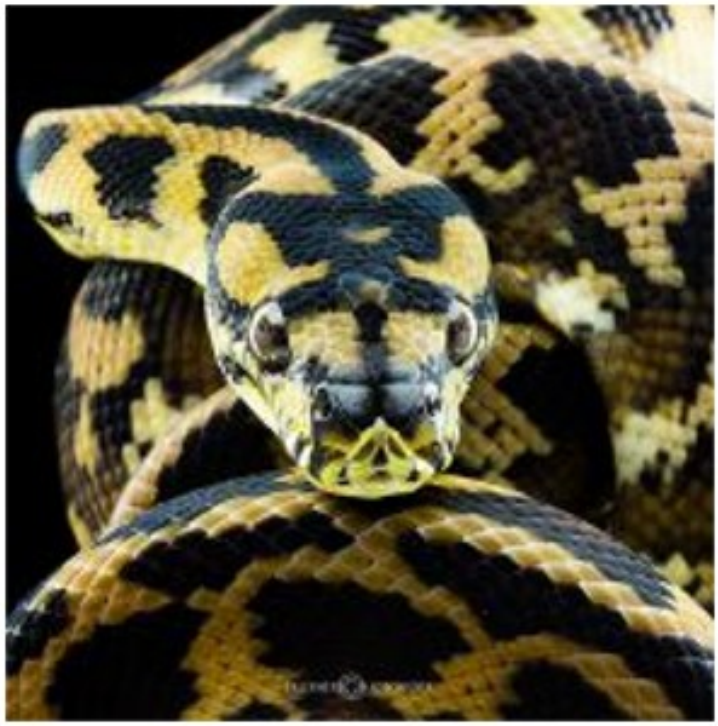
Technicien Supérieur Gestionnaire
Exploitant de Ressources Informatiques
et Réseaux

Decouverte



Python

Beginner's guide

O RLY?

L.Marchal

Table des matières

Python

Version 3.4

Presentation

Difference entre la version 2 et 3, la fonction print s'utilise seule dans python 2

Prompt (invite de commande)

- Au lancement python nous indique quelle version est utilisée ainsi que la version de l'interpréteur GCC.

```
Python 2.7.9 (default, Mar  1 2015, 12:57:24)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- L'utilisateur Microsoft devra télécharger un interpréteur adapté à son environnement.
- Sous Linux, nous préférons personnellement travailler dans une simple fenêtre de terminal pour lancer l'interpréteur Python ou l'exécution des scripts, et faire appel à un éditeur de texte ordinaire tel que Gedit, Kate, ou un peu plus spécialisé comme Geany pour l'édition de ces derniers.(Celui que j'utilise)
- Dès lors python est prêt à exécuter les commandes, et agir comme une calculatrice.

```
>>> 5+3
8
>>> 2-9
-7
>>> 7+3*4
19
>>> (7+3)*4
40
>>> 20:3
  File "<stdin>", line 1
    20:3                                    |Attention se n'est pas le signe de ladivision
      ^                                     |Python nous indique clairement l'erreur
SyntaxError: invalid syntax
>>> 20/3
6
>>> 20//3
6
>>> 20.0/3
6.666666666666667
>>> 20%3
2
```

Python respecte les conventions mathématiques concernant l'ordre des priorités des opérateurs.

Division

Attention les divisions donneront des « entiers », afin d'obtenir une

valeur décimale il faut l'indiquer dans le `calcul`.

Synthaxes

Dans python les espaces sont optionnels ; nous n'avons pas à nous soucier des espaces à entrer.

Python est sensible à la casse.

Nom de variable

- Une variable commence toujours par une lettre.
- Le nom d'une variable est constitué de lettre <a-z>, <A-Z>,et de chiffre <1-9>.
- Les caractères accentués et les caractères spéciaux sont a bannir du code à l'exception du caractère « souligné » _

Composition

A ce sujet il y a consensus ; d'ordinaire les variables sont écrites en minuscule. Il est possible d'intégrer des majuscules à l'intérieur du nom afin de marquer son « importance » dans le code.

Certaines variables sont écrites entièrement en majuscule afin d'indiquer de ne pas modifier la variable.

Mots réservés

Ils sont au nombre de 33, ils sont déjà utilisés à d'autres fins, se sont les « directives » de programmation.:

and	as	assert	break	class	continue
def	del	elif	else	except	False
finally	for	from	global	if	import
in	is	lambda	None	nonlocal	not
or	pass	raise	return	True	try
while	with	yield			

Elles serviront à l'intérieur des programmes, on peut déjà deviner le comportement...if, and, or, for (conditions).

Variables

Python permet d'assigner une valeur à une variable.

Utiliser le signe = pour cela ; par exemple :

