Préambule

En 1989, le hollandais **Guido van Rossum** commence le développement du langage de programmation Python. Python est un langage **multi plateforme**, c'est-à-dire disponible sur plusieurs architectures (compatible PC, tablettes, smartphones, ordinateur low cost Raspberry Pi...) et systèmes d'exploitation (Windows, Linux, Mac, Android...). Le langage Python est gratuit, sous **licence libre**. C'est un des langages informatiques les plus populaires avec C, C++, C#, Objective-C, Java, PHP, JavaScript, Delphi, Visual Basic, Ruby et Perl (liste non exhaustive). Actuellement, Python en est à sa version 3. Cependant, la version 2 est encore majoritairement utilisée. Attention : Python 2 n'est pas compatible avec Python 3 ! Python est un langage de script. Grossièrement cela signifie que les commandes sont exécutés par le programme Python lui même plutôt que par le processeur. Elles n'ont pas besoin d'être compilées pour être exécutées. Il est donc plus simple de développer en Python qu'en C ou en Java, par exemple. La syntaxe de Python ne reprend pas la syntaxe la plus courante, dérivée du langage C. Il faut donc un temps d'adaptation (assez court) pour commencer à développer en Python si on a déjà développé en C, Java etc. Le trajet inverse est similaire.

Que peut-on faire avec Python ?

Beaucoup de choses !

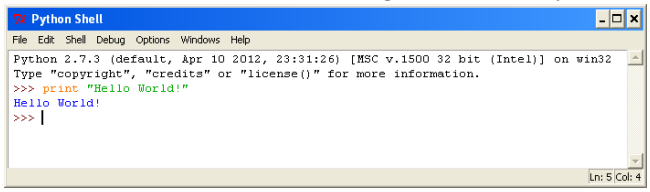
- du calcul scientifique (librairie `NumPy`)
- des graphiques (librairie `matplotlib`)
- du traitement du son, de la synthèse vocale (librairie `eSpeak`)
- du traitement d'image (librairie `PIL`), de la vision artificielle par caméra (framework `SimpleCV`)
- de la bio-informatique (librairie `Biopython`)
- des applications avec interface graphique GUI (librairies `Tkinter`, `PyQt`, `wxPython`, `PyGTK`...)
- des jeux vidéo en 2D (librairie `Pygame`)
- des applications multi-touch (framework `kivy` pour tablette et smartphone à écran tactile)
- des applications Web (serveur Web `Zope` ; frameworks Web `Flask`, `Django`)
- interfacer des systèmes de gestion de base de données (librairie `MySQLdb`...)
- des applications réseau (framework `Twisted`)
- communiquer avec des ports série RS232 (librairie `PySerial`), en Bluetooth (librairie `pybluez`)...
- ...

Des **dizaines de milliers** de librairies sont disponibles sur le dépôt officiel `PyPI`.

Installation

Au LYCEE des FLANDRES Par défaut, Python 2.7 **devrait** être installé sur vos machines. Si une autre version de Python existe (par exemple 2.5 ou 3.4) ce n'est pas forcement gênant. La majorité des instructions sont identiques et il n'est pas toujours indispensable de mettre à jour. Pour débuter ce cours vous pouvez utiliser la version existante. Il arrive souvent que les raccourcis du menu démarrer n'apparaissent pas. Il faut alors chercher à partir de la racine du disque dur principal **c:** pour localiser Python... N'étant pas maître des installations, je ne peux rien y faire. D'autre part nous utiliserons souvent des outils en ligne permettant de travailler directement dans le navigateur et simplifiant considérablement l'échange de documents : repl.it et google colab.

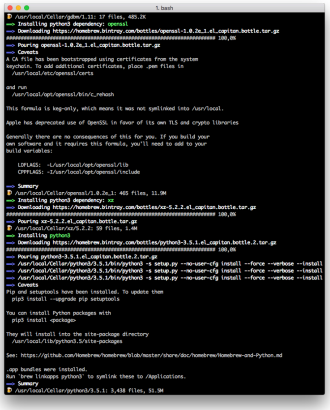
Sous Windows Sous Windows, pour installer Python avec l'**environnement de développement IDLE**, il suffit de télécharger puis d'exécuter le fichier d'installation qui se trouve sur le site officiel : https://www.python.org/downloads/windows Une fois installé, vous pouvez lancer IDLE en allant dans : Démarrer → Programmes → Python → IDLE (Python GUI)



Sous Linux Python est pré-installé sur la plupart des distributions Linux.

