

PARTIE 1 : Les premiers mots clé

print() : affiche la variable demandée ou le message proposé entre ' ' .

int() : Reconnait comme un entier ou donne la troncature .

float() : Reconnait comme un réel ou convertit en réel si c'est faisable

str() : Reconnait comme une chaîne de caractères ou convertit en chaîne de caractères si c'est faisable

GLOSSAIRE

page 1/2

input() : attend la saisie d'une donnée par l'utilisateur, instruction à n'utiliser qu'en phase d'apprentissage

Cette fonction peut être combinée afin de demander un entier : int(input()) ou un flottant ; float(input())

PARTIE 2 : Boucles et instructions conditionnelles

CONDITIONS		
égalité	==	<pre>>>> a=15/5 >>> a==3 >>> a==2 True False</pre>
différent	!=	<pre>>>> a=15/5 >>> a!= 3 >>> a!=2 False True</pre>
comparaison	< ;<= ;> ;>=	<pre>>>> a = 2 >>> a >= 4 False</pre>
et	and	<pre>>>> a=15 >>> a%5==0 and a%7==0 False</pre>
ou	or	<pre>>>> a=15 >>> a%5==0 or a%7==0 True</pre>
INSTRUCTIONS CONDITIONNELLES		
sialorssinon	if condition : instruction	<pre>a = int(input("Donner un nombre : ")) if a%7 == 0 : print(a,"est divisible par 7")</pre>
	if condition : instruction 1 else : instruction 2	<pre>a = int(input("Donner un nombre : ")) if a%7 == 0 : print(a,"est divisible par 7") else : print(a, "n'est pas divisible par 7")</pre>
	if condition 1 : instruction 1 elif condition 2 : instruction 2 elif condition 3 : instruction 3 else : instruction 4	<pre># images par une fonction affine par morceaux a = int(input("Donner un nombre : ")) if a<=2 : print("L'image de ",a,"est : ",2*a+1) elif a>2 and a<=5 : print("L'image de ",a,"est : ",a+3) else : print("L'image de ",a,"est : ",2*a-2)</pre>
BOUCLE NON BORNÉE		
Tant que	while condition : instruction	<pre>h,n = 90,0 #initialise les valeurs de h et n while h >= 1: h = 0.6*h n = n+1 print(n)</pre>
BOUCLE BORNÉE		
pourallant de ... à ...	for i in range(n) : instruction	<pre>Exemple 1: for i in range(5) : print("Hello") Affichage: Hello 7 * 0 = 0 Hello 7 * 1 = 7 Hello 7 * 2 = 14 Hello 7 * 3 = 21 Hello 7 * 4 = 28 Hello 7 * 5 = 35</pre>
	range (n) : répète n fois, n prenant ses valeurs entre 0 et n - 1	<pre>Exemple 2: for i in range(11): print("7 * ",i, " = ",7*i) Affichage: 7 * 6 = 42 7 * 7 = 49 7 * 8 = 56 7 * 9 = 63 7 * 10 = 70</pre>
	for i in range(a,b) : instruction	<pre>Exemple: somme = 0 #initialise somme for i in range (10,21) : somme = somme + i # somme des entiers de 10 à 20 print(somme) Affichage: 165</pre>
	range(a,b) où a et b sont des entiers : répète de i = a à i = b - 1	

Les boucles ne se finissent pas avec un end, c'est l'indentation qui permet de délimiter les boucles.

Pour une version « papier » des scripts on utilisera le symbole ____ pour signifier une indentation.

Délimiteurs particuliers : les messages de textes sont placés entre « »

les commentaires sont placés après un #

PARTIE 3 : Les Fonctions

Syntaxe :

```
def {nom de la fonction} ({liste de paramètres})  
    # cette liste peut être vide  
    instructions  
    return {résultat(s)} # peut renvoyer plusieurs données
```

GLOSSAIRE

page 2/2

EXEMPLE 1 : Fonction à 2 paramètres : Calculer le volume d'un cylindre connaissant les dimensions

```
from math import pi  
def volume_cylindre(h,R):  
    return(pi*R**2*h)
```

Calculer le volume du cylindre de hauteur 5 et de rayon 2,3

```
>>> volume_cylindre(5,2.3)  
83.09512568745001
```

Affichage :

EXEMPLE 2 : Fonction avec un paramètre : Somme des n premiers entiers

```
def somme(n):  
    S=0  
    for k in range(n+1):  
        S=S+k  
    return(S)
```

Tester pour n = 50.
Tester pour les entiers de 50 à 100

```
>>> somme(50)  
1275  
>>> somme(100)-somme(49)
```

Affichage : 3825

Une autre technique pour définir les fonctions : **lambda** : cf **exercices 29** + document d'accompagnement lycée.

POINT INFO : Les bibliothèques :

Pour l'instant nous utiliserons principalement les bibliothèques **math** et **random**.

Pour avoir la liste des fonctions disponibles dans une bibliothèque on peut utiliser : **import math** puis **dir(math)** :

```
>>> import math  
  
>>> dir(math)  
['_doc_', '_loader_', '_name_', '_package_', '_spec_', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'loglp', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'trunc']
```

Pour avoir une aide spécifique sur une fonctionnalité, dès que la bibliothèque est importée, on peut utiliser : **help()**.

```
>>> from math import *  
  
>>> help(log)  
Help on built-in function log in module math:  
  
log(...)  
    log(x[, base])  
  
    Return the logarithm of x to the given base.  
    If the base not specified, returns the natural logarithm (base e) of x.
```

La bibliothèque **random** :

```
>>> from random import * # faire appel à toutes les fonctions disponibles dans  
cette bibliothèque  
  
>>> randint(1,100) # choisir un entier entre 1 et 100, bornes incluses  
78  
  
>>> randrange(2,9,2) # choisir un entier entre 2 et 9 (9 exclu) en ne prenant  
que les nombres de 2 en 2 : 2-4-6-8  
2  
  
>>> random() # choisir un nombre (float) entre 0 et 1  
0.038025689743349966
```

On prendra l'habitude de n'importer que les fonctionnalités nécessaires au script en construction : comme dans l'exemple 1 où seul le nombre π a été importé :

Lorsque le script fait appel à plusieurs bibliothèques ou lorsqu'il y a un risque de doublon entre une fonction définie par l'utilisateur et une fonction prédéfinie on utilise un alias, par exemple la fonction **randint** du module **random** peut être définie en faisant : **import random as rd** puis **rd.randint**.

```
>>> from math import pi  
>>> pi  
3.141592653589793  
  
import random as rd  
print(rd.randint(1,100))  
  
>>>  
75
```