PyQGIS code and documentation

GEO1005 (2017-18 Q2)

Spatial Decision Support for Planning and Crisis Management

Jorge Gil

Objectives

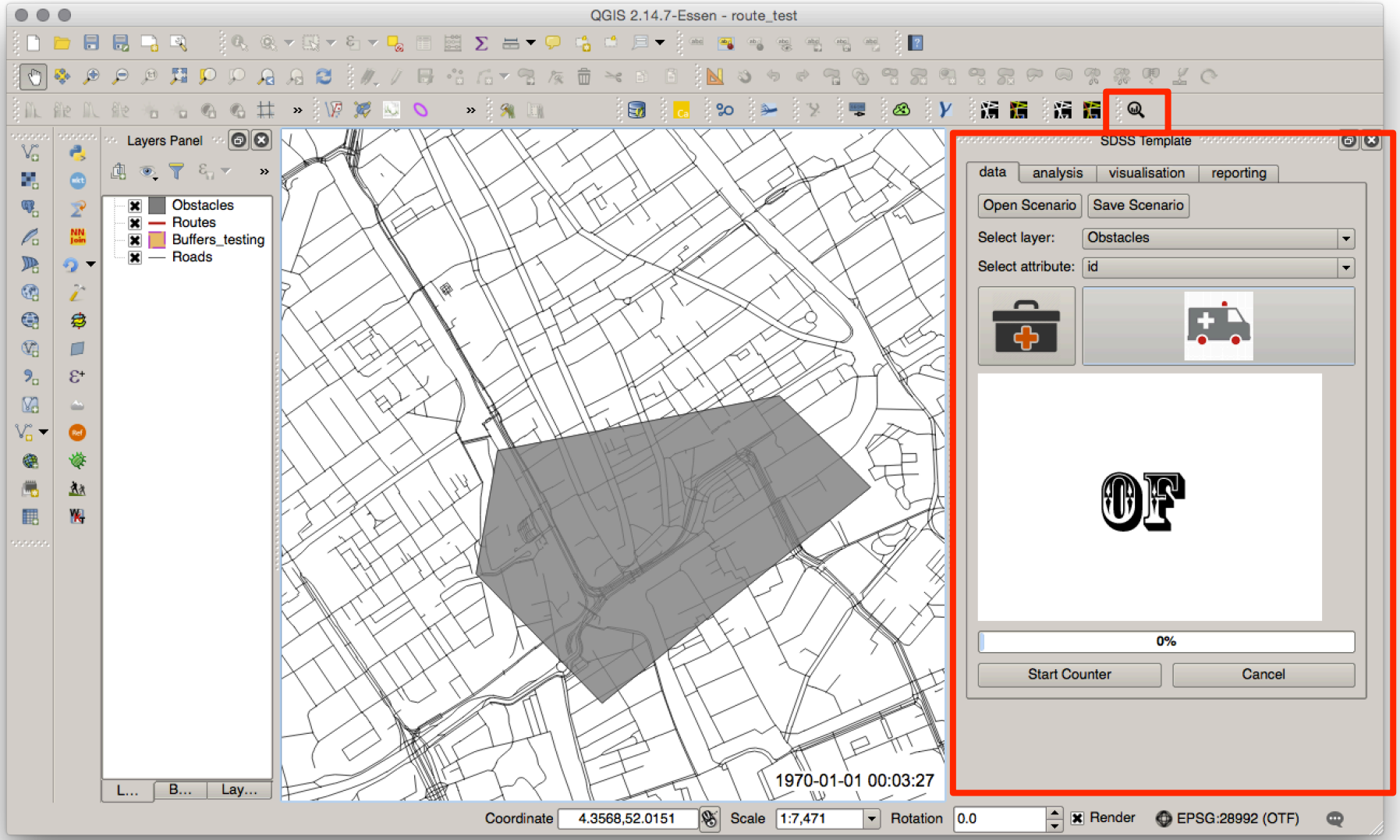
1. SDSS Sample plugin
2. Qt Designer / Creator tips
3. PyQGIS documentation
4. Time Manager plugin

SDSS Sample plugin

Two approaches for using the sample code:

1. Start from the sample plugin making changes (can be hard to manage)
2. Start new plugin from scratch copying pasting as needed (ideal)

SDSS sample plugin



- The interface
- The plugin structure
- The plugin functions

The development workflow:

1. Create basic plug-in using “Plugin Builder”
2. Open the plugin as a new project in PyCharm
3. Modify user interface (UI) in QtDesigner
4. Modify python code to add functionality
5. ***Deploy* (compiles and creates plugin in QGIS plugins folder)**
6. ***Reload* the plugin using “Plug-in Reloader”**
7. Test the plugin in QGIS
8. Check for erros in the Python Stacktrace or the Python console
9. Fix the code in Pycharm
10. *Compile* resources (only if icons or UI are modified)
11. ***Quick deploy* (updates plugin in plugins folder)**
12. → Go back to step 6 (until it works!)
13. *Commit* changes to Github repository
14. → Go back to step 3 (until the plugin is complete!)