```
>>> n=7
>>> n
7
>>> pi=3.14159
>>> msg="Quoi de neuf?"
>>> n
7
>>> pi
3.14159
>>> msg
'Quoi de neuf?'
```

La valeur 7 est assigné à n
La valeur 3,14159 à pi
Et la valeur quoi de neuf a msg.

Ainsi la valeur est stockée dans la variable il suffit alors de taper la variable pour que la valeur apparaisse.

Typage des variables

Nous avons créé précédemment 3 variables, qui sont de nature ; de types différents :

- 7 étant un nombre entier.{integer}
- 3,14 159 étant un nombre réel ou {float}en anglais.
- 'Quoi de neuf' étant une chaîne de caractère.{string}

Python sait définir automatiquement le type de la variable « au mieux » en fonction de la valeur indiquée.

C'est ce que l'on nomme le « typage dynamique » par opposition au « typage static » valable dans d'autres langages comme C, C++...

Le typages dynamique est très pratique et ergonomique, il permet de travailler sur des projets de développement complexe, et d'utiliser des bases de données, des listes ou des dictionnaires..

J'entrevois la boucle {for;to;in} me permettant de changer les mots de passe d'une liste d'utilisateurs dans l'active directory.

Pratique

Nous allons procéder à quelques exercices mathématiques, afin de comprendre l'organisation et l'ordre d'un script.

Surface.py

Ici un programme python pour calculer la surface d'un cercle.

On note que l'on peut assigner les valeurs aux variables de la manière suivante :

En séparant les variables par des virgules, ainsi que les valeurs.

```
r, pi = 12, 3.14159
s=pi*r**2
print(s)
print(type(r), type(pi),type(s))
```

La dernière ligne demande d'indiquer le « type » des variables.

```
452.38896
<class 'int'> <class 'float'> <class 'float'>
```

Virgule flottante

Pi que l'on a défini à 3,14159 est un nombre à virgule flottante {float}en Anglais.

Attention lors des calculs si python détecte une variable de type entier alors le résultat sera de type entier.

Afin d'obtenir une réponse plus cohérente il suffit d'ajouter une virgule à notre valeur afin qu'elle soit détectée comme {float}

```
>~ python
Python 2.7.9 (default, Mar  1 2015, 12:57:24)
[GCC 4.9.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 100%3
1
>>> 100/3
33
>>> 100//33
3
>>> 100//3
33
>>> 100.0/3
33.333333333333336
>>>
```

Toutefois ceci n'est pas le meilleur exemple car il y a une différence plus fine dans le résultat obtenu avec l'opérateur / ou //

Arrondi

Dans l'exemple suivant nous obtenons 2 résultats pour un calcul avec l'un et l'autre des opérateurs.

