TP 0: Initiation à Python

Python est un langage de programmation généraliste de haut niveau qui peut être téléchargé sur le site python.org et qui fonctionne sous Microsoft Windows, Mac OS et Linux.

1 Installation de Python

Le plus simple est d'installer Anaconda. Vous aurez Python avec tous les paquets dont nous avons besoin, les notebook jupyter et Spyder. Le fichier d'installation fait entre 450Mo et 520Mo, il vous faudra une bonne connexion.
Une autre solution consiste à installer Miniconda, bien plus léger, mais qui n'installe que le strict minimum. Il faudra installer le reste « à la main ».
Vous pouvez aussi accéder à un serveur jupyter avec des notebook python en vous connectant avec vos identifiants habituels à cette adresse : `https://jupyter.mecanique.univ-lyon1.fr` Attention, ce serveur n'est toutefois pas capable de supporter plus d'une cinquantaine de personnes en même temps.

Anaconda

Windows : Télécharger Anaconda ici. Il devrait suffire ensuite de double-cliquer sur le fichier téléchargé et de suivre les indications pour installer Anaconda.

MacOS : Comme pour Windows, télécharger Anaconda ici. Il ne reste plus qu'a double-cliquer sur le fichier téléchargé et de suivre les instructions.

Linux : Il faut télécharger le script d'installation ici.

1. Ouvrir un terminal (dépend de votre distribution Linux). Vous pouvez chercher une application qui s'appelle *Terminal* ou *Shell*.
2. Se déplacer à l'endroit où le fichier a été enregistré :
   `cd chemin/vers/le/fichier/téléchargé`
3. Donner les droits d'exécution au fichier :
   `chmod u+x Anaconda3-2022.05-Linux-x86_64.sh`
4. Exécuter le fichier :
   `./Anaconda3-2022.05-Linux-x86_64.sh`
5. Accepter la licence (appuyer sur la touche ESPACE jusqu'à arriver à la fin et entrer *yes*).
6. Laisser le chemin d'installation par défaut en appuyant sur la touche ENTRÉE (vous pouvez le changer si vous voulez. Si vous ne savez pas, laissez celui par défaut).
7. Répondre *yes* lorsque l'on vous demande si vous voulez initialiser Anaconda 3 en lançant *conda init*.

Vous pouvez maintenant ouvrir un nouveau terminal et lancer les notebook ou Spyder :
`jupyter notebook`
`spyder`

Miniconda

L'installation devrait être similaire à celle d'Anaconda. La page avec les téléchargements se trouve ici. Prendre les versions Miniconda 3, 64 bits. Pour MacOS, la version pkg est la plus simple, la version bash se comporte comme pour Linux.
Une fois Miniconda installé, il faudra aussi installer les paquets dont nous avons besoin, à savoir :

- numpy
- scipy
- matplotlib

Si vous voulez Spyder ou les notebook jupyter, vous pouvez les installer avec conda, ce sont des paquets comme les autres.
Pour installer un paquet sous Linux (ou MacOS, si pas d'application graphique) :

1. Ouvrir un terminal
2. Entrer la commande conda suivante :

   `conda install nom_du_paquet`

Par exemple, pour installer les trois paquets dont nous avons besoin :
`conda install numpy scipy matplotlib`

Avertissement : Cette très brève introduction ne suffira pas à connaître et manipuler les objets de base que l'on utilisera en python. Il est **très fortement conseillé** pour celles et ceux qui n'ont jamais fait de python de chercher des TP d'introduction pour découvrir certaines bases, par exemple en suivant le tutoriel `https://docs.python.org/3/tutorial/index.html`. On se propose ici seulement d'introduire quelques notions utilisées dans le TP 1.

2 Python et l'interface spyder

Python dispose de nombreuses *bibliothèques* de fonctions déjà programmées, notamment :

- NumPy pour l'analyse numérique, en particulier le traitement de matrices et de tableaux de données multidimensionnels, ainsi que les constantes π (`np.pi`) et e (`np.e`) et les fonctions mathématiques usuelles : racine carrée (`np.sqrt`), logarithme (`np.log`) et exponentielle (`np.exp`), fonctions trigonométriques (`np.cos`, `np.sin`, `np.acos`, etc.), partie entière inférieure (`np.floor`) ou supérieure (`np.ceil`), etc. ;
- Matplotlib pour tracer des graphiques ;
- Pandas pour importer, explorer et traiter des données (tutoriel).

