ESTRUCTURA DE DATOS Y PROGRAMACIÓN ORIENTADA A OBJETOS

Tema Nº6:PyQt5 y QtDesigner

Indicador de logro Nº6:

Diseña formularios mediante el uso de los diferentes componentes de la librería PyQt5 a través de un lenguaje de programación.

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº6:**

PyQt5 y QtDesigner

**Subtema 6.1:**

PyQt5 y QtDesigner

**PYQT5 INTERFACES GRÁFICAS CON PYTHON**

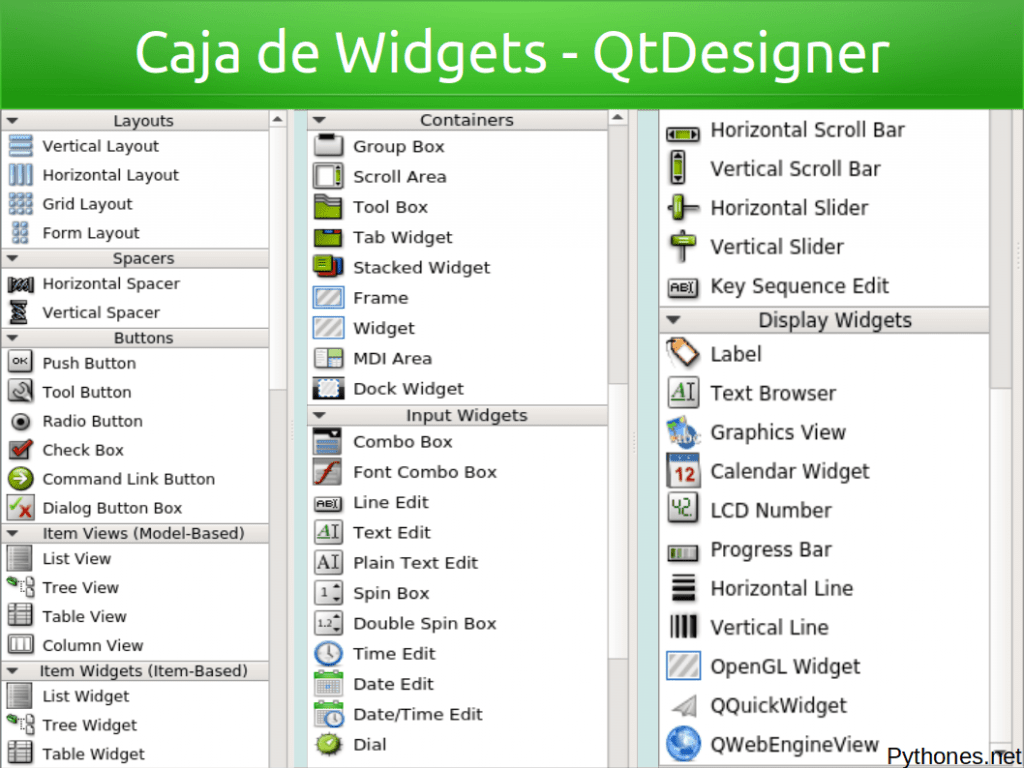
**¿Qué es PyQT5?**

Es un binding de la biblioteca gráfica QT para el lenguaje de programación Python. PyQT5 nos permite crear interfaces gráficas con python de manera rápida y sencilla, la legibilidad del código de Python hace que sea una tarea sumamente sencilla realizar interfaces gráficas, además que también posee una interfaz de diseño para crear nuestras interfaces gráficas.

La flexibilidad que tiene esta biblioteca, es que podemos diseñar por completo nuestras interfaces y luego comenzar a programar, esto es un punto muy importante, ya que si has utilizado Tkinter para realizar interfaces gráficas con Python sabrás de lo forzoso que es tener que ir diseñando y programando a la vez.

