Introduction à la programmation, cours 1

Python est un langage relativement facile d'accès ayant une syntaxe plus user friendly que d'autres langages comme C++. Il y a deux version de Python, Python2 et Python3. Les différences entre les deux se situent principalement au niveau de la syntaxe. Nous, on va utiliser Python2 car c'est la version installer par défaut sur les machines de l'ENS.

Pour savoir quelle version de Python vous avez, ouvrez un terminal. Pour se faire :

> **En classe:** cliquer sur l'icône Terminal qui se trouve sur le bureau.
> **Sur un Mac:** ouvrir le dossier "Application", puis le dossier "utilités" et double-cliquer sur "Terminal"
> **Sur Windows:** Cliquer sur "Start". Dans la ligne de recherche, taper "cmd" et "Entrée"

Tapez « python » dans le terminal, le message suivant s'affiche :

```
Python 2.7.10 (default, Aug 17 2018, 19:45:58)
[GCC 4.2.1 Compatible Apple LLVM 10.0.0 (clang-1000.0.42)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
[1]+  Stopped                 python
```

Si la première ligne indique Python 2... vous avez python 2. Si vous avez votre machine perso et que vous utilisez Python3, il faudra faire des ajustements par rapport aux codes qu'on vous donne.

Premier programme : Hello, world!

Notre premier programme va, c'est la traduction, afficher « Hello, world ! » à l'écran. On vous guide pas à pas :

1. Ouvrir un terminal.
2. Créer un nouveau dossier en tapant la ligne de commande suivante dans le terminal :
   `mkdir Initiation_a_la_programmation` and press enter .
   mkdir = make directory = créer un dossier
3. Il faut maintenant se placer dans le nouveau dossier , tapez dans le terminal :
   `cd Initiation_a_la_programmation` and press enter.
   cd = change directory
4. Chaque dossier a une adresse (appelée « chemin »). Pour savoir dans quel dossier vous vous trouvez (ce dossier s'appelle « dossier courant ») et quel est son chemin, taper la commande `pwd` dans le terminal, and press enter. Noter le chemin. Ne fermer pas le terminal.
   pwd = print working directory
5. On va maintenant ouvrir un éditeur de texte.
   - **En classe:** Double-cliquer sur l'icône Gedit editor qui se trouve sur le bureau.
   - **Mac & Windows:** Télécharger un éditeur de texte, par exemple Sublime 2
6. Dans l'éditeur de texte, taper le code suivant :

   ```
   print "Hello, world!"
   ```

   Sauvegarder votre fichier sous le nom `helloworld.py` dans le dossier que vous avez crée à l'étape 2 (vous pouvez utiliser le chemin pour le trouver).
7. Retourner dans le terminal. Taper `ls` dans le terminal, ceci va afficher la listes des dossier et des fichiers se trouvant dans le dossier courant. Vérifier que le fichier helloworld.py y est. Tapez `python helloworld.py` dans le terminal and press enter. Tadaamm, vous venez d'exécuter votre premier programme.

Terminal: les lignes de commandes les plus utiles :

Faites (pleins) de testes de ces mots clés pour essayer de comprendre de quoi on parle.
Définition : on appelle *dossier courant* le dossier dans lequel on est.
cd .. – Quitter la dossier courant et aller dans le dossier parent.
ls – Affiche la liste des fichiers et dossiers du dossier courant
tab (la touche du clavier) – à la place de taper le nom complet d'un fichier ou d'un dossier, taper les premières lettre puis appuyer sur la touche tab du clavier. Le reste du nom va s'écrire automatiquement comme par magie.
python – Si vous voulez tester des lignes de codes vite fait, sans avoir à utiliser un éditeur de texte, vous pouvez faire la chose suivante : tapez python dans le terminal and press enter. Bim, changement de mode, le terminal s'est transformé en un genre d'éditeur de texte python.
Taper par exemple print "Yeah man!" and press enter.
Pour sortir de ce mode press control + z.

Types et variables

Python est un langage non *typé*. C'est à dire que lorsque vous définissez une variable, pas besoin de définir son type. Voici les trois principaux types :

Les nombres. Il y a deux types de nombres, les entiers et les flottants (les nombres à virgules). Pour les entiers, on utilise la syntaxe suivante :

```
mon_entier = 13
print mon_entier
```

Pour les flottants il faut ajouter un « . », ainsi Python considère notre variable comme un flottant et non comme un entier, on verra des exemples où c'est important :

```
mon_flottant = 7.0
print mon_flottant
```

Strings. Les *Strings* sont des chaînes de caractères. Il y a deux façons de les définir :

```
chaine_1 = 'hello'
print chaine_1
chaine_2 = "hello"
print chaine_2
```

La différence est subtile, je vous laisse analyser laquelle est-elle en vous aidant des lignes de codes suivantes (**Remarque : à chaque fois qu'on voit une nouvelle notion, on vous donne un exemple. Ensuite, il faut que vous jouiez avec, que vous le modifiez pour bien la comprendre et vous l'approprier).**

```
chaine_1 = "L'utilisation des apostrophes ne pose pas de problème"
print chaine_1
chaine_2 = 'Il a dit, "Utiliser des apostrophes ne pose pas de problème"'
print chaine_2
```

Exercice 1. qu'est-ce que le script suivant fait ? Corrigez-le. Print "Yeah man!"

