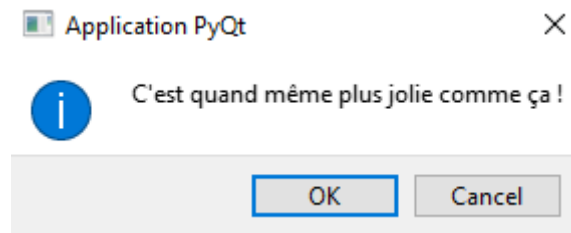


BTS SN Spécialité IR	Création de programme fenêtré avec PyQt	1 <sup>ère</sup> année
-------------------------	---	------------------------

**Mise en situation :** Vous connaissez maintenant les bases du langage python, cependant pour l'instant nous avons toujours affiché tous nos résultats dans la console de pyCharm. Hors lorsque l'on développe un programme complet, on préfère généralement avoir un résultat un peu plus joli et surtout plus intuitif dans l'utilisation.

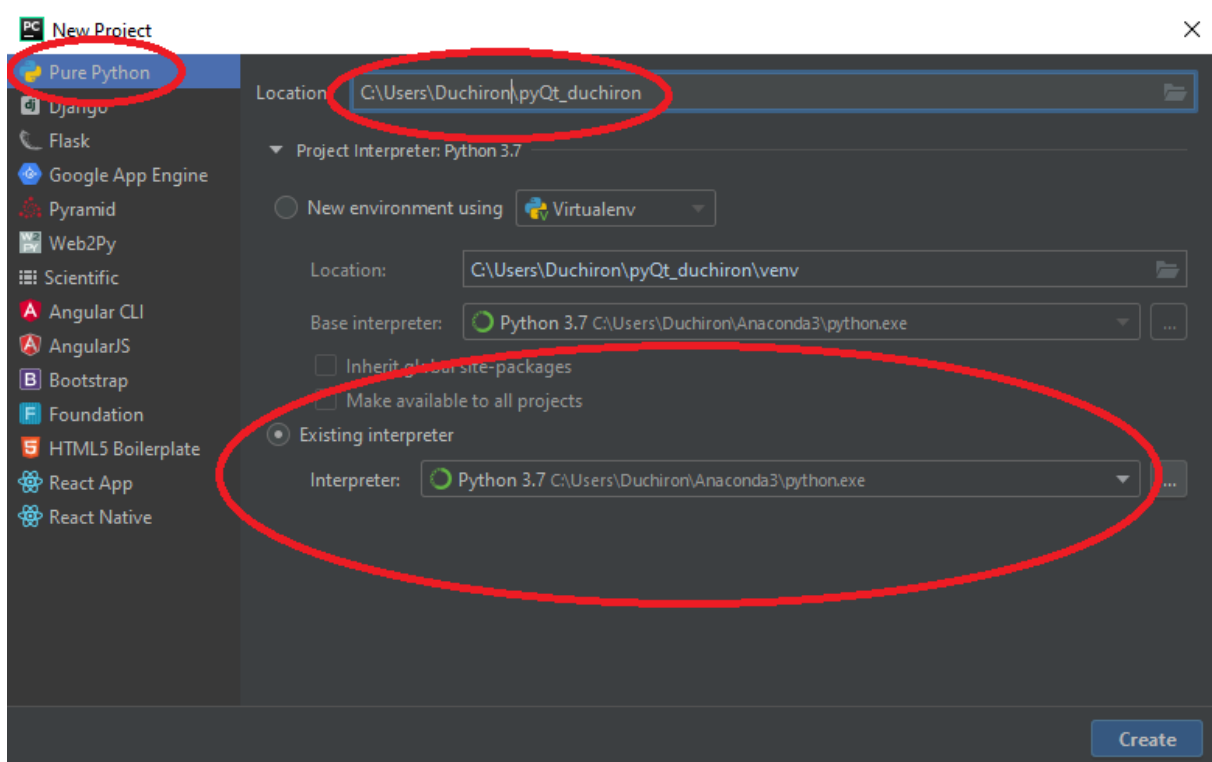
PyQt est un ensemble de bibliothèques qui permet cela, et vous allez apprendre à l'utiliser !



