



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

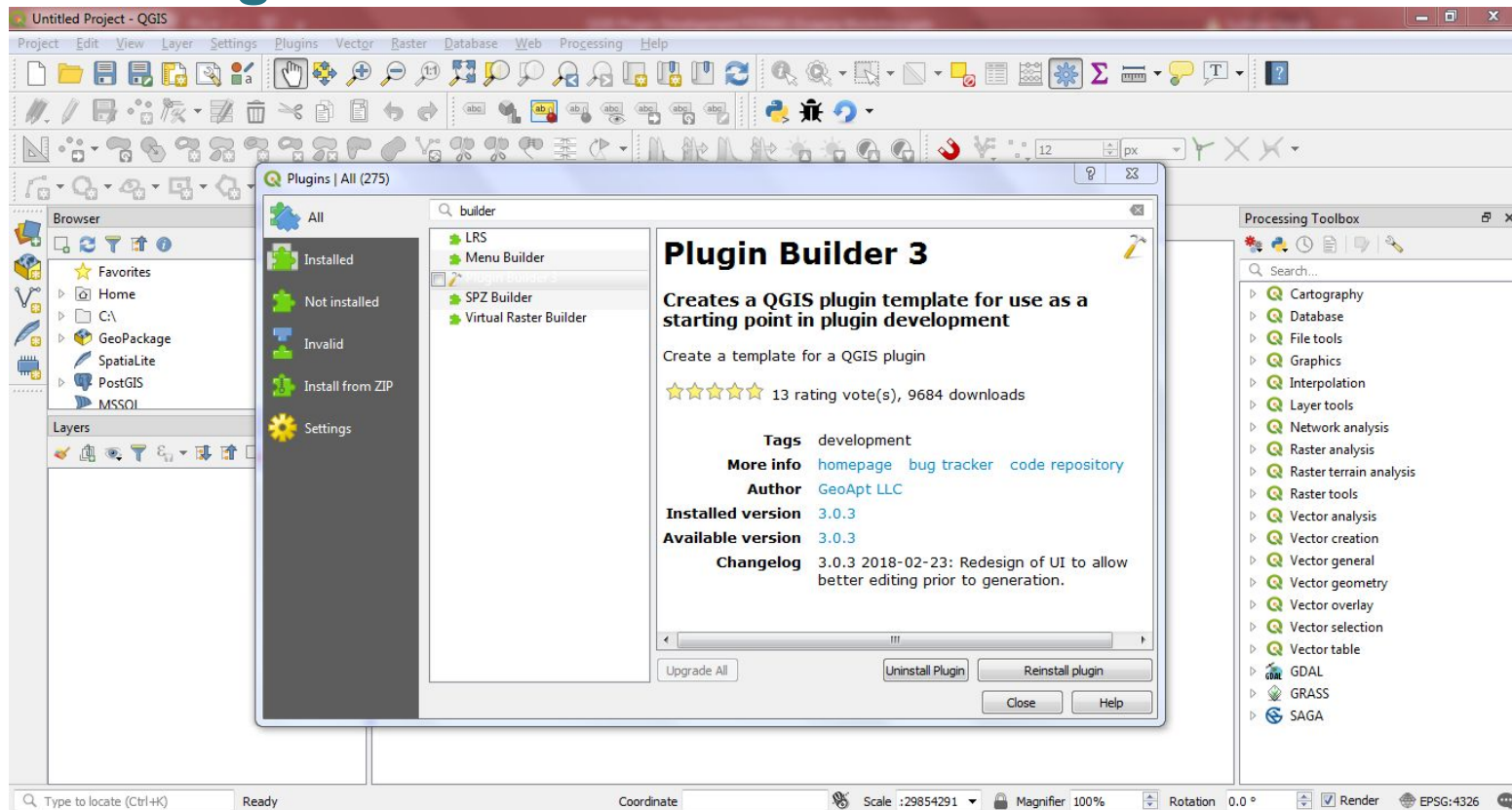
- Plugins are python-based (not C, this is possible but discouraged).
- QGIS3 will break your QGIS2 plugin. http://qgis.org/api/api_break.html
 - <https://github.com/qgis/QGIS/wiki/Plugin-migration-to-QGIS-3>
- Essential resources:
 - https://docs.qgis.org/testing/en/docs/pyqgis_developer_cookbook/plugins.html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

Time to open QGIS and install the “Plugin Builder” plugin.

