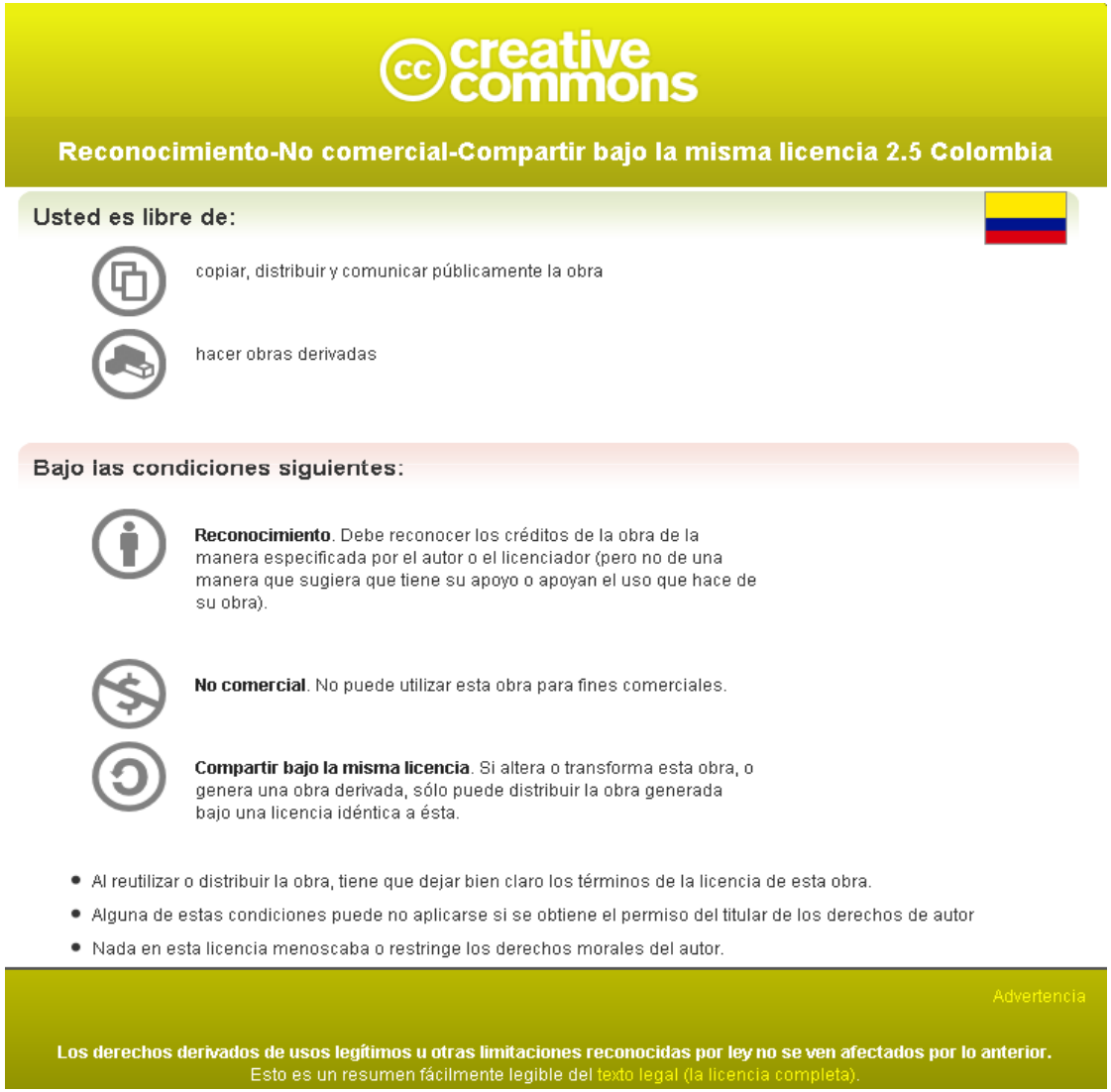
Interfaces Graficas de Usuario en Python: Primeros paso en PyQT4

JESSE PADILLA AGUDELO

Ingeniero Electrónico

Licencia de la Presentación



Objetivos

- Desarrollar Interfaces Graficas de Usuario en Python usando la biblioteca multiplataforma ***QT***, específicamente su binding para ***Python***, ***PyQT*** en su versión 4.

Introducción

- Con el propósito de crear interfaces gráficas de usuario, en ***Python*** podemos elegir entre varias bibliotecas, tales como ***PyGTK, wxPython, PyQt, Tkinter*** entre otros, en esta presentación analizaremos los aspectos básicos de ***PyQt*** y como crear interfaces graficas fácil y rápidamente usando ***Python***.

Índice

1. Introducción a QT y PyQT
2. Primeros Pasos
   - Ventanas
   - Botones
   - Menús
   - Cajas de Entrada
3. QT Designer

QT

- ***QT*** es una biblioteca multiplataforma para desarrollar interfaces gráficas de usuario. La biblioteca la desarrolla la que fue su creadora, la compañía noruega ***Trolltech***, actualmente renombrada a ***QT Software***, y que desde junio de 2008 es propiedad de ***Nokia***. Utiliza el lenguaje de programación C++ de forma nativa y además existen múltiples bindings para otros lenguajes, como:
  - Python (PyQt)
  - Perl (PerlQt)
  - Ruby (QtRuby)
  - Mono (Qyoto)
  - Java (Qt Jambi)
  - Gambas (gb.qt)
  - PHP (PHP-Qt)

PyQT

- PyQt es un conjunto de herramientas para crear aplicaciones GUI.
- Es una fusión entre Python y la biblioteca QT.
- El sitio oficial de casa para PyQt está en www.riverbankcomputing.co.uk, Fue desarrollado por Phil Thompson.

PyQT

- Multiplataforma, PyQT esta disponible en Windows, GNU/Linux, MacOS.
- Es Software Libre, licenciado bajo la Licencia GNU/GPL.
- Versión actual PyQT4

PyQT

- Se estructura en Python como un conjunto de módulos con mas de 300 clases y 6000 métodos.
- Estos módulos son:
  1. QtGui
  2. QtCore
  3. QtNetwork
  4. QtXml
  5. QtSvg
  6. QtOpenGL
  7. QtSQL

PyQt

- ***QtGui*** -> Componentes gráficos y clases relacionadas.
- ***QtCore*** -> Núcleo de ***Qt***, no contiene funciones graficas, trabaja con funciones de tiempo, directorios, tipos de datos, urls, entre otros.
- ***QtNetwork*** -> Modulo que contiene un conjunto de clases para trabajar en entornos de red.

PyQt

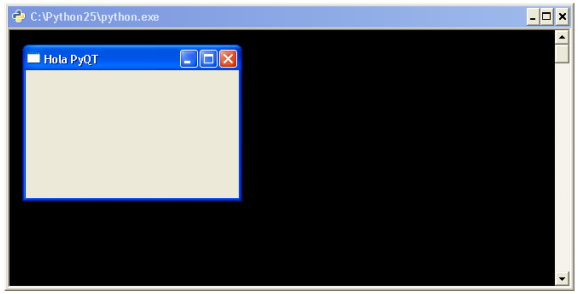
- ***QtXml*** -> Conjunto de clases para trabajar con archivos XML.
- ***QtSvg*** -> Conjunto de clases para visualizar imágenes SVG.
- ***QtOpenGL*** -> Conjunto de clases para renderizado 2D y 3D usando OpenGL
- ***QtSql*** -> Conjunto de clases para trabajar con bases de datos.

Primeros Pasos

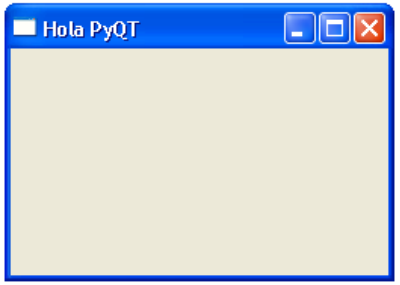
- Descargar e Instalar el interprete de Python, en este caso Python version 2.5.4
- Descargar e Instalar la biblioteca PyQt4, PyQt GPL version 4.4.3
- Un IDE o en su defecto un editor de texto listo.

Primeros Pasos: extension ***.pyw***

- Las aplicaciones en Python que contengan interfaz grafica de usuario, en sistemas Windows los guardaremos con extensión ***.pyw*** con el propósito de que la consola de comandos de este no sea visible.



