Feuille de TP 1

Initiation à Python

1 Survivre avec le terminal

Unix est une famille de systèmes d'exploitation (ou OS pour *operating system*). Un système d'exploitation gère le fonctionnement et la répartition des différentes ressources, que ce soit pour l'utilisateur ou les processus (dont les logiciels). Ces ressources sont aussi bien de la puissance de calcul (processeur ou CPU) que le stockage (mémoire vive, disque dur, clef usb, cache, ...) et les différents périphériques (clavier, souris, écran, enceintes, ...).

Il apparaît en 1969 et désigne aujourd'hui toute la famille des OS qui en sont dérivés, dont macOS, GNU/Linux, BSD, Android (le seul OS connu qui n'appartient pas à cette famille est Windows). Ces exemples montrent bien que cet OS a la particularité d'être multi-plateforme : il fonctionne aussi bien sur des smartphones, des ordinateurs portables et les très gros serveurs de données des agences spatiales (qui stockent et traitent les images très haute résolution des satellites et télescopes). Ses autres spécificités sont d'être multi-utilisateurs (chaque personne est « identifiée » et son utilisation du système peut être plus ou moins limitée - il peut par exemple être interdit d'installer de nouveaux logiciels) et multi-tâches (plusieurs programmes peuvent être exécutés en même temps).

GNU/Linux est de plus un OS libre. En particulier, chacun peut l'utiliser, le modifier, le copier et le distribuer librement.

Les données sauvegardées sur un ordinateur sont stockées sous forme de fichiers, qu'il s'agisse de texte, tableau, base de donnée, image, musique, vidéos, etc. Sur les fichiers *complexes*, l'extension du fichier détermine sa nature. Par exemple, un fichier de musique pourra porter l'extension `.mp3` s'il est encodé au format mp3. Si un autre encodage est utilisé, le fichier aura une autre extension.

Les fichiers les plus simples sont stockés sous forme de texte brut, sans mise en page (ce n'est donc pas le cas des documents sauvegardés par un traitement de texte comme LibreOffice Writer ou World). Pour de tels fichiers, l'extension n'est pas obligatoire mais l'usage veut qu'une extension soit choisie en fonction du contenu du fichier. Les « codes » python sont enregistrés avec l'extension `.py`, les page internet avec l'extension `.html`, etc. Sans cette indication, il devient difficile de deviner qu'un fichier contient un code, ou même de savoir comment le lancer (l'exécuter).

Afin de pouvoir retrouver ces fichiers, ils sont organisés en dossier ou répertoire. Ces derniers sont des « boîtes » contenant différents fichiers ... ou d'autres répertoires. Cela permet au final d'obtenir une arborescence de fichiers structurée et organisée. Cette dernière est nécessaire au système d'exploitation pour bien fonctionner : il doit savoir trouver facilement les fichiers nécessaires à son bon fonctionnement (noyau, interface graphique), tout comme les fichiers temporaire qu'il devra régulièrement supprimer (sans effacer pour autant les bases indispensables à son bon fonctionnement).

Tous les dossiers d'un utilisateur peuvent être stockés dans son `home` (répertoire qui porte généralement le même nom que l'utilisateur) ou dans des sous-dossiers de ce dernier. Les environnements de bureau les plus courants créent des sous-répertoires comme `Documents`. **Comme nous réutiliserons des fichiers d'une séance à l'autre, il est primordial de conserver vos codes de façon organisée et de les trier dans des dossiers.**

Exercice 1.1 : B.A. BA du shell

1. Ouvrir un navigateur de fichier. Se placer dans `Documents` (ou dans un sous dossier de ce dernier si vous préférez) et y créer un dossier `M208`.
2. Se déplacer dans le shell : ouvrir un terminal, taper la commande `pwd` pour connaître l'emplacement courant. A l'aide de la commande `cd ~/Documents` vous pouvez vous rendre dans le dossier Documents.
3. La commande `ls -lhrt` permet de visualiser le contenu de ce fichier. Pour aller dans un dossier, il faut taper `cd nom_dossier`. La commande `cd` sans autre option vous ràmène dans votre dossier `home` (le dossier contenant Documents, Photos, Bureau, etc).

