

Software Engineering Lab #3

GUI Programming with Qt for Python and Qt Designer

Background

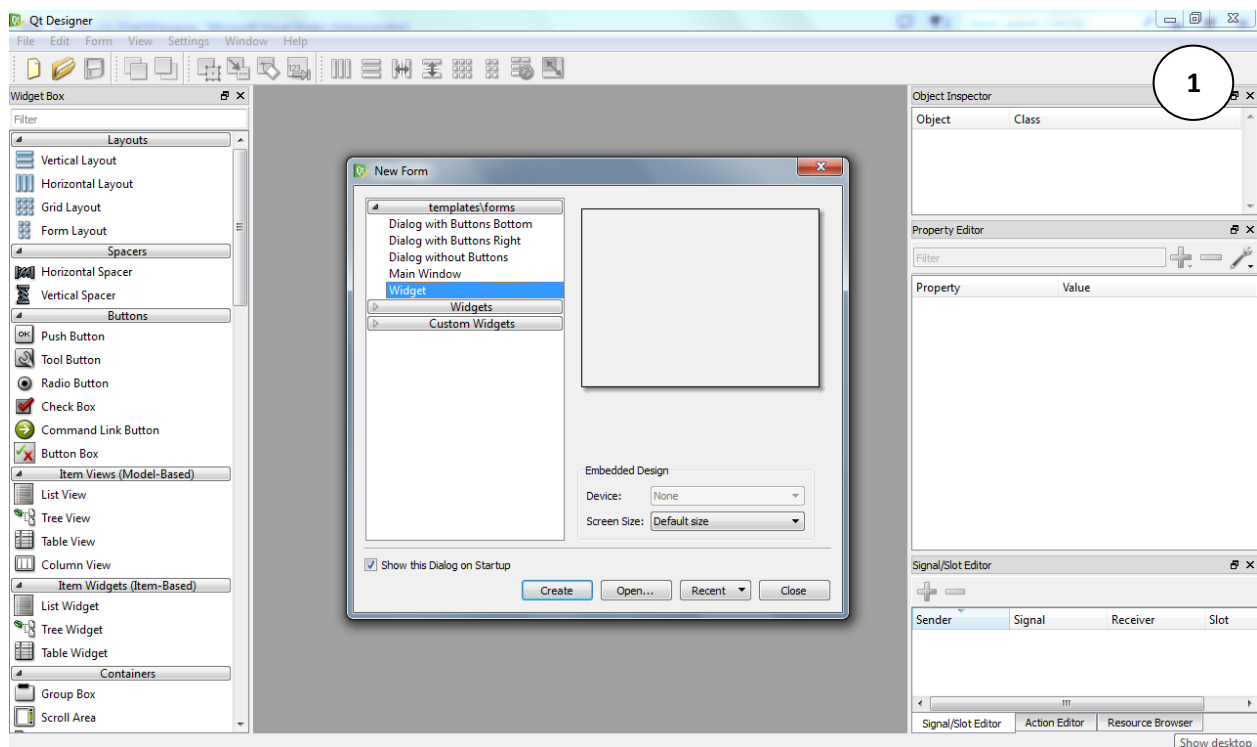
Qt Designer is a GUI design tool for creating a Qt widget. It is a part of Qt SDK which can be downloaded from <https://pypi.org/project/PySide6>. Qt Designer is also a part of Qt for Python.

