Résumé

▪ Introduction :

Qt est une bibliothèque **GUI multiplateforme** :

- **GUI** (Graphic User Interface) : outil de création d'interfaces graphiques.
- **Multiplateforme** : fonctionne sous Windows, Linux et MacOS.

Qt est écrit à l'origine en C++. Mais, il est disponible dans les autres langages grâce à une technique appelé le : **binding** (l'utilisation d'une bibliothèque logicielle dans un autre langage de programmation que celui avec lequel elle a été écrite).

Il existe sous deux déclinaisons : **PySide** et **PyQt**.

Nous allons utiliser **PyQt** comme outil pour créer des interfaces graphiques pour nos programmes écrits en python.

Remarque :

Python supporte beaucoup d'autres **GUI**, outre **PyQt**, à titre d'exemple : **tkinter, wxPython, Kivy**

Qt a été choisie car il s'agit de la bibliothèque **GUI** la plus populaire en termes d'utilisation.

▪ Pourquoi PyQt ?

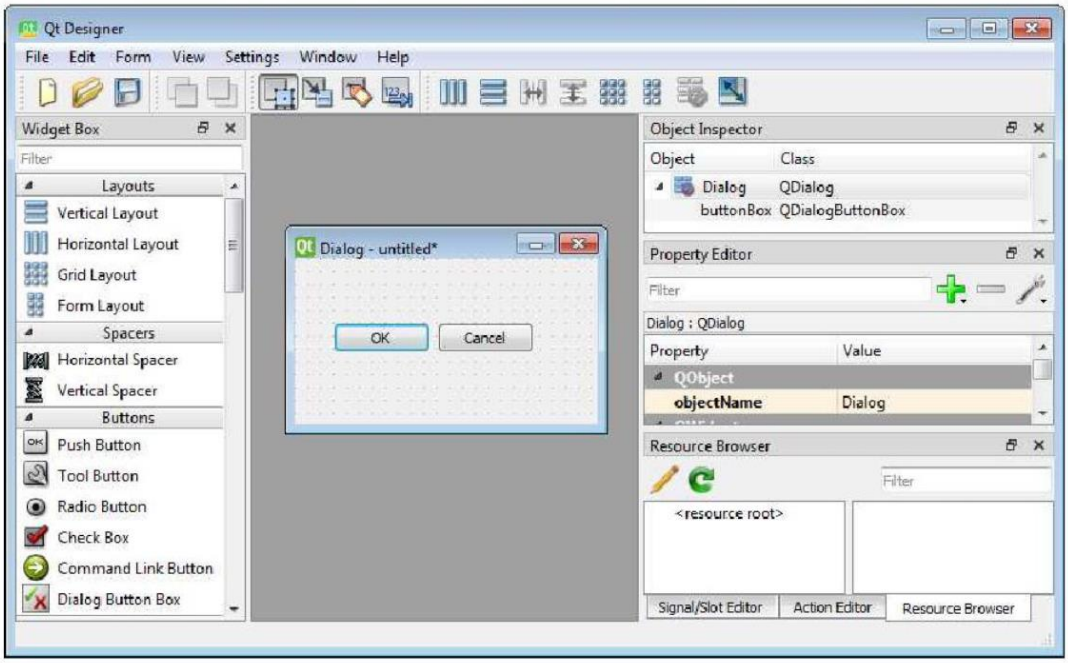
PyQt possède l'avantage d'offrir deux modes de création des interfaces graphiques :

- **L'interface est créée, codée et programmée intégralement en python : très difficile et non convenable pour les débutants,**
- **L'interface est créée et codée à l'aide d'un outil graphique. Elle est programmée ensuite en python : plus facile pour les débutants,**

Nous adopterons cette deuxième méthode dans la suite.

L'interface sera créée à l'aide de **QtDesigner**, cette interface sera exploitée en python à l'aide de la bibliothèque **PyQt**.

Création d'interfaces graphiques



Qt Designer : Outil graphique de création et de codage des interfaces pour Qt

▪ Installation des programmes :

1) Editeur de code :

Pour utiliser **PyQt** il faudra premièrement installer un éditeur de code.

Nous avons opté pour **Thonny**. Il s'agit d'un éditeur gratuit, créé dans un but éducatif.

Thonny est livré avec une version préinstallée de **python**.

Il est donc inutile d'installer une autre version sur votre PC.

