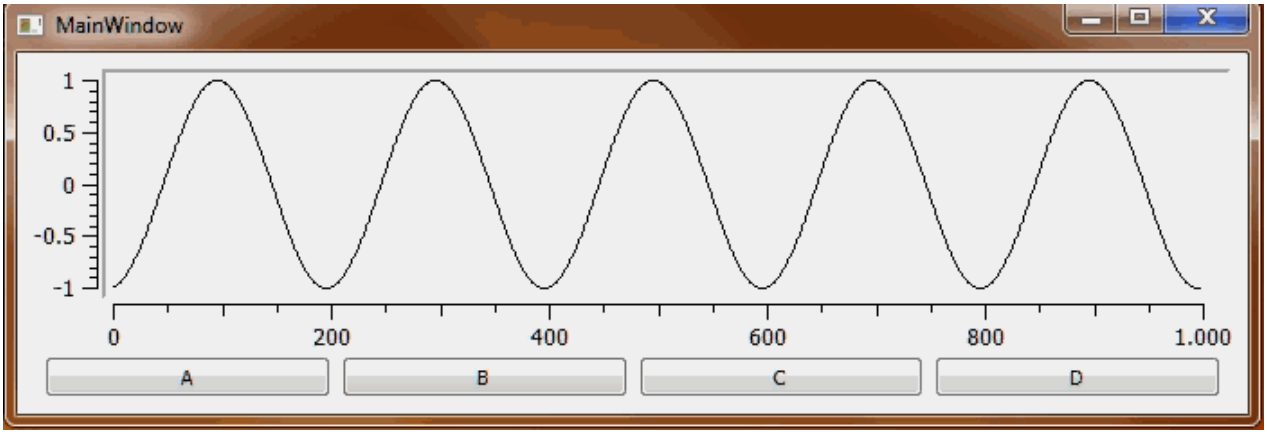
Realtime Data Plotting in Python

Categories: General, Python

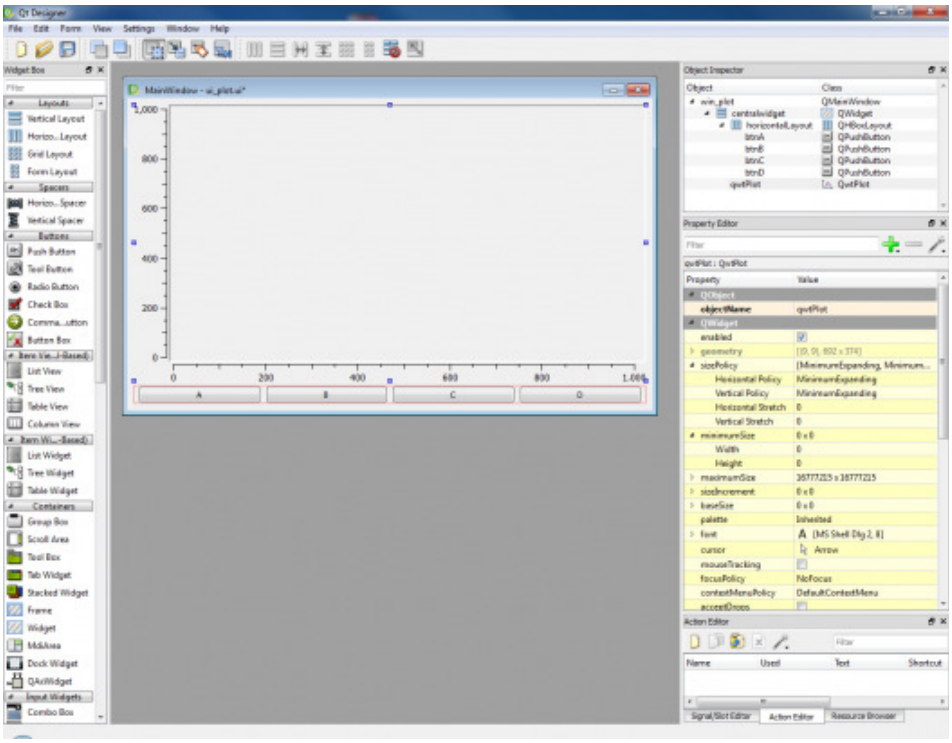
12 comments

May 8, 2013

I love using python for handing data. Displaying it isn't always as easy. Python fast to write, and numpy, scipy, and matplotlib are an incredible combination. I love matplotlib for displaying data and use it all the time, but when it comes to realtime data visualization, matplotlib (admittedly) falls behind. Imagine trying to plot sound waves in real time. Matplotlib simply can't handle it. I've recently been making progress toward this end with PyQwt with the Python X,Y distribution. It is a cross-platform solution which should perform identically on Windows, Linux, and MacOS. Here's an example of what it looks like plotting some dummy data (a sine wave) being transformed with numpy.roll().



How did I do it? Easy. First, I made the GUI with QtDesigner (which comes with Python x,y). I saved the GUI as a .ui file. I then used the pyuic4 command to generate a python script from the .ui file. In reality, I use a little helper script I wrote designed to build .py files from .ui files and start a little "ui.py" file which imports all of the ui classes. It's overkill for this, but I'll put it in the ZIP anyway. Here's what the GUI looks like in QtDesigner:



After that, I tie everything together in a little script which updates the plot in real time. It takes inputs from button click events and tells a clock (QTimer) how often to update/replot the data. Replotting it involves just rolling it with

numpy.roll(). Check it out:

```
import ui_plot #this was generated by pyuic4 command
import sys
import numpy
from PyQt4 import QtCore, QtGui
import PyQt4.Qwt5 as Qwt

numPoints=1000
xs=numpy.arange(numPoints)
ys=numpy.sin(3.14159*xs*10/numPoints) #this is our data

def plotSomething():
    global ys
    ys=numpy.roll(ys,-1)
    c.setData(xs, ys)
    uiplot.qwtPlot.replot()

if __name__ == "__main__":
    app = QtGui.QApplication(sys.argv)
    win_plot = ui_plot.QtGui.QMainWindow()
    uiplot = ui_plot.Ui_win_plot()
    uiplot.setupUi(win_plot)

    # tell buttons what to do when clicked
    uiplot.btnA.clicked.connect(plotSomething)
    uiplot.btnB.clicked.connect(lambda: uiplot.timer.setInterval(100.0))
    uiplot.btnC.clicked.connect(lambda: uiplot.timer.setInterval(10.0))
    uiplot.btnD.clicked.connect(lambda: uiplot.timer.setInterval(1.0))

    # set up the QwtPlot (pay attention!)
    c=Qwt.QwtPlotCurve()  #make a curve
    c.attach(uiplot.qwtPlot) #attach it to the qwtPlot object
    uiplot.timer = QtCore.QTimer() #start a timer (to call replot events)
    uiplot.timer.start(100.0) #set the interval (in ms)
    win_plot.connect(uiplot.timer, QtCore.SIGNAL('timeout()'), plotSomething)

    # show the main window
    win_plot.show()
    sys.exit(app.exec_())
```

I'll put all the files in a ZIP to help out anyone interested in giving this a shot. Clicking different buttons updates the graph at different speeds. If you make something cool with this concept, let me know! I'd love to see it.

DOWNLOAD PROJECT: realtime_python_graph.zip