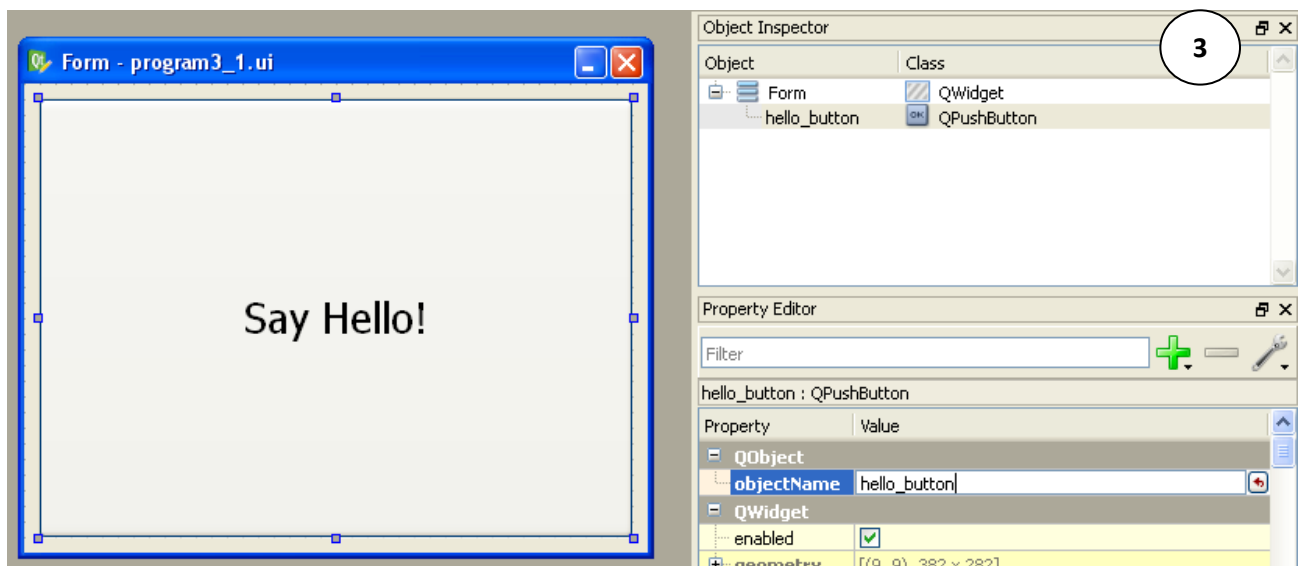
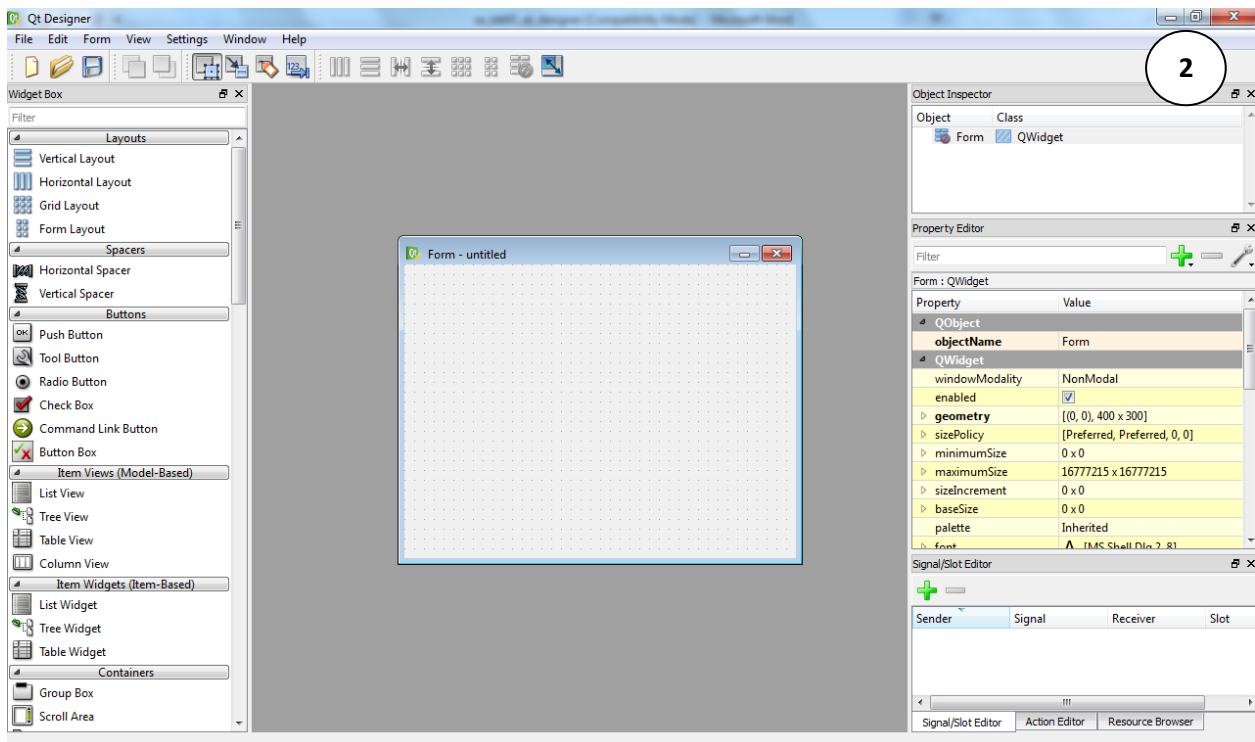
Before you start designing a GUI widget, you must first launch Qt Designer. Qt Designer is a part of PySide6 and it is located in directory `<python-installation-dir>\Lib\site-packages\PySide6`. You can launch **designer.exe** to start Qt Designer.

