Statistique – Visualisation de données

C'est bien de savoir calculer le minimum, le maximum, la moyenne, les quartiles d'une série. C'est encore mieux de les visualiser tous sur un même graphique !

Vidéo ■ Statistique - Visualisation de données - partie 1
Vidéo ■ Statistique - Visualisation de données - partie 2
Vidéo ■ Statistique - Visualisation de données - partie 3
Vidéo ■ Graphiques avec tkinter
Vidéo ■ Boutons avec tkinter

Activité 1 (Statistique de base).

Objectifs : calculer les principales caractéristiques d'une série de données : minimum, maximum, moyenne et écart-type.

Dans cette activité `liste` désigne une liste de nombres (entiers ou flottants).

1. Écris ta propre fonction `somme(liste)` qui calcule la somme des éléments d'une liste donnée. Compare ton résultat avec la fonction `sum()` décrite ci-dessous qui existe déjà en Python. En particulier pour une liste vide vérifie que ton résultat est bien 0.

python : `sum()`

Usage : `sum(liste)`
Entrée : une liste de nombres
Sortie : un nombre

Exemple : `sum([4,8,3])` renvoie 15

Tu peux maintenant utiliser la fonction `sum()` *dans tes programmes !*

2. Écris une fonction `moyenne(liste)` qui calcule la moyenne des éléments d'une liste donnée (et renvoie 0 si la liste est vide).
3. Écris ta propre fonction `minimum(liste)` qui renvoie la plus petite valeur des éléments d'une liste donnée. Compare ton résultat avec la fonction Python `min()` décrite ci-dessous (qui en plus sait calculer le minimum de deux nombres).

python : `min()`

Usage : `min(liste)` ou `min(a,b)`
Entrée : une liste de nombres ou bien deux nombres
Sortie : un nombre

Exemple :
- `min(12,7)` renvoie 7
- `min([10,5,9,12])` renvoie 5

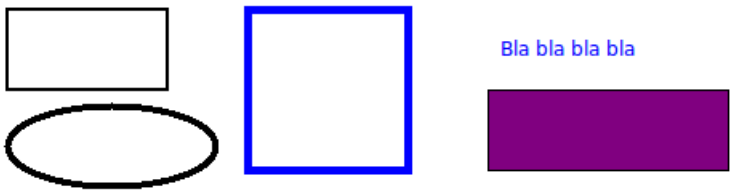
Tu peux maintenant utiliser la fonction `min()`*, et aussi bien sûr la fonction* `max()` *dans tes programmes !*

4. La ***variance*** d'une série de données $(x_1, x_2, \dots, x_n)$ est définie comme la moyenne des carrés des écarts à la moyenne. C'est-à-dire :
$$v = \frac{1}{n}\big((x_1-m)^2 + (x_2-m)^2 + \cdots + (x_n-m)^2\big)$$
où m est la moyenne de $(x_1, x_2, \dots, x_n)$.
Écris une fonction `variance(liste)` qui calcule la variance des éléments d'une liste.
Par exemple, pour la série $(6, 8, 2, 10)$, la moyenne est $m = 6.5$, la variance est
$$v = \frac{1}{4}\big((6-6.5)^2 + (8-6.5)^2 + (2-6.5)^2 + (10-6.5)^2\big) = 8.75.$$

5. L'***écart-type*** d'une série $(x_1, x_2, \dots, x_n)$ est la racine carrée de la variance :
$$e = \sqrt{v}$$
où v est la variance. Programme une fonction `ecart_type(liste)`. Avec l'exemple ci-dessus on trouve $e = \sqrt{v} = \sqrt{8.75} = 2.95\dots$

6. Voici les températures mensuelles moyennes à Brest et à Strasbourg.
```
temp_brest = [6.4,6.5,8.5,9.7,11.9,14.6,15.9,16.3,15.1,12.2,9.2,7.1]
temp_strasbourg = [0.9,2.4,6.1,9.7,13.8,17.2,19.2,18.6,15.7,10.7,5.3,2.1]
```
Calcule la température moyenne sur l'année à Brest puis à Strasbourg. Calcule l'écart-type des températures à Brest puis à Strasbourg. Quelles conclusions en tires-tu ?

