DEBUTER AVEC PYTHON

1) Les variables

Il y a 4 types de variables: les nombres (entiers ou décimaux), les chaînes de caractères, les listes et les dictionnaires:

256 est un entier
23.5 est un décimal
"bonjour" et 'maison' sont des chaînes (entourées de guillemets ou d'apostrophes)
[1,5,3,7] est une liste de longueur 4

Il n'y a pas besoin de déclarer les variables. Elles sont créées à la première utilisation.
Exemples
nombre = 35 est une instruction qui crée la variable nombre et qui lui affecte la valeur 35,
phrase = "Bonjour Max" crée la variable phrase et lui affecte la valeur "Bonjour Max".
suite = [1,5,3,7] crée la variable suite...
On a accès aux différents termes de la suite par la notation suite[k] qui renvoie le terme numéro k.
Attention la numérotation commence à 0.
suite[1] vaut donc 5.
On peut procéder de la même façon pour les chaînes :
si nom = 'theoreme', nom[3] vaut 'o'.

2) Entrées – sorties

La fonction d'entrée est **input**
La syntaxe est la suivante:
nombre = input("Entrer un nombre: ")
L'ordinateur va afficher : *Entrer un nombre:*
Par exemple on tape 17, suivi de la touche "Entrée"
Le programme crée alors la variable nombre et lui affecte la valeur "17".
ATTENTION: input génère uniquement des **chaînes.**
Pour convertir une chaîne en nombre on utilise les fonctions **int**. **float** ou mieux **eval**
Ainsi, on ajoute la ligne: nombre = int(nombre),
ou on compose les deux fonctions: nombre=int(input("Entrer un nombre: "))

La fonction de sortie est **print** .

3) Un premier programme

Voici un programme qui va demander votre nom, votre année de naissance, puis qui va vous saluer et vous donner votre âge en 2012.
Pour écrire un programme, on lance **IDLE**, et on choisit **File**, **New Window**
Entrer le programme suivant (on met toujours une seule instruction par ligne)

```
prenom=input("Quel est ton prénom? ")
annee=int(input("Quelle est ton année de naissance? "))
age=2012-annee
salutation="Bonjour "+prenom
print (salutation)
print ("En 2012 tu auras ",age," ans")
```

Enregistrer le programme par **File**, **Save As** (Choisir par exemple "**salut.py**")
Pour le lancer **Run**, **Run module**
Remarques:
IDLE utilise une coloration très pratique: chaînes en vert, commandes en violet, ….
L'instruction *salutation="Bonjour "+prenom* , contient l'opérateur + de **concaténation** (mise bout à bout) des chaînes
Mettre des espaces dans les **input** et **print** pour aérer.

4) Exercices

Exercice 1: écrire un programme qui vous demande un nombre, calcule son carré, et vous répond par exemple: *"Le carre de 124 est 15376"*
La fonction puissance est représentée par **
Exemple: 5**2=25

Exercice 2: écrire un programme qui vous demande les coordonnées de 2 points A et B, puis qui vous donne les coordonnées du milieu de [AB].

Exercice 3: écrire un programme qui vous demande les coordonnées de 2 points A et B, puis qui vous donne la distance AB (valeur approchée).
Rappel: AB = $\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$, la fonction racine est **sqrt**
Exemple: sqrt(36) = 6, sqrt(20) = 4.47213595499958
Python ne dispose pas de cette fonction par défaut, pour l'utiliser il faut l'importer du module **math**
Votre programme doit comporter au début la ligne: **from math import sqrt**

5) La boucle FOR

Voici un exemple

```
for k in range(10):
    print( k,)
```

Ce programme de 2 lignes effectue une **boucle**. Il affiche ceci
0 1 2 3 4 5 6 7 8 9
La variable k prend toutes les valeurs de 0 à 10 exclu.
La ligne qui finit l'instruction **for** doit finir par 2 points
Tout ce qui doit être fait dans la boucle doit ensuite être décalé de 4 espaces (IDLE le fait automatiquement). Cela s'appelle **l'indentation**.
Python utilise l'indentation pour les boucles, les tests, les créations de fonctions, …….

range peut comporter jusqu'à 3 paramètres: START, END, STEP

```
for k in range(12, 54, 3):
    print (k,)
```

affiche 12 15 18 21 24 27 30 33 36 39 42 45 48 51

for est aussi très pratique pour traiter les chaînes, et donc faire du codage

```
mot='BONJOUR'
for lettre in mot:
    print (lettre)
```

