

# Programmation avancée Python

créé par st. Oleksii Nikonov. Vérifié par proff. Julien Velcin

June 1st 2020

# 1 Partie 0: project arrangement

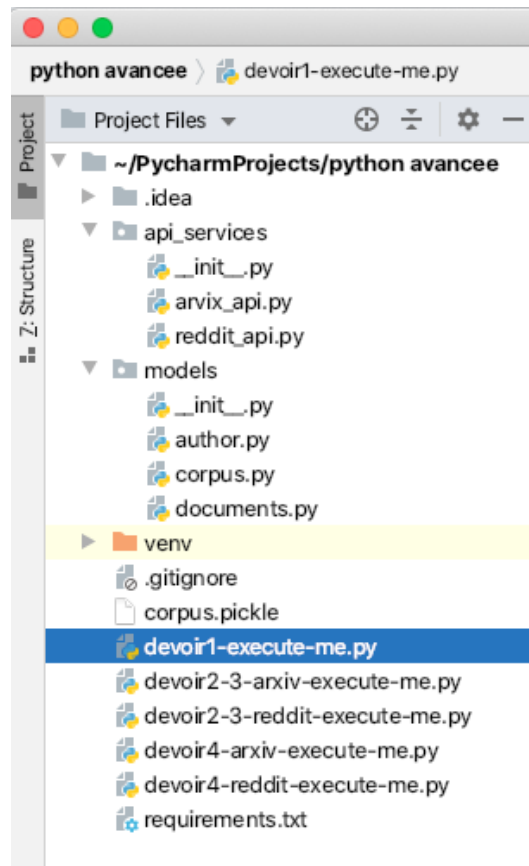
L'objectif principal de ces travaux dirigés est de mettre au point un moteur de recherche d'information maison.

## 1.1 L'arrangement du project

Le project est divisé par des fichiers exécutives et des modules pour le rendre plus lisible:

- des fichiers exécutives peut être en cours d'exécution dans Idea PyCharm ou les autres logiciel;
- des services travaillant avec api de Reddit ou Arxiv;
- des modèles divisée par les Classes différents comme Corpus, Author et Documents.

Le logique est partagé sur les fichiers différents pour les rendre plus compréhensible.



L'arrangement du project

## 2 Partie 1: charhement des donnees et premier exploration

Dans 1ere partie c'était implimenté:

- les service pour faire des demandes à Reddit et Arvix;
- le fichier d'appel des services et demonstrantion des resultats.

### 2.1 Le reddit fetching service

Le service prend comment le parametre d'un mot et fait le requête avec lui, si le mot n'était pas passé le valeur par default serait appliqué.

```
# 1.1
import praw
import pandas as pd

def request_and_parse_docs(query="hot"):
    posts_csv = []
    posts = []
    # ici on utilise nos credentials obtenu sur reddit
    reddit = praw.Reddit(client_id="2vXzVc3mrchCVQ",
                          client_secret="j-9P_b1ZVyNIhNPtxitAMs1_rDo",
                          user_agent="my user agent")

    # ici on fait l'iteration sur the post recu
    for post in reddit.subreddit(query).hot(limit=10):
        # 1.1 Quels sont les champs disponibles
        # Il y a beaucoup des chapms mais les plus utilisable sont ecrits cidessous
        # 1.1 Alimentez la liste docs avec ce texte uniquement.
        # Vous pouvez d ej`a vous d ebarrasser des sauts de ligne (\n) en les rempla
        # cant par des espaces.

        # Voici on remplace le character avec espace
        converted = post.selftext.replace('\n', ' ')
        posts_csv.append([post.title, post.author.name, post.score, post.id, post.subreddit, post.body])
        posts.append({'title': converted, 'author': post.author.name, 'body': converted, 'url': post.url})
    # 1.1 Quel est le champ contenant le contenu textuel : tout sauf subreddit
    # cree un tableau par pandas
    columns = ['title', 'author', 'score', 'id', 'subreddit', 'url', 'num_comments', 'body', 'url']
    csv = pd.DataFrame(posts_csv, columns=columns)
    return posts
```

