**Exercice 1.**
**a)** Fonction `moyenne()` ; il suffit de sommer les éléments de la liste, avant de diviser par le nombre d'éléments.

```
## Exo 1.a)
def moyenne(X):
    S = 0
    for k in range(len(X)):
        S += X[k]
    return S/len(X)
```

ou encore, en utilisant le fait que les listes sont **itérables** :

```
def moyenne(X):
    S = 0
    for x in X:
        S += x
    return S/len(X)
```

**b)** Vérifions que lorsque le nombre d'éléments augmente, la moyenne a tendance à se rapprocher de 5 :

```
## Exo 1.b)
from random import random

X1 = [10 * random() for k in range(10) ]
X2 = [10 * random() for k in range(100) ]
X3 = [10 * random() for k in range(1000) ]
X4 = [10 * random() for k in range(10000) ]
X5 = [10 * random() for k in range(100000) ]
print(len(X1),"elements : moyenne =", moyenne(X1))
print(len(X2),"elements : moyenne =", moyenne(X2))
print(len(X3),"elements : moyenne =", moyenne(X3))
print(len(X4),"elements : moyenne =", moyenne(X4))
print(len(X5),"elements : moyenne =", moyenne(X5))
```

ou mieux, à l'aide d'une boucle **for** :

```
from random import random
for k in range(1,6):
    X = [10 * random() for boucle in range(10**k) ]
    print(len(X),"elements : moyenne =", moyenne(X))
```

ce qui produit :

```
In [2]: (executing lines 11 to 22 of "Untitled.py")
10 elements : moyenne = 4.486206047389738
100 elements : moyenne = 4.949375651731299
1000 elements : moyenne = 5.133744153051188
10000 elements : moyenne = 5.03306948994417
100000 elements : moyenne = 5.002069502162295
```

**Exercice 2. a)** Vérifions que la moyenne de $Y = X - \overline{X}$ est nulle :

```
# 2.a)
for k in range(1,6):
    X = [10 * random() for boucle in range(10**k) ]
    m = moyenne(X)
    Y = [ x-m for x in X ]
    print("Moyenne de Y =", moyenne(Y))
```

On obtient :

```
In [4]: (executing lines 25 to 31 of "Untitled.py")
Moyenne de Y = 3.552713678800501e-16
Moyenne de Y = -4.574118861455645e-16
Moyenne de Y = 2.6014745913016667e-15
Moyenne de Y = 3.3521452280638185e-14
Moyenne de Y = 2.2066717342283937e-14
```

La moyenne obtenue n'est pas nulle, mais très proche de l'être (de l'ordre de $10^{-15}$) ; en fait mathématiquement elles sont toutes nulles, et ces valeurs non nulles ne sont dues qu'aux erreurs d'arrondi dans les calculs de moyenne.

**A cause des limitations dans la représentation des nombres à virgule dans la mémoire de l'ordinateur, souvent une valeur calculée de l'ordre de $10^{-15}$ correspond à un résultat qui sans erreurs d'approximation serait égal à zéro.**

**b)** Démonstration mathématique que :

$$\boxed{\overline{(X - \overline{X})} = 0}$$

Soit $n$ l'effectif de la série statistique $X = (x_i)_{1 \leq i \leq n}$ et $\overline{X} = \dfrac{1}{n}\sum_{k=1}^{n} x_i$ sa moyenne.

$$\begin{aligned}
\overline{(X - \overline{X})} &= \frac{1}{n}\sum_{k=1}^{n}(x_i - \overline{X}) \\
&= \frac{1}{n}\sum_{k=1}^{n} x_i - \frac{1}{n}\sum_{k=1}^{n}\overline{X} \\
&= \overline{X} - \frac{1}{n} \times n \times \overline{X} \\
&= 0
\end{aligned}$$