   Enfin, notez les commandes `cp fichier1 fichier` qui permet de copier un fichier1 dans un fichier2. Enfin, la commande `mkdir nom_dossier` permet de créer un dossier qui s'appellera `nom_dossier`. S'en servir pour créer un dossier TP1.
4. Télécharger dans le dossier `Documents/M208/TP1`, le code *test.py* à l'adresse : `http://matplotlib.org/examples/animation/double_pendulum_animated.py`. Ouvrir un terminal et se rendre dans ce dossier. Exécuter le code python à l'aide de la commande `python double_pendulum_animated.py`.

2 Premier pas avec l'interpréteur python

Python est un langage conçu pour être interprété. Un interpréteur exécute au fur et à mesure les instructions du code sources, sans étape supplémentaire (pas d'étape de compilation).

Exercice 1.2 : Une super calculatrice

Ouvrir une console ipython à l'aide de la commande `ipython` (ou `ipython3`). Le terminal lance alors une console python interactive.

L'écran affiche d'abord la version de python (normalement 3.X avec X entre 0 et 6, sinon ré-essayer avec `ipython3`) puis une aide minimaliste. Il propose ensuite à l'utilisateur d'entrer ces propres instructions (c'est l'invite de commande précédée de `In`). Il va les exécuter au fur et à mesure en préfixant la ligne contenant le résultat par `Out`.

Taper les commandes suivantes :

```
1+3
2*3
1+3*2
(1+3)*2
0.5*2
2/1
2 > 1
not True
True or False
True and False
```

Exercice 1.3 : Un monde de variables

Taper les commandes suivantes dans la console (il est possible de faire un copier-coller ligne par ligne) :

1. Les variables

```
print("\nLes␣Variables")
# Ceci est un commentaire
a = 2
b = 3
print("a␣=␣", a, "␣et␣b␣=␣", b)
a+b
```

```
print("a+b = {0:d}".format(a+b))
print("a*b = {0:d}".format(a*b))
print("a puissance b = {0:10.3e}".format(a**b))
a +=2 # Ajoute 2 et enregistre la nouvelle valeur dans a : raccourci pour "a = a+2"
print("a = ",a)
a -=2 # Retranche 2 : raccourci pour "a = a-2"
print("a = ",a)
a *=2 # Multiplie par 2 : raccourci pour "a = a*2"
print("a = ",a)
a /=2 # Divise par 2 : raccourci pour "a = a/2"
```

2. Les types

```
a = 2
print(type(a))
b = 2.3
print(type(b))
c = float(a)
print(type(c))
d = 2.
print(type(d))
a = True
print(type(a))
print(2.==b)
print(type(2.==b))

a = 2
a += 1
print("a=", a)
print(type(a))
a *= 2.
print("a=", a)
print(type(a))
```

Les variables ont un type que python détermine tout seul et qui peut être modifié (automatiquement) au cours du programme.

3. Division : par défaut, la division `/` est la division décimale ; la division entière s'écrit `//`. Pour avoir le reste, il faut utiliser la commande `%` :

```
a = 3
print(a/2, type(a/2))
print(a//2, type(a//2))
print(a%2)
```

Exercice 1.4 : Chaînes de caractères

1. Saisir les instructions suivantes (après avoir exécuté la commande input il faut saisir votre prénom) :

```
name = input("Entrer votre prénom : ")
print("Bienvenue",name,"!")
```

2. Répéter les mêmes instructions que précédemment mais en saisissant des espaces supplémentaires après votre nom. Que se passe-t-il ? Ré-essayer avec le code :

```
name = input("Entrer votre prénom : ")
name = "Bienvenue "+name.rstrip()
print(name)
name += " !"
```

A quoi correspond l'addition ?

3. a) Comparer les deux codes suivants (saisir 2 à chaque fois) :

```
a = input("Saisir un nombre entier : ")
print(a+2)
```

```
a = int(input("Saisir un nombre entier : "))
print(a+2)
```

b) Saisir le nombre décimal "0.5". Que se passe-t-il ? A l'aide de la commande `float`, proposer un code demandant à l'utilisateur de saisir un nombre décimal.

4. Demander à l'utilisateur de saisir un nombre puis afficher la somme de ce nombre avec 2. Pour cela on pourra utiliser la commande `float(a)` qui converti la variable `a` en nombre décimal. Faire le même test avec la commande `int` mais en saisissant un nombre décimale à l'invite (par exemple 0.5).
5. Demander à l'utilisateur de saisir la longueur de la base d'un triangle puis sa hauteur. Afficher alors l'aire du triangle.