```
>>> a=3
>>> b=5
>>> c=7
>>> a-b//c
3
>>> a-b/c
2.2857142857142856
```

Explication

L'opérateur // arrondi à la valeur supérieure , ce qui peut répondre à certaines attentes.

L'opérateur / effectue une véritable division ce qui répond à d'autre attente.

Affectations multiples

Python permet également de définir la même valeur à différentes variables de manière simultanée :

```
>>> x=y=7
>>> x
7
>>> y
7
```

Ou de manière dite parallèles :

```
>>> a=b=7.7586234
>>> a
7.7586234
>>> b
7.7586234
```

Attention :

Dans python le séparateur décimal est un point, alors que la virgule (,) sert à la séparation des valeurs ou des variables.

Modulo

Le modulo qui s'écrit avec % permet d'obtenir le reste d'une division, ce qui permet de savoir si on nombre est divisible par un autre en donnant un résultat entier et non réelle.

```
>>> 10%3
1
>>> 10%5
0
```

```
>>> 10%4
2
>>> 10%2
0
```

Voilà... l'idée du modulo est : ça tombe juste... ou pas.

Rappel des opérateurs

Petit système mnémotechnique pour se souvenir de l'ordre des priorités :

PEMDAS

P pour les parenthèses	()
E pour les exposants	10**2
M&D pour multiplication et division	* et /
A&S pour addition et soustraction	+ et -

Instructions

Nous savons assigner des valeurs simultanément à différentes variable :

```
>>> h,m,s=15,27,34
```

Imprimer un message avec print() :

```
print(« blablabla = »)
```

Construire un calcul avec les opérateurs :

```
h*3600+m*60+s
```

```
Python 3.4.2 (default, Oct  8 2014, 10:45:20)
[GCC 4.9.1] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> h,m,s=15,27,34
>>> print("nombre de secondes écoulées depuis minuit=",h*3600+m*60+s)
nombre de secondes écoulées depuis minuit= 55654
>>>
```

Attention :

Il y a des limites à ce que l'on peut faire.

Tous les caractères à gauche du signe = doivent être des variables.

Dans le cas présent le signe = n'est pas le signe d 'égalité des mathématiques, il a une fonction précise , c'est le symbole d'assignation.

Interactivité

Python peut réclamer des valeurs à l'utilisateur du programme, dont voici le code :

```
a=int(input("A="))
b=int(input("B="))
c=a*b
print("valeur de c:",c)
```

Le programme réclame avec (input(« A= »)) des valeurs « entier » indiqué par le int en début de ligne : int =integer..

Un message sera envoyé à l'utilisateur afin qu'il saisisse la dite valeur avec « A= »

Pour exécuter un programme python il faut se rendre dans le répertoire contenant le programme :

```
>~ python3 box.py
A=5
B=6
valeur de c: 30
```

Affectation parallèle

L'affectation parallèle permet d'imiter le comportement de la commande SWAP issu du langage *basic* .

Instruction

While

While permet la répétition d'une instruction ce que l'on nomme une boucle.Il existe également for ... in ..., que j'ai bien l'intention d'utiliser pour l'AD.

Dans le cas présent nous :

- assignons la valeur 0 à a
- indiquons avec while de répéter les instructions « tant que » a est inférieur à 7
- le double point : indique qu'il s'agit d'une instruction composée
- incrémentons +1 sur a
- demandons une sortie

```
Python 3.4.2 (default, Oct  8 2014, 10:45:20)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> a=0
>>> while(a<7):
        a=a+1
        print(a)


1
2
3
4
5
6
7
>>>
```

Attention

- Si la condition est fausse alors le corps de la boucle n'est jamais executé.
- Si la condition reste toujours vraie la boucle se répète indéfiniment .

  Par exemple ne pas exécuter ceci :

```
>>> n = 3
>>> while n < 5:
...
print("hello !")
```

Dans le cas présent nous voyions que rien ne viendra changer la valeur de n ; qui donc restera toujours inférieur à 5 donc la boucle ne s'arrêtera pas.

La notion est très simple.

Création de tableau

```
a=0
while a<12:
        a=a+1
        print(a,a**2,a**3)
```

- La variable a vaut zero
- Instruction répété tant que a vaut moins de 12
- On incrémente a
- Demandons une liste des valeurs de a, de ses carrés, ainsi que ses cubes.