The development workflow

spatial_decision_dockwidget.py - SpatialDecision - [~/github/GEO1005/SpatialDecision]

debug qgis plugin sdss

utility_functions.py | spatial_decision.py | metadata.txt | spatial_decision_dockwidget.py | resources.qrc

Project

- SpatialDecision ~/github/GEO1005/Spat
  - external
  - help
  - i18n
  - icons
  - scripts
  - test
  - .gitignore
  - __init__.py
  - icon.png
  - Makefile
  - metadata.txt
  - pb_tool.cfg
  - plugin_upload.py

Structure

- spatial_decision_dockwidget.py
  - SpatialDecisionDockWidget(QtGui.QDock
  - TimedEvent(QtCore.QThread)

```
23
24  from PyQt4 import QtGui, QtCore, uic
25  from qgis.core import *
26  from qgis.networkanalysis import *
27  from qgis.gui import *
28  import processing
29
30  # matplotlib for the charts
31  from matplotlib.backends.backend_qt4agg import FigureCanvasQTAgg as FigureCanvas
32  from matplotlib.figure import Figure
33
34  # Initialize Qt resources from file resources.py
35  import resources
36
37  import os
38  import os.path
39  import random
40  import csv
41  import time
42
43  from . import utility_functions as uf
44
45
46  FORM_CLASS, _ = uic.loadUiType(os.path.join(
47      os.path.dirname(__file__), 'spatial_decision_dockwidget_base.ui'))
48
49
50  class SpatialDecisionDockWidget(QtGui.QDockWidget, FORM_CLASS):
51
52      closingPlugin = QtCore.pyqtSignal()
53      #custom signals
54      updateAttribute = QtCore.pyqtSignal(str)
55
56      def __init__(self, iface, parent=None):
57          """Constructor."""
58          super(SpatialDecisionDockWidget, self).__init__(parent)
59          # Set up the user interface from Designer.
60          # After setupUI you can access any designer object by doing
61          # self.<objectname>, and you can use autoconnect slots - see
62          # http://qt-project.org/doc/qt-4.8/designer-using-a-ui-file.html
63          # #widgets-and-dialogs-with-auto-connect
64
```

SpatialDecision... › __init__()

6: TODO | 9: Version Control | Terminal | Python Console | Event Log

83:19 LF UTF-8 Git: master

- Pycharm features:
  - File structure
  - Code completion
  - Code verification (style, modules, variables)
  - Integration with GitHub
  - Refactoring
  - Interface for deployment
  - Remote debugging

- Edit **pb_tool.cfg** to change the name of your plugin folder and avoid clashes

```
[plugin]
# Name of the plugin. This is the name of the directory that will
# be created in .qgis2/python/plugins
name: SpatialDecision
```

- Add additional ui, folders and files that are required by your plugin to pb_tool.cfg

- Edit **metadata.txt** to change description, authors, etc.