Alternatively, if you have installed Qt SDK, you will get Qt Designer as well which is a part of Qt SDK (you can always launch it from the start menu).

Program 3.1: Your First Qt Designer Form

1. Create the following UI from Qt Designer and save it as “program3_1.ui”





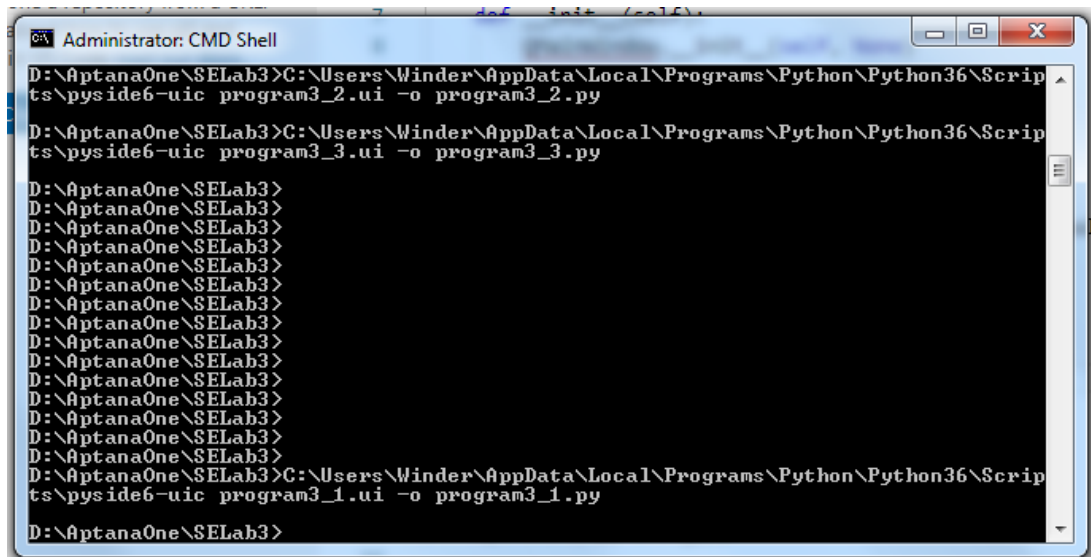
Instructions:

- Qt Designer starts with a New Form dialog. In this example, we will create a form from a widget form template which is the simplest one. To do so, select **“Widget”** type from the template then click **Create**.
- Drag a **“Push Button”** into the form. There is no need to place it to exactly where you want to place it, as you can use the layout to adjust the precise position of this button later.
- To apply a layout to the form, right click on the form and select **“Lay out”** then select one of the Lay Out options. In this example, you can use **“Lay Out Horizontally”** or **“Lay Out Vertically”**. You can also try out other options and see what will happen.
- If you want to change the layout, you can apply another layout to the form. If you want to remove layout, you can use **“Break Layout”**.
- To change the appearance of the form or the button, use the **“Property Editor”**. With this you can edit the style sheet or change the font of the text in the widget.

