2a_Cours 2a Introduction a la syntaxe python_part1_sans_corrections

November 3, 2020

Table of Contents

1 Rappel

Nous avons appris :

```
1) A naviguer un peu dans console (python, cd, C: ou D:, cd C:CePC.., cd ./Documents, etc.)
2) A lancer un jupyter notebook (entre autres)
3) A installer correctement (i.e. sur un environnement virtuel) les librairies python
```

Les questions qui ont été posées :

```
1) Qu'est ce qu'un "Kernel" ?
```

Le Kernel est le "coeur" python sur lequel s'execute le notebook.

- Il peut donc se lancer (run d'une cellule) et s'arrêter (le "stop" du kernel, pratique en cas....de boucle infinie par exemple).
- Il peut également se relancer (pour tout "supprimer" - y compris si nécessaire les sorties) et "recommencer" à zéro (i.e. effacer toutes les variables).

2) Anaconda prompt pour MAC (mais également pour linux)

Anaconda prompt n'existe pas sous MAC/Linux. Il faut donc utiliser simplement le terminal standard (en principe, Anaconda est bien configuré sous linux et MAC, python répond donc sans problème).

2 Tour d'horizon de la syntaxe python

Les fondamentaux de la syntaxe python sont bien résumés dans un poly qui n'est plus maintenu mais qui reste globalement à jour (Xavier Dupré - ENSAE). Il vous est fortement recommandé d'être curieux et d'aller fouiller sur le site de Xavier Dupré (une mine d'or !!!)

Dans une premier temps, nous allons nous concentrer sur les premières parties : langages, variables types modifiables, types immuables, tests et boucles

http://www.xavierdupre.fr/app/teachpyx/helpsphinx/c_resume/python_sheet.html#le-langage

http://www.xavierdupre.fr/enseignement/initiation/resume_utile.pdf

Rappels :

- commenter du code (""" Ceci est un commentaire """, #Ceci est un commentaire)
- Utilisez l'option "markdown" dans une cellule d'un dataframe pour tout ce qui n'est pas du code ;)

[75]:
```
#Rappel : les types de base

i = 3                     # entier
r = 3.3                   # réel
s = "exemple"             # chaîne de caractères
c = (4,5)                 # couple de valeurs (ou tuple)
l = [ 4, 5, 6.5]          # listes de valeurs ou tableaux
x = l[0]                  # obtention du premier élément de la liste
d = { "un":1, "deux":2 } # dictionnaire de valeurs
y = d["un"]               # obtention de la valeur associée à l'élément "un"
couple = 4, 5             # collage de deux valeurs en un couple ou (tuple)
print(type(l))            # on affiche le type de la variable l
mat = [ [0,1], [2,3] ]    # liste de listes
print(mat[0][1])          # obtention du second élément de la première liste
n = None                  # None signifie que la variable existe mais qu'elle ne␣
↪contient rien
                          # elle est souvent utilisé pour signifier qu'il n'y a␣
↪pas de résultat
                          # car... une erreur s'est produite, une liste était␣
↪vide
```

```
<class 'list'>
1
```

A ne pas oublier :

INDENTATION *INDENTATION* indentation **INDENTATION** *INDENTATION* indentation **INDENTATION** *INDENTATION* indentation

3 Exercices - car c'est en sciant que Léonard de Vinci

Ces exercices sont fortement inspirés (voire pour certains copiés) des exercices proposés par Xavier Dupré

3.0.1 Ex : Calculatrice (combien font 5 + 10 ?)

[ ]:

3.0.2 Ex : Quel résultat si on incrémente le chiffre de 2 d'une unité supplémentaire ? puis de 5 unités ?

[ ]:

3.0.3 Ex : Quelle est la deuxième lettre du mot "LETTRE" ? Quelles sont les trois dernières lettres du mot "LETTRE" ?

[ ]:

3.0.4 Ex : Comment énumérer efficacement tous les nombres entre 1 et 10 (sans les copier à la main) ?

[ ]:

3.0.5 Ex : Comment ajouter un "s" à un mot inscrit au singulier ?

[ ]:

3.0.6 Ex : Peut-on ajouter une lettre à un nombre ?

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

3.0.7 Ex : Comment créer un test pour n'afficher que les nombres supérieurs à 10 (et 10 sinon ?)

[ ]:

3.0.8 Ex : Comment créer un test pour n'afficher que les nombres supérieurs à 10 et pairs ?

[13]:

3.0.9 Ex : Comment créer un test pour vérifier qu'une proposition est juste ?

[ ]:

3.0.10 Ex : Comment changer le type d'une variable ?

[ ]:

[ ]:

[ ]:

3.0.11 Ex : Comment vérifier si un élément est dans une liste

[2]:
```
#Est-ce que 1) "toto" et 2) "mickey" sont dans la liste des mes amis ?
liste_de_mes_amis = ['toto','personne']
```

3.0.12 Ex : Comment interagir avec une chaine de caractère

[3]:
```
#Séparer mon nom prenom et les rendre avec nom en majuscule, prenom en minuscule

str_initial = "John Doe"


#Pour les autres fonctions : jouez avec le poly !!!
```

3.0.13 Ex : Jouons un peu avec LA LISTE des jours de la semaine

[20]:
```
#Créer une liste des jours de la semaine :
```

[4]:
```
# #Méthode1 :
liste_jours_de_la_semaine =␣
 ↪['lundi','mardi','mercredi','jeudi','vendredi','samedi','dimanche']

print(liste_jours_de_la_semaine)
```

```
['lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi', 'samedi', 'dimanche']
```

[5]:
```
# Methode2 :
liste_jours_de_la_semaine = []
liste_jours_de_la_semaine.append('lundi')
liste_jours_de_la_semaine.append('mardi')
#...

print(liste_jours_de_la_semaine)
```

```
['lundi', 'mardi']
```

[23]:
```
#Combien y a-t-il de jours dans la semaine ?
```

[ ]:
```

```

[ ]:
```

```

[25]:
```
#Et si on ne conservait que les jours travaillés ?
```

[41]:
```
liste_jours_de_la_semaine =␣
 ↪['lundi','mardi','mercredi','jeudi','vendredi','samedi','dimanche']
#methode 1 :
```

[42]:
```
liste_jours_de_la_semaine =␣
 ↪['lundi','mardi','mercredi','jeudi','vendredi','samedi','dimanche']
#methode 2 :
```

[6]:
```
#Et si on ajoutait une nouveau jour : le "jourdi" entre le 2ème et le 3ème jour
```

3.0.14 Ex : Comptons les nombres impairs jusqu'à 20 via la COMPREHENSION DE LISTE

[7]:
```
#Il existe une manière très élégante de créer des listes sur lesquelles␣
↪s'appliquent des conditions.
#Exemple : créons la liste des chiffres impairs compris entre 0 et 20 - compris␣
↪(en une seule ligne idéalement)
```

3.0.15 Ex : Créez un DICTIONNAIRE avec comme clef, le numéro du jour de la semaine et en valeur le nom du jour correspondant

[78]:
```
# Créez un dictionnaire avec les jours de la semaine
```

[79]:
```
#Methode 1 :

dict_jour_de_la_semaine = {}
dict_jour_de_la_semaine[1]='lundi'
dict_jour_de_la_semaine[2]='mardi'
dict_jour_de_la_semaine[3]='mercredi'
dict_jour_de_la_semaine[4]='jeudi'
dict_jour_de_la_semaine[5]='vendredi'
dict_jour_de_la_semaine[6]='samedi'
dict_jour_de_la_semaine[7]='dimanche'
```

[80]:
```
#Methode 2 :

dict_jour_de_la_semaine ={1:'lundi', 2 : 'mardi', 3: 'mercredi', \
                          4: 'jeudi', 5: 'vendredi', 6: 'samedi', 7 :␣
↪'dimanche'}

#NB : le saut de ligne !!!
#NB2 : how to comment entire blocks ?
#NB3 : auto-completion !!!
```