## 1. Installation des éléments nécessaires :

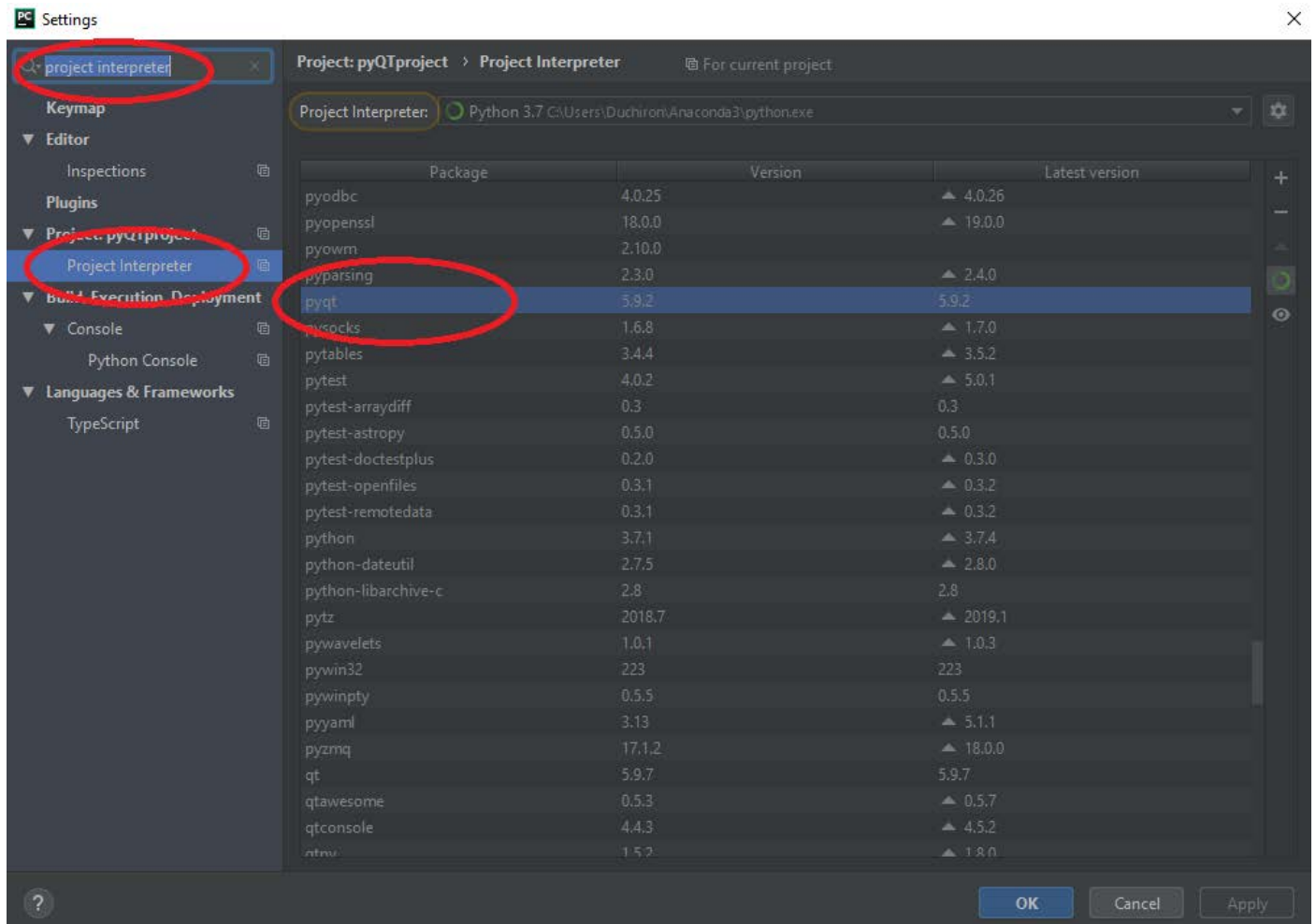
PyQt contient des milliers de bibliothèques, avant toute chose il faut s'assurer que pyCharm sait où les trouver.

