

## TP 3 - Une application complète en PyQt

© B. Besserer, R. Péteri, S. Bourbia

Année universitaire 2025-2026

### 1 Cahier des charges

Le but du TP est d'écrire une application complète sous PyQt permettant d'afficher graphiquement un histogramme (Figure 1) à partir de valeurs contenues dans un fichier texte, que le programme devra ouvrir et lire. Il aura aussi la possibilité de *serialiser* les objets pour les sauvegarder.

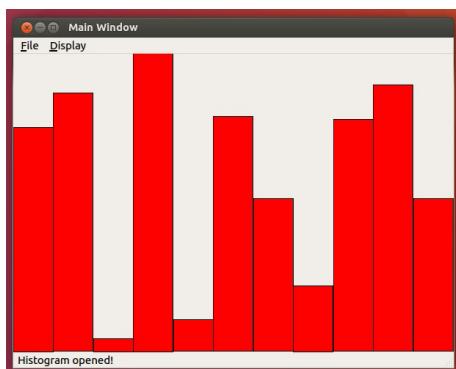


FIGURE 1 – Aspect final de l'application Histogramme

Un code source partiel du projet (`TP3_PyQt_base.py`), est à récupérer sous Moodle. Il contient une classe héritée de `QMainWindow`, qui sera la fenêtre principale de notre application et qui permettra de définir des menus ou une barre d'états (`Status bar`). Remarque pour ceux qui sont sous macOS : le menu `File` n'apparaît pas tant qu'il n'y a pas d'items associés (`QAction`).

Le code fourni contient aussi la classe `myHisto` qui permet la gestion de l'histogramme. La classe disposera d'un attribut capable de stocker les valeurs de l'histogramme : `m_list` qui est **une liste d'entiers**, d'un attribut `m_size` qui est le nombre de bins (de barres) de l'histogramme, et d'un attribut `m_max` qui est sa valeur maximum.

Dans la suite, la taille de l'histogramme sera fixé à 10 bins (c'est à dire 10 "rectangles").

1. Après avoir testé le programme fourni, ajouter dans le menu `File` de la fenêtre principale de l'application les items : `Open`, `Save`, `Restore`, `Bye` avec des raccourcis clavier. On placera aussi entre les items `Restore` et `Bye` un séparateur (méthode `addSeparator()` de la classe `QMenu`)

2. Ajouter aussi un menu **Display** avec les items **Clear** et **Color** qui serviront dans la suite du TP.
3. Pour chaque item de menu (**QAction**), créer les slots associées (**Open**, **Save**,...) et leurs connexions. La barre d'état (status bar) devra afficher respectivement "*Histogram opened !*", "*Histogram saved !*", "*Histogram cleared !*" et "*Histogram restored !*" lors de la sélection des items de menu précédents.
4. Implémenter aussi le slot associé à l'item '**Bye**' pour sortir de l'application.

## 2 Chargement de l'histogramme

### Chargement de l'histogramme par fichier

On souhaite écrire la méthode pour charger l'histogramme à partir du fichier disponible sous Moodle **histovalues.dat**. Ce fichier comprend 10 lignes, le numéro de ligne étant l'indice du bin, et le chiffre étant la valeur du bin correspondant.

Le fichier doit pouvoir être ouvert en passant par le menu **File**, et une boîte de dialogue standard de sélection de fichier. La classe de la boîte de dialogue pour la sélection du fichier est : **QFileDialog**. Lors de sa création, les fichiers seront filtrés sur l'extension **.dat** et elle devra afficher par défaut le répertoire contenant votre fichier **histovalues.dat**. Vous rechercherez les autres informations nécessaires pour l'usage de cette boîte de dialogue (n'hésitez pas à regarder le cours!).