Install Plugin Builder



First Form: General Information

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

Help <Previous Next > Cancel

CapWords

HelloWorld

Title Case

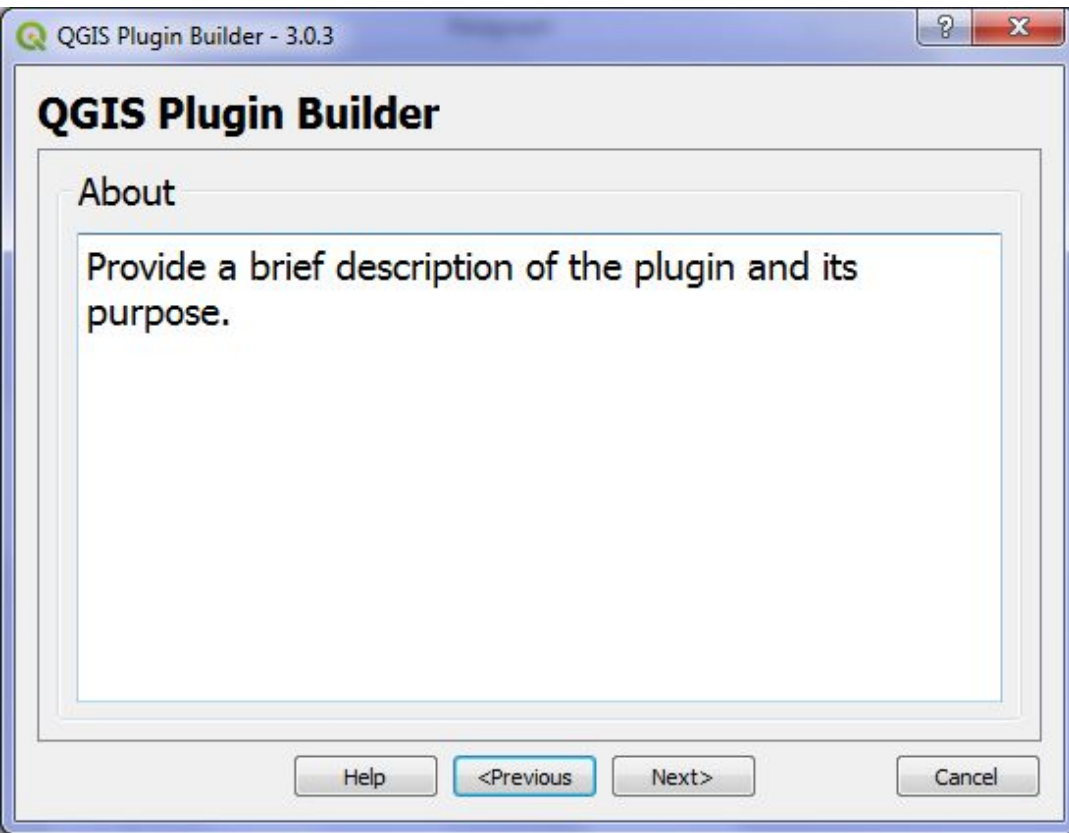
Hello World

Sentence case

Print "Hello, World!"

lowercase_with_underscore hello_world

Second Form: Description



The screenshot shows a window titled "QGIS Plugin Builder - 3.0.3". The main area is labeled "QGIS Plugin Builder" and contains a tab titled "About". Inside the "About" tab, there is a text area with the placeholder text "Provide a brief description of the plugin and its purpose." At the bottom of the window, there are four buttons: "Help", "<Previous", "Next>", and "Cancel".

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:

Text for the menu item:

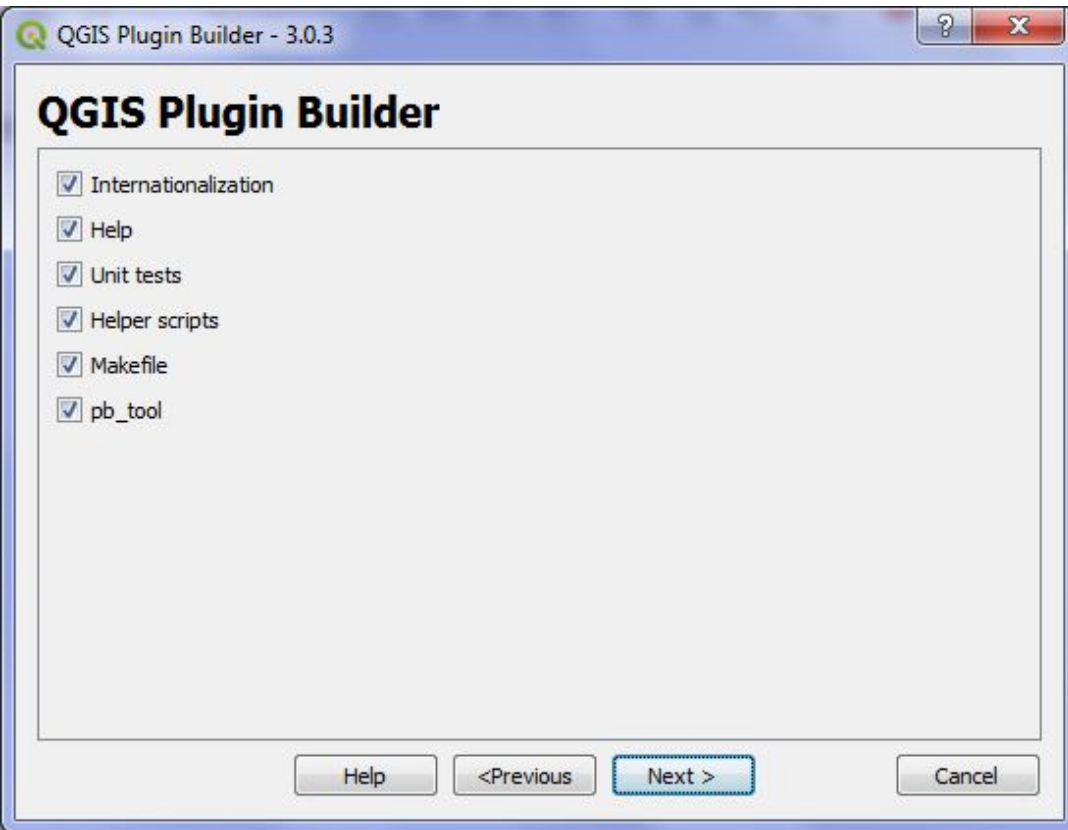
Menu:

Help <Previous Next> Cancel

- Tool button with dialog
- Tool button with dialog
- Tool button with dock widget
- Processing Provider

Hello World

Fourth Form: Optional Extras



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

Repository

Publication (recommended Items)

Home page

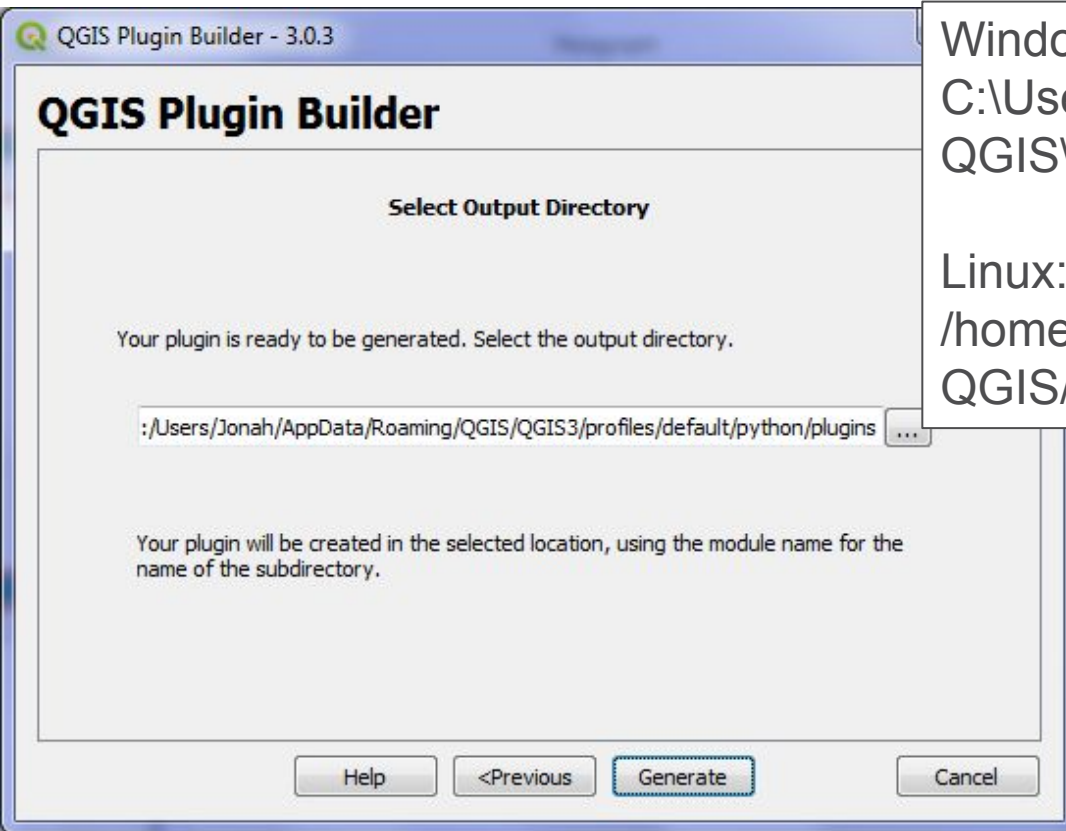
Tags ...