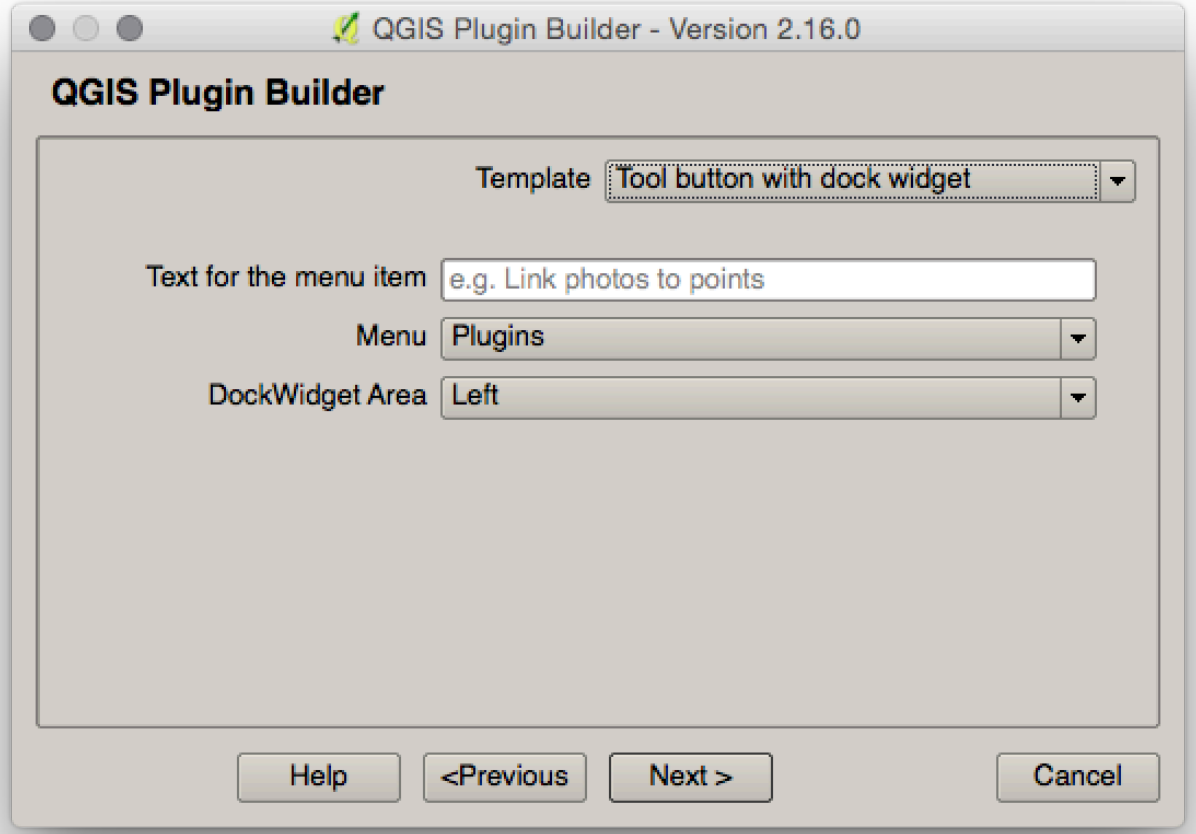
  Add tags: GEO1005, SDSS, vector

Adding icons to toolbar buttons:

- Use PNG files (square, e.g. 24x24, 32x32, 64x64)
- Copy them to a subfolder (e.g. ‘icons’)
- Edit pb_tool.cfg
  - # Other directories to be deployed with the plugin.
  - # These must be subdirectories under the plugin directory
    extra_dirs: icons
- Toolbar icon: Edit spatial_decision.py
  - *def* initGui(self):
    icon_path = ':/plugins/SpatialDecision/icons/sdss_icon.png’
- Plugin icon: Edit metadata.txt
  - category=Plugins
    icon=icons/sdss_icon.png
- **Compile or Deploy**

Qt Designer / Creator tips

Select the desired Template when creating the plugin with Plugin Builder

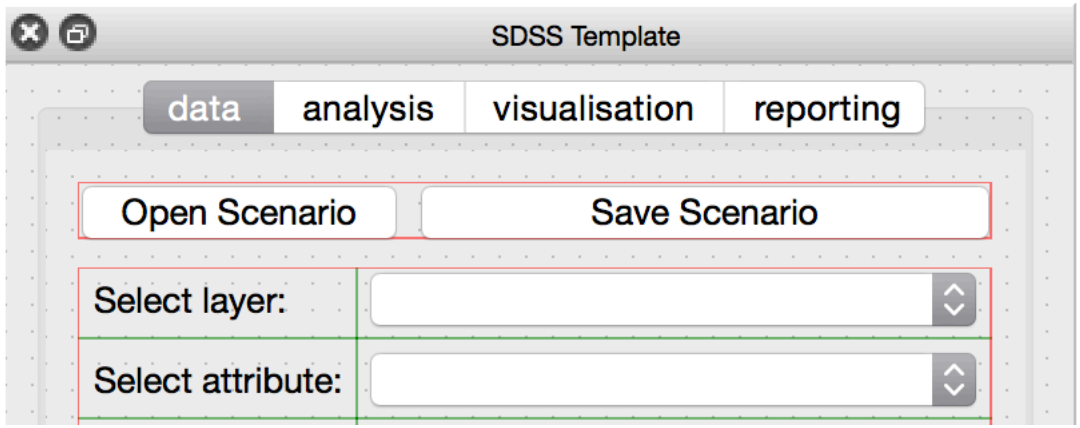


Give sensible names to widgets

[description][Type]

