installation de Python et démarrage

1.1 Python

1.1.1 Un programme c'est quoi ?

L'objectif d'un langage de programmation est de donner des ordres à un ordinateur ou à tout appareil similaire comme les tablettes, les smartphone, les appareils photo numériques, etc.

Le langage de base de l'informatique est constitué d'une série de 0 et de 1 (en réalité du courant électrique), seule possibilité de communiquer directement avec les

machines. Comme il est très compliqué pour un humain d'écrire et de lire des 0 et des 1, on a inventé des langages de programmation qui permettent d'avoir une écriture plus proche de nos capacités. Suivant les langages un **compilateur** ou un **interpréteur** permettent alors de transcrire ce langage en 0 et 1 et de communiquer avec l'ordinateur.

En général on utilise un **éditeur de développement intégré** (EDI) pour écrire les programmes : cet éditeur permet d'écrire les lignes du programme (*lignes de code*) tout en contrôlant la **syntaxe** (la manière d'écrire le programme) ; une fois le programme écrit on l'exécute dans l'EDI et on le corrige (*déboguer* ou *debugger*) suivant les indications de ce dernier.

Typiquement un programme professionnel peut contenir plusieurs milliers de ligne de code voire plusieurs millions… On ne fait pas tout ça d'un seul coup ni tout seul évidemment ! Il y a des équipes avec des chefs de projet, des développeurs, des analystes, etc. qui vont s'attaquer à la fabrication du programme. La réalisation d'un projet peut durer plusieurs années, être remanié, modifié, amélioré… et pourtant il peut rester des bugs ou des failles malgré tout le soin apporté.

Ceci dit, pour apprendre à coder on n'a encore jamais trouvé mieux que d'apprendre en solo et de réaliser soi-même un certain nombre de programmes afin de maîtriser les principes de base, et ceci quel que soit le langage utilisé !

Pour notre part nous ferons des programmes relativement courts (le plus gros doit faire environ 400 lignes de code) mais en essayant d'être sûrs qu'ils fonctionnent bien…

1.1.2 Les langages et Python

Il existe une quantité colossale de langages, la plupart créés dans un but bien déterminé : faire du calcul, gérer des données, piloter un drone, naviguer sur Internet, commander une machine à laver, etc. Il existe des langages de bas niveau, très proches du langage directement compréhensible par les machines comme l'**assembleur**, des langages intermédiaires comme le **C** nécessitant beaucoup de travail de la part des programmeurs pour fonctionner dans des environnements divers ou des langages de haut niveau comme **Java** ou **Python** disposant de nombreux outils permettant de travailler rapidement quelle que soit la situation.

Les langages de programmation ressemblent par divers côtés aux langages humains : il y a un **vocabulaire**, souvent constitué de mots anglais (*print*, *while*, *for*, *and*, …), une **grammaire** qui donne les règles à utiliser et une **sémantique** qui permet de donner un

sens aux lignes successives du programme. C'est d'ailleurs assez compliqué de basculer un programme d'un langage à un autre.

En général les langages utilisent des bibliothèques (*library*) de programmes déjà écrits et validés par la communauté des développeurs : on importera le code déjà utilisable directement à partir de l'EDI.

Python est un langage de programmation **interprété** : en général il y a deux types de méthodes pour programmer, la *compilation* et l'*interprétation*. La compilation fournit un logiciel **exécutable** (on dit un *exécutable* simplement) directement utilisable par l'ordinateur alors que l'interprétation nécessite un intermédiaire, appelé **interpréteur**, qui va prendre en charge les instructions et les transcrire de manière à ce qu'elles soient utilisables par la machine. Le prototype des langages compilés est le C et ses divers dérivés comme C++ ou Objective C.

On pourrait comparer compilation et interprétation aux situations « se parler en face à face » ou « se parler via un téléphone » : le téléphone jouant le rôle de l'interpréteur.

L'inconvénient de l'interprétation est que ça ralentit nettement le déroulement des programmes puisqu'il y a une étape supplémentaire entre les instructions et la réalisation. Une méthode utilisée en Java (langage très utilisé dans l'industrie) comme en Python est de compiler tout ou partie des instructions fréquemment utilisées à l'avance et d'utiliser ces parties précompilées au moment de l'éxécution. On ne parle pas d'ailleurs de *programme* Python mais de *script* Python : il s'agit simplement d'un texte répondant à certaines contraintes qui sera lu par l'interpréteur.

Ceci dit les ordinateurs vont tellement vite en général que tu ne te rendras même pas compte d'un éventuel ralentissement, surtout avec le type de programmes que tu risques de réaliser.

