# Create GUI in Python using Qt

Mads M Pedersen ([mmpe@dtu.dk](mailto:mmpe@dtu.dk))  
Document version 1.4, 13/3-2013  
QtGuiLoader version 1.4, 13/3-2013

## Introduction

Qt is a C++ cross-platform application and user-interface framework. It is developed by Nokia and contains a comprehensive library of GUI classes.

Using the visual editor “Qt Designer” GUIs are easily created and maintained.

This tutorial describes a simple way to integrate Qt GUIs in Python programs using PyQt and the Python module QtGuiLoader.

In example 1 a flexible widget is created, which can be used as Widget, Dialog or MainWindow. Follow this example in case you do not need menubar, toolbar and statusbar. Otherwise follow example 2 in which a MainWindow with a menubar is created.

## Requirements

Install Pythonxy from pythonxy.com (Only for windows)  
Or  
Install Python 2.7, Qt (including QtDesigner), and PyQt

Copy QtGuiLoader.py to your working folder

## Example 1

In this example a flexible widget is created, which can be used as Widget, Dialog or main window. Follow this example in case you do not need menubar, toolbar and statusbar

### Create ui

* Open Qt Designer
* Select “Widget” and <Create>\*
* Add a Horizontal Layout
* Add LineEdit to layout
* Add PushButton to layout

\*It is possible to create “Main Window” and “Dialog” as well, but select “Widget” as it can be used for all three purposes when integrating in Python. If however a menu is required then Main Window must be selected, see example 2.

### Add action

* Create new action in Action Editor
  + Text: PrintText
  + Name: actionPrintText

### Connect button to action

* Add Signal/Slot in Signal/Slot Editor
  + Sender: pushButton
  + Signal: clicked()
  + Receiver: actionPrintText
  + Slot: trigger()

### Prepare for integrate in python

* Save as “MyWidgetUI.ui”
* Create an empty “MyWidgetUI.py” file at same location as “MyWidgetUI.ui”\*\*.   
  Its content will be autogenerated if the file is completely empty or older than “MyWidgetUI.ui”

\*\*This is required in order to ”import MyWidget”.

### Integrating in Python

The widget can be integrated as QMainWindow, QDialog or QWidget. When using QtGuiLoader, actions are automatically connected to methods with the same name.

All PyQt elements are found in the ui-object, e.g. x.ui.lineEdit and documentation of the PyQt elements is found at <http://pyqt.sourceforge.net/Docs/PyQt4/classes.html>

Before a gui can be shown, a QApplication must be started. This is automatically done when calling QtMainWindowLoader and QtDialogLoader from QtGuiLoader if the application argument is True.

The following examples are found in UseQtGuiLoader.py

#### Main window

A main window is a separate window, i.e. two main windows can be closed independently.

In order to use the MyWidgetUI as QMainWindow, subclass QtMainWindowLoader from QtGuiLoader and instantiate with MyWidgetUI as ui\_module argument.

|  |
| --- |
| import MyWidgetUI  from QtGuiLoader import QtMainWindowLoader  class **MyMainWindow**(QtMainWindowLoader):  def **\_\_init\_\_**(*self*):  QtMainWindowLoader.\_\_init\_\_(*self*, ui\_module=MyWidgetUI, application=True)    def **actionPrintText**(*self*):  print *"Mainwindow text: %s"*%*self*.ui.lineEdit.text()  MyMainWindow().start() |

#### Dialog

A Dialog is a sub window, i.e. if its parent window is closed the dialog is closed too. A dialog may be modal, such that other windows cannot be operated before the dialog is closed.

In order to use the MyWidgetUI as QDialog, subclass QtDialogLoader from QtGuiLoader and instantiate with “MyWidgetUI” as ui\_module argument.

|  |
| --- |
| import MyWidgetUI  from QtGuiLoader import QtDialogLoader  class **MyDialog**(QtDialogLoader):  def **\_\_init\_\_**(*self*,parent,application,modal):  QtDialogLoader.\_\_init\_\_(*self*, MyWidgetUI, parent, application,modal)    def **actionPrintText**(*self*):  print *"Mainwindow text: %s"*%*self*.ui.lineEdit.text()  MyDialog(parent=None,application=True,modal=True).start() |

#### Widget

A widget can be inserted into an existing widget or layout.

In order to use the MyWidgetUI as Widget, subclass QtWidgetLoader from QtGuiLoader and instantiate with “MyWidgetUI” as ui\_module argument and the parent of the widget as parent.

|  |
| --- |
| *"""Use as widget with actionhandlers in widget subclass"""*  class **MyWidget**(QtWidgetLoader):  def **\_\_init\_\_**(*self*, parent):  QtWidgetLoader.\_\_init\_\_(*self*, ui\_module=MyWidgetUI, parent)  def **actionPrintText**(*self*):  print *"Mainwindow text: %s"*%*self*.ui.lineEdit.text()  app = QtGui.QApplication(sys.argv)  window = QtGui.QMainWindow()  w = MyWidget(parent=window)  window.show()  sys.exit(app.exec\_()) |

## Example 2

In this example a main window with menubar is created. Follow this example if menubar, toolbar and/or statusbar are required

