QGIS Plugin Development

Purpose

The purpose of this document is to provide resources to help fellow geogeeks members who are interested in QGIS Plugin development with tips, tricks and troubleshooting to help them get started.This a live document feel free to comment and contribute.

Contributors

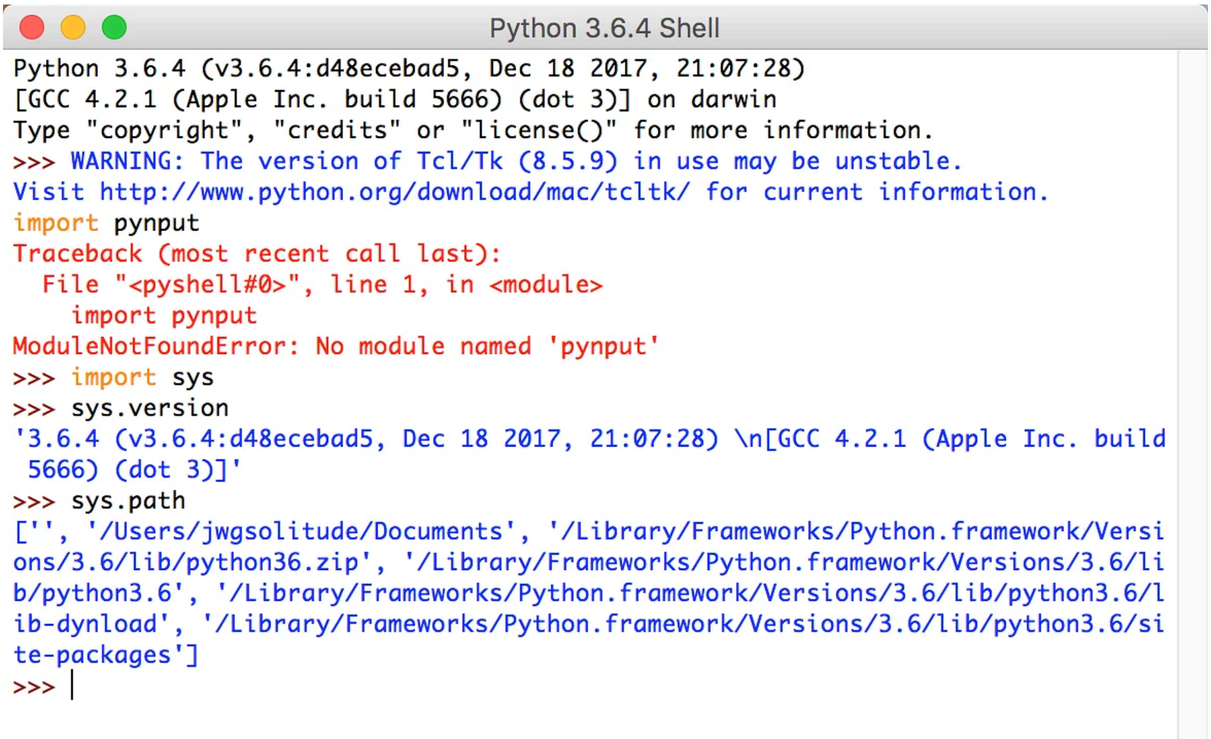
Jack Green	jackgreengis@gmail.com
Tom Lynch	tom@trailmarker.io

Choosing an IDE

The first thing you will need to start building a plugin is an IDE (Integrated development environment) which is just a fancy text editor to make it easier to write and maintain your code. There are lot of different IDE's available for python coding and it comes down to personal preference of which one you choose. Below is a list of some of the more common options and a bit of info on them:

- IDLE
- PyCharm
- VS Code

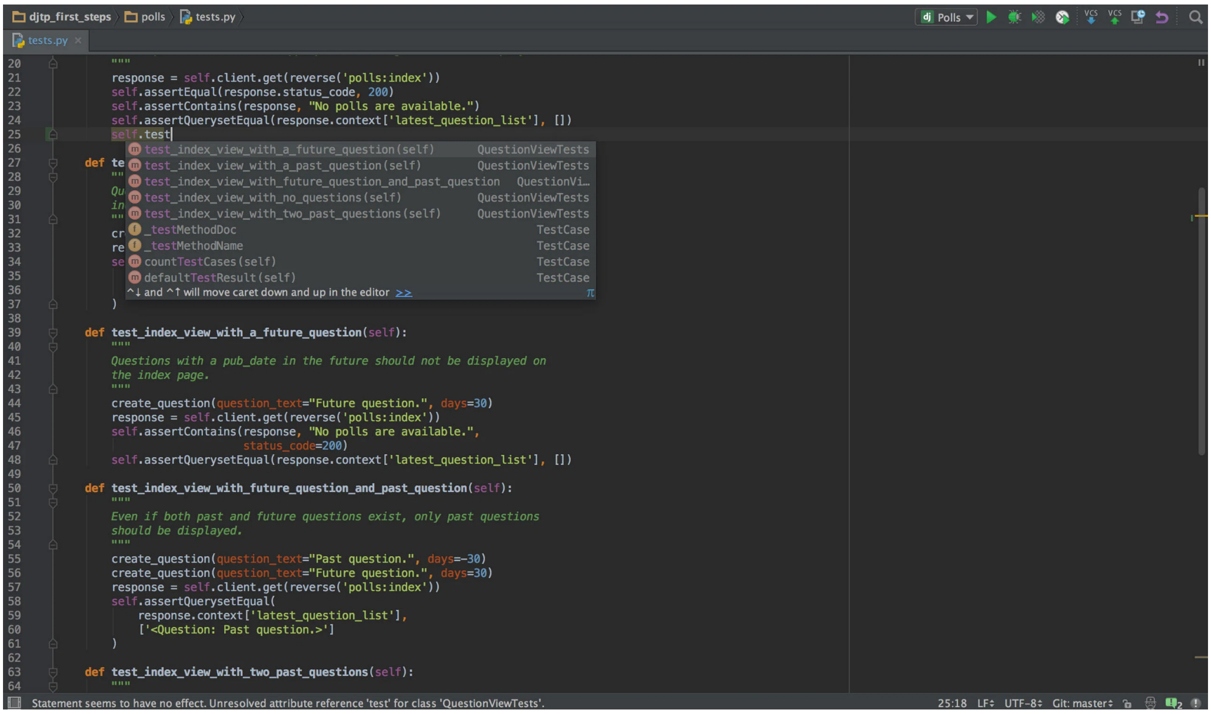
IDLE



```
Python 3.6.4 (v3.6.4:d48ecebad5, Dec 18 2017, 21:07:28)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable.
Visit http://www.python.org/download/mac/tcltk/ for current information.
import pynput
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    import pynput
ModuleNotFoundError: No module named 'pynput'
>>> import sys
>>> sys.version
'3.6.4 (v3.6.4:d48ecebad5, Dec 18 2017, 21:07:28) \n[GCC 4.2.1 (Apple Inc. build
 5666) (dot 3)]'
>>> sys.path
['', '/Users/jwgsolitude/Documents', '/Library/Frameworks/Python.framework/Versi
ons/3.6/lib/python36.zip', '/Library/Frameworks/Python.framework/Versions/3.6/li
b/python3.6', '/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/l
ib-dynload', '/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/si
te-packages']
>>> 
```

