Using Qt Designer UI files in Python

Qt Designer is a great tool for UI development. The GUI you create is saved as a “.ui” file, which is an XML type schema. This can then be translated into a Python object on the fly.

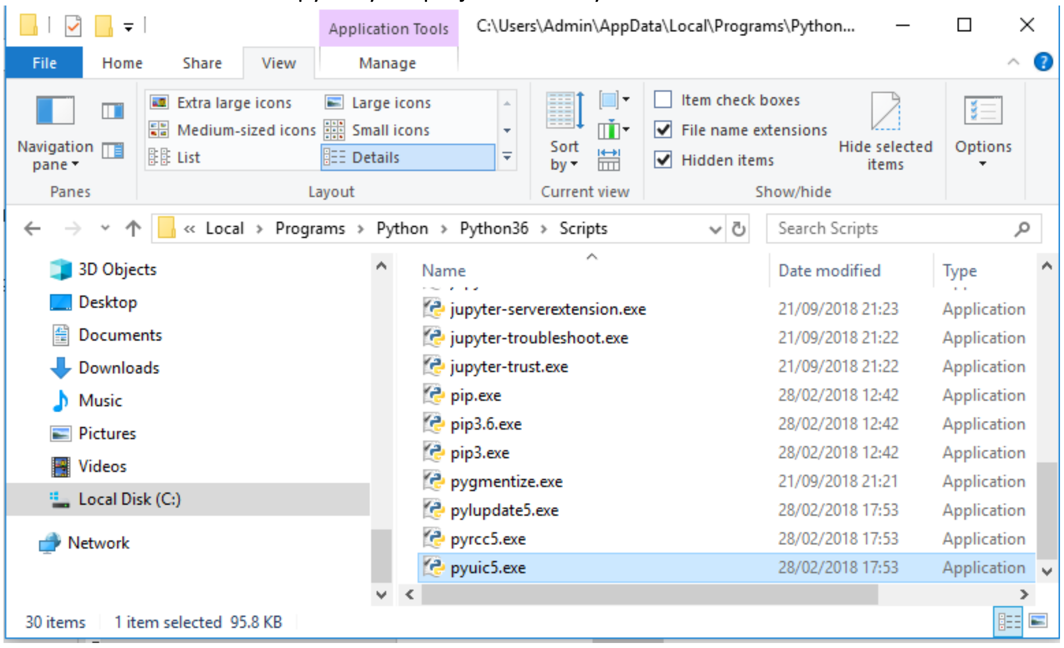
The best way to achieve this is to use a script to convert your GUI into a Python script, which allows you to access all of the objects in the IDE so you can tweak them as needed. This document outlines this process. You will need:

- the converter.py script in this directory
- the path to the uic application
- the path to the “.ui” file you created in QtDesigner

1. The text below is the code to enable the conversion of your GUI. First you import the os module. The next line creates a string where the first argument is the location of the pyuic5.exe file. You’ll need to find it in your computer and paste the path here, as in screenshot below. Put this path in between the quotes on the third line, remembering to add second “\”s if they are missing in the path. The next line is the path to your “.ui” file – add it between the quotes, again remembering the “\”s. The fifth line has the path to the destination – this is where your python version of the GUI file will be created. It is best to have this in your project, in the same directory as your main program. The last line can stay as is.

```
import os
conversion_str =
'C:\\Users\\admin\\AppData\\Local\\Programs\\Python\\Python36\\Scripts\\pyuic5.exe ' \
          '-x C:\\Users\\Admin\\PycharmProjects\\PROG9000\\bank_gui.ui ' \
          '-o C:\\Users\Admin\\PycharmProjects\\PROG9000\\gui.py'

os.system(conversion_str)
```

Save this file as “converter.py” in your project directory.