- Créer un nouveau projet intitulé ; PyQt\_nom
- Dans l'interface de création du projet : Créer le projet comme vous l'avez appris en respectant les champs suivants :



BTS SN Spécialité IR	Création de programme fenêtré avec PyQt	1 <sup>ère</sup> année
-------------------------	---	------------------------

- Ensuite dans le menu file/settings :
  - Rechercher : project interpreter, vous devriez arriver sur la fenêtre suivante dans laquelle vous allez chercher PyQt, s'il n'y est pas, cliquer sur le + à droite pour l'ajouter !



Voilà, tout est maintenant fonctionnel pour utiliser les bibliothèques de PyQt.

BTS SN Spécialité IR	Création de programme fenêtré avec PyQt	1 <sup>ère</sup> année
-------------------------	---	------------------------

## 2. Ma première fenêtre :

Copier le code suivant et exécuter le :

```
from PyQt5 import QtWidgets
from PyQt5.QtWidgets import QApplication, QWidget
import sys

def applicationpyQt():
    app = QApplication(sys.argv)
    windows = QWidget()
    windows.show()
    sys.exit(app.exec_())

if __name__ == '__main__':
    applicationpyQt()
```

Analysons ce code :

```
from PyQt5 import QtWidgets
from PyQt5.QtWidgets import QApplication, QWidget
import sys
```

Cette première partie contient l'importation de toutes les bibliothèques nécessaires pour faire une simple fenêtre avec Qt

```
def applicationpyQt():
    app = QApplication(sys.argv)
    windows = QWidget()
    windows.show()
    sys.exit(app.exec_())
```

Cette deuxième partie est un peu plus complexe, il s'agit de la fonction qui va créer et gérer la fenêtre.

`app = QApplication(sys.argv)` : Crée l'application

`windows = QWidget()` : Crée l'objet qui servira de conteneur pour les autres éléments

`windows.show()` : Affiche l'objet principal

`sys.exit(app.exec_())` : Exécute l'application

Bon c'est bien, mais ça manque un peu de contenu !

Ajouter la ligne :

`windows.setWindowTitle('PyQt')` juste après la création de l'objet windows

**Q1 : Observer la fenêtre, quel est le changement ? \_\_\_\_\_**

BTS SN Spécialité IR	Création de programme fenêtré avec PyQt	1 <sup>ère</sup> année
-------------------------	---	------------------------

### 3. La documentation Qt :

Qt est ce qu'on appelle un framework, c'est-à-dire qu'il s'agit d'un ensemble de milliers de bibliothèques développées pour faire des applications complètes. Pour utiliser ce framework il existe une documentation officielle de Qt qui permet de savoir comment utiliser les différentes fonctions.

Ajouter la ligne : `windows.setGeometry(200, 200, 200, 200)` à la suite du dernier ajout.

### Q2. Quel est le changement ? \_\_\_\_\_

D'après les explications précédentes, pourquoi faut-il toujours laisser les lignes `windows.show()` et `sys.exit(app.exec_())` à la fin de la fonction ?

---



---



---

La documentation de la méthode `setGeometry` est décrite ici :

<https://doc.qt.io/qtforpython/PySide2/QtWidgets/QWidget.html#PySide2.QtWidgets.PySide2.QtWidgets.QWidget.setGeometry>

**Exercice 1 :** En utilisant cette documentation, changer les dimensions de la fenêtre de façon à obtenir une fenêtre de 500 pixels de large, 300 pixels de haut, placer à 50 pixels du haut et à 50 pixels de la gauche de l'écran.

(N.B. : W signifie généralement width, et H signifie généralement height)

De façon général, l'ensemble de la documentation Qt est disponible ici :

<https://doc.qt.io/qtforpython/modules.html>

Désormais lorsque vous cherchez la façon d'utiliser un objet Qt, vous devez vous référer à cette documentation.

### 4. Ajoutons du texte :

Remplissons un peu la fenêtre avec du texte, pour cela on utilise l'objet `QLabel` du module `QtWidgets`.