<insert description>

PyCharm



```
        """
        response = self.client.get(reverse('polls:index'))
        self.assertEqual(response.status_code, 200)
        self.assertContains(response, "No polls are available.")
        self.assertQuerysetEqual(response.context['latest_question_list'], [])
        self.test

    def te
        """
        Qu
        in
        """
        cr
        re
        se

        )

    def test_index_view_with_a_future_question(self):
        """
        Questions with a pub_date in the future should not be displayed on
        the index page.
        """
        create_question(question_text="Future question.", days=30)
        response = self.client.get(reverse('polls:index'))
        self.assertContains(response, "No polls are available.",
                            status_code=200)
        self.assertQuerysetEqual(response.context['latest_question_list'], [])

    def test_index_view_with_future_question_and_past_question(self):
        """
        Even if both past and future questions exist, only past questions
        should be displayed.
        """
        create_question(question_text="Past question.", days=-30)
        create_question(question_text="Future question.", days=30)
        response = self.client.get(reverse('polls:index'))
        self.assertQuerysetEqual(
            response.context['latest_question_list'],
            ['<Question: Past question.>']
        )

    def test_index_view_with_two_past_questions(self):
        """
```

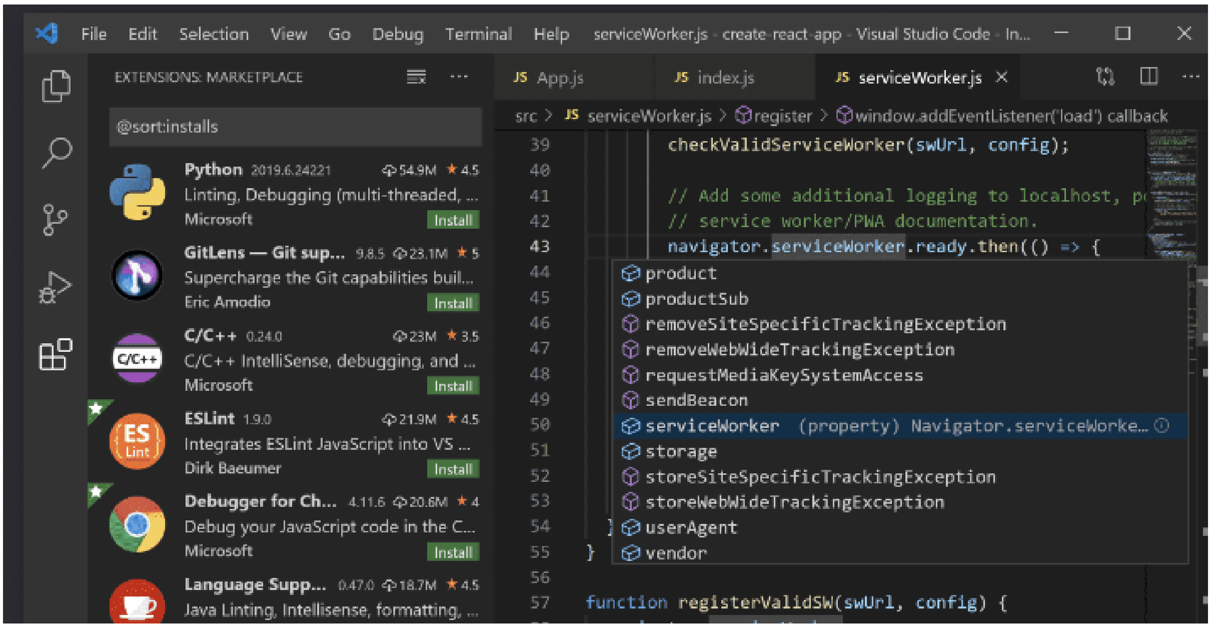
```
m test_index_view_with_a_future_question(self)       QuestionViewTests
m test_index_view_with_a_past_question(self)         QuestionViewTests
m test_index_view_with_future_question_and_past_question   QuestionVi…
m test_index_view_with_no_questions(self)            QuestionViewTests
m test_index_view_with_two_past_questions(self)      QuestionViewTests
f _testMethodDoc                                             TestCase
f _testMethodName                                            TestCase
m countTestCases(self)                                       TestCase
m defaultTestResult(self)                                    TestCase
^↓ and ^↑ will move caret down and up in the editor >>              π
```

Statement seems to have no effect. Unresolved attribute reference 'test' for class 'QuestionViewTests'.

