seance_5_intro_et_json

May 20, 2022

1 2A.i - Données non structurées, programmation fonctionnelle

Une table dans une base de données est déjà le résultat d'une réflexion sur la façon de les représenter.

[1]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

[1]:
```
<IPython.core.display.HTML object>
```

1.1 Avant-propos : programmation fonctionnelle ou numpy ?

- toolz/cytoolz : programmation fonctionnelle
- numpy : calcul matriciel

Données : twitter_for_network_100000.db.zip or twitter_for_network_100000.db.zip (xavierdupre.fr).

[2]:
```
import pyensae.datasource
pyensae.datasource.download_data("twitter_for_network_100000.db.zip")
```

[2]:
```
['.\\twitter_for_network_100000.db']
```

[3]:
```
import numpy as np

def my_sum(l):
    res = 0
    for it in l:
        res += it
    return res
l = list(range(100000))
a = np.arange(100000)

print("User defined method or cross-method")
%timeit my_sum(a) # user defined with numpy array
%timeit sum(a)    # built-in with numpy array
%timeit np.sum(l) # numpy function with list
%timeit my_sum(l) # user definedwith list
print("Builtin function")
%timeit sum(l)    # built-in with list
print("Numpy function")
%timeit np.sum(a) # numpy function
%timeit a.sum()   # numpy method
```

```
User defined method or cross-method

c:\python372_x64\lib\site-packages\ipykernel_launcher.py:6: RuntimeWarning:
overflow encountered in long_scalars


21.3 ms ± 548 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)

c:\python372_x64\lib\site-packages\ipykernel_launcher.py:1: RuntimeWarning:
overflow encountered in long_scalars
  """Entry point for launching an IPython kernel.

17.1 ms ± 611 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
6.24 ms ± 274 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
4.95 ms ± 658 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
Builtin function
1.54 ms ± 80.8 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
Numpy function
49.6 µs ± 366 ns per loop (mean ± std. dev. of 7 runs, 10000 loops each)
50.4 µs ± 4.17 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)
```

Il y a un rapport de 10 dans le temps d'exécution entre la méthode **"user defined"** et la méthode **"builtin"**. On retrouve ce même rapport entre la méthode "builtin" et les méthodes numpy. On peut noter que mélanger les objets numpy et non-numpy donne de très mauvais résultats.
Dans le cas de la programmation fonctionnelle, nous nous situerons plutôt dans le cas "builtin" :
Project
Computation
Data Structures
Code de l'utilisateur
Python
Python
CyToolz
C
Python
Pandas/NumPy
C
C
Dans le cas de manipulation de donnée structurée de taille "raisonnable", pandas et numpy restent plus performants. Ils peuvent toutefois être limités par plusieurs points :

- manipulation de données plus complexes avec des sous-listes, des champs manquants
- ils sont construits sur le principe de chargement en mémoire des données

[4]:
```python
import os, psutil, gc, sys
if not sys.platform.startswith("win"):
    import resource

def memory_usage_psutil():
    gc.collect()
    process = psutil.Process(os.getpid())
    mem = process.memory_info()[0] / float(2 ** 20)

    print( "Memory used : %i MB" % mem )
    if not sys.platform.startswith("win"):
```

```
        print( "Max memory usage : %i MB" % (resource.getrusage(resource.RUSAGE_SELF).
    ↪ru_maxrss//1024) )
```

[5]:
```
memory_usage_psutil()
```

```
Memory used : 114 MB
```

[6]:
```
import cytoolz as ct # import groupby, valmap, compose
import cytoolz.curried as ctc ## pipe, map, filter, get
import sqlite3
import pprint
try:
    import ujson as json
except:
    print("ujson not available")
    import json

conn_sqlite = sqlite3.connect("twitter_for_network_100000.db")
cursor_sqlite = conn_sqlite.cursor()
```

Le code suivant va lire plusieurs gigaoctets de données, et la consommation maximale de mémoire du process ne va augmenter que de quelques Mo. De plus ce code manipule des dictionnaires qu'il serait compliqué de faire rentrer dans un DataFrame pandas.