### 2.2 Le arvix fetching service

Le même nous avons pour arvix function

```

# 1.2
import requests, xmltodict, urllib, re

res = requests.get("http://export.arxiv.org/api/query?search_query=all:france&start=0&max_results=10")

dictt = xmltodict.parse(res.content)

# print(json.dumps(dictt, indent=4))
# 1.2 Quels sont les champs disponibles ?
# id, updated, published, title, summary, author, arxiv, link, arxiv, category,

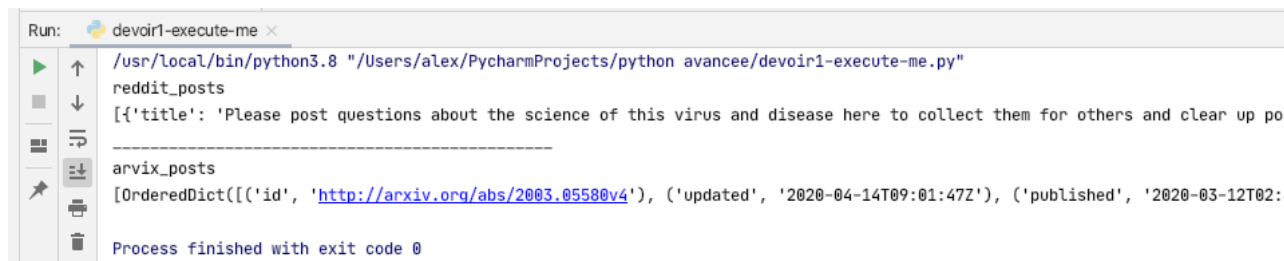
# 1.2 Quel est le champ contenant le contenu textuel ?
# title, summary

# 1.3
def request_and_parse_docs(query='covid'):
    url = "http://export.arxiv.org/api/query?search_query=all:" + query + "&start=0&max_results=10"
    result = urllib.request.urlopen(url).read().decode()
    #le resultat est xml type ce pour ca il faut le convertir au dictionnaire
    parsedResult = xmltodict.parse(result)
    return parsedResult['feed']['entry']

```

## 2.3 Le resultat de l'execution des fonction

Le service est une fonction qui travail avec web-services et retourne le list de data



```

Run: devoir1-execute-me x
/usr/local/bin/python3.8 "/Users/alex/PycharmProjects/python avancee/devoir1-execute-me.py"
reddit_posts
[{'title': 'Please post questions about the science of this virus and disease here to collect them for others and clear up po
-----
arxiv_posts
[OrderedDict([('id', 'http://arxiv.org/abs/2003.05580v4'), ('updated', '2020-04-14T09:01:47Z'), ('published', '2020-03-12T02:
Process finished with exit code 0

```

Le resultat de fetching les deux sources

## 3 Partie 2-3: acces a partir des meta-donnees; heritage

Dans cette partie nous avons crée des classes pour gester l'information en choix propre. Chaque classe est responsable de sa partie de l'information fourni.

### 3.1 Classe: Author

```
class Author():
# 2.4
    def __init__(self, name):
        self.name = name
        self.ndoc = 0
        self.production = {}

    def increment_ndoc(self):
        self.ndoc += 1

    def set_production(self, doc):
        self.production[doc['title']] = doc['id']

    def get_production(self):
        return self.production

# 2.5
# on ecrase la method str pour fournir l'information qu'on a besoin (p.e. pour print opera
    def __str__(self):
        return f'{self.name}'
```

### 3.2 Class: Document

Cette classe réuni la classe de base (parent) et 2 classes dérivées (enfant) qui ont leur-même methods de gestion d'information. Le but principal que les classes sont crees avec *Polymorphism*

La classe parent.

```
from gensim.summarization.summarizer import summarize

class Document():
# 2.1
    def __init__(self, json):
        self.json = json
        self.type = None

# 3.4 Here is the type getter is set
    def get_type(self):
        return self.type

# cet setter permet de change le document
    def set_json(self, json):
        for key in json:
```

```

        self.json[key] = json[key] if self.json[key] is not None else None

    def __json__(self):
        return {
            'titre': self.json['titre'],
            'date': self.json['date'],
            'url': self.json['url'],
            'text': self.json['text']
        }

# 2.2
# on ecrase la method str pour fournir l'information qu'on a besoin (p.e. pour print opera
    def __str__(self):
        return f'Document: {self.json["title"]}'

# 2.8
# on ecrase la method repr pour fournir l'information qu'on a besoin
    def __repr__(self):
        return f'Doc title: {self.json["title"]}, release date: {self.json["date"]}'

# 4.3
    def summarize_me(self):
        return summarize(self.json['text'])

