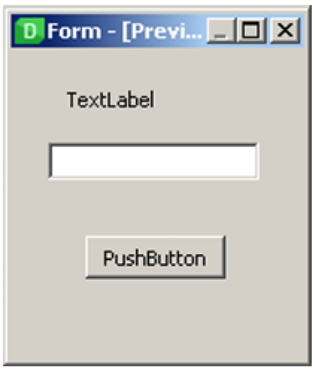
Interface graphique avec Python et Qt5

Dans de nombreux domaines comme la domotique ou la météo, l'interface graphique est privilégiée au mode texte. On va faire la programmation des interfaces graphiques (**GUI** : Graphical User Interface) avec PyQt5 qui permet d'utiliser la bibliothèque Qt version 5 avec Python.



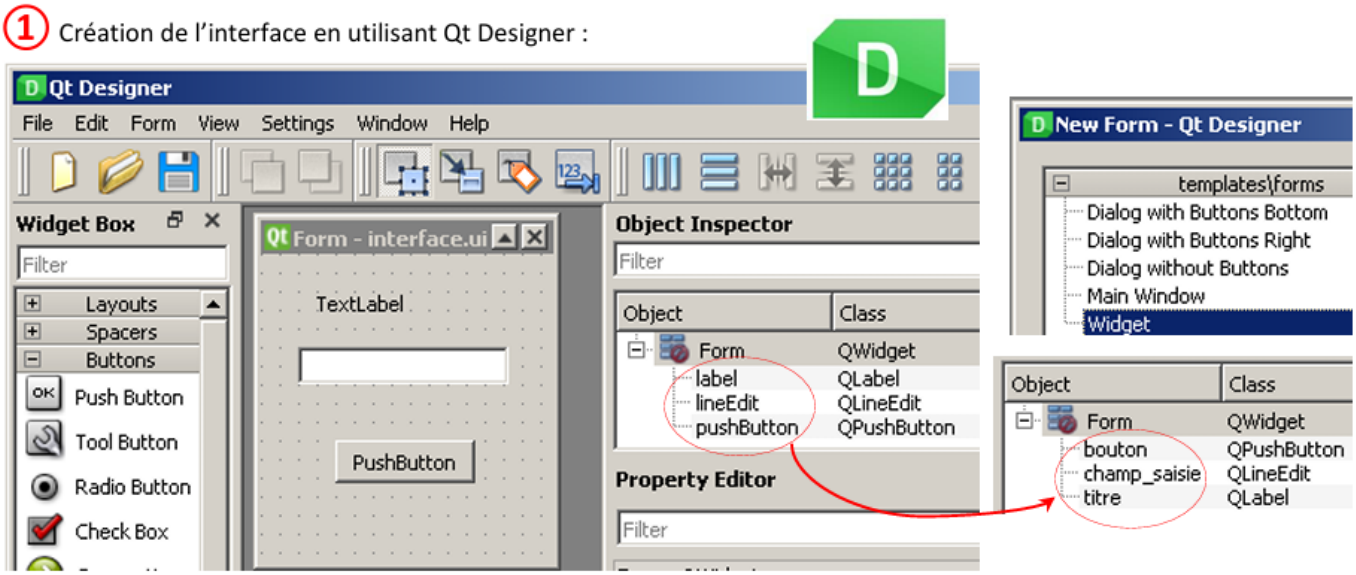
Qt offre des composants d'interface graphique (**widgets**), en utilisant les mécanismes des signaux et slots. Examples de widgets : fenêtre, label, Line Edit, Push Button,...

Label Line Edit Push Button

Un **signal** est un message envoyé par un widget lorsqu'un évènement se produit.(exemple : On a cliqué sur un bouton). Un **slot** est la fonction qui est appelée lorsqu'un événement s'est produit. On dit que le signal appelle le slot. Un signal est connecté au slot via la fonction **connect()** .

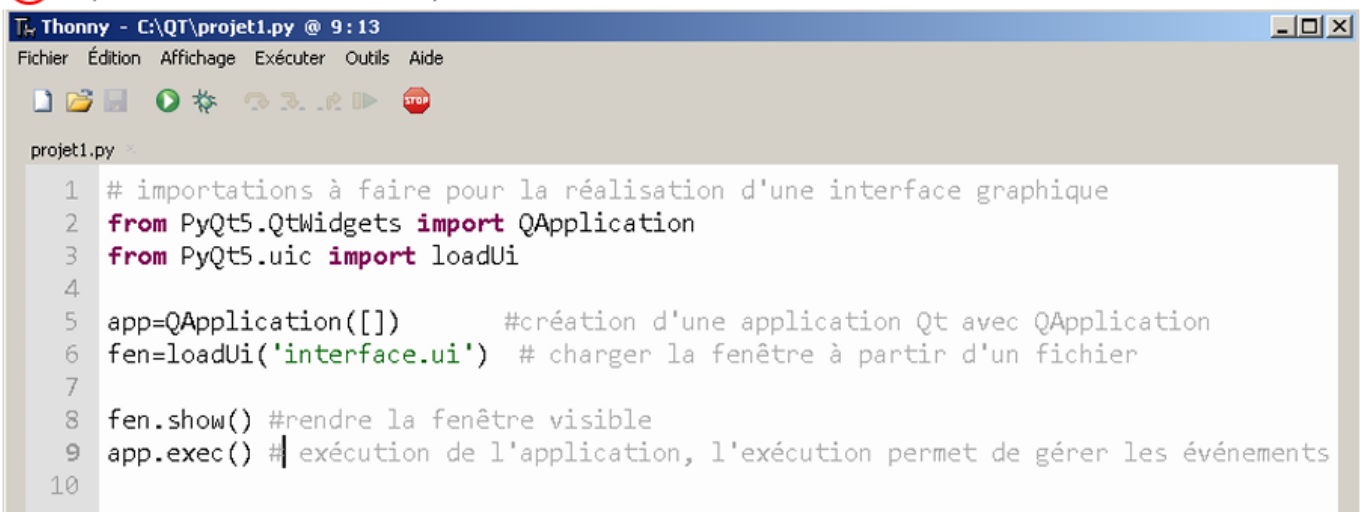
NB : On va utiliser Thonny IDE et Qt Designer après installation de PyQt5 et pyqt5-tools.
Voir les étapes d'installation sous Windows dans la vidéo suivante: https://tinyurl.com/thonny-qt5

