



Ateliers Python+Qt : Premiers pas : Découvrir le langage Python.

par X. HINAULT

www.mon-club-elec.fr



Tous droits réservés – 2013.

Document gratuit.

Ce support PDF d'atelier Python + Qt vous est offert.

Pour acheter d'autres supports d'ateliers Python + Qt rendez-vous ici :

http://www.mon-club-elec.fr/pmwiki_mon_club_elec/pmwiki.php?n=MAIN.PYQT

Vous avez constaté une erreur ? une coquille ? N'hésitez pas à nous le signaler à cette adresse : support@mon-club-elec.fr

Truc d'utilisation : visualiser ce document en mode diaporama dans le visionneur PDF. Navigation avec les flèches HAUT / BAS ou la souris.

En mode fenêtre, activer le panneau latéral vous facilitera la navigation dans le document. Bonne lecture !

PyQt : Découvrir le langage Python

Par X. HINAULT – Décembre 2012 – www.mon-club-elec.fr – Tous droits réservés



Je me propose ici de vous donner les rudiments indispensables pour pouvoir coder ensuite une interface graphique avec le langage Python : rien d'insurmontable rassurez-vous, surtout si vous connaissez déjà un langage de programmation (C, Arduino, Java...).
Vous ne serez pas un « expert en Python » à la fin de ce tuto, mais vous en saurez assez pour vous lancer !
Et si vous avez la désagréable sensation d'être perdu à la fin du tuto, rassurez-vous, ça va vite passer !!

Ce que l'on va faire ici

- Dans ce tutoriel, je vais vous présenter une introduction rapide au langage Python : à la fin de ce tuto, vous ne serez pas encore un « expert » en Python, mais vous aurez les bases pour vous débrouiller... et surtout comprendre les nombreux tutos PyQt que je vous propose sur ce site.
- Pour débiter avec Python, il est préférable de déjà connaître un peu la programmation et même idéalement maîtriser les bases d'un langage : C/C++, Java, Javascript, ou même le langage Arduino ou encore le langage Processing.
- Je me place ici dans la situation où vous connaissez déjà l'un de ces langages : ce tuto va vous présenter rapidement les correspondances entre ce que vous connaissez déjà et le langage Python : dans quelques minutes, vous serez à même d'écrire un premier code et de l'exécuter.

- Donc, pour profiter de ce tuto, il est préférable de savoir ce que sont :
 - les variables et les types (int, float, double)
 - les boucles (for, while, etc...)
 - les conditions (if, else, else if) et les opérateurs logiques
 - les fonctions et les classes (ensemble de fonctions regroupées entre-elles)
 - voire idéalement les objets, l'héritage, etc...
- Si vous n'y connaissez vraiment rien, si les notions de variables, de boucles, de conditions, de tableaux... ne vous disent absolument rien, je vous conseille d'abord d'aller faire un tour du côté des tutos PDF dédiés à l'Arduino disponibles sur ce site.
- Prêt... ? Allez, on se lance...

Pré-requis : un système Gnu/Linux opérationnel

- Je suppose ici que vous disposez d'un système Gnu/Linux graphique opérationnel avec Python installé :
 - soit un poste fixe avec une distribution Ubuntu installée et opérationnelle par exemple
 - soit une plateforme embarquée, type RaspberryPi avec une distribution Raspbian installée et opérationnelle
- Pour écrire des programmes on utilisera l'éditeur libre Geany

Le langage Python

- [Le langage Python est un langage interprété](#), c'est à dire que les instructions sont exécutées « à la volée » à partir du code : il n'est donc pas nécessaire de compiler le code avant de l'exécuter et l'exécution peut se faire directement à partir du code brut. Ceci est particulièrement intéressant pour des développements « amateurs » : on pourra écrire son code, lancer l'exécution, modifier ou corriger, relancer l'exécution, etc... à la façon Processing pour ceux qui connaissent.
- [Le langage Python est exécutable en « mode console »](#) : conséquence directe du langage interprété, les instructions pourront également être exécutées en mode « console » comme nous allons le voir. Il est ainsi très facile d'écrire ses premières lignes de codes quelques minutes seulement après l'avoir installé ! Très utile également d'utiliser la console Python à tout moment pour tester une nouvelle instruction que l'on ne connaît pas, le résultat d'une ligne de code, etc...
- [Le langage Python est un langage dit de « haut niveau »](#) : il est beaucoup plus souple qu'un langage tel que le C/C++ ou le Java pour la gestion des types de paramètres passés aux fonctions et dispose de plusieurs spécificités natives très puissantes, notamment les listes (équivalent de tableaux de taille variable) ou encore la boucle for qui voit ses possibilités étendues. Concrètement, il sera possible d'écrire en 1 ligne ce qui