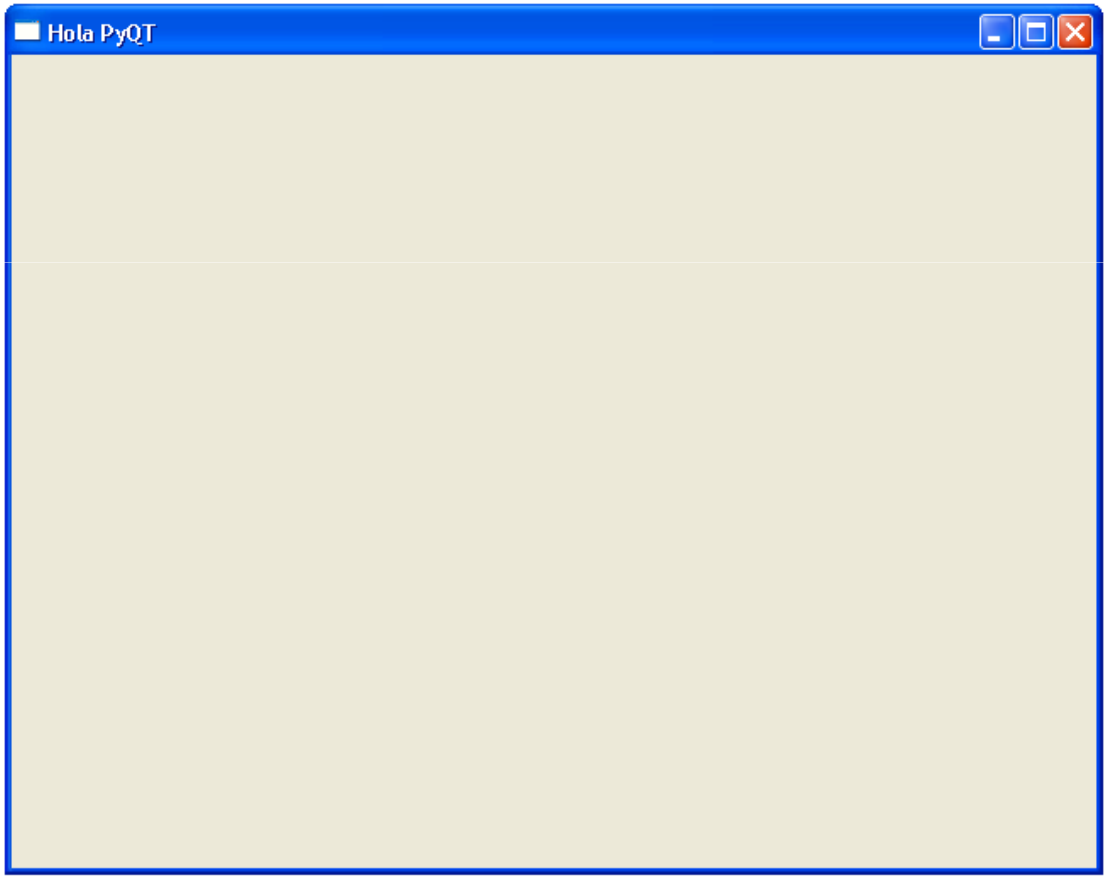
.py



.pyw

Primeros Pasos: Reto #1

- Crear una ventana usando Python y PyQt4.



Primeros Pasos: Reto #1

```
1  import sys
2  from PyQt4 import QtGui
3
4  app = QtGui.QApplication(sys.argv)
5
6  widget = QtGui.QWidget()
7  widget.resize(640, 480)
8  widget.setWindowTitle('Hola PyQT')
9  widget.show()
10
11 sys.exit(app.exec_())
```

Primeros Pasos: Reto #1

- Ahora entendamos las líneas de esta aplicación.

  import sys

  from PyQt4 import QtGui

- Las líneas [1] y [2] permiten importar los módulos necesarios para usar ***PyQt***, el modulo ***sys*** deja disponibles todas las funciones y atributos de ***Python***, el modulo ***QtGui*** como lo mencionado anteriormente importa los componentes gráficos de ***Qt***.

Primeros Pasos: Reto #1

app = QtGui.Qapplication(sys.argv)

- Línea [4], todas las aplicaciones en ***PyQt*** necesitan crear un objeto de tipo aplicación, este objeto se encuentra en el modulo ***QtGui***
- ***sys.argv*** – es un parámetro para recibir, una lista de argumentos desde la línea de comandos al ser lanzada.

Primeros Pasos: Reto #1

widget = QtGui.QWidget()

Línea 6, Qwidget es una clase de todos los objetos de la interfaz de usuario de ***PyQt***, no necesita parámetros para inicializar el objeto.

widget.resize(640,480)

Este método redimensiona el widget, a la resolución indicada.

widget.setWindowTitle('Hola PyQt')

Coloca un titulo al widget en la barra de titulo.

widget.show()

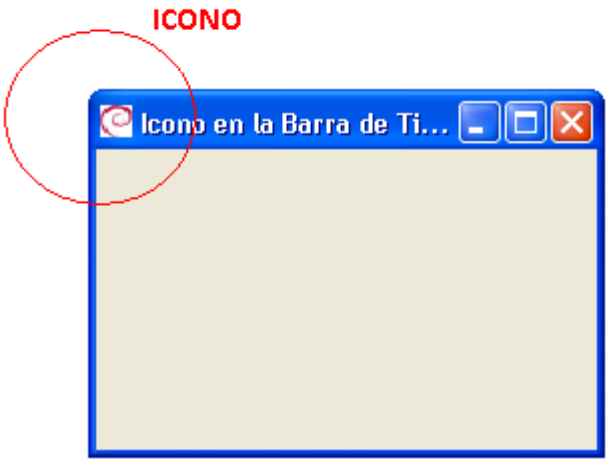
Muestra el widget en la ventana.

sys.exit(app.exec_())

Entramos al main loop de la aplicación.

Primeros Pasos: Reto #2

- Crear una ventana usando Python y PyQt4, con un icono de la aplicación en la barra de titulo.



Primeros Pasos: Reto #2

```
1  import sys
2  from PyQt4 import QtGui
3
4  class Icono(QtGui.QWidget):
5      def __init__(self, parent=None):
6          QtGui.QWidget.__init__(self, parent)
7
8          self.setGeometry(300, 300, 250, 150)
9          self.setWindowTitle('Icono en la Barra de Titulo')
10         self.setWindowIcon(QtGui.QIcon('icono.jpg'))
11
12 app = QtGui.QApplication(sys.argv)
13 icon = Icono()
14 icon.show()
15 sys.exit(app.exec_())
```

Primeros Pasos: Reto #2

```
class Icono(QtGui.QWidget):
    def __init__(self, parent = None):
        QtGui.Qwidget.__init__(self, parent)
```

- Linea [4], Heredamos icono de la clase QtGui.Qwidget.
- Linea [5] y [6], definimos el método __init__ de la clase Icono, dentro de este llamamos el método __init__ de la clase padre.

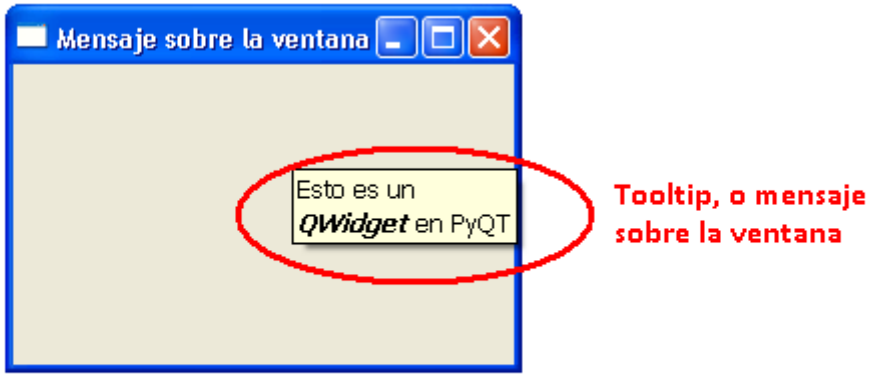
Primeros Pasos: Reto #2

Self.setGeometry(300,300,250,150)
Self.setWindowTitle('Icono en la Barra de Titulo')
Self.setWindowIcon(QtGui.QIcon('icono.jpg'))

Son 3 Clases heredadas de la clase QtGui.Qwidget, setGeometry como primer parámetro recibe los datos de x,y donde va a ubicar la ventana en la pantalla, setWindowTitle coloca un titulo a la ventana en la barra de titulo, y setWindowIcon establece el icono de la aplicación, esta necesita crear un objeto de tipo icono QIcon que reciba la ruta de la imagen.

Primeros Pasos: Reto #3

- Crear una ventana usando Python y PyQt4, al arrastrar el mouse sobre esta se debe desplegar un mensaje de información.