Pour travailler avec Python, il est commode d'utiliser l'interface Spyder, qui est installée sur les ordinateurs de l'université, et dont une version *open source* est téléchargeable à l'adresse `https://www.spyder-ide.org`. Si vous l'avez déjà utilisé, vous pouvez aussi utiliser un Notebook.
Cette interface comporte :

- une console (*shell*), permettant d'exécuter des commandes et des scripts Python (en bas à droite),
- un éditeur permettant d'écrire des scripts Python (colonne de gauche), que l'on peut ensuite exécuter dans la console (toujours en bas à droite),
- un espace en haut à droite qui permet d'afficher en particulier un navigateur de fichiers et une fenêtre d'aide (à utiliser sans modération).

3 Quelques notions de base de python

1. Suivez les commandes au dessus pour installer Python3 (avec une préférence par Anaconda). Lancer Spider.

2. *Se repérer* :

   Ouvrir un nouveau *script* (fichier contenant des commandes Python) et l'enregistrer en utilisant un nom se terminant par `.py`. Taper une commande dans le script, par exemple `1+2`. Pour exécuter le script dans la console, utiliser la touche `F5`, ou la touche `F9` pour exécuter la ligne courante ou la sélection.

   Pour obtenir de l'aide sur une fonction, par exemple `print`, on peut utiliser directement le moteur de recherche situé dans la fenêtre d'aide, ou taper dans la console `?print` ou `help(print)`.

   Deux mises en garde : affichage des résultats et importation des bibliothèques

   Mise en garde 1 *Si on écrit une commande dans un script, les calculs sont exécutés mais pas affichés si on ne le demande pas expressément. Par exemple, taper la commande suivante dans un script et l'exécuter (*`F9` *pour exécuter), puis faire de même (*`Entrée` *pour exécuter) dans la console :*

   ```
   5+2
   ```

   Faire de même avec la commande suivante et observer ce qui se passe.

   ```
   print(5+2)
   ```

   À présent, taper dans un script ou dans la console la commande `exp(2)` : on constate que la fonction `exp` n'est pas définie par défaut.

   Mise en garde 2 *Il faut donc importer la bibliothèque* `numpy` *– une fois par session, inutile de le faire à chaque calcul. On importe* `matplotlib` *et* `pandas` *en même temps, ce qui conduit à écrire dans tous les scripts le préambule suivant.*

   ```
   import numpy as np
   import matplotlib.pyplot as plt
   import pandas as pan
   ```

3. *Affectation* : Dans la console, écrire les commandes suivantes (ligne par ligne), et comprendre ce qu'elles font

   ```
   print("bonjour")
   a = 3
   a
   print(a)
   type(a)
   b = 2*a**2+1
   b
   c = [1,2,3,4,5]
   type(c)
   c[0]
   c[1]
   c[2]
   c[:2]
   c[3:]
   ```

4. *Boucle for* : Que fait le code suivant :

```
for i in [0,1,2,3]:
  print(i)
for j in range(4):
  print(j)
```

5. *Commentaires* : On utilise "#" pour commenter un bout de code. Cela peut être utile de chercher le raccourci clavier (souvent "Ctrl+/" ou "Ctrl+Maj+1")

```
a = 1 #ceci est un commentaire
#a = 3 #ceci ne sera pas exécuté
print(a)
#Penser à bien commenter votre code pour pouvoir le relire facilement
```

6. *Condition if* : Que fait le code suivant :

```
a= 8
if a>4:
  print("a est plus grand que 4")
else:
  print("a est plus petit ou egal a 4")
```

7. *Indentation* : Python utilise l'indentation pour comprendre la fin des boucles, conditions, fonctions... Tester le code suivant, et corriger l'erreur.

```
for i in range(10):
if not i==4:
print(i)
```

8. *Aide* : Lorsque vous cherchez comment utiliser une fonction, penser à chercher dans l'aide, et sur internet.

```
s = sum(1,2,3,4) #souleve une erreur
help(sum) #il lui faut un "iterable" par exemple une liste
s = sum([1,2,3,4])
print(s)
```

