Cours 10 :
Utilisation de modules sous python

MPSI - Lycée Thiers

2014/2015

Importer un module

• Un module est un fichier ayant pour extension `.py` contenant des définitions de constantes et fonctions. Tout programmeur `python` peut réaliser un module. Importer un module permet d'utiliser ses constantes et fonctions.

Nous avons déjà utilisé plusieurs modules : `math, fractions, random, time`.

• Pour importer un module :

1 Première méthode : A l'aide de l'instruction `from`
Pour importer une fonction ou constante :

```
>>> from math import sqrt
>>> sqrt(2)
1.4142135623730951
```

```
>>> from math import cos, pi
>>> cos(pi/4)
0.7071067811865476
```

Pour importer toute la bibliothèque :

```
>>> from math import *
>>> sin(pi/4)
0.7071067811865476
```

Importer un module

• Un module est un fichier ayant pour extension `.py` contenant des définitions de classes, constantes et fonctions. Tout programmeur `python` peut réaliser un module. Importer un module permet d'utiliser ses objets.

Nous avons déjà utilisé plusieurs modules : `math, fractions, random, time`.

• Pour importer un module :

2 Deuxième méthode : Sans l'instruction `from`. Méthode conseillée.

```
>>> import math
>>> math.sqrt(2)
1.4142135623730951
>>> math.cos(math.pi/4)
0.7071067811865476
```

Fonctions et constantes doivent être précédées d'un préfixe : le nom du module suivie d'un point '.'.

On peut définir un alias à l'aide de l'instruction as :

```
>>> import math as mth          # Création d'un alias mth
>>> mth.sqrt(2)              # Utiliser le préfixe mth. devant un objet de math
1.4142135623730951
>>> mth.cos(mth.pi/4)
0.7071067811865476
```

Exemple : le module random

Voici quelques fonctions fournies par le module `random` :

`randrange(a,b,k)`	Choisit un entier aléatoirement dans `range(a,b,k)`
`randint(a,b)`	Choisit un entier aléatoirement dans $[[a, b]]$
`choice(List)`	Choisit un entier aléatoirement dans la liste `List`
`random()`	Choisit un float aléatoirement dans $[0, 1[$
`uniform(a,b)`	Choisit un float aléatoirement dans $[a, b[$

Ici aléatoirement signifie selon une loi uniforme (quasiment).

Exemple : Que fait le programme suivant ?

```
import random as rand
def de6(n):
    tirs = [0, 0, 0, 0, 0, 0]
    for i in range(n):
        t = rand.randint(1,6)
        tirs[t-1] += 1

    for j in range(6):
        tirs[j] = tirs[j]*100.0/n
        tirs[j] = '{:.2f}'.format(tirs[j]) + '%'      # 2 nbres apres la virgule
    return tirs
```

A l'aide de la méthode `format` pour le type `str`,l'expression `'{:.2f}'.format(x)`, pour un float `x`, retourne une chaîne de caractères représentant l'écriture décimale du nombre `x` avec 2 chiffres après la virgule.

Exemple : le module random

Réponse : le programme simule `n` lancers d'un dé 6, et compte le nombre de fois où chaque face apparaît pour finalement retourner le pourcentage d'apparition de chaque face.

Maintenant vérifions le théorème des grands nombres : pour un grand nombre de tirs les fréquences d'apparition de chaque face devraient tendre vers leur probabilité de tir, ici donc être égales (la loi est uniforme : le dé est non pipé).

```
>>> de6(10)
['30.00%', '0.00%', '20.00%', '20.00%', '10.00%', '20.00%']

>>> de6(100)
['10.00%', '22.00%', '19.00%', '20.00%', '15.00%', '14.00%']

>>> de6(1000)
['16.00%', '15.40%', '16.10%', '16.90%', '18.50%', '17.10%']

>>> de6(10000)
['16.45%', '16.66%', '16.50%', '16.84%', '17.37%', '16.18%']

>>> de6(100000)
['16.57%', '16.67%', '16.80%', '16.62%', '16.66%', '16.68%']

>>> de6(1000000)
['16.65%', '16.69%', '16.66%', '16.64%', '16.63%', '16.74%']

>>> de6(10000000)
['16.66%', '16.66%', '16.65%', '16.66%', '16.69%', '16.68%']
```

Exemple : Méthode de Monte Carlo

• Les méthodes de simulation de Monte Carlo permettent le calcul approché de valeurs numériques par des procédés aléatoires.
Elles sont particulièrement employées pour le calcul approché d'intégrales en dimension > 1 (pour le calcul d'aire, de volume).