Cet éditeur est disponible en téléchargement sur l'adresse : **https://thonny.org/**

2) Installation des packages :

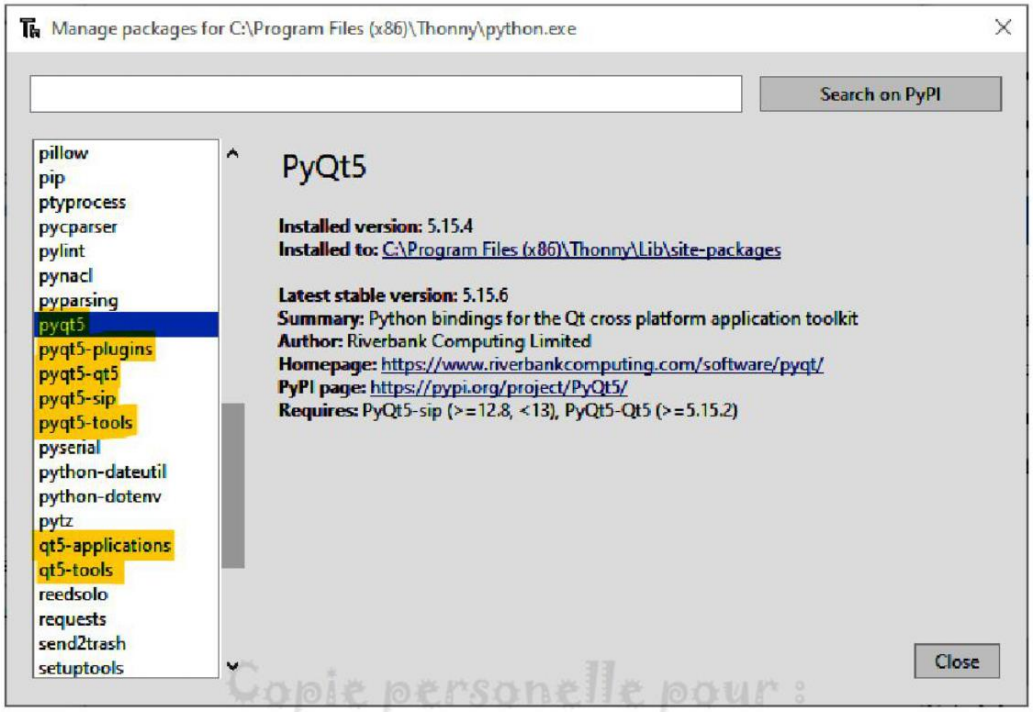
Avant de pouvoir développer des applications avec des interfaces graphiques, il faut installer les packages nécessaires : **PyQt** dans notre cas.

Pour cela, après le lancement de l'éditeur **Thonny**, ouvrir le menu :

Tools > Manage packages...

La fenêtre suivante s'ouvre, dans la liste des packages.

Vérifier, dans la liste, l'existence des packages suivants : (marqués en surbrillance dans la figure suivante).



Les packages requis pour développer à l'aide de PyQt

Pour installer un package il suffit de taper son nom dans la fenêtre de recherche, puis de cliquer sur le bouton [**Search on PyPI**].

Les packages requis, pour développer à l'aide de **PyQt**, sont :

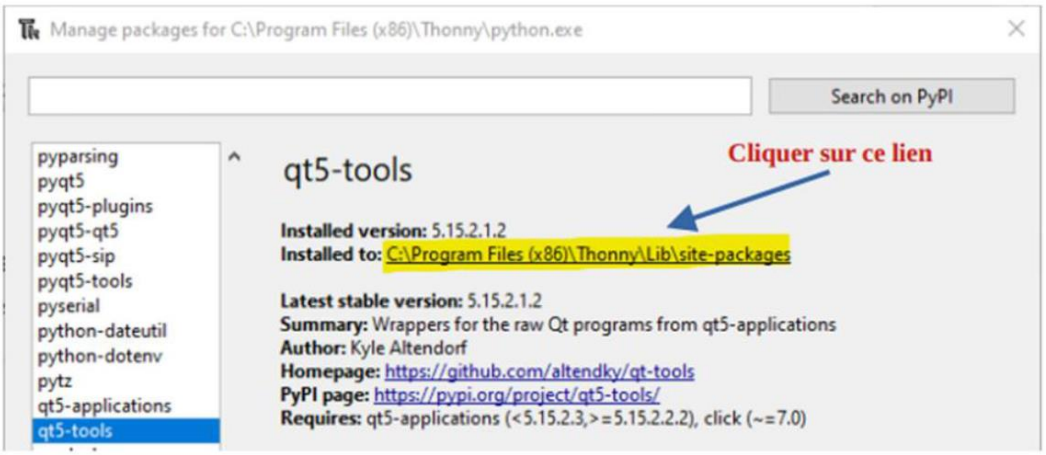
- **pyqt5**
- **pyqt5-tools**
- **pyqt5-qt5**
- **pyqt5-sip**
- **qt5-tools**
- **qt5-applications**

Une fois ces dépendances, sont correctement, installées on peut passer à l'écriture du programme.