[7]:
```
cursor_sqlite.execute('SELECT content FROM tw_users' )
object_to_sum = ctc.pluck( "followers_count", ctc.map( json.loads, ctc.pluck( 0,␣
    ↪cursor_sqlite ) ) )
print(sum(object_to_sum))
```

```
108086205
```

[8]:
```
memory_usage_psutil()
```

```
Memory used : 120 MB
```

Dans le cadre du TP d'aujourd'hui, les données que nous allons utiliser peuvent largement tenir en mémoire, et de façon générale, lorsqu'on développe des codes pour gérer des gros volumes de données, on les teste sur des volumes de données qui tiennent en mémoire.
Dans le cadre de la gestion de volume important de données, on ne pourra donc pas stocker des résultats intermédiaire, on va donc composer des fonctions pour qu'elles produisent directement le résultat final.
Cas classique :

```
resultat_intermediaire_1 = f( donnees )
resultat_intermediaire_2 = g( resultat_intermediaire_1 )
resultat_final = h( resultat_intermediaire_2 )
```

Programmation fonctionnelle :

```
resultat_final = h( g( f( donnees ) ) )
```

2 Données structurées SQL et NOSQL

SQL signifie *Structured Query Language*, les tables ont un nombre de colonnes fixes, et chaque colonne possède un type particulier. Comment gère-t-on un nombre d'objets variables ? La plupart du temps avec une table secondaire qui contiendra une ligne par élément de la liste.
Par exemple, si l'on veut stocker la liste des films possédés par une personne.
Table person
Id
Name
1
Jean
2
Paul
3
Jacques
Table related_items
Id
Person_id
Value
1
1
Star wars
2
1
Cyrano
3
1
Lord of the rings
4
2
Mad max
5
2
Dr Horrible
Ce système est très "structuré" (comme son nom l'indique) et peut s'avérer assez lourd si l'on a affaire à des données moins bien structurées, avec beaucoup de listes, des données présentes ou non.
On voit donc se développer de plus en plus des systèmes alternatifs, dit NoSQL (pour Not Only Sql).
Nous allons en voir trois :

- Sqlite3 support for Json
- mongodb
- PostGreSql

Ils sont analogues sur la nature des données stockées, elles le sont au format Json.

2.1 Json ? Qu'est ce que c'est ?

Il s'agit du format majoritaire pour les API informatiques internet.
Par exemple les données renvoyées par twitter (sur lesquelles nous travaillerons aujourd'hui) sont sous ce format.
Il signifie : JavaScript Object Notation. Il se base essentiellement sur des dictionnaires (association clé/valeur) et des listes. Il est très proche des objets python (modulo les false/False)

