# Quelques instructions Par l'exemple Python 2.5

## Entrées/Sortie :

Afficher à l'écran :

| | |
|---|---|
| print a | Affiche la valeur de a |
| print ''a'' ou print 'a' | Affiche la lettre « a » |
| print ''La valeur de a est :'',a | Affichage mixte (texte et valeurs) |

Demander :

| | |
|---|---|
| x=input('Nombre de côtés ? ') | Demande un nombre |
| ch=raw_input('Quel est votre nom ? ') | Demande un texte |

GROUPE ALGORITHMIQUE

## Boucles :

| | |
|---|---|
| for l in ['a','e','i','o','u','y'] | Lister les voyelles |
| for i in range(5) | Liste tous les nombres entiers de 0 à 4 (intervalle $[\![0 ; 5[\![$ ) |
| for i in range(1,6) | Liste tous les nombres entiers de 1 à 5 (intervalle $[\![1 ; 6[\![$ ) |
| for i in range(3,10,2) | Parcours l'intervalle $[\![3 ; 10[\![$ avec un pas de 2 : 3, 5, 7,9 |
| while i<10 : …… | Exécute la suite d'instructions tant que i<10 |

## Opérateurs logiques (pour les tests):

| | | | | | |
|---|---|---|---|---|---|
| == | Egal à | != | Différent de | >= | Supérieur ou égal …. |

## Opérations

// donne le quotient entier de la division (exemple : 7//2 ou 7.0//2 donnent 3)

/ donne le quotient décimal (ou entier si les nombres le sont) ; exemples : 7/2 donne 3 et 7.0/2 donne 3.5

% donne le reste de la division ; exemple : 7%2 donne 1

int donne la partie entière pour les nombres positifs ; exemple : int(3.2) donne 3

## Tests :

```
x= int(input("entrer un nombre entier :"))
if x%2==0:
        print "Nombre pair"
else:
        print "Nombre impair"
```

Affiche si le nombre entré est pair ou non :
Le symbole % donne le reste de la division par 2
Le symbole == indique un test d'égalité
La déclaration d'un bloc de commande se faite grâce à « : »
On décale le bloc, et on aligne les instructions verticalement
La fin du bloc est détectée par le retour d'indentation (alinéa)

```
x= input("entrer un nombre entier :")
# commentaires non pris en compte lors de l'exécution  bla bla bla, il suffit de commencer la ligne par #
if x%1 !=0 :
    print 'votre nombre n'est pas entier'
    x=int(x)
    print 'vous auriez pu choisir sa partie entière', x
elif x%2==0:
        print "Nombre pair"
else:
        print "Nombre impair"
```

## Tirage aléatoire :

Votre programme doit alors commencer par : **from random import ***

| | |
|---|---|
| randint(1, 6) | Tire un entier dans l'intervalle [1 ; 6] |
| random() | Tire un "réel" dans l'intervalle [0 ; 1[ |

## La tortue :

On peut dessiner avec une tortue (LOGO), on commence le programme par

**from turtle import ***

| | |
|---|---|
| forward(integer) | avance le crayon sur la distance donnée |
| backward(integer) | recule le crayon sur la distance donnée |
| circle(radius) | dessine un cercle de rayon donné |
| width(integer) | largeur du trait |
| up() | lève la main (déplace sans tracer) |
| down() | descend la main pour pouvoir écrire |
| color(str) | couleur du crayon |
| write(str) | écrit le texte à la position du curseur |
| position() | retourne les coordonnées de votre crayon |
| left(integer) | tourne la main de nombre de degré vers la gauche |
| right(integer) | tourne la main de nombre de degré vers la droite |
| reset() ou clear() | efface votre dessin |
| goto(integer,integer) | déplace le crayon aux coordonnées indiquées |

## Pour utiliser des caractères accentués :

Entrer avant cette ligne : vous pouvez la mettre au début du programme

```
# -*- coding: iso8859-1 -*-
```

## Travailler avec les listes :

Exemple remplir une liste des premiers carrés parfaits :

```
L=[]
for i in
range(10):
    L.append(i*i)
while j<len(L):
    print L[j]
    j=j+1
```

On initialise une liste vide

L.append('ce que je veux') ajoute « ce que je veux » à la liste L

Len(L) : Donne le nombre d'éléments de la liste L

L[i] : renvoie l'élément d'indice i (Attention, le premier élément a pour indice 0)

Remarque : on peut avantageusement remplacer les 3 dernières lignes par

```
for c in L:
    print c
```

## Travailler avec les chaines de caractère : (Il y a une grande analogie avec les listes)

```
mot="bonjour"
cherche="jour"
if mot.find(cherche)>0:
    deb=mot.find(cherche)
    fin=deb+len(cherche)
    print "Dans",mot,"si je retire",cherche,"il reste",mot[:deb]+mot[fin:]
else :
    print "Dans",mot,"il n'y a pas",cherche
```

Longueur d'une chaine : len(ch)

`mot.find(cherche)` Renvoie la position de la sous-chaine **cherche** dans **mot** (Renvoie -1 s'il ne la trouve pas)

$i^{ème}$ caractère de la chaine : ch[i] (Attention, le 1er caractère a pour indice 0, comme pour les listes)

Prendre une sous-chaine d'une chaine : ch[indice_debut :indice_fin]

## Les fonctions :

```
def puissance(a,b):
    c=1
    for i in range(b):
        c=c*a
    return c

print puissance(2,4)
```

On peut créer de nouvelles fonctions, un exemple ici avec la fonction puissance, il suffit de lui donner un nom (ici puissance), à la fin des actions, on renvoie alors le résultat grâce à l'instruction **return**.