SOUS MAC OSX Je n'ai pas de Mac donc c'est à prendre avec des pincettes... Cependant d'après ce que je lis il suffit d'utiliser le gestionnaire de paquets HomeBrew :

```
$ brew install python3
```



[caption id="" align="aligncenter" width="820"]

L'installation de Python avec HomeBrew (c)[/caption]

IDLE IDLE est un environnement de développement intégré (IDE en anglais : Integrated Development Environment) pour Python. IDLE propose un certain nombre d'outils :

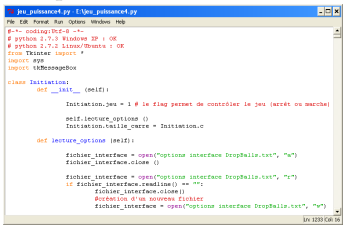
- un éditeur de texte (pour écrire le programme)
- un interpréteur (pour exécuter le programme)
- un débogueur (pour tester le programme)

Il existe d'autres IDE pour Python : Thonny, Sublime Text, Atom, VS Code, Spyder ...

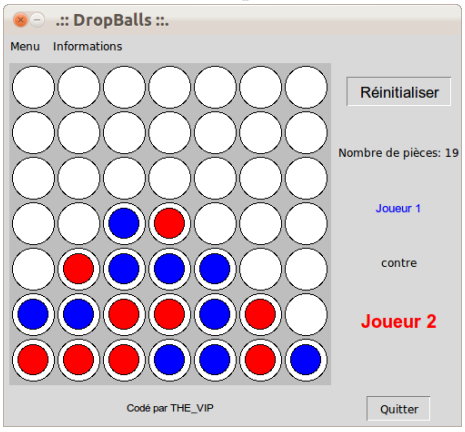
Scripts

Un programme est une séquence d'instructions. Dans le cas d'un programme en langage Python, on parle souvent de **script Python**. Un script se présente sous la forme d'un fichier texte avec l'extension **.py**

Jeu de puissance 4 Voici un exemple de script Python : jeu_puissance4.py Enregistrer ce script sur votre bureau. Ouvrir IDLE : Démarrer → Programmes → Python → IDLE (Python GUI) File → Open → jeu_puissance4.py L'éditeur de texte s'ouvre avec le code source du script (environ 1000 lignes de code, soit plusieurs dizaines d'heures de travail pour un développeur expérimenté) :



Pour exécuter le script : Run → Run module (ou touche F5) A vous de jouer !



Python et les caractères accentués

Par défaut, Python 2 ne gère pas les caractères accentués dans les scripts. La version 3 de python résout ce problème en autorisant la majorité des caractères courant. Cependant, évitez d'utiliser des variables contenant des caractères spéciaux, réservez les aux chaînes de caractères.

```
>>> é
  File "<stdin>", line 1
    é
    ^
SyntaxError: invalid syntax
```

On prendra donc soin, d'insérer en haut de chaque script (par exemple : **exemple.py**) l'encodage suivant :

```
# coding=utf-8
```

Oui, il est précédé d'un # et donc paraît être un commentaire mais les premières lignes sont interprétées comme des réglages. On trouvera souvent le préambule suivant :

```
#!/usr/bin/env python
# coding=utf-8
```

ou #! /usr/bin/python . Cette première ligne, appelée SHEBANG, permet au système d'exploitation linux de reconnaître un script exécutable. De manière générale, on recommande d'éviter les accents, cédilles etc. dans les codes sources. C'est parfois impossible, aussi doivent-ils être traités avec prudence en définissant un encodage.

Python et l'indentation

En Python la structure d'un programme est **définie par son indentation**. Ce sont les espaces qui séparent le bord gauche de l'éditeur du début de l'instruction.

```
if 2>1:
    print("bravo !")
```

Le print précédent est indenté par 4 espaces. C'est le format standard d'indentation. On trouvera parfois des variantes telle qu'une tabulation, 2 espaces, 3 espaces... On veillera donc à se méfier des copier coller hâtifs en Python. Si l'indentation n'est pas la même entre la source et la destination, le programme va renvoyer une erreur. Que se passe-t-il si on n'indente pas ?

```
>>> if 2>1:
... print "bla"
  File "<stdin>", line 2
    print "bla"
        ^
```

```
IndentationError: expected an indented block
```

Python renvoie une erreur. Vous pouvez maintenant poursuivre votre découverte.