# Code

Within the *byra* folder, five files set the layout and logic for the tool. The following sections detail the contents of each file.

## byra\_dialog\_base.ui

Editable in Qt Creator software (<https://www.qt.io/download> – get the open source version). Graphically sets the dialog layout (containers and controls). NOTE: the text fields are currently set to the test data paths on the development machine for ease of testing. Should be cleared out before release.

## byra.py

Core logic of tool. Functions with important functionality are:

* \_\_init\_\_  
  connects click buttons to callback functions
* selectFolder, selectFile  
  Utility functions used to set text field values to paths selected in file dialogs.
* run  
  All the real work. Creates a validator (see *byra\_validate.py*), runs input validation, then displays the message bar with validation results. If the validation passes, creates a preprocessor (see *byra\_preprocessor.py*). Depending on SEC output checkboxes, runs the SEC generators and creates a CSV checklist of results.

## byra\_dialog.py

Automatically generated file. The only customization is the accept function, which warns the user if input paths are invalid.

## byra\_validate.py

Runs input validation. See code for details. The runChecks function gets called externally (i.e. in *byra.py*) and in turn checks each input. As part of the validation process, it creates a list of all species in the ByRA ratings and a dictionary of gears of concern for each species. These data structures (species + gear) are used in both the validator and the preprocessor, as well as *byra.py* in CSV checklist generation.

## byra\_preprocessor.py

Uses the validator results to run the preprocessing. Three functions, each of which generates one of the SECs (e.g. intensitySEC). All called by *byra.py* (lines 340-360).

# Tips

* A useful QGIS plugin during development is *Plugin Reloader*, which allows you to reload a plugin after making edits without having to restart QGIS. It’s flagged as experimental, so check the Show Experimental checkbox in the plugin manager to find it.
* All the geoprocessing tools (clip, intersection, etc.) run with one command, processing.run, which takes the name of the tool and a dictionary of parameters. It’s not well documented (nothing related to python for QGIS 3 is yet, for that matter) but the following commands are helpful for looking up tool names and inputs:  
  # List all tool names  
  for alg in QgsApplication.processingRegistry().algorithms():   
   print("{}:{} --> {}".format(alg.provider().name(), alg.name(), alg.displayName()))  
  # Get tool help   
  processing.algorithmHelp("qgis:clip")  
  Also, if you run a tool manually you can see what the parameters were in the Log tab in the tool dialog after it runs.
* Debugging the plugin is possible without an IDE. At the top of the file you wish to debug, add:  
  from PyQt5.QtCore import pyqtRemoveInputHook   
  import pdb  
  Then add the following lines where you want to set the breakpoint:  
  pyqtRemoveInputHook()  
  pdb.set\_trace()  
  Once you’ve done that, launch QGIS from the command line. On a Mac in Terminal, that looks like /Applications/QGIS3.app/Contents/MacOS/QGIS. Run the tool and the python debugger will start on the command line.