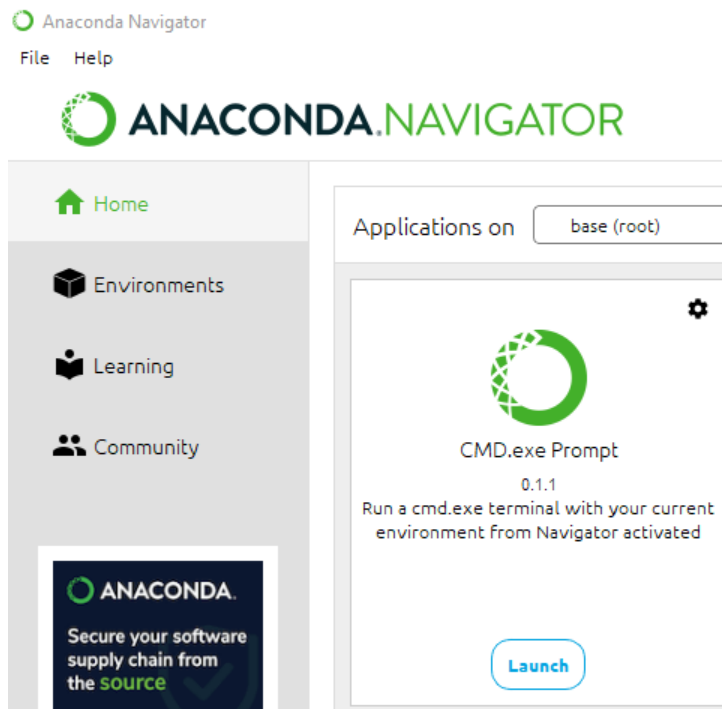


Chapter 7 – First Application (Python3)

