Programmation sous Python

Un programme, un algorithme ou encore un script est constitué d'une **successions d'instructions** bien ordonnées qui utilisent des **variables**.
Le site repl.it vous permet d'écrire votre programme et de le tester en ligne.

I. Les variables

Il existe différents types de **variables** sous Python, on trouve entre autres :

int : les nombres entiers
float : les nombres décimaux
str : les chaînes de caractères
list : les listes

On peut écrire à peu près n'importe quel nom pour une variable, mais il ne faut pas qu'il y ait d'accents, ni d'espaces, ni de caractères spéciaux (sauf l'underscore _).
Remarque 1 : la casse est prise en compte (majuscule et minuscule)
Remarque 2 : certains noms ne peuvent pas être utilisés comme le nom d'une variable car ils désignent une instruction déjà existante de Python : input print else if del …

Le **type** d'une variable est généralement automatiquement défini en fonction de ce qu'on peut lui **affecter** avec le symbole =
On peut vérifier le type d'une variable en utilisant les instructions type() et **print()**

```
abc = 3                  # On affecte le nombre entier 3 à la variable abc
print (type(abc))        # Le programme affiche <type 'int'>

Pi = 3.14159             # Bien utiliser le symbole . à la place de ,
print (type(Pi))         # Le programme affiche <type 'float'>

nom = "Jean Luc"         # Il faut mettre le texte entre guillemets
print (type(nom))        # Le programme affiche <type 'str'>

Fib = [1,1,2,3,5,8,13]   # Les crochets permettent de créer une liste
print (type(Fib))        # Le programme affiche <type 'list'>
```

II. Saisie et affichage

A. Affichage

L'affichage se fait à l'aide de l'instruction **print()** (N'oubliez pas les parenthèses!)

```
nom="Jyn Erso"
print("Bienvenue", nom)     # Affiche "Bienvenue Jyn Erso"
```

B. La saisie

Pour demander à l'utilisateur de saisir un nombre ou un texte, on utilise l'instruction **input()**

ATTENTION : Avec l'instruction input() la variable est par défaut une chaîne de caractères (str) !
Il faut donc la convertir en nombre entier ou décimal si vous voulez effectuer des opérations

Exemple 1 :

```
print ("Quel est votre nom?")
nom=input()
```

Exemple 2 :

```
poids=float(input("Quel est votre poids?"))
```

III. Opérations sur les variables

A. Opérations sur les nombres

Les 4 opérations élémentaires se font à l'aide des opérateurs courants + - * /

```
var=var+1          #  Augmente var de 1
print(10/6)        #  Affiche le quotient décimal de 10 par 6
10//6              #  Calcule la partie entière du quotient (le quotient de la division euclidienne)
10%6               #  Calcule le reste de la division euclidienne de 10 par 6
2**3               #  Permet le calcul d'une puissance : 2**3 signifie 2*2*2
a=float(8)         #  Convertit le nombre entier 8 en un nombre décimal
b=int(8.7)         #  Convertit le nombre décimal 8,7 en un nombre entier : 8
round(x,3)         #  Arrondit le nombre x au millième près
```

Remarque : le **module math**.
Il existe de nombreuses fonctions mathématiques qui ne sont pas chargées par défaut.

```
import math
print(math.sqrt(25))
print(math.cos(math.pi))
```

ou

```
from math import *
print(sqrt(25))
print(cos(pi))
```

#Charge toutes les fonctions du #module math

Remarque : **sqrt** pour square root, la **racine carrée**
De même avec les instructions suivantes :

```
random()           #Génère un nombre (décimal!) aléatoire entre 0 et 1
cos() sin() tan()  #cosinus, sinus ou la tangente d'une mesure d'angle exprimée en radians
```

B. Opérations sur les chaînes de caractères

```
nom = input ("Quel est votre nom?")
prenom = input("Quel est votre prénom ?")
login = prenom[0] + "." + nom
email = prenom + nom + "@coruscant.com"
print (login)
print (email)
```

#**prenom[0] renvoie la 1ère lettre du mot**,
#prenom[1] la 2ème lettre et **+** sert à**concaténer**
du texte
#Affiche J.Erso par exemple
#Affiche JynErso@coruscant.com par exemple

```
prenom = "Vader"
prenom[-1]                  #Renvoie "r", la dernière lettre du prénom
len(prenom)                 #Renvoie la longueur de la chaîne : len(prenom)=5
prenom[0:2]                 #Renvoie "Va" le 1er indice 0 est inclus mais pas le dernier d'indice 2
print(prenom[:3])           #Affiche "Vad"
print(prenom[3:])           #Affiche "er"
prenom = prenom[:3] + "o" + prenom[4:]     #prenom devient "Vador"
```