Les parties précompilées sont logées dans des *modules* ou *librairies* (groupe de modules) déjà vérifiées que l'on déclarera vouloir utiliser au début d'un programme.

Les avantages de Python par rapport à Java sont multiples : facilité d'écriture et de lecture des programmes, grande bibliothèque de modules, gratuité et liberté totale d'utilisation du langage et de ses dérivés.

Il existe des pseudo langages encore plus simples d'accès ou faciles d'utilisation comme **Scratch** avec une interface très conviviale mais ne présentant pas autant d'intérêt que Python au niveau du potentiel d'activités possibles et surtout des capacités d'apprentissage.

1.1.3 Installer Python

Tu peux installer Python tout seul, c'est très simple mais il vaut mieux te faire aider par un adulte surtout si tu utilises un Mac (Apple) ou si tu n'as pas les droits d'administration sur ton ordinateur.

De quoi as-tu besoin pour faire tourner Python ?

* d'un ordinateur sous Windows ou OSX Apple ou Linux… ce n'est pas une blague, pour l'instant Python ne fonctionne pas très bien sur les tablettes[1] ou les smartphones (qui ont presque les mêmes capacités qu'un ordinateur) mais pour lesquels le *système d'exploitation* n'est pas tout à fait au niveau ;

* l'interpréteur Python qui traduira tes ordres pour la machine ;

* un éditeur de texte qui enverra les instructions à l'intepréteur.

Facultatif mais en fait indispensable : l'EDI (Environnement de Développement Intégré) qui fait tout ça et même beaucoup plus. Pour Mac ou Linux le plus simple est d'utiliser l'EDI fourni avec la distribution Python et normalement déjà installé, nommé **IDLE**. Pour IDLE tu peux consulter la page :

https://hkn.eecs.berkeley.edu/~dyoo/python/idle_intro/indexfr.html

Pour Windows on va travailler avec un autre EDI appelé **Pyscripter** : télécharger sur le site **pythonpourenfants.free.fr** le programme appelé *PythonPourEnfants.exe*, le lancer et te laisser guider pour l'installation ; tout ce qu'il te faut pour utiliser les scripts du livre sera alors disponible dans le dossier choisi (normalement *Python pour Enfants*).

1.1.4 Quelle version de Python ?

Quand tu vas sur le site officiel de Python (python.org) tu trouves une multitude de versions disponibles : en fait il y a deux versions principales pas totalement compatibles, la version 2 et la version 3. Chaque version a subi diverses améliorations et on en est actuellement[2] à 2.7.8 pour la V.2 et à 3.4.2 pour la V.3. La version 3 ne comprend pas tout ce que fait la version 2 et réciproquement, même si la plus grande

[1] Il existe un interpréteur Python pour Android disponible sur le market : QPython - Python for Android qui permet de lire les scripts de même pour Apple IOS.

[2] En informatique les versions d'un logiciel évoluent plus ou moins vite. En général on a un numéro de version principal (en 2014 Windows en est à sa version 8 et va vers la 9) suivi d'un numéro secondaire qui indique diverses améliorations importantes et éventuellement des numéros supplémentaires indiquant des changements minimes.

partie des instructions fonctionnent dans les deux versions. Les problèmes viennent d'ailleurs plutôt des modules qui ont besoin d'être adaptés à chaque version, ce qui n'est pas encore forcément le cas.

Idéalement il vaudrait mieux utiliser la version 3 mais en fait ça n'a pas vraiment d'importance, aussi pour profiter du maximum de programmes disponibles sur Internet on va travailler avec la version 2.7. Sur le site il te suffit de télécharger l'installateur et te laisser guider.

1.1.5 Les fichiers de Python

Le disque dur d'un ordinateur est organisé en dossiers dans lesquels on peut trouver d'autres dossiers ainsi que des fichiers de données. Pour les programmes c'est un peu différent car ils ont une vie propre et il vaut mieux les laisser tranquilles…

Les scripts Python ont un nom[1] de la forme *nom_de_script*.**py**

les modules précompilés ont en plus un nom de la forme *nom_module*.**pyc**

L'installateur va te proposer un dossier principal pour Python, accepte le sans problème. Par contre tes propres scripts et autres fichiers doivent être rangés proprement par toi-même de manière à :

- pouvoir les retrouver facilement ;

- regrouper scripts et données dans le même dossier.

