ταξινομημένη λίστα τιμών, στις περισσότερες γλώσσες προγραμματισμού αυτό μπορεί να γίνει αντιληπτό ως ένας πίνακας, μια λίστα ή μια ακολουθία.

Βιβλιοθήκη json

```
j_data = json.dumps(data) # data to json
data = json.loads(j_data) # json to  data
```

Python	JSON Equivalent
dict	object
list, tuple	array
str	string
int, float, int	number
True	true
False	false
None	null

Βιβλιοθήκη json - Μετατροπή λεξικού σε json

```
import json

person_dict = {'name': John', 'age': 18, 'children': None}

person_json = json.dumps(person_dict)

# Output: {"name": "John", "age": 18, "children": null}
print(person_json)
```

Βιβλιοθήκη json – ανάγνωση από αρχείο

```
{
"name": "Bob", "languages": ["English", "Fench"]
}
```

```
import json

with open('path_to_file/person.json') as f:
        data = json.load(f)

# Output: {'name': 'Bob', 'languages': ['English', 'Fench']}

print(data)
```

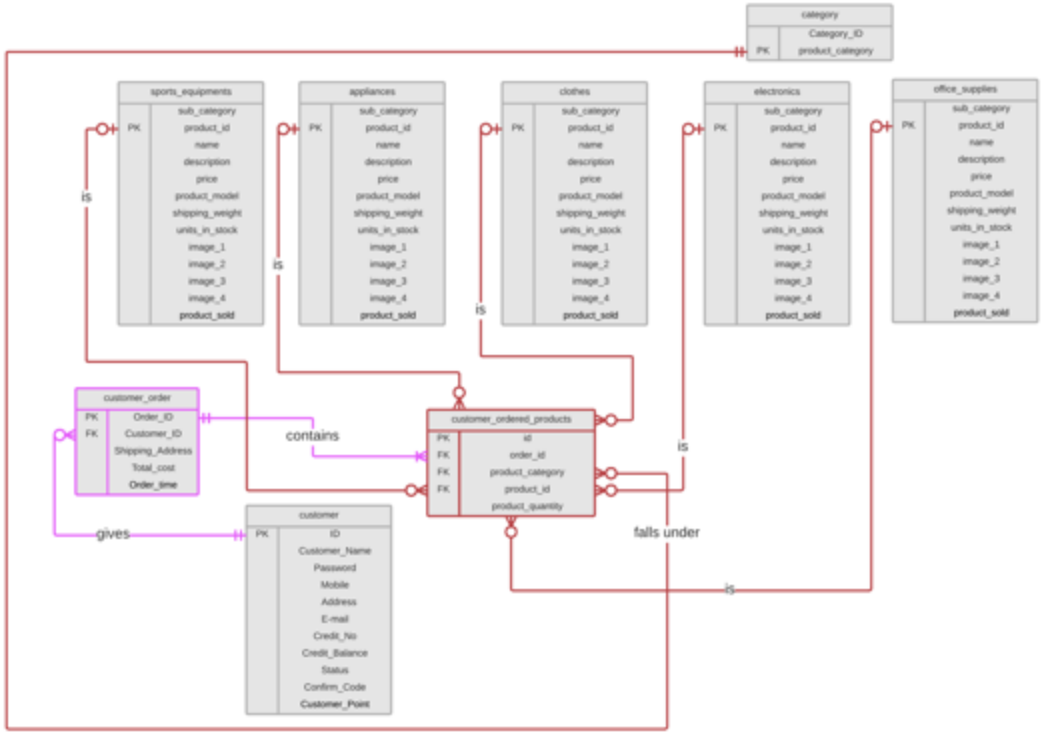
Βιβλιοθήκη json – αποθήκευση σε αρχείο

```
import json

person_dict = {"name": "Bob", "languages": ["English", ”Greek"],
"married": True, "age": 35 }

with open('person.txt', 'w') as json_file:
        json.dump(person_dict, json_file)
```

Σχεσιακές βάσεις δεδομένων

- Εισαγωγή στο σχεσιακό μοντέλο SQL
- Η βιβλιοθήκη sqlite3

Σχεσιακή βάση δεδομένων: μία συλλογή δεδομένων οργανωμένη σε συσχετισμένους πίνακες που παρέχει ταυτόχρονα ένα μηχανισμό για ανάγνωση, εγγραφή, τροποποίηση ή και πιο πολύπλοκες διαδικασίες πάνω στα δεδομένα.