Primeros Pasos: Reto #3

```
1  import sys
2  from PyQt4 import QtGui
3  from PyQt4 import QtCore
4
5  class mensaje(QtGui.QWidget):
6      def __init__(self, parent=None):
7          QtGui.QWidget.__init__(self, parent)
8
9          self.setGeometry(300, 300, 250, 150)
10         self.setWindowTitle('Mensaje sobre la ventana')
11
12         self.setToolTip('Esto es un <b><i>QWidget</i></b> en PyQT')
13         QtGui.QToolTip.setFont(QtGui.QFont('OldEnglish', 10))
14
15 app = QtGui.QApplication(sys.argv)
16 tooltip = mensaje()
17 tooltip.show()
18 app.exec_()
```

Primeros Pasos: Reto #3

self.setToolTip('Esto es un Qwidget en PyQt')

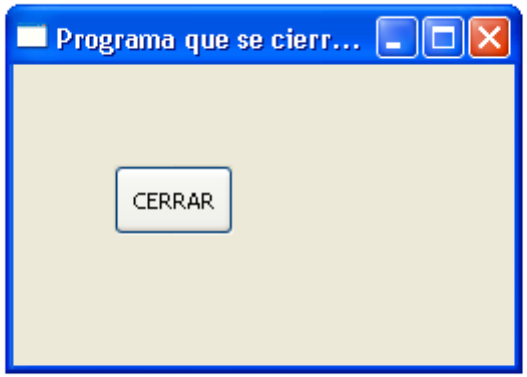
- Los tooltips se crean a traves del metodo setToolTip(), los mensajes pueden usar formato de texto enriquecido RTF

QtGui.QToolTip.setFont(QtGui.QFont('OldEnglish',10))

- Este método cambia la fuente a OldEnglish y le asigna un tamaño de 10

Primeros Pasos: Reto #4

- Crear una ventana que contenga un botón y que al recibir una señal de cerrar esta termine la aplicación.



Primeros Pasos: Reto #4

```
1  import sys
2  from PyQt4 import QtGui, QtCore
3
4  class QuitButton(QtGui.QWidget):
5      def __init__(self, parent=None):
6          QtGui.QWidget.__init__(self, parent)
7
8          self.setGeometry(300, 300, 250, 150)
9          self.setWindowTitle('Programa que se cierra con un boton')
10
11         quit = QtGui.QPushButton('CERRAR', self)
12         quit.setGeometry(50, 50, 60, 35)
13
14         self.connect(quit, QtCore.SIGNAL('clicked()'),
15             QtGui.qApp, QtCore.SLOT('quit()'))
16
17 app = QtGui.QApplication(sys.argv)
18 qb = QuitButton()
19 qb.show()
20 sys.exit(app.exec_())
```

Primeros Pasos: Reto #4

quit = QtGui.QPushButton('Cerrar', self)

quit.setGeometry(50,50,65,35)

- Creamos un objeto de tipo botón y asignamos la posición y el tamaño de este sobre la ventana.

Primeros Pasos: Reto #4

self.connect(quit, QtCore.SIGNAL('clicked()'), QtGui.qApp, QtCore.SLOT('quit()'))

- PyQt4 maneja los eventos que se presenten en una aplicación con un mecanismo de ***Señales (SIGNAL) y Ranuras(SLOT)***. En este caso cuando hacemos click sobre el botón se emite una señal de click, el ***SLOT*** relaciona un objeto emisor en este caso el botón con un objeto receptor (la aplicación) con la acción a realizar dado dicho evento, en este caso cerrar la aplicación (el método ***quit()***)

Primeros Pasos: Reto #5

- Crear una ventana heredando la clase ***MainWindow*** de ***PyQt***, su método ***__init__*** debe recibir la resolución deseada de la ventana.
- Crear un botón sobre la ventana, su función será cerrar la ventana
- La aplicación también debe contener un menú desde el cual se pueda cerrar también el programa
- La aplicación debe mostrar una barra de estado sobre la ventana principal y cambiar dependiendo de donde se encuentre el cursor con la información de lo que esta señalando.

Primeros Pasos: Reto #5

Primeros Pasos: Reto #5

```python
import sys
from PyQt4 import QtGui, QtCore

class VentanaPrincipal(QtGui.QMainWindow):
    def __init__(self, ancho, largo):
        QtGui.QMainWindow.__init__(self)
        # Creo la ventana
        self.setGeometry(100,100,ancho,largo)
        self.setWindowTitle('Reto #5')
        self.statusBar().showMessage('Reto #5 Listo')
        self.setToolTip('Ventana')
        # Creo el Boton Salir
        bsalir = QtGui.QPushButton('Cerrar', self)
        bsalir.setToolTip('Boton Salir')
        bsalir.setGeometry((ancho - 50) / 2,(largo - 35) / 2,50,35)
        bsalir.setShortcut('Ctrl+Q')
        bsalir.setStatusTip('Boton Salir')
        self.connect(bsalir, QtCore.SIGNAL('clicked()'), QtGui.qApp, QtCore.SLOT('quit()'))
        # Creo el elemento del menu salir
        msalir = QtGui.QAction(QtGui.QIcon('icono.jpg'), 'Cerrar', self)
        msalir.setShortcut('Ctrl+Q')
        msalir.setStatusTip('Menu Salir')
        self.connect(msalir, QtCore.SIGNAL('triggered()'), QtGui.qApp,
            QtCore.SLOT('quit()'))
        # Creo una barra de menu, el menu archivo y a este le agrego el elemento salir
        menu = self.menuBar()
        menu.setToolTip('Menu')
        archivo = menu.addMenu('&Archivo')
        archivo.addAction(msalir)

app = QtGui.QApplication(sys.argv)
principal = VentanaPrincipal(320,280)
principal.show()
app.exec_()
```

Primeros Pasos: Reto #5

self.statusBar().showMessage('Reto #5 Listo')

- Este método muestra el mensaje Reto #5 Listo en la barra de estad, la barra de estado es la barra inferior izquierda donde se muestra información de la aplicación.

self.setToolTip('Ventana')

- Este método nos muestra en mensaje ventana al desplazar el cursor del mouse sobre la ventana.

Primeros Pasos: Reto #5

bsalir = QtGui.QPushButton('Cerrar', self)
bsalir.setToolTip('Boton Salir')
bsalir.setGeometry((ancho - 50) / 2,(largo - 35) / 2,50,35)
bsalir.setShortcut('Ctrl+Q')
bsalir.setStatusTip('Boton Salir')

- Creamos el botón bsalir, que es un objeto de tipo ***QPushButton***, colocamos un mensaje (‘Boton Salir’) que al desplazar el cursor del mouse sobre este sea visible, con ***setGeometry()*** ubicamos el botón en la ventana, el calculo presente lo centra en esta y además define su tamaño.
- El Método ***setShortcut()*** asigna un atajo del teclado al botón en este caso el atajo es presionar las teclas ***Ctrl + Q***
- Por ultimo asignamos un mensaje al pasarnos por el botón para la barra de estado con el método ***setStatusTip()***

Primeros Pasos: Reto #5

self.connect(bsalir, QtCore.SIGNAL('clicked()'), QtGui.qApp, QtCore.SLOT('quit()'))

- Asignamos la señal de click del botón salir al slot quit() que cierra la aplicación.

msalir = QtGui.QAction(QtGui.QIcon('icono.jpg'), 'Cerrar', self)
msalir.setShortcut('Ctrl+Q')
msalir.setStatusTip('Menu Salir')
self.connect(msalir, QtCore.SIGNAL('triggered()'), QtGui.qApp, QtCore.SLOT('quit()'))

- Creamos un objeto de tipo QAction, este se mostrar en la ventana como ‘Cerrar’, el cual se encargara de cerrar la aplicación desde el menú de la ventana.

