Australian Government
Geoscience Australia

QGIS Plugin Development

FOSS4G-Oceania 2018 Workshop

Agenda: activities and lectures

Introduction and Background
Software installation
Creating a pre-filled plugin template
Creating a custom icon
Compiling resources
Create a custom icon
BREAK TIME
Configuring IDE (PyCharm)
Create your GUI with QtDesigner
Using Plugin Reloader and First Aid for testing
Uploading to the plugin repository
The PyQGIS and PyQt frameworks
Making your own plugin

- Using your own original idea
- Using a provided scenario

Topics not covered in this workshop

- Processing Providers
- Dockable Widgets
- Internationalisation (translations)
- Documentation (sphinx: HTML, LaTeX, epub, man, QtHelp)
- Testing
  - unit testing
  - assertions
  - doctest
  - property-based testing
  - code profiling
- Remote debugging
- Security (sql injection, user input sanitation, web security)

QGIS Python Plugin Background

- A way to extend or customise the functionality of QGIS
  - custom analysis workflow
  - automation of tasks
- Examples of popular plugins

QuickMapServices: catalog of webmaps and a way to add them to QGIS

MMQGIS: A collection of QGIS vector layer operations

Semi-Automatic Classification Plugin: supervised classification of remote sensing images

qgis2web: Export QGIS map to an OpenLayers/Leaflet webmap

QGIS Python Plugin Background

- Open Source
  - There is a central plugin repository: http://plugins.qgis.org/
  - QGIS Plugin Manager provides an interface to the plugin repository
  - Plugin source code can be viewed by anyone
    - issues can be raised
    - repository can be forked
    - pull requests can be made
  - 3rd party repositories (or mirrors) are possible

QGIS Python Plugin Background

- Plugins are python-based (not C, this is possible but discouraged).
- QGIS3 will break your QGIS2 plugin. http://qgis.org/api/api_break.html
  - https://github.com/qgis/QGIS/wiki/Plugin-migration-to-QGIS-3
- Essential resources:
  - https://docs.qgis.org/testing/en/docs/pyqgis_developer_cookbook/plugins.html
  - http://www.qgis.org/pyqgis-cookbook/
  - http://www.qgis.org/api/
  - https://qgis.org/pyqgis/master/

	QGIS v2	QGIS v3
Python	2	3
PyQt	4	5

Activity: Learning by Doing: software requirements

- QGIS version > 3
- QtDesigner version > 5
- PyCharm or your preferred Integrated Development Environment (IDE)

Windows:

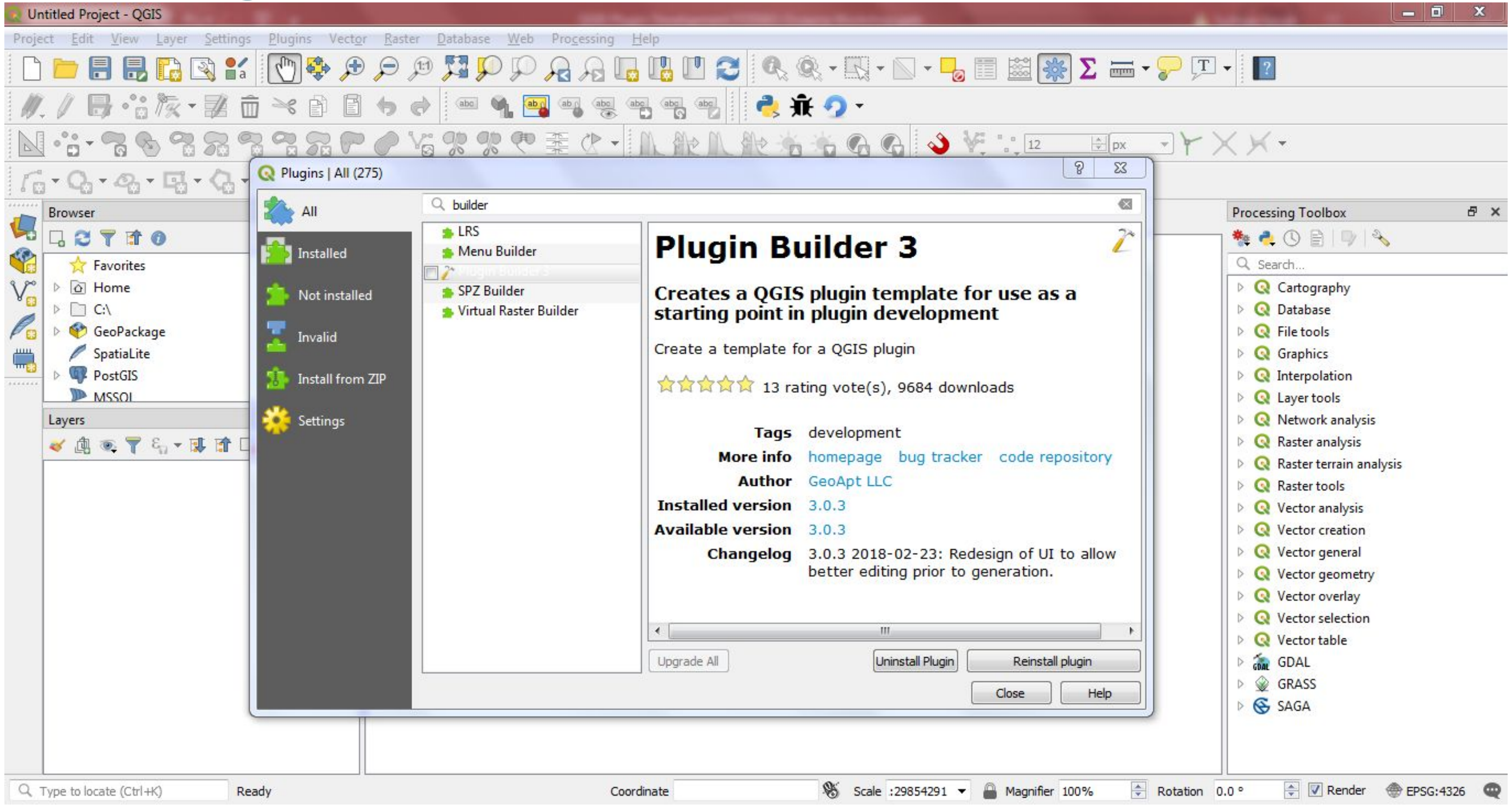
OSGeo4W installer

Linux:

QGIS repository

```
sudo apt-get install pyqt5-dev-tools
sudo apt-get install qttools5-dev-tools
```

Install Plugin Builder



Search: builder

Plugin Builder 3

Creates a QGIS plugin template for use as a starting point in plugin development

Create a template for a QGIS plugin

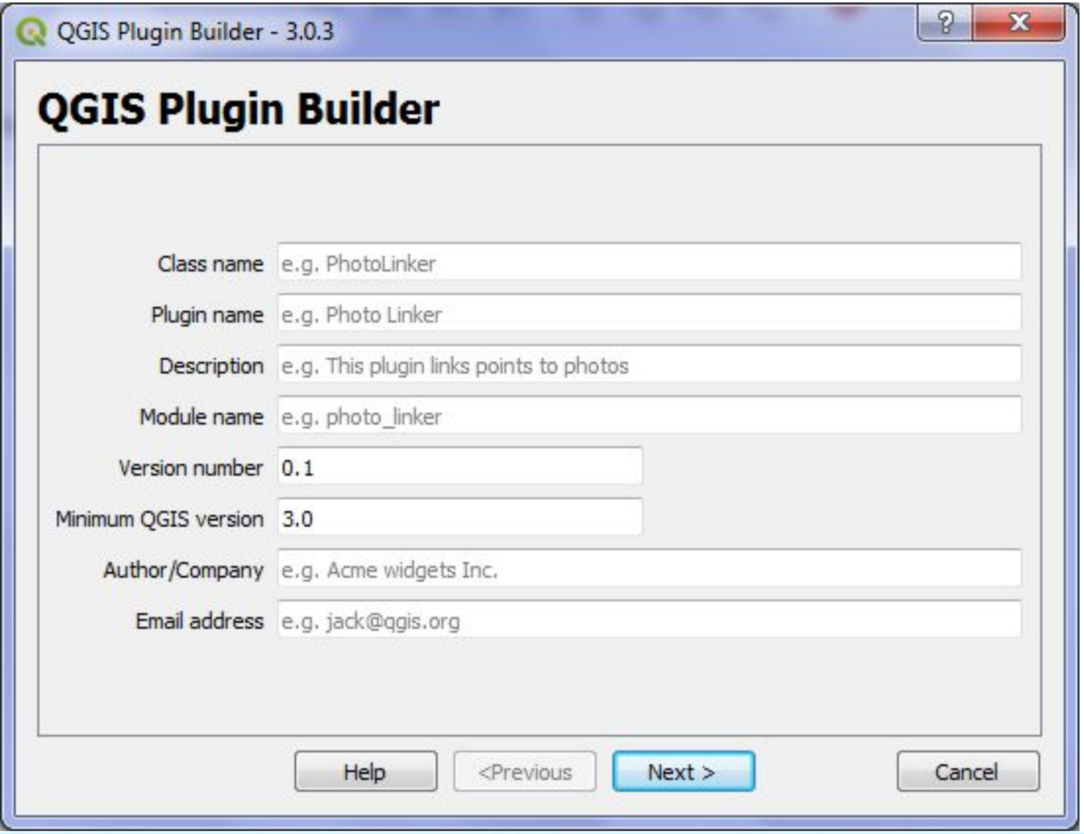
☆☆☆☆☆ 13 rating vote(s), 9684 downloads

Tags	development
More info	homepage bug tracker code repository
Author	GeoApt LLC
Installed version	3.0.3
Available version	3.0.3
Changelog	3.0.3 2018-02-23: Redesign of UI to allow better editing prior to generation.

