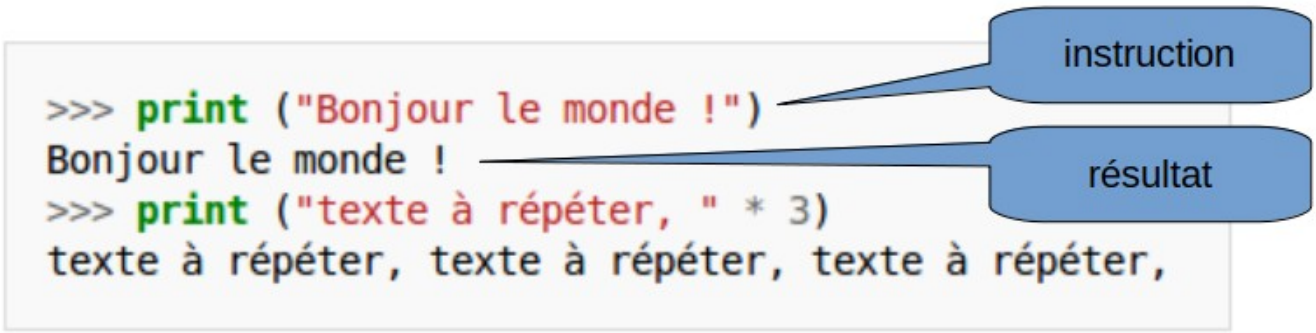
1. Qu'est-ce que python ?

Python est un langage de programmationrécent, inventé en 1991 par Guido Van Rossum (hollandais).
Python est un langage interprété, c'est-à-dire qu'il n'est pas nécessaire de le compiler avant de l'exécuter.
Python est un langage multiplateforme, c'est-à-dire disponible sur plusieurs architectures (PC, tablettes, smartphones, ...) et systèmes d'exploitation (Windows, Linux, Mac, Android...).
Le langage Python est gratuit, sous licence libre.
Actuellement, Python en est à sa version 3. Cependant, la version 2 est encore largement utilisée. Attention : Python 2 n'est pas compatible avec Python 3 !

2. L'interpréteur Python

L'interpréteur exécute les instructions au fur et à mesure qu'elle sont tapées. Le triple chevron >>> est l'invite de Python (prompt en anglais) et qui signifie que Python attend une commande. L'esprit d'utilisation de l'interpréteur est un peu le même que celui d'une calculatrice.
Sous Ubuntu on démarre l'interpréteur Python3 en tapant python3 sous la console. Il est alors possible d'écrire des instructions :



3. Programmer en Python avec un éditeur

L'interpréteur présente vite des limites dès lors que l'on veut exécuter une suite d'instructions plus complexe. Comme tout langage informatique, on peut enregistrer ces instructions dans un fichier, que l'on appelle communément un script Python, qui aura pour extension .py.

Exemple d'un programme helloWorld.py :

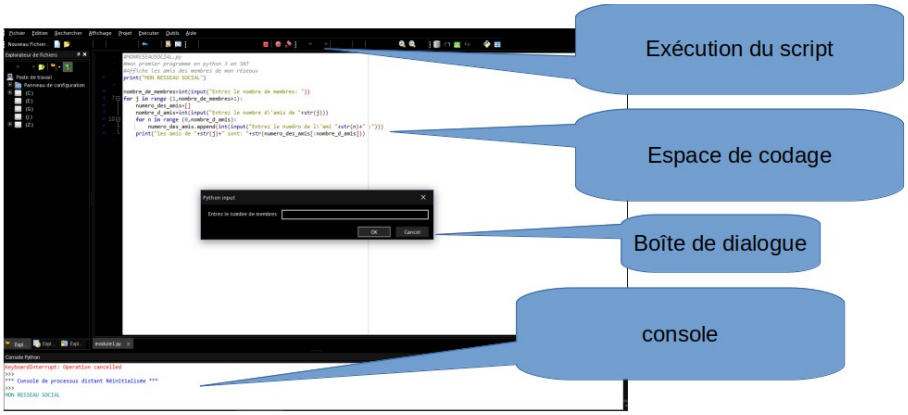
```
# helloWorld.py
# mon premier programme en Python 3
# affiche le message "Hello World !"

print ("Hello World !")
```

précède du commentaire

Pour exécuter le programme, il faut taper sous la console :

$ python helloWorld.py

Mais vous vous utiliserez edupython qui a sa console intégrée et qui interprète le code.



4. Les variables

Une variable est un espace mémoire dans lequel il est possible de stocker une valeur (une donnée).

Le nom des variables en Python peut-être constitué de lettres minuscules (a à z), de lettres majuscules (A à Z), de nombres (0 à 9) ou du caractère souligné (_).
Néanmoins, un nom de variable ne doit pas débuter ni par un chiffre, ni par _ et ne peut pas contenir de caractère accentué. Il faut absolument éviter d'utiliser un mot « réservé » par Python comme nom de variable (par exemple print).