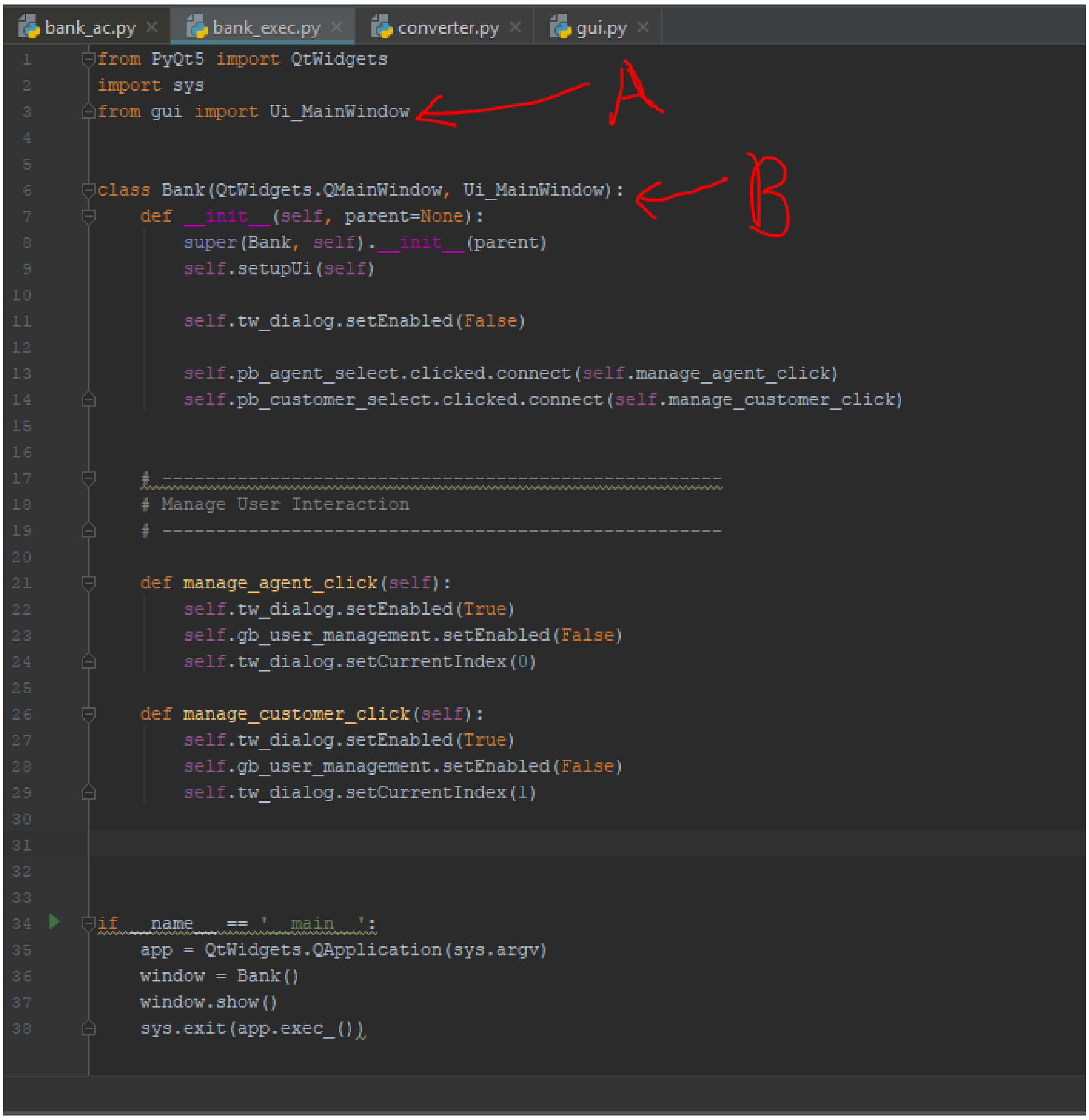
Name	Date modified	Type
jupyter-serverextension.exe	21/09/2018 21:23	Application
jupyter-troubleshoot.exe	21/09/2018 21:22	Application
jupyter-trust.exe	21/09/2018 21:22	Application
pip.exe	28/02/2018 12:42	Application
pip3.6.exe	28/02/2018 12:42	Application
pip3.exe	28/02/2018 12:42	Application
pygmentize.exe	21/09/2018 21:21	Application
pylupdate5.exe	28/02/2018 17:53	Application
pyrcc5.exe	28/02/2018 17:53	Application
pyuic5.exe	28/02/2018 17:53	Application

30 items 1 item selected 95.8 KB

2. When you run "converter.py", you will now have a Python version of the GUI. Open it and have a look. It may not make a lot of sense, but you will import this as the GUI object into your main application. The advantage of this method is that you will be able to directly access each widget and all the methods available while building your program.

3. The next step is to import the GUI object into your main application. In the screenshot below, A shows how I import the "Ui_MainWindow" class from the "gui" module (I named the converted GUI as "gui.py").

   Then, in B, when you create your GUI class for your application, you want to inherit from QtWidgets.QMainWindow and Ui_MainWindow.