Scale :29854291 Magnifier 100% Rotation 0.0 °

GEOSCIENCE AUSTRALIA

First Form: General Information

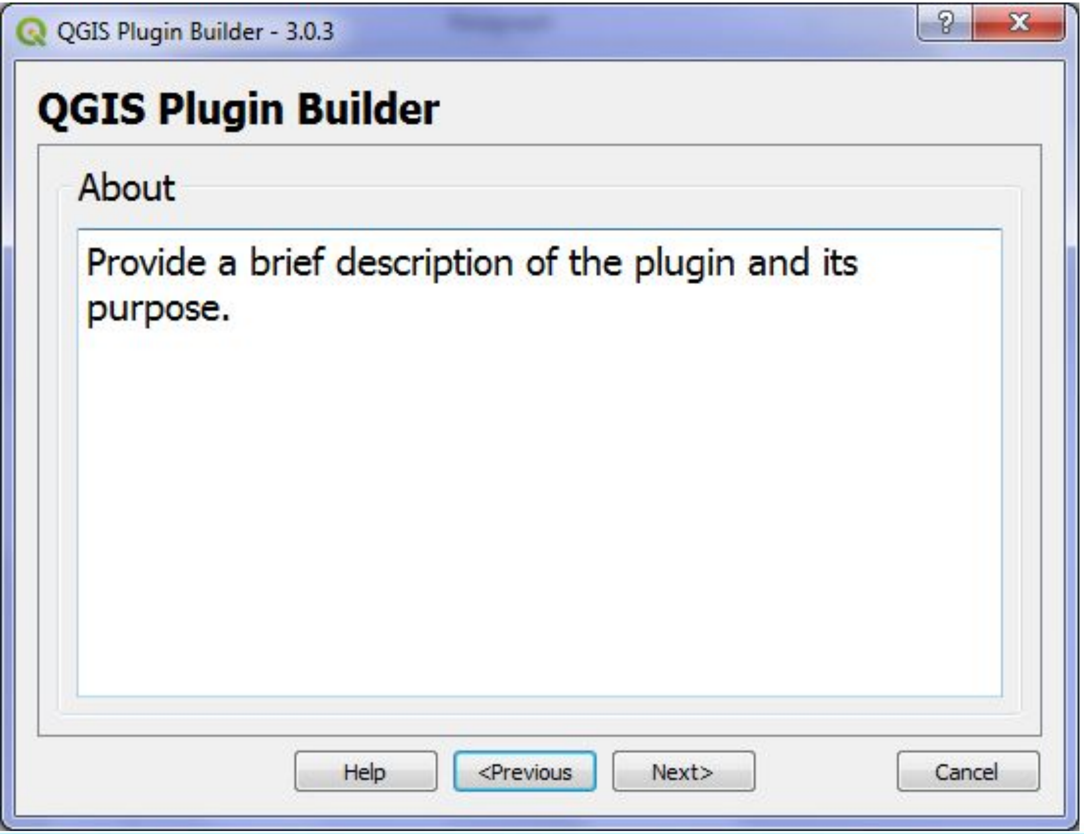

QGIS Plugin Builder - 3.0.3

QGIS Plugin Builder

Class name	e.g. PhotoLinker
Plugin name	e.g. Photo Linker
Description	e.g. This plugin links points to photos
Module name	e.g. photo_linker
Version number	0.1
Minimum QGIS version	3.0
Author/Company	e.g. Acme widgets Inc.
Email address	e.g. jack@qgis.org

CapWords	HelloWorld
Title Case	Hello World
Sentence case	Print “Hello, World!”
lowercase_with_underscore	hello_world

Second Form: Description


QGIS Plugin Builder - 3.0.3

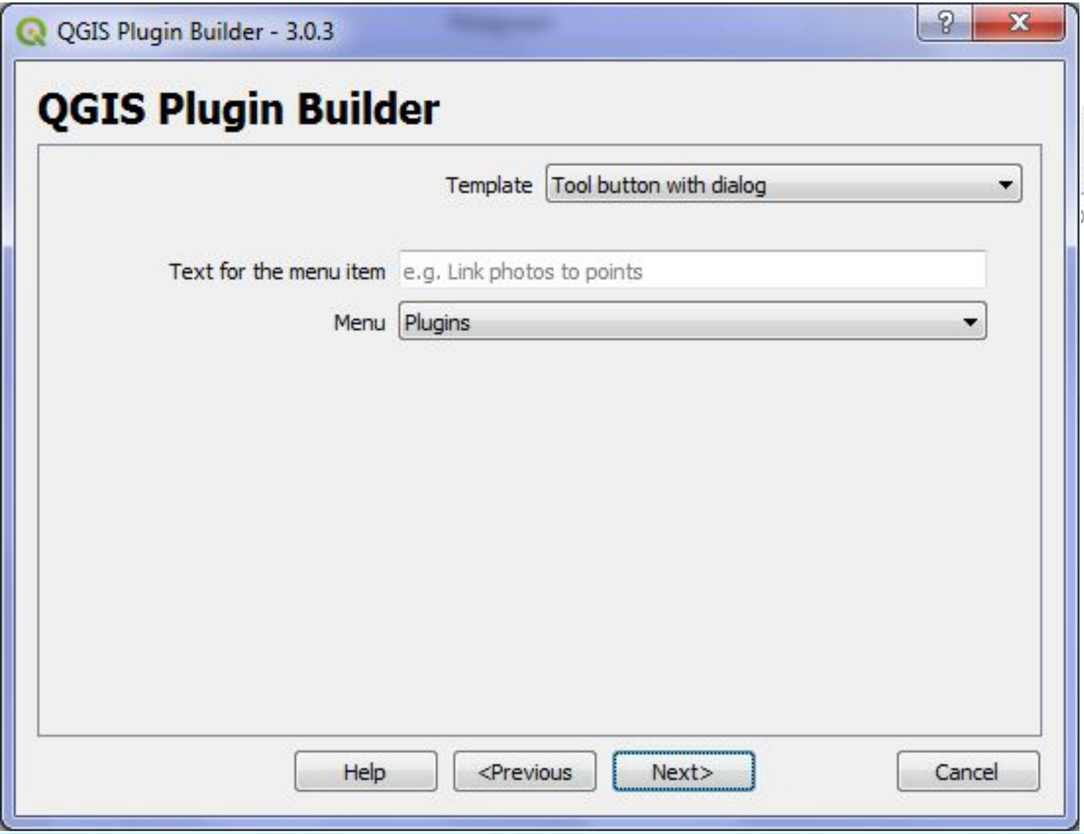
QGIS Plugin Builder

About

Provide a brief description of the plugin and its purpose.

My first plugin. Prints “Hello, World!” to the python console.

Third Form: Plugin Type and Menu

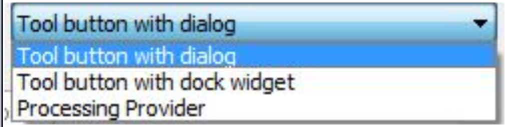

QGIS Plugin Builder - 3.0.3

QGIS Plugin Builder

Template: Tool button with dialog

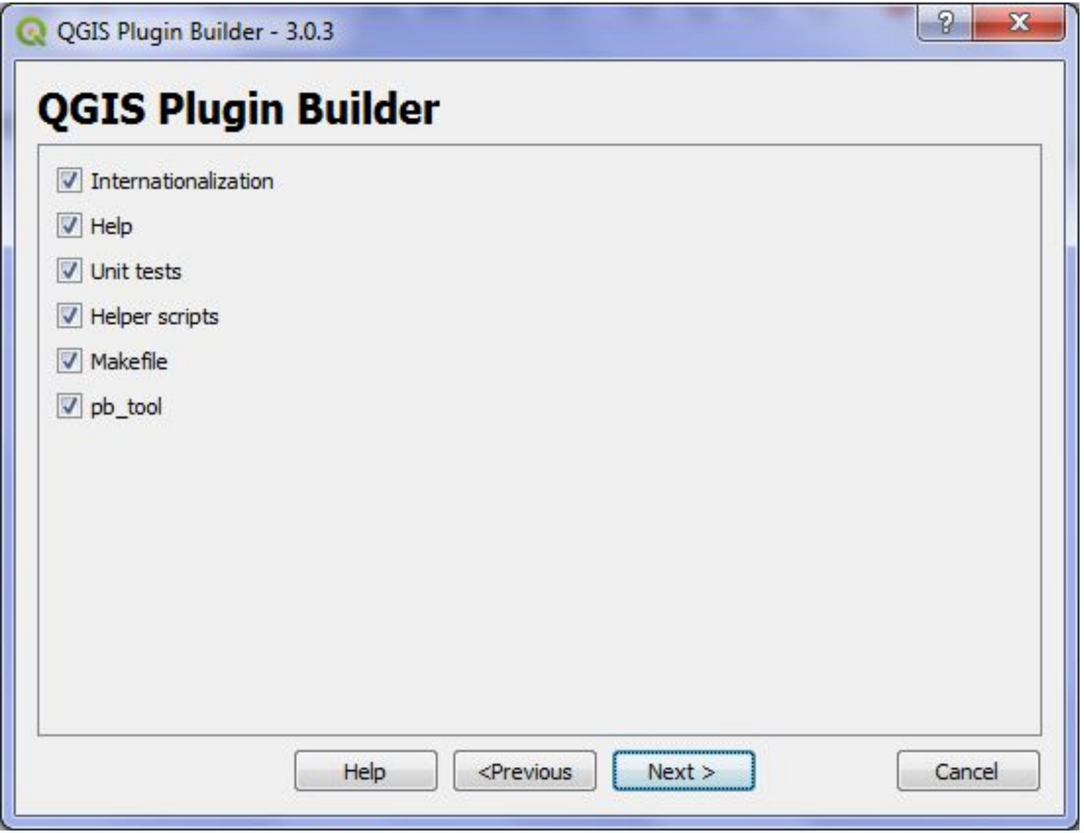
Text for the menu item: e.g. Link photos to points