[81]:
```
#Methode 3 :
dict_jour_de_la_semaine = {}
for num, jour in enumerate(liste_jours_de_la_semaine) :
    dict_jour_de_la_semaine[num+1]=jour

#NB : enum !
```

[82]:
```
dict_jour_de_la_semaine
```

[82]:
```
{1: 'lundi',
 2: 'mardi',
```

```
 3: 'jourdi',
 4: 'mercredi',
 5: 'jeudi',
 6: 'vendredi',
 7: 'samedi'}
```

3.0.16 Les tableaux numeriques - bref aperçu

[83]:
```
#Nous reviendrons sur ce sujet lorsque nous aborderons la librairie pandas.
import numpy
a = numpy.array ( [0,1] )
print(a)
print(type(a))
```

```
[0 1]
<class 'numpy.ndarray'>
```

3.0.17 Ex : Comment copier une variable, une liste, etc. ?

[9]:
```
# Ce qui semble fonctionner pour une variable (type immuable)...
```

[8]:
```
#...ne fonctionne plus pour les listes et les dictionnaires (type immutable) !
```

3.1 Différence VALEUR vs. ADRESSE / REFERENCE

Pour plus de détail, voir :

https://miamondo.org/2017/02/16/python-copier-une-liste-avec-le-module-copy/

Cette remarque s'applique à tout type modifiable, liste, dictionnaire ou toute autre classe. La suppression d'une variable n'implique pas la suppression de toutes les variables se référant à une seule et même instance de classe (sera en particulier le cas pour les dataframes, lorsque nous aborderons la question du module pandas -> .copy())

Reportez-vous à la cheat sheet !

3.2 TESTS ET BOUCLES

Encore une fois, reportons nous à la cheat sheet !

La structure de base du test :

- if, elif, else !

Les deux types de boucles :

- la boucle while

- la boucle for

3.2.1 Ex : Reprenons notre exemple précédent sur les listes de la semaine

[11]:
```
#Vérifions si le "jourdi" existe dans notre liste ?
```

3.2.2 Ex : Clef / valeur dans un dictionnaire

[10]:
```
#Quelles clefs contient le dictionnaire ?
```

[12]:
```
#Et quel numéro (indice) a le "lundi" ?
```

[13]:
```
#Methode 2
```

3.2.3 Ex : Ouvrons notre premier fichier en python - et travaillons un peu nos listes / dictionnaires

[14]:
```
"""Etape 0 : chargez les données du document liste_eleve.txt (cette étape est␣
 ↪répétée pour toutes les questions \
qui suivent)"""

with open("liste_eleve.txt", "r") as f :
    print(f.readlines())
```

```
['1 DUPONT Toto L3 Math-Eco ERASMUS\n', '2 MARTIN Elise L3 Analyse-Eco
ERASMUS\n', '3 ZINZIN Antoine L3 Math-Eco\n', '4 BLABLA Nicolas L3 Analyse-
Eco\n', '5 UPSALA David L3 Analyse-Eco Césure\n', '6 DATA Scientist L3 Eco\n',
'7 NANTES Naoned L3 Eco_Appliquée\n', '8 INSTRUMENT Demusique L3 Math-Eco
Déléguée\n', '9 NANANI Nanana L3 Econométrie\n', '10 LACIGALE Etlafourmi L3
Analyse-Eco\n', '11 BABAR Celeste L3 Analyse-Eco\n', '12 LELOUP Etlagneau L3
MIASH']
```

[15]:
```
#Question 1 : lancer un script pour afficher tous les prénoms des élèves de la␣
 ↪classe
#et devant chaque prénom indiqué le numéro attribué à l'élève (ordre␣
 ↪alphabétique des noms de famille)
```

[17]:
```
#Question 2: faites une liste des prenoms de la classes et affichez cette␣
 ↪liste, triée par ordre
#alphabétique

liste_des_eleves = []
```

