2019

PyQt5 Tutorial

A SAMPLE APPLICATION

LILIAN SAO DE RIVERA

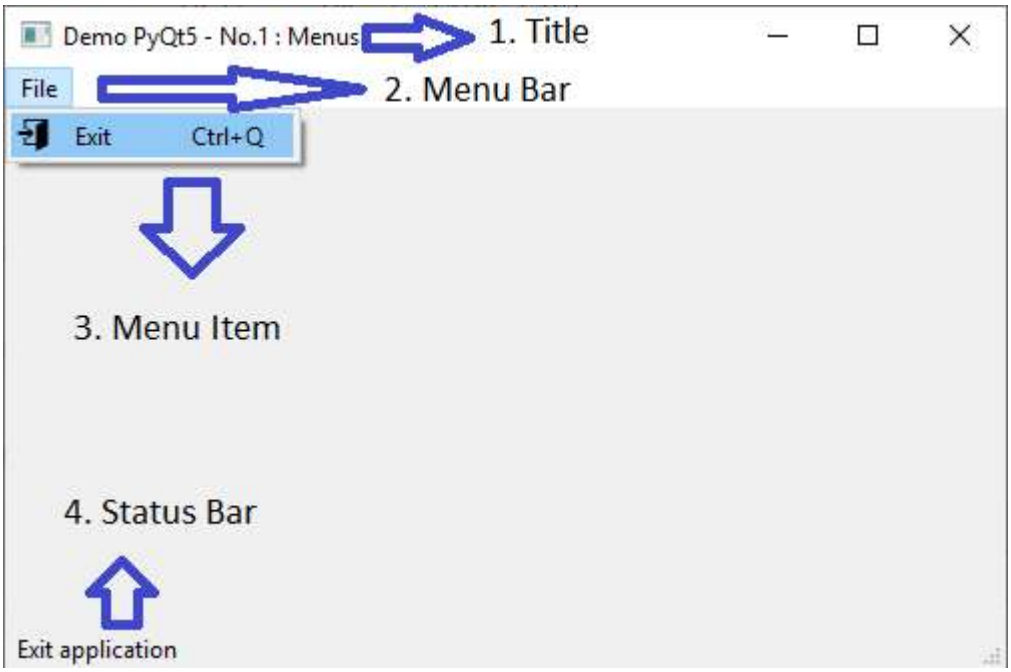
Contents

Basic PyQt5 Application

No.1 Create a Menu and a Menu Item.

The figure below shows a basic window in PyQt5 that shows a Menu bar with one option, a Menu Item with four characteristics, an icon (door), a label (“Exit”) a short key (“Ctrl-Q”) and as short status bar description(“Exit application”). Each time the user hovers over the item menu the status bar shows the description of the action to be taken.

Fig. 1 Basic PyQt5 Menu Window



The following is the code that produces this application. All the explanations for the code are in the form of comments.