ERD – dba.stackexchange.com

Structured Query Language (SQL) - CRUD

Το CRUD είναι ένα ακρωνύμιο για τους τέσσερις βασικούς τύπους εντολών SQL:

- Δημιουργία (create)
- Ανάγνωση (read)
- Ενημέρωση (update)
- Διαγραφή (delete)

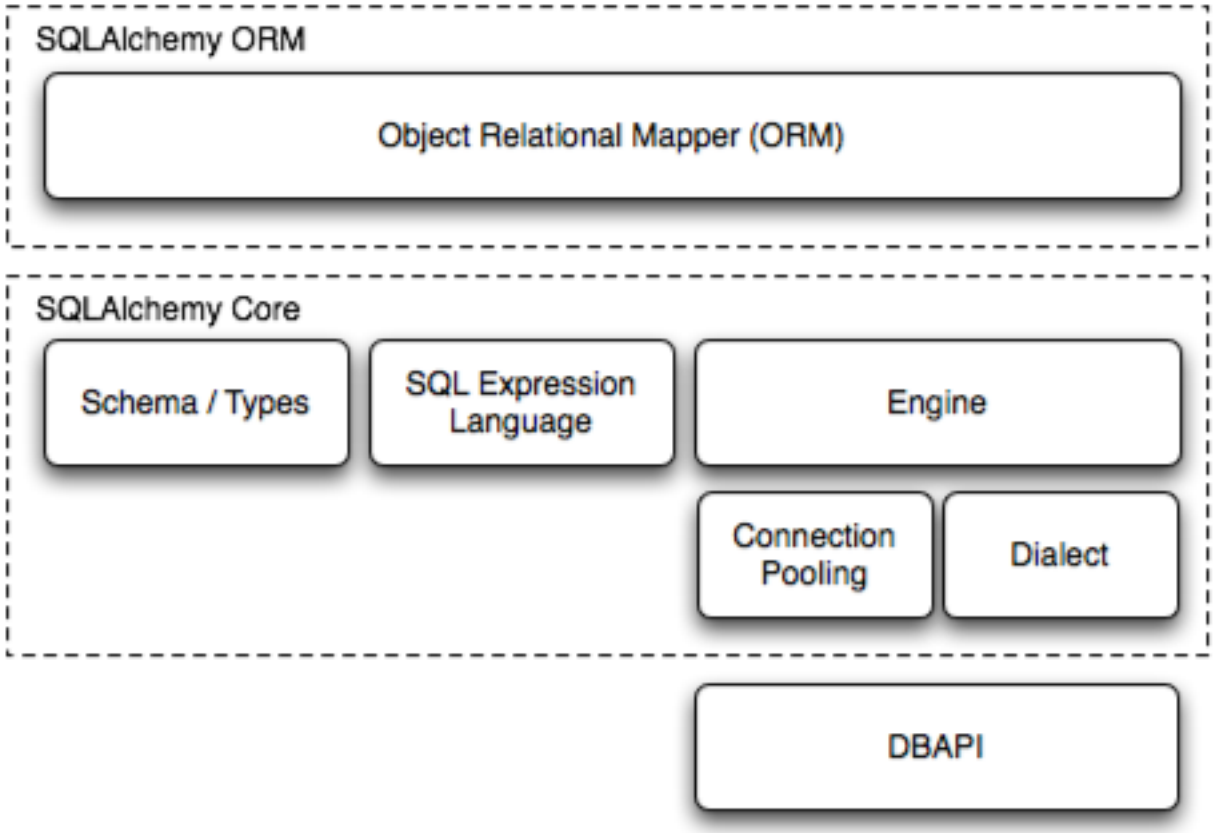
Οι περισσότερες εφαρμογές έχουν κάποια λειτουργικότητα CRUD

Μια εφαρμογή CRUD είναι μια εφαρμογή που χρησιμοποιεί φόρμες για τη λήψη δεδομένων από και προς μια βάση δεδομένων.