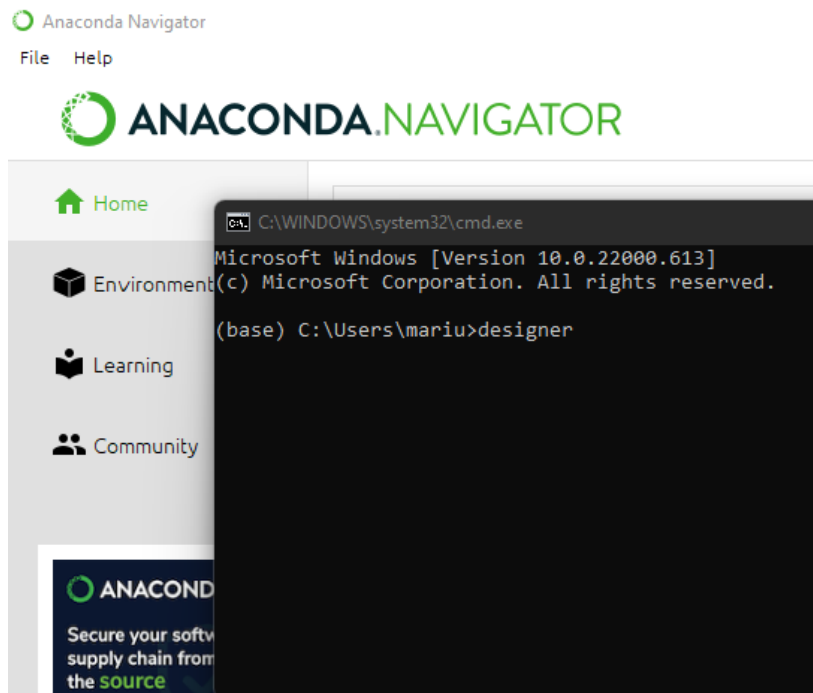
First Application in Qt Designer

Step 1 Open Qt Designer

- Launch the CMD.exe Prompt from within Anaconda Navigator

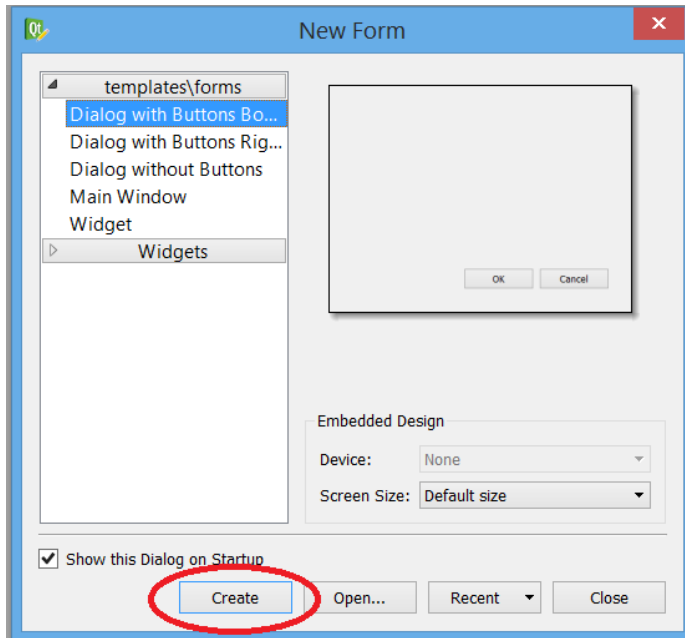


- In the command prompt type designer and press enter



Step 2 Selecting a template

- When opening Qt Designer, you are requested to select a template for your application. You can select any of these templates that would be applicable
- Click on the “Dialog with Buttons Bottom”
- Click the Create button



- A new form with the caption “untitled” is created with a button box that contains an OK and Cancel button.

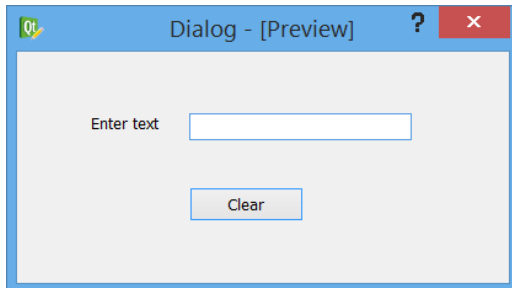
Step 3 Adding widgets

- The Buttons template automatically add buttons for OK and Cancel but to learn how to do it manually delete the buttons box.
 - Click on the OK or Cancel button to highlight the buttons box
 - Press the delete button
 - You should have an entirely blank form
- Add the following widgets and set the properties. For now we will leave the objectNames as the default: **Note that Python is case sensitive so make sure of the letter casing on objectnames**

Widget	Property	Value
QLabel (Display widgets section)	objectName	label
	text	Enter Text
QLineEdit (Input widgets section)	objectName	lineEdit


QPushButton (Buttons section)	objectName text	pushButton Clear
----------------------------------	--------------------	---------------------

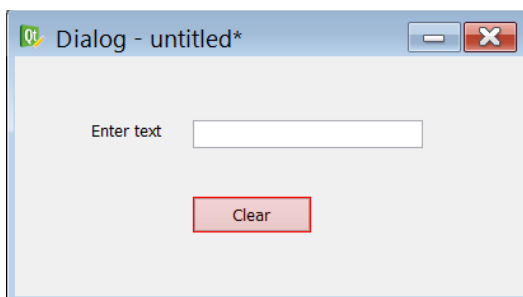
- Preview the form by pressing Ctrl+R
- Your form should look similar to the one below:



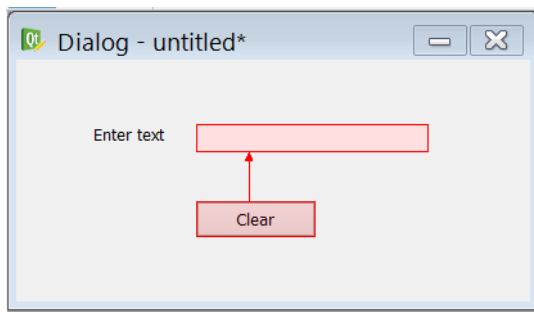
- Close the form by clicking the windows close button.