☐ Flag the plugin as experimental

Help <Previous Next > Cancel

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

Last Form: Output Directory



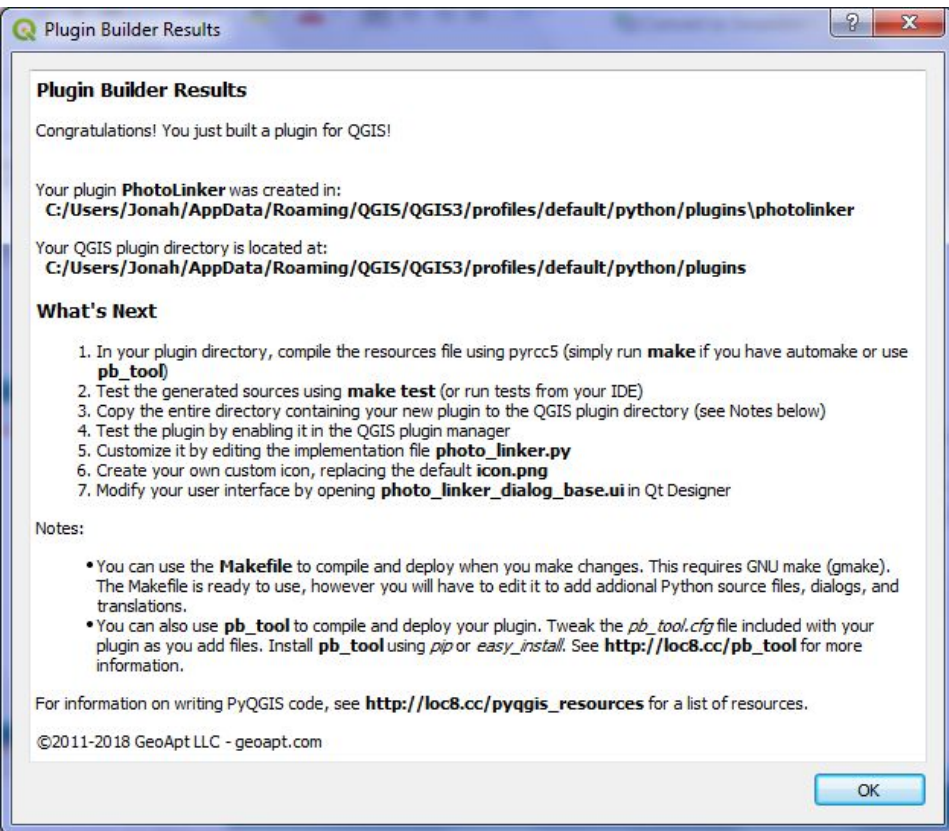
Windows:

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

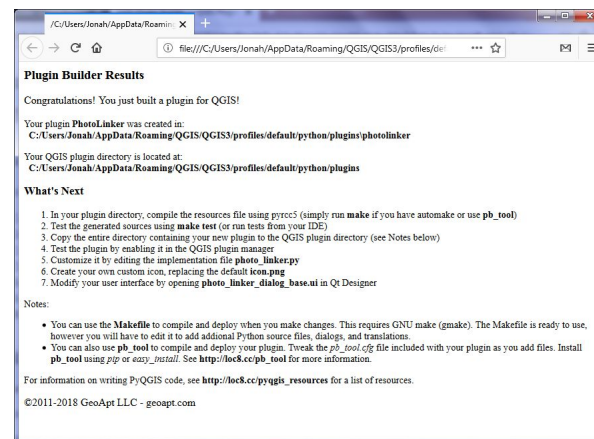
Linux:

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

Results



Also saved as a html file
in the plugin directory



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.

