# *Introduction à la programmation, cours 2*

## 1) Entrée / Sortie

La plus part des programmes interagissent avec un utilisateur. Un programme peut vouloir dire des choses à l'utilisateur, c'est à dire afficher des trucs à l'écran (**Sortie**). Pour cela on utilise la fonction **print** qu'on a déjà vu au cours précédent. Mais un programme peut aussi avoir besoin que l'utilisateur lui communique des données (en générale à l'aide du clavier) (**Entrée**). Pour cela, on utilise la fonction **input()** :

```
variable = input()  #l'exécution s'arrête, et attend que l'utilisateur (ici c'est vous) tape quelque chose au
                    # clavier puis press enter
print variable      # affiche ce que vous avez entré au clavier
```

Prenons un exemple un peu plus évolué, on va écrire un programme qui demande à l'utilisateur son nom, puis lui dit son nom.

```
print "Quel est votre nom ?"
var_nom = input()
print "Vous vous appelez : ", var_nom
```

Remarquez qu'à l'exécution de **input()**, l'exécution s'arrête et attend que l'utilisateur tape quelque chose au clavier et presse **entrée**.
Vous vous demandez peut-être pourquoi il y a des parenthèses après le input(). Peut-être avez-vous même déjà essayé d'y mettre quelque chose voire ce qu'il se passait (si c'est le cas, bravo, c'est comme ça qu'on progresse, en faisant des testes hasardeux). Essayez le code ci-dessous et tirez vos conclusions.

```
var_age = input("Quel est votre nom ?")
print "Vous vous appelez : ", var_nom
```

Ainsi, quand l'ordinateur exécute la fonction input() il :

- interrompt l'exécution du programme
- affiche éventuellement un message à l'écran
- attend que l'utilisateur entre une donnée au clavier et appuie sur entrée
- reprend l'exécution du programme

Ce que l'utilisateur écrit au clavier est transformé en une valeur de type *chaîne de caractères* par Python (notez qu'un nombre peut être interprété comme une chaîne de caractère). Par exemple, le code suivant va vous donner un erreur :

```
print "Je vais vous montrer que je sais multiplier un nombre par 2."
var = input("Tapez un nombre")
var_double = var*2
print "Je le multiplie par deux, ça me donne : ", var_double
```

L'erreur vient du fait que Python interprète ce qu'écrit l'utilisateur comme une chaîne de caractères et pas comme un entier. Dieu merci, il existe une façon de changer le type d'une variable en utilisant la fonction **int()** :

```
print "Je vais vous montrer que je sais multiplier un nombre par 2."
var = input("Tapez un nombre")
var_int = int(var) # ici var_int est un entier.
var_double = var_int * 2
print "Je le multiplie par deux, ça me donne : ", var_double
```

Il est en fait préférable de le faire comme suit :

```
print "Je vais vous montrer que je sais multiplier un nombre par 2."
var = int(input("Tapez un nombre"))
var_double = var * 2
print "Je le multiplie par deux, ça me donne : ", var_double
```

Faites un teste pour voir ce qu'il se passe si, dans le code ci-dessus, l'utilisateur ne tape pas un entier.
Comme vous vous en doutez, la fonction **float()** existe aussi.
Au passage, il existe une fonction qui **renvoie le type d'une donnée**, c'est la fonction **type()** .

```
var_1 = "tata yoyo"
print type(var_1)
var_2 = 3
var_3 = 3.0
var_4 = input("tapez un nombre")
var_5 = int(var_4)
var_6 = float(var_4)
```

**Exercice_0:** De quels types sont les variables var_2, var_3, var_4, var_5 et var_6 dans l'exemple ci-dessus? Réfléchissez-y puis afficher-lez (en utilisant `print var_2` etc) pour vérifier que vous aviez bon.

**Exercice_1**: Écrivez un programme qui demande à l'utilisateur comment il s'appelle, puis lui demande son âge, et enfin affiche : « Bonjour [nom], comment allez-vous ? L'année prochaine vous aurez : [age] ans ».