Menu: Plugins



Hello World

Fourth Form: Optional Extras

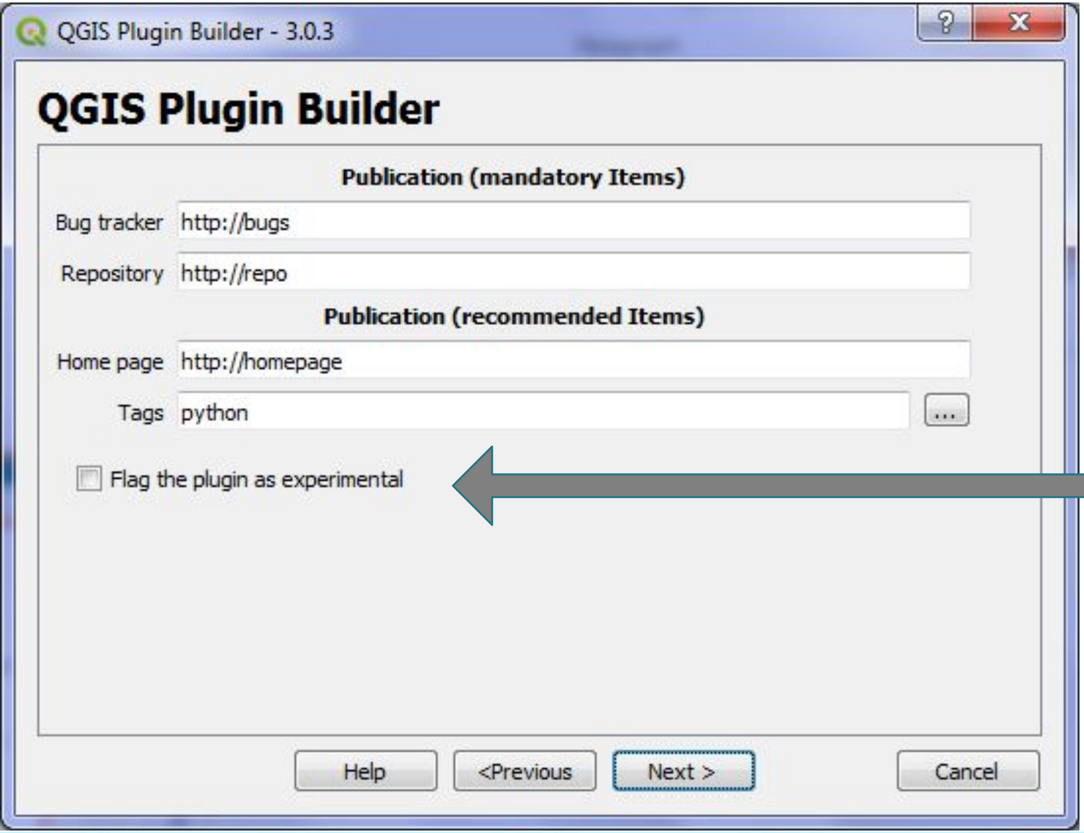

QGIS Plugin Builder - 3.0.3

QGIS Plugin Builder

- [✓] Internationalization
- [✓] Help
- [✓] Unit tests
- [✓] Helper scripts
- [✓] Makefile
- [✓] pb_tool

Translation into locales
Create sphinx template for help files
Create generic test data
Script to upload plugin
Generate GNU makefile
Tool to compile and deploy plugins

Fifth Form: Repository Information


QGIS Plugin Builder - 3.0.3

QGIS Plugin Builder

Publication (mandatory Items)

Bug tracker	http://bugs
Repository	http://repo

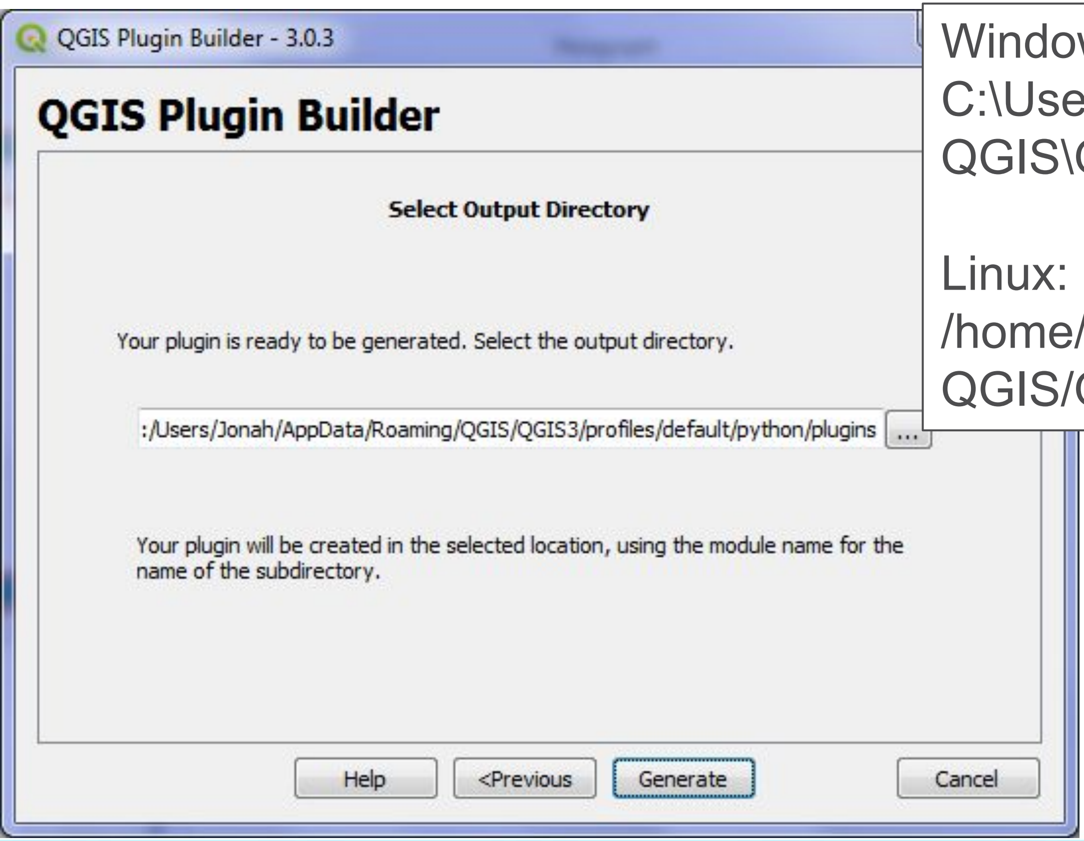
Publication (recommended Items)

Home page	http://homepage
Tags	python

☐ Flag the plugin as experimental

Ensure your plugin is flagged as experimental until you are happy with it's functionality.

Last Form: Output Directory


QGIS Plugin Builder - 3.0.3

QGIS Plugin Builder

Select Output Directory

Your plugin is ready to be generated. Select the output directory.

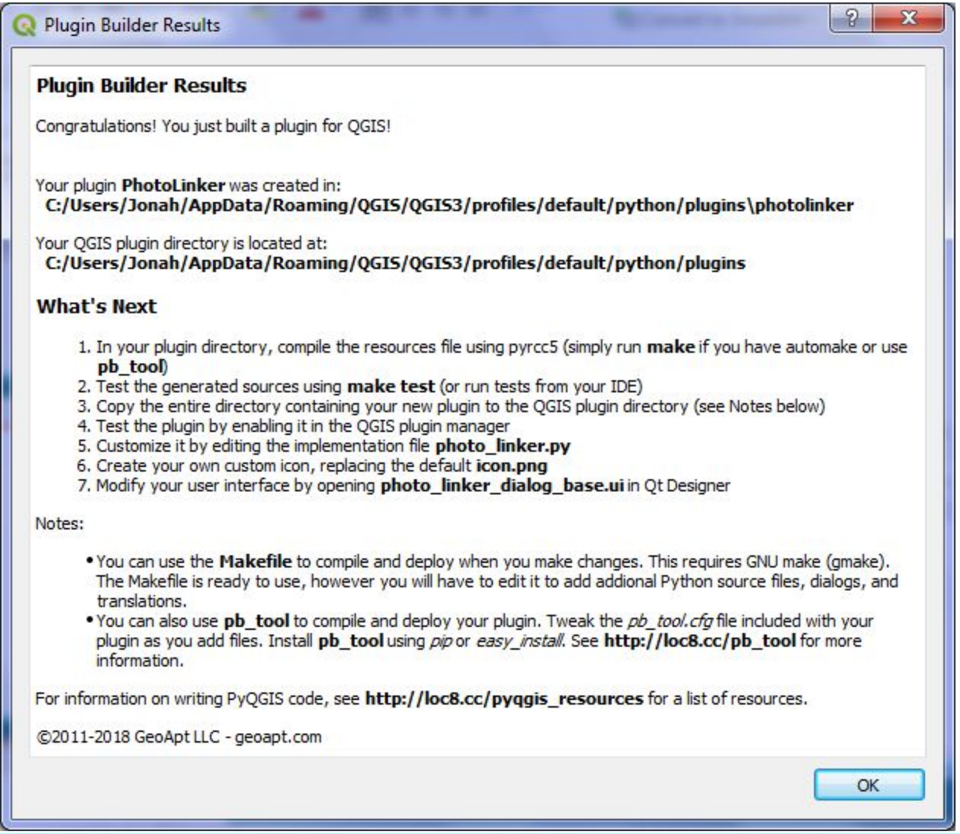
:/Users/Jonah/AppData/Roaming/QGIS/QGIS3/profiles/default/python/plugins

Your plugin will be created in the selected location, using the module name for the name of the subdirectory.