Primeros Pasos: Reto #5

menu = self.menuBar()

menu.setToolTip('Menu')

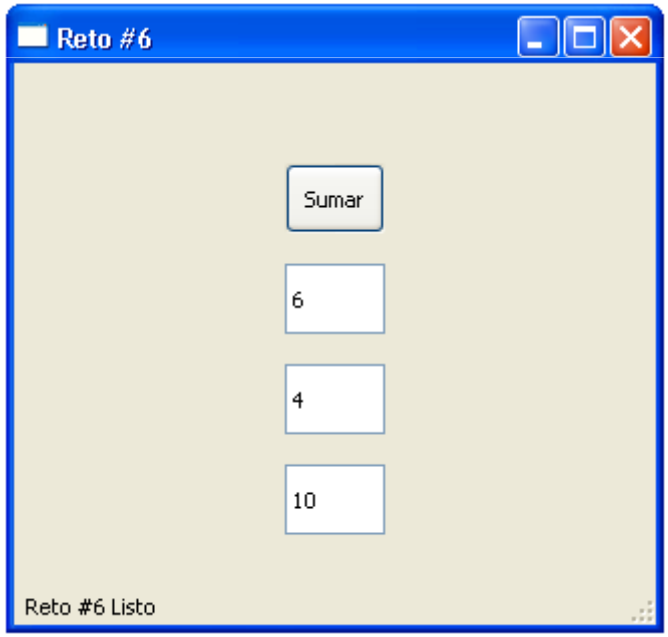
archivo = menu.addMenu('&Archivo')

archivo.addAction(msalir)

- Creamos un objeto menú de tipo ***menuBar()***, a este le agregamos un menú te tipo archivo, y por ultimo le agregamos a este la acción salir.

Primeros Pasos: Reto #6

- Desarrollar una aplicación en ***PyQT*** que en dos campos de texto reciba dos números y con un botón los sume y muestra la respuesta en un tercer espacio.



Primeros Pasos: Reto #6

```
1  import sys
2  from PyQt4 import QtGui, QtCore
3
4  class InputDialog(QtGui.QMainWindow):
5      def __init__(self):
6          QtGui.QWidget.__init__(self)
7
8          self.setGeometry(100,100,320,280)
9          self.setWindowTitle('Reto #6')
10         self.statusBar().showMessage('Reto #6 Listo')
11         self.setToolTip('Ventana Principal')
12
13         self.boton = QtGui.QPushButton('Sumar', self)
14         self.boton.setGeometry(135, 50, 50, 35)
15         self.connect(self.boton, QtCore.SIGNAL('clicked()'), self.sumar)
16
17         self.edit1 = QtGui.QLineEdit(self)
18         self.edit1.setGeometry(135, 100, 50, 35)
19
20         self.edit2 = QtGui.QLineEdit(self)
21         self.edit2.setGeometry(135, 150, 50, 35)
22
23         self.edit2 = QtGui.QLineEdit(self)
24         self.edit2.setGeometry(135, 150, 50, 35)
25
26         self.edit3 = QtGui.QLineEdit(self)
27         self.edit3.setGeometry(135, 200, 50, 35)
28
29     def sumar(self):
30         self.edit3.setText( str( int(self.edit1.text()) + int(self.edit2.text())) )
31
32 app = QtGui.QApplication(sys.argv)
33 icon = InputDialog()
34 icon.show()
35 app.exec_()
```

Primeros Pasos: Reto #6

self.edit1 = QtGui.QLineEdit(self)

self.edit1.setGeometry(135, 100, 50, 35)

- Los ***LineEdit*** son objetos que permiten capturar texto desde la interfaz de usuario, en este caso creamos edit1 y edit2 que son de tipo ***QLineEdit()*** y los ubicamos en la pantalla y asignamos su tamaño con el método ***setGeometry()***

Primeros Pasos: Reto #6

- El método sumar, suma el contenido de los LineEdit 1 y 2 y permite visualizar la respuesta a través del LineEdit 3.

def sumar(self):

self.ventana.edit3.setText(str(int(self.ventana.edit1.text()) + int(self.ventana.edit2.text())))

- ***self.ventana.edit1.text()*** y ***self.ventana.edit2.text()*** son métodos que permiten capturar el texto contenido en los Line Edits, por defecto este texto es una cadena, por lo cual para operar los dos valores debemos pasarlos a variables numéricas en este caso tipo entero(***int***).
- El método ***self.ventana.edit3.setText()*** nos permite colocar una cadena de texto sobre el Line Edit, pero este recibe cadena de caracteres y no números por lo cual debemos convertir el resultado entero a cadena con la sentencia ***str().***

Qt Designer

Code less.
Create more.
Deploy everywhere.

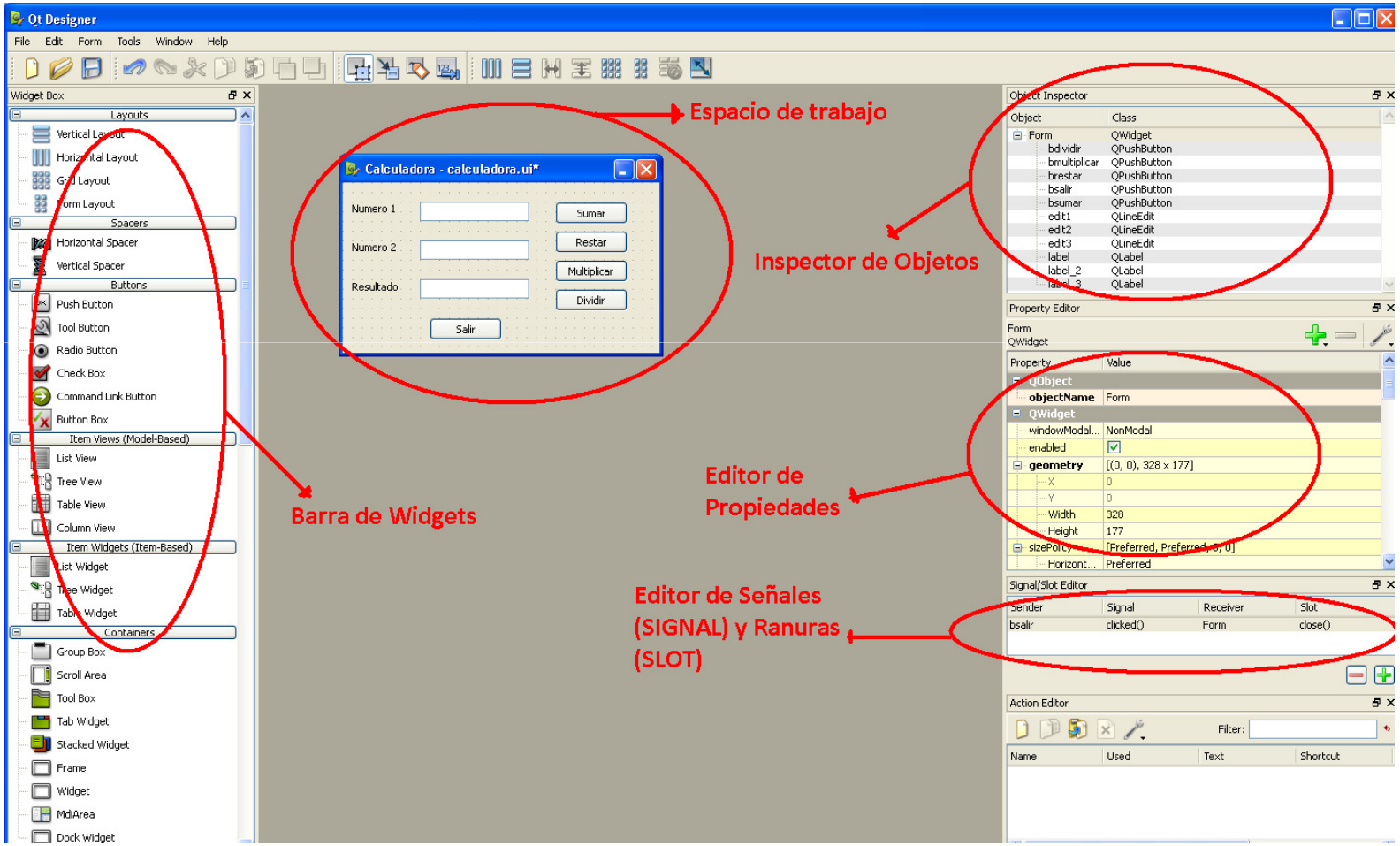
Qt Designer

- Es una herramienta potente de la que dispone Qt, la cual permite diseñar de forma muy sencilla, rápida y eficiente interfaces graficas de usuario, esta aplicación permite construir un GUI de forma grafica e intuitiva, sin tener que pensar desde el código como se vera esta y empezar a jugar con coordenadas y pixeles desde el fuente de la aplicación.