[16]:
```
#Question 3 : lancer un script pour afficher tous les prénoms des élèves de la␣
↪classe qui
#ont fait Analyse_Eco ou l'an passé
```

[18]:
```
#Question 4 : créez une liste de dictionnaire. Chaque dictionnaire se␣
↪présentant de la sorte :
#{'prenom':<le prenom>, 'nom':<le nom>, 'dernier_diplome_obtenu':<dernier␣
↪diplome>,'filiere':<filiere>}

les_infos_importantes = []
```

[65]:
```
#les_infos_importantes
```

[19]:
```
#Question 6 : affichez la liste des élèves du début de la liste, jusqu'au␣
↪premier élève ayant fait
#Eco l'an passé (à partir du dictionnaire les infos importantes)
```

3.2.4 Remarques sur le mot clef "with"

[20]:
```
#NB : il existe différents modes pour ouvrir un document (r, w, w+, etc.)
#D'autres instructions permettent de lire plus rapidement les lignes (exemple␣
↪ci-dessous)

with open('liste_eleve.txt', 'r') as r :
    print(r.readlines())
```

```
['1 DUPONT Toto L3 Math-Eco ERASMUS\n', '2 MARTIN Elise L3 Analyse-Eco
ERASMUS\n', '3 ZINZIN Antoine L3 Math-Eco\n', '4 BLABLA Nicolas L3 Analyse-
Eco\n', '5 UPSALA David L3 Analyse-Eco Césure\n', '6 DATA Scientist L3 Eco\n',
'7 NANTES Naoned L3 Eco_Appliquée\n', '8 INSTRUMENT Demusique L3 Math-Eco
Déléguée\n', '9 NANANI Nanana L3 Econométrie\n', '10 LACIGALE Etlafourmi L3
Analyse-Eco\n', '11 BABAR Celeste L3 Analyse-Eco\n', '12 LELOUP Etlagneau L3
MIASH']
```

- **Ouverture des fichiers** : **attention** à ne pas écraser, à ouvrir, avec le bon mode, etc.
- En Python, **le mode with** (qui s'utilise assez fréquemment), permet d'appliquer une condition à tout ce qui est dans l'indentation - et uniquement à cela. Pour l'ouverture d'un fichier, cela permet par exemple de rendre implicite la commande close() - et les désagréments qui vont avec (voir par exemple : https://openclassrooms.com/fr/courses/235344-apprenez-a-programmer-en-python/232431-utilisez-des-fichiers)

3.3 Librairie os - Parce qu'il n'est jamais trop tôt pour prendre de bonnes habitudes

Pour manipuler les **chemins vers les différents documents**, il est conseillé de se servir de la **librairie os**.

Celle-ci permet de :

- récupérer le répertoire courant (**os.getcwd()**),
- de récupérer le nom du fichier ou le path du répertoire dans lequel est contenu un document (**os.dirname / os.basename**),
- de lister tous les documents contenus dans un répertoire : **os.listdir()**,
- ou encore de faire de belles jointures (path + nom du document) en évitant les problèmes de version (/ vs \, etc.) : **os.path.join**

3.3.1 Ex: Affichez tous les éléments contenus dans le répertoire courant

[ ]:
```
import os

for element in os.listdir(os.getcwd()):
    print(element)
```

3.4 [Pour aller plus loin] Le format JSON

[73]:
```
#SAUVERGARDER EN JSON
import json

with open('les_infos_importantes.json', 'w') as fp:
    json.dump(les_infos_importantes, fp)
```

3.4.1 JSON !

Le json est un format très pratique d'**échange de données**. Il est utile pour communiquer des informations entre systèmes (quelques soit le langage de programmation) et tend à supplanter son ancêtre (ceci est un abus de langage), le langage à balise .xml.

Il est utilisé pour les interfaces d'échanges entre système (**API**).

En Python, **le json peut se charger comme un dictionnaire** et c'est bien pratique !

3.5 [Pour aller plus loin] Les PEP

How to code well ?

https://www.python.org/dev/peps/pep-0008/