**c)** Fonction `variance()` ; le mieux est d'utiliser la fonction `moyenne()`.
Première solution, en n'appelant qu'une fois la fonction `moyenne()` :

```
def variance(X):
    m = moyenne(X)
    S = 0
    for x in X:
        S += (x-m)**2
    return S/len(X)
```

Deuxième solution, en appelant deux fois la fonction `moyenne()` :

```
def variance(X):
    m = moyenne(X)
    X2 = [(x-m)**2 for x in X]
    return moyenne(X2)
```

**d)** Vérifions que la variance est toujours positive :

```
for k in range(1,6):
    X = [10 * random() for boucle in range(10**k) ]
    print("variance de X =", variance(X))
```

ce qui produit :

```
In [8]: (executing lines 34 to 49 of "Untitled.py")
variance de X = 9.81157149640175
variance de X = 9.572585725856692
variance de X = 8.562261782201826
variance de X = 8.390590663430379
variance de X = 8.335409443213269
```

La variance se rapproche de $\frac{10^2}{12} \approx 8,333..$ à mesure que le nombre d'éléments augmente.

**Exercice 3.** Fonction `minimum()` et `maximum()` :

À l'aide d'une parcours de la liste par indice :

```
def minimum(X):
    m = X[0]
    for k in range(1,len(X)):
        if X[k] < m:
            m = X[k]
    return m
```

```
def maximum(X):
    M = X[0]
    for k in range(1,len(X)):
        if X[k] > M:
            M = X[k]
    return M
```

À l'aide d'une parcours de la liste par élément :

```
def minimum(X):
    m = X[0]
    for x in X:
        if x < m:
            m = x
    return m

def maximum(X):
    M = X[0]
    for x in X:
        if x > M:
            M = x
    return M
```

**Exercice 3.**
**a)** Fonction `covariance()` :
Première méthode : en définissant $(X-\overline{X})(Y-\overline{Y})$ par compréhension de liste :

```
def covariance(X,Y):
    mX = moyenne(X)
    mY = moyenne(Y)
    XY = [ (X[k]-mX)*(Y[k]-mY) for k in range(len(X)) ]
    return moyenne(XY)
```

Deuxième méthode : en calculant la somme $\sum(x_k-\overline{X})(y_k-\overline{Y})$.

```
def covariance(X,Y):
    mX = moyenne(X)      # Xbar
    mY = moyenne(Y)      # Ybar
    n = len(X)
    S = 0        # somme (x_k - Xbar) * (y_k - Ybar)
    for k in range(n):
        S = S + (X[k]-mX)*(Y[k]-mY)
    return S/n
```

**b)** Fonction `droiteRegression()` ; elle appelle les fonctions `covariance()`, `moyenne()` et `variance()` :

```
def droiteRegression(X,Y):
    a = covariance(X,Y)/variance(X)
    b = moyenne(Y) - a*moyenne(X)
    return [a,b]
```

**c)** Fonction `coefCorrelation()` ; elle nécessite d'écrire une fonction renvoyant l'écart-type d'une série statistique :

```
def ecartType(X):
    return variance(X)**0.5

def coefCorrelation(X,Y):
    return covariance(X,Y)/(ecartType(X)*ecartType(Y))
```

**Exercice 4.**

**a)** La réaction chimique est d'ordre 1 si et seulement si :

$$\begin{aligned} & C(t) = C_0 \exp(-\lambda t) \\ \iff & \ln(C(t)) = \ln(C_0) - \lambda t \\ \iff & Y(t) = -\lambda t + \ln(C_0) \end{aligned}$$

**b)** La réaction chimique est d'ordre 2 si et seulement si :

$$\begin{aligned} & C(t) = \frac{C_0}{1+\lambda C_0 t} \\ \iff & \frac{1}{C(t)} = \frac{1+\lambda C_0 t}{C_0} \\ \iff & Z(t) = \frac{1}{C_0} + \lambda t \end{aligned}$$

