Calcul de valeurs moyennes et d'écart-type

Rappelons le programme officiel :

5. Probabilité – statistiques	
Variable aléatoire.	Utiliser les fonctions de base des bibliothèques **random** et/ou **numpy** (leurs spécifications étant fournies) pour réaliser des tirages d'une variable aléatoire. Utiliser la fonction **hist** de la bibliothèque **matplotlib.pyplot** (sa spécification étant fournie) pour représenter les résultats d'un ensemble de tirages d'une variable aléatoire. Déterminer la moyenne et l'écart-type d'un ensemble de tirages d'une variable aléatoire.

I. Liste de valeurs expérimentales

1) Calcul de la moyenne et de l'écart-type

On importe la bibliothèque numpy avec l'alias np. On appellera alors les différentes fonctions de cette bibliothèque avec le préfixe np. (avec un point séparant np du nom de la fonction).

On entre les valeurs expérimentales de x sous la forme d'une colonne de valeurs avec numpy.

On peut calculer la moyenne avec la fonction average (ou mean)et l'écart-type avec std (dans numpy).

On calcule l'incertitude-type – c'est l'écart-type divisé par $\sqrt{nombre\ de\ valeurs\ de\ x}$.

```
# On importe les bibliothèques.
import numpy as np
from math import sqrt          #  On importe la racine carrée (sqrt) du module math

# On donne la liste des valeurs de x:
x= np.array([1.1, 0.9, 0.95, 1.05])

# On calcule la moyenne la pente, l'écart-type et l'incertitude-type :
xmoy =np.average( x )          # Moyenne des valeurs de x
ex = np.std( x )               # écart- type sur la pente
ux = ex / sqrt( len (x) )      # on calcule l'incertitude-type en divisant n'écart-type par le nombre de valeurs de la liste x

print("La moyenne des valeurs de x est : ", xmoy, "unité à préciser")
print("L'incertitude-type vaut : " , ux, " unité ")
```

On écrit l'incertitude-type avec 2 chiffres significatifs. C'est elle qui permet de déterminer quelle est la précision à donner à la valeur moyenne. La moyenne ne sera pas plus précise que l'incertitude-type.

2) Histogramme

Tracer un histogramme est facile avec la fonction hist de matplotlib.pyplot :
On importe la fonction hist, on écrit la liste des valeurs à considérer (sous la forme d'une liste, ou d'une colonne de valeurs, comme on veut), on trace l'histogramme avec plt.hist.

- on précise le nom de la liste de valeurs à représenter ;

- on précise les bornes des valeurs en abscisses (ce peut être des valeurs fixées à la main, ou alors on prendre la valeur minimale de la liste x – que l'on écrit min(x) , ou encore la valeur maximale de la liste – que l'on écrit max(x) ;
- on précise le nombre d'intervalles de cet histogramme avec : bins = nombre désiré ;
- on précise la couleur avec color = 'la couleur voulue' (optionnel)

Comme pour tout graphe, on peut préciser ce que l'on a en abscisses avec plt.xlabel, ce que l'on a en ordonnées avec plt.ylabel, préciser le titre avec plt.title.

Exemple :

```
# J'importe la bibliothèque matplotlib.pyplot
import matplotlib.pyplot as plt
import numpy as np

x = [1, 1, 1.2, 1.3, 1.3, 2, 1.5, 1.5, 1.5]              # Liste de valeurs
plt.hist (x, range = ( min(x) , max(x) ), bins = 5, color = 'green')          # L'histogramme utilisera les valeurs de la liste x ; il sera tracé pour des valeurs comprises entre la valeur minimale de la liste précédente et la valeur maximale, il y aura 5 intervalles, et les colonnes seront en vert.
plt.xlabel('x')
plt.ylabel("Nombre d'occurrences")
plt.show()
```

II. Simulation d'un tirage aléatoire (simulation de Monte-Carlo)

Supposons qu'une grandeur m dépende des variables x_1 et x_2, selon $m = f(x_1 , x_2)$. On suppose que l'on connaît x_1 et x_2 avec une certaine précision – soit x_{01} l'estimation du mesurande x_1 avec l'incertitude-type u_{x1}, et x_{02} l'estimation du mesurande x_2 avec l'incertitude-type u_{x2}.

On calcule $m = f(x_1 , x_2)$ pour un très grand nombre de couples (x_1 , x_2) dont les valeurs respectives sont prises aléatoirement dans les intervalles $[x_i - \sqrt{3}\,u(x_i)\,;\; x_i + \sqrt{3}\,u(x_i)]$ avec la fonction random.uniform ou la fonction random.normal de la bibliothèque numpy .

random.uniform (-1, 1, N) renvoie une colonne de N valeurs tirées aléatoirement selon une loi uniforme (c'est-à-dire équiprobable) entre -1 et 1.

random.normal (xi, u(xi), N) renvoie une colonne de N valeurs tirées aléatoirement selon une loi normale, c'est-à-dire gaussienne, centrées sur xi avec l'incertitude-type u(xi).

Pour les tirages aléatoires, l'incertitude-type correspond à l'écart-type.

```
# On importe la bibliothèque numpy.
import numpy as np
from math import sqrt

# On entre les données (à adapter).
x1 = ..
ux1 = ..
x2 = ..
ux2 = ..
N =100000

# On crée une colonne de valeurs aléatoires de x1 et de x2 avec une loi uniforme :
x1MC = x1 + ux1 * sqrt(3) * np.random.uniform(-1, 1, N)
x2MC = x2 + ux2 * sqrt(3) * np.random.uniform(-1, 1, N)
```

```
# Variante : On crée une colonne de valeurs aléatoires de x1 et de x2 avec une loi normale :
x1MC = np.random.normal ( x1, ux1, N)
x2MC = np.random.normal ( x2, ux2, N)

# On calcule les N valeurs de m correspondantes:
m = f( x1MC , x2MC)

# On fait l'étude statistique.
m_moy = np.average(m)          # calcul de la moyenne
ec = np.std(m)                 # calcul de l'écart-type
print ("m_moy = ", m_moy, 'unité')
print ("ec = ", ec, "unité"

# Affichage du résultat
print ("La valeur moyenne de x est de ", round( m_moy, 2), "unité")      # round permet d'arrondir avec un nombre de décimales donné.
print ("L'incertitude-type est de ", round( ec, 2), 'unité')
```

On peut compléter avec l'histogramme des valeurs de m obtenues :

```
import matplotlib.pyplot as plt
plt.hist(m, range = ( min(m), max(m)), bins = 50)          # adapter le nombre de colonnes (bins)
plt.xlabel("Valeurs de m")
plt.ylabel("Occurrences")
plt.show()
```