prendrait plusieurs en C par exemple. On ressent à l'usage une certaine « simplicité », en tout cas, une utilisation intuitive.

- [La syntaxe de base du langage Python est simplifiée](#) : pas d'utilisation des { } ou des ; de fin de ligne : la structuration du code est basée uniquement sur l'indentation, autrement dit l'utilisation de simples tabulations. Un peu déroutant au début, à l'usage cette syntaxe s'avère facile à utiliser et donne des codes aérés et visuellement structurés.
- [L'espace de développement minimal pour commencer à coder est simple à installer](#) : en fait, un simple éditeur de texte à coloration syntaxique suffit. Il existe aussi des IDE complets, des plugins pour Eclipse, etc... mais rien de tout cela n'est nécessaire, ni même utile pour se lancer ! Ce point est très important, notamment en phase de découverte et de test. Il est possible d'aller « droit au but » et de ne pas se perdre dans des installations complexes et des procédures que l'on ne maîtrise pas... Rien à voir par exemple avec l'installation nécessaire pour écrire un premier code Java sous Eclipse...
- [Le langage Python est peu gourmand en ressources système](#), et c'est ce qui en fait toute sa force, notamment sur de petits systèmes tels que le [RaspberryPi](#) : pas de Java à installer, une exécution immédiate ou presque, notamment pour les interfaces graphiques comme celles que je vous propose dans les tutoriels dédiés à PyQt.
- [Le langage Python est polyvalent et dispose de très nombreux modules d'extension](#) et permet simplement de gérer le système, le réseau, le port série, les sons, les calculs mathématiques, astronomiques notamment. [Le portage de nombreuses bibliothèques essentielles et utiles existe en Python](#) : Qt pour la réalisation des interfaces graphiques, OpenCV pour le traitement d'image et la vision par ordinateur, OpenGL pour la 3D, etc...
- Noter que dans le cas précis d'un système Gnu/Linux, et donc notamment sur le RaspberryPi, l'utilisation du langage Python et des modules disponibles s'avère particulièrement simple, une simple ligne de commande suffisant la plupart du temps pour installer le paquet voulu.
- Enfin, [le langage Python a une réputation qui n'est plus à faire](#) : il est très utilisé dans le monde scientifique ou par des organismes réputés tels que la NASA ou encore ILM (LucasFilms...) ou encore le CEA.
- En bref, [de quelques lignes de codes à une application complète utilisant de nombreuses fonctionnalités, Python sera à la hauteur !](#)

Les deux grandes façons d'utiliser Python

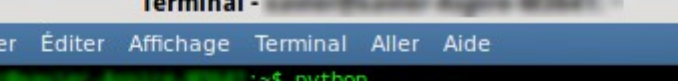
- Le langage Python est un langage interprété : de ce fait, 2 modes principaux d'utilisation sont possibles
 - soit en mode console : les instructions sont exécutées une à une après leur saisie
 - soit en mode « exécution du code » : le code est exécuté directement, soit depuis l'éditeur, soit par double clic.

Le plus simple : en mode console

- Pour lancer l'interpréteur Python en ligne de commande, saisir dans un terminal la commande :

`$python`

- Dans l'interpréteur, après l'invite se modifie et devient `>>>` : saisir alors l'instruction Python à exécuter :