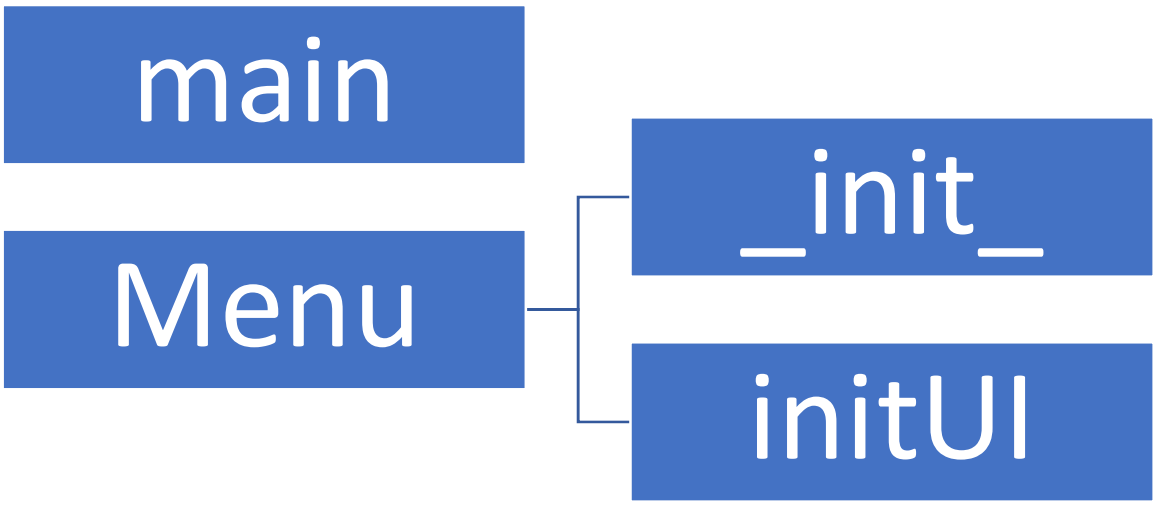
```
#::-----------------------------------------------------------------
#:: To create a Menu with options this are the libraries and components that
#:: requiered. For each new option we will be o adding new components
#::-----------------------------------------------------------------
import sys
from PyQt5.QtWidgets import QMainWindow, QAction, QMenu, QApplication
from PyQt5.QtGui import QIcon
```

```
#::------------------------------------------------------------
#:: Definition of a Class for the main manu in the application
#::------------------------------------------------------------
class Menu(QMainWindow):

    def __init__(self):

        super().__init__()
        #::-----------------------
        #:: variables use to set the size of the window that contains the menu
        #::-----------------------
        self.left = 100
        self.top = 100
        self.width = 500
        self.height = 300

        #:: Title for the application

        self.Title = 'Demo PyQt5 - No.1 : Menus'

        #:: The initUi is call to create all the necessary elements for the menu

        self.initUI()

    def initUI(self):

        #::-------------------------------------------------
        # Creates the manu and the items
        #::-------------------------------------------------
        self.setWindowTitle(self.Title)
        self.setGeometry(self.left, self.top, self.width, self.height)
        self.statusBar()
        #::-----------------------------
        # 1. Create the menu bar
        # 2. Create an item in the menu bar
        # 3. Creaate an action to be executed when the option
        #    in the  menu bar is choosen
        #::-----------------------------
        mainMenu = self.menuBar()
        fileMenu = mainMenu.addMenu('File')

        #::--------------------------------------
        # Exit action
        # The following code creates the the da Exit Action along
        # with all the characteristics associated with the action
        # The Icon, a shortcut , the status tip that would appear in the window
        # and the action
        #  triggered.connect will indicate what is to be done when the item in
        # the menu is selected
        # These definitions are not available until the button is assigned
        # to the menu
        #::--------------------------------------

        exitButton = QAction(QIcon('enter.png'), '&Exit', self)
        exitButton.setShortcut('Ctrl+Q')
        exitButton.setStatusTip('Exit application')
        exitButton.triggered.connect(self.close)
```

```
        #:: This line adds the button (item element ) to the menu

        fileMenu.addAction(exitButton)

        #:: This line shows the windows

        self.show()

#::------------------------
#:: Application starts here
#::------------------------
def main():
    app = QApplication(sys.argv)  # creates the PyQt5 application
    mn = Menu()  # Cretes the menu
    sys.exit(app.exec_())  # Close the application

if __name__ == '__main__':
    main()
```

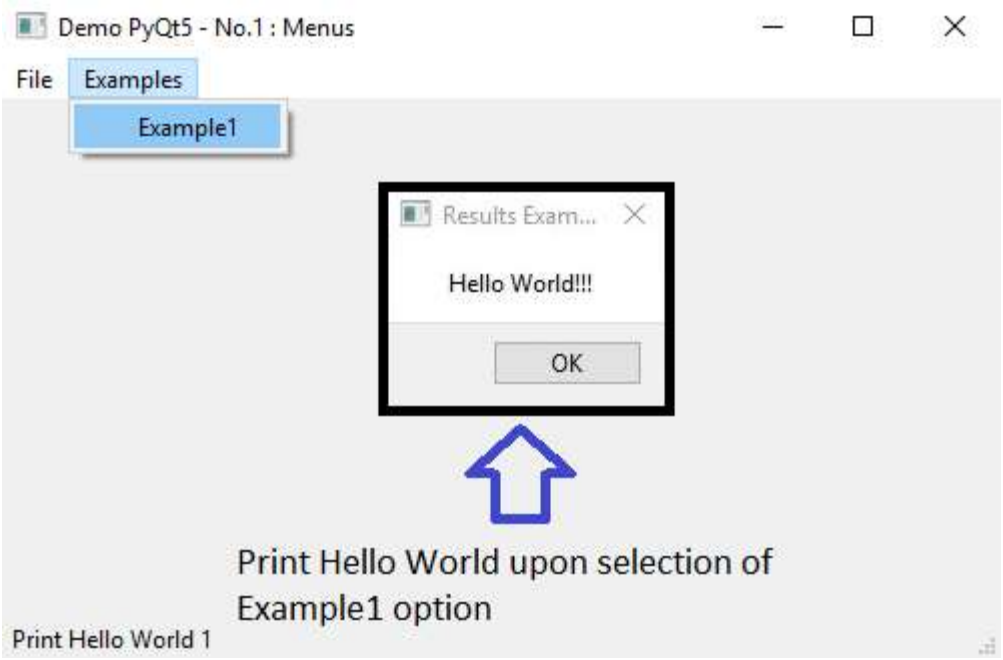
The figure below describes the different elements in the code. The purpose of the figure is to illustrate the iterations and relation amongst the objects.