Windows:
C:\Users\user\AppData\Roaming\QGIS\QGIS3\profiles\default\python\plugins

Linux:
/home/***user***/.local/share/QGIS/QGIS3/profiles/default/python/plugins

Results



Plugin Builder Results

Congratulations! You just built a plugin for QGIS!

Your plugin **PhotoLinker** was created in:
C:/Users/Jonah/AppData/Roaming/QGIS/QGIS3/profiles/default/python/plugins\photolinker

Your QGIS plugin directory is located at:
C:/Users/Jonah/AppData/Roaming/QGIS/QGIS3/profiles/default/python/plugins

What's Next

1. In your plugin directory, compile the resources file using pyrcc5 (simply run **make** if you have automake or use **pb_tool**)
2. Test the generated sources using **make test** (or run tests from your IDE)
3. Copy the entire directory containing your new plugin to the QGIS plugin directory (see Notes below)
4. Test the plugin by enabling it in the QGIS plugin manager
5. Customize it by editing the implementation file **photo_linker.py**
6. Create your own custom icon, replacing the default **icon.png**
7. Modify your user interface by opening **photo_linker_dialog_base.ui** in Qt Designer

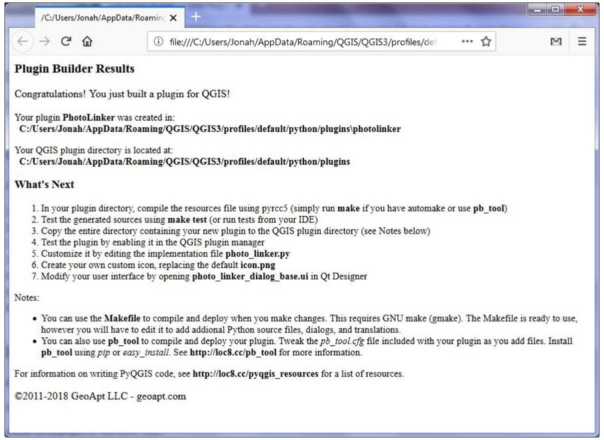
Notes:

- You can use the **Makefile** to compile and deploy when you make changes. This requires GNU make (gmake). The Makefile is ready to use, however you will have to edit it to add addional Python source files, dialogs, and translations.
- You can also use **pb_tool** to compile and deploy your plugin. Tweak the *pb_tool.cfg* file included with your plugin as you add files. Install **pb_tool** using *pip* or *easy_install*. See **http://loc8.cc/pb_tool** for more information.

For information on writing PyQGIS code, see **http://loc8.cc/pyqgis_resources** for a list of resources.

©2011-2018 GeoApt LLC - geoapt.com

Also saved as a html file in the plugin directory


/C:/Users/Jonah/AppData/Roamin
file:///C:/Users/Jonah/AppData/Roaming/QGIS/QGIS3/profiles/def

Plugin Builder Results

Congratulations! You just built a plugin for QGIS!

Your plugin **PhotoLinker** was created in:
C:/Users/Jonah/AppData/Roaming/QGIS/QGIS3/profiles/default/python/plugins\photolinker

Your QGIS plugin directory is located at:
C:/Users/Jonah/AppData/Roaming/QGIS/QGIS3/profiles/default/python/plugins

What's Next

1. In your plugin directory, compile the resources file using pyrcc5 (simply run **make** if you have automake or use **pb_tool**)
2. Test the generated sources using **make test** (or run tests from your IDE)
3. Copy the entire directory containing your new plugin to the QGIS plugin directory (see Notes below)
4. Test the plugin by enabling it in the QGIS plugin manager
5. Customize it by editing the implementation file **photo_linker.py**
6. Create your own custom icon, replacing the default **icon.png**
7. Modify your user interface by opening **photo_linker_dialog_base.ui** in Qt Designer

Notes:

- You can use the **Makefile** to compile and deploy when you make changes. This requires GNU make (gmake). The Makefile is ready to use, however you will have to edit it to add addional Python source files, dialogs, and translations.
- You can also use **pb_tool** to compile and deploy your plugin. Tweak the *pb_tool.cfg* file included with your plugin as you add files. Install **pb_tool** using *pip* or *easy_install*. See **http://loc8.cc/pb_tool** for more information.

For information on writing PyQGIS code, see **http://loc8.cc/pyqgis_resources** for a list of resources.

©2011-2018 GeoApt LLC - geoapt.com

Compiling QT Resources (optional)

http://pyqt.sourceforge.net/Docs/PyQt5/resources.html

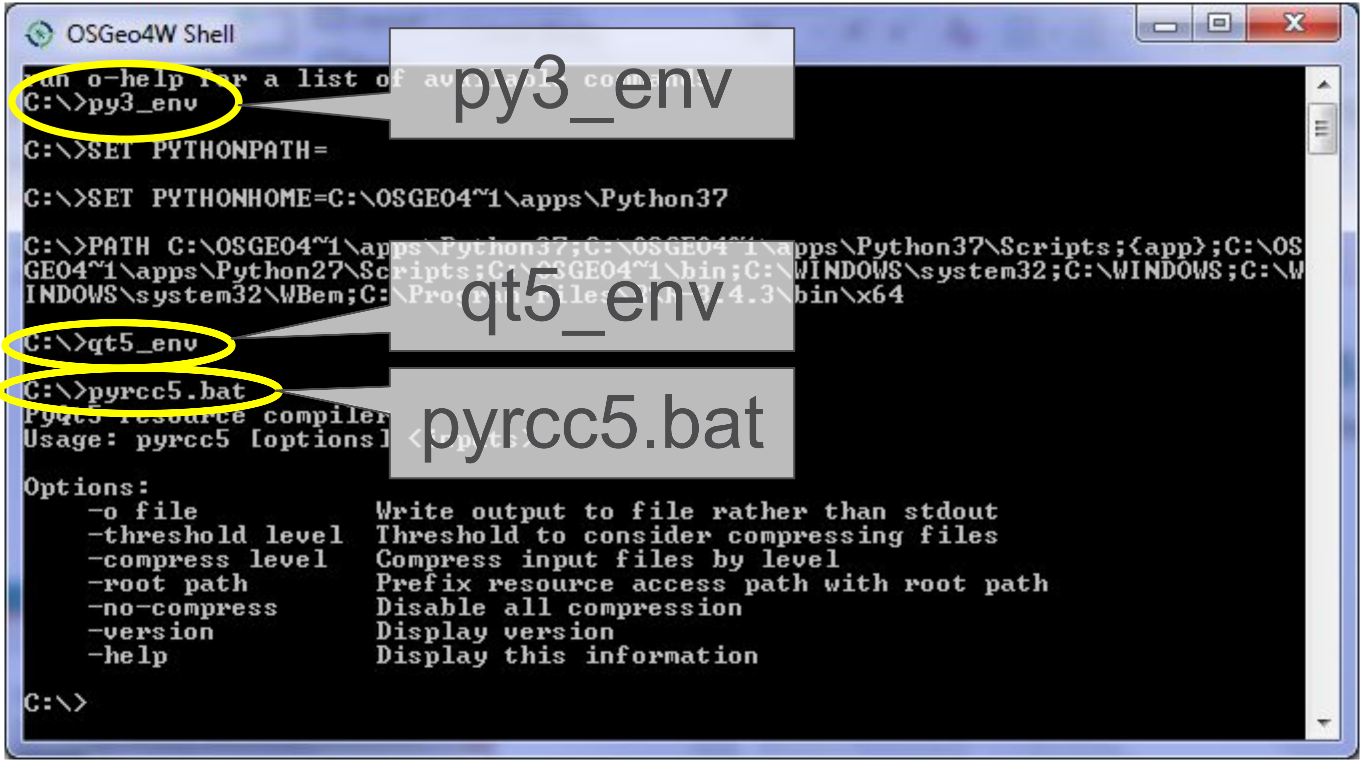
PyQt5 supports Qt’s resource system. This is a facility for embedding resources such as icons and translation files in an application. This makes the packaging and distribution of those resources much easier.

A .qrc resource collection file is an XML file used to specify which resource files are to be embedded.

pyrcc5 reads the .qrc file, and the resource files, and generates a Python module that only needs to be imported by the application in order for those resources to be made available just as if they were the original files.

