

Syntaxe avancé des fonctions et modules

Les paramètres d'une fonction

- Il est possible de donner une valeur par défaut à un paramètre d'une fonction.
- Un paramètre défini avec une syntaxe

`def fct(param=valeur):`

est appelé paramètre par mot-clé

```
def MaFonction(pays = "Maroc"):
    print("Je suis de : " + pays)
```

```
MaFonction("Egypte")
MaFonction("Espagne")
MaFonction()
```

Affichage:

```
Je suis de : Egypte
Je suis de : Espagne
Je suis de : Maroc
```

Les paramètres d'une fonction

On a la possibilité de donner des valeurs par défaut à plusieurs paramètres:

```
def MaFonction(x=0,y=0):  
    return x+y
```

```
print( MaFonction(35,5) )  
print( MaFonction(20) )  
print( MaFonction() )
```

Affichage:

40
20
0

Il est aussi possible de donner le nom de l'argument lorsqu'on fait l'appel:

```
print( MaFonction(y=18) )  
print( MaFonction(x=13,y=45) )
```

Affichage:

18
58

Les paramètres d'une fonctions

- Il est possible de faire un mélange d'**arguments positionnels** et par mot-clé. Ainsi les **arguments positionnels** doivent toujours être placés avant les arguments par mot-clé :

```
def fct(a, b, x=0, y=0, z=0):  
    return a, b, x, y, z
```

```
print( fct(1, 1))           #appel de la fonction avec a=b=1 et x=y=z=0  
print( fct(1, 1, z=5))      #appel de la fonction avec a=b=1, x=y=0 et z=5  
print( fct(1, 1, z=5, y=32)) #appel de la fonction avec a=b=1, x=0, y=32 et z=5
```

Les fonctions lambda

- La fonction **lambda** est une petite fonction contenant **qu'une seule expression**. Elle peut agir sous anonymat parce qu'elle ne nécessite aucun nom. Elles sont très utiles lorsqu'il faut effectuer des petites tâches avec moins de code.

Les fonctions lambda

- Syntaxe :



Pour appeler cette fonction, on écrit: `print(f(5))` #25

- Équivalente à:

```
def carre(x):  
    return x*x
```

Certaines fonctions utiles

- **abs(x)** : Permet de faire disparaître le signe de x. Par exemple **abs(3.65)** vaut 3.65 et **abs(-4.3)** vaut 4.3
- **round(x,n)** : Donne l'arrondi de x avec n chiffres après la virgule. Par exemple **round(12.345678,3)** vaut 12.345
- **min(a,b)** et **max(a,b)** : Donnent respectivement le plus petit et le plus grand des nombres entre a et b

Notions de modules

- Il existe plusieurs fonctions prédéfinis en python, mais ces fonctions ne sont pas utiliser directement, elles se trouve dans des modules.
- **Un module** en langage python, est un fichier python prédéfinie qui regroupe un ensemble de fonctions prêtes à être utilisées. C'est une sorte de bibliothèque qui regroupe des fonctions.
- Pour utiliser les fonctions définie dans un module , il faut **importer ce module ou importer exactement la fonction souhaitée.**

Utiliser les modules

1. Pour appeler un module dans un fichier, on utilise la syntaxe suivante :
`import nomModule`

Puis pour travailler avec une de ses fonctions, on écrit:
`nomModule.NomFonction`

2. Si on veut préciser une fonction bien déterminé dans l'importation on peut la référer directement comme suit:

`from nomModule import nomFonction`

ou bien:

`from nomModule import *`

Étoile ici (*) signifie **toutes** les fonctions.

Le module mathématique: **math**

- Si on a un programme qui utilise des fonctions mathématiques, il faut d'abord importer le module **math** à l'aide de l'instruction:

import math

- Ce module (ou bibliothèque) est constituée de plusieurs fonctions que vous allez trouver dans le tableau dessous:

Les fonctions du module `math`

Nom de la fonction	Signification	Exemple
<code>ceil(x)</code>	Retourne l'entier immédiatement supérieur au nombre réel x	<code>math.ceil(7.5)</code> -> 8
<code>floor(x)</code>	Retourne l'entier immédiatement inférieur au nombre réel x	<code>math.floor(7.5)</code> -> 7
<code>factorial(x)</code>	Retourne le factoriel d'un entier x	<code>math.factorial(4)</code> -> 24
<code>sqrt(x)</code>	Retourne la racine carrée de x	<code>math.sqrt(16)</code> -> 4
<code>pow(x,y)</code>	Retourne x puissance y	<code>math.pow(5,2)</code> -> 25
<code>log(x)</code>	Retourne le logarithme népérien de x	<code>math.log(1)</code> -> 0
<code>log10()</code>	Retourne le logarithme décimal de x	<code>math.log(10)</code> -> 1
<code>Sin(x)</code>	Retourne le sinus de x	<code>math.sinus(180)</code> -> -0.8011
<code>Cos()</code>	Retourne le cosinus de x	<code>math.cos(180)</code> -> -0.5984
<code>pi</code>	Constante qui représente π <i>mathématique</i>	<code>math.pi</code>

Exemples

- Exemples d'utilisation de fonctions mathématiques:

```
#exemple 1
import math

x=math.log(1)
print(x)

#exemple 2
from math import floor

y=floor(18.123)
print(y)
```

Le module random

- Ce module contient des fonctions relatifs à la génération des nombre aléatoires.

import random

fonction	Signification	Exemple
random()	Retourne une valeur réelle aléatoire entre 0 et 1 exclus	X=random()
randint(x,y)	Retourne un entier entre x et y	X=randint(5,25)
choice(L)	Retourne une valeur aléatoire d'une liste L passée en paramètre.	L=[15,4,8,2,30] X=choice(L)
shuffle(L)	Permet de permuter d'une manière aléatoire les éléments d'une liste L	L=[15,4,8,2,30] Shuffle(L)