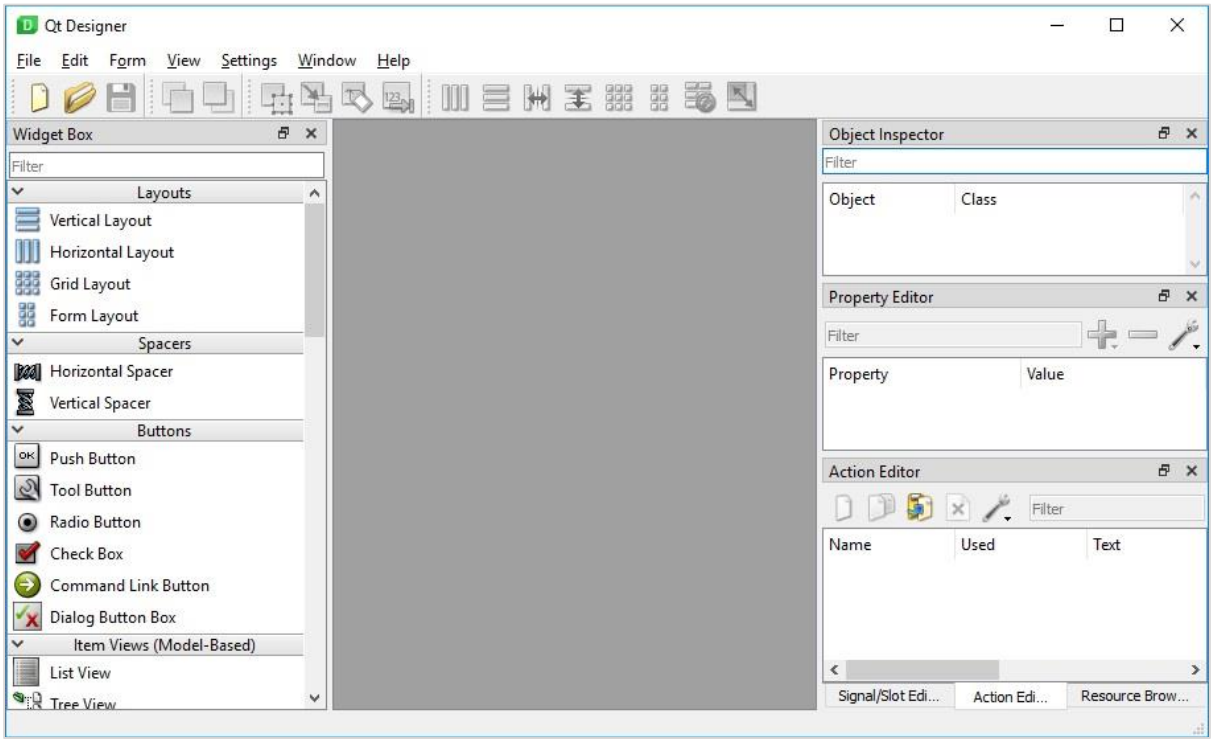
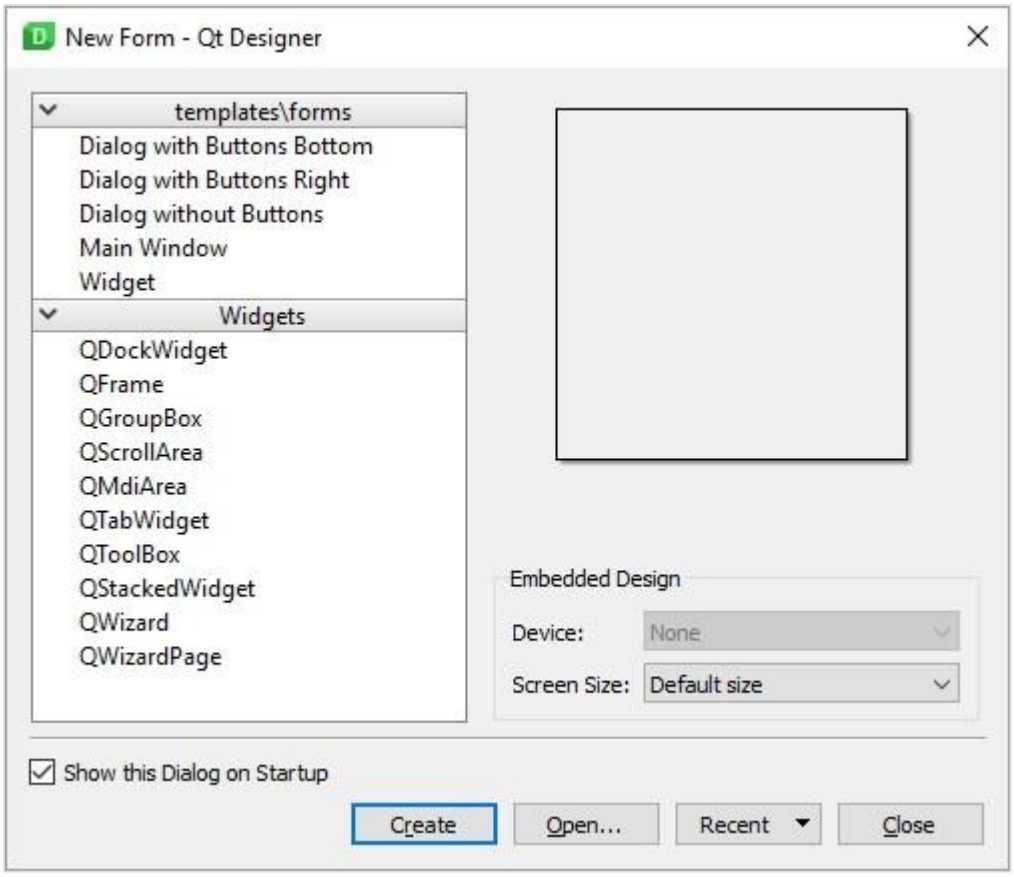
5. PyQt5 — Using Qt Designer