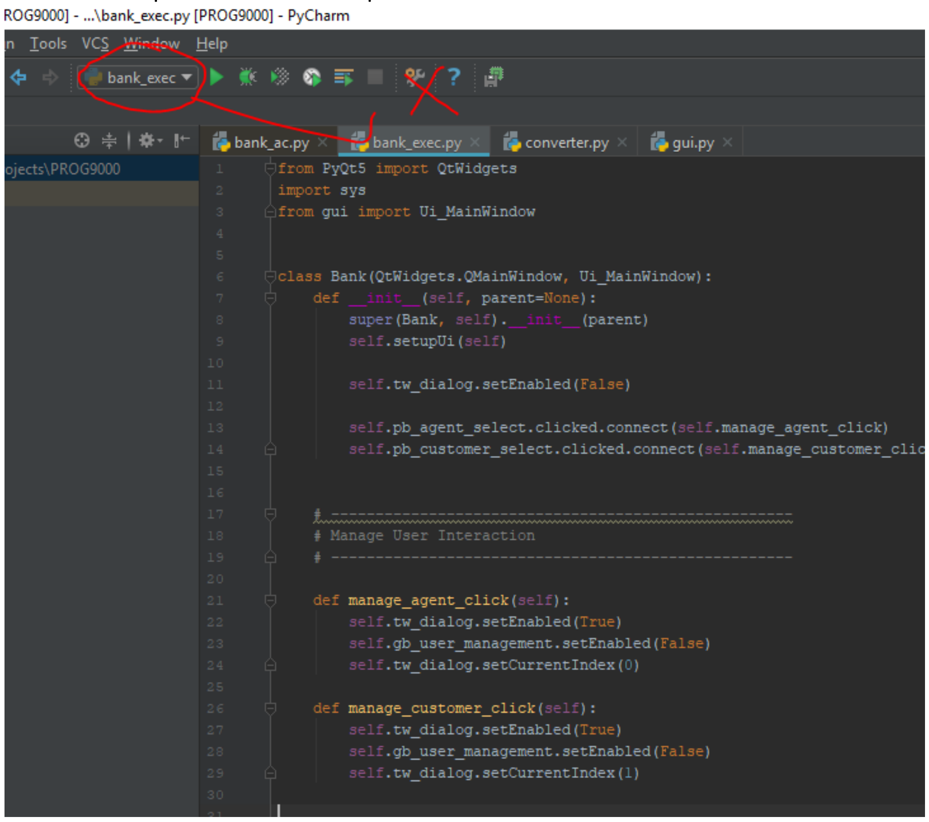


```python
from PyQt5 import QtWidgets
import sys
from gui import Ui_MainWindow


class Bank(QtWidgets.QMainWindow, Ui_MainWindow):
    def __init__(self, parent=None):
        super(Bank, self).__init__(parent)
        self.setupUi(self)

        self.tw_dialog.setEnabled(False)

        self.pb_agent_select.clicked.connect(self.manage_agent_click)
        self.pb_customer_select.clicked.connect(self.manage_customer_click)


    # ----------------------------------------------------
    # Manage User Interaction
    # ----------------------------------------------------

    def manage_agent_click(self):
        self.tw_dialog.setEnabled(True)
        self.gb_user_management.setEnabled(False)
        self.tw_dialog.setCurrentIndex(0)

    def manage_customer_click(self):
        self.tw_dialog.setEnabled(True)
        self.gb_user_management.setEnabled(False)
        self.tw_dialog.setCurrentIndex(1)



if __name__ == '__main__':
    app = QtWidgets.QApplication(sys.argv)
    window = Bank()
    window.show()
    sys.exit(app.exec_())
```