- You may need to set the name of the widget in the “**objectName**” field in the QObject section in the Property Editor. In this example, you must set the name of the button to be “**hello_button**” as this will be referred later in the source code of the example program.
- You can see a preview of what the form will look like when it is displayed on the screen. To see a preview of the form, select Form -> Preview... from the Qt Designer menu.

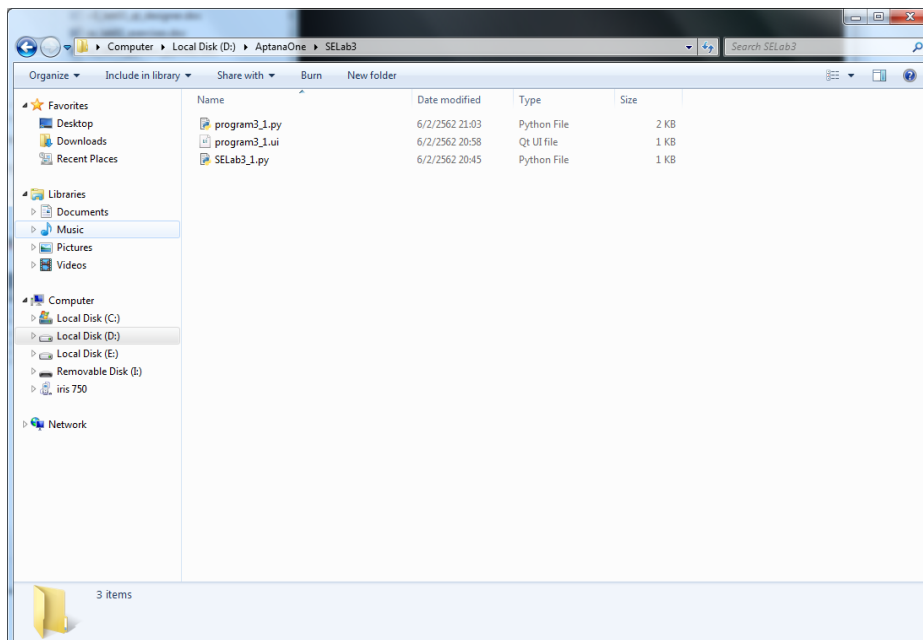
2. Compile .ui file into .py file with pyuic6 by using command

<python-installation-dir>\Scripts\pyuic6 input.ui -o output.py.



```
Administrator: CMD Shell
D:\AptanaOne\SELab3>C:\Users\Winder\AppData\Local\Programs\Python\Python36\Scripts\pyuic6 program3_2.ui -o program3_2.py
D:\AptanaOne\SELab3>C:\Users\Winder\AppData\Local\Programs\Python\Python36\Scripts\pyuic6 program3_3.ui -o program3_3.py
D:\AptanaOne\SELab3>
D:\AptanaOne\SELab3>
D:\AptanaOne\SELab3>
D:\AptanaOne\SELab3>
D:\AptanaOne\SELab3>
D:\AptanaOne\SELab3>
D:\AptanaOne\SELab3>
D:\AptanaOne\SELab3>
D:\AptanaOne\SELab3>
D:\AptanaOne\SELab3>
D:\AptanaOne\SELab3>
D:\AptanaOne\SELab3>
D:\AptanaOne\SELab3>
D:\AptanaOne\SELab3>
D:\AptanaOne\SELab3>
D:\AptanaOne\SELab3>
D:\AptanaOne\SELab3>
D:\AptanaOne\SELab3>C:\Users\Winder\AppData\Local\Programs\Python\Python36\Scripts\pyuic6 program3_1.ui -o program3_1.py
D:\AptanaOne\SELab3>
```

After running the command, the output .py file will be created in the same directory as that of the .ui file.