Step 4 Adding actions by connecting signals and slots

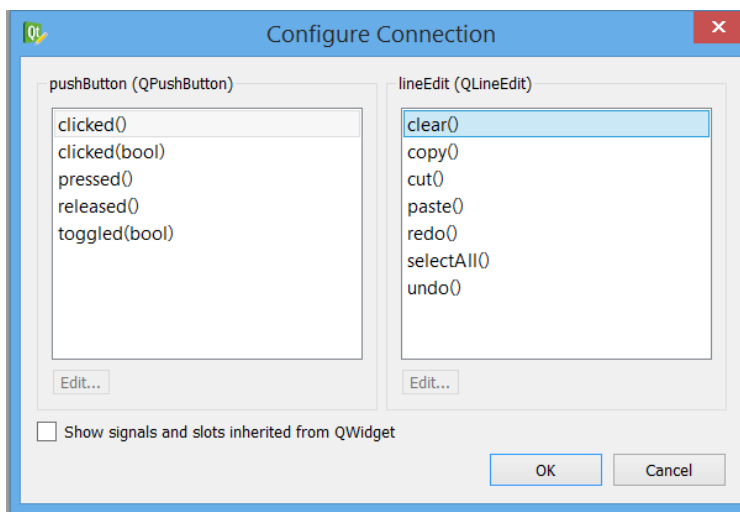
- We want to clear the text in the lineEdit widget when the user clicks on the Clear pushButton widget. To do this we link the pushButton's clicked() signal to the lineEdit's clear() slot.
- To explain signals and slots in Delphi terms, signals in Python are similar to events in Delphi for example the Onclick event in Delphi would correspond to the clicked() signal in Python. Slots in Python are similar to methods in Delphi for example the clear method in Delphi would correspond to the clear() slot of an object in Python.
- You are currently in widget editing mode and to apply signal/slot connections you need to be in signals and slots editing mode.
- Click the Edit Signals/Slots button on the toolbar menu 
- Move your cursor onto the Clear pushButton. It will turn red.



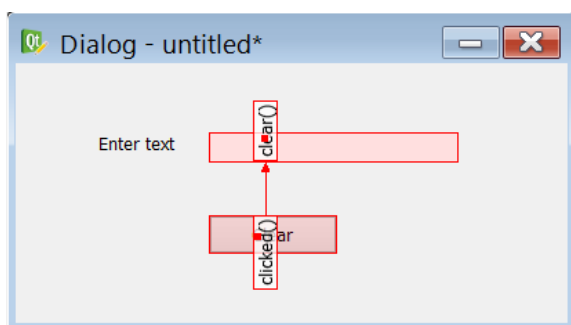
- Click and hold your mouse button, drag your cursor over to the lineEdit widget
- This indicates that a signal will be activated on the Clear pushButton that should trigger a slot on the lineEdit.



- The Configure Connection dialog is displayed
- To link the pushButton clicked() signal to the lineEdit clear() slot do the following:
- Under the pushButton column select the clicked() signal
- Under the lineEdit column select the clear() slot



- Click the OK button to confirm the selections and do the connection between the signal and slot

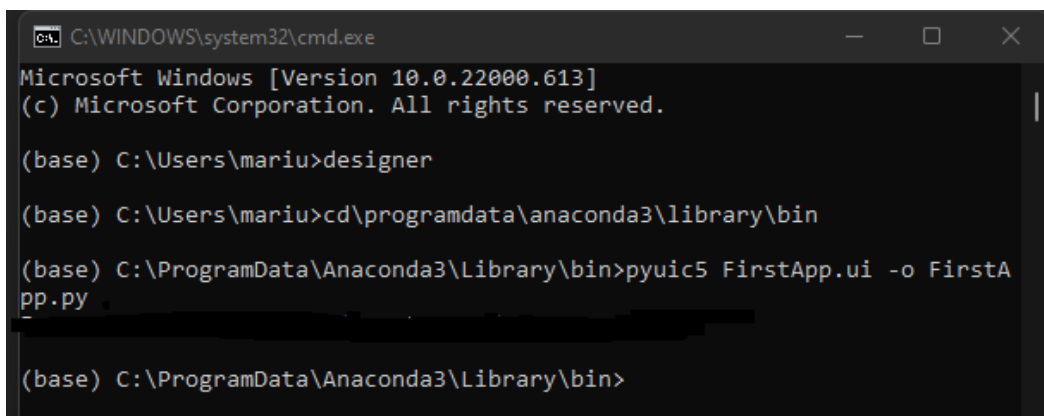


Step 5 Save the form

- Click File | Save
- Choose a location to save the form file to where you can easily find it so that we can copy it later to convert to a py file.
- Change the name to FirstApp.ui **(note the case!! Python is case sensitive)**