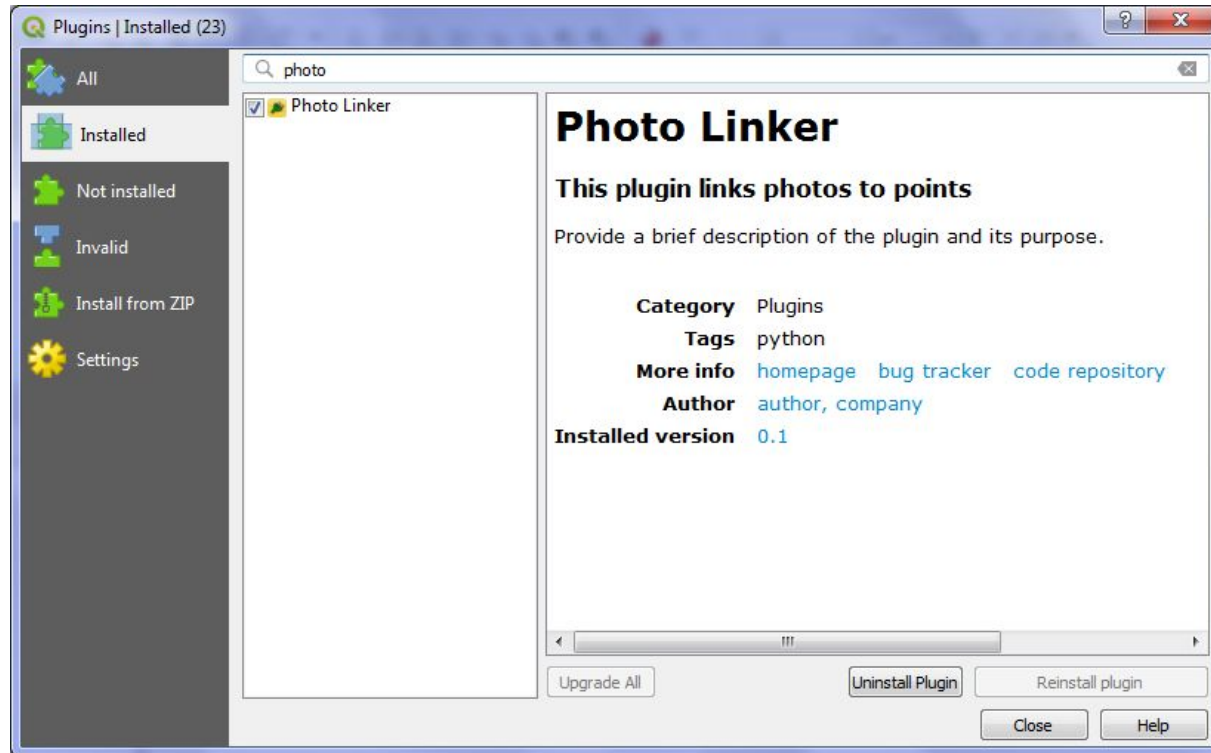
```
OSGeo4W Shell
C:\>py3_env
C:\>SET PYTHONPATH=
C:\>SET PYTHONHOME=C:\OSGeo4W\1\apps\Python37
C:\>PATH C:\OSGeo4W\1\apps\Python37;C:\OSGeo4W\1\apps\Python37\Scripts;{app};C:\OSGeo4W\1\apps\Python27\Scripts;C:\OSGeo4W\1\bin;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\system32\Wbem;C:\Program Files\MSI\MSI4.3\bin\x64
C:\>qt5_env
C:\>pyrcc5.bat
PyQt5 resource compiler
Usage: pyrcc5 [options] <input>