3. Write the following source code in Visual Studio and make sure that it is located in the same directory as that of the .py file above.

Program 3.1: Hello GUI

```
1  import sys
2  from PySide6.QtWidgets import *
3  from PySide6.QtCore import *
4  from program3_1 import Ui_Form
5
6  def say_hello():
7      print("Hello!")
8
9  if __name__ == '__main__':
10     app = QApplication(sys.argv)
11     Form = QWidget()
12     ui = Ui_Form()
13     ui.setupUi(Form)
14     ui.hello_button.clicked.connect(say_hello)
15     Form.show()
16
17     sys.exit(app.exec_())
```

How this program works:

- First, you need to import the `Ui_Form` from the module created by `pyside2-uic` command. For this Example `from program3_1 import Ui_Form`, The class `Ui_form` is the class generated by the command as the representation of .ui file in a .py format. This class contains a method name `setupUi` which is necessary to form an interface object as when designing with Qt designer.

```
Form = QWidget()
ui = Ui_Form()
ui.setupUi(Form)
```

- After creating the main window, we need to connect the action with the clicked signal of the button in the form. We can connect the clicked signal to an action:

```
ui.hello_button.clicked.connect(say_hello)
```

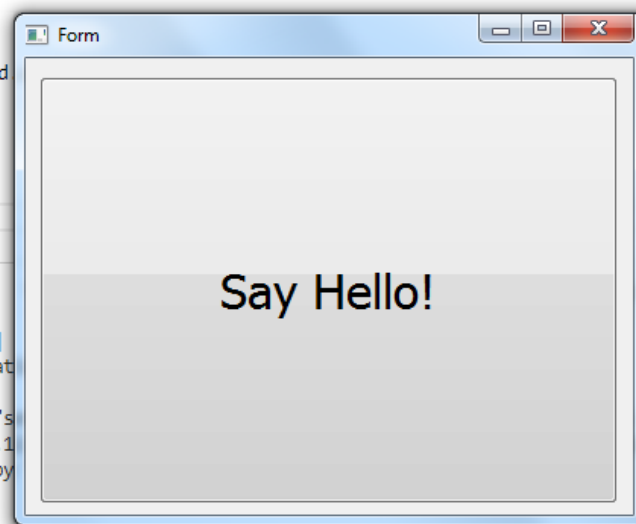
This statement connects the `clicked` signal of the `hello_button` to the function named `say_hello`.

```
11  Form = QWidget()
12  ui = Ui_Form()
13  ui.setupUi(Form)
14  ui.hello_button.clicked
15  Form.show()
16
17  sys.exit(app.exec_())
18
19
```

PROBLEMS 8 OUTPUT DEBUG CONSOLE

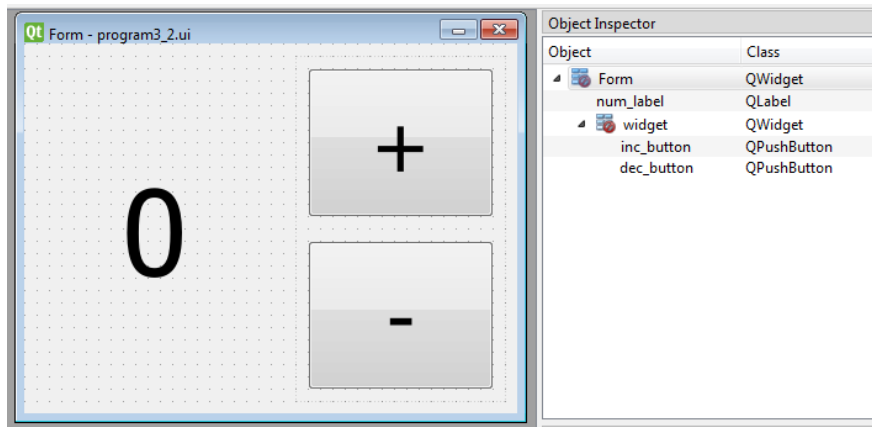
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation

```
D:\testWeb>cd d:\testWeb && cmd /C "s
de\extensions\ms-python.python-2018.1
13688 d:\AptanaOne\SELab3\SELab3_1.py
Hello!
Hello!
```