Step 6 Convert the .ui file to a .py file

- The .ui file created by Qt Designer is an xml file that cannot be used in Python
- The .ui file must be converted to a Python .py script file.
- Open file Explorer and copy the FirstApp.ui file you saved in step 5 to this folder:
C:\ProgramData\Anaconda3\Library\Bin
- We will use a command prompt utility for converting the .ui file to a Python script
- Open the Command prompt and change to this folder:
C:\ProgramData\Anaconda3\Library\Bin
- Type `pyuic5 FirstApp.ui -o FirstApp.py` and press Enter **(note the case!! Python is case sensitive, so even on file names the case must be the same throughout)**
- The screen will pause for a while and return a blank line if the conversion was successful



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.22000.613]
(c) Microsoft Corporation. All rights reserved.

(base) C:\Users\mariu>designer

(base) C:\Users\mariu>cd\programdata\anaconda3\library\bin

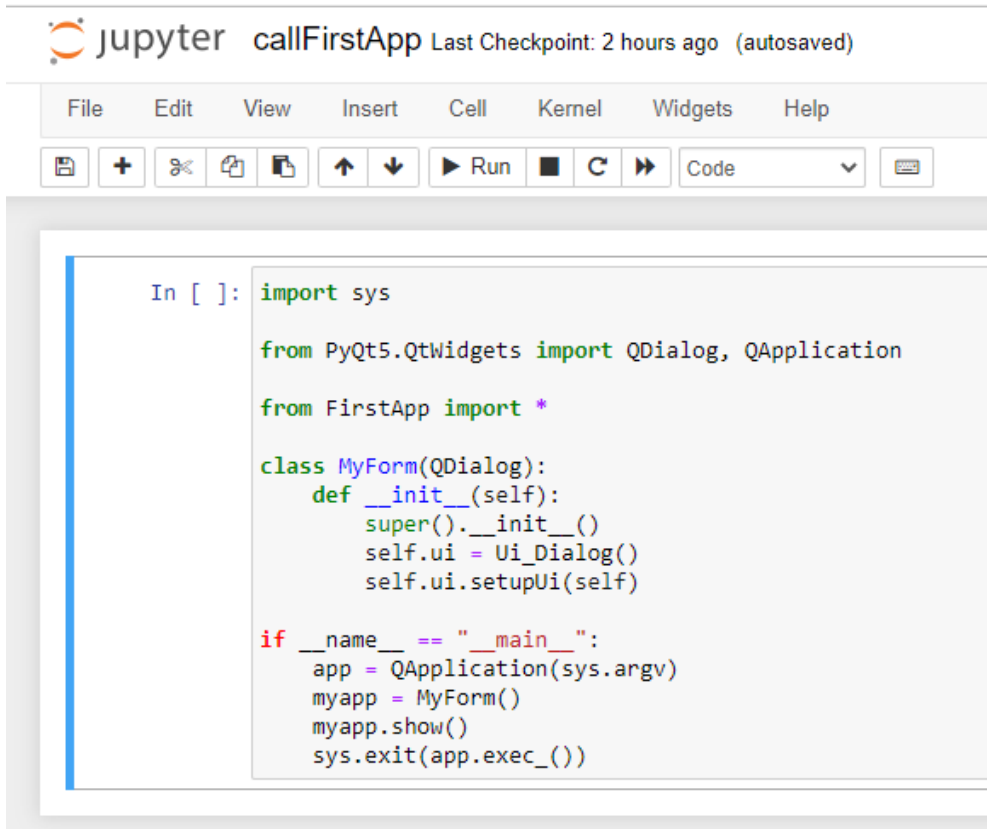
(base) C:\ProgramData\Anaconda3\Library\bin>pyuic5 FirstApp.ui -o FirstApp.py