Voici la ligne à ajouter pour créer un label de texte vide :

```
text = QtWidgets.QLabel(windows)
```

**Explication :** on nomme le Label « text », on le déclare comme `QLabel`, on lui donne comme paramètre l'objet 'windows' qui était notre fenêtre. C'est cette affectation qui permet de faire en sorte que le Label soit dans la fenêtre.

BTS SN Spécialité IR	Création de programme fenêtré avec PyQt	1 <sup>ère</sup> année
-------------------------	---	------------------------

## **Exercice 2 :**

- A partir de la documentation du QLabel (disponible [ici](#)), faites-en sorte d'ajouter le texte : « Vive la NSI » dans le label.
- Déplacer le label pour qu'il soit à peu près au centre de la fenêtre
- Modifier maintenant votre programme pour que le Label affiche le résultat de la multiplication entre 2 variables contenant des nombres de votre choix.

## **5. Ajoutons des boutons :**

Nous progressons vers une véritable application, maintenant il est temps d'ajouter des boutons ! ([Documentation ici](#))

Voici le prototype pour la création d'un bouton :

```
bouton = QtWidgets.QPushButton('texte', conteneur)
```

## **Exercice 3 :**

- Créer un bouton avec le texte 'Go' et qui est contenu dans la fenêtre
- Déplacer le pour qu'il soit juste sous le label

## **Cours : Les signaux et les slots :**

*Avoir un bouton c'est bien mais s'il ne fait rien, ça n'est pas très utile. Pour cela il est essentiel de comprendre un des fonctionnements de Qt, les signaux et les slots :*

**Les signaux :** Certains objets de Qt peuvent émettre des signaux dans certains cas bien précis, qui sont préprogrammés dans les bibliothèques de Qt. Ces signaux sont des événements cela permet donc de prévenir le reste du programme que tel ou tel événement s'est produit. Voici deux exemples de quelques signaux usuels :

- Clicked* : est cliqué
- Activated* : est activé
- Etc.*

*Chaque objet possède potentiellement des signaux différents, en effet il paraît logique qu'on n'obtienne pas les mêmes usages entre un bouton, une zone de texte ou une liste déroulante.*

*Pour connaître les signaux disponibles sur un objet, il faut se référer à la documentation de Qt.*

**Exercice 4 :** A partir de la documentation sur les boutons, identifier les signaux et donner l'événement auxquels ils correspondent :

BTS SN Spécialité IR	Création de programme fenêtré avec PyQt	1 <sup>ère</sup> année
-------------------------	---	------------------------

**Les slots :** Le slot c'est une fonction qui sera exécutée lorsque le signal sera reçu. Par conséquent, c'est dans cette fonction que l'on va placer le code que l'on souhaite exécuter lors de la réception du signal. Un signal une fois émit doit toujours être connecté sur un slot.

Il existe certains slots qui sont préprogrammés et utilisable directement, mais la plupart du temps il faudra coder soi-même la fonction qui va servir de slot.

Exemple : voici la ligne qui permet de connecter un bouton à un slot :

```
button.clicked.connect(bouton_go)
```

Détaillons la ligne :

- *button* : C'est le nom de mon objet bouton que j'ai créé avant dans mon code.
- *clicked* : C'est le signal que je connecte, ici le signal sera émis lorsque l'utilisateur cliquera sur le bouton
- *connect* : pour indiquer sur quoi on connecte le signal
- *bouton\_go* : C'est le nom de la fonction qui va être exécuté lorsque le signal sera reçu.

**Exercice 5-1 :** Essayer d'ajouter la ligne précédente à votre code (en la modifiant pour que cela corresponde à votre bouton), quel est le problème rencontré ?

---



---



---

**Exercice 5-2 :** Ajouter la fonction *bouton\_go* à votre code et faites-en sorte qu'elle affiche 'Bonjour' dans le Label.

## 6. Récupérer du texte entré par l'utilisateur :

Pour avoir un programme véritablement interactif, il faut que l'utilisateur puisse entrer lui-même des informations, et particulièrement du texte.