Par exemple tu utilises l'image *snake.gif* et le son *snake.wav* dans un script appelé *snake.py* : tu as intérêt à regrouper tous les fichiers dans un seul dossier *snake* et à travailler dans ce dossier. Quand tu auras besoin de l'image, tu vas utiliser une instruction comme

```
f = open("snake.gif", "r")
```

Python va alors chercher l'image dans le même dossier que le script. Si tu ne ranges pas toutes tes ressources au même endroit, tu seras obligé de préciser à Python l'endroit où les trouver, ce qui risque de te poser des problèmes si tu travailles sur plusieurs machines.

---

[1] Les fichiers ont un nom suivi d'une **extension** qui permet de savoir quel outil les manipule : *machin*.**gif** est certainement une image, *truc*.**txt** est un texte brut, etc. Pour voir le nom en entier sous Windows, sélectionner l'affichage avec les détails dans les dossiers (décoche l'option *masquer l'extension des fichiers dont le type est connu* dans **Options des dossiers** / **Affichage**).

1.1.6 Le package : note aux adultes

Tout le matériel nécessaire (logiciels, scripts, ressources, compléments) est disponible sur le site (**http://pythonpourenfants.free.fr**) : vous pouvez me contacter directement depuis cette adresse.

Le package fourni (uniquement Windows) contient la version 2.7 de Python avec l'éditeur IDLE, l'éditeur Pyscripter que nous utiliserons, et toutes les bibliothèques utilisées dans le livre, particulièrement les deux modules créés spécifiquement : **easyguifr.py** et **easydessin.py** (accompagnés de leurs versions compilées *easyguifr.pyc* et *easydessin.pyc*). Ces modules sont disponibles sur le site évidemment : si il y a des nouvelles versions et que vous souhaitiez faire la mise à jour, allez dans le dossier \App\Lib\lib-tk et copiez simplement les nouveaux fichiers à la place des anciens.

Pour installer le package, double-cliquez dessus et laissez vous faire… L'installation peut se faire sur n'importe quel support : disque dur interne ou externe, clé USB, carte SD… Cette installation est portable et ne nécessite pas de droits spéciaux : si par exemple vous l'installez sur une clé USB, vous pourrez alors l'utiliser sur n'importe quelle machine (sous Windows uniquement) sans manipulation supplémentaire.

Sur le site web se trouvent également diverses informations ainsi que les liens vers la documentation. Particulièrement, les documentations en fichier .py et en fichier HTML des deux modules doivent suffire et ne seront pas commentées dans ce livre. Pour tous les autres modules, ils sont fournis en standard avec Python.

Pour un affichage correct des caractères accentués dans Pyscripter, sélectionner pour les nouveaux fichiers <Edition>, <Format de Fichier>, <**UTF-16BE**>.

1.2 Mise en oeuvre

1.2.1 Utiliser Pyscripter

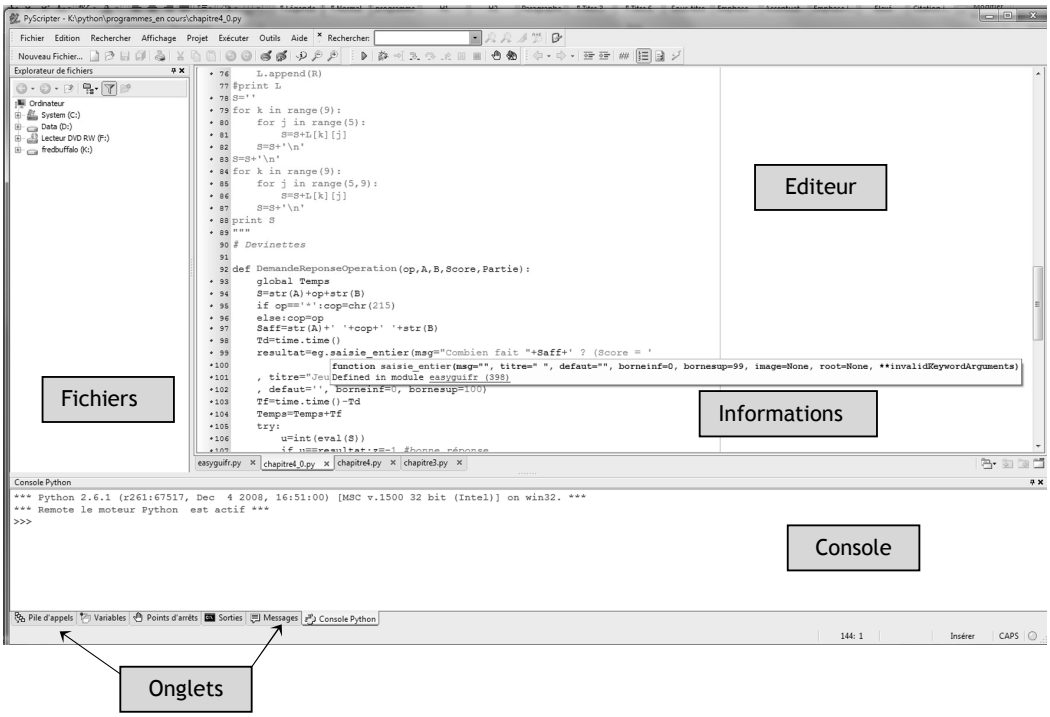
Pyscripter que tu as installé est un éditeur interpréteur débogueur complet.