[9]:
```
import pprint
```

```
cursor_sqlite.execute('SELECT content FROM tw_users LIMIT 1')
user = cursor_sqlite.fetchone()[0]
print("#"*15 + " user raw json " + "#"*15)
print( user )
print("#"*15 + " user as python dict " + "#"*15)
pprint.pprint( json.loads( user ) )
cursor_sqlite.execute('SELECT content FROM tw_status LIMIT 1')
print("#"*15 + " status as python dict " + "#"*15)
pprint.pprint( json.loads( cursor_sqlite.fetchone()[0] ) )
```

```
############### user raw json ###############
{"utc_offset": 7200, "friends_count": 454, "entities": {"description": {"urls":
[]}, "url": {"urls": [{"expanded_url": "http://www.havas.com", "display_url":
"havas.com", "indices": [0, 22], "url": "http://t.co/8GcZtydjWh"}]}},
"description": "Havas Group CEO", "id": 1103159180, "contributors_enabled":
false, "geo_enabled": false, "name": "Yannick Bollor\u00e9", "favourites_count":
873, "verified": true, "protected": false, "created_at": "Sat Jan 19 08:23:33
+0000 2013", "statuses_count": 654, "lang": "en", "time_zone": "Ljubljana",
"screen_name": "YannickBollore", "location": "", "id_str": "1103159180", "url":
"http://t.co/8GcZtydjWh", "followers_count": 7345, "listed_count": 118,
"has_extended_profile": false}
############### user as python dict ###############
{'contributors_enabled': False,
 'created_at': 'Sat Jan 19 08:23:33 +0000 2013',
 'description': 'Havas Group CEO',
 'entities': {'description': {'urls': []},
              'url': {'urls': [{'display_url': 'havas.com',
                                'expanded_url': 'http://www.havas.com',
                                'indices': [0, 22],
                                'url': 'http://t.co/8GcZtydjWh'}]}},
 'favourites_count': 873,
 'followers_count': 7345,
 'friends_count': 454,
 'geo_enabled': False,
 'has_extended_profile': False,
 'id': 1103159180,
 'id_str': '1103159180',
 'lang': 'en',
 'listed_count': 118,
 'location': '',
 'name': 'Yannick Bolloré',
 'protected': False,
 'screen_name': 'YannickBollore',
 'statuses_count': 654,
 'time_zone': 'Ljubljana',
 'url': 'http://t.co/8GcZtydjWh',
 'utc_offset': 7200,
 'verified': True}
############### status as python dict ###############
{'contributors': None,
 'coordinates': None,
 'created_at': 'Wed Jul 22 17:14:47 +0000 2015',
 'entities': {'hashtags': [{'indices': [16, 28], 'text': 'Agriculture'}],
              'symbols': [],
```

```
              'urls': [{'display_url': 'blog-fillon.com/2015/07/plan-d...',
                        'expanded_url': 'http://www.blog-
fillon.com/2015/07/plan-de-soutien-du-gouvernement-a-l-agriculture-
decevant.html',
                        'indices': [139, 140],
                        'url': 'http://t.co/wZ3HkhHvuM'}],
              'user_mentions': [{'id': 34598169,
                                 'id_str': '34598169',
                                 'indices': [3, 14],
                                 'name': 'Fix RICHARD',
                                 'screen_name': 'FixRichard'},
                                {'id': 551669623,
                                 'id_str': '551669623',
                                 'indices': [113, 128],
                                 'name': 'François Fillon',
                                 'screen_name': 'FrancoisFillon'}]},
 'favorite_count': 0,
 'favorited': False,
 'geo': None,
 'id': 623904030251708416,
 'id_str': '623904030251708416',
 'in_reply_to_screen_name': None,
 'in_reply_to_status_id': None,
 'in_reply_to_status_id_str': None,
 'in_reply_to_user_id': None,
 'in_reply_to_user_id_str': None,
 'is_quote_status': False,
 'lang': 'fr',
 'place': None,
 'possibly_sensitive': False,
 'retweet_count': 23,
 'retweeted': False,
 'retweeted_status': {'contributors': None,
                      'coordinates': None,
                      'created_at': 'Wed Jul 22 17:09:33 +0000 2015',
                      'entities': {'hashtags': [{'indices': [0, 12],
                                                 'text': 'Agriculture'}],
                                   'symbols': [],
                                   'urls': [{'display_url': 'blog-
fillon.com/2015/07/plan-d...',
                                             'expanded_url': 'http://www.blog-
fillon.com/2015/07/plan-de-soutien-du-gouvernement-a-l-agriculture-
decevant.html',
                                             'indices': [113, 135],
                                             'url': 'http://t.co/wZ3HkhHvuM'}],
                                   'user_mentions': [{'id': 551669623,
                                                      'id_str': '551669623',
                                                      'indices': [97, 112],
                                                      'name': 'François Fillon',
                                                      'screen_name':
'FrancoisFillon'}]},
                      'favorite_count': 4,
                      'favorited': False,
                      'geo': None,
```

```
                      'id': 623902715584888832,
                      'id_str': '623902715584888832',
                      'in_reply_to_screen_name': None,
                      'in_reply_to_status_id': None,
                      'in_reply_to_status_id_str': None,
                      'in_reply_to_user_id': None,
                      'in_reply_to_user_id_str': None,
                      'is_quote_status': False,
                      'lang': 'fr',
                      'place': None,
                      'possibly_sensitive': False,
                      'retweet_count': 23,
                      'retweeted': False,
                      'source': '<a href="http://www.apple.com" '
                                'rel="nofollow">iOS</a>',
                      'text': '#Agriculture : "Le plan du Gouvernement '
                              "n'apporte aucune solution durable aux filières "
                              'en crise" @FrancoisFillon '
                              'http://t.co/wZ3HkhHvuM',
                      'truncated': False,