Cours 1 (Graphiques avec tkinter)**.**
Pour afficher ceci :



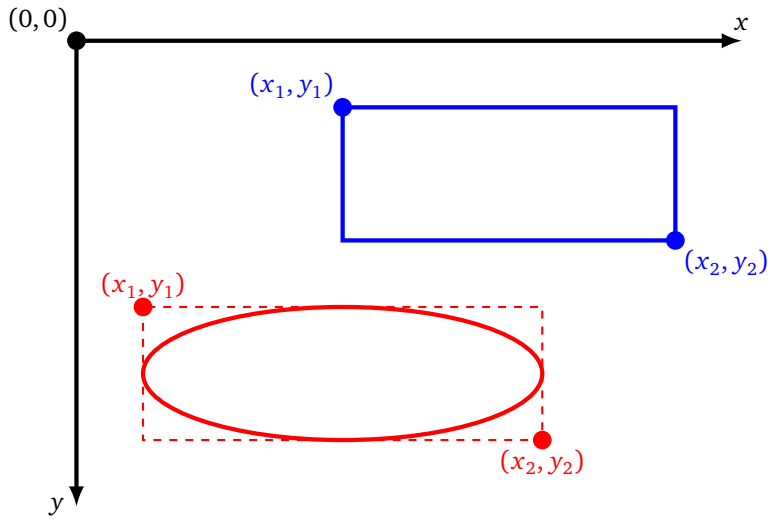
Le code est :

```
# Module tkinter
from tkinter import *
```

```
# Fenêtre tkinter
root = Tk()

canvas = Canvas(root, width=800, height=600, background="white")
canvas.pack(fill="both", expand=True)

# Un rectangle
canvas.create_rectangle(50,50,150,100,width=2)

# Un rectangle à gros bords bleus
canvas.create_rectangle(200,50,300,150,width=5,outline="blue")

# Un rectangle rempli de violet
canvas.create_rectangle(350,100,500,150,fill="purple")

# Un ovale
canvas.create_oval(50,110,180,160,width=4)

# Du texte
canvas.create_text(400,75,text="Bla bla bla bla",fill="blue")

# Ouverture de la fenêtre
root.mainloop()
```

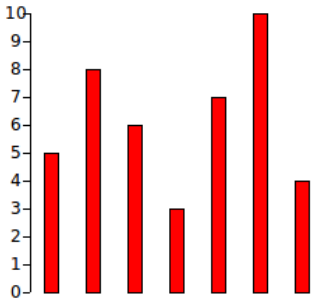
Quelques explications :

- Le module `tkinter` nous permet de définir des variables `root` et `canvas` qui définissent une fenêtre graphique (ici de largeur 800 et de hauteur 600 pixels). On décrit ensuite tout ce que l'on veut ajouter dans la fenêtre. Et enfin la fenêtre est affichée par la commande `root.mainloop()` (tout à la fin).
- Attention ! Le repère graphique de la fenêtre a son axe des ordonnées dirigé vers le bas. L'origine $(0,0)$ est le coin en haut à gauche (voir la figure ci-dessous).
- Commande pour tracer un rectangle : `create_rectangle(x1,y1,x2,y2)` ; il suffit de préciser les coordonnées (x_1, y_1), (x_2, y_2) de deux sommets opposés. L'option `width` ajuste l'épaisseur du trait, `outline` définit la couleur de ce trait, `fill` définit la couleur de remplissage.
- Une ellipse est tracée par la commande `create_oval(x1,y1,x2,y2)`, où (x_1, y_1), (x_2, y_2) sont les coordonnées de deux sommets opposés d'un rectangle encadrant l'ellipse voulue (voir la figure). On obtient un cercle lorsque le rectangle correspondant est un carré !
- Du texte est affiché par la commande `canvas.create_text(x,y,text="Mon texte")` en précisant les coordonnées (x, y) du point à partir duquel on souhaite afficher le texte.