25:18 LF÷ UTF-8÷ Git: master÷

<insert description>

VS Code


File
Edit
Selection
View
Go
Debug
Terminal
Help
serviceWorker.js - create-react-app - Visual Studio Code - In...
EXTENSIONS: MARKETPLACE
JS App.js
JS index.js
JS serviceWorker.js
src > JS serviceWorker.js > register > window.addEventListener('load') callback
product
productSub
removeSiteSpecificTrackingException
removeWebWideTrackingException
requestMediaKeySystemAccess
sendBeacon
serviceWorker (property) Navigator.serviceWorke...
storage
storeSiteSpecificTrackingException
storeWebWideTrackingException
userAgent
vendor

@sort:installs

Python 2019.6.24221 54.9M ★ 4.5
Linting, Debugging (multi-threaded, ...
Microsoft Install

GitLens — Git sup... 9.8.5 23.1M ★ 5
Supercharge the Git capabilities buil...
Eric Amodio Install

C/C++ 0.24.0 23M ★ 3.5
C/C++ IntelliSense, debugging, and ...
Microsoft Install

ESLint 1.9.0 21.9M ★ 4.5
Integrates ESLint JavaScript into VS ...
Dirk Baeumer Install

Debugger for Ch... 4.11.6 20.6M ★ 4
Debug your JavaScript code in the C...
Microsoft Install

Language Supp... 0.47.0 18.7M ★ 4.5
Java Linting, Intellisense, formatting, ...

```
    checkValidServiceWorker(swUrl, config);

    // Add some additional logging to localhost, p
    // service worker/PWA documentation.
    navigator.serviceWorker.ready.then(() => {

  }
}

function registerValidSW(swUrl, config) {
```

If you have come from a coding background then you would have more than likely used Visual Studio Professional at some point. VS Code is a comfortable starting point for your python development if that is the case.

It is free to download and contains a lot of good extensions like Python to provide intellisense, auto-indentation and colour coding to assist in your development. When coupled with additional QGIS Plugins active debugging of your plugins is available.

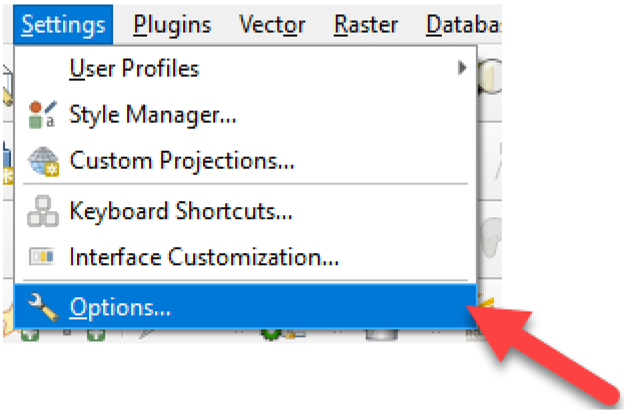
Set up a Development Environment

You will want to set yourself up with a working area to build and test your plugins. This can be anywhere you like on your local hard drive. You may also want to consider using source control (such as github) to version your code.