Program 3.2: Spin a number up and down

1. Create the following UI using Qt Designer again and save it as “program3_2.ui”



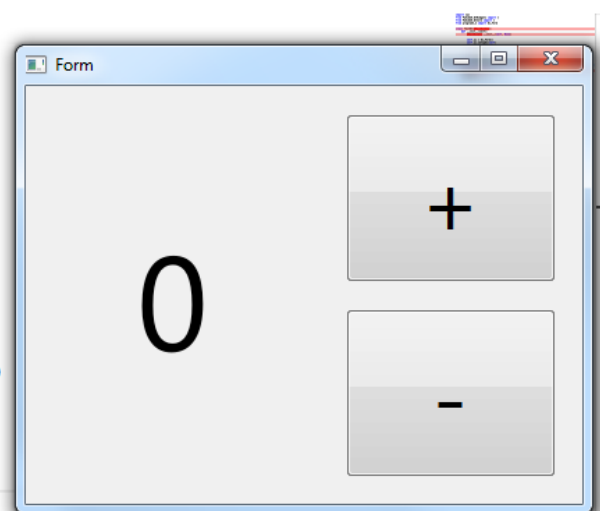
Instructions:

- Starts Qt Designer, select “**Widget**” type from the template then click **Create**.
- Drag a label and a widget onto the form. Apply a horizontal layout to the form.
- Drag two push buttons onto the widget that is just added to the form. This will make this widget be a parent of both buttons. Apply a vertical layout to this widget.
- Adjust the location and the size of the label and the buttons in the form until you get the same layout as the form shown in the figure above.
- Set the name of the label to be “**num_label**”. Then, set the name of the two buttons to be “**inc_button**” and “**dec_button**” respectively.

2. Compile .ui file into .py file with pyuic6

3. Write the following source code in Visual Studio Code and make sure that it is located in **the same directory as that of the .py file above**.

```
1  import sys
2  from PySide6.QtWidgets import *
3  from PySide6.QtCore import *
4  from program3_2 import Ui_Form
5
6  class TestUI(QMainWindow):
7      def __init__(self):
8          QMainWindow.__init__(self, None)
9
10         self.ui = Ui_Form()
11         self.ui.setupUi(self)
12
13         self.num = 0
14         self.ui.inc_button.clicked.connect(self.inc_value)
15
16     def inc_value(self):
17         self.num += 1
18         self.ui.num_label.setText(str(self.num))
19
20 if __name__ == '__main__':
21     app = QApplication(sys.argv)
22     w = TestUI()
23     w.show()
24     sys.exit(app.exec_())
```



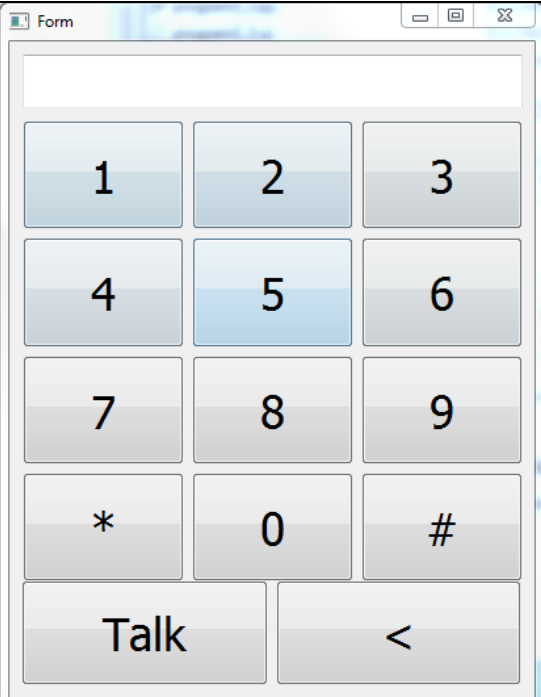
How this program works:

- In this program, you create a subclass of `QMainWindow` namely `My_window`. A `My_window` object will be used as the main window.
- In `__init__` method of `My_window`, the form is loaded with `self` (of class `My_window`) as a parent.
- After the form is loaded, `self.ui.setupUi(self)` is used to set the form as the central widget.
- After setting the central widget, obtain the references to the label and the buttons and then connect the clicked signal to an action.

How to use Qt Creator to design Graphical Interfaces for PyQt (in XML format)

This section describes how to use Qt Creator to create a Graphical Interface for your Qt for Python program. You will need Qt Creator and PySide6-Tools (pyuic and pyrcc).

1. with Qt Creator, create a new Qt Design Form by choosing “Main Window” for template. Then save as “program3.3.ui” . Add this to the center of the centralwidget. You should get the outcome like this:

UI format	XML format
	<pre>1 <?xml version="1.0" encoding="UTF-8"?> 2 <ui version="4.0"> 3 <class>Form</class> 4 <widget class="QWidget" name="Form"> 5 <property name="geometry"> 6 <rect> 7 <x>0</x> 8 <y>0</y> 9 <width>400</width> 10 <height>500</height> 11 </rect> 12 </property> 13 <property name="windowTitle"> 14 <string>Form</string> 15 </property> 16 <widget class="QLineEdit" name="lineEdit"> 17 <property name="geometry"> 18 <rect> 19 <x>10</x> 20 <y>10</y> 21 <width>381</width> 22 <height>41</height> 23 </rect> 24 </property> 25 </widget> 26 <widget class="QWidget" name="gridLayoutWidget"> 27 <property name="geometry"></pre>

2. You can create a new Python module by pyside6-uic with this XML file using the command **pyside6-uic input.ui -o output.py** . Running this command you will obtain the python module as shown below.

```
# -*- coding: utf-8 -*-

#####
## Form generated from reading UI file 'program3_3.ui'
##
## Created by: Qt User Interface Compiler version 6.0.0
##
## WARNING! All changes made in this file will be lost when recompiling UI file!
#####

from PySide6.QtCore import *
from PySide6.QtGui import *
from PySide6.QtWidgets import *

class Ui_Form(object):
    def setupUi(self, Form):
        if not Form.setObjectName():
            Form.setObjectName(u"Form")
        Form.resize(400, 500)
        self.lineEdit = QLineEdit(Form)
        self.lineEdit.setObjectName(u"lineEdit")
        self.lineEdit.setGeometry(QRect(10, 10, 381, 41))
        self.gridLayoutWidget = QWidget(Form)
        self.gridLayoutWidget.setObjectName(u"gridLayoutWidget")
        self.gridLayoutWidget.setGeometry(QRect(10, 60, 381, 351))
        self.gridLayout = QGridLayout(self.gridLayoutWidget)
        self.gridLayout.setObjectName(u"gridLayout")
        self.gridLayout.setContentsMargins(0, 0, 0, 0)
        self.button_star = QPushButton(self.gridLayoutWidget)
        self.button_star.setObjectName(u"button_star")
        sizePolicy = QSizePolicy(QSizePolicy.Minimum, QSizePolicy.Expanding)
        sizePolicy.setHorizontalStretch(0)
        sizePolicy.setVerticalStretch(0)
        sizePolicy.setHeightForWidth(self.button_star.sizePolicy().hasHeightForWidth())
        self.button_star.setSizePolicy(sizePolicy)
        font = QFont()
        font.setPointSize(26)
        self.button_star.setFont(font)
```