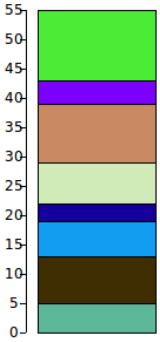


Activité 2 (Graphiques)**.**

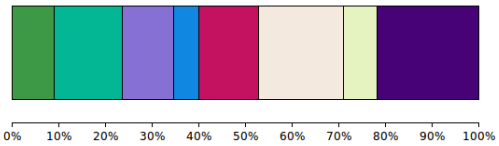
Objectifs : visualiser des données par différents types de graphiques.



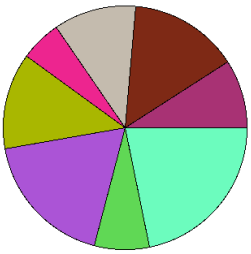
Graphique en barres



Graphique cumulatif



Graphique en pourcentage



Graphique en secteurs

1. **Graphique en barres.** Écris une fonction `graphique_barres(liste)` qui affiche les valeurs d'une liste sous la forme de barres verticales.

   Indications :

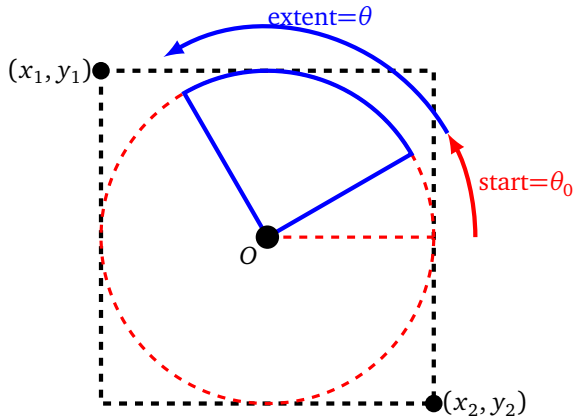
   - Dans un premier temps, ne t'occupe pas de tracer l'axe vertical des coordonnées avec les indications chiffrées.
   - Tu peux définir une variable `echelle` qui permet d'agrandir tes rectangles, afin qu'ils aient une taille adaptée à l'écran.
   - Si tu souhaites tester ton graphique avec une liste au hasard voici comment construire une liste aléatoire de 10 entiers compris entre 1 et 20 :

```
from random import *
liste = [randint(1,20) for i in range(10)]
```

2. **Graphique cumulatif.** Écris une fonction `graphique_cumulatif(liste)` qui affiche les valeurs d'une liste sous la forme de rectangles les uns au-dessus des autres.
3. **Graphique en pourcentage.** Écris une fonction `graphique_pourcentage(liste)` qui affiche les valeurs d'une liste sous la forme d'un rectangle horizontal de taille fixe (par exemple 500 pixels) et qui est divisé en sous-rectangles représentant les valeurs.
4. **Graphique en secteurs.** Écris une fonction `graphique_secteurs(liste)` qui affiche les valeurs d'une liste sous la forme d'un disque de taille fixe et divisé en secteurs représentant les valeurs.

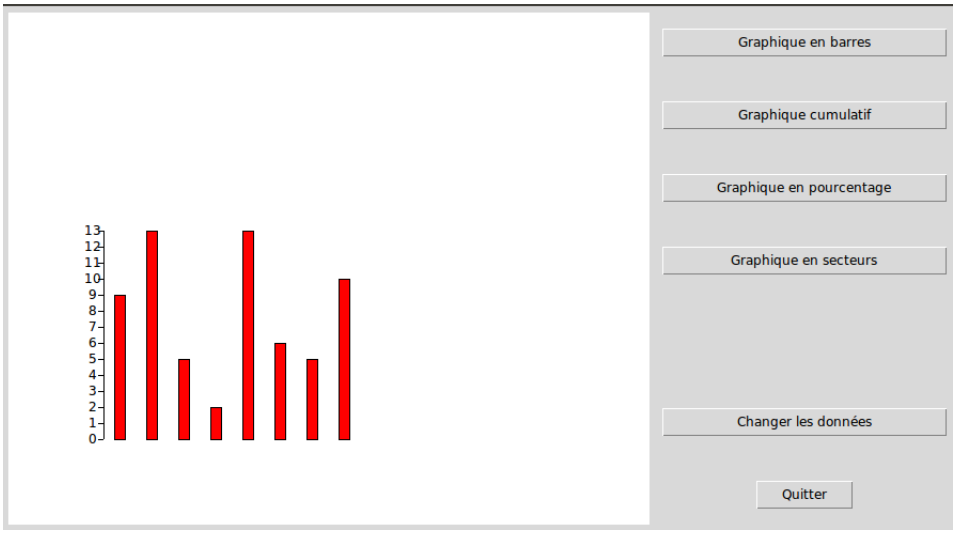
   La fonction `create_arc()`, qui permet de dessiner des arcs de cercles, n'est pas très intuitive. Il faut penser que l'on dessine un cercle, en précisant les coordonnées des coins d'un carré qui l'entoure, puis en précisant l'angle de début et l'angle du secteur (en degrés).

