

Les bases de

PYTHON

Ce tutoriel est en cours de rédaction et vous apprendra les bases du langage python en quelques parties (**j'avais à la base rédigé ce tuto pour le SiteduZero, mais ne l'ai jamais fais validé, et il n'est pas encore fini**)

Le Python, ce langage à la fois utile et plutôt simple, que vous allez apprendre ici, pour ensuite aller plus loin dans d'autres tutoriaux sur ce langage.

extrait Wikipedia :

Python est un langage de programmation interprété multi-paradigme. Il favorise la programmation impérative structurée, et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions ; il est ainsi similaire à Perl, Ruby, Scheme, Smalltalk et Tcl.

Le langage Python est placé sous une licence libre proche de la licence BSD1 et fonctionne sur la plupart des plates-formes informatiques, des supercalculateurs aux ordinateurs centraux, de Windows à Unix en passant par Linux et MacOS, avec Java ou encore .NET. Il est conçu pour optimiser la productivité des programmeurs en offrant des outils de haut-niveau et une syntaxe simple à utiliser. Il est également apprécié par les pédagogues qui y trouvent un langage où la syntaxe clairement séparée des mécanismes de bas niveau, permet une initiation plus aisée aux concepts de base de la programmation.

Pour résumer , un langage qui à une grande utilité, utilisable , autant en orienté objet que en Web (eh oui), et qui est aussi un langage à haut niveau, se rapprochant beaucoup de la langue humaine (anglophone) comme beaucoup de langages. Autrement dit il a un très bon rapport "simplicité/utilité" et est très bien pour débiter .

Ce tuto n'est qu'une initiation au Python, il n'est pas complet mais vous aurez de bonnes bases pour continuer dans ce langage . Une fois ce tuto terminé je vous suggèrerais d'étendre vos connaissances en Python en lisant le tuto ci dessous

<http://python.developpez.com/cours/TutoSwinnen/>

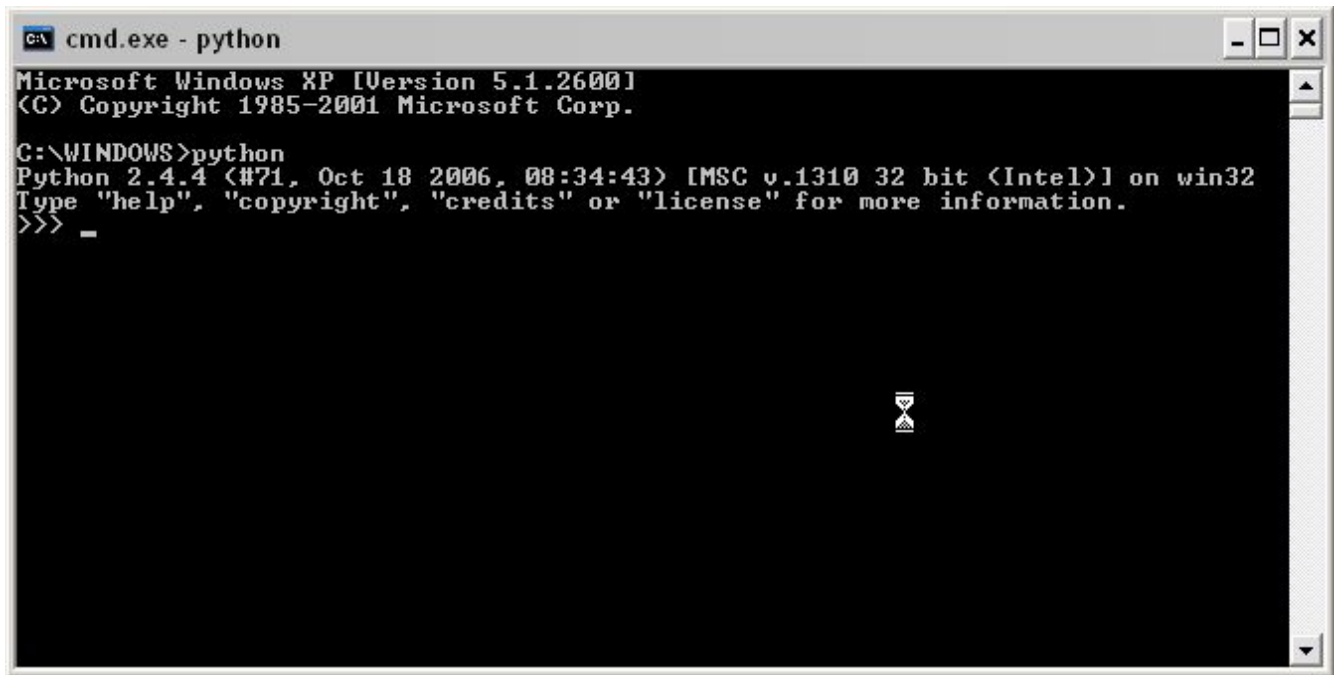
sur ce je vous souhaite un bon cours. Cordialement , LaSourisVerte