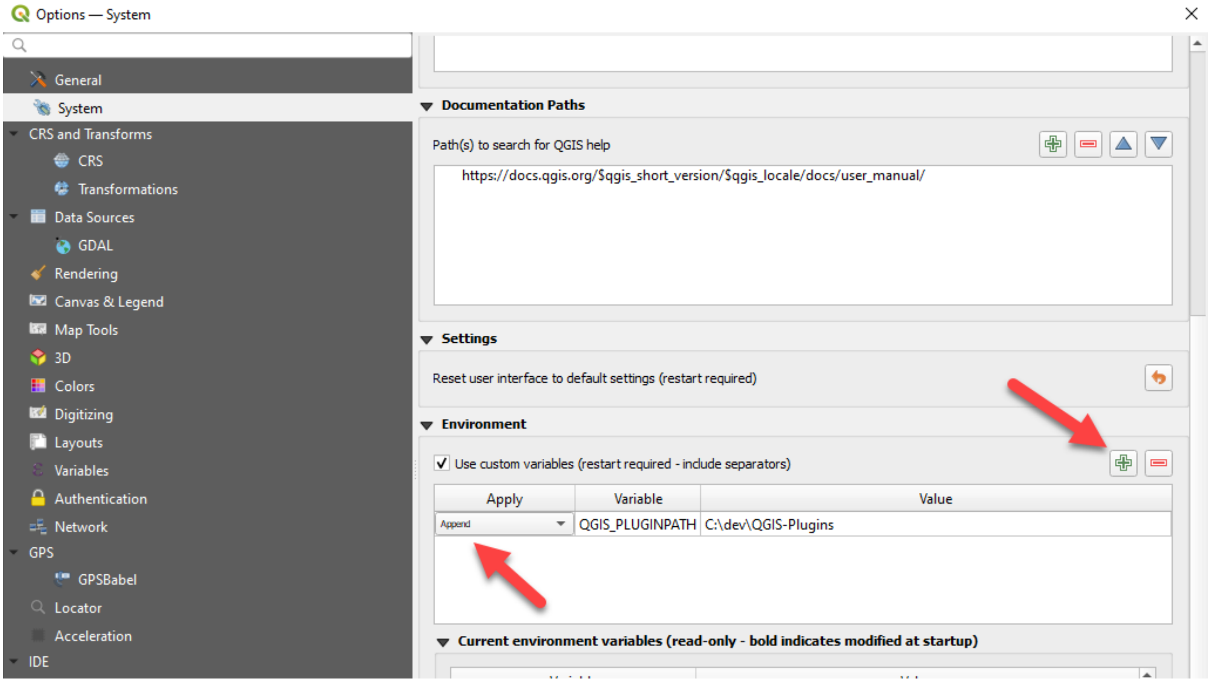
Plugin Dev Location

Once you have a location sorted you will need to tell QGIS where it is so your plugins can be loaded directly from there for testing in QGIS.

1. Got to Settings > Options…



2. Under System scroll down to Environment and add a new custom variable
    a. Apply: Append
    b. Variable: QGIS_PLUGINPATH
    c. Value: Parent directory for your plugins, for example if you are working on a plugin in `C:\dev\QGIS-Plugins\Cool-New-Plugin`, the value should be `C:\dev\QGIS-Plugins`



Path(s) to search for QGIS help

https://docs.qgis.org/$qgis_short_version/$qgis_locale/docs/user_manual/

Reset user interface to default settings (restart required)