**c)** Déclaration des listes `T` et `C` :

```
T = [0,3,5,7,10,12,15,18,20,23,25]
C = [34.97,17.52,13.23,10.13,8.16,6.55,5.78,5.22,4.66,
     4.15,3.86]
```

**d)** Déclaration des listes `Y` et `Z` :

```
from math import log
Y = [ log(x) for x in C ]
Z = [ 1/x for x in C ]
```

**e)** On compare les coefficients de corrélation de $(T,Y)$ et de $(T,Z)$

```
print("Ordre 1 :",coefCorrelation(T,Y))
print("Ordre 2 :",coefCorrelation(T,Z))
```

On obtient :

```
In [22]: (executing lines 2 to 42 of "<tmp 2>")
Ordre 1 : -0.9465919256586861
Ordre 2 : 0.9983509845709143
```

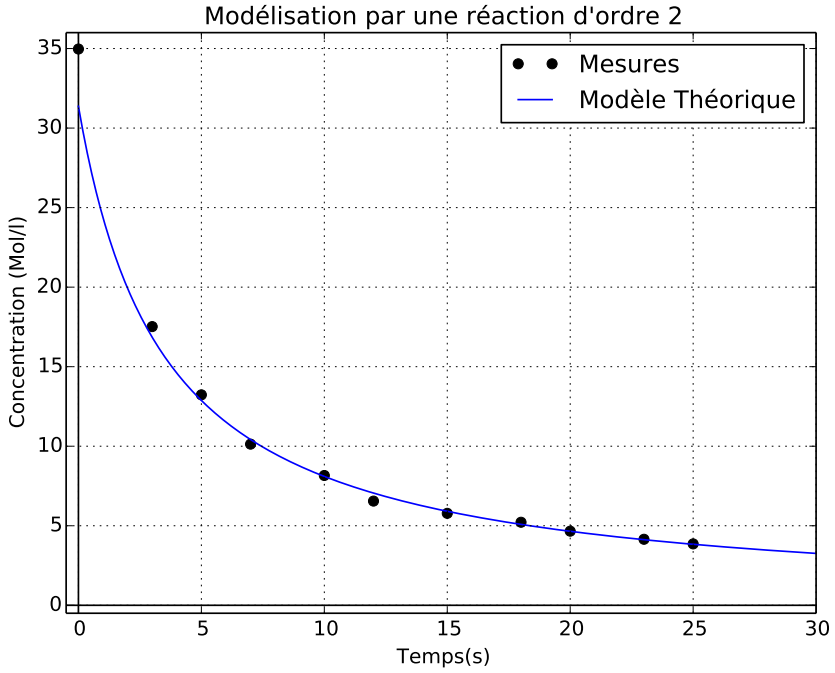
Le coefficient de corrélation est bien meilleur pour l'ordre 2, il est très proche de 1. On choisit donc cette modélisation.

**f)** Ainsi, avec $(a,b)$ les coefficients de la droite de régression linéaire $y = a \cdot x + b$ calculés :

$$\frac{1}{C(t)} = a \cdot t + b \quad \iff C(t) = \frac{1}{a \cdot t + b}$$

Tracé graphique :

```
# Calculs nécessaires
a,b = droiteRegression(T,Z)
tMin, tMax = 0, 30
N = 1000
X = [tMin + (tMax-tMin)*k/(N-1) for k in range(N)]
Y = [ 1/(a*x+b) for x in X ]

# Tracé
from matplotlib.pyplot import *
figure(1)
axhline(color='black')
axvline(color='black')
grid()
axis([-0.5,30,-0.5,36]) # Fenêtre [xMin,xMax,yMin,Ymax]
plot(T,C,marker='o',linestyle='None',color='black',
                                        label='Mesures')
plot(X,Y,label="Modèle Théorique")
legend()
title("Modélisation par une réaction d'ordre 2")
xlabel("Temps(s)")
ylabel("Concentration (Mol/l)")
show()
```

On obtient la figure graphique :