3) Editeur d'interfaces graphiques (QtDesigner) :

Après l'installation des packages il est conseillé de créer un raccourci vers **QtDesigner** sur le bureau.

Pour procéder, ouvrir le menu : Tools > Manage packages…et puis cliquer sur le lien mis en surbrillance.



Le fichier **designer.exe** se trouve dans le dossier **qt_applications\Qt\bin** sous le dossier **sites-packages**.

Créer un raccourci à partir du fichier **designer.exe** :

- **Cliquer** avec le bouton droit au-dessus du fichier.
- **Envoyer** vers > Bureau (Créer un raccourci).

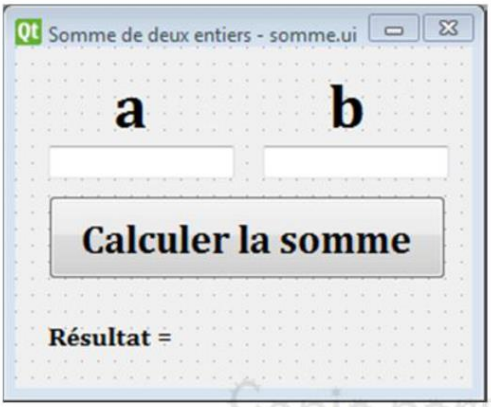
A la fin de toute cette procédure nous pouvons commencer le développement d'applications à interfaces graphiques basées sur **PyQt5**.

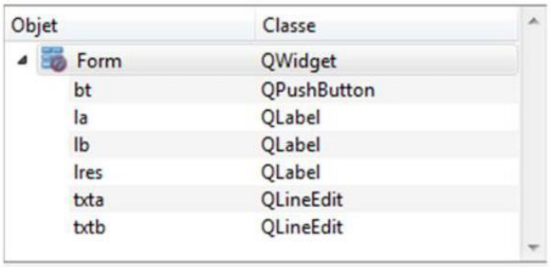
Exemple : Somme de de deux entiers

1) Création de l'interface :

On veut créer une application **PyQt** qui calcule et affiche la somme de deux entiers saisis dans des champs de texte appropriés.

Un croquis de l'interface graphique est présenté dans la figure suivante.





On commence, premièrement, par lancer **QtDesigner** et créer notre interface.

Elle sera composée de :

- d'un bouton **bt** de type **QPushButton**
- trois labels **la**, **lb**, **lres** de type **QLabel**
- deux champs de texte **txta** et **txtb** de type **QlineEdit**

Enregistrer l'interface, dans le dossier de travail, sous le nom : **somme.ui**

Une fois l'interface est prête, on peut lancer notre éditeur de code **python** préféré, **Thonny**, et taper le code de notre programme.

2) Affichage de l'interface :

Créer un fichier, dans le dossier de travail, nommé **somme.py**.

Puis taper le code suivant :