[1/ La console \(afficher du texte \)](#)

Le texte

La console , un outil très utile je dirais même indispensable, mais aussi utile pour apprendre le python en simplicité et créer ses premiers programmes simplement

aperçu de la console python :



```
cmd.exe - python
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS>python
Python 2.4.4 (#71, Oct 18 2006, 08:34:43) [MSC v.1310 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```

elle a à peu près la même apparence sous Windows Linux et MacOS, vous pouvez y tester des lignes de code ou y exécuter vos programmes

Premières *lignes de code*

Ouvrez le console Python et tapez :

```
print "123"
```

Cela affiche :

```
>>> 123
```

Vous l'aurez compris, print sert à afficher du texte

Essayez plusieurs fois, vous aurez remarqué que avec l'encodage certaines lettres notamment celles avec accents ne s'affichent pas et sont remplacées .

Sauter des lignes(texte)

Pour sauter des lignes , il suffit de rentrer \n

par exemple rentrez :

```
print "123\n456"
```

cela affichera:

```
>>>123
>>>456
```

entraenez vous plusieurs fois , bien évidemment vous pouvez aussi le faire comme cela :

```
print "123"
print "456"
```

mais cela est faisable seulement en script car on ne peut exécuter qu'une commande à la fois en console.

Les guillemets ?

Et oui les guillemets, comment les afficher dans un texte ?

il suffit de mettre un antislash \ juste avant un guillemet

par exemple :

```
print "en dessous\ns\'affiche un texte\n \"entre guillemets\""
```

comme vous l'aurez remarqué il faut également en mettre avant une apostrophe car oui, les apostrophes peuvent remplacer les guillemets en python

on peut écrire

```
print "123"
```

#ou bien

```
print '123'
```

Les commentaires

je vais faire court , les commentaire sont des textes qui ne seront pas visibles dans le programme mais uniquement dans son code source pour se repérer il suffit de le mettre à coté d'une ligne dans votre code source précédé d'un #

exemple:

```
#!/usr/bin/python
# -*- coding: iso-8859-1 -*-
print "phrase" # on affiche le mot phrase
raw_input()
```

vous verrez que ça vous servira beaucoup

[2/ Variables / calculs basiques](#)

Oui les variables sont un outil indispensable dans presque tous les langages de programmation, du moins dans tous les langage de programmation impérative structurée .

nous allons voir ici comment nous en servir en python .

Calculs de base

En Python , vous aurez surement remarqué que l'on peut utiliser print de cette façon

```
print 1
```

au lieu de

```
print "1"
```

Mais alors pourquoi on ne peut pas faire => `print un_mot` ?

print marche sans guillemets car on parle d'une valeur , le chiffre est une valeur, on peut très bien taper 1+2 dans la console cela affichera

```
>>>1+2  
>>>3
```

à titre d'information :

+ Addition
- Soustraction
* Multiplication
/ Division
% modulo

Variables

Les variables sa varie ^^ ok je vous avance pas à grand chose là.

En clair si vous avez déjà appris un langage de programmation vous devez déjà connaître les variables.

une variables c'est le fait de donner une valeur à un caractère ou a une chaine de caractères

Exemple:

```
x = 25  
y = 17
```

si vous tapez ces deux dernières lignes dans la console et que vous tapez ensuite