```

Les classes enfants

Chaque des classes a ses method uniques pour fournir le data en choix props, par exemple ArxivDocument a une list des authors tandis que RedditDocument a le nombre de commentaires. Mais toutfois tout les deux ont le title, date, url et text chapms à grace de son parent. Aussi on voit (dans les images de la resultat le corpus pickled et réstauré pour les deux cas.

```

# 3.1
class RedditDocument(Document):
# 3.1 on appelle le constructeur pour initialiser cette class
    def __init__(self, title, text, doc):
        json = {'title': title, 'date': doc['date'], 'url': doc['url'], 'text': text}
# ici nous appellons le method __init__ of parent
        super().__init__(json)
        self.ncomment = doc['num_comments']
        self.author = doc['author']
        self.type = 'reddit'

    def increment_ncomment(self):
        self.ncomment += 1

```

```

def get_ncomment(self):
    return f'Comment quantity: {self.ncomment}'

    @staticmethod
    def get_text_key():
        return 'body'

# 3.2
class ArxivDocument(Document):
    # 3.2 on appelle le constructeur pour initialiser cette class
    def __init__(self, title, text, doc):
        json = {'title': title, 'date': doc['published'], 'url': doc['link'], 'text': text}
        # ici nous appelons le method __init__ of parent
        super().__init__(json)
        authors = doc['author']
        self.authors = [a['name'] for a in authors] if len(authors) != 1 else [authors['name']]
        self.type = 'arxiv'

    def add_coauthors(self, aut):
        self.authors.append(aut)

    def get_authors(self):
        return f'Author list: {self.authors}'

    @staticmethod
    def get_text_key():
        return 'summary'

```

### 3.3 Le resultat d'utilisation des classes

Pickled corpus



Pickled corpus est enregistré et chargé de la fichier

L'utilisation à l'ensemble Reddit



Le resultat d'implimentation des method des classes pour Reddit

L'utilisation à l'ensemble Arxiv



```

Run: devoir2-3-arxiv-execute-me x
/usr/local/bin/python3.8 "/Users/alex/PycharmProjects/python avancee/devoir2-3-arxiv-execute-me.py"
1406
1448
1509
849
680
1001
1653
1114
1229
81
['Doc title: Similarities and Evolutionary Relationships of COVID 19 and Related Viruses, release date: 2020-03-12T02:33:13Z', '
Author list: ['Ren Kong', 'Guangbo Yang', 'Rui Xue', 'Ming Liu', 'Feng Wang', 'Jianping Hu', 'Xiaoqiang Guo', 'Shan Chang', 'Ale
----- revived corpus -----
['Doc title: Similarities and Evolutionary Relationships of COVID 19 and Related Viruses, release date: 2020-03-12T02:33:13Z', '

Process finished with exit code 0

```

Le resultat d'implimentation des method des classes pour Arxiv

## 4 Partie 4: acces a partir du contenu

### 4.1 Corpus et Factory

Dans le 4eme partie nous avons implimentes quelque libraries pour obtenir l'analyse de text en profondeur. Bien sur nous avons utilisé l'usine pour créer les classes different de document et le statistique détaillée de mot dans un doc et partout.

```

import re, pickle
import pandas as pd
from collections import defaultdict
from models.documents import RedditDocument, ArxivDocument
from models.author import Author

```

```

# 3.5 Factory
# on utilise defaultdict comme un switheur en dependant de valeur passé
# il y a besoin de lambda: 'undefined' parce que le defaultdict attendant une fonction comme
# s'il mode fourne une autre cle (non 'reddit', non 'arxiv') le defaultdict levera l'excepti
class DocFactory():
    @staticmethod
    def generate_goc(mode):
        return defaultdict(lambda: 'undefined', {
            'reddit': RedditDocument,