Object	Class
SpatialDecisionDockWidgetBase	QDockWidget
dockWidgetContents	QWidget
sampleWidgets	QTabWidget
dataTab	QWidget
formLayout	QFormLayout
openScenarioButton	QPushButton
saveScenarioButton	QPushButton
gridLayout_3	QGridLayout
ambulanceButton	QPushButton
medicButton	QPushButton
selectAttributeCombo	QComboBox
selectAttributeLabel	QLabel
selectLayerCombo	QComboBox
selectLayerLabel	QLabel
gridLayout	QGridLayout
cancelCounterButton	QPushButton
counterProgressBar	QProgressBar
startCounterButton	QPushButton
logoLabel	QLabel

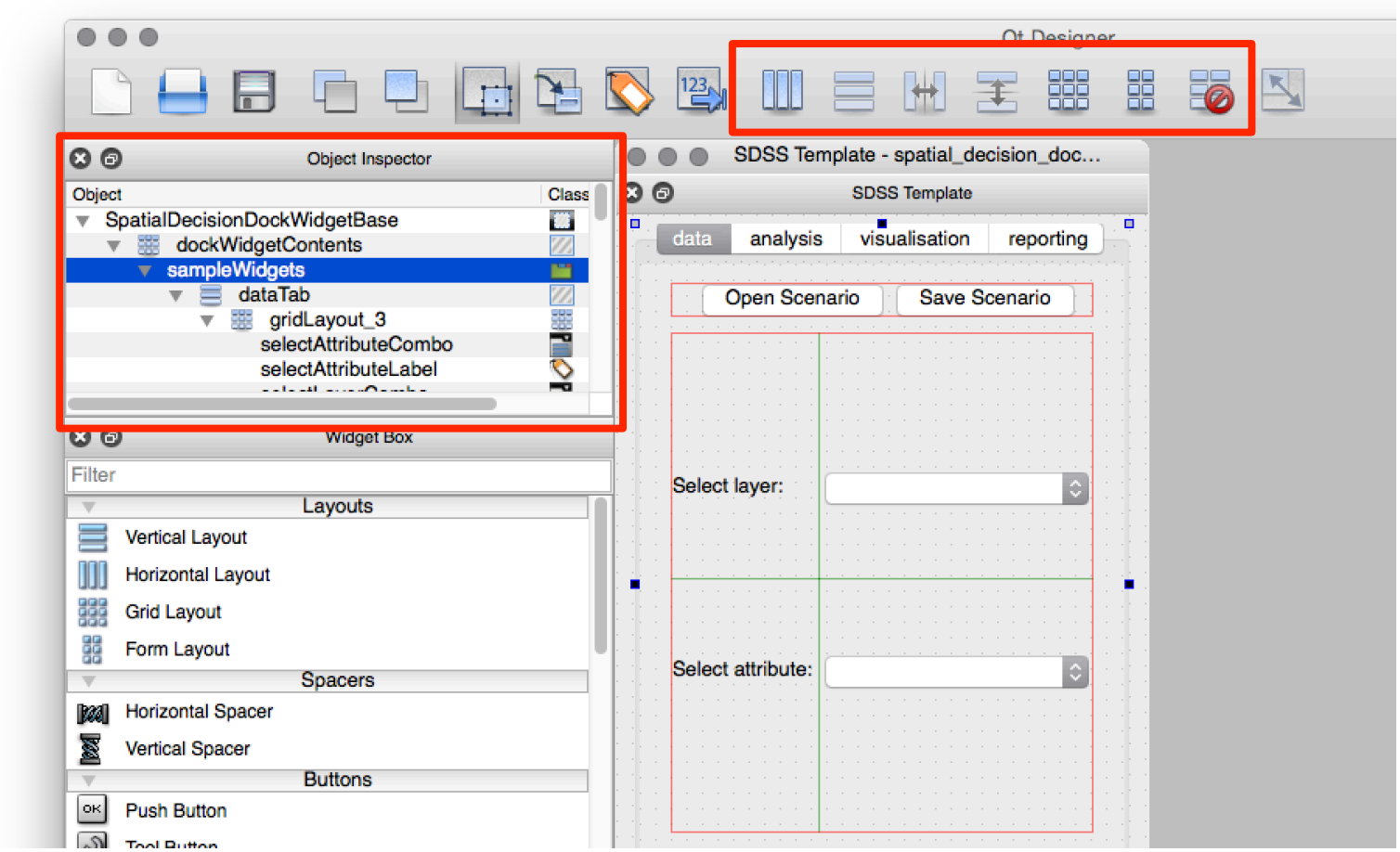
- The signal > slot principle
  - Widget emits signal