```
print x + y  
>>>42
```

Vous pouvez donc utiliser beaucoup de caractères dans les calculs , parenthèses comprises .

Différents types de variables

Oui toute variable a un type.

Il y a les variables Str, int, ou float

Mais à quoi correspondent ces abréviations ?

Les noms ne sont pas importants à retenir mais souvenez vous quand même que :

Str = String = Chaîne de caractères

Int = Integer = Nombre entier

Float = Float = Nombre à virgule

Attribuer une valeur quelconque à une variable

Je vous ai déjà montré comment faire des calculs avec des variables .

Nous allons maintenant voir comment faire contenir du texte à ces variables, ou encore des nombres précis.

1/ Integer

Nous l'avons déjà vu il suffit de faire :

```
>>>x = 2
>>>y = 3
print x*y
6
```

2/ Float

Les Float ce sont les nombres à virgule, oui car si vous faites dans python par exemple:

```
>>>15/2
```

il vous affichera un approximatif entier, en arrondissant à 7
mais n'affichera pas le reste.

les float sont donc les nombres à virgule.

pour obtenir 7.5 à ce calcul il faut donc faire

```
15./2
```

ou

```
15/2.
```

ou encore

```
15./2.
```

Pour simplifier on va dire que sa sert à dire que l'on veut un résultat précis au moins au dixième .

3/ String

String, c'est une chaîne de caractères,
autrement dit cela sert à donner une valeur en texte à une variable .

Pour ceux qui auront déjà appris des langages où le type de variable se déclare , comme en RapidQ ou plusieurs dérivés basic par exemple, pour déclarer le type d'une variable String on devrait écrire.

Code : VB.NET

```
DIM nom_de_variable AS STRING
```

En Python on ne s'encombre pas de ça. il suffit de taper le nom de la variable suivi d'un symbole égale "=" lui même suivi de la valeur.

Pour une chaîne de caractères
il faudrait bien mettre

```
x = "texte"
```

et non pas

```
x = text
```

Autrement Python va croire que la variable **x** est égale à la variable **text**, or il n'existe aucune variable appelée text , il y aura donc une erreur.

Cela nous en fait venir à une sorte de "report" de variable, par exemple:

```
x = "LaSourisVerte"  
y = x  
  
print y
```

Cela affichera bien **LaSourisVerte**, *car on ordonne d'afficher y qui est lui même égal à x .*

Afficher des variables

Afficher des variables à la suite, comment faire ?

eh bien essayez cela en console

```
a = 2
b = 5

print a b
```

eh oui cela n'affiche rien d'autre qu'un erreur, il faut séparer les variables pour en afficher plusieurs à la suite .

le séparateur en Python est la virgule " , "

Donc pour afficher des variables à la suite, on met une virgule à la fin de la variable ou du texte.

par exemple :

```
a = "LaSourisVerte"
print "mon nom est : "a
```

cela ne marchera pas, il faut taper :

```
a = "LaSourisVerte"
print "mon nom est : ",a
```

!NE JAMAIS OUBLIER LA VIRGULE

On peut évidemment calculer les valeurs entre elles .

exemple :

```
x = 12
y = 25
z = 20
print (x*y)/z
```

Mais pour les afficher on fait seulement :

```
x = "Je suis"
y = "un"
z = "zer0"
print x,y,z
```

ATTENTION AUX TYPES

En effet , si l'on fait:

```
x = "Je suis"
y = "un"
z = "zer0"
print (x*y)/z
```

cela est **IMPOSSIBLE**, les variables contenant des valeurs **Str** il est impossible de multiplier du texte par du texte .

3/ Les conditions

Conditions ? what this ?

Les condition

Vous savez ce que c'est dans la vie de tous les jours, sa correspond à "si alors ..."

mais il peut également arriver d'avoir affaire à des conditions comme "si ... alors ... sinon si ... alors ... sinon ..."

et bien en programmation c'est exactement la même chose

! Les conditions servent à faire agir le programme en fonction des actions précédentes.

en anglais les mots correspondants à si, sinon, et sinon si sont if, else, et else if, dans le langage python on l'écrira elif donc les trois conditions que l'on va voir seront if, else, et elif.

Nous allons les voir unes par unes.

bonne lecture à vous .

Vos premiers scripts

A partir de maintenant fini la console, sa vous lasse d'utiliser la console ? tant mieux , on va commencer à créer nos premiers programmes.

Pour cela il vous suffit de créer un nouveau fichier sur le bureau, et d'y mettre l'extension **.py** exemple **programme.py**

pour débiter ce programme les deux premières lignes seront toujours les mêmes, regardez dans les annexes du tuto pour l'encodage si vous voulez en savoir plus, mais les deux premières lignes de vos programmes pour ce tuto seront toujours :

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

Éditez votre code avec notepad++ ou l'éditeur IDLE fourni avec Python, ou encore un autre éditeur de texte simple peu importe.

tapez les deux lignes précédentes :

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

et ajoutez y :