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
app = QApplication ([])
win = loadUi ("somme.ui")
win.show()
app.exec_()
```

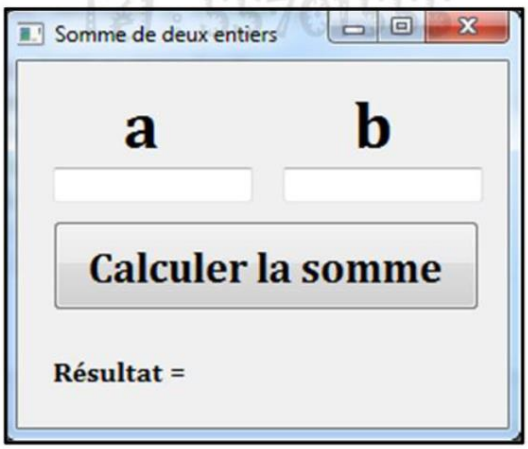
Le code précédent permet de charger puis d'afficher l'interface créée à l'aide de **QtDesigner** et enregistrée dans le fichier **somme.ui**

Il s'agit du **code minimal** qu'on peut écrire en **PyQt5**.

Il comporte exactement six lignes qu'on expliquera une par une :

- **Ligne 1 :** permet d'importer le module **uic** utilisé pour la manipulation et le chargement des fichier de type **ui.**
- **Ligne 2 :** permet d'importer la classe **Qapplication** qui constitue le noyau d'une application **PyQt.**
- **Ligne 3 :** on crée une instance de l'application **(QApplication)** dans la variable **app** qui servira par la suite au lancement de l'application.
- **Ligne 4 :** on génère l'interface graphique enregistrée dans le fichier **somme.ui** à l'aide la fonction **loadUi (...)** notre interface, **initialement invisible**, sera accessible à travers la variable nommée **win.**
- **Ligne 5 :** on commande l'affichage de la fenêtre **win**, après cette instruction notre fenêtre devient **visible.**
- **Ligne 6 :** on lance l'exécution de l'application qui va s'occuper de l'interaction avec l'utilisateur et avec le système d'exploitation hôte.

Lancer l'exécution du programme, après un instant la fenêtre de l'application apparaît :



Remplir les champs de texte par des valeurs de votre choix, puis cliquer sur le bouton **[Calculer la somme]**. **Qu'est ce qui se passe ? Rien** !

Nous n'avons pas encore indiqué quel est le comportement de notre application lorsqu'on clique sur le bouton appuyé.

3) Les évènements :

Contrairement à la **programmation séquentielle**, où les instructions d'un programme sont exécutées une par une dans l'ordre de leurs apparitions, que nous avons pratiqué jusqu'ici, nous parlons de **programmation évènementielle** lorsqu'il s'agit de programmer une interface graphique.

En **programmation évènementielle** des bouts de code sont exécutés suite à un évènement qui peut être :

- Le clic sur un bouton de la souris,
- La sélection d'un élément dans une liste de choix,
- Le clic sur un bouton dans l'interface graphique,
- La modification du contenu d'un champ de texte,
- etc.

Dans notre application nous allons nous intéresser à l'évènement **clicked** qui surgit suite au clic sur le bouton **[Calculer la somme]**.

Nous devons connecter cet évènement à la fonction qui sera exécutée suite à un clic sur ce bouton.


```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
app = QApplication ([])
win = loadUi ("somme.ui")
win.show()
# Connecter l'évènement clicked à la fonction calcul_somme
win.bt.clicked.connect (calcul_somme)
app.exec_()
```

Taper la ligne ajoutée en gras, ci-dessus.

Cette ligne veut dire littéralement : **connecter** la fonction **calcul_comme** à l'évènement **clicked** du bouton **bt** de la fenêtre **win**.

Lors de la création de l'interface, à l'aide de **QtDesigner**, nous avons donné le nom : **bt** au bouton qui nous intéresse.

Maintenant, nous devons définir la fonction **calcul_somme** qui :

- **Vérifie** que les champs de texte ne sont pas vides.
- **Vérifie** que les champs contiennent des valeurs numériques.
- **Calcule** et **affiche** la somme des deux valeurs.

4) Code complet :

```
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication
def calcul_somme ():
   sa = win.txta.text ()
   sb = win.txtb.text ()
   if not (sa.isdigit()) or not (sb.isdigit()):
      win.lres.setText ("Erreur")
   else :
      s = int (sb) + int (sa)
      win.lres.setText ("La somme est : " + str (s))
app = QApplication([])
win = loadUi("somme.ui")
win.show()
win.bt.clicked.connect(calcul_somme)
app.exec_()
```


La fonction **calcul_somme** est appelée gestionnaire d'évènement, elle sera exécutée suite au clic sur le bouton **bt**.

Cette fonction comporte 4 étapes :

- **Étape 1 : récupérer** le contenu des champs de texte, cette opération est réalisée de la façon suivante : **nom_fenetre.nom_composant.text()**
  Dans notre cas la fenêtre est appelée **win** et les champs de texte **txta** et **txtb**.
- **Étape 2 : vérifier** la validité du contenu des deux champs. Si l'un des deux champs est vide et qu'il ne contient pas uniquement des chiffres on affiche une erreur.
- **Étape 3 :** si les deux champs sont valides on convertit **sa** et **sb** en des entiers et on fait leur somme.

- **Étape 4 :** le résultat de l'opération doit se faire dans un label. Pour modifier le contenu d'un label on écrit : **nom_fenetre.nom_composant.setText(texte)**
  Le label qui contient le résultat a été appelée **lres**.
  Il contiendra le contenu de la variable **s**.

5) Résumé :

Pour créer une application à l'aide de **PyQt** :

- **Créer** l'interface graphique à l'aide de **QtDesigner**, et l'enregistrer au format **UI**,
- **Écrire** le code minimal qui permet de générer l'interface graphique depuis le **fichier.ui**, puis de l'afficher,
- **Ajouter** le code qui connecte les évènements aux fonctions appropriées,
- **Écrire** les gestionnaires d'évènements pour répondre aux évènements.