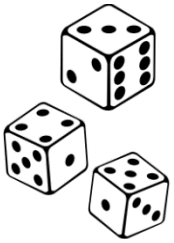
Le module random



1. Le module random

random, adjectif :
Concerne toute chose qui est effectuée au hasard, de manière arbitraire. Caractérise un événement accidentel, imprévisible.

Rappel : Les modules sont des programmes Python qui contiennent des fonctions que l'on est amené à réutiliser souvent (on les appelle aussi bibliothèques ou *libraries*). Les développeurs de Python ont mis au point de nombreux modules qui effectuent une quantité phénoménale de tâches.

Le module random de Python donne accès aux nombres aléatoires et permet des opérations "stochastiques.

Exemple :

```
import random
random.randint(0,10)
4
```

Regardons de plus près cet exemple :

- L'instruction import permet d'accéder à toutes les fonctions du module random
- Ensuite, nous utilisons la fonction (ou méthode) randint(a,b) du module random. Attention cette fonction renvoie un nombre entier aléatoirement entre a inclus et b inclus (contrairement à range() par exemple). Remarquez la notation objet random.randint() où la fonction randint() peut être considérée comme une méthode de l'objet random.

Il existe un autre moyen d'importer une ou des fonctions d'un module :

```
from random import randint
randint(0,10)
7
```

À l'aide du mot-clé from, vous pouvez importer une fonction spécifique d'un module donné. Remarquez que dans ce cas il est inutile de répéter le nom du module, seul le nom de ladite fonction est requis.

ADOPTONS PYTHON

On peut également importer toutes les fonctions d'un module :

```
from random import *
x = [1, 2, 3, 4]
shuffle(x)
x
[2, 3, 1, 4]
shuffle(x)
x
[4, 2, 1, 3]
randint(0,50)
46
uniform(0,2.5)
0.64943174760727951
```

Comme vous l'avez deviné, l'instruction from random import * importe toutes les fonctions du module random. On peut ainsi utiliser toutes ses fonctions directement, comme par exemple shuffle(), qui permute une liste aléatoirement.

Dans la pratique, plutôt que de charger toutes les fonctions d'un module en une seule fois (from random import *), nous vous conseillons de charger le module (import random) puis d'appeler explicitement les fonctions voulues (random.randint(0,2)).

Enfin, si vous voulez vider de la mémoire un module déjà chargé, vous pouvez utiliser l'instruction del :

```
import random
random.randint(0,10)
2
del random
random.randint(0,10)
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
NameError: name 'random' is not defined
```

Vous pouvez constater qu'un rappel d'une fonction du module random après l'avoir vidé de la mémoire retourne un message d'erreur.

Enfin, il est également possible de définir un alias (un nom plus court) pour un module :

```
import random as rand
rand.randint(1, 10)
6
```

Dans cet exemple, les fonctions du module random sont accessibles via l'alias rand.

2. Obtenir de l'aide sur les modules importés

Pour obtenir de l'aide sur un module rien de plus simple, il suffit d'invoquer la commande help() :

```
import random
help(random)
...
```

On peut se promener dans l'aide avec les flèches ou les touches page-up et page-down (comme dans les commandes Unix man, more ou less). Il est aussi possible d'invoquer de l'aide sur une fonction particulière d'un module de la manière suivante help(random.randint).

La commande help() est en fait une commande plus générale permettant d'avoir de l'aide sur n'importe quel objet chargé en mémoire.

```
x = range(2)
help(x)
Help on list object:

class list(object)
 |  list() -> new list
 |      list(sequence) -> new list initialized from sequence's items
 |
 |  Methods defined here:
 |
 |  __add__(...)
 |      x.__add__(y) <==> x+y
 |
...
```

Enfin, si on veut connaître en seul coup d'œil toutes les méthodes ou variables associées à un objet, on peut utiliser la commande dir :

```
import random
dir(random)
['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random', 'SG_MAGICCONST',
'SystemRandom',
'TWOPI', 'WichmannHill', '_BuiltinMethodType', '_MethodType', '__all__',
'__builtins__',
'__doc__', '__file__', '__name__', '_acos', '_ceil', '_cos', '_e', '_exp', '_hexlify',
'_inst', '_log', '_pi', '_random', '_sin', '_sqrt', '_test', '_test_generator',
'_urandom', '_warn', 'betavariate', 'choice', 'expovariate', 'gammavariate',
'gauss',
'getrandbits', 'getstate', 'jumpahead', 'lognormvariate', 'normalvariate',
'paretovariate', 'randint', 'random', 'randrange', 'sample', 'seed', 'setstate',
'shuffle', 'uniform', 'vonmisesvariate', 'weibullvariate']
>>>
```

Exercice 1 : importer et utiliser les fonctions du module random

instruction à saisir	**résultat obtenu** **+ commentaire ou explication**
from random import random	
random( )	
random( )	
random( )	
random(4)	
from random import randrange	
randrange(3) (répéter plusieurs fois cette instruction)	
randrange(100,200) (répéter plusieurs fois cette instruction)	
randrange(0,1001,100) (répéter plusieurs fois cette instruction)	
help(random)	
help(randrange)	
uniform(2,3)	
import random	
random.uniform(2,3) (répéter plusieurs fois cette instruction)	
help(random)	
help(random.uniform)	
help(random.random)	
help(random.randrange)	
random()	
random.random()	

ADOPTONS PYTHON

maListe=["alpha","bravo","charlie","delta","oscar","papa","romeo","tango","yankee","zulu"]	
choice(maListe)	
random.choice(maListe) (répéter plusieurs fois cette instruction)	
random.shuffle(maListe)	
maListe	
random.choice(range(10)) (répéter plusieurs fois cette instruction)	
random.sample(maListe,3) (répéter plusieurs fois cette instruction)	
random.sample(range(10),2) (répéter plusieurs fois cette instruction)	
random.gauss(0,1) (répéter plusieurs fois cette instruction)	

En résumé : Les principales fonctions du module random sont :

random( ) renvoie un nombre flottant aléatoire entre 0 et 1.

randrange(p,n,pas) renvoie un nombre entier aléatoire entre p inclus et n exclu, tous les pas.
uniform(a,b) renvoie un nombre flottant aléatoire entre a et b.

choice(maListe) renvoie un élément choisi aléatoirement dans la liste maListe.

sample(maListe,k) renvoie un échantillon de k éléments choisis aléatoirement dans la liste maListe. (cela fonctionne aussi avec un range).

shuffle(maListe) mélange aléatoirement les éléments de maListe (cette fonction modifie la liste en la mélangeant).

triangular(a,b) renvoie un nombre flottant entre a et b choisi aléatoirement selon la loi triangulaire.

gauss(m,σ) renvoie un nombre flottant choisi aléatoirement selon la loi normale de moyenne m et d'écart-type σ.

seed() permet d'initialiser le générateur de nombres aléatoires[7] à partir de l'heure interne du système d'exploitation de l'ordinateur ou de la valeur qu'on lui donne en paramètre.