   ```
   canvas.create_arc(x1,y1,x2,y2,start=debut_angle,extent=mon_angle)
   ```

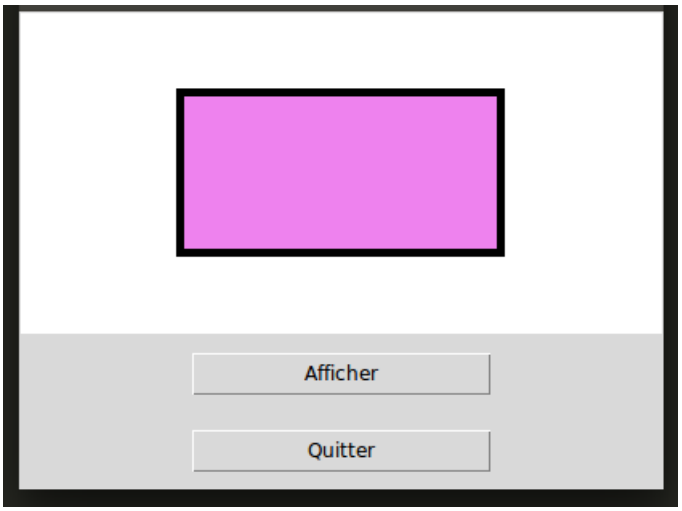


   L'option `style=PIESLICE` affiche un secteur au lieu d'un arc.

5. **Bonus.** Rassemble ton travail en un programme qui laisse la possibilité à l'utilisateur de choisir le diagramme qu'il souhaite en cliquant sur des boutons, et aussi la possibilité d'obtenir une nouvelle série aléatoire de données. *Pour afficher et gérer les boutons avec* `tkinter`*, vois le cours ci-dessous.*



Cours 2 (Boutons avec tkinter)**.**
Il est plus ergonomique d'afficher des fenêtres où les actions sont exécutées en cliquant sur des boutons. Voici la fenêtre d'un petit programme avec deux boutons. Le premier bouton change la couleur du rectangle, le second termine le programme.



Le code est :

```
from tkinter import *
from random import *

root = Tk()
canvas = Canvas(root, width=400, height=200, background="white")
canvas.pack(fill="both", expand=True)

def action_bouton():
    canvas.delete("all")          # Efface tout
    couleurs = ["red","orange","yellow","green","cyan","blue","violet","purple"]
    coul = choice(couleurs)       # Couleur au hasard
    canvas.create_rectangle(100,50,300,150,width=5,fill=coul)
    return

bouton_couleur = Button(root,text="Afficher", width=20, command=action_bouton)
bouton_couleur.pack()

bouton_quitter = Button(root,text="Quitter", width=20, command=root.quit)
bouton_quitter.pack()

root.mainloop()
```

Quelques explications :

- On crée un bouton par la commande `Button`. L'option `text` personnalise le texte qui s'affiche sur le bouton. On ajoute le bouton créé à la fenêtre par la méthode `pack`.
- Le plus important est l'action associée au bouton ! C'est l'option `command` qui reçoit le nom de la fonction à exécuter lorsque le bouton est cliqué. Pour notre exemple `command=action_bouton`, associe au clic sur le bouton un changement de couleur.