Con Tkinter se puede diseñar primero y luego programar, pero el punto que contradice todo acá es que en Tkinter el diseño no lo hacemos a través de una interfaz de diseño como la posee PyQT5, si no que el diseño lo hacemos escribiendo más código.

**ENTORNO DE QTDESIGNER**



**Actividad:**

Combinar 2 combobox

* Uso del componente Combo Box (caso desarrollado).



En el ejemplo, creare una pequeña aplicación gráfica que tendrá 2 combobox, una etiqueta (Resultado) y un botón (Ver).

La importancia de la aplicación será, que al seleccionar una opción del primer combobox, de diferentes opciones en el segundo combobox, y dependiendo de lo que seleccionemos en este último, será lo que se muestre en la etiqueta (Resultado) al apretar el botón (Ver).

El primer combobox tendrá 2 cursos de programación (Python y Java). El segundo combobox tendrá los nombres de los alumnos del curso (según el curso seleccionado).

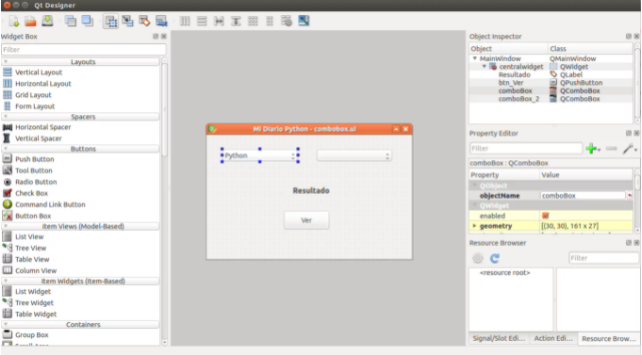
**Nombres de los objetos en QtDesigner:**

**Primer comboBox**: comboBox

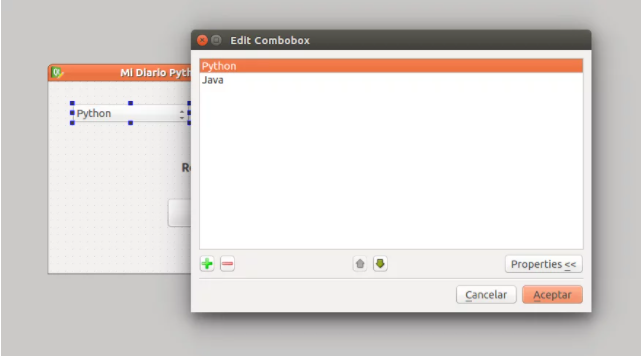
**Segundo comboBox**: comboBox\_2

**Etiqueta**: Resultado  
**Botón**: btn\_Ver

**Diseño Gráfico de la aplicación en QtDesigner**



A el **primer combobox** le agregamos los elementos (Python y Java) dando doble click sobre el mismo, en QtDesigner. Se verá un **símbolo de +** para agregar elementos.



**Guardar el proyecto**

Ahora, crearemos una carpeta en el escritorio con el nombre: combobox. Dentro de ella guardaremos el diseño de Designer con el nombre combobox.ui. Por último, dentro de la misma carpeta creamos un archivo python con el nombre: combobox.py . Abrimos el archivo y copiamos lo siguiente:

#!/usr/bin/python

# -\*- coding: utf-8 -\*-

# www.pythondiario.com

# Combinar 2 ComboBox

import sys

from PyQt4 import QtCore, QtGui, uic

# Cargar nuestro archivo .ui

form\_class = uic.loadUiType("combobox.ui")[0]

class MyWindowClass(QtGui.QMainWindow, form\_class):

def \_\_init\_\_(self, parent=None):

QtGui.QMainWindow.\_\_init\_\_(self, parent)

self.setupUi(self)

#Rellana los datos por primera ves del comboBox\_2

self.llenar\_comboBox2()