```
Python 3.4.2 (default, Oct  8 2014, 10:45:20)
[GCC 4.9.1] on linux
Type "copyright", "credits" or "license()" for more information.
>>> ========== RESTART ==========
>>>
1 1 1
2 4 8
3 9 27
4 16 64
5 25 125
6 36 216
7 49 343
8 64 512
9 81 729
10 100 1000
11 121 1331
12 144 1728
>>>
```

Si a=12 alors la boucle s'arrête.

Print() permet de retourner les valeurs obtenu sur une seule ligne.

Suite de fibonacci

La suite de Fibonacci (vu dans le DaVinci code, c'est une des énigmes) est simplement la suite de nombre s'additionnant avec le précédent résultat...

```
a,b,c=1,1,1
while c<11:
    print (b, end =" ")
    a,b,c=  b, a+b, c+1
```

- Dans le cas présent il y a une affectation parallèle pour les variables a, b et c
- une boucle while : tant que c reste inférieur à 11 (donc 10), et indiquons qu'il y a des instructions composées avec les 2 points :
- Demandons un retour : print ()
  Nota : print() introduit un saut de ligne comme vu précédemment.
- Nous avions des lignes de 3 valeurs puis un retour à la ligne ce qui nous forme une sorte de tableau (une suite de lignes et de colonnes forment un tableau)
- end = " " indique que nous ne souhaitons pas de saut de ligne, mais un espace.

```
1 2 3 5 8 13 21 34 55 89
```

- En enlevant end = " "

```
1
2
3
5
8
13
21
34
55
89
```

Revenons sur la suite de Fibonacci

Variables	a	b	c
Variables initiales	1	1	1
Valeurs successives au cours de l'itération	1 2 3 5 8 13	2 3 5 8 13 21	2 3 4 5 6 7
Valeurs de remplacement	b	a+b	c+1

Nous comprenons mieux l'itération c+1, qui n'a qu'une fonction de compteur qui stoppera la boucle.

Si nous souhaitons connaître une suite de Fibonacci plus grande il suffit de modifier l'instruction while.

Et si nous souhaitons la suite ligne par ligne il suffit de supprimer le end= » » comme ceci :

```
a,b,c=1,1,1                    |Assignation simultanée
while c<50:                    |Condition  « tant que » c est strictement inférieur à 50
    print(b)                   |Ecrit la valeur de b
    a, b,c=b, a+b, c+1         |
```

Résultat

```
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987
1597
2584
4181
6765
10946
17711
28657
46368
75025
121393
196418
317811
514229
832040
```

Table de multiplication

A l'issu de cet exercice le mécanisme m'est devenu bien plus clair.

```
a,b=1,1                    |Assignation simultanée
while a<21:                |Condition while
    print (a,b*7)          |Ecrit les valeurs de a et b fois 7
    a,b=a+1,b+1            |Incrémentation (compteur)
```

```
1 7
2 14
3 21
4 28
5 35
6 42
7 49
8 56
9 63
10 70
11 77
12 84
13 91
14 98
15 105
16 112
17 119
18 126
19 133
20 140
```

L'objectif ici étant de créer la table de multiplication par 7 et d'en afficher les 20 premiers résultats, que moi je souhaite sur une colone.

- Nous affectons 1 aux variables a et b
- Instruction while : tant que a est inférieur à 21
- Instruction print a ainsi que b*7
- Itération : nous redéfinissons les valeurs des variables a et b, créant ainsi le compteur iteratif (?), itération également appliqué à b afin de pouvoir créer la table de multiplication
- Si nous supprimons la dernière ligne, rien alors ne viendra modifier la valeur de a créant ainsi une boucle sans fin, affichant les valeurs 1 et 7.

Attention

Je viens de tester, c'est pas franchement une bonne idée : votre application risque de freezer.

References :

PEP8