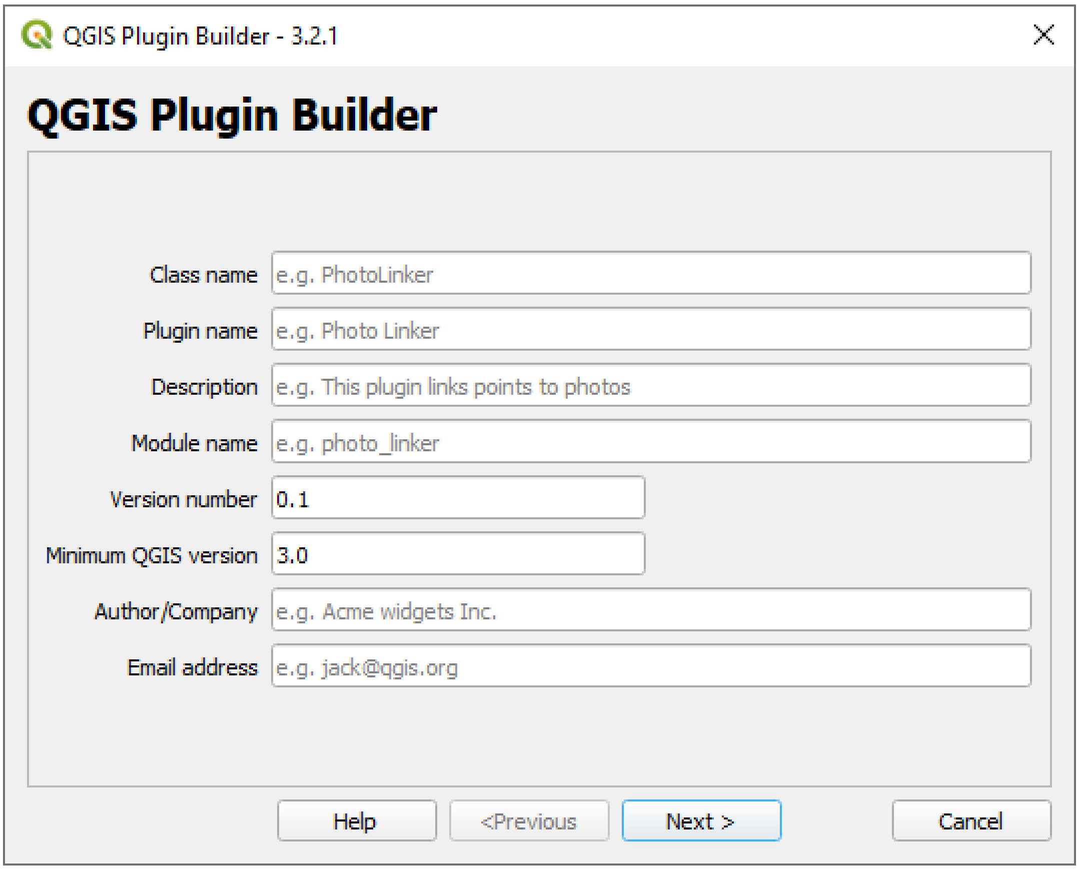
☑ Use custom variables (restart required - include separators)

Apply	Variable	Value
Append	QGIS_PLUGINPATH	C:\dev\QGIS-Plugins

Get Helper Plugins

There are bunch of plugins that will help you on your plugin journey, here are a few you should look at installing

Plugin Builder 3



QGIS Plugin Builder - 3.2.1

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

This is a great plugin to get you started on creating your first plugin. It creates a new plugin for you with all the different files (there are plenty) that you need in order to build a plugin. A must have for a new developer.

Plugin Reloader

A really simple plugin that let you Reload / Refresh your plugin after you make some changes without the need to close and reopen QGIS or adding the plugin to QGIS again.

Debugvs

Only relevant if you are using VS Code for your development, this plugin will allow you to actively debug your plugin within QGIS.

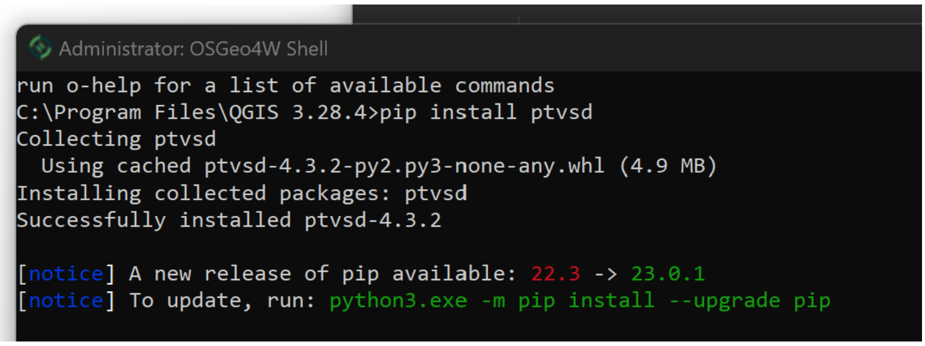
To use debugvs you will also need to have a `launch.json` file to allow the code to be debugged. This can be the same file for all of your plugins.

To set up debugvs, follow these steps:

1. In QGIS, install debugvs using the *Plugins* | *Manage and Install Plugins* feature
2. debugvs requires an additional Python package called `ptvsd` to be installed.
    a. Open the OSGeo4W Shell as Administrator (on Windows, search your Start Menu for this tool which is packaged with QGIS)
    b. Use the pip package manager to install the required package via the command

    ```
    pip install ptvsd
    ```

    as shown below (ignore the pip version warning—this appears on QGIS 3.28.34 LTR but will vary on different QGIS versions).