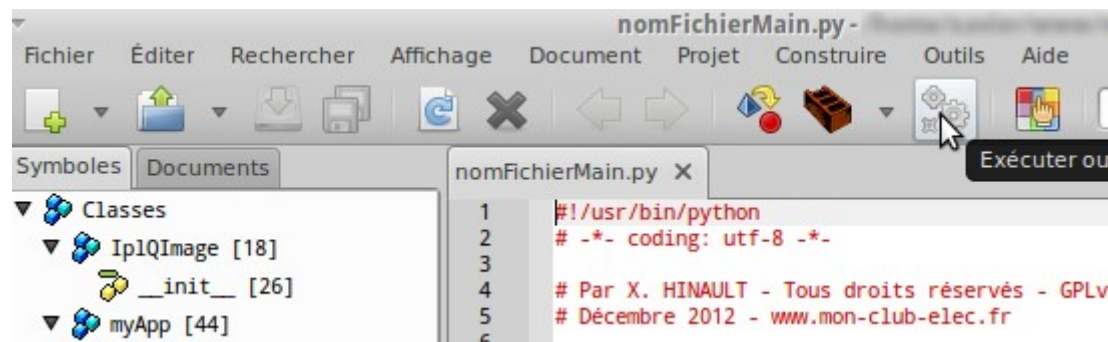
The screenshot shows a terminal window titled "Terminal - ubuntu@ubuntu: /tmp/". The menu bar includes "Fichier", "Éditer", "Affichage", "Terminal", "Aller", and "Aide". The terminal content shows the execution of the Python interpreter, which displays the version (2.7.3), the date and time (Aug 1 2012, 05:16:07), and the platform (linux2). The user enters a simple script that assigns the value 2 to the variable 'a' and prints it. The output of the script is the number 2.

```
ubuntu@ubuntu: /tmp/$ python
Python 2.7.3 (default, Aug 1 2012, 05:16:07)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> a=2
>>> print a
2
>>>
```

Cette possibilité est très très utile et pratique, en phase de découverte ou d'apprentissage et à tout moment pour tester un bout de code ou l'effet d'une instruction !

Le plus utile : exécuter le code à partir d'un éditeur à coloration syntaxique

- Si vous avez suivi le tuto précédent, vous avez déjà installé Geany. Sinon, faites-le. Une fois fait, vous êtes opérationnels !
- Trois façon de lancer l'exécution du code :
 - soit par simple **double clic** sur le fichier **nomFichierMain.py**
 - soit en **ligne de commande** avec **./nomfichierMain.py** (se placer au préalable dans le bon répertoire avec le commande **cd chemin ...**)
 - ou directement **depuis l'éditeur Geany** avec le bouton , le fichier **nomFichierMain.py** étant ouvert.



Règles de syntaxe de base du langage Python et comparatif simplifié avec les langages C et Java

- Le plus simple pour appréhender le langage Python est de comparer sa syntaxe à celle d'un langage supposé connu, ici le C ou le Java. Une fois les règles de transposition comprises, on retombe vite sur ses pieds... et on découvrira au fur et à mesure les éléments spécifiques du langage Python.

Indentation

Python	C / C++ / Arduino	Java / Processing
Les sections de code sont définies/délimitées par l'indentation (tabulations)	Les sections de code sont définies/délimitées par des { } : à toute accolade { doit correspondre une accolade }	Les sections de code sont définies/délimitées par des { } : à toute accolade { doit correspondre une accolade }
<pre>niveau1 niveau2 niveau2 niveau2 niveau3 niveau3 niveau2 niveau1 niveau2 niveau3</pre>	<pre>{ niveau1 { niveau2 niveau2 { niveau3 } } }</pre>	<pre>{ niveau1 { niveau2 niveau2 { niveau3 } } }</pre>

Commentaire

Python	C / C++ / Arduino	Java / Processing
<pre># commentaire une ligne """ commentaire multilignes """</pre>	<pre>// commentaire une ligne /* commentaire multilignes */</pre>	<pre>// commentaire une ligne /* commentaire multilignes */</pre>

Fin de ligne

Python	C / C++ / Arduino	Java / Processing
Pas de point virgule, juste le retour de chariot en fin de	Point virgule en fin de ligne	Point virgule en fin de ligne

ligne instruction1 instruction2	instruction1 ; instruction2 ;	instruction1 ; instruction2 ;
-------------------------------------------	----------------------------------	----------------------------------

Le code Python minimal

Le code python minimal exécutable est le suivant :

```
#!/usr/bin/python
# -*-coding:UTF-8 -*

#-- code des classes à mettre ici
# ...

#--- obligatoire pour rendre code exécutable ---
if __name__ == "__main__": # cette condition est vraie si le fichier est le programme exécuté
    # code exécuté au début de l'exécution ici.
    exit; # fin programme
```

où les lignes :

```
#!/usr/bin/python
# -*-coding:UTF-8 -*
```

indiquent le chemin de l'interpréteur Python à utiliser (ici, cas d'un système Gnu/Linux) et l'encodage à utiliser.