The PyQt installer comes with a GUI builder tool called **Qt Designer**. Using its simple drag and drop interface, a GUI interface can be quickly built without having to write the code. It is however, not an IDE such as Visual Studio. Hence, Qt Designer does not have the facility to debug and build the application.

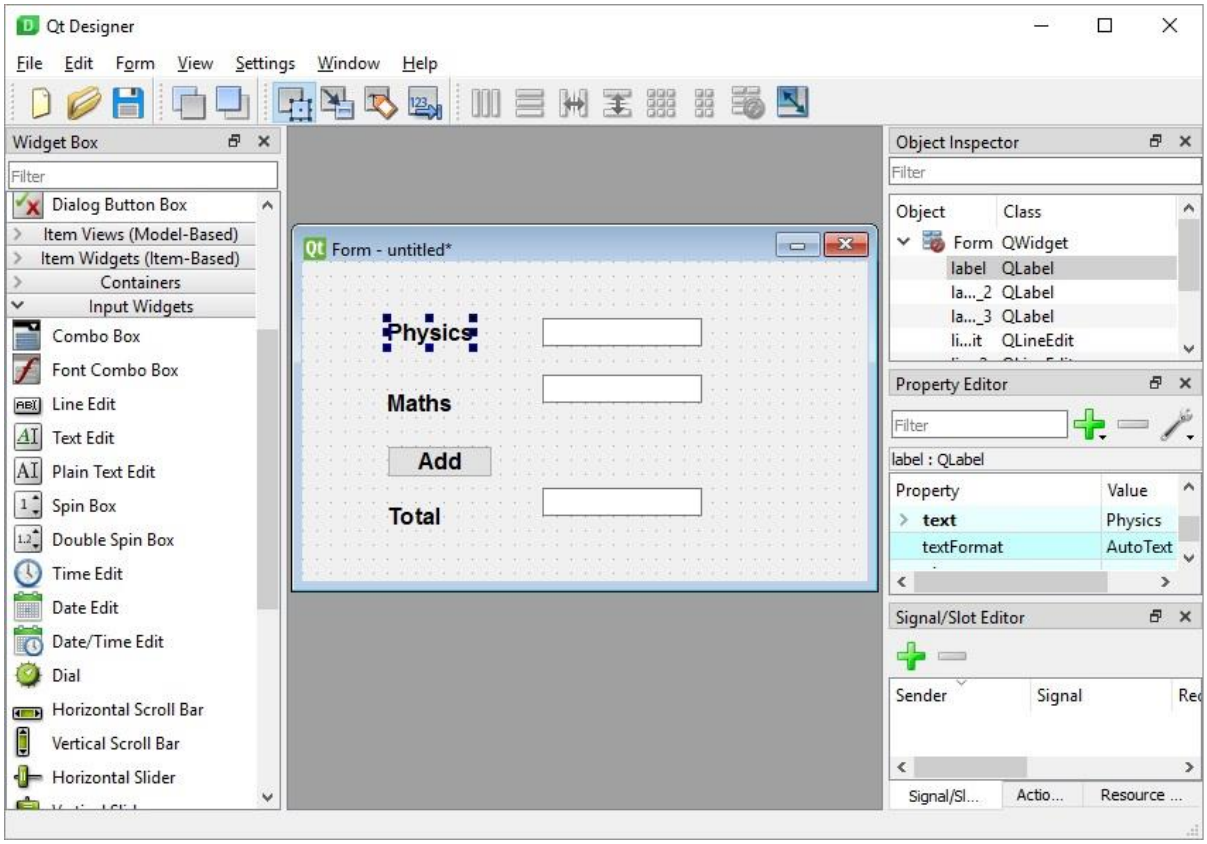
Start Qt Designer application which is a part of development tools and installed in scripts folder of the virtual environment.



Start designing GUI interface by chossing File -> New menu.



You can then drag and drop required widgets from the widget box on the left pane. You can also assign value to properties of widget laid on the form.