• Le calcul approché d'intégrales de fonctions réelles d'une variable repose sur le résultat :

Pour n suffisamment grand, et pour $i \mapsto x_i$ une variable aléatoire suivant une loi uniforme sur l'intervalle $[a, b]$ (c'est à dire n points tirés au hasard dans $[a, b]$ selon une loi uniforme)), alors avec une probabilité proche de 1 :

$$\int_a^b f(t)dt \approx \frac{b-a}{n} \sum_{i=1}^{n} f(x_i)$$

Exercice : écrire une fonction `montecarlo(f,a,b,n)` qui retourne une valeur approchée de l'intégrale de f sur $[a, b]$ selon la méthode de Monte Carlo et avec n points.

Réponse :

```
import random as rand
def montecarlo(f,a,b,n):    # Calcul approchée de l'integrale de f sur [a,b]
    s = 0
    for i in range(n):
        x = rand.uniform(a,b)
        s += f(x)
        s = s*(b-a)/n
    return s
```

Exemple : Méthode de Monte Carlo

Par exemple avec $f : x \mapsto x^2$ sur $[0,1]$ le résultat devrait être proche de $\frac{1}{3}$:

```
>>> f = lambda x:x**2
>>> montecarlo(f,0,1,100)
0.32889436948779177

>>> montecarlo(f,0,1,1000)
0.3400354481596589
>>> montecarlo(f,0,1,1000)
0.31687077247647943

>>> montecarlo(f,0,1,10000)
0.3347958934090399
>>> montecarlo(f,0,1,100000)
0.3328616254606106
```

Exercice : En déduire un calcul approché de π par la méthode de Monte Carlo.

Avec $f : x \mapsto \sqrt{1-x^2}$ l'intégrale sur $[0,1]$ vaut $\frac{\pi}{4}$ (aire d'un quart du disque unitaire).

```
>>> f = lambda x : 4*(1-x**2)**0.5
>>> montecarlo(f,0,1,100000)
3.1427991454635724
>>> montecarlo(f,0,1,10000000)
3.1413319841070737
>>> pi
3.141592653589793
```

Illustration : Approximation de π

• Encore une approximation de π par une méthode de type Monte-Carlo, mais avec illustration graphique.

• On tire au sort N points dans le domaine carré $[-1;1]\times[-1;1]$ selon une loi uniforme.

La probabilité pour un point d'être choisi dans le disque unitaire est égale au quotient de l'aire du disque unitaire par l'aire du domaine carré,
soit :

$$\boxed{\frac{\pi}{4}}$$

Ainsi d'après le théorème des grands nombres, si l'on tire un grand nombre de points, c. à d. si N est suffisamment grand, la proportion des points tirés se trouvant dans le disque unitaire, c. à d. avec $x^2+y^2\leqslant 1$, est presque surement proche de $\pi/4$.

• **Algorithme :**

```
Choisir N points (x,y) dans le domaine [-1,1]×[-1,1] selon une loi uniforme.
Calculer la proportion de points choisis (x,y) vérifiant x²+y²⩽1.
En multipliant par 4 ce résultat on obtient une approximation de π.
```

```
import matplotlib.pyplot as plt        # pyplot
import numpy as np                     # numpy
import random as rand                  # random
#

def monpi(N):
    X = [rand.uniform(-1,1) for i in range(N)]        # choix aléatoire
    Y = [rand.uniform(-1,1) for i in range(N)]

    oui = non = 0                          # Calcul de la proportion dans le disque
    for i in range(N):
        if X[i]**2 + Y[i]**2 <= 1:
            oui += 1
        else:
            non += 1

    print('approximation de pi trouvée :')            # impression du résultat
    print(4 * oui / (oui+non))
    print(np.pi)

    # Tracé du graphique
    plt.plot([-1,1,1,-1,-1],[-1,-1,1,1,-1], 'b-')    # tracé du carré

    T = np.linspace(0,2*np.pi,256)
    Xdisk = np.cos(T)
    Ydisk = np.sin(T)
    plt.plot(Xdisk,Ydisk,'b')                    # tracé du disque

    for i in range(N):                        # tracé du nuage de point
    if X[i]**2 + Y[i]**2 <= 1:
        plt.plot([X[i]],[Y[i]], 'g+')          # vert si dedans
```

```
>>> monpi(10000)
approximation de pi trouvée :
3.152
3.14159265359
```