Creating a simple GUI Application:

Steps:

1) Define a QT container object to hold your widgets:

# This gui approach does not require exact parameters for window placement

**class Scope(QtGui.QWidget):**

2) Initialize the object with, and unique widgets, layouts and event handlers:

**def \_\_init\_\_(self):**

1. inherited behavior

**super().\_\_init\_\_()**

1. a layout for placement of widgets within the container. In this case it is a grid layout:

# Layouts allow you to keep widgets in place as window is resized.

## Create a grid layout to manage the widgets size and position

**layout = QtGui.QGridLayout()**

**self.setLayout(layout)**

1. Create widgets to be placed within the container object and format their properties

**btn = QtGui.QPushButton('Plot Channel')**

**text = QtGui.QLineEdit('enter text')**

**listw = QtGui.QListWidget()**

**plot = pg.PlotWidget()**

1. Define how to position the widgets within the layout:

# addWidget (widget, row to place, column to place, <rows to span>, <columns to span>)

**layout.addWidget(btn, 0, 0) # button goes in upper-left**

**layout.addWidget(text, 1, 0) # text edit goes in middle-left**

**layout.addWidget(listw, 2, 0) # list widget goes in bottom-left**

**layout.addWidget(plot, 0, 1, 3, 1) # plot goes on right side, spanning 3 rows**

1. Define Actions and link them to functions:

**self.btn.clicked.connect(self.buttonClicked)**

1. Reveal the GUI:

**self.show()**

3) Write the event handlers:

**def buttonClicked(self):**

**time, data = self.run\_channel()**

**self.plot.clear()**

**self.plot.plot(time, data)**

4) Write the \_\_main\_\_() method:

**def \_\_main\_\_():**

**app = QtGui.QApplication(sys.argv)**

**scope = Scope()**

**sys.exit(app.exec\_())**