No.1 Create a Menu and a Menu Item.



No.2 Create a second option and print a message box upon request

We are going to use the previous code to add an extra option to the main menu. The purpose of this section is to show how to add a messagebox with the words “Hello World !!!” upon the selection of an option.



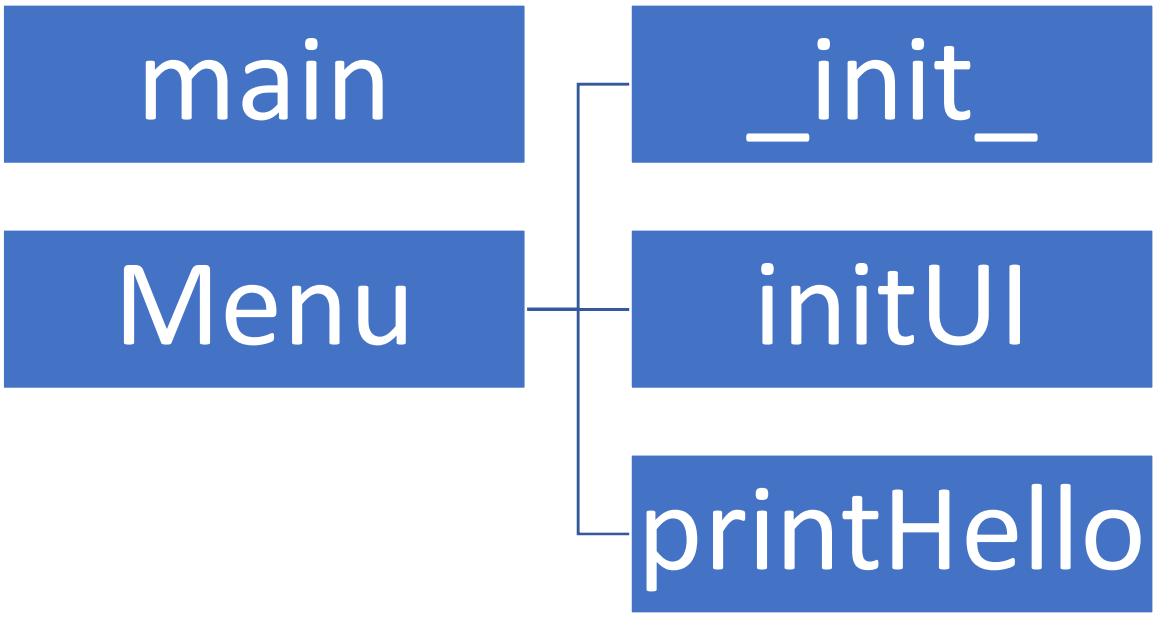
The code below shows the code to implement the new functionality. The additional code is highlighted in green. The only new command used in this section is QMessageBox. This new method is imported from the QtWidgets library by PyQt5.

```
#::-------------------------------------------------------------------
#:: To create a Menu with options this are the libraries and components that
#:: requiered. For each new option we will be o adding new components
#::-------------------------------------------------------------------
import sys
from PyQt5.QtWidgets import QMainWindow, QAction, QMenu, QApplication
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import QMessageBox

#::-------------------------------------------------------------
#:: Definition of a Class for the main manu in the application
#::-------------------------------------------------------------
class Menu(QMainWindow):

    def __init__(self):

        super().__init__()
        #::-----------------------
        #:: variables use to set the size of the window that contains the menu
        #::-----------------------
        self.left = 100
        self.top = 100
        self.width = 500
```

```
        self.height = 300

        #:: Title for the application

        self.Title = 'Demo PyQt5 - No.1 : Menus'

        #:: The initUi is call to create all the necessary elements for the menu

        self.initUI()

    def initUI(self):

        #::-------------------------------------------------
        # Creates the manu and the items
        #::-------------------------------------------------
        self.setWindowTitle(self.Title)
        self.setGeometry(self.left, self.top, self.width, self.height)
        self.statusBar()
        #::------------------------------
        # 1. Create the menu bar
        # 2. Create an item in the menu bar
        # 3. Creaate an action to be executed the option in the  menu bar is choosen
        #::------------------------------
        mainMenu = self.menuBar()
        fileMenu = mainMenu.addMenu('File')

        #:: Add another option to the Menu Bar

        exampleWin = mainMenu.addMenu ('Examples')

        #::---------------------------------------
        # Exit action
        # The following code creates the the da Exit Action along
        # with all the characteristics associated with the action
        # The Icon, a shortcut , the status tip that would appear in the window
        # and the action
        #  triggered.connect will indicate what is to be done when the item in
        # the menu is selected
        # These definitions are not available until the button is assigned
        # to the menu
        #::---------------------------------------

        exitButton = QAction(QIcon('enter.png'), '&Exit', self)
        exitButton.setShortcut('Ctrl+Q')
        exitButton.setStatusTip('Exit application')
        exitButton.triggered.connect(self.close)

        #:: This line adds the button (item element ) to the menu

        fileMenu.addAction(exitButton)

        #::-----------------------------------------------------
        #::Add Example 1 We create the item Menu Example1
        #::This option will present a message box upon request
        #::-----------------------------------------------------

        example1Button = QAction("Example1", self)
        example1Button.setStatusTip("Print Hello World 1")
        example1Button.triggered.connect(self.printhello)

        #:: We addd the example1Button action to the Menu Examples
        exampleWin.addAction(example1Button)
```

```
        #:: This line shows the windows

        self.show()

    def printhello(self):
        QMessageBox.about(self, "Results Example1", "Hello World!!!")

#::------------------------
#:: Application starts here
#::------------------------

def main():
    app = QApplication(sys.argv)  # creates the PyQt5 application
    mn = Menu()  # Cretes the menu
    sys.exit(app.exec_())  # Close the application

if __name__ == '__main__':
    main()
```

