Python-ECG1-02-bibliothèques-cor

September 7, 2022

1 P2 - Utiliser une bibliothèque

1.1 1- La bibliothèque numpy

La bibliothèque numpy sera la biliothèque que nous allons utiliser le plus fréquemment.

Elle contient notamment **toutes les fonctions usuelles**.

Pour charger la bibliothèque, on écrit ceci :

[1]:
```
import numpy as np
# import nom_bibliothèque as alias
```

Fonctions usuelles

Fonction	Syntaxe Python
exp	np.exp(...)
ln	np.log(...)
racine carrée	np.sqrt(...)
valeur absolue	np.abs(...)
partie entière	np.floor(...)

[2]:
```
# Exemple : calcul de exp(3)
np.exp(3)
```

[2]:
```
20.085536923187668
```

Constantes usuelles

Fonction	Syntaxe Python
nombre e	np.e
nombre pi	np.pi

[3]:
```
# Pour calcul le nombre e on écrit :
np.e
```

[3]:
```
2.718281828459045
```

[4]:
```
# On peut aussi passer par exp(1)
np.exp(1)
```

[4]:
```
2.718281828459045
```

[5]:
```
np.pi
```

[5]:
```
3.141592653589793
```

1.1.1 Exercice 1

Réaliser les calculs suivants :

$$\frac{\sqrt{5+e}}{\pi}$$

$$\ln(|1-\exp(6)|)$$

$$\left\lfloor \frac{17\pi}{5} \right\rfloor$$

$$\frac{e^5+2}{5\times 10^6}$$

[6]:
```
np.sqrt(5+np.e)/np.pi
```

[6]:
```
0.8843220287568424
```

[7]:
```
np.log(np.abs(1-np.exp(6)))
```

[7]:
```
5.99751817063104
```

[8]:
```
np.floor(17*np.pi/5)
```

[8]:
```
10.0
```

[9]:
```
(np.e**5+2)/(5*10**6)
```

[9]:
```
3.0082631820515314e-05
```

1.1.2 Exercice 2

Réaliser les calculs suivants et commenter les résultats.

$$(\sqrt{2})^2 - 2$$

$$3 \times 0,1 - 0,3$$

[10]:
```
np.sqrt(2)**2 - 2
```

[10]:
```
4.440892098500626e-16
```

[11]:
```
3*0.1 - 0.3
```

[11]:
```
5.551115123125783e-17
```

Ces résultats sont dûs à des manipulations de valeurs approchées. En binaire, 0.1 n'est pas un nombre décimal (son écriture comporte un nombre infini de chiffres après la virgule) donc python utilise une valeur approchée.

1.1.3 Exercice 3

Écrire un programme affichant les solutions de l'équation $ax^2 + bx + c = 0$ **dans le cas** $\Delta > 0$ **uniquement** et utilisant la fonction racine carrée.

Tester avec les trois équations suivantes, qu'il faudra résoudre à la main en amont :

$$2x^2 - 10x + 8 = 0, \quad 3x^2 + 5x - 2 = 0, \quad x^2 - x - 1 = 0.$$

[12]:
```
a = 2
b = -10
c = 8

Delta = b**2 - 4*a*c
x1 = (-b-np.sqrt(Delta))/(2*a)
x2 = (-b+np.sqrt(Delta))/(2*a)

print('Les solutions sont', x1, 'et', x2)
```

```
Les solutions sont 1.0 et 4.0
```

[13]:
```
a = 3
b = 5
c = -2

Delta = b**2 - 4*a*c
x1 = (-b-np.sqrt(Delta))/(2*a)
x2 = (-b+np.sqrt(Delta))/(2*a)
```

```
print('Les solutions sont', x1, 'et', x2)
```

```
Les solutions sont -2.0 et 0.3333333333333333
```

[14]:
```
a = 1
b = -1
c = -1

Delta = b**2 - 4*a*c
x1 = (-b-np.sqrt(Delta))/(2*a)
x2 = (-b+np.sqrt(Delta))/(2*a)

print('Les solutions sont', x1, 'et', x2)
```

```
Les solutions sont -0.6180339887498949 et 1.618033988749895
```

1.2 2 - Bibliothèque numpy.random

La bibliothèque **numpy.random** est une bibliothèque faisant partie de numpy et permettant de simuler des phénomènes **aléatoires**.

On commence par charger la bibliothèque avec un alias.

[15]:
```
import numpy.random as rd
```

Aujourd'hui nous allons voir comment générer un nombre aléatoire de l'intervalle $]0,1[$ et un entier aléatoire.

- **rd.random()** : renvoie un nombre aléatoire de $]0,1[$
- **rd.randint(a,b)** : renvoie un entier aléatoire de $[a,b[$. Attention, b est exclu.

[16]:
```
# Exécutez plusieurs fois pour observer l'aléatoire
rd.random()
```

[16]:
```
0.04811898453632568
```

1.2.1 Exercice 4

1) Générer un entier aléatoire compris entre 5 et 10

[17]:
```
rd.randint(5,11)
```

[17]:
```
6
```

2) Simuler le lancer d'un dé équilibré à 6 faces.

[18]:
```
rd.randint(1,7)
```

[18]: 6

3) Simuler le lancer d'un dé équilibré à 16 faces.

[19]:
```
rd.randint(1,17)
```

[19]: 3

1.3 3 - Pour finir

[20]:
```
# À exécuter si vous l'osez....
import antigravity
```