Ici le code principal est laissé vide : aucune fonction, aucune classe, aucune instruction.

La section finale est obligatoire pour rendre le code exécutable :

```
#--- obligatoire pour rendre code exécutable ---
if __name__ == "__main__": # cette condition est vraie si le fichier est le programme exécuté
    exit # fin programme
```

Lorsque le code est exécuté, ce sont les instructions à ce niveau qui sont exécutées en premier. Seule l'instruction exit n'est pas obligatoire, mais ici mise pour que cette partie ne soit pas vide (sinon, une erreur sera générée). Ce code est exécutable et ne produit aucune erreur :

```
-----
(program exited with code: 0)
Press return to continue
█
```

Utiliser les entrées / sortie console

- Une première instruction est **print()** qui affiche des messages dans la sortie console. Exemple :

```
print("coucou") # affiche message
```

- L'instruction pour récupérer une saisie à partir de la console est **input ()**. Exemple :

```
a=input("a ?")
```

Un premier code Python de base

- Le code Python de base est souvent un peu plus élaboré avec déclaration d'une fonction principale, appelée **main()** ou tout autre nom de son choix. Cette classe principale sera la première appelée au niveau du code d'initialisation, ce qui donne :

```
#!/usr/bin/python
# -*-coding:UTF-8 -*

#-- classes python à mettre ici
# ...

def main(): # fonction principale
    print("coucou") # affiche message
    # mettre le code actif ici

#--- obligatoire pour rendre code exécutable ---
if __name__ == "__main__": # cette condition est vraie si le fichier est le programme exécuté
    main() # appelle la fonction principale
    # fin programme
```

ce qui donne :


```
coucou!  
  
-----  
(program exited with code: 0)  
Press return to continue
```

Voici un résumé très simplifié des principales formes syntaxiques du langage Python, en comparaison avec le C/C++ et le Java. Le but ici est de montrer les correspondances entre le langage que vous connaissez déjà et le langage Python.

Les variables en Python

Python	C / C++ / Arduino	Java / Processing
<pre>a=2 # un int b=2.0 # un float</pre>	<pre>int a=2 ; // un int float b=2.0 ; // un float</pre>	<pre>int a=2 ; // un int float b=2.0 ; // un float</pre>

Les chaînes de caractères en Python

Python	C / C++ / Arduino	Java / Processing
<pre>myStr= "coucou"</pre>	<pre>String myStr="coucou" ;</pre>	<pre>String myStr="coucou" ;</pre>

Les fonctions en Python

Python	C / C++ / Arduino	Java / Processing
<pre>def nomFonction(self, param1, param2) : code </pre>	<pre>typeRenvoi nomFonction(type1 param1, type2 param2) { code }</pre>	<pre>typeRenvoi nomFonction(type1 param1, type2 param2) { code }</pre>

Les classes en Python

Définition d'une classe

Python	C / C++ / Arduino	Java / Processing
--------	-------------------	-------------------

<pre> class Person(object): # création de la classe def __init__(self,count): # constructeur code def fonctionClasse (self,param1,param2): code # fin de la classe </pre>		<pre> public class Chain { // création de la classe // variables de classe private Person first = null; public Chain(type param) // constructeur { code } // fin constructeur public fonctionClasse (type param) { code } // fin fonction classe } // fin classe </pre>
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Déclaration d'instances de classe

Python	C / C++ / Arduino	Java / Processing
<pre> nomInstance=ClasseInstance(params) </pre>	<pre> ClasseInstance nomInstance (params); </pre>	<pre> ClasseInstance nomInstance ; nomInstance=new ClasseInstance(params) ; </pre>

Les conditions en Python

Python	C / C++ / Arduino	Java / Processing
<pre> if condition : code else : code </pre>	<pre> if (condition) { code } else { code } </pre>	<pre> if (condition) { code } else { code } </pre>