Create PyQt5 resources file to store icons (optional)

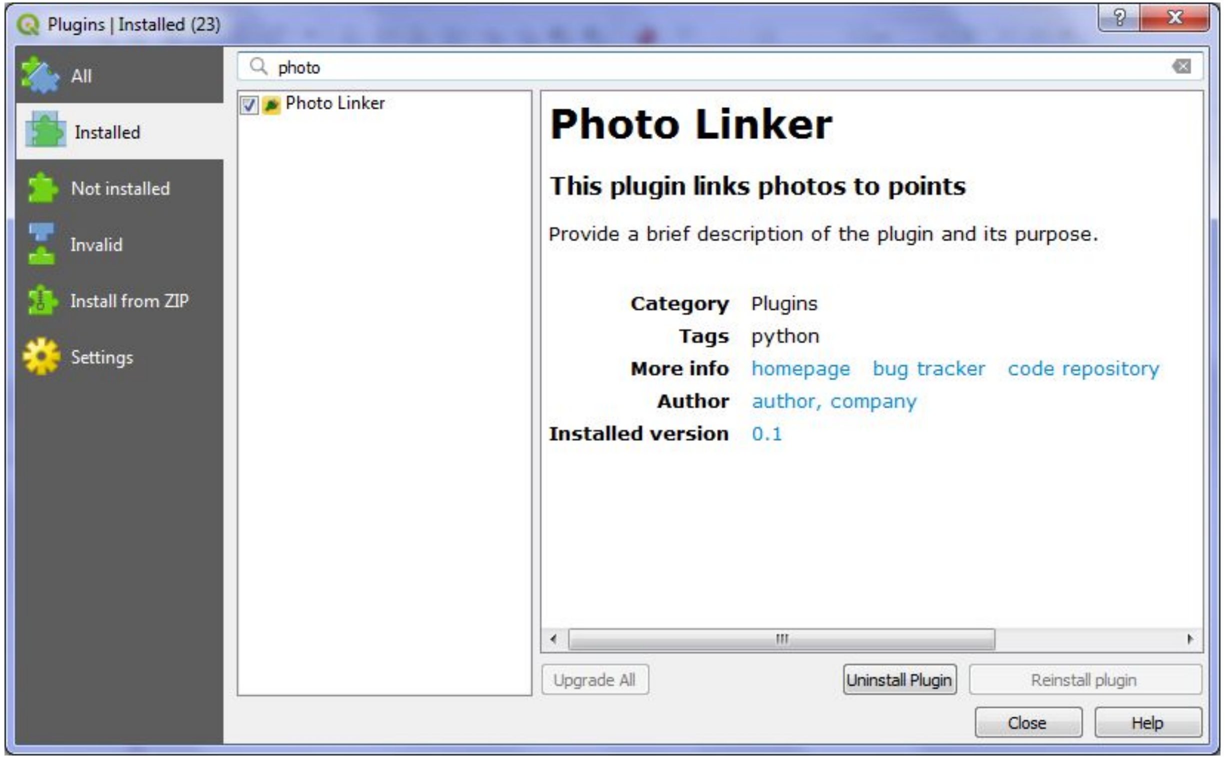
Windows: use OSGeo4W Shell to access pyrcc5 utility.



```
run o-help for a list of available commands
C:\>py3_env

C:\>SET PYTHONPATH=

C:\>SET PYTHONHOME=C:\OSGEO4~1\apps\Python37

C:\>PATH C:\OSGEO4~1\apps\Python37;C:\OSGEO4~1\apps\Python37\Scripts;{app};C:\OS
GEO4~1\apps\Python27\Scripts;C:\OSGEO4~1\bin;C:\WINDOWS\system32;C:\WINDOWS;C:\W
INDOWS\system32\WBem;C:\Program Files\...4.3\bin\x64

C:\>qt5_env

C:\>pyrcc5.bat
PyQt5 resource compiler
Usage: pyrcc5 [options] <inputs>

Options:
    -o file             Write output to file rather than stdout
    -threshold level    Threshold to consider compressing files
    -compress level     Compress input files by level
    -root path          Prefix resource access path with root path
    -no-compress        Disable all compression
    -version            Display version
    -help               Display this information

C:\>
```

Turn on your plugin

Open QGIS Plugin Manager and tick the box for your new plugin.



☑ Photo Linker

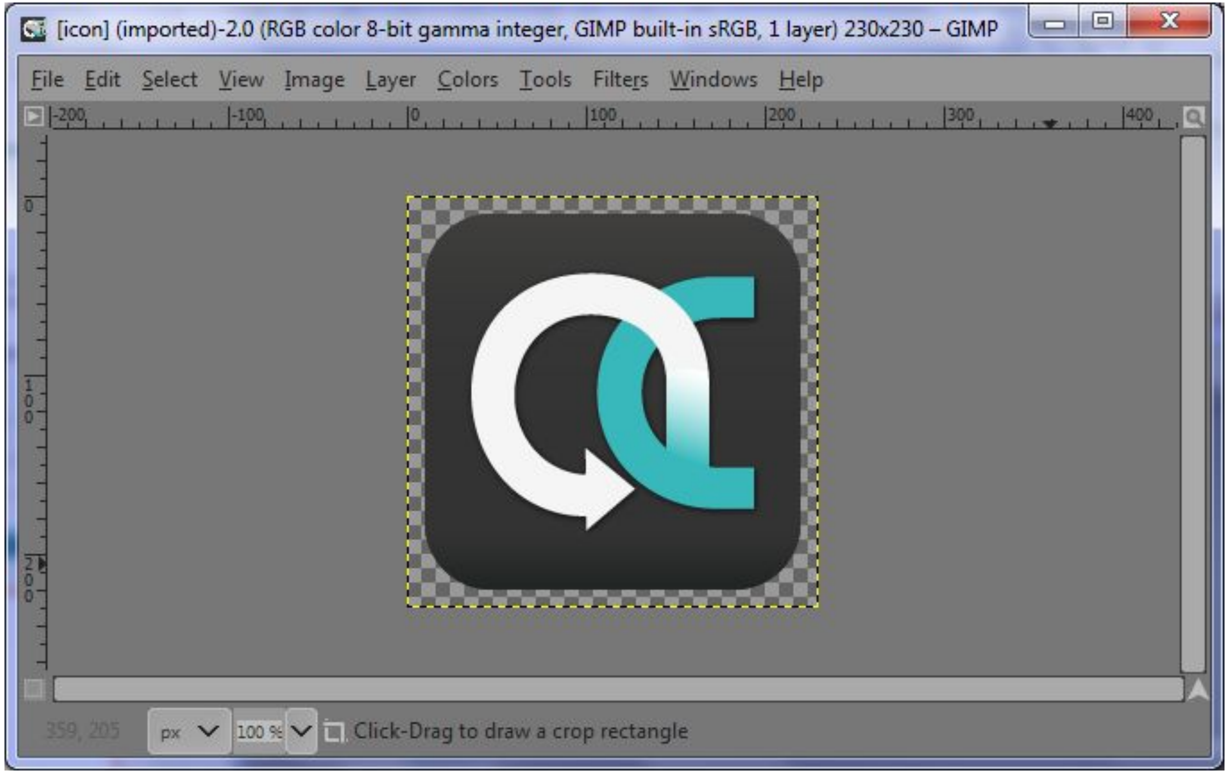
Photo Linker

This plugin links photos to points

Provide a brief description of the plugin and its purpose.

Category	Plugins
Tags	python
More info	homepage bug tracker code repository
Author	author, company
Installed version	0.1

Create a custom icon


[icon] (imported)-2.0 (RGB color 8-bit gamma integer, GIMP built-in sRGB, 1 layer) 230x230 – GIMP
File Edit Select View Image Layer Colors Tools Filters Windows Help
px
100 %
Click-Drag to draw a crop rectangle

QGIS2 used 24x24 raster, QGIS3 can use any sized square PNG

BREAK TIME [20 minutes]

During the break:

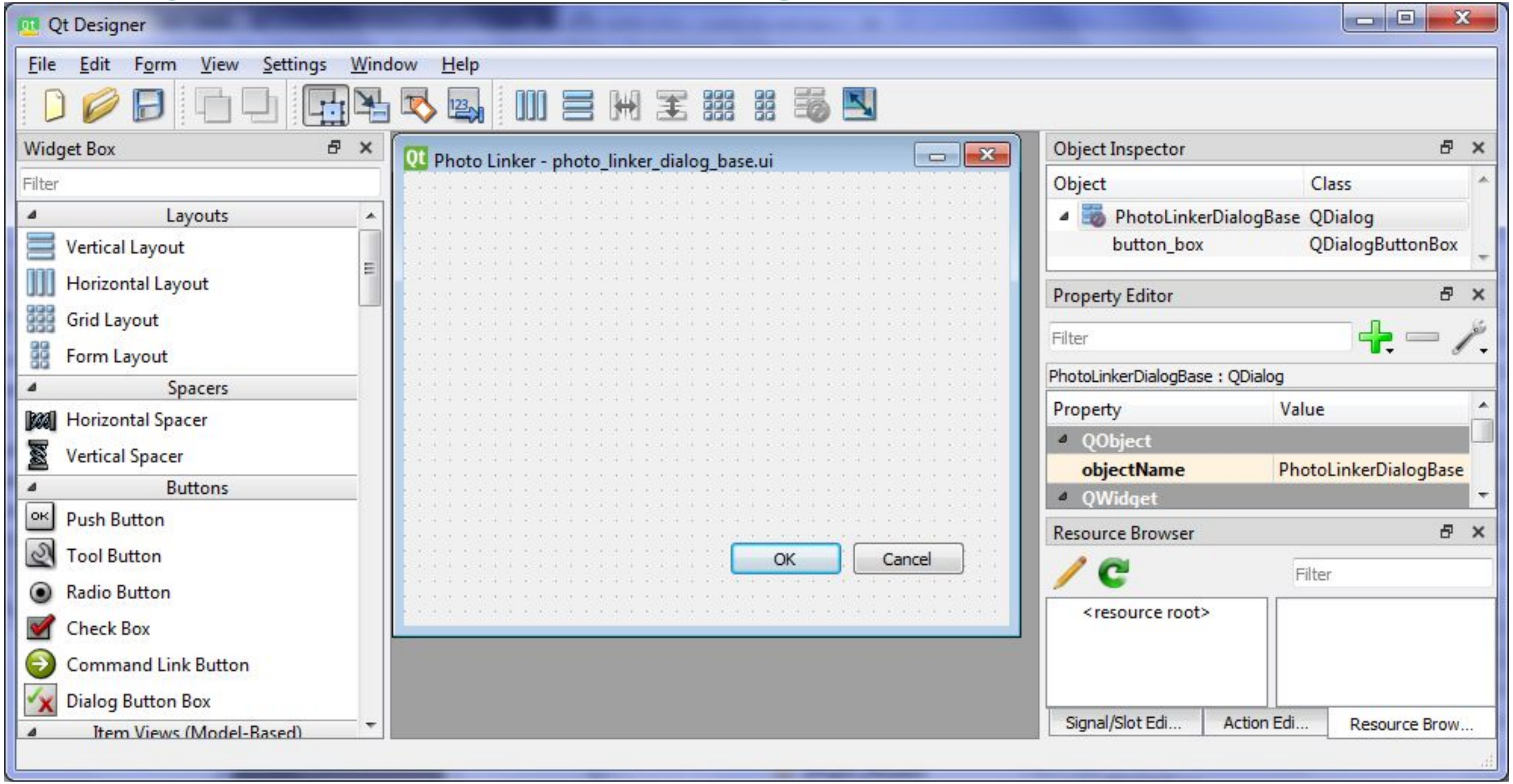
Create a custom icon (optional)