Un dernier exemple qui montre l'indentation:

```
#Table des carrés et des cubes des nombres de 20 à 30

print( 'Voici la liste des carrés')
for nombre in range(20,31):
   carre=nombre**2
   print ("le carré de ", nombre, " est ", carre)
print ('Voici la liste des cubes')
for nombre in range(20,31):
   cube=nombre**3
   print ("le cube de ", nombre, " est ", cube)
print( "Voila j'ai fini !")
```

Remarque: le dièse en début de ligne permet de commenter le programme pour s'y retrouver
Tout ce qui est derrière est ignoré par Python.
Il est indispensable de mettre des commentaires dans les programmes longs (des milliers de lignes…)
Ainsi on arrive à travailler dessus plus facilement.

Exercice: écrire un programme qui vous demande un nombre et affiche la table de multiplication.
En sortie il doit vous afficher:

```
Table de multiplication par 7
1  fois 7 =  7
2  fois 7 =  14
3  fois 7 =  21
4  fois 7 =  28
5  fois 7 =  35
6  fois 7 =  42
7  fois 7 =  49
8  fois 7 =  56
9  fois 7 =  63
10  fois 7 =  70
```

6) Le test IF

Etudions un exemple:

```
nombre=int(input("Entrer un nombre quelconque entre 1 et 10: "))
if (nombre<1 or nombre>10):
   print ("Vous êtes trop nul!")
else:
   print( "Vous êtes trop fort")
```

Si la condition qui est entre parenthèses après le **if** est réalisée, l'instruction qui suit est exécutée,
sinon (else !) c'est celle qui suit le **else** qui est réalisée.
Les tests classiques sont:
if(A = = B) teste l'égalité de A et B. Attention il faut mettre 2 =
if(A ! = B) "si A est différent de B"
if(A < B), if(A<=B), ….
Et on peut compléter par des connecteurs logiques: or (OU) and (ET)…….

Exercice :
Ecrire un programme qui demande 2 nombres a et b et qui résout l'équation ax+b=0.
Il faut traiter le cas où a = 0.

7) La boucle WHILE

Etudions un exemple:

```
nombre=int(input("Entrer un nombre entier quelconque non nul: "))
S=0
n=1
while (n <=nombre ):
    S=S+n
    n=n+1
print('La somme est',S)
```

Tester ce programme avec diverses valeurs de **nombre**.
Que fait-il ?

Exercice :
On considère une suite de nombres entiers, qui commence par 4, et telle qu'un nombre soit la moitié du précédent auquel on ajoute 10.
Le premier nombre est 4, le deuxième est donc 12 (4/2 + 10), le troisième est 16, …..
On constate que ces nombres se rapprochent de 20.
(Pour les terminales $u_0 = 4$ et $u_{n+1} = 0{,}5u_n + 10$ la limite de u est 20)

Ecrire un programme qui demande une valeur A proche de 20 (exemple A = 19,9999999),
et qui retourne le numéro du premier nombre de la suite qui dépasse A.

8) Exercices

a)Ecrire un programme qui vous demande les coordonnées de 2 points A et B, puis qui vous dit si la droite (AB) coupe l'axe des abscisses et qui donne alors les coordonnées du point d'intersection.

b)Etude d'un triangle
Vous demandez les coordonnées de 3 points A, B, C du plan.
Le programme doit vous « répondre » :
les points sont alignés (et le programme s'arrête là avec un avertissement)
ou
le triangle ABC est rectangle ou le triangle est isocèle....voire les deux à la fois !

c)Programme de jeu : le juste prix
L'ordinateur choisit un nombre entier au hasard entre 10000 et 40000.
Vous êtes chargés de deviner ce nombre le plus rapidement possible.
L'ordinateur doit afficher :
« Proposition n°1 : », puis « Proposition n°2 : »,.....
Il vous répond par « C'est plus » ou « C'est moins ».
Il faut utiliser le module random et la fonction randrange :
Au début du programme vous écrirez :
from random import randrange (n'importe que la fonction randrange)
ou
from random import * (importe toutes les fonctions de hasard)
Vous pouvez compléter ce programme avec un commentaire sur le score :
« trop fort » ou « trop nul »....
Vous pouvez aussi imposer un temps limite, en exploitant le module time !