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.



Create a custom icon



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

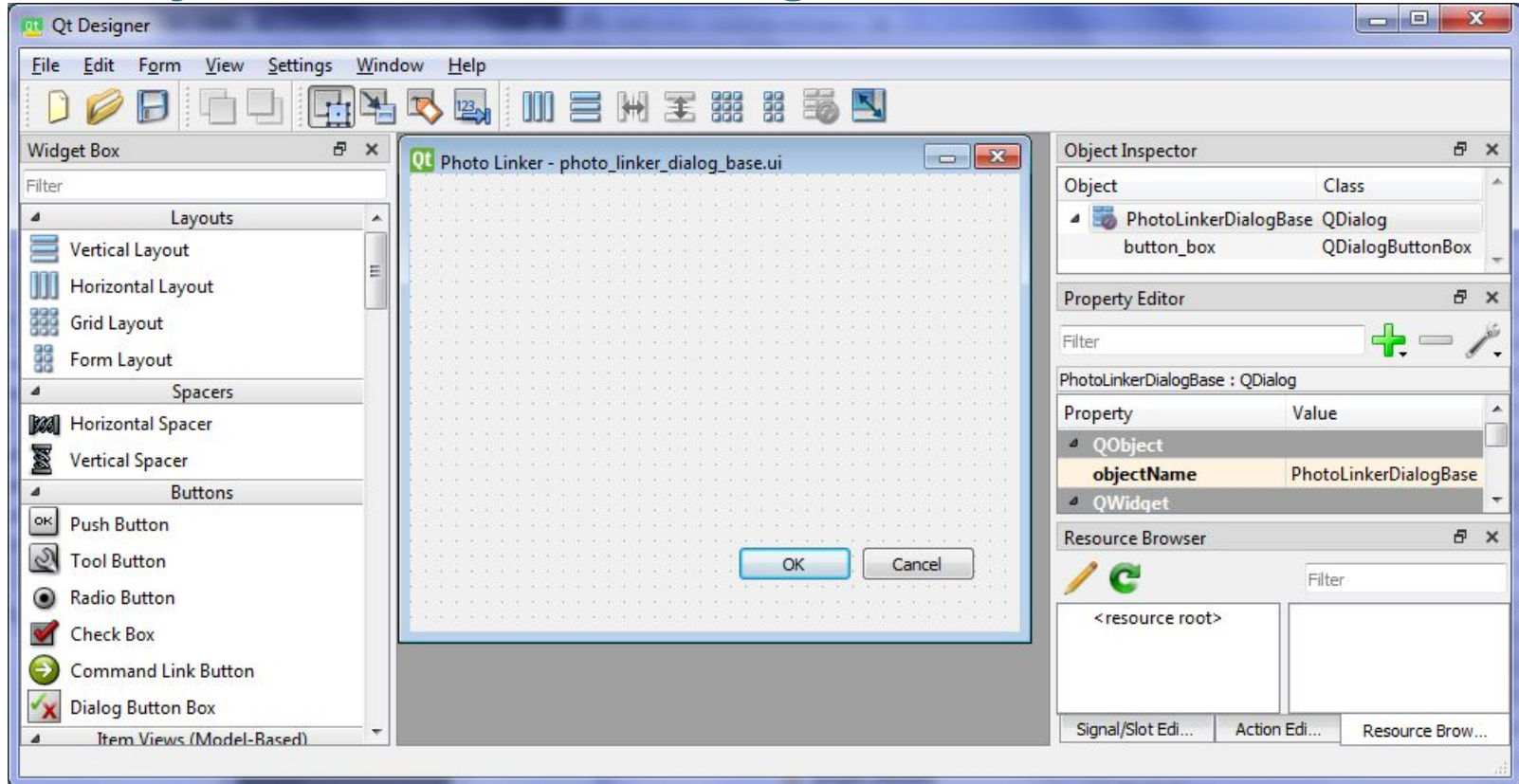
Uploading to the plugin repository

The PyQGIS and PyQt frameworks

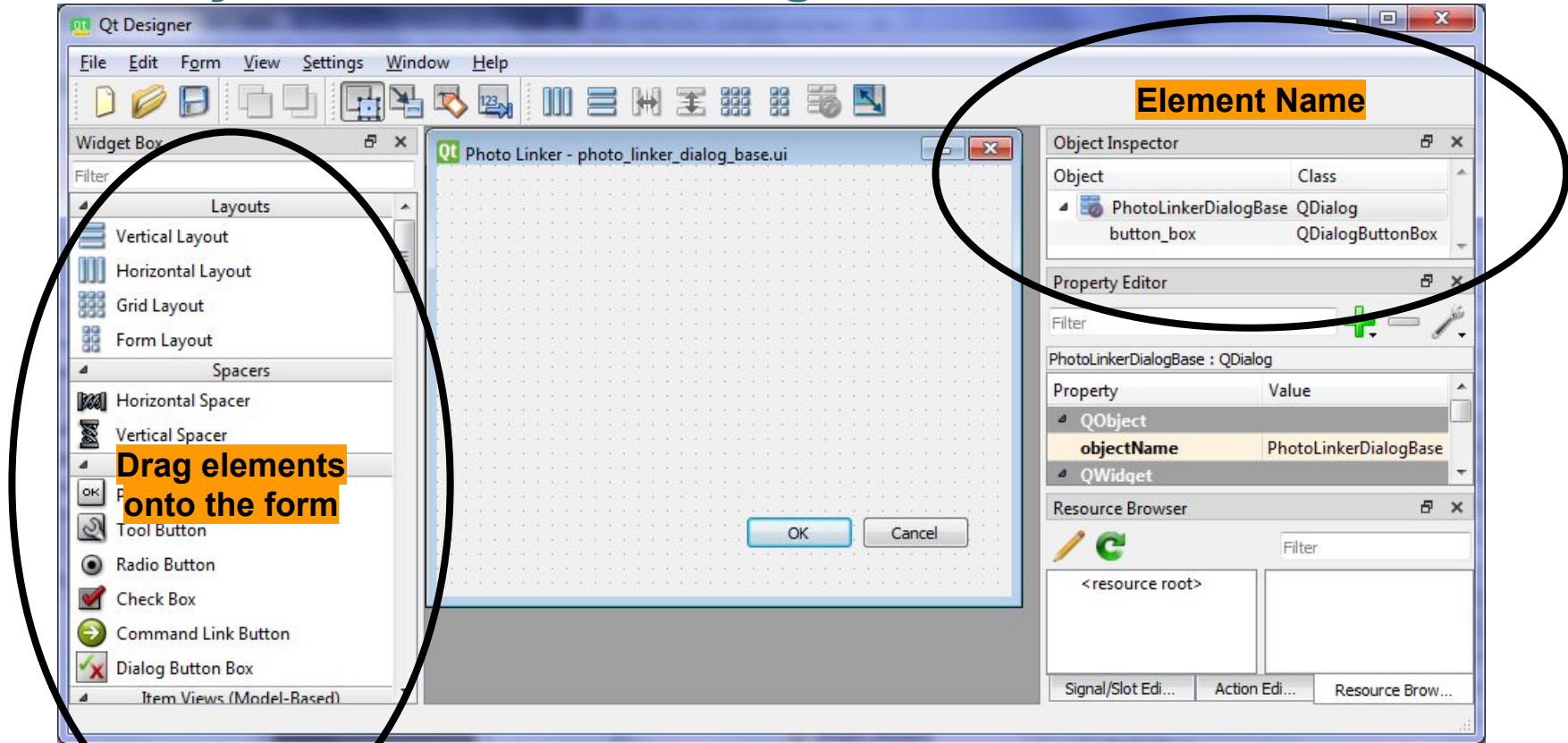
Making your own plugin

- Using your own original idea
- Using a provided scenario

Create your GUI with QtDesigner



Create your GUI with QtDesigner



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"
```