Éditeur : il te permet d'écrire ton programme tout en surveillant la syntaxe et en coloriant les divers types de commande utilisées. Également, quand tu passes délicatement ta souris sur les diverses instructions que tu as écrites, diverses informations te sont fournies. Très pratique pour retrouver un module ou une documentation oubliés.

Interpréteur : tu peux faire exécuter ton script directement dans Pyscripter.

Débogueur : comme il y aura forcément des bugs, tu pourras alors lui demander de suivre pas à pas le programme tout en contrôlant le contenu des variables !

1.2.2 Éditer



L'**éditeur** est le cadre principal : les lignes de code sont numérotées pour plus de facilité de lecture. Dans le programme on peut avoir des informations sur les fonctions présentes dans un module : si le module n'est pas chargé on peut cliquer dans le bandeau pour le faire apparaître ; cette possibilité est active pendant l'écriture du code.

Diverses couleurs de texte permettent de se repérer plus facilement (les couleurs dépendent des réglages de Pyscripter, aussi c'est difficile de donner des précisions ici) : à surveiller en permanence, l'apparition d'une ligne ondulante rouge sous le texte. Il s'agit généralement d'une erreur de syntaxe (on n'a pas écrit correctement les instructions) : les plus fréquentes sont l'oubli des **deux points :** ou d'une **parenthèse**. Attention, le texte contenant l'erreur peut être AVANT la ligne rouge.

Le cadre **fichiers** permet de sélectionner les fichiers à charger : attention, normalement le fichier en cours remplace le précédent, aussi si tu as un doute sur la qualité de tes modifications, va dans les options de Pyscripter (<Outils>, <Options>, <Options de

l'IDE>) et sélectionne <Créez des fichiers de backup> afin d'avoir toujours une ancienne copie de ton travail.

La **console** (*shell*) te permet de travailler directement sans passer par les scripts : ce n'est pas le plus utile mais des fois ça dépanne. En plus c'est là que se font toutes les sorties basées sur `print`, les messages d'information ou les erreurs d'exécution.

Normalement Pyscripter tel que je te le fournis est en français : tu peux changer de langue avec <**Affichage**> <**Langage**> (ou <**View**> <**Language**> en anglais) ; de même il y a un texte qui se met automatiquement en tête d'un nouveau fichier :

```
# -*- coding: utf-8 -*-
# [nom fichier] créé par [nom utilisateur] le [date]
# Sujet :
#
#--------------------------------------------
```

Tu peux facilement modifier ce texte avec <**Fichier**> <**Nouveau**> <**Nouveau Fichier**> puis <**Gérer les modèles de fichier**>. Pour la suite tu te débrouilleras très bien…

Voilà pour l'essentiel : en fait il y a une myriade de possibilités que l'on exploite davantage quand on travaille en professionnel mais que tu peux découvrir par toi-même également !

1.2.3 Déboguer

Un programme tel qu'on l'écrit au premier jet ne marche JAMAIS !

Il faut le debugger[1], ce qui prend en général à peu près autant de temps que d'écrire le programme lui-même, voire plus si on a un bug vicelard caché dans un coin…

Pyscripter comprend tout l'attirail nécessaire au programmeur, ce qui est bien suffisant pour plusieurs années de programmation.

Un des outils les plus intéressants dans le débogage est l'**inspecteur de variables** comme on le voit sur la figure : le contenu des variables locales utilisées s'affiche automatiquement dans la fenêtre de suivi ; si aucun programme n'est lancé on peut voir (ou demander) les variables globales, ce qui permet de se rendre compte de la quantité de choses chargées au démarrage…

---

[1] Ou *déboguer* en français, le mot *bug* vient des petits insectes qui se cachaient au chaud dans les relais des premiers ordinateurs, traduit par *bogue*, enveloppe piquante de la châtaigne en français.