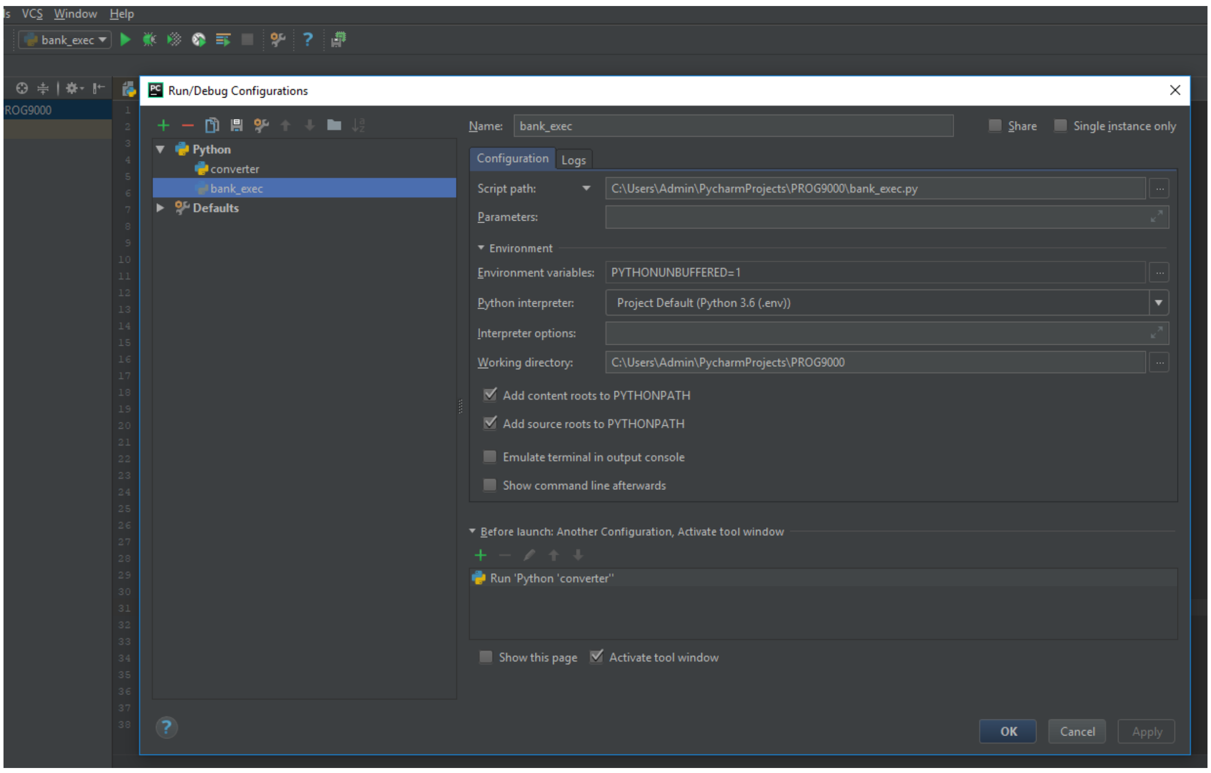
   This is slightly different from the original way we did it, but if you make the modifications as listed above, it should work without any problem.

4. In order to have the conversion script run automatically every time you run the main program (so that you can incorporate any updates you make to your GUI), there is an easy method. Open the

dropdown marked 'X' in the first screencap below (this allows access to the configuration of the running of your scripts). Click "Edit configurations" and pick the name of the script that is your main executable for the project, in my case "bank_exec". At the bottom of the dialog (shown in second screencap below), there is a window named "Before Launch". Here you can set any tasks you want to happen before you run this script. Click the "+" icon, pick "Run another configuration" and choose the converter script. It should show up in the window.



```python
from PyQt5 import QtWidgets
import sys
from gui import Ui_MainWindow


class Bank(QtWidgets.QMainWindow, Ui_MainWindow):
    def __init__(self, parent=None):
        super(Bank, self).__init__(parent)
        self.setupUi(self)

        self.tw_dialog.setEnabled(False)

        self.pb_agent_select.clicked.connect(self.manage_agent_click)
        self.pb_customer_select.clicked.connect(self.manage_customer_clic


    # --------------------------------------------------
    # Manage User Interaction
    # --------------------------------------------------

    def manage_agent_click(self):
        self.tw_dialog.setEnabled(True)
        self.gb_user_management.setEnabled(False)
        self.tw_dialog.setCurrentIndex(0)

    def manage_customer_click(self):
        self.tw_dialog.setEnabled(True)
        self.gb_user_management.setEnabled(False)
        self.tw_dialog.setCurrentIndex(1)
```



Name: bank_exec
Share [ ] Single instance only [ ]

Script path: C:\Users\Admin\PycharmProjects\PROG9000\bank_exec.py
Parameters:

Environment
Environment variables: PYTHONUNBUFFERED=1
Python interpreter: Project Default (Python 3.6 (.env))
Interpreter options:
Working directory: C:\Users\Admin\PycharmProjects\PROG9000

[✓] Add content roots to PYTHONPATH
[✓] Add source roots to PYTHONPATH
[ ] Emulate terminal in output console
[ ] Show command line afterwards

Before launch: Another Configuration, Activate tool window
Run 'Python 'converter''

[ ] Show this page [✓] Activate tool window