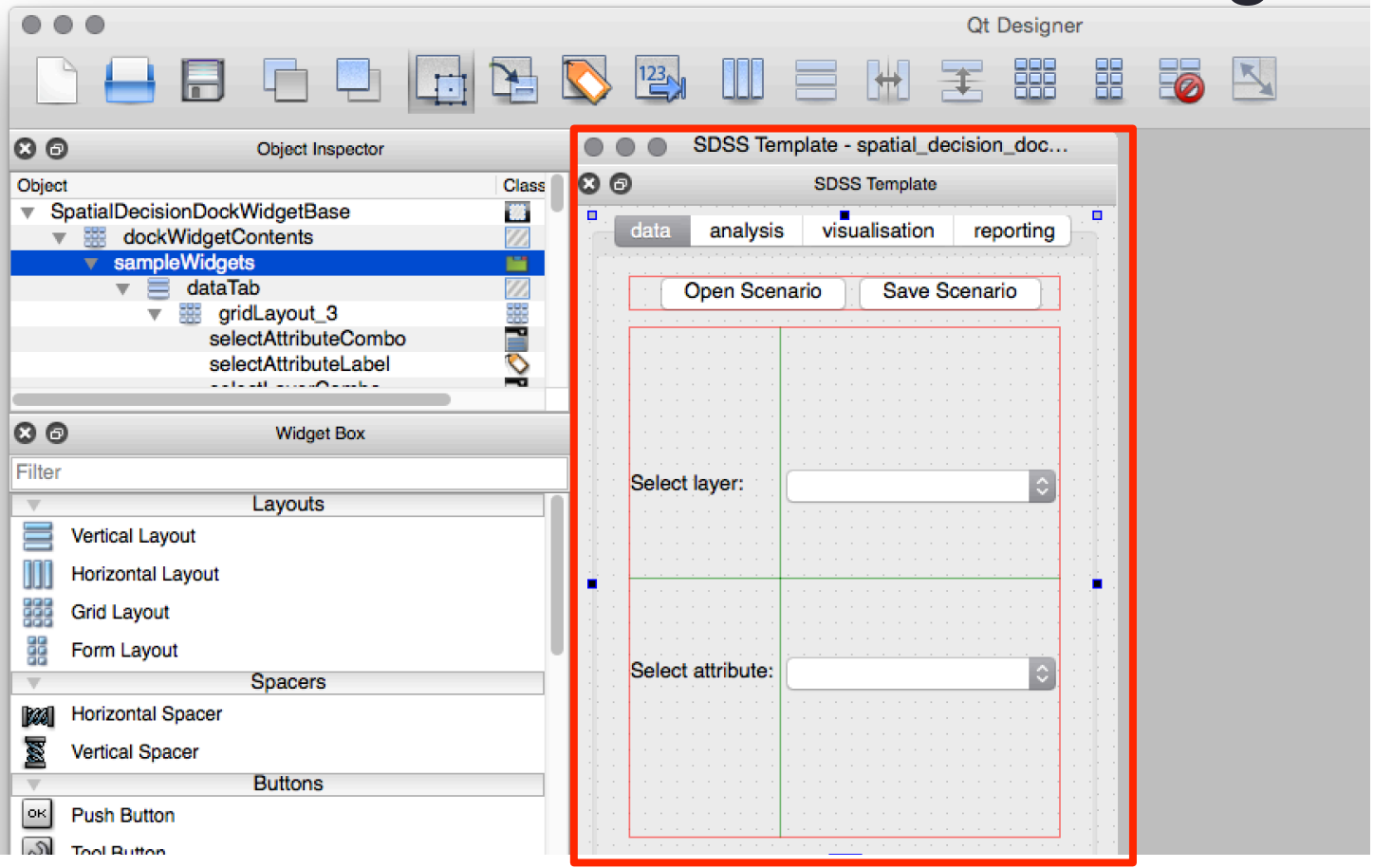
  - Function has slot to receive signal

```
self.iface.legendInterface().itemRemoved.connect(self.updateLayers)
self.iface.legendInterface().itemAdded.connect(self.updateLayers)
self.openScenarioButton.clicked.connect(self.openScenario)
self.saveScenarioButton.clicked.connect(self.saveScenario)
self.selectLayerCombo.activated.connect(self.setSelectedLayer)
self.selectAttributeCombo.activated.connect(self.setSelectedAttribute)
```

Use Layouts to keep your widgets in place when dialog changes size/shape.