9. *Variables aléatoires* : Comprendre et expliquer le code suivant (utiliser toutes les ressources d'aide nécessaire pour comprendre ce que font les fonctions) :

```
b = np.random.binomial(1,1/2,30)
print(b)
b = np.random.binomial(1,1/2,30)
print(b)#résultat différent car on simule des Bernoulli, le retour est aléatoire
np.mean(b)
m = np.random.binomial(1,1/2,(4,3))
print(m)
```

10. *Écriture en compréhension* : On adore ou on déteste :

```
l1 = [i**2 for i in range(10) if i%2]
l2 = [i**2 if i%2 else -1 for i in range(10)]
print(l1)
print(l2)
```

Attention : Cet exercice n'est pas suffisant pour faire une bonne introduction à Python, mais a seulement pour but de vous faciliter le démarage. Encore une fois, il est conseillé de s'entraîner, en essayant d'améliorer les exemples, et de découvrir d'autres notions (manipulation de matrices, options pour les graphiques...)

4 Compléments

Écrire dans un script et lancer les commandes suivantes. Interpréter les résultats.

Commandes de base

```
b = np.cos(1); print("b = %s" % b)
# remplacement de %s
# par la chaine qui suit le signe %
```

Création de tableaux NumPy

```
# une liste Python
u = [1, 9, -4, 0.5]
print(2*u)
# un tableau NumPy
v = np.array([1, 9, -4, 0.5])
print(2*v)
print(v[2])
print(v[0])
# la numerotation commence a zero

print(v.size)
print(np.size(v))
print(v.shape)
```

Intervalles et répétitions

```
print(np.linspace(0, 2 * np.pi, 10))
print(np.linspace(20, 1, 10))
x,dx=np.linspace(0,6.3,10,retstep=True)
print(x)
print(dx)

x = np.arange(0, 10, 0.5)
print(x)

dx = 0.5
x = np.arange(0, 10+dx/2, dx)
print(x)

y = np.arange(10, -dx/2, -dx)
print(y)

print(np.repeat(3.2, 4))
x = np.array([[1,2],[3,4]])
print(np.repeat(x, 2))
print(np.repeat(x, 3, axis=1))
print(np.repeat(x, 4, axis=0))
```

Opérations sur les vecteurs

```
A = np.array([1, 2, 3, 6])
B = np.array([0, -4, 9, 4]);
print(np.exp(A))
print(A+B)
print(A*B)
print(A/B)

print(2*A)
print(A+1)
print(A+np.array([1,2,3]))
```

Sélection

```
print(A==2)
print(A[A>2])
```

Sommes et sommes cumulées

```
u = np.arange(1,10,1)
print(np.sum(u))
print(np.cumsum(u))
```

Création et opération sur les matrices

```
A.shape = (2,2); print(A)
print(A.transpose())
C = B.reshape(2,2).T
print(C); print(A @ C)
print(np.matmul(A,C))
print(C[1,]); print(C/A[1,])
```

Tracés de graphiques

Pour tracer le graphique d'une fonction mathématique, on utilise la bibliothèque `matplotlib`, dont la partie pertinent `pyplot` a été importée sous le petit nom de `plt`.

```
x = np.linspace(0, 2*np.pi, 100)
plt.plot(x, np.sin(x))
plt.show()

plt.plot(np.cos(x), np.sin(x))
plt.show()

plt.plot(x, np.cos(x), x, np.sin(x))
plt.show()

plt.plot(x, np.cos(x))
plt.plot(x, np.sin(x))
plt.show()
```

On constate que tous les graphes sont tracés dans la même fenêtre, qui n'apparaît qu'à l'exécution de la commande `plt.show()` (au moins dans les anciennes versions de Spyder). Voici comment ouvrir plusieurs fenêtres.

```
x = np.linspace(0, 2*np.pi, 100)
y = np.linspace(0., 1., 100)

plt.figure(1)
plt.plot(x, np.sin(x))

plt.figure(2)
plt.plot(y, y * np.sin(y))

plt.show()

plt.plot([0, 1], [1, 0], "b")
plt.plot(0.1, 0.9, "r+")
plt.xlabel("x")
plt.ylabel("y")
plt.title("joli graphe")
plt.legend(["segment", "point"])
plt.show()

#Plusieurs graphes dans la même image

fig,ax=plt.subplots(1,2)
ax[0].plot(x, np.sin(x))
ax[1].plot(y, y * np.sin(y))
fig.suptitle("2 jolis graphes")
ax[0].set_title("sinus")
plt.show()
```