```
print "quelque chose"
```

pour faire :

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

print "...."
```

Enregistrez votre code, puis fermez l'éditeur de texte .

Les windowsiens, double-cliquez seulement sur votre fichier sur votre bureau il s'exécutera, les linuxiens ouvrez la console , faite le chemin jusqu'à votre bureau, et tapez :

```
python nom_de_votre_programme.py
```

Une fenêtre s'ouvre et se referme de suite hein ? :p

Normal , l'interpréteur à interprété le script et l'a fermé une fois cela fait 😊

Mais comment faire alors ? 🤔

c'est la qu'interviennent `raw_input()` et `input()`

Explications :

Ces deux commandes servent à faire interagir l'utilisateur en le faisant rentrer du texte ou des valeurs.

pas dur à retenir, `input()` sert à entrer des valeurs en chiffre
`raw_input()` des Str, autrement dit des phrases ou mots .

Pour que votre code ne se ferme pas directement il suffit d'ajouter `input()` ou `raw_input()` à la fin du code , car le programme attendra que vous ayez validé avec entré pour quitter le programme , en gros faites :

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

print "...."
input()
```

magique sa marche, vous avez créer votre premier programme .Par la suite nous allons apprendre à creer votre premier programme faisant interagir l'utilisateur .

If, else... les conditions de base

Le **IF**, la condition signifiant "si"

nous allons commencer par la, étant la plus utilisée, nous allons voir qu'elle n'en est pas moins simple .

utilisons if

Nous avons vu plus haut `input()` et `raw_input()`

et bien nous pouvons faire du contenu de ces champs une valeur, exemple :

```
variable = input()
```

ou bien :

```
variable = raw_input()
```

En partant du fait que je vais ici vous enseigner l'utilisation des conditions en Python, **A NE SURTOUT PAS OUBLIER** ==> une condition se finit toujours par deux points : avant de donner l'instruction, exemple :

```
if variable == 1:
```

Bon notre premier code :

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

print "rentrez votre age : (chiffres uniquement)"
age = input()
print "vous avez: ",age," ans"
input()
```

nous n'auront pas utilisé de condition dans ce dernier code mais c'est pour vous rappeler comment placer une variable au milieu d'un texte.

maintenant nous allons fixer une limite d'age et c'est ici que nous allons utiliser pour la première fois **else**, qui signifie "sinon".

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

print "rentrez votre age : (chiffres uniquement) "
age = input()
if age >= 18:
    print"Bienvenue"
else:
    print"Dehors !!!"
raw_input()
```

Oula !!! on à raté plein de trucs là, des choses qu'on ne connaît pas

Bon reprenons ligne par ligne en commençant à la ligne 3 😊

```
print "rentrez votre age : (chiffres uniquement) "
```

Cela sert, vous l'aurez compris à afficher le texte: rentrez votre age : (chiffres uniquement).

```
age = input()
```

Nous l'avons vu plus haut, pour simplifier, on donne à la variable age, le contenu de ce que vous aurez tapé dans le champ texte input() qui ne peut contenir que des valeurs en chiffre .

```
if age >= 18:
    print"Bienvenue"
```


Nous y voilà, notre première condition, qui dit que :
si age est SUPERIEUR OU EGAL à 18, on affiche le message bienvenue,
>= est un comparateur, pour faire simple il existe plusieurs comparateurs:

== Absolument égal
<= Plus petit ou égal
>= Plus grand ou égal
< Plus petit
> Plus grand
!= Différent

à partir de là vous avez à peu près compris comment fonctionne une condition, passons à la fin du code.