After the break:

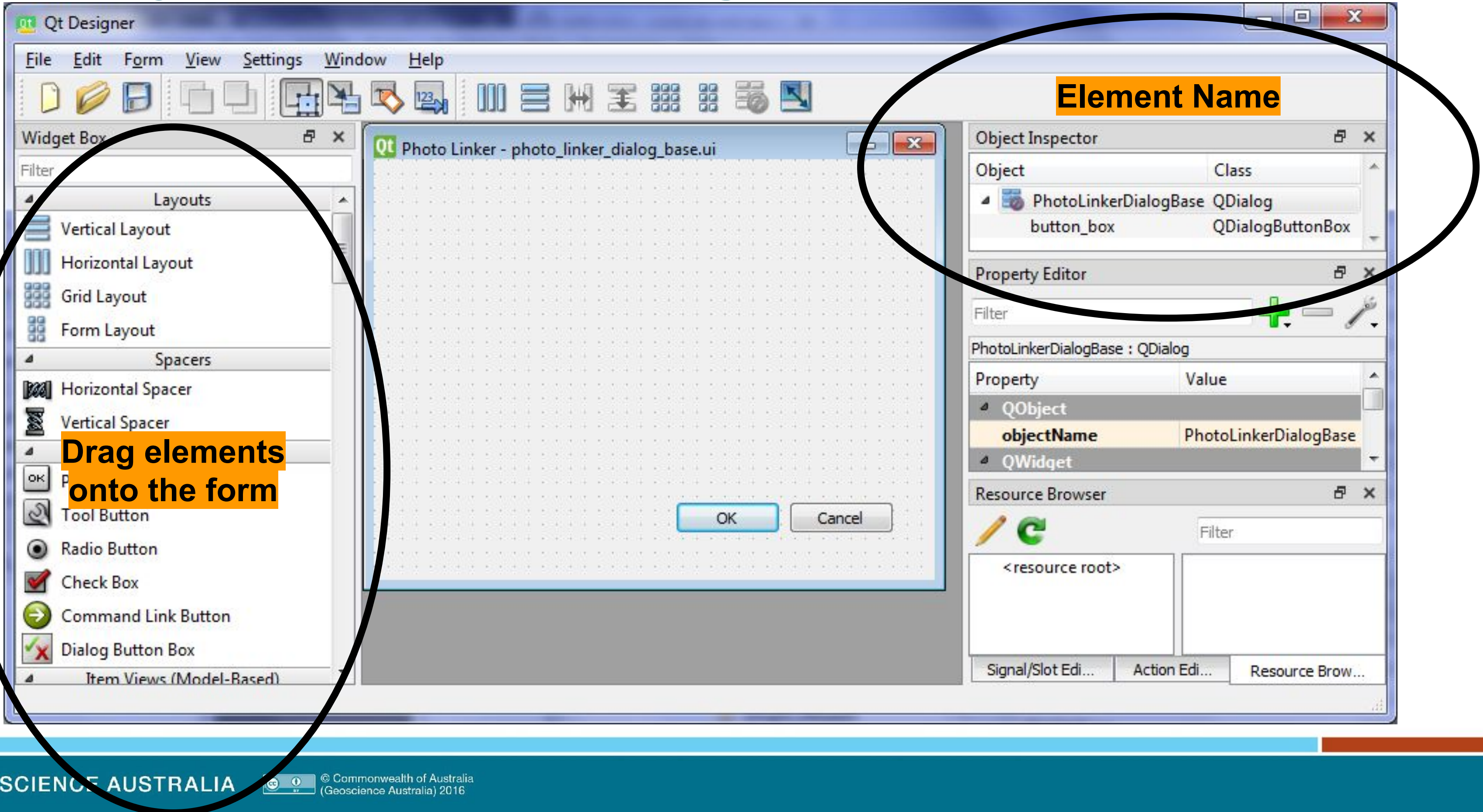
Create your GUI with QtDesigner
Configuring IDE (PyCharm)
Using Plugin Reloader and First Aid for testing
Uploading to the plugin repository
The PyQGIS and PyQt frameworks
Making your own plugin

- Using your own original idea
- Using a provided scenario

Create your GUI with QtDesigner



Create your GUI with QtDesigner



Object	Class
PhotoLinkerDialogBase	QDialog
button_box	QDialogButtonBox

Property Editor

PhotoLinkerDialogBase : QDialog

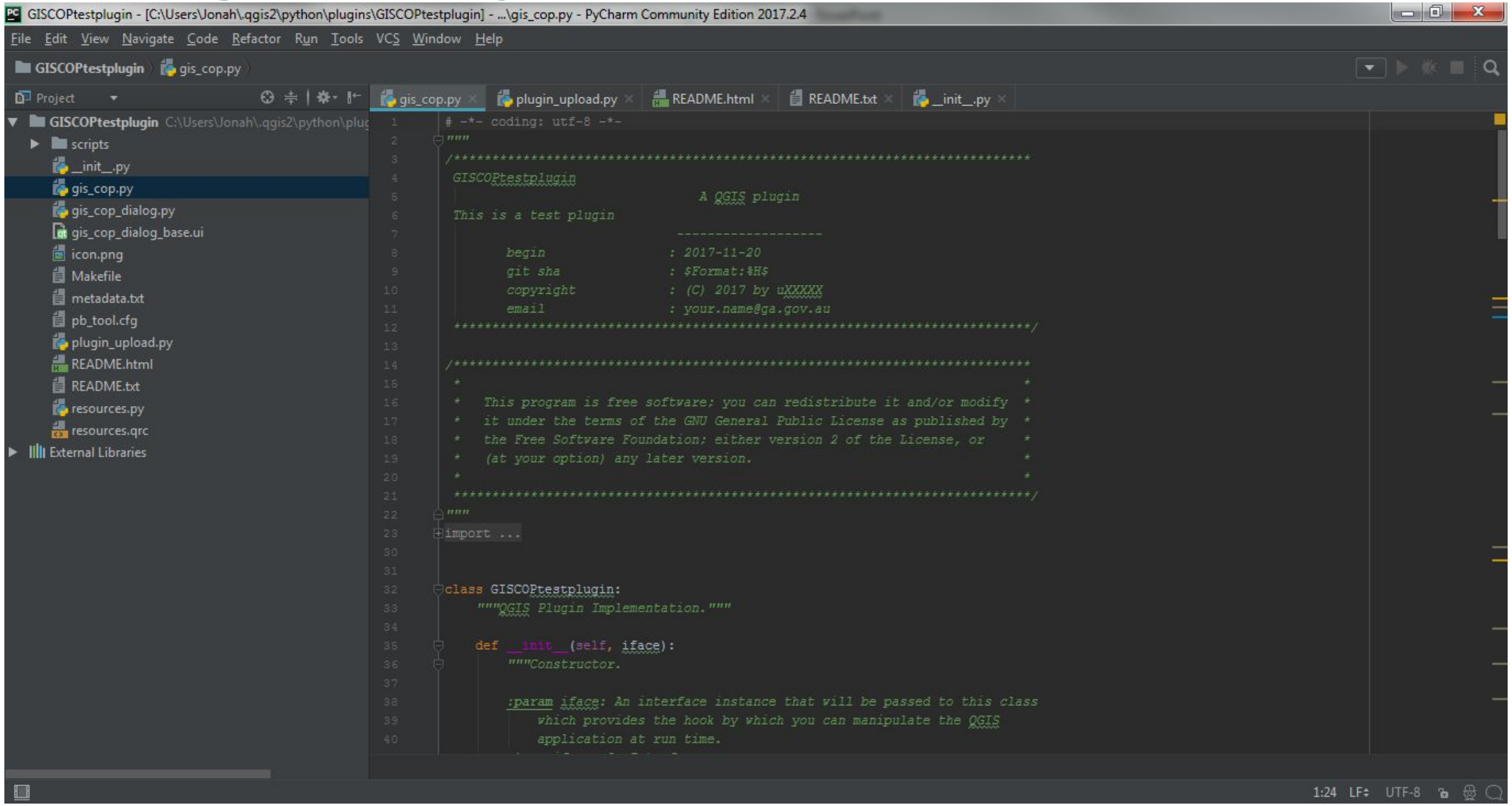
Property	Value
QObject	
objectName	PhotoLinkerDialogBase
QWidget	