- Start grouping from the “inside”: small groups of widgets
- To the “outside”: the entire dialog



Don’t use the resource editor in QtDesigner!

Set icons in the DockWidget __ini__ function

```
# add button icons
self.medicButton.setIcon(QtGui.QIcon(':icons/medic_box.png'))
self.ambulanceButton.setIcon(QtGui.QIcon(':icons/ambulance.png'))
```

Edit resources.qrc file and compile

```
1  <RCC>
2    <qresource prefix="/">
3      <file>icons/ambulance.png</file>
4      <file>icons/icon.png</file>
5      <file>icons/medic_box.png</file>
6    </qresource>
7  </RCC>
```

If using a new folder, add it to pb_tool.cfg

```
# Other directories to be deployed with the plugin.
# These must be subdirectories under the plugin directory
extra_dirs: icons
```

PyQGIS documentation

PyQGIS Developer Cookbook – many useful recipes and code snippets to try

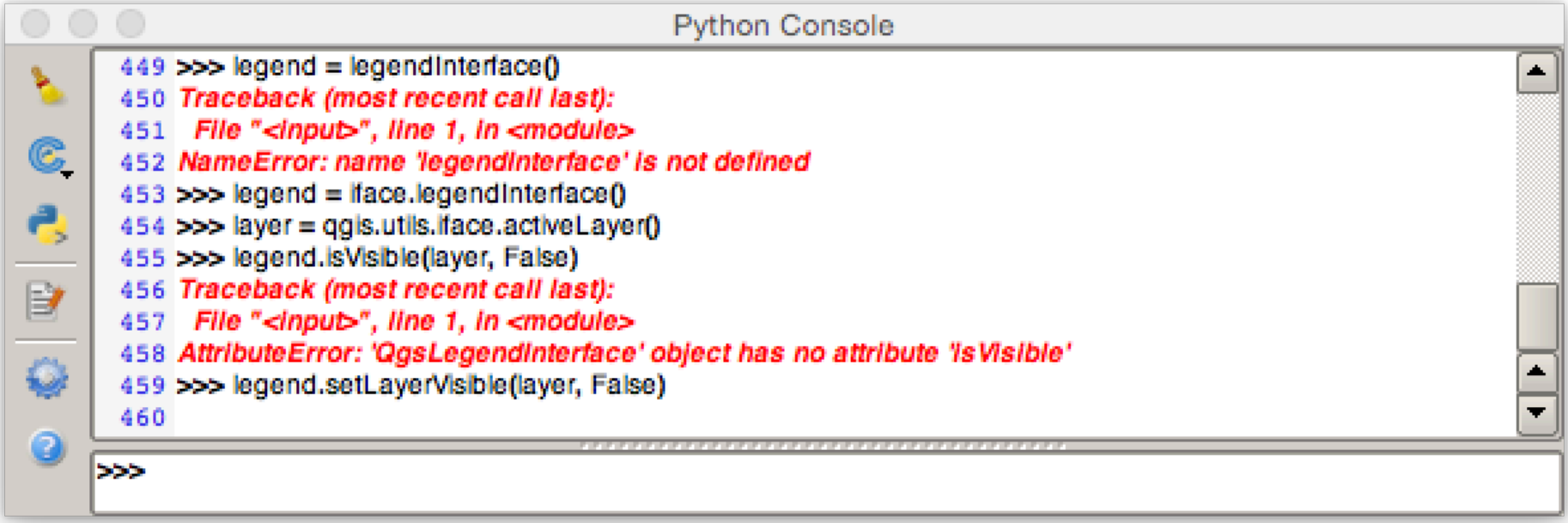
http://docs.qgis.org/2.14/en/docs/pyqgis_developer_cookbook/

QGIS Processing guide - algorithms from the console

https://docs.qgis.org/2.14/en/docs/user_manual/processing/console.html

General overview for using processing, also applicable to code in python plugins.