C. Opérations sur les listes

```
Divers = [5,"art",7]        #Une liste peut contenir des variables de types différents
Divers.append(2)            #L'instruction append permet de rajouter une variable à la liste
print (Divers)              #Affiche [5,"art",7,2]
print (Divers[1])           #Affiche "art"  chaque élément est indexé, 0 pour le 1er, 1 pour le 2nd
Divers[2]=8                 #La valeur 7 est remplacée par 8
del Divers[1]               #L'élément "art" est supprimé de la liste
Divers.sort()               #Trie la liste par ordre croissant
len(Divers)                 #Renvoie le nombre d'éléments dans la liste
Divers.index(5)             #Renvoie l'index du nombre 5, ici 0
Divers[-2]                  #Renvoie l'avant-dernier élément de la liste
Divers[-2 :]                #Renvoie les 2 derniers éléments de la liste
Divers[ :2]                 #Renvoie les 2 premiers éléments de la liste
Divers[:]=[]                #Vide la liste
```

IV. Structures conditionnelles

Les instructions **if else elif** permettent des instructions conditionnelles mais demandent une syntaxe précise :

```
if nombre>=0 :                  #les deux points sont nécessaires
        print ("positif")       #il faut décaler les instructions avec la touche de tabulation
else :                          #les deux points sont à nouveau nécessaires
        print("strictement négatif")
```

On peut « emboîter » plusieurs conditions, mais alors « else if » s'écrit **elif**

```
if s !=59 :             #  != signifie différent
        s=s+1
elif m==59 :            #  == permet le test d'une égalité, Python renvoie vrai ou faux
        s=0
        m=0
        h=h+1
else :
        s=0
        m=m+1
print (h,m,s)           # Ici on est sorti des deux conditions
```

V. Boucles

Les boucles permettent de répéter une suite d'instructions.

A. Tant que

La boucle « tant que » (**while** en anglais) s'utilise jusqu'à ce qu'une certaine condition soit vérifiée. On ne sait généralement pas combien de fois les instructions vont être répétées.

```
S=0
i=1
while S<100 :           #Tant que S est st. Inférieur à 100, les instructions qui suivent sont répétées
        S=S+1/i         #Comme pour les instructions conditionnelles, il faut utiliser la tabulation
        i=i+1           #à chaque instruction répétée
print(i)                #Cette instruction n'est pas répétée
```

B. Pour

La boucle « pour » (**for** en anglais) s'utilise lorsque l'on souhaite répéter des instructions un nombre déterminé de fois

```
list=[3,2,1]
for i in list :
        print(i)
print("Partez")
```

```
list=[]
for i in range(5):          #pour i allant de 0 à 4
  list.append(i**2)
print (list)
```

```
for count in range(3):    #Répète 3 fois
  print("Olé !")
```

Remarque : On peut « emboîter » les instructions **while for** et **if** mais il faut alors décaler davantage les instructions avec la tabulation.

```
for i in range(1,10):                       #pour i allant de 1 à 9
        if i%2==0 :
                print("le nombre " , i , "est pair")
        else :
                print("le nombre " , i , "est impair")
```

VI. Les fonctions

Les **fonctions** permettent de **mémoriser une suite d'instructions**. Imaginons un programme de calculs : On considère un nombre, on lui ajoute 4, on le multiplie par 2 puis on rajoute le nombre de départ.

```
def prog(x) :          #Ici cette fonction attend 1 argument x.
        r=x+4
        r=r*2
        r=r+x
        return r       #La fonction prog renvoie le nombre r

print(prog(-3))        #Le programme appelle la fonction prog, puis affiche -1
```

```
def puissance(a,n)     #Ici cette fonction attend 2 arguments.
        print a**n

puissance(2,5)         #Le programme affiche 32 mais la valeur n'est pas stockée
```

Attention : prog(-3)==-1 est vrai, alors que puissance(2,5)==32 est faux car la fonction puissance affiche un nombre mais ne le renvoie pas

Plus complexe :

```
def test(n):                                                    #Ici on définit une fonction mais
  t=1                                                           #elle n'est pas exécutée tant
  for i in range (2,n):                                         #qu'elle n'est pas appelée
    if n%i==0:
      print ("Ce nombre n'est pas premier car il est divisible par ",i)
      t=0                                                       #t est un utilisé comme un
  if t==1:                                                      #booléen. Au début du program-
    print ("Ce nombre est premier")                             #me il est par défaut à 1. Si à la
                                                                #fin de la boucle est il toujours à
print("Test pour savoir si un nombre est un nombre premier")    #1 c'est qu'il n'a aucun autre
print("Veuillez saisir un nombre entier supérieur à 2")         #diviseur que 1 et lui-même.
nombre = int(input())
test(nombre)                                                    #Ici on appelle la fonction « test »
```

VII. Module graphique

Python possède un **module graphique** qui s'appelle **turtle**. Comme pour le module math il n'est pas chargé par défaut. On peut donc commencer un programme par l'instruction : **from turtle import ***

Instructions usuelles :

```
reset()          #Efface tout
goto(x,y)        #Aller au point de coordonnées (x ;y)
forward()        #Avancer d'une distance exprimée en pixels
backward()       #Reculer
penup()          #Relever le crayon
pendown()        #Abaisser le crayon
pencolor("red")  #Couleur du stylo en rouge
left(angle)      #Tourner à gauche d'une mesure d'angle exprimé en degré)
right(angle)     #Tourner à droite
speed(3)         #Vitesse entre 1 et 11 (du moins au plus rapide)
position()       #Renvoie les coordonnées du crayon
```