(base) C:\ProgramData\Anaconda3\Library\bin>
```

- Copy the FirstApp.ui and FirstApp.py files from the
C:\ProgramData\Anaconda3\Library\Bin folder to a new folder for this application under your documents

Step 7 Create a source file (.pyw) that imports the .py file

- We will now create a Python source file that will import the .py file created in step 6 above and from which we will invoke the user interface
- Open the Jupyter Notebook and create a new Python3 file and add the following code **(note the indentation and case!!)**



The image shows a Jupyter Notebook interface. At the top, the Jupyter logo is followed by the text "callFirstApp" and "Last Checkpoint: 2 hours ago (autosaved)". Below this is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". Under the menu bar is a toolbar with icons for saving, adding cells, undo, redo, running, and other functions. The main area contains a code cell with the following Python code:

```
In [ ]: import sys

from PyQt5.QtWidgets import QDialog, QApplication

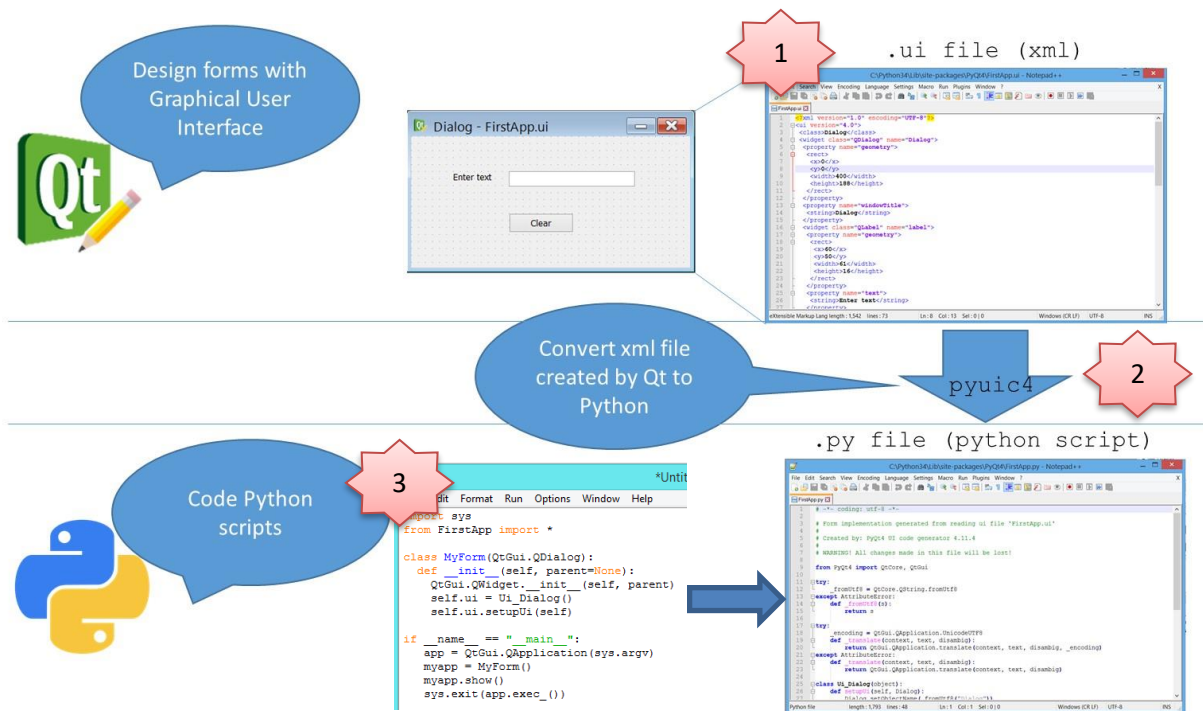
from FirstApp import *

class MyForm(QDialog):
    def __init__(self):
        super().__init__()
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)

if __name__ == "__main__":
    app = QApplication(sys.argv)
    myapp = MyForm()
    myapp.show()
    sys.exit(app.exec_())
```

- Save the file as callFirstApp
- Run the application by selecting Run button
- The dialogbox you created should appear.
- Test your application

Below is a graphical representation and description of the process we followed:



1. We created the GUI with Qt Designer, which produced an XML file (.ui)
2. The XML file cannot be interpreted by Python so we used pyuic4 to convert it to a python script (.py)
3. We created a source file Python script that imported the .py file that was created from the .ui file to show the form.