Use the Python console to try the API…




```
layer = self.iface.activeLayer()
legend = self.iface.legendInterface()
legend.setLayerVisible(layer, False)
```

QGIS API – the complete reference

http://qgis.org/api/2.14/

These are the most used classes:

http://qgis.org/api/2.14/classQgisInterface.html

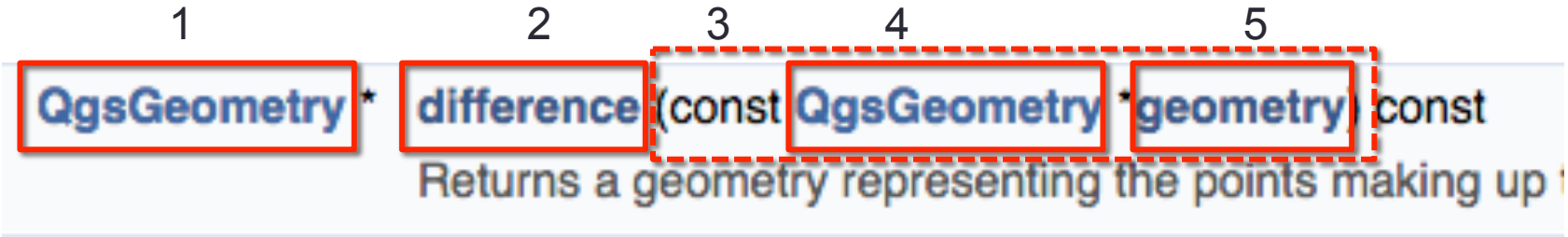
http://qgis.org/api/2.14/classQgsVectorLayer.html

http://qgis.org/api/2.14/classQgsFeature.html

http://qgis.org/api/2.14/classQgsGeometry.html

http://qgis.org/api/2.14/classQgsField.html

Example: Function of the QgsGeometry Class: applies to QgsGeometry objects → geomobj.difference(…)



1. Returned value type
2. Function name
3. Parameter(s) of the function
4. Parameter type
5. Parameter name

Based on the C++ documentation, ignore the *, &, const, virtual or other symbols you might find.

PyQt reference - Details on the user interface widgets

http://pyqt.sourceforge.net/Docs/PyQt4/qwidget.html

- Top level object, its methods are applicable to every widget
- Has links to all the different types of widgets at the top

http://pyqt.sourceforge.net/Docs/PyQt4/qabstractbutton.html

- General button, other button types inherit this, links at the top

http://pyqt.sourceforge.net/Docs/PyQt4/qtgui.html

- Complete list of GUI classes

The QGIS community – the most useful resource!

StackExchange – someone has asked it before…

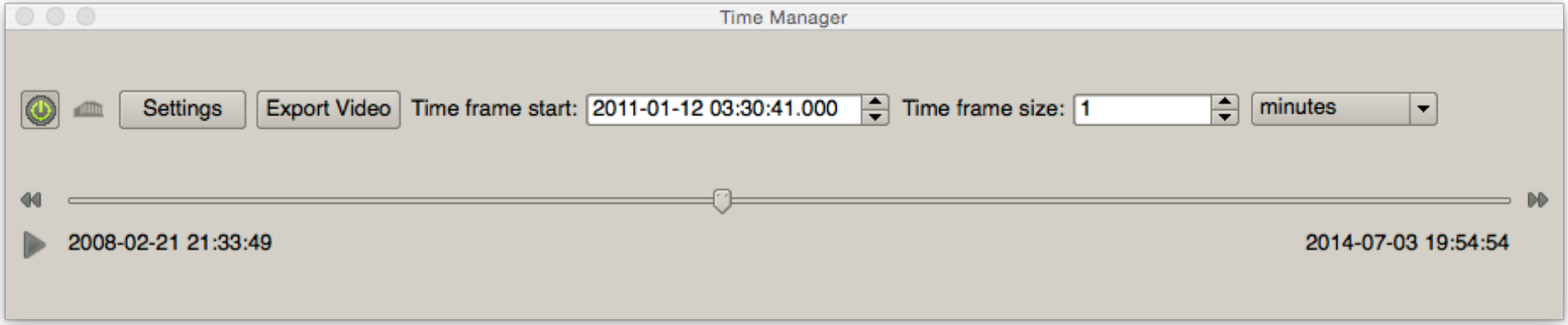
https://gis.stackexchange.com/questions/tagged/pyqgis