The designed form is saved as demo.ui. This ui file contains XML representation of widgets and their properties in the design. This design is translated into Python equivalent by using pyuic5 command line utility. This utility is a wrapper for uic module of Qt toolkit. The usage of pyuic5 is as follows:

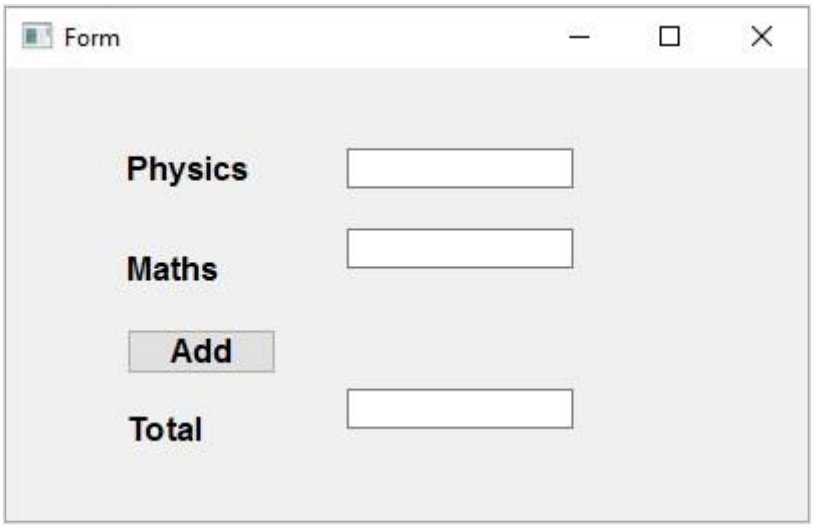
```
pyuic5 -x demo.ui -o demo.py
```

In the above command, -x switch adds a small amount of additional code to the generated Python script (from XML) so that it becomes a self-executable standalone application.

```
if __name__ == "__main__":
   import sys
   app = QtGui.QApplication(sys.argv)
   Dialog = QtGui.QDialog()
   ui = Ui_Dialog()
   ui.setupUi(Dialog)
   Dialog.show()
   sys.exit(app.exec_())
```

The resultant python script is executed to show the following dialog box:

```
python demo.py
```



The user can input data in input fields but clicking on Add button will not generate any action as it is not associated with any function. Reacting to user-generated response is called as **event handling**.

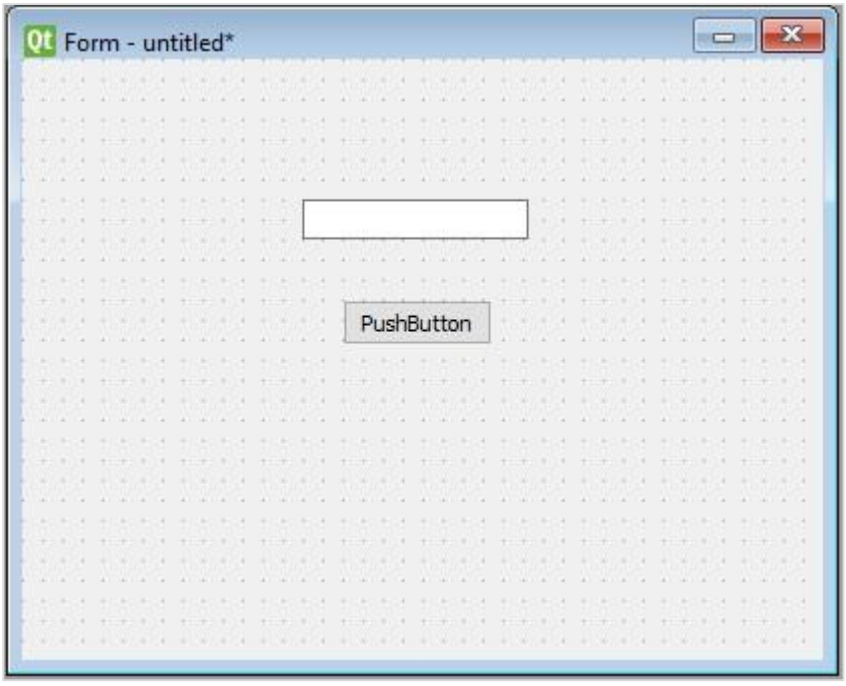
6. PyQt5 — Signals & Slots

Unlike a console mode application, which is executed in a sequential manner, a GUI based application is event driven. Functions or methods are executed in response to user's actions like clicking on a button, selecting an item from a collection or a mouse click etc., called **events**.

Widgets used to build the GUI interface act as the source of such events. Each PyQt widget, which is derived from QObject class, is designed to emit '**signal**' in response to one or more events. The signal on its own does not perform any action. Instead, it is 'connected' to a '**slot**'. The slot can be any **callable Python function**.

Using Qt Designer's Signal/Slot Editor

First design a simple form with a LineEdit control and a PushButton.



It is desired that if button is pressed, contents of text box should be erased. The QLineEdit widget has a clear() method for this purpose. Hence, the button's **clicked** signal is to be connected to **clear()** method of the text box.

To start with, choose Edit signals/slots from Edit menu (or press F4). Then highlight the button with mouse and drag the cursor towards the textbox