Set up IDE (PyCharm) Windows Python Environment

https://nathanw.net/2014/05/10/setting-up-pycharm-for-pyqgis-and-qt/

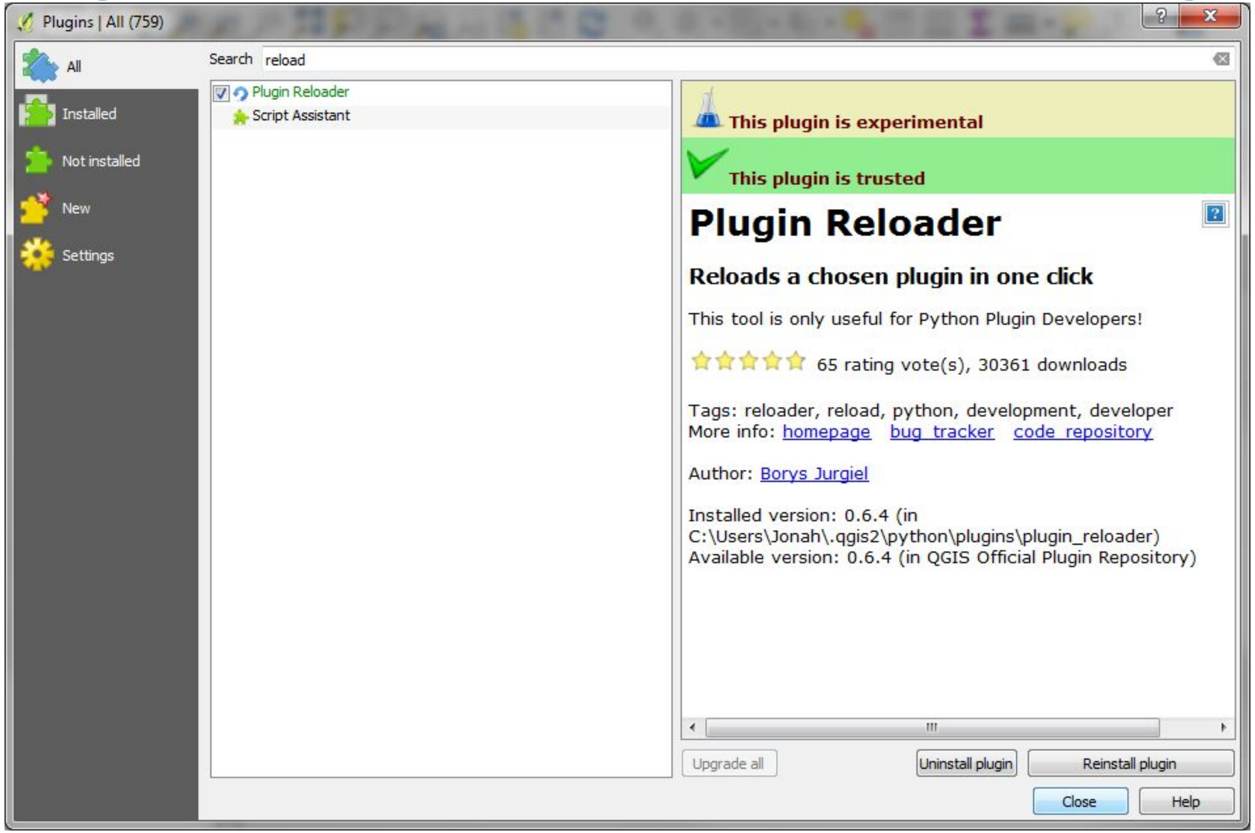
```
SET OSGEO4W_ROOT=C:\OSGeo4W64

SET QGISNAME=qgis

SET QGIS=%OSGEO4W_ROOT%\apps\%QGISNAME%

SET QGIS_PREFIX_PATH=%QGIS%

SET PYCHARM="C:\Program Files\JetBrains\PyCharm Community Edition 2017.2.3\bin\pycharm.exe"

CALL %OSGEO4W_ROOT%\bin\o4w_env.bat

SET PATH=%PATH%;%QGIS%\bin

SET PYTHONPATH=%QGIS%\python;%PYTHONPATH%

start "PyCharm aware of QGIS" /B %PYCHARM% %*
```

Open plugin with pycharm



```
# -*- coding: utf-8 -*-
"""
/***************************************************************************
 GISCOPtestplugin
                                 A QGIS plugin
 This is a test plugin
                             -------------------
        begin                : 2017-11-20
        git sha              : $Format:%H$
        copyright            : (C) 2017 by uXXXXX
        email                : your.name@ga.gov.au
 ***************************************************************************/

/***************************************************************************
 *                                                                         *
 *   This program is free software; you can redistribute it and/or modify  *
 *   it under the terms of the GNU General Public License as published by  *
 *   the Free Software Foundation; either version 2 of the License, or     *
 *   (at your option) any later version.                                   *
 *                                                                         *
 ***************************************************************************/
"""
import ...


class GISCOPtestplugin:
    """QGIS Plugin Implementation."""

    def __init__(self, iface):
        """Constructor.

        :param iface: An interface instance that will be passed to this class
            which provides the hook by which you can manipulate the QGIS
            application at run time.
```

Install Plugin Reloader and First Aid for testing



Search: reload

- [✓] Plugin Reloader
- Script Assistant

This plugin is experimental

This plugin is trusted

Plugin Reloader

Reloads a chosen plugin in one click

This tool is only useful for Python Plugin Developers!

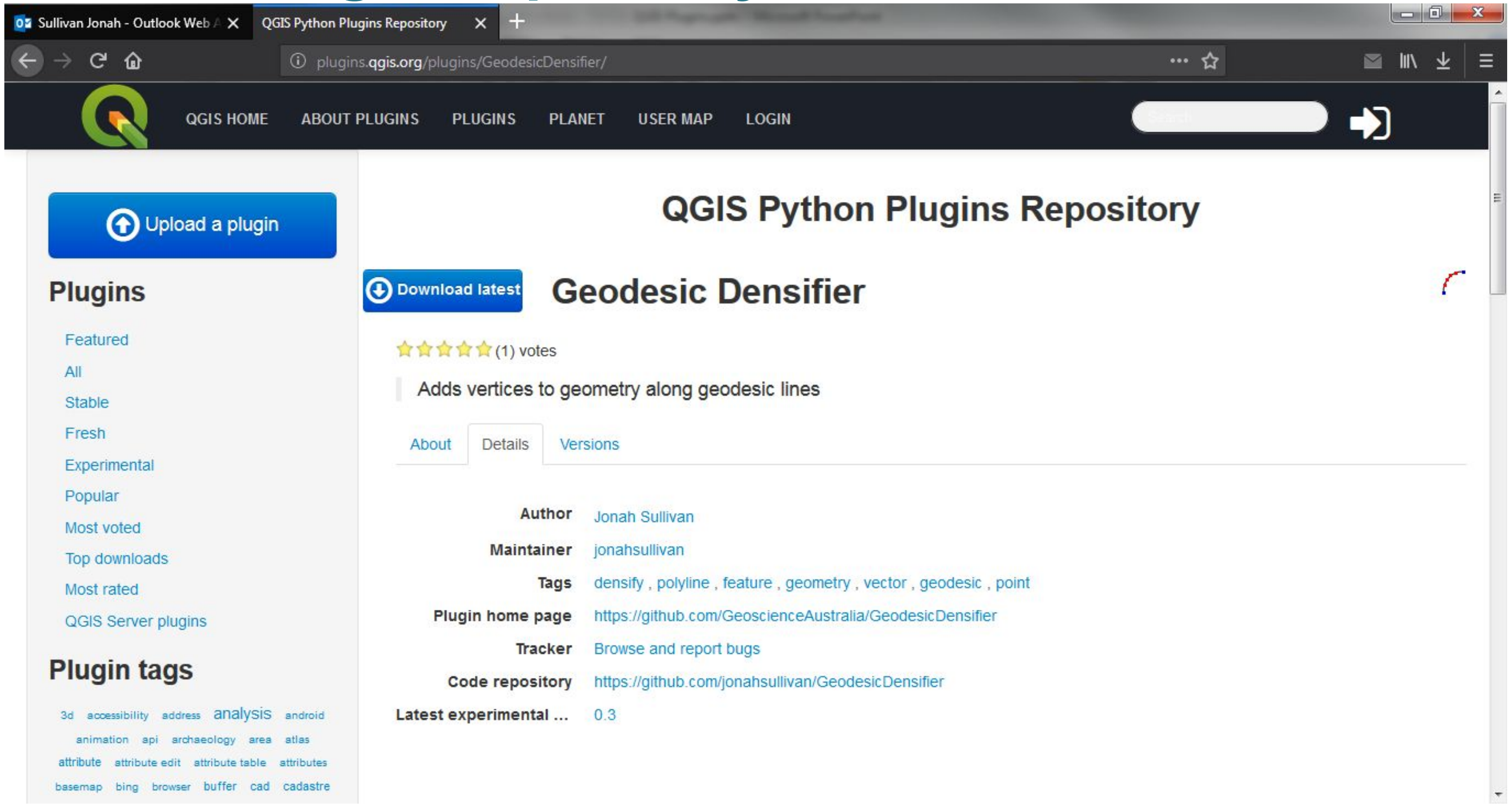
★★★★★ 65 rating vote(s), 30361 downloads

Tags: reloader, reload, python, development, developer
More info: homepage bug tracker code repository

Author: Borys Jurgiel

Installed version: 0.6.4 (in C:\Users\Jonah\.qgis2\python\plugins\plugin_reloader)
Available version: 0.6.4 (in QGIS Official Plugin Repository)