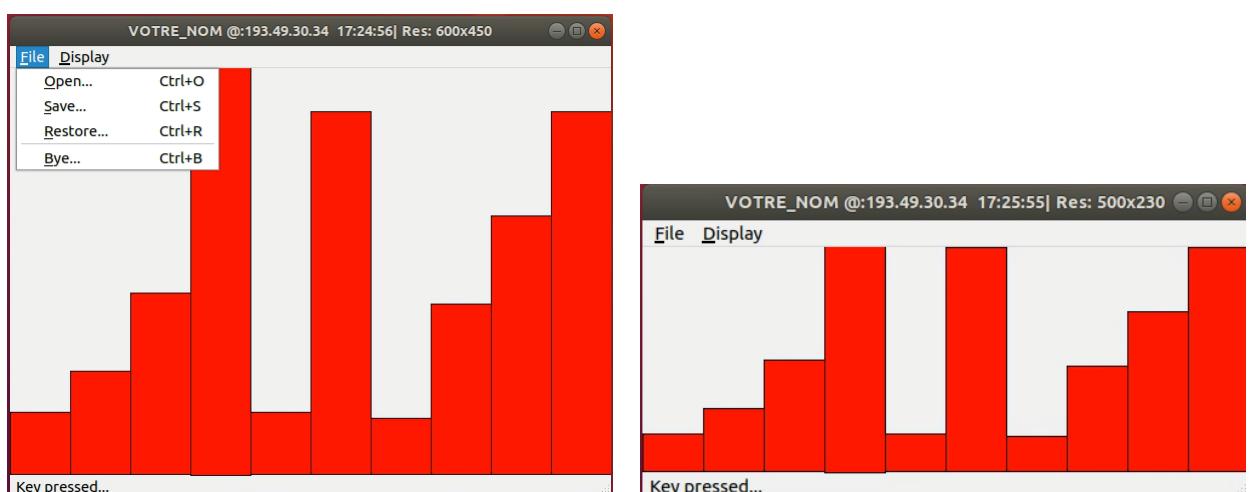
Lorsque vous aurez récupéré le chemin du fichier à ouvrir, il faut bien sûr décoder le fichier pour remplir les attributs de la classe **myHisto** pour l'affichage. On utilisera donc la classe **QFileDialog** pour ouvrir une boîte de dialogue permettant de sélectionner le fichier à charger, la classe **QFile** pour lire un fichier, et enfin **readLine()** pour lire dans le fichier ligne par ligne.

Au niveau du tracé, l'histogramme devra s'adapter aux redimensionnements de la fenêtre (abscisse et ordonnée- attention aux erreurs d'arrondi, convertir en **int** avant les fonctions de dessin !), aura des couleurs de contour et de remplissage comme sur la Figure 1.



*Pour valider, il faudra montrer à l'enseignant :*

- la fenêtre après chargement et tracé de l'histogramme, puis clic pour ouvrir sur le menu **File**
- un redimensionnement (taille approximativement  $500 \times 230$ ) pour montrer que le tracé s'adapter bien aux dimensions de la fenêtre principale.



## Chargement de l'histogramme par Drag & Drop

On peut activer des fonctionnalités de Drag & Drop pour les widgets. Dans notre cas, en glissant et en déposant le fichier de données de l'histogramme, le résultat doit être similaire à l'ouverture de même fichier par l'intermédiaire du menu file→open.

Pour cela, vous devez surcharger les méthodes :  
`dropEvent(self, event)` et `dragEnterEvent(self, event)`  
Dans votre fichier, cela ressemblera à :

```
def dragEnterEvent(self, event):
    if event.mimeData().hasUrls():
        event.accept()
    else:
        event.ignore()

def dropEvent(self, event):
    for url in event.mimeData().urls():
        filename= url.toLocalFile()
        # puis lire le fichier et affecter à la variable membre
        ...
```

Il faudra aussi ajouter dans le constructeur de la classe `MyMainWindow: setAcceptDrops(true)` Implémenter donc l'ouverture par Drag&Drop. Dans un premier temps, avant la lecture du fichier, nous vous conseillons d'afficher dans la barre d'état le chemin récupéré par Drag & Drop de ce fichier.

## 3 Remise à zéro de l'histogramme et création par valeurs aléatoire

1. Implémenter la fonction pour que l'histogramme soit remis à zero lors de l'appui sur l'item de menu `Clear`.
2. L'histogramme pourra aussi être initialisé lors de l'appui sur la touche 'R' par tirage aléatoire d'une valeur entre 0 et 99 (voir module `random`). Implémentez cette fonctionnalité.