Administrator: OSGeo4W Shell

```
run o-help for a list of available commands
C:\Program Files\QGIS 3.28.4>pip install ptvsd
Collecting ptvsd
  Using cached ptvsd-4.3.2-py2.py3-none-any.whl (4.9 MB)
Installing collected packages: ptvsd
Successfully installed ptvsd-4.3.2

[notice] A new release of pip available: 22.3 -> 23.0.1
[notice] To update, run: python3.exe -m pip install --upgrade pip
```

3. In QGIS, use the menu option *Plugins* | *Enable Debug for Visual Studio* | *Enable Debug for Visual Studio* to ask debugvs to allow Visual Studio Code to attach to your running QGIS (You will need to restart QGIS at least once during the overall process of installing the plugin and its supporting package).
4. Create a folder called .vscode in your plugin directory (if one doesn't already exist)
5. Create a text file called launch.json in this folder with the following in the file

```
{
    "version": "0.2.0",
    "configurations": [

        {
            "name": "Python: QGIS Plugin",
            "type": "python",
            "request": "attach",
            "port": 5678,
            "host": "localhost"
        }
    ]
}
```

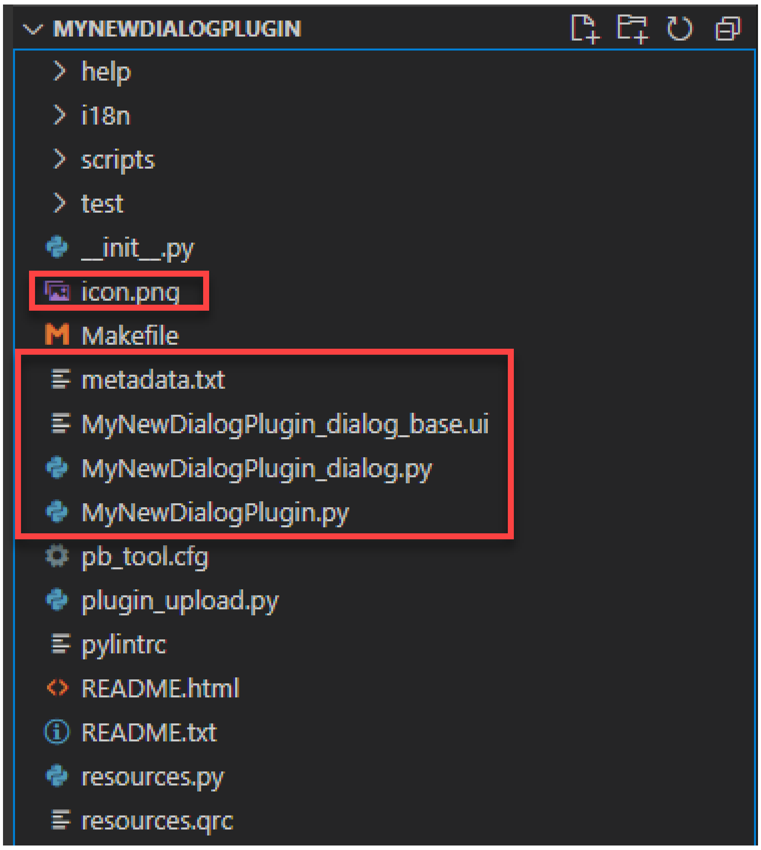
You can then copy and paste this folder/file into all your plugins to enable them for debugging.