```

```

        'arvix': ArxivDocument
    })[mode]

# 2.7
class Corpus():
    def __init__(self, mode, parsed_docs):
        # 2.3
        self.collection = {}
        self.id2doc = {}
        # 2.6
        self.authors = {}
        self.id2aut = {}
        self.naut = 0
        self.ndoc = 0
        # ici on courrons l'usine et
        self.doc_instance = DocFactory.generate_goc(mode)
        # on prepare tout les variable qu'on a besoin
        # 4.5 on utilise Set pour facilement eviter les doublons
        self.dictionary = set()
        # 4.4
        self.concordancier = pd.DataFrame(
            columns=['word', 'contexte gauche', 'motif trouve', 'contexte droit'])
        # 4.3-4.6-4.7 on enregistre les mots occurance et le nombre d'occurrences
        self.stats = pd.DataFrame(
            columns=['word', 'frequence_de_mot_par_tout', 'frequence_de_mot_par_docs'])
        self.proceed_docs(parsed_docs)

# Cette fonction est basique de Corpus pour initiate lui-meme
def proceed_docs(self, parsed_docs):
    for doc in parsed_docs:
        # 1.3
        raw_text = doc[self.doc_instance.get_text_key()]
        text, text_as_separate_words = self.nettoyer_texte(raw_text)
        # on affiche la taille de la document
        print(len(text))
        # voici on verifie si la document et long de 100 mots, si pas - on le saute
        if len(text) < 100:
            continue

        self._concorde(text_as_separate_words, text)

        peeled_title = ' '.join(re.findall(r'\w+', doc['title']))
        self.collection[peeled_title] = self.doc_instance(peeled_title, text, doc)
        self.id2doc[self.ndoc] = peeled_title

```

```

        for author in doc['author']:
            if isinstance(author, dict):
                name_ = author['name']
                self.add_author(name_)
                self.authors[name_].increment_ndoc()
                self.authors[name_].set_production({'id': self.ndoc, 'title': peeled_title})
            # ici on ajoute le quantité des doc qu'on a processés
            self.ndoc += 1

    def add_author(self, name_):
        # on controle l'auteur s'il deja exist à dictionnaires
        if not (name_ in self.authors):
            self.authors[name_] = Author(name_)
            self.id2aut[self.naut] = name_
            self.naut += 1
        # si Oui on eleve un Exception personnalisee
        else:
            raise AuthorError(name_)

# 4.4
    def nettoyer_texte(self, raw_text):
        # tout les lines nouvelles sont remplacées
        raw_text = raw_text.lower().replace('\n', ' ')
        # et le text est separé aux sentences
        split_sentences = re.split(r' *[\.\?!][\'"\)\]]* *', raw_text)
        text_as_separate_words = []
        for s in split_sentences:
            # on trouve tout les mots et chiffres par \w+
            # parceque la date est un chiffre et il peut etre une cle (p.e. 1789)
            text_as_separate_words.append(re.findall(r'\w+', s))
        # les mots separe se reunissent dans sentence et en fin au text
        text = '. '.join([' '.join(x) for x in text_as_separate_words])
        return text, text_as_separate_words

# 4.1
    def _search(self, word, text):
        # on trouve le mot et son location dans le text
        result = re.search(word, text, re.IGNORECASE)
        return word, text[0: result.start()], text, text[result.end(): len(text)]

# 4.2
    def _concorde(self, text_as_separate_words, genuine_text):
        j = -1
        for i, sentence_as_separate_words in enumerate(text_as_separate_words):
            self.dictionary.update(sentence_as_separate_words)
            for w in sentence_as_separate_words:

```

```

words_csv = self._search(w, genuine_text)
self.concordancier.loc[len(self.concordancier)] = words_csv

# Pour compter le nombre de documents contenant chacun des mots
# et mot meme frequence
# on fait:
# - inspect si le mot est deja au tableau
# - et si on passe le meme sentence une fois plus
is_not_exist = w not in self.stats['word'].values
is_one_and_the_same_sentence = j == i
if is_not_exist:
    self.add_word_to_stats(w)
else:
    if is_one_and_the_same_sentence:
        self.update_word_frequency_stats(w)
    else:
        self.update_docs_frequency_stats(w)
j = i

def add_word_to_stats(self, word):
    self.stats.loc[len(self.concordancier)] = [word, 1, 1]

# 4.3
def update_word_frequency_stats(self, word):
    frequence_par_tout = 'frequence_de_mot_par_tout'
    self.stats.loc[(self.stats.word == word), frequence_par_tout] \
        = self.stats.loc[(self.stats.word == word), frequence_par_tout].values[0] + 1

# 4.7
def update_docs_frequency_stats(self, word):
    self.update_word_frequency_stats(word)
    frequence_par_docs = 'frequence_de_mot_par_docs'
    self.stats.loc[(self.stats.word == word), frequence_par_docs] \
        = self.stats.loc[(self.stats.word == word), frequence_par_docs].values[0] + 1

def get_concordancier(self, n=10):
    count = len(self.stats)
    # avec le line subséquent on fait le pandas nous affiches tout tableau
    return 'Total des mots: ' + str(count), self.concordancier[:n].to_string()

# 4.3
def get_stats(self, n=10):
    count = len(self.stats)
    # sorted = self.stats.sort_values(by=['frequence_de_mot_par_tout'])[:n]
    sorted = self.stats.sort_values(by=['frequence_de_mot_par_tout'], ascending=False)[:n]
    return 'Total des mots: ' + str(count), sorted