Un des objets qui permet cela est le `QLineEdit`, la façon de le créer est la même que celle du bouton. ([Documentation ici](#))

Ensuite la méthode pour récupérer le texte s'appelle `text()`.

**Exercice 6 :** A partir de la documentation et des signaux de `QLineEdit` :

- Créer un objet `QLineEdit` sous le bouton.
- Créer le slot `text_change` qui va afficher dans le label la valeur contenue dans le `QLineEdit`.
- Connecter le au bon signal de l'objet pour que le contenu du label change lorsque le `QLineEdit` est modifié.

BTS SN Spécialité IR	Création de programme fenêtré avec PyQt	1 <sup>ère</sup> année
-------------------------	---	------------------------

## 7. QtDesigner :

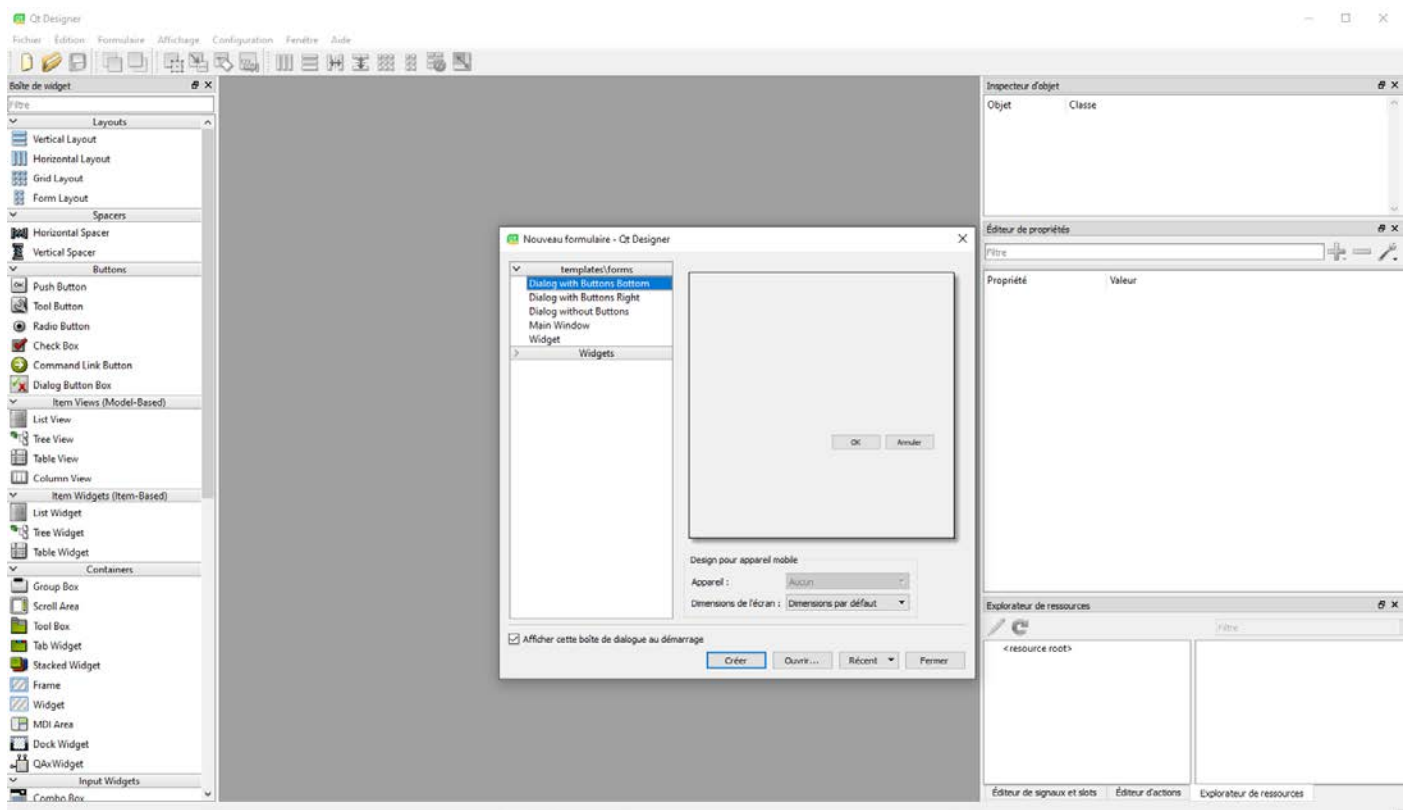
Si vous avez pris la peine de jeter un coup d'œil à la documentation de Qt vous avez vu qu'il existe des centaines et des centaines d'objet graphiques. Il est compliqué de tous les connaître et de savoir lequel utiliser.