                      'user': {'contributors_enabled': False,
                               'created_at': 'Thu Apr 23 12:27:59 +0000 2009',
                               'default_profile': False,
                               'default_profile_image': False,
                               'description': '#CM #ComPol #SocialMedia '
                                              '#Politique ¿  #PDL2015 '
                                              '#AvecRetailleau ¿  #Angers '
                                              '#Paris #Nantes',
                               'entities': {'description': {'urls': []},
                                            'url': {'urls': [{'display_url':
'about.me/fixrichard',
                                                              'expanded_url':
'http://about.me/fixrichard',
                                                              'indices': [0,
                                                                          22],
                                                              'url':
'http://t.co/apaFxTrJMQ'}]}},
                               'favourites_count': 10728,
                               'follow_request_sent': False,
                               'followers_count': 2488,
                               'following': False,
                               'friends_count': 865,
                               'geo_enabled': True,
                               'has_extended_profile': False,
                               'id': 34598169,
                               'id_str': '34598169',
                               'is_translation_enabled': False,
                               'is_translator': False,
                               'lang': 'fr',
                               'listed_count': 178,
                               'location': 'Angers, Pays de la Loire',
                               'name': 'Fix RICHARD',
                               'notifications': False,
                               'profile_background_color': 'C71E4E',
```

```
                                'profile_background_image_url':
'http://abs.twimg.com/images/themes/theme14/bg.gif',
                                'profile_background_image_url_https':
'https://abs.twimg.com/images/themes/theme14/bg.gif',
                                'profile_background_tile': True,
                                'profile_banner_url':
'https://pbs.twimg.com/profile_banners/34598169/1436884606',
                                'profile_image_url':
'http://pbs.twimg.com/profile_images/621396058682343424/IMzOsat1_normal.jpg',
                                'profile_image_url_https':
'https://pbs.twimg.com/profile_images/621396058682343424/IMzOsat1_normal.jpg',
                                'profile_link_color': '0C4499',
                                'profile_sidebar_border_color': 'FFFFFF',
                                'profile_sidebar_fill_color': 'C0DFEC',
                                'profile_text_color': '333333',
                                'profile_use_background_image': True,
                                'protected': False,
                                'screen_name': 'FixRichard',
                                'statuses_count': 20446,
                                'time_zone': 'Paris',
                                'url': 'http://t.co/apaFxTrJMQ',
                                'utc_offset': 7200,
                                'verified': False}},
 'source': '<a href="http://twitter.com/download/iphone" '
           'rel="nofollow">Twitter for iPhone</a>',
 'text': 'RT @FixRichard: #Agriculture : "Le plan du Gouvernement n\'apporte '
         'aucune solution durable aux filières en crise" @FrancoisFillon '
         'http://t.c...',
 'truncated': False,
 'user': {'contributors_enabled': False,
          'created_at': 'Thu Apr 12 08:09:49 +0000 2012',
          'default_profile': False,
          'default_profile_image': False,
          'description': '',
          'entities': {'description': {'urls': []},
                       'url': {'urls': [{'display_url': 'blog-fillon.com',
                                         'expanded_url': 'http://www.blog-
fillon.com',
                                         'indices': [0, 22],
                                         'url': 'http://t.co/kqslZOa4nj'}]}},
          'favourites_count': 17,
          'follow_request_sent': False,
          'followers_count': 256321,
          'following': False,
          'friends_count': 1425,
          'geo_enabled': True,
          'has_extended_profile': False,
          'id': 551669623,
          'id_str': '551669623',
          'is_translation_enabled': False,
          'is_translator': False,
          'lang': 'fr',
          'listed_count': 1455,
          'location': '',
```

```
          'name': 'François Fillon',
          'notifications': False,
          'profile_background_color': '0D37E0',
          'profile_background_image_url': 'http://pbs.twimg.com/profile_backgrou
nd_images/598167675/40rdxs0i9ay7rkh2a4no.jpeg',
          'profile_background_image_url_https': 'https://pbs.twimg.com/profile_b
ackground_images/598167675/40rdxs0i9ay7rkh2a4no.jpeg',
          'profile_background_tile': False,
          'profile_banner_url':
'https://pbs.twimg.com/profile_banners/551669623/1440524758',
          'profile_image_url':
'http://pbs.twimg.com/profile_images/555688874446319616/0k5Rrgra_normal.jpeg',
          'profile_image_url_https':
'https://pbs.twimg.com/profile_images/555688874446319616/0k5Rrgra_normal.jpeg',
          'profile_link_color': '3B3B42',
          'profile_sidebar_border_color': 'FFFFFF',
          'profile_sidebar_fill_color': 'EFEFEF',
          'profile_text_color': '333333',
          'profile_use_background_image': False,
          'protected': False,
          'screen_name': 'FrancoisFillon',
          'statuses_count': 7725,
          'time_zone': 'Athens',
          'url': 'http://t.co/kqslZ0a4nj',
          'utc_offset': 10800,
          'verified': True}}
```


