En aquesta unitat fem una primera aproximació a la persistència de dades i a les bases de dades. Sovint en un programa tenim certa informació relativament complexa que volem que estigui disponible per a altres execucions d'aquest mateix o d'altres programes. En la unitat 5 vam veure que podíem emmagatzemar etiquetadors amb pickle. D'aquesta manera podíem entrenar un etiquetador amb un programa i guardar l'etiquetador per fer-lo servir en una altre programa tantes vegades com vulguem.

En aquesta unitat, que pretén ser una introducció bàsica a aquest tema, tractarem un arxiu que és una base de dades terminològica de Terminologia Oberta en format TBX (*Term Base eXchange*, un format estàndard basat en XML per a l'intercanvi de bases de dades terminològiques.

En el primer apartat repassarem com fer un tractament d'aquest XML fent servir xmltodict, estudiat en la unitat anterior. Després veurem les següents llibreries:

- pickle
- shelve
- sqlite3

D'aquesta unitat disposes dels següents arxius:

- La unitat en pdf: 11-Bases_de_dades-cat.pdf
- Els arxius i programes necessaris: programes11-cat.zip

11.1. Tractament bàsic de l'arxiu TBX

En aquesta unitat treballarem amb un arxiu TBX que conté un diccionari terminològic de dret administratiu de Terminologia Oberta del TermCat (cdladministratiu.tbx, que està en els arxius de la unitat). Com que els arxius TBX són arxius XML farem un tractament bàsic fent servir xml2dict (estudiats a la unitat 10). El programa que fa aquest tractament bàsic és el programa-11-1.py. El codi és molt senzill:

```
import xmltodict

xml=open('cdldretadministratiu.tbx')
xmldict = xmltodict.parse(xml.read())
id=0
for termentry in xmldict["martif"]["text"]["body"]["termEntry"]:
    id+=1
    cat=[]
    spa=[]
    for langset in termentry["langSet"]:

        if langset["@xml:lang"]=="ca":
            cat.append(langset["tig"]["term"])

        if langset["@xml:lang"]=="es":
            spa.append(langset["tig"]["term"])

    print(id,cat,spa)
```

Aquest codi simplement mostarà per la pantalla un id d'entrada (que s'incrementa automàticament) i els termes en català i castellà per a cada entrada. A continuación mostrem un fragment:

```
1 ['accessibilitat'] ['accesibilidad']
2 ['acció'] ['acción']
3 ['acció de nul·litat'] ['acción de nulidad']
4 ['acció de regrés', 'acció de repetició'] ['acción de regreso', 'acción de repetición']
5 ['acció de responsabilitat'] ['acción de responsabilidad']
6 ['acomiadament disciplinari'] ['despido disciplinario']
7 ['acord administratiu'] ['acuerdo administrativo']
8 ['acord mutu', 'acord amistós'] ['acuerdo amigable', 'mutuo acuerdo']
```

```
9 ["acord d'iniciació"] ['acuerdo de iniciación']
10 ["acord de necessitat d'ocupació"] ['acuerdo de necesidad de ocupación']
```

Fixeu-vos que algunes entrades tenen diverses denominacions en cada una de les llengües.

Partirem d'aquest programa de tractament bàsic per estudiar cada una de les opcions de persistència de dades o base de dades.