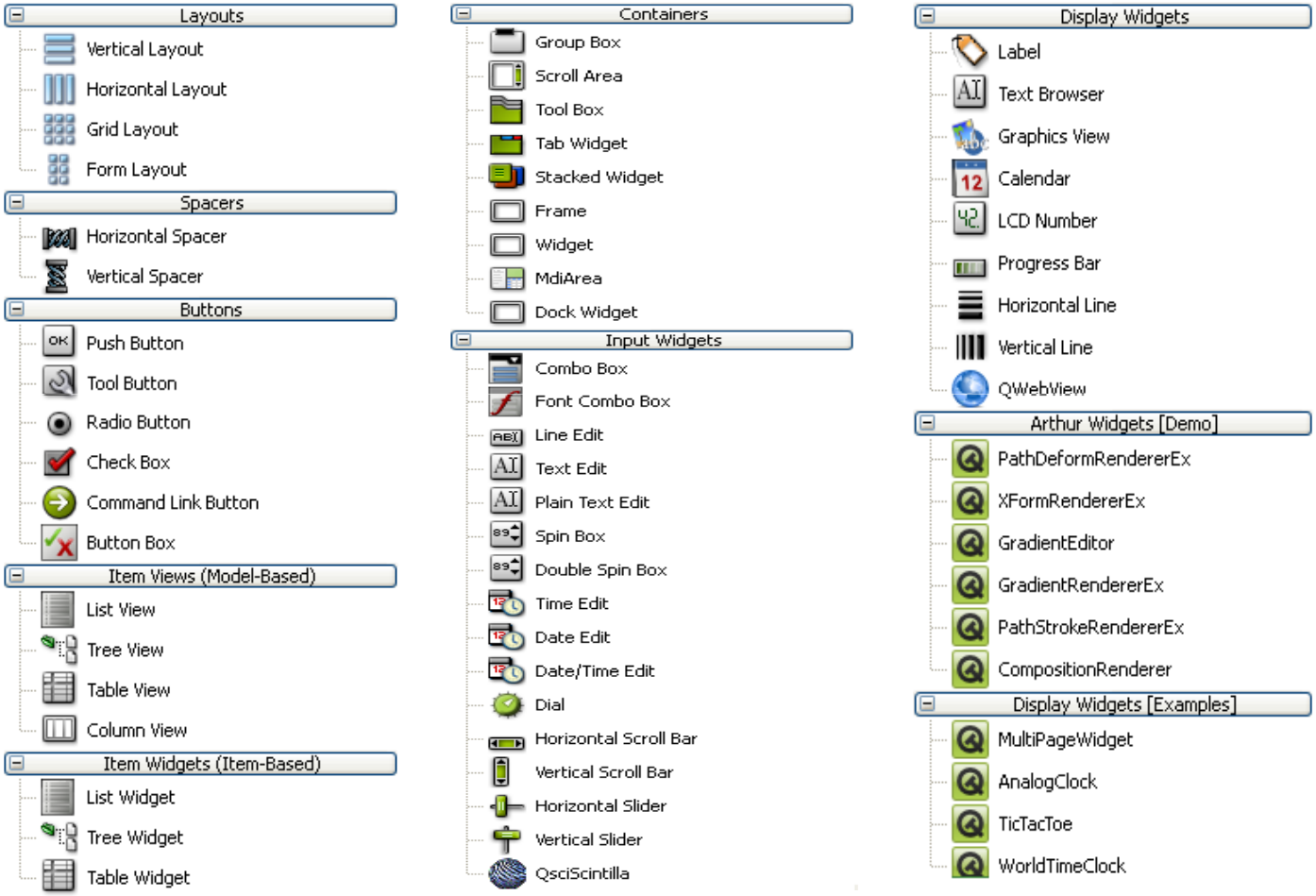
Características de Qt Designer

- Ofrece una caja muy rica de componentes para interfaces de usuario.
- Posee un inspector de objetos para explorar los componentes usados en nuestro desarrollo.
- Posee un editor de propiedades, que nos permite cambiar las características de los objetos presentes en nuestra interfaz, características como el nombre del objeto, el texto que presenta este en la interfaz, fuente, tamaño, entre otros.
- Posee un editor de Señales(SIGNAL) y ranuras (SLOT), donde podemos asignar comportamientos predefinidos a ciertos objetos de nuestra interfaz, aunque es mas versátil establecerlos desde el código fuente de nuestra aplicación.

Interfaz de Qt Designer



Componentes de Qt Designer

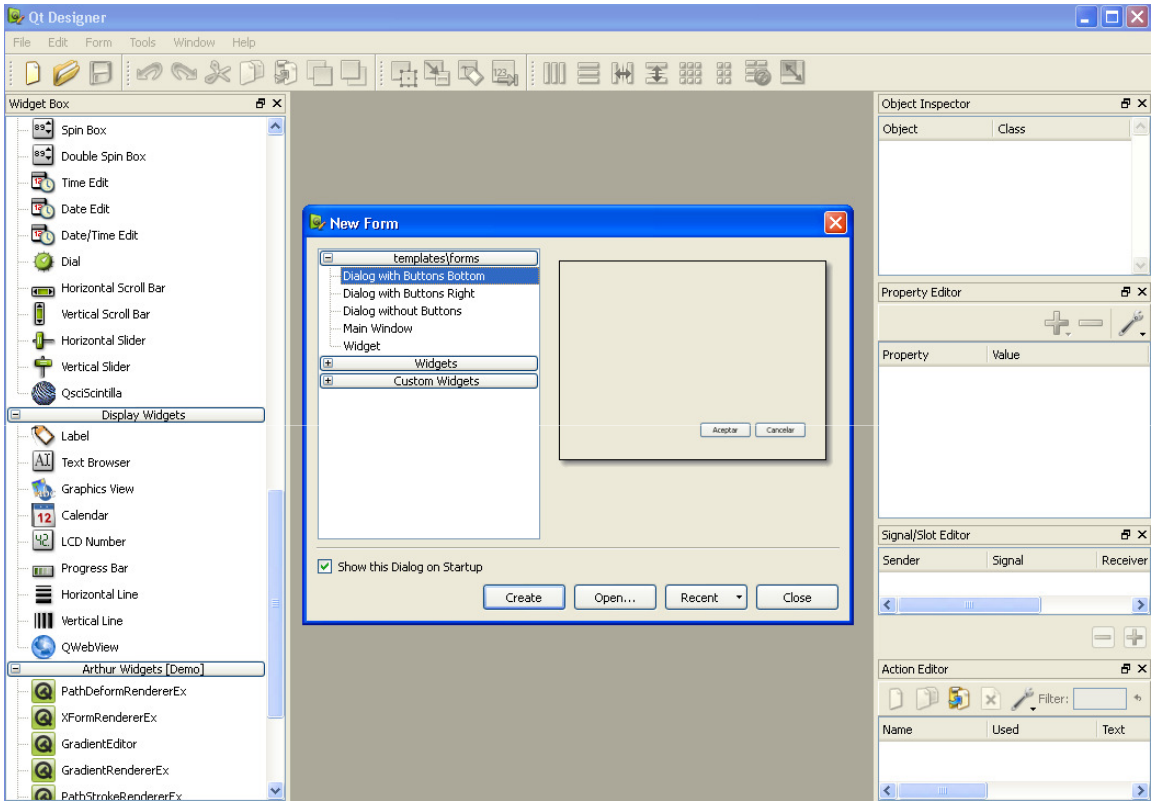


Exploremos Qt Designer con un pequeño reto

- Desarrollar una aplicación con Python, PyQt y Qt Designer que realice las operaciones basicas (+ - * /) entre dos numeros y permita visualizar el resultado.