*Pour validez ce point, faites une démonstration des fonctionnalités implémentées.*



2

## 4 Sérialisation binaire des données de l'histogramme

1. En utilisant la fonction `dump` du module `pickle`, créer la fonction de sérialisation permettant de sauvegarder la donnée membre `m_list` de la classe `Histogramme` dans le fichier 'saveHisto.bin' lors du choix de l'item `Save` du menu.
2. Créer la fonction de désérialisation permettant de restaurer les données de l'histogramme contenues dans le fichier 'saveHisto.bin' lors du choix de l'item `Restore` du menu.

*Pour validez ce point, faites une démonstration des fonctionnalités implémentées.*



3

## 5 Changement de couleur du tracé de l'histogramme

On va maintenant sélectionner la couleur du tracé et l'afficher à côté de l'article de menu **Display→Color** (voir Fig. 2) :



FIGURE 2 – Après changement de couleur du tracé

Une boîte de dialogue de sélection de couleur doit s'ouvrir lors d'un clic sur l'article de menu "Color". Une fois la couleur choisie récupérée dans la donnée membre `colorIcon`, on l'utiliser pour peindre la `QIcon` associée la `QAction` de l'article de menu "Color" : `self.colorAct.setIcon(QIcon(self.colorIcon))`

1. Modifiez votre code pour que la couleur de tracé initiale soit affiché dans l'icone associée à l'article de menu "Color".
2. Compléter pour que le tracé et l'affichage de l'histogramme soit effectué dans la couleur choisie, ainsi que l'icône associée à l'article de menu "Color" (Fig. 2).



4 Valider en montrant à l'enseignant votre programme reproduisant la figure 2

## 6 Tracé "Camembert" ...

Créer un nouveau menu **Draw** comprenant les items **Bar** et **Pie**. La selection de l'item **Bar** affichera l'histogramme comme précédemment, tandis que la sélection de l'item **Pie** l'affichera en mode "Camembert". La fenêtre principale sera alors de taille  $500 \times 500$  et le camembert sera centré (voir figure 3).

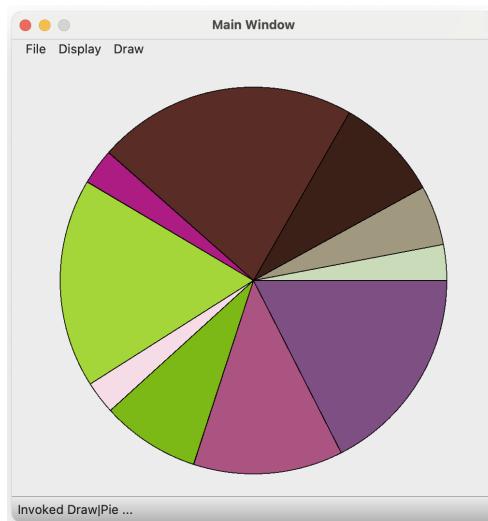


FIGURE 3 – Représentation camembert de l'histogramme

Chaque portion du camembert sera proportionnelle à la population d'un bin par rapport à la population totale de l'histogramme.

Pour ce faire :

- Vous créerez une méthode `total()` dans la classe `histogramme` qui retournera la population totale de l'histogramme.
- On calculera pour chaque secteur les angles représentant chaque bin de l'histogramme, et qui sera un multiple de `360/self.mHist.total()`.
- Chaque secteur du camembert devra être affiché dans une couleur aléatoire par l'instruction :  
`qp.setBrush(QColor(QRandomGenerator.global_().generate()%256,  
QRandomGenerator.global_().generate()%256,  
QRandomGenerator.global_().generate()%256))`
- L'affichage d'une portion de camembert se fera avec la méthode `drawPie(...)` de la classe `QPainter`.

Vous vérifieriez que les possibilités de Drag&Drop ou de tirage aléatoires fonctionnent toujours en mode camembert (mais il n'y a pas de raisons pour que cela ne soit pas le cas - c'est le paradigme modèle/vue).

*Pour valider, il faudra montrer à l'enseignant votre programme reproduisant la fig. 3, aux couleurs près évidemment !*