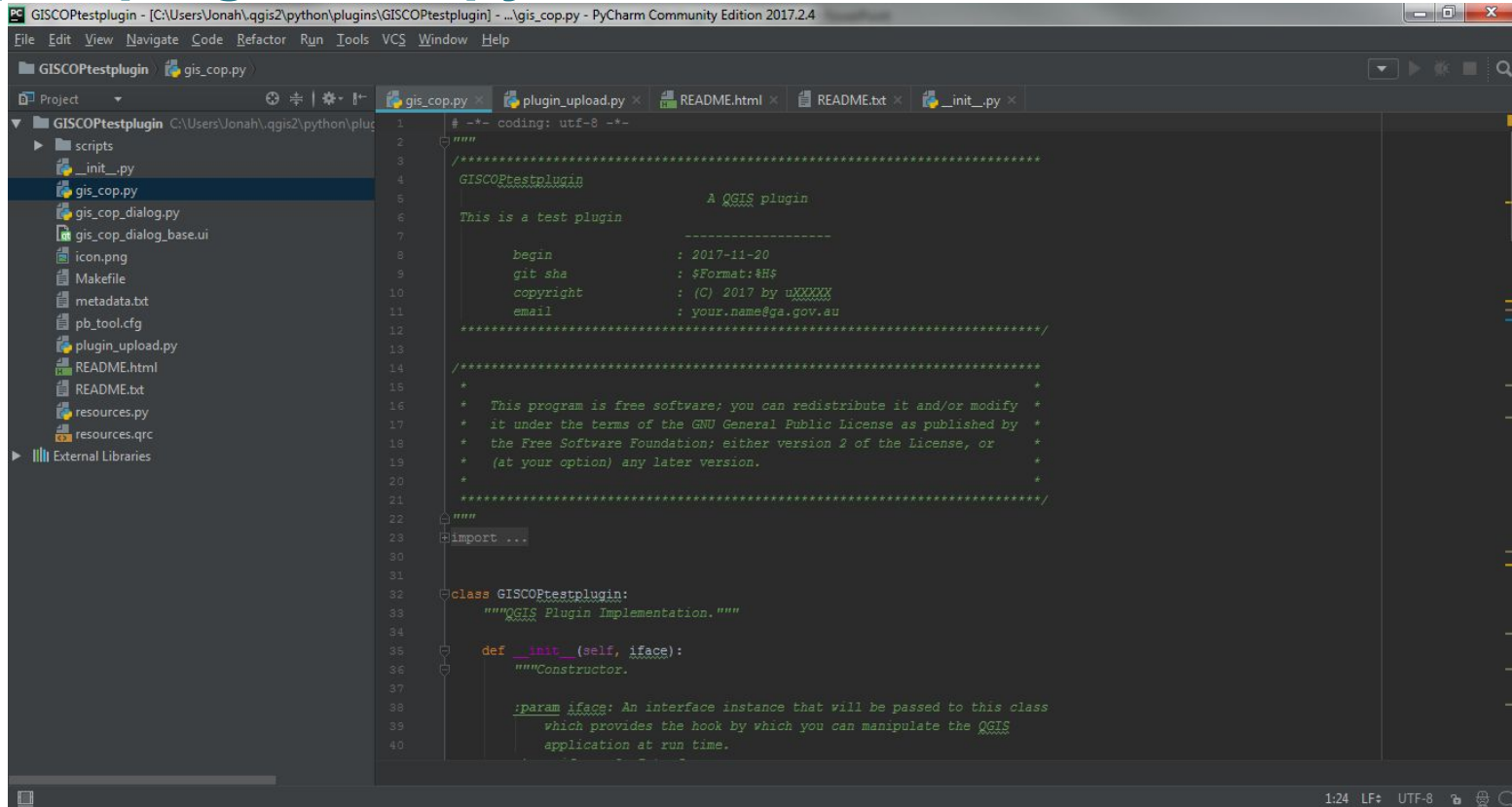
```
CALL %OSGEO4W_ROOT%\bin\o4w_env.bat
```