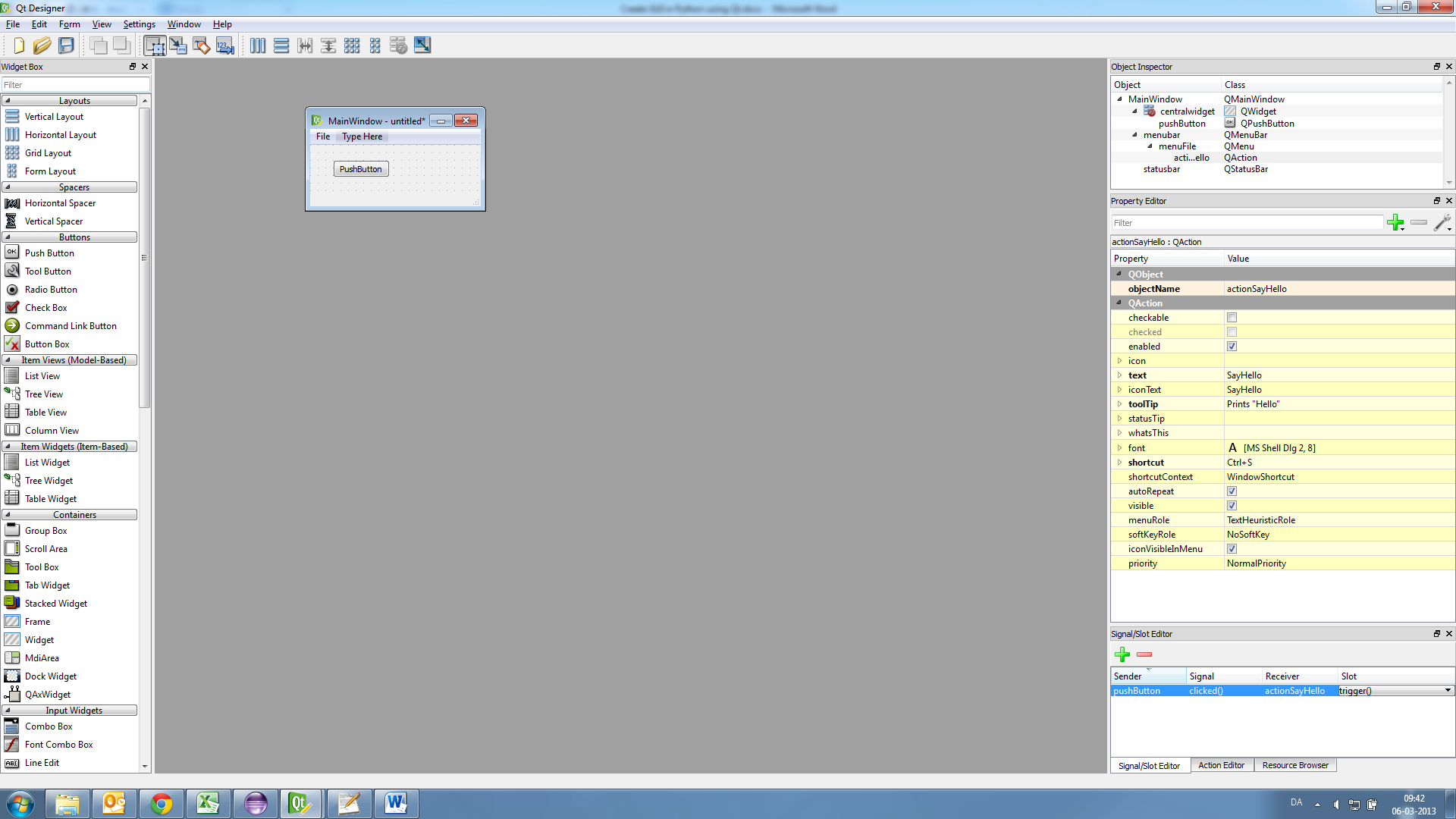
### Create ui

* Open Qt Designer
* Select “MainWindow” and <Create>
* Add PushButton to window

### Add action

* Create new action in Action Editor  
  Text: SayHello  
  Name: actionSayHello  
  Shortcut: Ctrl+S

### Connect action to menu and button

* Doubleclick on “Type Here” in the menubar of the window and type “File”
* Drag actionSayHellow from Action Editor to the file menu
* Add Signal/Slot in Signal/Slot Editor  
  Sender: pushButton  
  Signal: clicked()  
  Receiver: actionSayHello  
  Slot: trigger()

### Prepare for integrate in python

* Save as “MyMainWindowUI.ui”
* Create an empty “MyMainWindowUI.py” file at same location as “MyMainWindowUI.ui”\*\*.   
  Its content will be autogenerated if the file is completely empty or older than “MyMainWindowUI.ui”

\*\*This is required in order to “import MyMainWindow”.

### Integrating in Python

When loading via QtMainWindowLoader, actions are automatically connected to methods of the same name.

All PyQt elements are found in the “ui”-object, e.g. “x.ui.pushButton” and documentation of the PyQt elements is found at <http://pyqt.sourceforge.net/Docs/PyQt4/classes.html>

The following example are found in UseQtGuiLoader.py

In order to integrate MyMainWindowUI in Python, subclass QtMainWindowLoader from QtGuiLoader and instantiate with “MyMainWindowUI” as ui\_module argument.

|  |
| --- |
| import MyMainWindowUI  from QtGuiLoader import QtMainWindowLoader  class **MyMainWindowWithMenu**(QtMainWindowLoader):  def **\_\_init\_\_**(*self*):  QtMainWindowLoader.\_\_init\_\_(*self*, MyMainWindowUI)    def **actionSayHello**(*self*):  print *"hello"*  MyMainWindowWithMenu().start() |

In this example the action “actionSayHello” can be invoked from the menu, by its shortcut or by the pushButton