Le NoSql / json permet donc une alternative au schéma classique suivant :
Table person
Id
Name
1
Jean
2
Paul
3
Jacques
Table related_items
Id
Person_id
Value
1
1
Star wars
2
1
Cyrano
3
1
Lord of the rings
4
2
Mad max
5

2
Dr Horrible
Qui serait :
Table person_with_items
Id
Name
Item_list
1
Jean
['Star wars', 'Cyrano', 'Lord of the rings']
2
Paul
['Mad max', 'Dr Horrible']
3
Jacques
Cette dernière structure serait vraiment un exemple de nosql dans son sens de Not Only Sql, il y a un mixe de données structurées et non structurées. Il y a également certaines base de données où il n'y a plus du tout de structure, comme mongodb, qui est qualifiée de document-oriented.
Table person_with_items
{'Id': 1, 'Name': 'Jean', 'Item_list' : ['Star wars', 'Cyrano', 'Lord of the rings']}
{'Id': 2, 'Name': 'Paul', 'Item_list' : ['Mad max', 'Dr Horrible']}
{'Id': 3, 'Name': 'Jacques', 'Item_list' : []}
Il faut toutefois remarquer que cette dernière structure au moins deux inconvénients par rapport à une structure Sql avec une sous-table :

- vous ne pouvez pas accéder directement aux objets de 'Item_list' sans passer par la table person
- les informations de Item_list ne peuvent être partagées entre plusieurs objets, on peut donc avoir à restocker les informations

Voir module psycopg.

[10]:
```
try:
    import psycopg2
    from psycopg2.extras import Json
    postgre_ok = True
except ImportError:
    postgre_ok = False

if postgre_ok:
    db_name = 'cours_ensae'
    conn_string = "host='localhost' dbname='{0}' user='python' password='kyojin'".
 ↪format( db_name )
    try:
        conn_psql = psycopg2.connect(conn_string)
        cursor_psql = conn_psql.cursor()
        postgre_ok = True
    except psycopg2.OperationalError:
        postgre_ok = False
```

[11]:
```
if postgre_ok:
    conn_psql.server_version
```

[12]:
```
if postgre_ok:
    conn_psql.rollback()
```

[13]:
```
if postgre_ok:
    def get_data_sql(doc_id):
        cursor_psql.execute("SELECT id, company FROM document WHERE id = %s",␣
 ↪(doc_id,))
        res_1 = cursor_psql.fetchone()
        cursor_psql.execute("SELECT id FROM ticket WHERE document_id = %s ORDER BY␣
 ↪id", (doc_id,))
        res_2 = cursor_psql.fetchall()
        tickets_id = [it[0] for it in res_2 ]
        cursor_psql.execute("SELECT id FROM coupon WHERE ticket_id = ANY( %s ) ORDER␣
 ↪BY id", (tickets_id,))
        res_3 = cursor_psql.fetchall()
        return res_1 + (res_2,) + (res_3,)

    %timeit get_data_sql(10000)
    get_data_sql(10000)
```

