Division des enseignements en informatique

TP Introduction à PyQt4

Ce TP traite de la réalisation d'applications graphiques (avec fenêtres, boutons, etc.) en Python. La création d'applications graphiques nécessite le choix d'un framework particulier. Notre choix se portera sur PyQt qui permet de lier Python à la librairie Qt développée par la société Trolltech.

Les raisons de ce choix sont la portabilité des interfaces développées en PyQt (elles peuvent fonctionner sur tous les systèmes d'exploitation), la maturité de la bibliothèque, sa relative simplicité d'utilisation ainsi que sa disponibilité sous Python 2 et 3.

Nous utiliserons dans ce TP la version 4 de PyQt avec Python 3.

L'objectif est de guider le lecteur dans la réalisation de programmes comportant une interface graphique.

1 Installation

Avant de vous lancer dans l'installation de la librairie PyQt, assurez-vous qu'elle n'est pas déjà présente sur votre poste. Dans une console Python, tapez

```
import PyQt4
```

Si vous obtenez un message d'erreur, c'est qu'il vous faut installer la librairie.

1.1 Installation sous Linux

Il suffit d'installer le paquets python3-pyqt4, par exemple en ligne de commandes dans un terminal :

```
sudo apt-get install python3-pyqt4
```

Le gestionnaire de paquets va se charger d'installer la librairie et tous ses prérequis pour qu'elle fonctionne correctement. Un fois l'installation terminée, vous pouvez contrôler le résultat en important PyQt4 depuis un interpréteur Python.

1.2 Installation sous Windows

La solution la plus simple consiste à télécharger l'exécutable correspondant à votre version d'ordinateur sur https://riverbankcomputing.com/software/pyqt/download.

Testez l'import de la librairie PyQt4 une fois l'installation terminée.

2 Premier exemple : Hello World

Pour commencer la découverte de la librairie PyQt, nous réaliserons une application, classique en informatique, affichant un message “Hello World !” dans une fenêtre. Nous complèterons cette interface avec un bouton permettant de quitter l’application.

Nous travaillerons dans un script `hello_world.py`.

Pour pouvoir utiliser la librairie PyQt, commençons par l’importer dans notre script :

```
from PyQt4.GtGui import *
```

Puis créons une application Qt avec `QApplication`. Cette application s’appellera “Hello world” :

```
app = QApplication(["Hello world"])
```

Toute application PyQt sera toujours constituée d’un objet `QApplication`.

Nous ajoutons ensuite des widgets à cette application. Les widgets sont les composants de base d’une interface en Qt (boutons, menus...). Ici nous désirons créer une simple fenêtre (il s’agit d’un `QWidget`) :

```
fen = QWidget()
```

Et rendons ce widget visible :

```
fen.show()
```

Enfin, exécutons l’application :

```
app.exec()
```

Qt nous permet de paramétrer la taille et la position de la fenêtre ou d’en changer le titre. Avant l’instruction d’affichage de la fenêtre ajoutez :

```
fen.setWindowTitle("Hello World !")
fen.resize(500, 250)
fen.move(300, 50)
```

Nous allons maintenant ajouter une étiquette avec un texte “Hello World !” et un bouton permettant de quitter l’application.

Comme nous souhaitons ajouter deux composants, il est nécessaire d’indiquer au programme comment les agencer dans la fenêtre. Pour ce faire, nous utiliserons une grille. Il existe plusieurs types de grilles en Qt. Ici nous positionnerons les widgets de haut en bas avec l’utilisation d’une grille verticale `QVBoxLayout()` :

```
layout = QVBoxLayout()
fen.setLayout(layout)
```

Puis les deux composants sont ajoutés à la grille. Le premier widget ajouté sera placé en tout en haut, puis les suivants se positionneront les uns en dessous des autres.

L’étiquette est un objet `QLabel`. Nous précisons son contenu lors de sa création et l’ajoutons à la grille :

```
label = QLabel("Hello World !!")
layout.addWidget(label)
```

Le bouton est un objet `QPushButton` pour lequel nous précisons également le texte affiché puis l’ajoutons à la grille :

```
bouton = QPushButton("Quitter")
layout.addWidget(bouton)
```

Pour terminer notre programme, il ne nous reste plus qu'à associer à un clic sur le bouton "Quitter" l'action de fermer l'application.

Le principe en Qt pour réaliser cette tache est de connecter l'action de clic sur le bouton (le signal `clicked`) à la fonction `quitter()`. La syntaxe est :

```
bouton.clicked.connect(quitter)
```

La fonction `quitter()` contient simplement :

```
def quitter():
    app.exit(0)
```

3 Hello World : deuxième version

La plupart du temps, lorsque nous réaliserons des interfaces graphiques, nous écrirons une classe[1] correspondant à la fenêtre et y ajouterons les widgets.

Le fonctionnement est strictement identique. Il s'agit juste d'une ré-écriture. Pour le programme précédent, cela donnera :

```
import sys
from PyQt4.QtGui import *


class MaFenetre(QWidget):
    def __init__(self):
        QWidget.__init__(self)

        # Personnalisation de la fenêtre
        self.setWindowTitle("Hello World !")
        self.resize(500, 250)
        self.move(300, 50)

        # Création d'un gestionnaire de mise en forme
        self.layout = QVBoxLayout()
        self.setLayout(self.layout)

        # Ajout d'une étiquette
        self.label = QLabel("Hello World !!")
        self.layout.addWidget(self.label)

        # Ajout d'un bouton
        self.bouton = QPushButton("Quitter")
        self.layout.addWidget(self.bouton)

        # Connection du signal clicked à la fonction quitter
        self.bouton.clicked.connect(self.quitter)

        # Affichage de la fenetre
        self.show()
```

1. la notion de classe est une notion importante en programmation, mais il n'est pas nécessaire ici de savoir en écrire pour comprendre comment fonctionne le programme.

```
    def quitter(self):
        """ Méthode permettant de quitter l'application courante """
        sys.exit(0)


if __name__ == '__main__':
    app = QApplication(["Hello world"])
    fen = MaFenetre()
    app.exec()
```

Ecrivez ce programme dans un nouveau script `hello_world_v2.py`.