11.2. Pickle

El mòdul Pickle ja el coneixem de la unitat 5 on el fèiem servir per emmagatzemar etiquetadors morfosintàctics. Aquest mòdul ens permet serialitzar i guardar a disc objectes de Python. El programa-11-2.py converteix la base de dades terminològica en un diccionari on la clau és el id i el valor és un diccionari on les claus son els codis de llengua i els valors són llistes que contenen les denominacions en cada una de les llengües. Al final el programa guarda aquest objecte diccionari en un arxiu que anomenarem cdldretadministratiu.pickle.

```
import xmltodict
import pickle

xml=open('cdldretadministratiu.tbx')
xmldict = xmltodict.parse(xml.read())
id=0

bddict={}

for termentry in xmldict["martif"]["text"]["body"]["termEntry"]:
 id+=1
 cat=[]
 spa=[]
 for langset in termentry["langSet"]:

    if langset["@xml:lang"]=="ca":
       cat.append(langset["tig"]["term"])

    if langset["@xml:lang"]=="es":
       spa.append(langset["tig"]["term"])

 print(id,cat,spa)
 bddict[id]={}
 bddict[id]["cat"]=[]
 for denocat in cat:
    bddict[id]["cat"].append(denocat)
 bddict[id]["spa"]=[]
 for denospa in spa:
    bddict[id]["spa"].append(denocat)

pickle.dump(bddict, open("cdldretadministratiu.pickle", "wb"))
```

Ara al programa-11-3.py carregeum aquest diccionari i fem alguna cerca senzilla.

```
import pickle

bddict = pickle.load(open("cdldretadministratiu.pickle", "rb"))

#escrivim directament la informació de l'entrada amb id=234
print(bddict[234])

#ara volem buscar la traducció al castellà del terme "unitat d'actuació urbanística"
#com que no sabem quina id té hem de recórrer tot el diccionari
```

```
for id in bddict.keys():
 for termecat in bddict[id]["cat"]:
    if termecat=="unitat d'actuació urbanística":
       print("SPA:",", ".join(bddict[id]["spa"]))
```

Que ens proporcionarà dues sortides, la informació corresponent a l'entrada amb id=234 i l'equivalent castellà del terme "unitat d'actuació urbanística":

```
{'spa': ['concesión de servicio público'], 'cat': ['concessió de servei públic']}
SPA: unidad de actuación, unidad de actuación urbanística
```

11.3. Shelve

Un "shelf" és un tipus de dades persistent semblant a un diccionari però on els valors (no les claus!) poden ser qualsevol objecte de Pyhton. La seva utilització és força semblant a pickle. Al programa-11-4.py podem veure com crear un shelf que conté la base de dades terminològica. Fixeu-vos en el detall de que les claus del diccionari han de ser cadenes (strings).

```
import xmltodict
import shelve

xml=open('cdldretadministratiu.tbx')
xmldict = xmltodict.parse(xml.read())
id=0

bddict={}

bddict = shelve.open("cdldretadministratiu.shelve")

for termentry in xmldict["martif"]["text"]["body"]["termEntry"]:
 id+=1
 cat=[]
 spa=[]
 for langset in termentry["langSet"]:

    if langset["@xml:lang"]=="ca":
       cat.append(langset["tig"]["term"])

    if langset["@xml:lang"]=="es":
       spa.append(langset["tig"]["term"])

 print(id,cat,spa)
 bddict[str(id)]={}
 dictaux={}
 dictaux["cat"]=[]
 dictaux["spa"]=[]
 for denocat in cat:
    dictaux["cat"].append(denocat)
 bddict[str(id)]["spa"]=[]
 for denospa in spa:
    dictaux["spa"].append(denospa)
 bddict[str(id)]=dictaux


bddict.close()
```

Ara, al programa-11.5.py llegim aquest shelf i mostrem la informació de l'entrada 234 (fixeu-vos que posem "234" perquè ha de ser una cadena). També mostrem l'equivalent de traducció castellà del terme català "unitat d'actuació urbanística". Fixeu-vos també com en aquest cas hem hagut de recórrer tot el diccionari.

```
import shelve

bddict = shelve.open("cdldretadministratiu.shelve")

print(bddict["234"])

for id in bddict.keys():
 for termecat in bddict[id]["cat"]:
    if termecat=="unitat d'actuació urbanística":
       print("SPA:",", ".join(bddict[id]["spa"]))
```