# 2) Un nouveau type: les booléens

Nous avons déjà vu trois types de données différents: les entiers, les flottants et les chaînes de caractères. On va maintenant voire un quatrième type, fondamentale en informatique: **les booléens**. Une variable de type booléenne ne peut prendre que deux valeurs: Vraie ou Faux. En Python: **True** ou **False** (notez les majuscules à True et False).
Une **expression booléenne** est une expression qui voit *True* ou *False*.
Voici quelques expression booléennes (c'est à dire des trucs qui valent True ou False) utilisant les opérateurs suivants: == (« est égal à »; != (« est différent de »; >; <; >= (« est supérieur ou égal à); <= (« est inférieur ou égal à »). Faites vos propres testes pour biens comprendre ce que tout cela signifie (et remarquez à quel point c'est intuitif).

```
Print 3 ==3
print 3 == 5
print 3 + 2 == 5
print 3 != 3       #!= veut dire différent
print 3 != 4
print 3 > 4
print 3 <= 4
print 3 <= 3
```

On peut bien sûr définir des variables de types booléennes :

```
var_bool_1 = True
var_bool_2 = False
print var_bool_1, var_bool_2
```

Vous vous demandez peut-être à quoi ces booléens peuvent-il bien servir?! Tout va s'éclairer quand on va voir les structures conditionnelles.

# 3) Structures conditionnelles : if / else

## 1. Instruction conditionnelle : if

**Objectif** : effectuer une action seulement si une condition est vérifiée.

**Syntaxe en Python** :

```
if <condition> :
        # 4 espaces d'indentations!
        instructions à exécuter si la condition est vraie.
```

ATTENTION A L'INDENTATION . L'indentation permet à python de savoir dans quel bloque se trouve une instruction. Je vous en dis pas plus, c'est très intuitif, vous finirez bien par comprendre ;)

Un exemple plus concret, essayer d'abord avec un nombre positif, puis un négatif :

```
var = int(input("Tapez un nombre"))
if var > 0 :
        print "le nombre que vous avez tapé est positif"
```

Essayez de taper un nombre flottant, genre 3.2. Que se passe-t-il ? Comment résoudre le problème ?

## 2. Instruction conditionnelle : if … else

**Objectif** : effectuer une action si une condition est vérifiée, une autre si elle est fausse.

**Syntaxe en Python** :

```
if <condition> :
        instructions à exécuter si la condition est vraie.
else :
        instruction à exécuter si la condition est fausse.
```

Notez que le else n'est pas suivi d'une condition.

Un exemple:

```
var = int(input("Tapez un nombre"))
if var > 0 :
        print "Le nombre que vous avez tapé est positif"
else :
        print "Le nombre que vous avez tapé est négatif ou nul"
```

Dans l'exemple ci-dessus, que se passe-t-il si vous tapez un float, genre 3.2? Analyser le message d'erreur qui est envoyer. Comme réparer l'erreur?

## 3. Instruction conditionnelle avec elif

Ici, je vous laisse comprendre tout seul grâce à un exemple.

```
var_age = int(input("Quel âge avez-vous"))
if var_age < 0 :
        print "Vous mentez !"
elif var_age < 3 :
        print "Que quelqu'un enlève ce bébé de devant l'ordi!"
elif var_age < 12 :
        print "Tu ferais mieux d'aller jouer au grand air!"
else :
        print "Vous avez un âge raisonnable."
```

**Exercice_2***:* Ecrire un programme qui:

- demande à l'utilisateur un premier entier en affichant "Entier a=?"
- demande à l'utilisateur un deuxième entier en affichant "Entier b=?"
- Si la somme des deux entiers est supérieure ou égale à 100, le programme affiche "La somme dépasse 100". Sinon, le programme affiche "La somme est (*véritable valeur de a+b*)."