Python est sensible à la casse, ce qui signifie que les variables
TesT, test ou TEST sont différentes.
Enfin, n'utilisez jamais d'espace dans un nom de variable puisque celui-ci est le séparateur d'instructions.

5. Les types de donnée en Python

Sous Python, il n'est pas nécessaire de définir le type des variables avant de pouvoir les utiliser. Il vous suffit en effet d'assigner une valeur à un nom de variable pour que celle-ci soit automatiquement créée avec le type qui correspond au mieux à la valeur fournie.

Le Python dispose de différents types de variables : entier, réel, chaîne de caractères, liste, etc.

```
a = 20                          // entier
pi = 3.14159                    // réel
msg = "Quoi de neuf ?"          // chaîne de caractères
tab = [9, 7, 6, 9]              // liste
```

6. Les opérateurs mathématiques

Opérations	Symboles	Exemples
addition	+	2 + 5 donne 7
soustraction	-	8 - 2 donne 6
multiplication	*	6 * 7 donne 42
puissance	**	5 ** 3 donne 125
division	/	7 / 2 donne 3.5 (en Python 3)
modulo	%	7 % 3 donne 1 (reste de la division entière)
quotient	//	7 // 3 donne 2 (quotient de la division entière)

7. Les opérateurs booléens

Opérations	Symboles
ET booléen	and
OU booléen	or
NON booléen	not

8. Affichage en console

Pour afficher du texte à l'écran on utilise la fonction print(). Attention, les parenthèses sont obligatoires en Python 3.

Exemples :

```
>>> print("je suis un texte")
je suis un texte
>>> print('je suis un texte')
je suis un texte
>>> print("j'ai bien compris")
j'ai bien compris
>>> print('j\'ai bien compris')
j'ai bien compris
>>> print("j'écris\nsur plusieurs\nlignes")
j'écris
sur plusieurs
lignes
```

On peut utiliser le guillemet ou la simple cote

\' pour afficher une apostrophe

\n pour insérer un saut de ligne (possible si on utilise des guillemets)

Il est possible de mélanger du texte avec du texte mais aussi avec le contenu de variables.

Exemples :

```
>>> print("Mon","prénom")
Mon prénom
>>> print("Mon","prénom",sep="_")
Mon_prénom
>>> a=20
>>> print("la valeur de a est",a)
la valeur de a est 20
>>> a=10
>>> b=5
>>> print(a,"+",b,"=",a+b)
10 + 5 = 15
>>> print("les amis de "+str(j)+" sont :"+str(numero_des_amis[:nombre_d_amis]))
Les amis de 2 sont : [1 4 5 6]
```

L'espace est ajouté automatiquement

permet de modifier le séparateur

permet d'afficher le tableau de 0 à nombre_d_amis

9. Saisie clavier

Pour que Python puisse lire ce que vous tapez au clavier, vous allez utiliser la fonction input(). Attention, avec Python 3 input() renvoie systématiquement une chaîne de caractères.

Exemples :

```
>>> message=input("Entrer un nom : ")          // lecture d'une
                                                chaîne de caractères
>>> n=1
>>> a=int(input("Entrer"+str(n)+"entier: "))                    // lecture de n entier ici n=1
str() permet de transformer le int en string et int() l'inverse

>>> b=float(input("Entrer un réel: "))                  // lecture d'un réel
>>>numero_des_amis=[ ] //crée une liste vide
>>>numero_des_amis.append(int(input("Entrez le numéro de l\'ami "+str(n)+" :"))) //
ajoute à la liste numero_des_amis le résultat de la lecture
```

10. Si... alors... sinon ...fin si

algorithme

si *condition*
alors *action 1*
sinon *action 2*
fin si

codage en Python

```
if a==2:
    print("La valeur est bien 2")
else:
    print(a,"est différent de 2")
```

Attention : l'indentation (tabulation) est obligatoire en Python.
Elle permet de délimiter les blocs.

11. Le sinon...si...

algorithme

si *condition1*
alors *action 1*
sinon si *condition2*
alors *action 2*
sinon *action 3*
fin si
fin si

codage en Python

```
if a==2:
    print("La valeur est bien 2")
elif a==3:
    print("La valeur est bien 3")
else:
    print(a,"est différent de 2 et de 3")
```

elif : contraction de else if

12. La boucle pour ...

On utilise cette boucle quand on connaît le nombre de répétitions à effectuer.

algorithme

pour indice de Vi à Vf
faire *actions*
fin pour

i : indice
Vi : valeur initiale
Vf : valeur finale non incluse

codage en Python

```
for i in range(0, 5):
    print("Hello",i)
```

```
Hello 0
Hello 1
Hello 2
Hello 3
Hello 4
```

Attention : l'indentation est obligatoire pour le contenu de la boucle

13. La boucle tant que ...

On utilise cette boucle quand on ne connaît pas le nombre de répétitions à effectuer.

algorithme

tant que *condition vraie*
faire *actions*
fin tant que

codage en Python

```
a = 0
while (a < 7):
    a = a + 1
    print(a)
```

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

Attention : l'indentation est obligatoire pour le contenu de la boucle