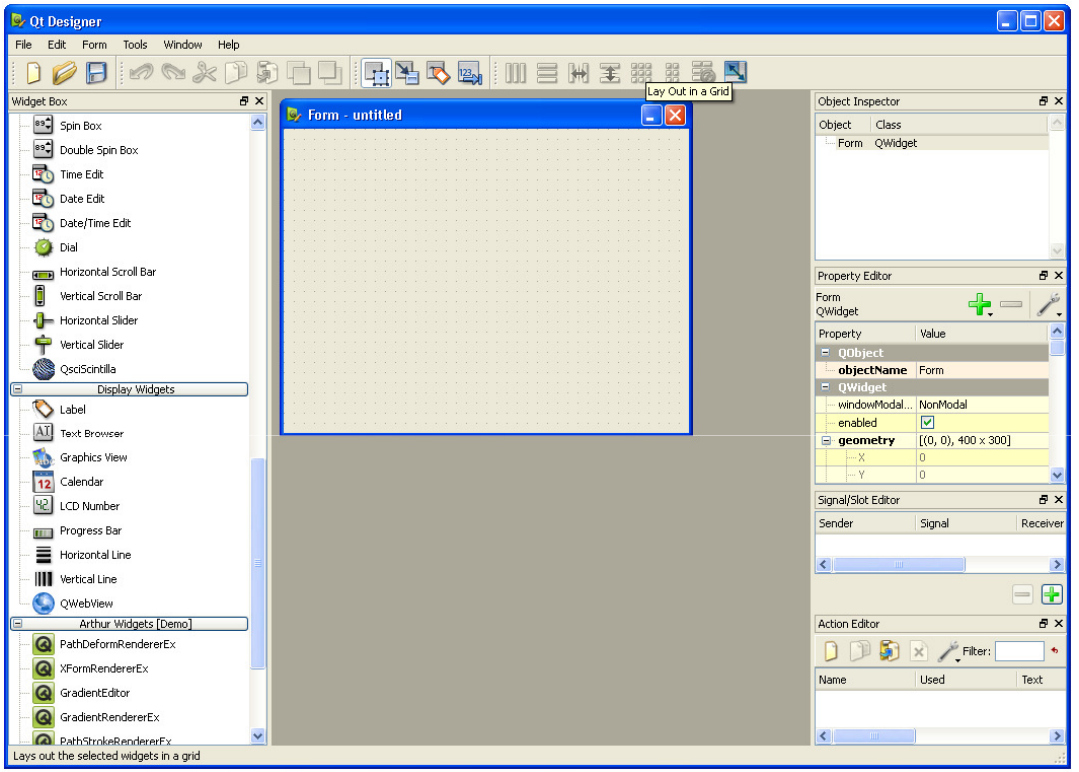
Calculadora

- Abrimos Qt Designer
- Podemos escoger entre ventanas de dialogo, widget y ventana principal, la ventana principal ya es una interfaz mas elaborada que contiene menús y otros componentes, el widget es una ventana mas simple desde la cual podemos trabajar desde cero, las ventanas de dialogo son ventanas sencillas muy especializadas y su propósito es recoger información especifica.



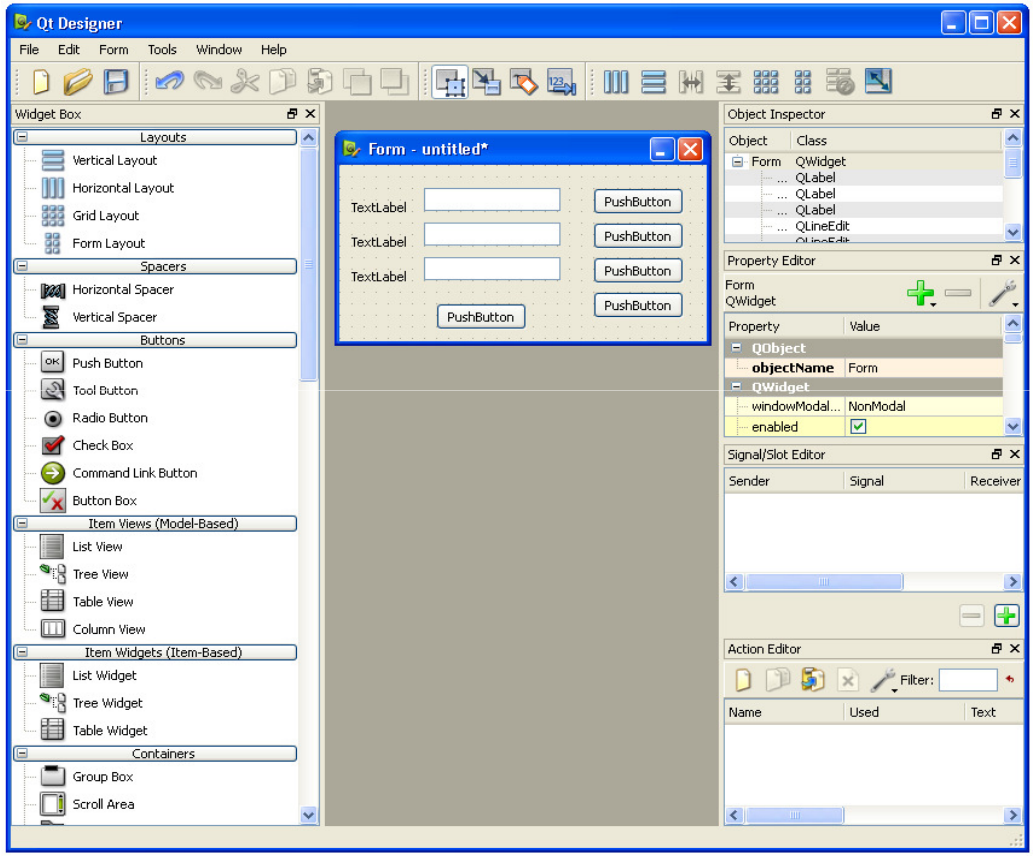
Calculadora

- Al seleccionar Widget se nos habilita el espacio de trabajo para crear nuestro GUI sobre el formulario que se nos presenta.



Calculadora

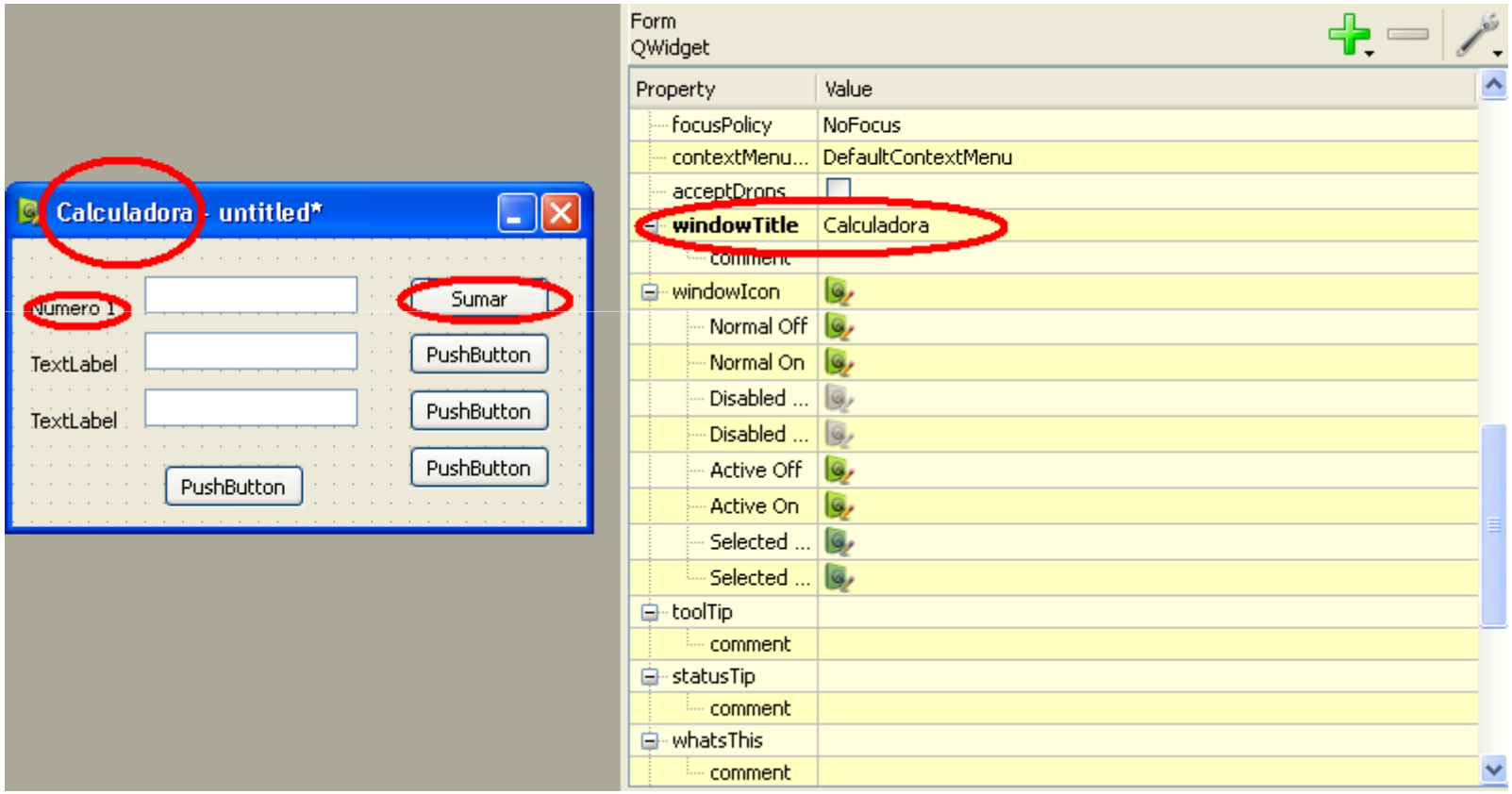
De la barra de Widget arrastramos al formulario los objetos que deseemos o necesitemos en nuestra interfaz de usuario, en este caso arrastramos 3 objetos de tipo ***Label***, ubicados en el sección ***Display Widget*** del ***Widget Box***, estos objetos nos sirven para visualizar texto no para ingresar, para ingresar texto arrastramos 3 objetos de tipo ***Line Edit***, este nos permite ingresar texto y visualizar este se encuentra en la sección ***Input Widgets***, por ultimo ingresamos 5 objetos de tipo ***Push Button*** que se asociaran a cada operación matemática mas un botón salir, este se encuentra en la seccion ***Buttons***.