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

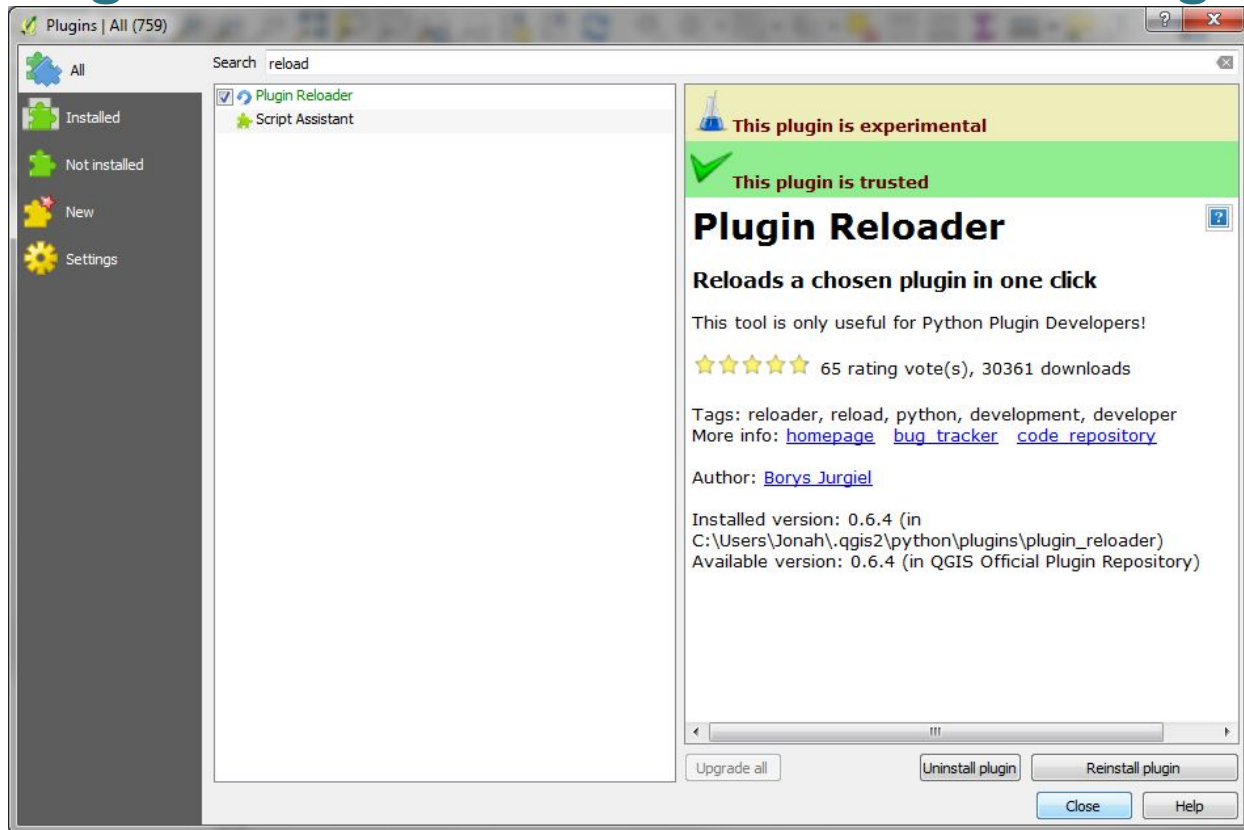
```
SET PYTHONPATH=%QGIS%\python;%PYTHONPATH%
```

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


Open plugin with pycharm



Install Plugin Reloader and First Aid for testing



Upload to Plugin Repository

The screenshot shows a web browser window with the URL `plugins.qgis.org/plugins/GeodesicDensifier/`. The page features a dark navigation bar with the QGIS logo and links to [QGIS HOME](#), [ABOUT PLUGINS](#), [PLUGINS](#), [PLANET](#), [USER MAP](#), and [LOGIN](#). On the left, a sidebar contains a blue 'Upload a plugin' button and a 'Plugins' section with filters like 'Featured', 'All', 'Stable', 'Fresh', 'Experimental', 'Popular', 'Most voted', 'Top downloads', 'Most rated', and 'QGIS Server plugins'. Below this is a 'Plugin tags' section with various tags like '3d', 'accessibility', 'address', etc. The main content area displays the 'Geodesic Densifier' plugin page, which includes a 'Download latest' button, a star rating of 5 stars (1 vote), and a description: 'Adds vertices to geometry along geodesic lines'. There are tabs for 'About', 'Details', and 'Versions'. A metadata table lists the author (Jonah Sullivan), maintainer (jonahsullivan), tags (density, 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), and latest experimental version (0.3).

Upload a plugin

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

QGIS Python Plugins Repository

Download latest

Geodesic Densifier

★★★★★ (1) votes

Adds vertices to geometry along geodesic lines

About Details Versions

Author	Jonah Sullivan
Maintainer	jonahsullivan
Tags	density , 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/>

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

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

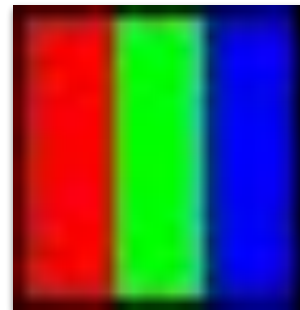
- Make a QLabel or QTextEdit change to “Hello, World!” and then back to blank with using a QPushButton

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