Σχεσιακές βάσεις δεδομένων

embedded data base

Αντιστοίχιση αντικειμένων σε σχεσιακούς πίνακες SQLAlchemy



Παράδειγμα εντολών sql στην python ανάκτηση από ένα πίνακα

```
def restore_data(self):
    try:
        conn = sqlite3.connect(self.file_name)
        curs = conn.cursor()
        curs.execute('select * from student;')
        for s in curs.fetchall():
            self.dd[s[0]] = list(s[1:])
    except: print('σφάλμα ανοίγματος αρχείου')
```

NoSQL βάσεις δεδομένων

Η πραγματική έννοια αυτού του όρου NoSQL είναι ότι αυτός ο τύπος δεν ακολουθεί τις αρχές των παραδοσιακών σχεσιακών βάσεων δεδομένων.

Αντ 'αυτού, έχουν τις αρχές τους και μπορούν να χειριστούν αποτελεσματικά την παραγωγή σύγχρονων βάσεων δεδομένων ιστού όπως το Twitter, το Facebook, το Google.

NoSQL βάσεις δεδομένων

Το NoSQL περιγράφει μια μέθοδο ανάκτησης δεδομένων που διαμορφώνεται με διαφορετικό τρόπο από τις παραδοσιακές σχεσιακές βάσεις δεδομένων όπως η MySQL και η PostgreSQL.

Οι βάσεις δεδομένων NoSQL χρησιμοποιούνται συνήθως σε δεδομένα μεγάλου όγκου και εφαρμογές ιστού που εκτελούνται σε πραγματικό χρόνο.

Το NoSQL αντιπροσωπεύει το "Not Only SQL", καθώς μπορεί να υπάρχουν και περιπτώσεις όπου χρησιμοποιείται η SQL.

Μια πιο σωστή ονομασία θα μπορούσε να είναι: "μη σχεσιακή βάση δεδομένων"

NoSQL βάσεις δεδομένων

- Better Scaling
- Significantly Cheaper
- Schema-less (Partially)
- Less Management
- Cheaper Hardware
- Heterogeneous Data

Name	Year	Type	Developer
MongoDB	2009	Document	10Gen
CouchDB	2005	Document	Apache
Cassandra	2008	Column Store	Apache
CouchBase	2011	Document	Couchbase
Riak	2009	Key-value	Basho Technologies
SimpleDB	2007	Document	Amazon

Βιβλιοθήκη - SQLite 3

sqlite3

- Είναι ένα σύστημα διαχείρισης σχεσιακής βάσης δεδομένων (RDBMS) σε μια βιβλιοθήκη C. Σε αντίθεση με άλλα συστήματα διαχείρισης βάσεων δεδομένων, δεν είναι τύπου πελάτη-διακομιστή. Αντίθετα, είναι ενσωματωμένο στο πρόγραμμα.

- Είναι συμβατό με το πρότυπο ACID και υλοποιεί το μεγαλύτερο μέρος του προτύπου SQL, γενικά ακολουθώντας τη σύνταξη της PostgreSQL.

- Το SQLite είναι μια δημοφιλής επιλογή ως ενσωματωμένο λογισμικό βάσης δεδομένων για local/client χρήση σε λογισμικό εφαρμογών, όπως τα προγράμματα περιήγησης στο διαδίκτυο. Είναι αναμφισβήτητα το πιο ευρέως αναπτυγμένο database engine, και χρησιμοποιείται σήμερα από αρκετούς διαδεδομένους browsers.