Exercise 2. Déclarer une chaîne de caractère (ma_chaine), un flottant (mon_nombre) et un entier (mon_entier). La chaîne doit contenir « kamoulox », le flottant doit contenir 0,07 et l'entier doit contenir 8.

```
# modifier ce code
ma_chaine= None
mon_nombre = None
mon_entier = None
print ma_chaine, mon_nombre, mon_entier
```

Remarquer que dans cet exemple la fonction *print* a trois *arguments*.

Opérateurs basiques

Python a les opérateurs arithmétiques classiques :

```
nombre = 1 + 2 * 3 / 4.0
print nombre
reste = 11 % 3
print reste
carre = 7 ** 2
cube = 2 ** 3
print carre
print cube
```

Il existe aussi des opérateurs sur les chaînes de caractères:

```
une_chaine = "Hello" + " " + "world!"    # créer une chaîne “Hello world!”
print une_chaine
print len(une_chaine)     # affiche la longueur de la chaîne, i.e. le nombre de symboles qu’elle contient (y compris les signes de ponctuations et les espaces)
print une_chaine.index("o")       # affiche la position de la lettre « o ». Quelle est la position de la première lettre?
print une_chaine.count("l")       # affiche le nombre de fois que la lettre « l » apparaît.
print une_chaine[3:7]     # affiche les symboles indexés de 3 à 7.
print une_chaine.startswith("Hello") # affiche « True » si une_chaine commence par « Hello », False sinon.
print une_chaine.endswith("end") # je vous laisse deviner ce que ça fait
quelques_mots = une_chaine.split(" ") # Je vous laisse encore une fois deviner. Faites des testes. Ce qu’il y a entre parenthèses est le séparateur, changer-le voir ce que ça donne.
print quelques_mots
```

Remarquez l’utilisation de # pour ajouter des commentaires aux codes. Quand vous codez, il est **impératif** de bien commenter votre code.

Les listes

Une *liste* est une structure de données, qui est un genre de tableau. On peut y mettre n’importe quel types de variables, et autant de variable qu’on veut. De même que pour le reste, il existe des fonctions pour manipuler les listes (cf ci-dessous). Les listes sont définis grâce a des crochets.

NB : En Python il faut respecter l’i**ndentation des bloques**. Utiliser la touche *tab* qui corresponds à 4 espaces.

```
liste_de_nombres = []   #créer une liste vide
print liste_de_nombres
nombres.append(1)       #la méthode « append » permet d’ajouter un élément à la fin de la liste
print liste_de_nombres
numbers.append(2)
print liste_de_nombres

print liste_de_nombres [0]     # on peut facilement accéder au i ième élément d’une liste grâce aux crochets
print liste_de_nombres [1]         # testez ce qu’il se passe si vous essayez d’afficher nombres[2]. Regarder bien quel erreur cela donne.

lettres = ['a', 'b', 'c']
for x in lettres:
    # attention à l’indentation
    print x         # admirer comme il est facile de parcourir une liste
```

Un opérateur sur les listes :

```
nbr_pairs = [2,4,6,8]
nbr_impairs = [1,3,5,7]
tous_les_nombres= nbr_pairs + nbr_impairs
print tout_les_nombres
```

Exercice 3. On a trois listes. En utilisant la méthode « append » , ajouter 1,2 et 3 à la liste « nombres », et les mots « hello » et « word » à la liste « chaine ».
Il y a aussi une variable « deuxieme_nom » dans laquel vous mettrez le second nom de la liste « noms » en utilisant la technique des crochets.

```
nombres = []     #créer une liste vide nommée nombres
chaine = []
noms = ["John", "Eric", "Jessica"]

# écrire votre code à partir d'ici, modifier None
deuxieme_nom = None

print nombres
print chaine
print deuxieme_nom
```

Manipuler les chaînes

Si vous voulez vous la racontez un peu :

```
# ce code affiche « Ouaich Pedro"
nom = "Pedro"
print "Ouaich, %s!"  %nom
```

On peut même faire la même chose avec plusieurs arguments de plusieurs types différents :

```
# ce code affiche « Pedro à 23 ans »
nom = "Pedro"
age = 23
print "%s a %d ans." % (nom, age)
```

ça marche aussi avec les listes :

```
ma_liste = [1,2,3]
print "Une liste: %s" % ma_liste
```

Voici la règle :

%s – chaine de caractères ou tout autre objet ayant une représentation en string, comme les listes par exemple.

%d - Entier

%f - flottants

Exercice 4. You will need to write a format string which prints out the data using the following syntax: Hello John Doe. Your current balance is $53.44.