4 Deuxième application : un compteur simple

Notre deuxième application sera pour l'instant constituée d'un bouton et d'une étiquette. A l'état initial, l'étiquette affiche le contenu d'un compteur qui vaut 0. Chaque clic sur le bouton fait avancer le compteur de 1 et l'étiquette se met à jour.

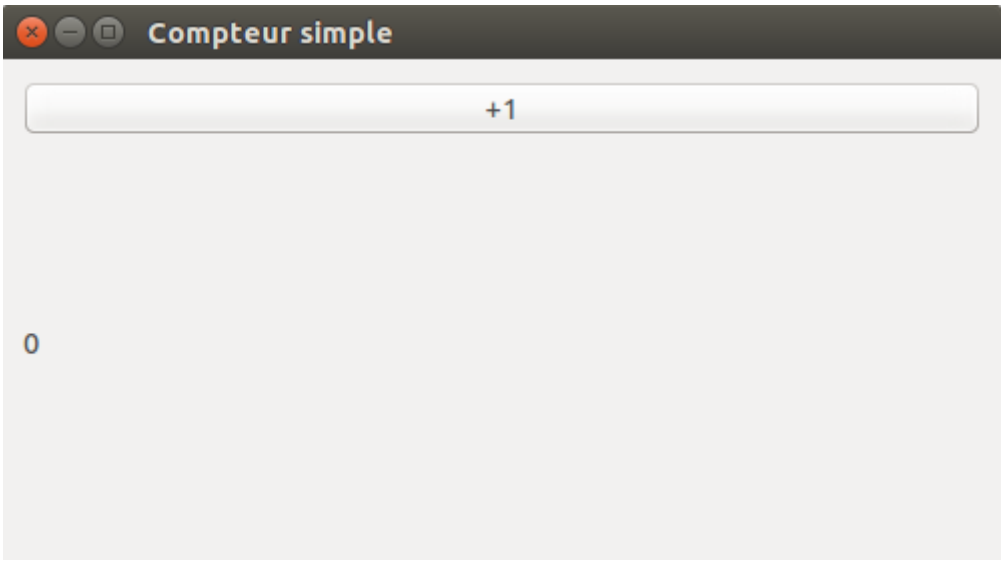

FIGURE 1 – Compteur simple - état initial

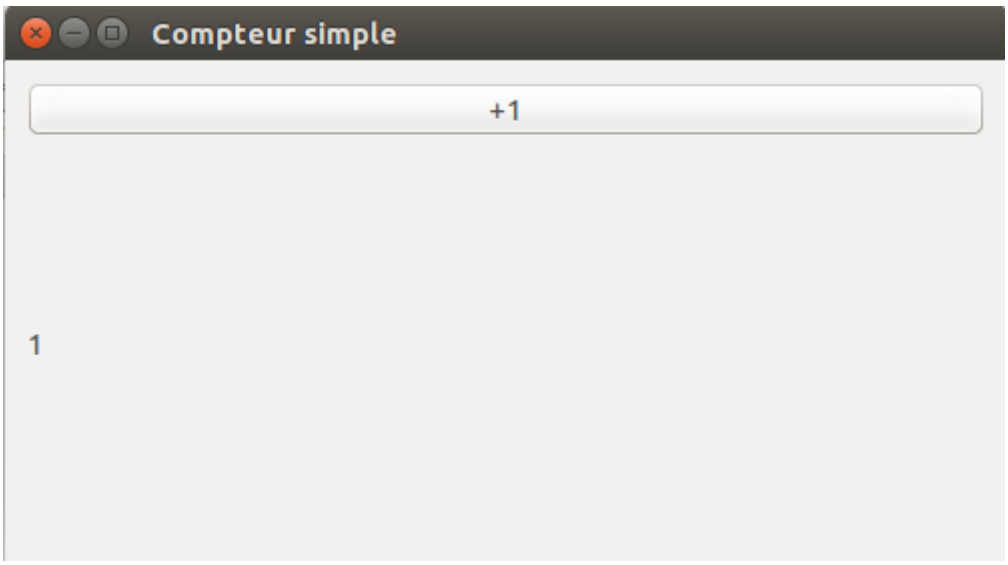

FIGURE 2 – Compteur simple - premier clic

La réalisation de cette application faiut appel aux mêmes éléments que pour la partie précédente. A vous de jouer !

5 Coordonnées de la souris

Pour notre troisième application, nous souhaitons récupérer les coordonnées de la souris lorsqu'elle survole l'application pour les afficher dans une étiquette.

Nous utiliserons la propriété `setMousetracking` de l'objet `QWidget` (http://doc.qt.io/qt-4.8/qwidget.html#mouseTracking-prop), ainsi que la méthode `mouseMoveEvent()` (http://doc.qt.io/qt-4.8/qwidget.html#mouseMoveEvent).

En consultant les liens vers la documentation de Qt ci-dessus, réalisez cette application.