Τύποι δεδομένων

- **NULL**. The value is a NULL value.
- **INTEGER**. The value is a signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value.
- **REAL**. The value is a floating point value, stored as an 8-byte IEEE floating point number.
- **TEXT**. The value is a text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE).
- **BLOB**. The value is a blob of data, stored exactly as it was input.

Παράδειγμα δημιουργίας βάσης δεδομένων

```
import sqlite3

sqlite_file = 'my_first_db.sqlite' # name of the sqlite database file
table_name1 = 'my_table_1' # name of the table to be created
table_name2 = 'my_table_2' # name of the table to be created
new_field = 'my_1st_column' # name of the column
field_type = 'INTEGER' # column data type

# Connecting to the database file
conn = sqlite3.connect(sqlite_file) c = conn.cursor()
# Creating a new SQLite table with 1 column
c.execute('CREATE TABLE {tn} ({nf} {ft})'\
        .format(tn=table_name1, nf=new_field, ft=field_type))

 # Creating a second table with 1 column and set it as PRIMARY KEY # note that
PRIMARY KEY column must consist of unique values!
c.execute('CREATE TABLE {tn} ({nf} {ft} PRIMARY KEY)'\
        .format(tn=table_name2, nf=new_field, ft=field_type))

# Committing changes and closing the connection to the database file
conn.commit()
conn.close()
```

Παράδειγμα δημιουργίας πίνακα

```
def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by db_file
    :param db_file: database file
    :return: Connection object or None
    """
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return None
```

Παράδειγμα ανάκτησης δεδομένων

```
def main():
    database = "C:\\sqlite\db\python.db"

    # create a database connection
    conn = create_connection(database)
    with conn:
        print("1. Query task by priority:")
        select_task_by_priority(conn,1)

        print("2. Query all tasks")
        select_all_tasks(conn)
```

Παράδειγμα εισαγωγής δεδομένων

```
def create_project(conn, project):
    """
    Create a new project into the projects table
    :param conn:
    :param project:
    :return: project id
    """
    sql = ''' INSERT INTO projects(name,begin_date,end_date)
              VALUES(?,?,?) '''
    cur = conn.cursor()
    cur.execute(sql, project)
    return cur.lastrowid
```

Παράδειγμα διαγραφής δεδομένων

```
def delete_task(conn, id):
    """
    Delete a task by task id
    :param conn:  Connection to the SQLite database
    :param id: id of the task
    :return:
    """
    sql = 'DELETE FROM tasks WHERE id=?'
    cur = conn.cursor()
    cur.execute(sql, (id,))
```

Παράδειγμα τροποποίησης δεδομένων

```
def update_task(conn, task):
    """
    update priority, begin_date, and end date of a task
    :param conn:
    :param task:
    :return: project id
    """
    sql = ''' UPDATE tasks
              SET priority = ? ,
                  begin_date = ? ,
                  end_date = ?
              WHERE id = ?'''
    cur = conn.cursor()
    cur.execute(sql, task)
```

παράδειγμα σύνδεσης με cluster (NoSQL)

```
## Cassandra driver library import
from cassandra.cluster import Cluster

## Cassandra instance create
cluster = Cluster()

## Cassandra connection establishment (using local)
session = cluster.connect()
```

παράδειγμα δημιουργίας keyspace (NoSQL)

```
## keyspace creation if not exists  assuming that the cluster is single-
node
session.execute("CREATE KEYSPACE IF NOT EXISTS tourismdata
" + \
                    "WITH replication " + \
                    "= {'class': 'SimpleStrategy', " + \
                    " 'replication_factor': 1}")

## use the keyspace previously created
session.set_keyspace('tourismdata')
```

παράδειγμα δημιουργίας keyspace (NoSQL)

```
## Creation of  table if not exists by executing CQL command
session.execute('CREATE TABLE IF NOT EXISTS catalog (' + \
                'country TEXT, ' + \
                gender map<timestamp,double>,, ' + \
                ‘age map<timestamp,double>,, ' + \
                ‘categ1 map<timestamp,double>,, ' + \
'PRIMARY KEY(country))’)

## Close the connection with Cassandra
cluster.shutdown
```