***Exemple 1:***
Entier a=? **l'utilisateur entre 5**
Entier b=? **l'utilisateur entre 7**
La somme est 12.

***Exemple 2:***
Entier a=? **l'utilisateur entre 90**
Entier b=? **l'utilisateur entre 12**
La somme dépasse 100.

# 4) Opérateurs sur les booléens

Ici, pas le choix, il va falloir faire un peu de math. Et même, plus précisément, de la logique. Nous allons voir quelques opérateurs logiques sur les booléens (c'est des gros mots mais en vrai, quand ça fait 20 ans qu'on manipule ce genre de chose, on trouve ça très facile ;)).

## 1. Le « ou » logique: or

(expr_1 **or** expr_2) vaut True si au moins une des deux expressions expr_1 et expr_2 est vraie. Elle vaut False si expr_1 et expr_2 sont toutes les deux fausses. Noter que le « ou » logique et différent du « ou » que l'on utilise dans « fromage ou dessert » par exemple (dans le cas d'un « ou » logique, on aurait le droit de prendre soit seulement du fromage, soit seulement du dessert, soit du fromage ET du dessert, cette dernière possibilité n'est pas possible dans un restaurant.
Voyons sur un exemple :

```
var_age = int(input("Quel âge avez-vous"))
if var_age <= 26 or var_age >= 60 :
        print "Vous avez le droit à une réduction"
else :
        print "Vous allez payer plein pot…"
```

**Question** : pour quel valeur l'expression booléenne (var_age <= 26 ou var_age >= 60) vaut-elle True? Faites des testes sur l'exemple ci-dessus pour bien comprendre. N'hésitez pas à changer la condition (var_age <= 26 ou var_age >= 60).

## 2. Le «et» logique: and

(expr_1 **and** expr_2) vaut vrai si les deux expressions expr_1 et expr_2 sont vraies.
Voyons sur un exemple :

```
var_age = int(input("Quel âge avez-vous"))
if var_age > 26 and var_age < 60 :
        print "Vous allez payer plein pot…"
else :
        print "Vous avez le droit à une réduction"
```

**Question** : pour quel valeur l'expression booléenne (var_age <= 26 and var_age >= 60) vaut-elle Vraie? Faites des testes sur l'exemple ci-dessus pour bien comprendre.

Rendez-vous sur moodle pour un quizz sur les booléens.

# 5) Quelques exos pour les plus rapides

**Exercice_1*** Ecrire un programme qui :

1. demande à l'utilisateur deux entiers successivement, puis
2. qui affiche la somme de ces deux entiers, puis
3. qui affiche la différence entre le premier entier et le deuxième entier.

Pour les phrases à afficher, vous pouvez suivre le modèle de l'exemple suivant:

**Premier entier:** ** l'utilisateur tape 3**
**Deuxième entier:** **l'utilisateur tape 5**
**Somme : 8.**
**Différence: -2.**

**Exercice_2*** Ecrire un programme qui :

- affiche le titre "Maximum de trois entiers",
- puis qui demande successivement trois entiers à l'utilisateur,
- puis qui affiche "Le maximum est" suivi de la valeur du maximum des trois entiers fournis.

Pour les phrases à afficher, vous pouvez suivre le modèle suivant:

**Maximum de trois entiers**
**1er entier**= * l'utilisateur tape 3*
**2eme entier**= * l'utilisateur tape 9*
**3eme entier**= * l'utilisateur tape -4*
**Le maximum est : 9**

**Exercise 3*.** Ecrire un programme qui demande à l'utilisateur de donner une séquence de noms d'étudiants (e.g. "Alvaro Arenas, Roberto Bruno, Alexis Krauthgamer"), et crée une liste de tous les étudiants, une liste contenant les étudiants dont le nom de famill commence par A – M, et enfin une liste des étudiants dont les noms de famille commencent par N – Z. (Pour faire cet exo, vous aurez besoin de reprendre le premier cours sur les listes bien sûr, en particulier on a vu la fonction split() qui ici vous sera bien util).