De plus il n'est pas aisé de placer les éléments exactement comme on le souhaite.

Heureusement il existe une interface graphique pour créer l'interface graphique ! Elle s'appelle QtDesigner.

- Pour la lancer taper : Designer dans le terminal de pyCharm de votre projet.

Vous devriez obtenir une fenêtre semblable à ça :

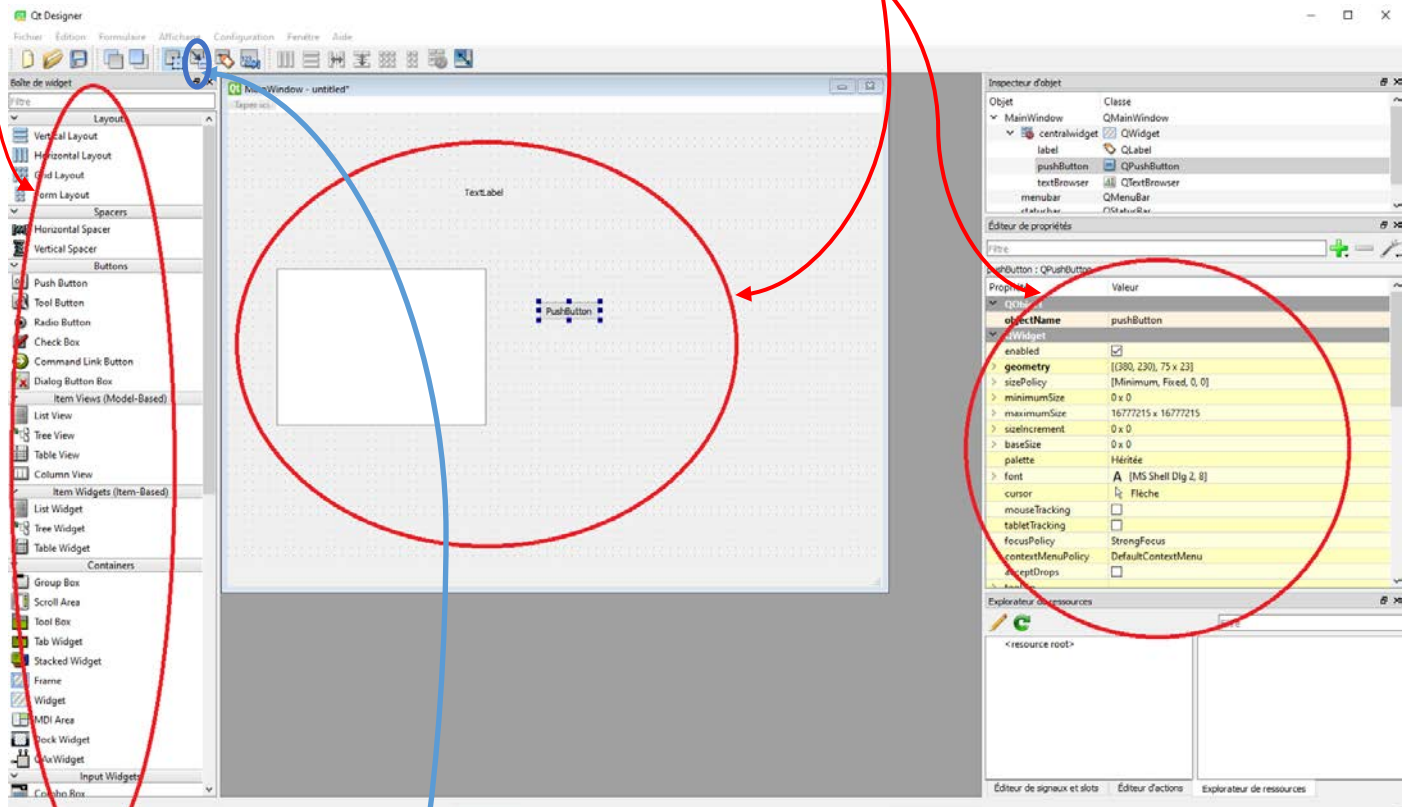


- Sélectionnez Main Windows puis cliquez sur créer



Vous arrivez dans la fenêtre de création.

- Au centre ce à quoi ressemble votre application
- A gauche tous les objets graphiques disponible
- A droite les propriétés de vos objets graphiques

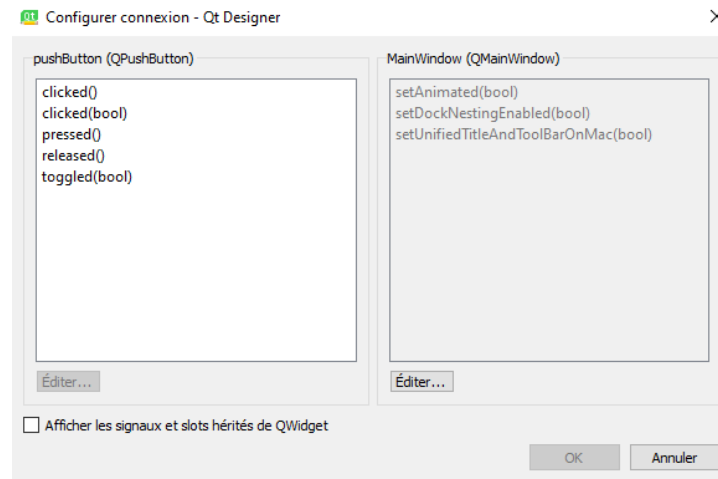


Une fois que vous avez ajouté tous les éléments que vous vouliez, vous pouvez configurer les signaux et les slots, ici :



BTS SN Spécialité IR	Création de programme fenêtré avec PyQt	1 <sup>ère</sup> année
-------------------------	---	------------------------

- Pour éditer les signaux et slots, sélectionnez un élément et glissez dans le vide, la fenêtre suivante va s'ouvrir :



- A gauche vous avez les signaux disponibles pour l'objet, sélectionnez celui que vous voulez.
- A droite cliquez sur Éditer... puis sur le + dans la rubrique slot
- Ajouter le slot que vous voulez

Une fois que vous avez configuré la partie graphique et les comportements, vous n'avez plus qu'à sauvegarder votre fichier au format .ui dans votre dossier de projet.

BTS SN Spécialité IR	Création de programme fenêtré avec PyQt	1 <sup>ère</sup> année
-------------------------	---	------------------------

### Il ne reste plus qu'à faire le lien avec python :

Dans le terminal de pyCharm exécuter la ligne de commande :

```
pyuic5 nomFichier.ui -o nomFichier.py
```