Upload to Plugin Repository


Sullivan Jonah - Outlook Web A
QGIS Python Plugins Repository
plugins.qgis.org/plugins/GeodesicDensifier/
QGIS HOME
ABOUT PLUGINS
PLUGINS
PLANET
USER MAP
LOGIN
Upload a plugin
Download latest

QGIS Python Plugins Repository

Plugins

- Featured
- All
- Stable
- Fresh
- Experimental
- Popular
- Most voted
- Top downloads
- Most rated
- QGIS Server plugins

Plugin tags

3d accessibility address analysis android animation api archaeology area atlas attribute attribute edit attribute table attributes basemap bing browser buffer cad cadastre

Geodesic Densifier

★★★★★ (1) votes

Adds vertices to geometry along geodesic lines

About | Details | Versions

Author	Jonah Sullivan
Maintainer	jonahsullivan
Tags	densify , polyline , feature , geometry , vector , geodesic , point
Plugin home page	https://github.com/GeoscienceAustralia/GeodesicDensifier
Tracker	Browse and report bugs
Code repository	https://github.com/jonahsullivan/GeodesicDensifier
Latest experimental ...	0.3

Overview of the PyQGIS and PyQt5 framework

PyQGIS: a generic term for the python framework in the QGIS environment

- qgis.core, qgis.gui, qgis.analysis
- pyqgis developer cookbook (examples):
  - https://docs.qgis.org/testing/en/docs/pyqgis_developer_cookbook/
- QGIS API documentation (python-based):
  - https://qgis.org/pyqgis/master/
- QGIS API documentation (C-based)
  - https://qgis.org/api/

Overview of the PyQGIS and PyQt5 framework

PyQt5: the python API for the QT application framework

- PyQt5.QtCore, PyQt5.QtGui, PyQt5.QtWidgets
- PyQt5 API documentation:
  - http://pyqt.sourceforge.net/Docs/PyQt5/

QGIS Processing Algorithms

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

```
>>> processing.algorithmHelp("native:buffer")

>>> algresult = processing.run("native:buffer",
                {'INPUT': '/data/lines.shp',
                 'DISTANCE': 100.0,
                 'SEGMENTS': 10,
                 'DISSOLVE': True,
                 'END_CAP_STYLE': 0,
                 'JOIN_STYLE': 0,
                 'MITER_LIMIT': 10,
                 'OUTPUT': 'memory'})

>>> buffered = algresult['OUTPUT']
```

Other python modules

```
>>>help('modules') #  for a full listing
```

Highlights:

- gdal/ogr
- matplotlib
- multiprocessing
- numpy
- networkx
- scipy
- shapely
- urllib
- wxpython

Querying the dialog form elements

The QGIS Plugin Builder creates a script that imports the QT Designer ui file as a class. The class is then imported into the main plugin script.

```
# import the dialog form as a class
from plugin_dialog import PluginDialog
# instantiate the dialog form class
def __init__(self):
    self.dlg = PluginDialog()
self.dlg.show()
# query the dialog class layerComboBox element’s value
self.dlg.LayerComboBox.currentLayer()
# query the dialog class radioButton element’s status
self.dlg.radioButton.isChecked()
```

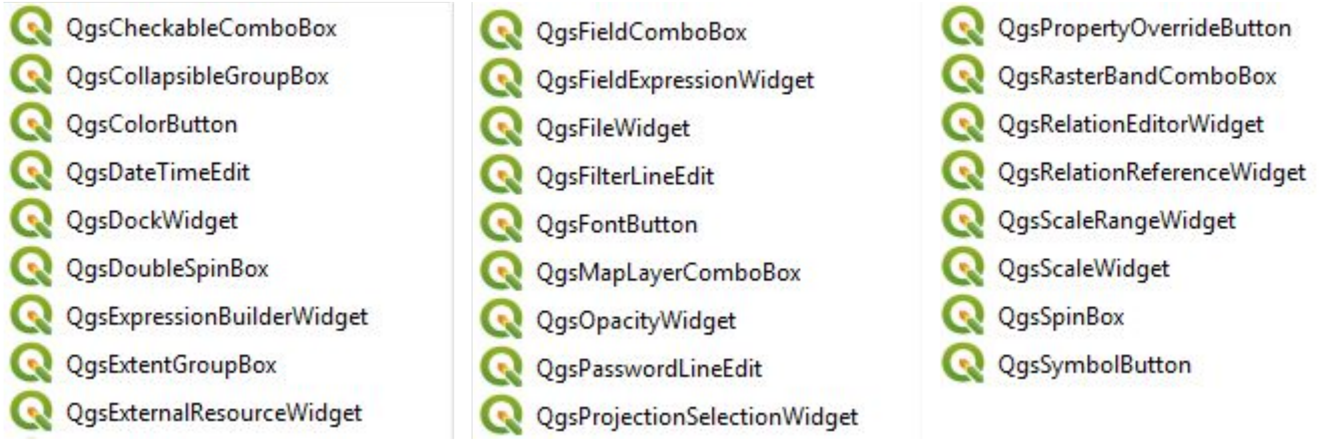
Setting values for the dialog form elements

Use the dialog box class to interact with it

```
# populate a comboBox element with sequential values
for i in range(0,10):
    self.dlg.comboBox.addItem(str(i))
# check a radioButton
self.dlg.radioButton.setChecked( True)
# set the text for a textLabel
self.dlg.textLabel.setText( “sample text”)
```

QGIS Custom Widgets

Avoid populating dialog form elements, use QGIS Custom Widgets



QgsCheckableComboBox
QgsCollapsibleGroupBox
QgsColorButton
QgsDateTimeEdit
QgsDockWidget
QgsDoubleSpinBox
QgsExpressionBuilderWidget
QgsExtentGroupBox
QgsExternalResourceWidget
QgsFieldComboBox
QgsFieldExpressionWidget
QgsFileWidget
QgsFilterLineEdit
QgsFontButton
QgsMapLayerComboBox
QgsOpacityWidget
QgsPasswordLineEdit
QgsProjectionSelectionWidget
QgsPropertyOverrideButton
QgsRasterBandComboBox
QgsRelationEditorWidget
QgsRelationReferenceWidget
QgsScaleRangeWidget
QgsScaleWidget
QgsSpinBox
QgsSymbolButton

Gotcha:

*.ui file needs to be manually edited

Fixing the *.ui file

INCORRECT (attempting to reference a C header file by default)

```
<customwidget>
   <class>QgsProjectionSelectionWidget</class>
   <extends>QWidget</extends>
   <header>qgsprojectionselectionwidget.h</header>
  </customwidget>
```

CORRECT

```
<customwidget>
   <class>QgsProjectionSelectionWidget</class>
   <extends>QWidget</extends>
   <header>qgis.gui</header>
  </customwidget>
```

PyQt5 Signals (user interaction with a GUI)

Interactive Programming relies on signals:

A signal is emitted when something of potential interest happens. A slot is a Python callable. If a signal is connected to a slot then the slot is called when the signal is emitted. If a signal isn’t connected then nothing happens. The code (or component) that emits the signal does not know or care if the signal is being used.

```
# slot - executes an action

def printValue(newValue):

    print(str(newValue))

# signal - initiates an action

self.dlg.spinBox.valueChanged.connect( printValue(self.dlg.spinbox.value())
```

Learning by Doing: *make your own plugin*

- Using your own original idea
- Using a provided scenario
  - **Hello, World!**, first step in interactive python
  - **Color Layer**, change a layer’s colour
  - **Line Slope**, add a line’s slope from raster

Pre-Built Plugin: Hello World, *print "Hello, World!"*

Because it is very simple, it is often used to illustrate the basic syntax of a programming language and is often the first program people write.

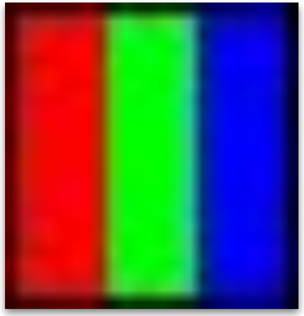
First Timer Task: make your first plugin

- Use Plugin Builder to make a template
- Use PyCharm or a text editor to write functionality
- Make the plugin print "Hello, World!" to console with the Ok button

Beginner Task: make it interactive

- Make a Qlabel or QTextEdit change to "Hello, World!" and then back to blank with using a QPushButton

Pre-Built Plugin: Color Layer, *change a layer's colour*



https://github.com/jonahsullivan/FOSS4G-Oceania-2018/tree/master/ColorLayer

Beginner Task: make it interactive!

- Fork the repository (a minimal yet functional plugin)
- Use the QgsMapLayerComboBox to select a layer
- Use signals to change the colour when each radio button is clicked

Advanced Task:

- Build it from scratch
  - make a template
  - design a GUI
    - where a user can select a layer and a colour
  - write the functionality
    - to change the colour of a layer's symbols
  - debug your code
    - using First Aid or PyCharm and Plugin Reloader

Pre-Built Plugin: Line Slope, *add line slope from raster*

https://github.com/jonahsullivan/FOSS4G-Oceania-2018/tree/master/LineSlope

Advanced Task: implement a script as a plugin

- Fork the repository (includes a functional script and sample data)
- make a plugin template
- design a GUI
    - a user selects a pre-existing line vector layer and raster layer as inputs, and an output layer
- write the functionality
    - create a new output layer
        - memory or tempfile
    - sample the raster at each end of the input layer
    - calculate slope of each line feature
    - write each feature and slope attribute to output layer
- debug your code
    - using First Aid or PyCharm and Plugin Reloader

Author:

Jonah Sullivan, GISP-AP **|** Maritime Jurisdiction Advisor

Georegulation Section

Environmental Geoscience Division

e jonah.sullivan@ga.gov.au

t +61 2 6249 9516 www.ga.gov.au