```

```

def get_doc_list_of_number(self, n=10):
    docs = []
    for i in range(n):
        title = self.id2doc[i]
        doc = self.collection[title]
        docs.append(doc.__repr__())
    return docs

def save(self):
    with open('corpus.pickle', 'wb') as f:
        pickle.dump(self, f)

def add_author_to_doc(self, name, doc_id):
    self.add_author(name)
    title = self.id2doc[doc_id]
    self.collection[title].add_coauthors(name)

def get_doc_by_id(self, id):
    title = self.id2doc[id]
    return self.collection[title]

class AuthorError(Exception):
    def __init__(self, author):
        super().__init__(f'Author "{author}" already exists in the author list of this document.

```

## L'utilisation à l'ensemble Arvix

```
Run: devoir4-arxiv-execute-me x
/usr/local/bin/python3.8 "/Users/alex/PycharmProjects/python avancee/devoir4-arxiv-execute-me.py"
1406
1448
1509
849
680
1001
1653
1114
1229
81
-----le statistique des mots triée par l'occurrence de mot dans un doc-----
Total des mots: 637
word frequency_de_mot_par_tout frequency_de_mot_par_docs
46 the 94 12
10 covid 78 1
123 of 73 1
11 19 67 1
62 and 64 1
137 to 36 4
45 for 29 1
545 a 27 1
2 we 25 7
34 in 20 7
-----Concordancier de mot trouvant dans le contexte-----
Total des mots: 637
word contexte gauche
0 today today we are all threatened by an unprecedented pandemic covid 19. how different is it from other
1 we today today we are all threatened by an unprecedented pandemic covid 19. how different is it from other
2 are today we today we are all threatened by an unprecedented pandemic covid 19. how different is it from other
-----Le resumé de la text certain-----
to mitigate the lack of publicly available covid 19 ct images for developing ct based diagnosis deep learning models of covi

Process finished with exit code 0
```

Le resultat d'implimentation des method des classes pour Arvix

## L'utilisation à l'ensemble Reddit

```
Run: devoir4-reddit-execute-me x
/usr/local/bin/python3.8 "/Users/alex/PycharmProjects/python avancee/devoir4-reddit-execute-me.py"
0
518
1583
488
114
0
0
175
0
0

-----le statistique des mots triée par l'occurrence de mot dans un doc-----
('Total des mots: 243',
  word  frequence_de_mot_par_tout  frequence_de_mot_par_docs
10  the                32                2
23  to                 18                1
15  a                  14                1
128 and                13                1
65  it                 12                5
12  i                  11                5
82  of                 9                 1
1   my                 9                 1
69  or                 8                 1
173 we                 6                 4)

-----Concordancier de mot trouvant dans le context-----
('Total des mots: 243', 'Empty DataFrame\nColumns: [word, contexte gauche, motif trouve, contexte droit]\nIndex

Process finished with exit code 0
```

Le resultat d'implimentation des method des classes pour Reddit

## 4.2 Error handling

Voici pour verifier comment mon methor de l'addition travail j'ai ajouté une Error personnalisée qui affiche que l'auteur ajoutant déjà est dans le dictionnaire du corpus

```
['Doc title: Similarities and Evolutionary Relationships of COVID 19 and Related Viruses, release date:
Author list: ['Mizuho Nishio', 'Shunjiro Noguchi', 'Hidetoshi Matsuo', 'Takamichi Murakami', 'AlexNikonov']
Traceback (most recent call last):
  File "/Users/alex/PycharmProjects/python_avancee/devoir2-3-arxiv-execute-me.py", line 14, in <module>
    corpus.add_author_to_doc('AlexNikonov', 2)
  File "/Users/alex/PycharmProjects/python_avancee/models/corpus.py", line 169, in add_author_to_doc
    self.add_author(name)
  File "/Users/alex/PycharmProjects/python_avancee/models/corpus.py", line 80, in add_author
    raise AuthorError(name_)
models.corpus.AuthorError: Author "AlexNikonov" already exists in the author list of this document.

Process finished with exit code 1
```

Error personnalisée dans Corpus