Cela a généré un programme python qui devrait ressembler à cela :

```

1  # -*- coding: utf-8 -*-
2
3  # Form implementation generated from reading ui file 'monFichier.ui'
4  #
5  # Created by: PyQt5 UI code generator 5.9.2
6  #
7  # WARNING! All changes made in this file will be lost!
8
9  from PyQt5 import QtCore, QtGui, QtWidgets
10
11 class Ui_MainWindow(object):
12     def setupUi(self, MainWindow):
13         MainWindow.setObjectName("MainWindow")
14         MainWindow.resize(800, 600)
15         self.centralwidget = QtWidgets.QWidget(MainWindow)
16         self.centralwidget.setObjectName("centralwidget")
17         self.text = QtWidgets.QLabel(self.centralwidget)
18         self.text.setGeometry(QtCore.QRect(250, 150, 121, 31))
19         self.text.setObjectName("text")
20         self.button = QtWidgets.QPushButton(self.centralwidget)
21         self.button.setGeometry(QtCore.QRect(250, 190, 75, 23))
22         self.button.setObjectName("button")
23         self.text_box = QtWidgets.QLineEdit(self.centralwidget)
24         self.text_box.setGeometry(QtCore.QRect(250, 230, 191, 31))
25         self.text_box.setObjectName("text_box")
26         MainWindow.setCentralWidget(self.centralwidget)
27         self.menubar = QtWidgets.QMenuBar(MainWindow)
28         self.menubar.setGeometry(QtCore.QRect(0, 0, 800, 21))
29         self.menubar.setObjectName("menubar")
30         MainWindow.setMenuBar(self.menubar)
31         self.statusbar = QtWidgets.QStatusBar(MainWindow)
32         self.statusbar.setObjectName("statusbar")
33         MainWindow.setStatusBar(self.statusbar)
34
35         self.retranslateUi(MainWindow)
36         self.button.clicked.connect(MainWindow.bouton_go)
37         self.text_box.textChanged['QString'].connect(MainWindow.text_change)
38         QtCore.QMetaObject.connectSlotsByName(MainWindow)
39
40     def retranslateUi(self, MainWindow):
41         _translate = QtCore.QCoreApplication.translate
42         MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
43         self.text.setText(_translate("MainWindow", "Vive la NSI"))
44         self.button.setText(_translate("MainWindow", "Go"))
45

```

BTS SN Spécialité IR	Création de programme fenêtré avec PyQt	1 <sup>ère</sup> année
-------------------------	---	------------------------

Voici le squelette du code principal qui va vous permettre d'utiliser ce fichier d'interface :

```

1  from PyQt5 import QtWidgets
2  from monFichier import Ui_MainWindow
3  import sys
4
5  #Pour utiliser le fichier d'interface, on créé une classe qui fait le lien
6  class ApplicationIHM(QtWidgets.QMainWindow):
7      def __init__(self):
8          super(ApplicationIHM, self).__init__()
9          self.ui = Ui_MainWindow()      #Création de la fenêtre
10         self.ui.setupUi(self)          #Initialisation de la fenêtre
11
12
13         def bouton_go(self):
14             #Code de la fonction du bouton
15
16         def text_change(self):
17             #Code de la fonction du QLineEdit
18
19     #Fonction main qui va créer l'application
20     def main():
21         app = QtWidgets.QApplication(sys.argv)
22         application = ApplicationIHM()
23         application.show()
24         sys.exit(app.exec_())
25
26     #Appel de la fonction main
27     if __name__ == "__main__":
28         main()
29

```

### Quelques explications :

- Désormais self.ui fait référence à l'interface graphique, donc tout ce qui concerne l'interface devra toujours commencer par self.ui

Exemple : Changer le texte du bouton :

- On commence par vérifier le nom du bouton dans le fichier généré automatiquement
- On le trouve à la ligne 20 : self.button = QtWidgets.QPushButton

On reconnaît la création du bouton vue dans avant dans l'activité. Donc le nom du bouton est « button » (le mot clef *self* est une auto-référence en Python nous reviendrons dessus plus tard)

- Pour changer le texte du bouton il faut donc écrire dans mon code principal :  
self.ui.button.setText('nouveau texte')

BTS SN Spécialité IR	Création de programme fenêtré avec PyQt	1 <sup>ère</sup> année
-------------------------	---	------------------------

### **Adapter le squelette du code :**

- Modifier la ligne 2 pour remplacer monFichier par le nom du fichier généré.
- Modifier la fonction bouton\_go pour qu'elle corresponde à l'action du bouton. (Si besoin modifier le nom de la fonction pour que cela corresponde à votre slot.)
- Modifier la fonction text\_change pour qu'elle corresponde à l'action du QLineEdit.
- Entraînez-vous en ajoutant des objets graphiques et des comportements !

### **8. Mini-Projet :**

**Vous allez maintenant développer une application complète pour vous entraîner.**

**Nous vous suggérons de créer une calculatrice 4 opérations pour commencer.**

**Vous pouvez solliciter votre professeur si vous avez votre propre idée d'application.**