[14]:
```
if postgre_ok:
    def get_data_sql_join(doc_id):
        cursor_psql.execute("SELECT d.id, d.company, t.id, c.id FROM document as d \
                                                                JOIN ticket as t on d.
 ↪id = t.document_id \
                                                                JOIN coupon as c on t.
 ↪id = c.ticket_id \
                                WHERE d.id = %s", (doc_id,))
        return cursor_psql.fetchall()

    %timeit get_data_sql_join(10000)
    get_data_sql_join(10000)
```

[15]:
```
if postgre_ok:
    def get_data_nosql(doc_id):
        cursor_psql.execute("SELECT id, company, content FROM document_nosql WHERE id␣
 ↪= %s", (doc_id,))
        return cursor_psql.fetchone()

    %timeit get_data_nosql(10000)
    get_data_nosql(10000)
```

mongodb (pymongo) lui ,ne connait pas de colonnes, que des documents, dont le format est analogue à un objet json.
Cela se traduit par une très grande simplicité, pas besoin de déclarer les tables, ni les bases de données ...

[16]:
```
mongo = False
if mongo:
    import pymongo

    mongo_client = pymongo.MongoClient( 'localhost', 27017 )
    mongo_db = mongo_client.ensae_db

    mongo_db.table_for_ensae.delete_many( {} )
    mongo_db.table_for_ensae.insert_one( {'nom' : 'Martin', 'prenom' : 'Nicolas',␣
 ↪'grades': [20,18,7,12]} )
```

```
    mongo_db.table_for_ensae.insert_one( {'nom' : 'Dupont', 'prenom' : 'Jean',␣
↪'grades': [11,5,7,12]} )
    mongo_db.table_for_ensae.insert_one( {'nom' : 'Martin', 'prenom' : 'Gilles',␣
↪'grades': [10,10,10,10]} )

    user = mongo_db.table_for_ensae.find_one( {'nom' : 'Dupont'} )
    user_list = mongo_db.table_for_ensae.find( {} )
    _ = list(map( pprint.pprint, user_list ))
```

Par contre certaines syntaxes usuelles en sql, ici le groupby, ont une écriture nettement plus complexes en mongodb.

[17]:
```
if mongo:
    result = mongo_db.table_for_ensae.group(['nom'],
                                            None,
                                            {'list': []}, # initial
                                            'function(obj, prev) {prev.list.push(obj)}')
    pprint.pprint( result )
```

Mon retour :

- mongodb malgré sa simplicité d'utilisation peut être très gourmand en ressources (consommation d'espace disque et/ou mémoire 15 fois supérieure à PostGreSql pour les mêmes données). Je la déconseille pour une application personnelle.
- Sqlite3 est plus archaïque, mais le fait d'avoir une base de donnée contenue dans un fichier est très pratique pour certains usages (déploiement chez un client ou pour des élèves)
- PostGreSql me semble le plus robuste pour un usage en serveur personnel.

2.2 Et les fichiers plats ?

Vous pouvez tout à fait utiliser des fichiers plats.
Ils offrent beaucoup de simplicités d'utilisation.
Ils auront des performances potentiellement très proches pour une lecture complète.

[18]:
```
cursor_sqlite.execute("SELECT content FROM tw_users LIMIT 10000" )

with open("tw_users.json", 'w') as f:
    for it_user in cursor_sqlite:
        f.write(it_user[0])
        f.write("\n")

with open("tw_users.json", 'r') as f:
    nb_total_followers = 0
    for it_user in f:
        nb_total_followers += json.loads( it_user )["followers_count"]

print( nb_total_followers )
```

```
4284281
```

2.3 Et pandas ?

Pandas attend lui des données plus ou moins structurées. Vous pouvez charger une base sous pandans avec la syntaxe suivante :

[19]:
```
import pandas as pd

df = pd.read_sql(  "SELECT id, screen_name from tw_users", conn_sqlite )
print( df.head() )
print( df.shape )
```

```
           id     screen_name
0  1103159180  YannickBollore
1  2865692548     yveslemasne
2    24732180     harlemdesir
3   359979086      jpraffarin
4   273341346  gilles_schnepp
(100071, 2)
```

[20]:
```

```