11.4. SQLite 3

En aquest apartat presentarem com podem crear i manipular bases de dades SQLite 3 amb Python. L'ús d'aquesta opció és molt interessant, ja que les bases de dades que creem des dels nostres programes es podran manipular també amb eines estàndard de SQLite 3. Aquest apartat, però, està destinat a usuaris amb una experiència prèvia amb bases de dades i coneixements del llenguatge SQL. Presentar tots aquest conceptes requeriria un llibre sencer i tota una assignatura. Tot i això, es poden adquirir els coneixements més bàsics amb tutorials que es troben a Internet, entre els que destaco el següents:

- Sobre llenguatge SQL: https://www.w3schools.com/sql/
- Sobre SQLite: http://www.sqlitetutorial.net/
- Sobre Python i SQLite: https://www.pythoncentral.io/introduction-to-sqlite-in-python/

En el programa-11-6.py creem la base de dades sqlite que anomenarem cdldretadministratiu.sqlite.

```
import sqlite3

conn=sqlite3.connect("cdldretadministratiu.sqlite")

cur = conn.cursor()

cur.execute("CREATE TABLE terme(terms_id INTEGER PRIMARY KEY AUTOINCREMENT, entry_id INTEGER, lang TEXT, terme TEXT)")

conn.commit()
```

En el programa-11-7.py llegim l'arxiu TBX i introduïm totes les dades a la base de dades creada:

```
import xmltodict

import sqlite3

xml=open('cdldretadministratiu.tbx')

xmldict = xmltodict.parse(xml.read())

id=0

entry_id=0

bddict={}

conn=sqlite3.connect("cdldretadministratiu.sqlite")

cur = conn.cursor()

data=[]

for termentry in xmldict["martif"]["text"]["body"]["termEntry"]:

   entry_id+=1


   for langset in termentry["langSet"]:

      record=[]

      record.append(entry_id)
```

```
        record.append(langset["@xml:lang"])
        record.append(langset["tig"]["term"])
        data.append(record)
cur.executemany("INSERT INTO terme (entry_id, lang, terme) VALUES (?,?,?)",data)
conn.commit()
```

En realitat, el que fem és crear una llista (data) que conté llistes (record) que contenen les dades bàsiques d'entry_id, llengua i terme. L'emmagatzematge a la base de dades es realitza en una sola vegada amb executemany. La introducció de dades una a una a la base de dades és molt més lenta i, per aquest motiu, es fa d'un sol cop. Donat que les dades de la llista es troben en memòria, si cal introduir un nombre molt elevat de dades, és convenient realitzar diverses operacions d'inserció a la base de dades i posar a zero les llistes després de cada inserció.

I per últim, al programa-11-8.py fem les cerques a la base de dades. Com podem veure, la cerca de les dades associades a un determinat entry_id, es pot fer en una sola consulta. En canvi per recuperar l'equivalent de traducció de "unitat d'actuació urbanística" necessitem fer dues cerques: primer cerquem l'entry_id corresponent a aquest terme i després cerquem els termes castellans associats amb l'entry_id trobat.

```
import sqlite3
conn=sqlite3.connect("cdldretadministratiu.sqlite")
cur = conn.cursor()

#busquem tota la informació sobre l'entrada "234"
busquem="234"
cur.execute('SELECT lang,terme from terme where entry_id="'+str(busquem)+'";')
data=cur.fetchall()
for d in data:
    print(d[0],d[1])

#busquem l'equivalent castellà de "unitat d'actuació urbanística"

#primer busquem el entry_id que tigui aquest terme:

busquem="unitat d'actuació urbanística"
cur.execute('SELECT entry_id from terme where terme="'+str(busquem)+'" and lang="ca";')
data=cur.fetchall()
for entry_id in data:
    #la segona cerca busquem tots el termes amb llengua es i que tinguin aquesta entry_id
    cur.execute('SELECT terme from terme where entry_id="'+str(entry_id[0])+'" and lang="es";')
    data2=cur.fetchall()
    for termespa in data2:
        print("SPA:",termespa[0])
```