Basic Plugin Structures

Button with Dialog

Click a button to launch a dialog window where you do some stuff. An example being the Select Features by Expression… button which will open the dialog box to that lets you build a query for selection

A basic plugin setup will look something like below for a new plugin called MyNewDialogPlugin



```
MYNEWDIALOGPLUGIN
  help
  i18n
  scripts
  test
  __init__.py
  icon.png
  Makefile
  metadata.txt
  MyNewDialogPlugin_dialog_base.ui
  MyNewDialogPlugin_dialog.py
  MyNewDialogPlugin.py
  pb_tool.cfg
  plugin_upload.py
  pylintrc
  README.html
  README.txt
  resources.py
  resources.qrc
```

Some of the files you are interested in:

Icon.png - this is the icon that will appear on the tool bar in QGIS

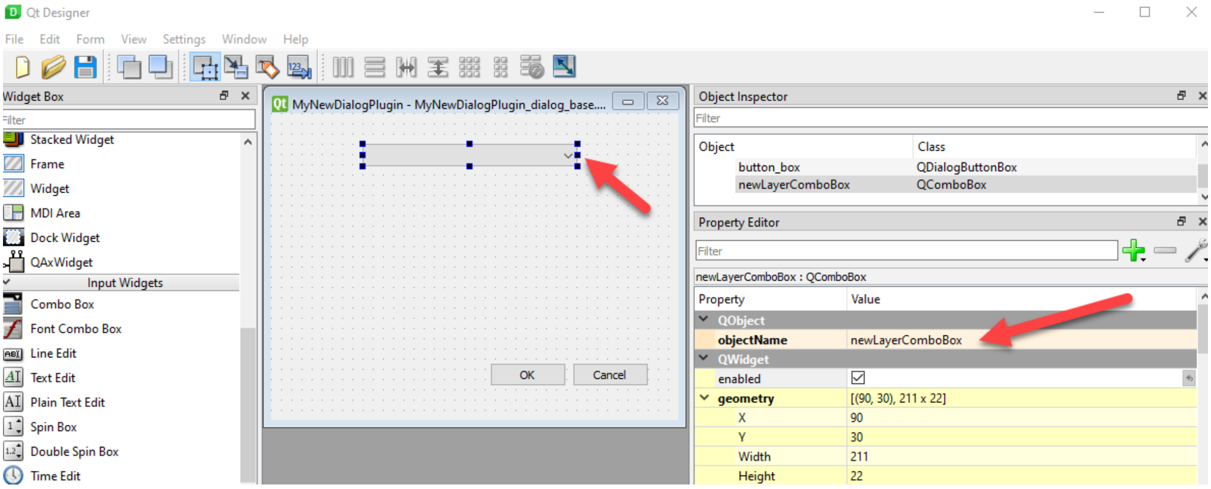
Metadata.txt - This holds information like the version number of your plugin, if you are going to release the plugin to others you will need to change this with each release, if the plugin is just for you then it doesn't matter too much

MyNewDialogPlugin_dialog_base_ui - this is the dialog or form itself. The design of this form is done within software called Qt Designer which is installed with QGIS (see Designing a Dialog using Qt Designer)

MyNewDialogPlugin_dialog.py - this is where the python code that is wired to the Dialog goes (e.g. when you click a button on the form this is where the code runs is located). This goes for any of the items on the dialog.

MyNewDialogPlugin.py - this is where the code goes that runs when the button is pressed. At its simplest it will simply be code to open the Dialog for the user.

Designing a Dialog using Qt Designer



Object	Class
button_box	QDialogButtonBox
newLayerComboBox	QComboBox

newLayerComboBox : QComboBox

Property	Value
QObject	
objectName	newLayerComboBox
QWidget	
enabled	☑
geometry	[(90, 30), 211 x 22]
X	90
Y	30
Width	211
Height	22

Use this designer to create the dialog that you want to use. The key part with this designer is the objectName that you give to each of the components. You will use this name to interact with the dialog in your python code.

Button with Dock Widget

Use a dock widget rather than a form to do all of your work. An example would be the Bookmark Manager for adding and removing spatial bookmarks.

Interactive Tool

A tool for interacting with the map canvas itself through clicking,, dragging, etc. by wiring these events to your button. Example being the Pan tool which is wired to different things on left click, right click, scroll wheel, etc.