Calculadora

- Hasta acá nuestra interfaz ha sido muy sencillo construirla pero aun es muy simple y necesita pulir detalles como que los ***labels*** indique con un texto que tipo de información deseamos que ingrese en los ***edit***, los botones indiquen a través de su nombre que operación realizan, esto lo conseguimos seleccionando objeto por objeto y cambiando sus características en el editor de propiedades, veamos como:

Calculadora



Calculadora

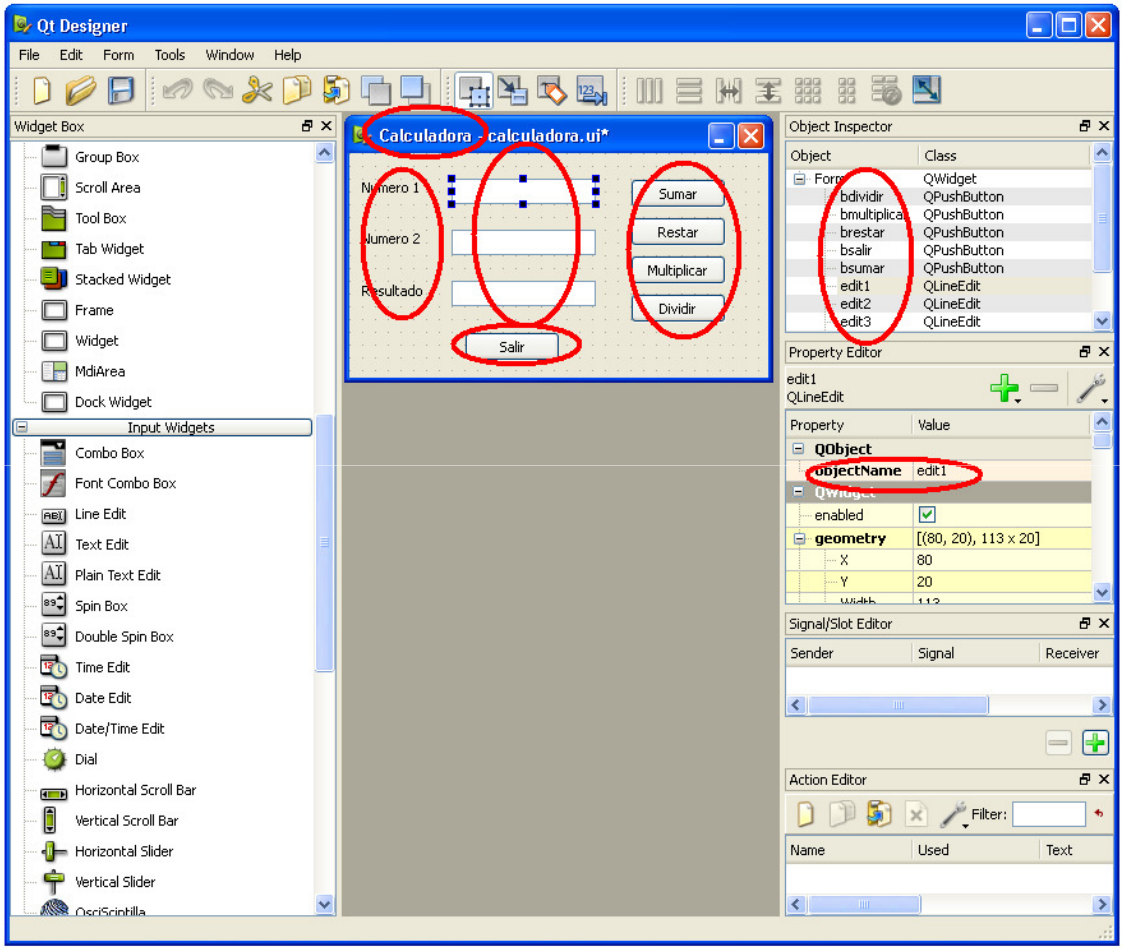
- Propiedades Importantes a modificar:
  - Formulario
    - ***WindowTitle*** – Titulo de la ventana visible en la barra de titulo
  - Botones
    - ***ObjectName*** – Nombre con el cual manipularemos el objeto desde el código de la aplicación, se generan unos poder defecto al crear el objeto es conveniente cambiarlo para administrar adecuadamente la interfaz
    - ***Text*** – Propiedad que nos cambiara el nombre del botón visible al usuario en la interfaz.
  - Labels
    - ***Text*** – Igual que la propiedad ***Text*** del botón
  - Line Edit
    - ***ObjectName –*** Igual que la propiedad ***ObjectName*** del botón

Calculadora

Después de editar las propiedades deseadas tendremos algo como lo presentado en esta imagen.

Lo objetos tienen múltiples propiedades, muy sencillas de usar solo es explorarlas y con unos pasos muy simples veremos sus resultados.

En el inspector de objetos veremos los nombres de los objetos creados tal cual los llamamos nosotros, y asi los invocaremos desde el codigo



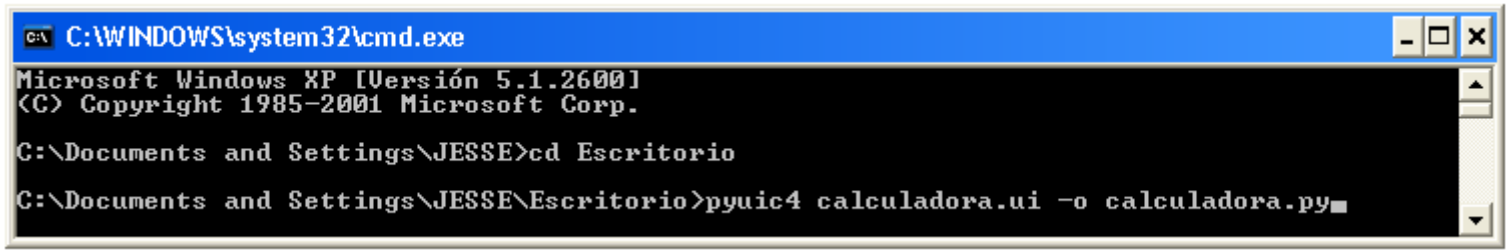
Calculadora: Usando pyuic

- Luego de haber terminado nuestra interfaz es hora de guardarla y usarla desde Python, nos vamos a ***File -> Save as*** y guardamos el archivo con el nombre deseado, esto nos generara un archivo de extensión **.ui**, el cual no nos sirve en Python por lo cual debemos transformarlo usando el comando ***pyuic*** desde la shell en Linux o el terminal de comando de Windows.

Calculadora

- Nos ubicamos en la ruta en la que guardamos nuestra interfaz, y ahora ejecutamos el comando ***pyuic*** en este caso como es PyQt4 usamos el comando,

pyuic4 archivo_origen.ui –o archivo_salida.py



Calculadora

Luego de haber ejecutado el comando ***pyuic*** ya tenemos un archivo de Python el cual contiene una clase con todas las características de nuestro ventana.

Si lo analizamos bien es una clase como la que trabajamos en los primeros retos, solo contiene los elementos en pantalla como se llaman los objetos y sus coordenadas, ahora desde nuestro archivo principal debemos solo incluir este archivo y cargar esta clase como objeto a manipular.