```
else:  
    print"Dehors jeune homme/fille."  
raw_input()
```

qui signifie:
sinon afficher le texte "Dehors jeune homme/fille"

pour résumer , si l'age est supérieur ou = à 18 ans , on affiche "bienvenue", sinon on affiche "accès interdit dehors"

Un espace en dessous de la condition ?

oui sa s'appelle l'indentation, ici j'ai mis des espaces mais je vous suggère de mettre un tab(touche au dessus de Maj), ceci est indispensable, **la plupart de mes erreurs de code personnelles viennent d'une mauvaise indentation. C'est une erreur très courante, Ajoutons aussi les 2 points à la fin de la condition qui ne sont pas à oublier .**

Nous avons maintenant vu Les conditions de base, on va maintenant y ajouter la condition Elif (sinon si) et aussi vois qu'une condition peux en contenir d'autres .

elif ... la condition magique

Dans certains cas, on veut choisir plusieurs options par les conditions, c'est là que elif intervient .

vous aurez déjà compris comment fonctionnent les conditions if et else avant, et bien elif est exactement pareil . Nous allons également voir qu'une condition peut en renfermer d'autres .

Exemple :

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

print "voulez vous rentrer un chiffre (o/n) :" #demande si l'on veut
rentrer un chiffre
choix = raw_input()
if choix == "o": #si le choix est positif
print "veuillez entrer un chiffre entre 1 et 5:" #on demande de
rentrer un chiffre entre 1 et 5
    chiffre = input()
    if chiffre == 1:
        print "votre chiffre est ",chiffre
    elif chiffre == 2:
        print "votre chiffre est ",chiffre
    elif chiffre == 3:
        print "votre chiffre est ",chiffre
    elif chiffre == 4:
        print "votre chiffre est ",chiffre
    elif chiffre == 5:
        print "votre chiffre est ",chiffre
    else: #si le chiffre n'est pas compris entre 1 et 5
        print "votre chiffre n'est pas compris entre 1 et 5" # on
l'affiche
else:
    print "tres bien , n'entrez pas de chiffre et appuyez sur entrer
pour continuer "
raw_input()
```

en gros on vous demande si vous voulez rentrer un chiffre, si vous mettez choisissez oui, il vous demandera de choisir un chiffre entre 1 et 5, puis vous affichera votre chiffre, si votre chiffre n'est pas compris entre 1 et 5 il vous le dira, sinon si vous ne voulez pas rentrer de chiffre il vous dira d'appuyer sur entrer pour fermer la fenêtre .

voilà je pense que vous avez les bases pour bien utiliser ces 3 conditions .

Nous l'aurons donc vu, les conditions sont indispensables dans la quasi totalité des programmes, pour ne pas dire la totalité.

retenez bien et n'hésitez pas à revenir y voir régulièrement si vous oubliez des choses

4/ Les boucles

Pour faire simple, je ne vais vous expliquer ici que le principe de la boucle **while**

en programmation, il faut considérer le **"while"**, comme un **" Tant que "**

Exemple :

```
variable = 1
while variable <= 10:
    print i
    i = i + 1
raw_input()
```

pour faire simple, on déclare une variable , dans le cas présent on l'appelle **"variable"** , on lui attribue la valeur **"1"** , et tant que **"Variable <= 1"**, eh bien on ajoute **"1"** à la variable, et on affiche la valeur de la variable .

assez simple, cela peut aussi être utilisé afin de ne pas fermer un programme à la fin de son utilisation, afin de revenir au début de celui ci :

```
var = 0
while var != 1:
    print " .... " # j'ecris mon programme ici
    choix = raw_input("voulez vous recommencer ? ( o/n ) : ")
    if choix == "n":
        var = 1
    else:
        "vous allez etre renvoye au debut du programme"
```

voilà j'ai fais court, mais vous aurez compris comment cela fonctionne 😊

IMPORTANT : tout comme pour les conditions , ne pas oublier l'indentation dans les instructions de la boucle

```
while i > 5:
instructions
.....
.....
```

Suite du cours bientôt ;)

cordialement , LaSourisVerte.