(as well as the qgis, qgis-plugins tags)

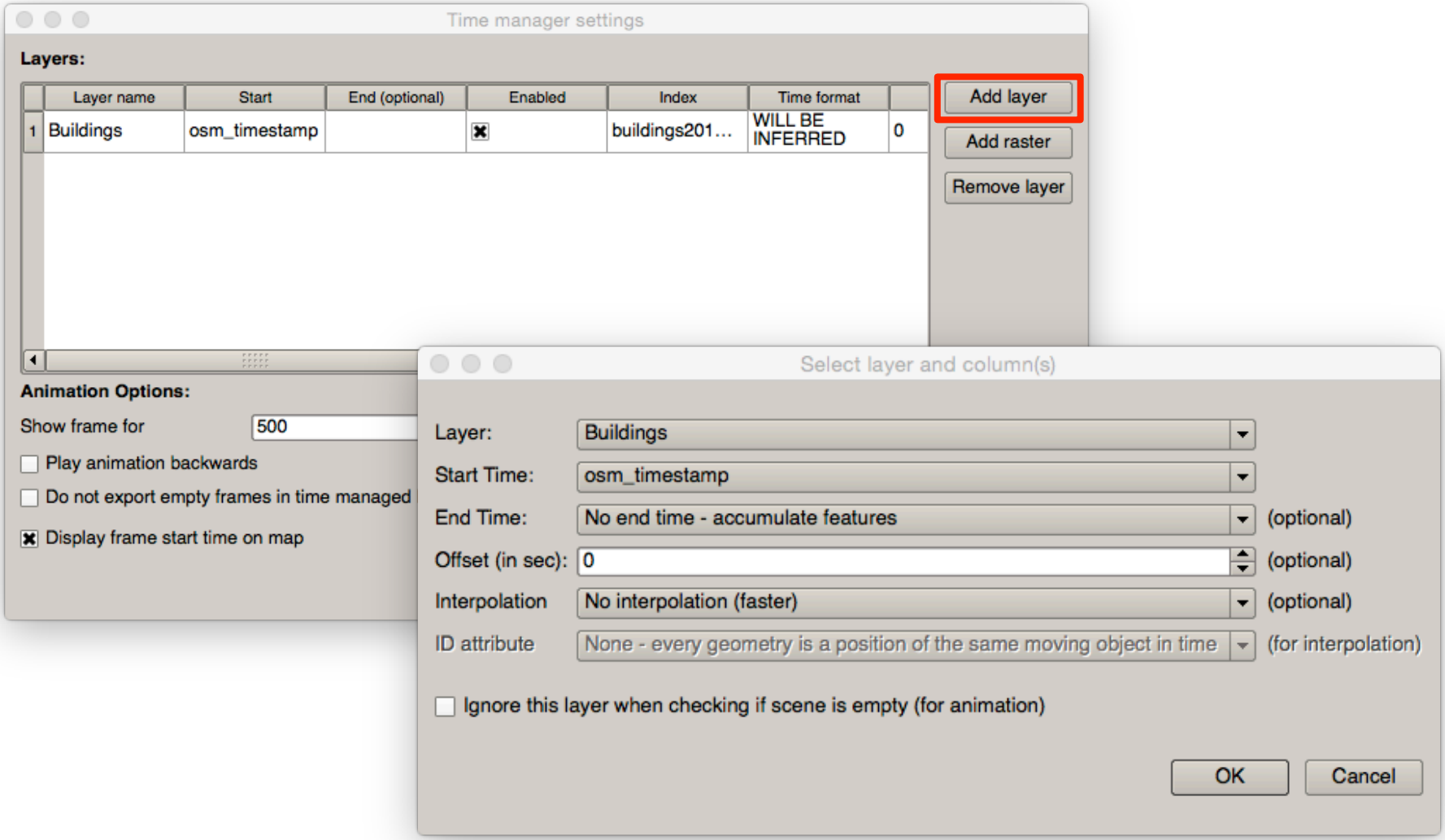
Search Google with keyword ‘pyqgis’ or ‘qgis’

Time manager plugin

- Time Manager: an event data “engine”
- Load or create a layer with an attribute with a time stamp
- Open the Time Manager panel: right click on a toolbar to see list of available panels and toolbars, check the box next to “Time Manager”



- Click “Settings” and “Add layer” select the layer and time attribute



Time manager plugin

- Use the slider or play the sequence of events
- It filters the features and the subset can be manipulated