Solo nos resta manipular las señales y los slots y nuestra aplicación estará terminada.

```
from PyQt4 import QtCore, QtGui

class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(328, 177)
        self.edit1 = QtGui.QLineEdit(Form)
        self.edit1.setGeometry(QtCore.QRect(80, 20, 113, 20))
        self.edit1.setObjectName("edit1")
        self.edit2 = QtGui.QLineEdit(Form)
        self.edit2.setGeometry(QtCore.QRect(80, 60, 113, 20))
        self.edit2.setObjectName("edit2")
        self.edit3 = QtGui.QLineEdit(Form)
        self.edit3.setGeometry(QtCore.QRect(80, 100, 113, 20))
        self.edit3.setObjectName("edit3")
        self.label = QtGui.QLabel(Form)
        self.label.setGeometry(QtCore.QRect(10, 20, 46, 14))
        self.label.setObjectName("label")
        self.label_2 = QtGui.QLabel(Form)
        self.label_2.setGeometry(QtCore.QRect(10, 60, 46, 14))
        self.label_2.setObjectName("label_2")
        self.label_3 = QtGui.QLabel(Form)
        self.label_3.setGeometry(QtCore.QRect(10, 100, 51, 16))
        self.label_3.setObjectName("label_3")
        self.bsumar = QtGui.QPushButton(Form)
        self.bsumar.setGeometry(QtCore.QRect(220, 20, 75, 23))
        self.bsumar.setObjectName("bsumar")
        self.brestar = QtGui.QPushButton(Form)
        self.brestar.setGeometry(QtCore.QRect(220, 50, 75, 23))
        self.brestar.setObjectName("brestar")
        self.bmultiplicar = QtGui.QPushButton(Form)
        self.bmultiplicar.setGeometry(QtCore.QRect(220, 80, 75, 23))
        self.bmultiplicar.setObjectName("bmultiplicar")
        self.bdividir = QtGui.QPushButton(Form)
        self.bdividir.setGeometry(QtCore.QRect(220, 110, 75, 23))
        self.bdividir.setObjectName("bdividir")
        self.bsalir = QtGui.QPushButton(Form)
        self.bsalir.setGeometry(QtCore.QRect(90, 140, 75, 23))
        self.bsalir.setObjectName("bsalir")

        self.retranslateUi(Form)
        QtCore.QMetaObject.connectSlotsByName(Form)

    def retranslateUi(self, Form):
        Form.setWindowTitle(QtGui.QApplication.translate("Form", "Calculadora", None, QtGui.QApplication.UnicodeUTF8))
        self.label.setText(QtGui.QApplication.translate("Form", "Numero 1", None, QtGui.QApplication.UnicodeUTF8))
        self.label_2.setText(QtGui.QApplication.translate("Form", "Numero 2", None, QtGui.QApplication.UnicodeUTF8))
        self.label_3.setText(QtGui.QApplication.translate("Form", "Resultado", None, QtGui.QApplication.UnicodeUTF8))
        self.bsumar.setText(QtGui.QApplication.translate("Form", "Sumar", None, QtGui.QApplication.UnicodeUTF8))
        self.brestar.setText(QtGui.QApplication.translate("Form", "Restar", None, QtGui.QApplication.UnicodeUTF8))
        self.bmultiplicar.setText(QtGui.QApplication.translate("Form", "Multiplicar", None, QtGui.QApplication.UnicodeUTF8))
        self.bdividir.setText(QtGui.QApplication.translate("Form", "Dividir", None, QtGui.QApplication.UnicodeUTF8))
        self.bsalir.setText(QtGui.QApplication.translate("Form", "Salir", None, QtGui.QApplication.UnicodeUTF8))
```

Calculadora

```
1  import sys
2  from PyQt4 import QtCore, QtGui
3  from calculadora import Ui_Form
4
5  class Principal(QtGui.QWidget):
6      def __init__(self):
7          QtGui.QWidget.__init__(self)
8
9          self.ventana = Ui_Form()
10         self.ventana.setupUi(self)
11
12         self.connect(self.ventana.bsalir, QtCore.SIGNAL('clicked()'), QtCore.SLOT('close()'))
13
14         self.connect(self.ventana.bsumar, QtCore.SIGNAL('clicked()'), self.sumar)
15         self.connect(self.ventana.brestar, QtCore.SIGNAL('clicked()'), self.restar)
16         self.connect(self.ventana.bmultiplicar, QtCore.SIGNAL('clicked()'), self.multiplicar)
17         self.connect(self.ventana.bdividir, QtCore.SIGNAL('clicked()'), self.dividir)
18
19     def sumar(self):
20         self.ventana.edit3.setText( str( float(self.ventana.edit1.text()) + float(self.ventana.edit2.text())) )
21     def restar(self):
22         self.ventana.edit3.setText( str( float(self.ventana.edit1.text()) - float(self.ventana.edit2.text())) )
23     def multiplicar(self):
24         self.ventana.edit3.setText( str( float(self.ventana.edit1.text()) * float(self.ventana.edit2.text())) )
25     def dividir(self):
26         self.ventana.edit3.setText( str( float(self.ventana.edit1.text()) / float(self.ventana.edit2.text())) )
27
28 def main():
29     app = QtGui.QApplication(sys.argv)
30     ventana = Principal()
31     ventana.show()
32     sys.exit(app.exec_())
33
34 if __name__ == "__main__":
35     main()
36
```

Calculadora

import sys
from PyQt4 import QtCore, QtGui
from calculadora import Ui_Form

- Importamos las módulos con los que veníamos trabajando anteriormente pero también importamos del modulo generado por el ***pyuic*** (calculadora.py) la clase que describe la interfaz de usuario si nos devolvemos un poco al código de esta interfaz de usuario notaremos que la clase se llama ***Ui_Form***

Calculadora

```
class Principal(QtGui.QWidget):
    def __init__(self):
        QtGui.QWidget.__init__(self)
        self.ventana = Ui_Form()
        self.ventana.setupUi(self)
```

- Como veníamos trabajando creamos una clase heredada de la clase ***QtGui.Qwidget*** si en la interfaz hubiésemos trabajado con una MainWindow pues acá heredaríamos de ***QtGui.QMainWindow***, declaramos el método ***__init__*** y convocamos en este el método **__init__** de la clase padre.
- Creamos un atributo de la clase llamado ventana, es un objeto de tipo ***Ui_Form()*** es decir este objeto como tal es un objeto de tipo la interfaz que creamos con Qt Designer.

Calculadora

self.connect(self.ventana.bsalir, QtCore.SIGNAL('clicked()'), QtCore.SLOT('close()'))

- Asociamos al botón ***bsalir*** (ObjectName que asignamos manualmente) de la interfaz cuando se presentan la señal clicked() el método close que cierra la aplicación, este corresponde al slot.

self.connect(self.ventana.bsumar, QtCore.SIGNAL('clicked()'), self.sumar)

self.connect(self.ventana.brestar, QtCore.SIGNAL('clicked()'), self.restar)

self.connect(self.ventana.bmultiplicar, QtCore.SIGNAL('clicked()'), self.multiplicar)

self.connect(self.ventana.bdividir, QtCore.SIGNAL('clicked()'), self.dividir)

- Ahora a los botones (+ - * /) les asociamos a la señal ***clicked()*** los métodos sumar, restar, multiplicar, dividir que son miembros propios de la clase.

Calculadora

- Los métodos (sumar, restar, multiplicar y dividir) conservan la misma estructura solo cambia la operación matemática, razón por la cual solo analizaremos el método sumar:

def sumar(self):

self.ventana.edit3.setText(str(float(self.ventana.edit1.text()) + float(self.ventana.edit2.text())))

- ***self.ventana.edit1.text()*** y ***self.ventana.edit2.text()*** son métodos que permiten capturar el texto contenido en los Line Edits, por defecto este texto es una cadena, por lo cual para operar los dos valores debemos pasarlos a variables numéricas en este caso tipo flotante (***float***).
- El método ***self.ventana.edit3.setText()*** nos permite colocar una cadena de texto sobre el Line Edit, pero este recibe cadena de caracteres y no números por lo cual debemos convertir el resultado en flotante a cadena con la sentencia ***str().***

Calculadora

```
def main():
    app = QtGui.QApplication(sys.argv)
    ventana = Principal()
    ventana.show()
    sys.exit(app.exec_())

if __name__ == "__main__":
    main()
```

- Por ultimo creamos la aplicación que lanzara nuestro GUI y sus funcionalidades.

Preguntas

Referencias

- ***Python para todos***, Raúl González Duque
- ***Inmersión en Python***, MARK PILGRIM – FRANCISCO CALLEGO – RICARDO CÁRDENAS.
- ***Aprenda a Pensar Como un Programador con Python***, ALLEN DOWNEY - JEREY ELKNER – CHIRIS MEYER
- ***Creating GUI Applications with PyQt and Qt Designer***, DAVID BODDIE
- ***Desarrollo de Aplicaciones de Escritorio,*** DANIEL M. MALDONADO
- ***Tutorial de desarrollo de GUI en Python y Qt***, BULMA

Enlaces

- www.python.org
- http://zetcode.com/tutorials/pyqt4/
- http://www.riverbankcomputing.co.uk/static/Docs/PyQt4/pyqt4ref.html#pyqtconfig-classes


¡¡GRACIAS!!