① Création de l'interface en utilisant Qt Designer :



File → New → Widget, Renommer les objets, puis enregistrer le fichier sous le nom: interface.ui dans un dossier "QT"

② Importation de l'interface avec Python :




```
1  # importations à faire pour la réalisation d'une interface graphique
2  from PyQt5.QtWidgets import QApplication
3  from PyQt5.uic import loadUi
4
5  app=QApplication([])        #création d'une application Qt avec QApplication
6  fen=loadUi('interface.ui')  # charger la fenêtre à partir d'un fichier
7
8  fen.show() #rendre la fenêtre visible
9  app.exec() # exécution de l'application, l'exécution permet de gérer les événements
10
```

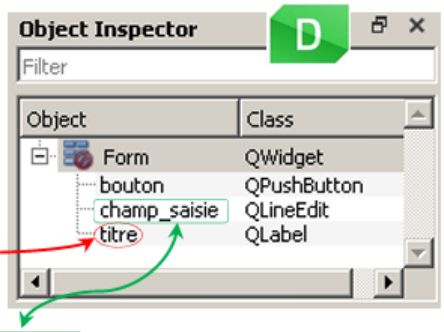
Dans Thonny ecrire un script Python projet1.py et l'enregistrer dans le même dossier "QT" permettant de charger l'interface graphique .

③ Utilisatation des widgets dans un script Python :

Label

- Modifier le texte d'un label : fen.titre.**setText**(' Un Titre ')
- Récupérer le texte d'un label dans une variable : ch=fen.titre.**text()**
- Effacer le contenu d'un label : fen.titre.**clear()**
- Mettre un nombre dans d'un label : fen.titre.**setNum**(18.5)

Voir plus: https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QLabel.html



Line Edit

fonction :

- Récupérer le texte d'un « Line Edit » dans une variable : ch=fen.champ_saisie.**text()**

Slots :

- Modifier le texte d'un « Line Edit » : fen.champ_saisie.**setText**(' Un texte ')
- Effacer le contenu d'un « Line Edit » : fen.champ_saisie.**clear()**
- Ajouter à la fin du contenu du champ de saisie : fen.champ_saisie.**insert** ('plus')

Signaux :

- Un signal est émis chaque fois que le texte est édité, et le traitement 1 est exécuté

```
def traitement1() :
        .........
fen.champ_saisie.textEdited.connect(traitement1)
```

Remarque : Il y a aussi `textChanged` :Detection de changement de champ de saisie par edition ou par affectation de variable.

- Un signal est émis chaque fois que la touche « Entrée » est enfoncée dans le champ de saisie, et le traitement 2 est exécuté.

```
def traitement2() :
        .........
fen.champ_saisie.returnPressed.connect(traitement2)
```

https://doc.qt.io/qtforpython-5/PySide2/QtWidgets/QLineEdit.html

Push Button

Signaux : • Un signal est émis chaque fois que le bouton est appuyé, et le traitement 3 est exécuté

```
def traitement3() :
        .........
fen.bouton.clicked.connect(traitement3)
```

Remarque 1: Il y a aussi les signaux `pressed()` et `released()`

Remarque 2: On peut fermer l'application en utilisant : `fen.bouton.clicked.connect(fen.close)`

On peut effacer le contenu du champ de saisie: `fen.bouton.clicked.connect(fen.champ_saisie.clear)`

Exemple de script Python : *(Ecrire un texte dans le champ de saisie puis cliquer sur le bouton pour le reproduire dans le label titre)*

```
# importations à faire pour la réalisation d'une interface graphique
from PyQt5.uic import loadUi
from PyQt5.QtWidgets import QApplication

def traitement1():
      x=fen.champ_saisie.text()
      fen.titre.setText(x)

app=QApplication([])            #Création d'une application Qt avec QApplication
fen=loadUi('interface.ui')      # Charger la fenêtre à partir d'un fichier
fen.show()                      #Rendre la fenêtre visible
fen.bouton.clicked.connect(traitement1)
app.exec()  #Exécution de l'application (permet de gérer les événements en boucle(event loop))
     #l'exécution  n'arrive à ce niveau qu'après avoir quitté l'application
```