Travaux dirigés – Outils Numériques

Feuille numéro 1

Manuel en ligne de *Python* : `https://docs.python.org/fr/3.5/tutorial/`

1 Échauffement

Avant de commencer à utiliser *Python* il va falloir se familiariser avec l'environnement de travail Linux.

1. À l'aide de l'identifiant `me` et du mot de passe `myself2000`, connectez vous sur un ordinateur. Attention, la connection est locale et d'un ordinateur à l'autre vous ne retrouverez pas vos données. Pour les séances suivantes, pensez à prendre une clef USB ou à vous envoyer vos fichiers par email.
2. Cherchez dans le lanceur le *terminal* (Konsole) et ouvrez le.
3. Rappel des commandes de base Linux : *pwd, ls, mkdir, cd, more, cp* et *rm*. Si on ne sait pas comment utiliser une commande, on peut en demander le manuel au système d' exploitation (OS), en tapant : *man _commande_* (ex. : *man ls*).
4. À l'aide de la commande `mkdir` créez un répertoire (dossier) `TD_votrenom_python`. Vérifiez qu'il a été créé à l'aide de la commande `ls`. Entrez dans ce dossier avec la commande `cd` puis créez le répertoire `td1`.
5. Dans le répertoire `td1` vous pouvez désormais lancer *Python* en tapant `python`.

Observation 1 :
— Toutes les opérations que vous effectuez sous *Python* sont perdues dès lors que vous quittez *Python* (commande `exit()`).
— Bien que d'aspects similaires, le *Terminal* et *Python* sont deux environnements différents. En particulier, ils parlent un langage différent et ne répondent donc pas aux mêmes commandes.

Observation 2 :
A la place d'utiliser *Python* "à l'ancienne"', on peut se servir d'un environnement comme *spyder*. Avec *spyder* on peut avoir accès aux valeurs des variables, écrire un *script* et le lancer, etc...

2 Une petite introduction à Python

Taper *spyder* depuis le terminal. On travaillera ici en mode interactif dans la *console Ipython* qui apparait dans la fenêtre en bas à droite.

— Verifier que les commandes *ls, cd, pwd* marchent également depuis la console python.
— Définir une variable (ex. : "a=2") et lui ajouter 3. Puis donner à la variable l'inverse de sa valeur. On peut vérifier la valeur de la variable en faisant "a [ENTER]" ou en utilisant "print(a)". [objectif : comprendre le concept d'assignation de valeur, e.g. $a \leftarrow 5$, et la différence avec l'utilisation d'une calculette)
— Essayer d'ajouter à la variable a une variable b. Cela ne fonctionnera que si la variable b est définie.
— Taper "%reset" [enter], puis vérifier les valeurs des variables définies.

3 Une petite introduction à la programmation avec Python

Dans l'éditeur (fenêtre de gauche), écrire des fonctions pour :

1. Calculer la somme de deux nombres a et b [objectif : apprendre à définir une fonction]
2. Indiquer si a est plus grand que b ou l'inverse. Après, démander à Python si $0.1 + 0.1 == 0.2$ et $0.1 + 0.2 == 0.3$. Discuter la différence. [objectif : apprendre à utiliser "if"] [objectif : apprendre à utiliser "if"]
3. Calculer la somme de 1^d, 2^d, ..., n^d avec n et d deux nombres [objectif : apprendre à utiliser "for"]

4. Calculer la factorielle d'un nombre n [objectif : apprendre à utiliser "while"]
5. Estimer les valeurs de $\exp(x)$, en utilisant la fonction factorielle de la question précédente [objectif : faire développement limité de $\exp(x)$ jusqu'au ordre 4 et calculer la valeur]
6. Trouver le maximum d'un ensemble de nombres $(a, b, c, \ldots, m)$ [objectif : définir un vecteur (avec *numpy*) et utiliser "for" et "if" pour trouver le maximum]. Pour définir les vecteurs : "import numpy as np", puis par exemple"vecteur=np.array([10, 8 , 11, 7 , 6 , 1])". Comparer avec la commande "np.max()".
7. Ordonner un ensemble $(a, b, c, \ldots, m)$ [un petit défi algorithmique]

4 Tracé de fonction d'une variable

On peut estimer $\exp(x)$, $\log(x)$ et $\sqrt{x}$ en faisant un dévelopment limité comme pour l'exercice précédent. On veut comparer les développements limités avec les approximations déja définies dans Python (à l'intérieur de "numpy") : np.exp(x), np.log(x) et np.sqrt(x). Tracer la courbe du développement limité de $\exp(x)$ et np.exp(x) en fonction de x, à l'aide de *matplotlib* ("import matplotlib.pyplot as plt", puis "plt.plot(x,y)" et "plt.show()".