- Attention ! il faut donner le nom de la fonction sans parenthèses : `commande=ma_fonction` et pas `command=ma_fonction()`.
- Pour associer au bouton « Quitter » la fermeture du programme, l'argument est `command=root.quit`.
- La commande `canvas.delete("all")` efface tous les dessins de notre fenêtre graphique.

Activité 3 (Médiane et quartiles)**.**

Objectifs : calculer la médiane et les quartiles d'un effectif.

1. Écris une fonction `mediane(liste)` qui calcule la valeur médiane des éléments d'une liste donnée. Par définition, la moitié des valeurs est inférieure ou égale à la médiane, l'autre moitié est supérieure ou égale à la médiane.

   Rappels. On note n la longueur de la liste et on suppose que la liste est ordonnée (du plus petit au plus grand élément).
   - **Cas n impair.** La médiane est la valeur de la liste au rang $\frac{n-1}{2}$. Exemple avec `liste = [12,12,14,15,19]` :
     — la longueur de la liste est $n = 5$ (les indices vont de 0 à 4),
     — l'indice du milieu est l'indice 2,
     — la médiane est la valeur `liste[2]`, c'est donc 14.
   - **Cas n pair.** La médiane est la moyenne entre la valeur de la liste au rang $\frac{n}{2} - 1$ et au rang $\frac{n}{2}$. Exemple avec `liste = [13,14,19,20]` :
     — la longueur de la liste est $n = 4$ (les indices vont de 0 à 3),
     — les indices du milieu sont 1 et 2,
     — la médiane est la moyenne entre `liste[1]` et `liste[2]`, c'est donc $\frac{14+19}{2} = 16.5$.
2. Les résultats d'une classe sont collectés sous la forme suivante d'un effectif par note :

   `effectif_notes = [0,0,0,0,0,1,0,2,0,1,5,1,2,3,2,4,1,2,0,1,0]`

   Le rang i va de 0 à 20. Et la valeur au rang i indique le nombre d'élèves ayant eu la note i. Par exemple ici, 1 élève a eu la note 5, 2 élèves ont eu la note 7,..., 5 élèves ont obtenus 10, ... Écris une fonction `notes_vers_liste(effectif_notes)` qui prend en entrée un effectif de notes et renvoie la liste des notes. Pour notre exemple la fonction doit renvoyer `[5,7,7,9,10,10,10,10,10,10,...]`.
   Déduis-en une fonction qui calcule la médiane des notes d'une classe à partir d'un effectif par note.
3. Écris une fonction `calcule_quartiles(liste)` qui calcule les quartiles Q_1, Q_2, Q_3 des éléments d'une liste donnée. Les quartiles répartissent les valeurs en : un quart en-dessous de Q_1, un quart entre Q_1 et Q_2, un quart entre Q_2 et Q_3, un quart au-dessus de Q_3. Pour le calcul, on utilisera que :
   - Q_2 est simplement la médiane de la liste entière (supposée ordonnée),
   - Q_1 est la médiane de la sous-liste formée de la première moitié des valeurs,
   - Q_3 est la médiane de la sous-liste formée de la seconde moitié des valeurs.

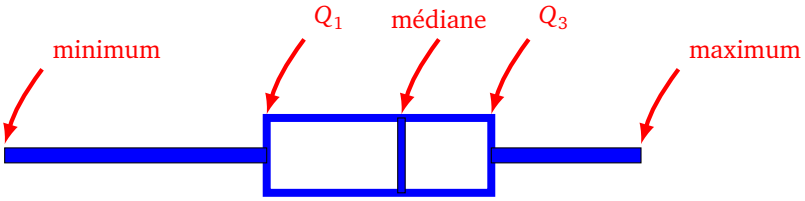
Pour l'implémentation, il faut une nouvelle fois discuter selon que la longueur n de la liste est paire ou pas.
Déduis-en une fonction qui calcule les quartiles des notes d'une classe à partir d'un effectif par note.