	}	}
--	---	---

Les boucles en Python

Python	C / C++ / Arduino	Java / Processing
<pre>for counter in range(1, 6): code</pre>	<pre>for (int i = 0; i < 100; i++) { code }</pre>	<pre>for (int i = 0; i < 100; i++) { code }</pre>

Tableaux et listes en Python

Python	C / C++ / Arduino	Java / Processing
<pre>xx = array([val1, val2, val3])</pre> <p>Une grande force de Python : multiplier (ou toute autre opération) 2 tableaux entraîne la multiplication terme à terme des 2 tableaux... en 1 ligne !</p> <p>Plus pratique que le tableau, utiliser une list (tableau de taille variable) :</p> <pre>a = [5, 0, 3, 9, 4, 3, 7] a.append(15) ## a = [5, 0, 3, 9, 4, 3, 7, 15] a.index(3) # =>2</pre>	<pre>int array[10];</pre>	<pre>int[] anArray; // déclare tableau anArray = new int[10]; // initialise tableau</pre>

Utilisation des modules en Python

Python	C / C++ / Arduino	Java / Processing
<pre>import module # importe toutes les classes du module accessible sous la forme module.classe</pre> <pre>from module import * # importe toutes les classes du module - accessible sous la forme directe classe</pre> <pre>from module import a,b,c # importe les classes a,b,c du module - accessible sous la forme directe classe</pre>	<pre>#include <module> ;</pre>	<pre>import module ;</pre>

Un code Python structuré type

- Voici à présent la structure globale type d'un programme Python structuré à minima :

```
#!/usr/bin/python
# -*-coding:UTF-8 -*

# modules à importer
# import module

# classe(s) du programme
class myApp:

    #-- fonctions de classe
    def ma_fonction(self,a,b):
        return(a*b)

# fonctions globales
def ma_fonction(a,b):
    return(a*b)

# classe principale
class main():
    print("prog mini Ok")
    # appel fonction globale
    a=input("a ?")
    b=input("b ? ")
    c=ma_fonction(a,b) # appelle la fonction globale
    print=",c"

    # appel fonction de classe
    classe=myApp() # déclare instance classe
    a=input("a ?")
    b=input("b ? ")
    d=classe.ma_fonction(a,b) # appelle la fonction de la classe
    print=",d"

#--- rendre code exécutable ---
if __name__ == "__main__": # cette condition est vraie si le fichier est le programme exécuté
    main() # appel la classe main()
```

Conclusion

- Le but de ce tuto est de vous donner les bases minimales pour vous lancer en Python en peu de temps, en présupposant que vous connaissez

déjà un langage : vous avez à ce stade l'essentiel à connaître pour écrire vos premiers codes et comprendre les tutos PyQt que je vous propose.

A ce stade, vous disposez des bases indispensables pour aborder la suite, et notamment le codage d'applications graphiques avec Python et Qt : vous êtes en mesure de suivre les tutos que je vous propose ici.
Vous pourrez approfondir le Python au fur et à mesure...

- Rien ne remplacera cependant un apprentissage systématique du langage Python en profondeur, ce qui peut se faire en étudiant une première fois (sans s'attarder sur les détails) l'ensemble d'un tutoriel d'introduction au langage Python et en approfondissant pas à pas en fonction des besoins.
- Comme toujours lorsque l'on apprend un nouveau langage, il y a un temps de rodage et de découverte... et une fois passée cette première étape, on enrichit peu à peu ses compétences, d'autant plus rapidement que l'on retrouve ce que l'on connaît déjà si l'on est habitué à coder.
- A présent, voyons comment mettre en place une interface graphique avec Qt Designer. **Si vous vous sentez un peu perdu, ne vous en faites pas : ça va passer ! Poursuivez la lecture des tutos de la rubrique « Premiers Pas » et vous verrez que le jeu en vaut largement la chandelle !**

Ressources utiles :

- Le site officiel Python : <http://www.python.org/>
- Référence du langage Python : <http://docs.python.org/2/reference/>
- Tutoriel site du zéro sur le langage Python (très bien fait) : <http://www.siteduzero.com/informatique/tutoriels/apprenez-a-programmer-en-python>
- La page Python du site developpez.com : <http://python.developpez.com/cours/?page=DocGeneral#Debutant>