#Señal para cambiar, segun la selecccion, el comboBox\_2

QtCore.QObject.connect(self.comboBox, QtCore.SIGNAL("currentIndexChanged(QString)"), self.llenar\_comboBox2)

self.btn\_Ver.clicked.connect(self.btn\_Ver\_clicked)

# Evento del boton Ver

def btn\_Ver\_clicked(self):

x = self.comboBox\_2.currentText()

self.Resultado.setText(x)

# Llena el comboBox\_2

def llenar\_comboBox2(self):

python = ["Diego", "Martin", "Lorena"]

java = ["Sergio", "Maria", "Miguel"]

self.comboBox\_2.clear()

if self.comboBox.currentText() == "Python":

self.comboBox\_2.addItems(python)

elif self.comboBox.currentText() == "Java":

self.comboBox\_2.addItems(java)

if \_\_name\_\_ == "\_\_main\_\_":

app = QtGui.QApplication(sys.argv)

MyWindow = MyWindowClass(None)

MyWindow.show()

app.exec\_()

**Análisis del código:**

Este código sirve como base para otros ejemplos que ustedes necesiten realizar. Lo único que se cambia es el archivo.ui que se necesita utilizar (en mi ejemplo cargo combobox.ui) y las señales y funciones que nuestra aplicación demande.

Como siempre, arrancamos importando lo necesario y cargando nuestro archivo de diseño(.ui):

import sys

from PyQt4 import QtCore, QtGui, uic

# Cargar nuestro archivo .ui

form\_class = uic.loadUiType("combobox.ui")[0]

**Luego definimos en la función**[**\_\_init\_\_**](https://www.pythondiario.com/2013/06/initpy-en-python-que-es-y-como-funciona.html)**:**

#Rellana los datos por primera ves del comboBox\_2

self.llenar\_comboBox2()

#Señal para cambiar, segun la selecccion, el comboBox\_2

QtCore.QObject.connect(self.comboBox, QtCore.SIGNAL("currentIndexChanged(QString)"), self.llenar\_comboBox2)

self.btn\_Ver.clicked.connect(self.btn\_Ver\_clicked)

**self.comboBox2()** -- Se detalla más adelante en la función llenar\_comboBox2  
**QtCore.QObject.connect(self.comboBox, QtCore.SIGNAL("currentIndexChanged(QString)"),self.llenar\_comboBox2)** -- Esta señal se encarga de conectar los 2 comboBox y hará que cambie el segundo, dependiendo de la selección del primero.

**Las funciones:**

**Llenar el comboBox 2**

def llenar\_comboBox2(self):

python = ["Diego", "Martin", "Lorena"]

java = ["Sergio", "Maria", "Miguel"]

self.comboBox\_2.clear()

if self.comboBox.currentText() == "Python":

self.comboBox\_2.addItems(python)

elif self.comboBox.currentText() == "Java":

self.comboBox\_2.addItems(java)

Esta función será la encargada de agregar los elementos al comboBox\_2 dependiendo de lo que se selecciones en el comboBox 1.

Se crean 2 listas con los integrantes de cada curso (Python y Java).  
**self.comboBox\_2.clear()** arranca la función borrando los elementos del comboBox\_2 para que no se acumulen.

**Self.comboBox.currentText() == "Python"** verifica si lo que está seleccionado actualmente en el comboBox 1 es Python (si es así, agrega los alumnos de la lista python). La otra opción hace lo mismo para Java.

Al agregar en el \_\_init\_\_ la señal **"currentIndexChanged(QString)"** asociada a la función: **llenar\_comboBox2**, el comboBox\_2 modifica los alumnos cada vez que elegimos un curso.

**Función del botón Ver:**

def btn\_Ver\_clicked(self):

x = self.comboBox\_2.currentText()

self.Resultado.setText(x)

Esta función guarda en la variable **x** el elemento que actualmente está seleccionado en el **comboBox\_2.**

Luego modifica la etiqueta **Resultado** con el nombre del alumno seleccionado al apretar el **botón Ver**.