Activité 4 (Diagramme en boîte)**.**

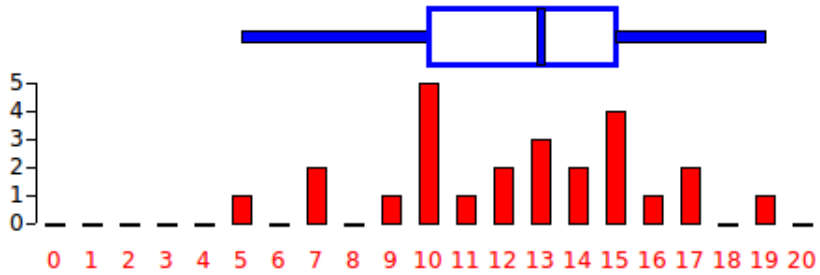
Objectifs : tracer des diagrammes en boîte.

Un ***diagramme en boîte*** (appelé aussi ***boîte à moustaches***) est un graphique qui représente les principales caractéristiques d'une série statistique : minimum, maximum, médiane et quartiles. Le schéma de

principe est le suivant :



Écris une fonction `diagramme_boite(effectif_notes)` qui trace le diagramme en boîte des notes d'une classe à partir d'un effectif par note (voir l'activité précédente).



Activité 5 (Moyenne mobile)**.**

Objectifs : calculer des moyennes mobiles afin de « lisser » des courbes.

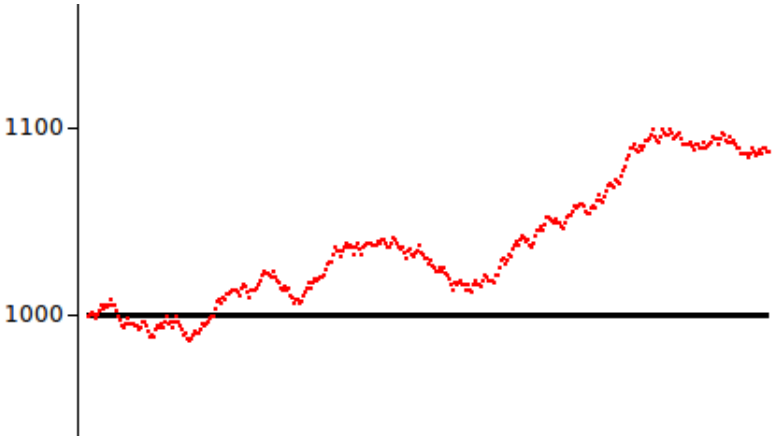
1. Simule le cours de la bourse de l'indice *Top 40* sur 365 jours. Au jour $j = 0$, l'indice vaut 1000. Ensuite l'indice d'un jour est déterminé en ajoutant une valeur au hasard (positive ou négative) à la valeur de l'indice de la veille :

   $$\text{indice du jour } j = \text{indice du jour } (j-1) \ + \ \text{valeur au hasard}$$

   Pour cette valeur au hasard, tu peux essayer une formule du style :

   ```
   valeur = randint(-10,12)/3
   ```

   Écris une fonction `cours_bourse()`, sans paramètre, qui renvoie une liste de 365 valeurs de l'indice *Top 40* selon cette méthode.
2. Trace point par point la courbe du cours sur une année. (Pour tracer un point, tu peux afficher un carré de taille 1 pixel.)



3. *Comme la courbe du cours journalier est très chaotique, nous souhaitons la lisser afin de la rendre plus lisible. Pour cela nous calculons des moyennes mobiles.*

   La moyenne mobile à 7 jours pour le jour j, est la moyenne des 7 derniers cours. Par exemple : la moyenne mobile (à 7 jours) pour le jour $j = 7$ est la moyenne des cours des jours $j = 1, 2, 3, 4, 5, 6, 7$. On peut changer la durée : par exemple la moyenne mobile à 30 jours est la moyenne des 30 derniers cours.

   Écris une fonction `moyenne_mobile(liste,duree)` qui renvoie la liste de toutes les moyennes mobiles d'une série de données, pour une durée fixée.

4. Trace point par point sur un même graphique : la courbe du cours sur une année (en rouge ci-dessous), la courbe de ses moyennes mobiles à 7 jours (en bleu ci-dessous) et la courbe des ses moyennes mobiles à 30 jours (en marron ci-dessous). Note que plus la durée est longue plus la courbe est « lisse ». (Bien sûr la courbe des moyennes mobiles à 30 jours ne commence qu'à partir du trentième jour.)