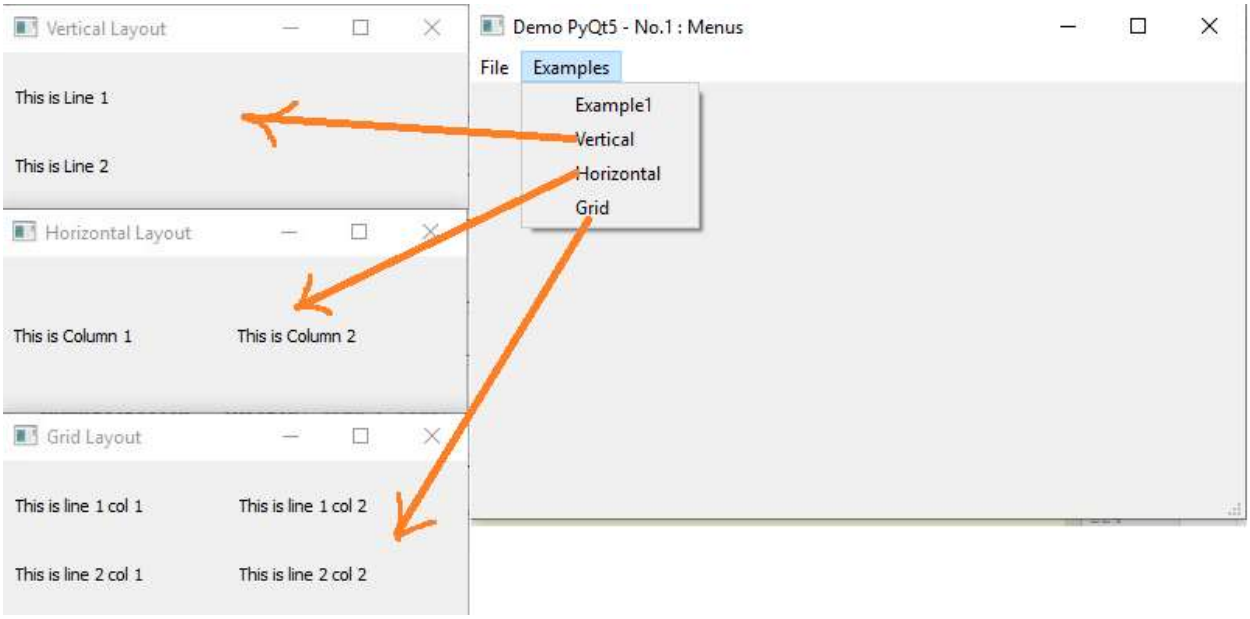
The figure below shows the new method and where it is located in the code.



No. 3. Management of different types of Layouts

The purpose of this section is to show the mechanics of presenting information in three different layouts: vertical, horizontal and grid. We will be manipulating at least two windows at the same time, the menu windows and the window that presents information in the desired layout. PyQt uses “signals” to transfer the control from one window to another upon request. These windows create communication with the user, to show data or ask for parameters to be used by the application, that is what is called “dialogs”. We will use a list of dialogs to keep track of the different iterations that are

active in the application. Signals and the list of dialogs will allow us to manage different windows with different items at the same time. The figure below shows the application with the three types of layout that can be generated with PyQT.



To implement these layouts we introduce the use of

GridLoyout, QVBoxLayout, QHBoxLayout to manage the different presentations.

QLabel to print the different labels on the windows

pyqtSignal(str) to manage the signals amongst the different windows.

The following section presents the code that implements this new functionality. The new sections are in grey.

```
#::----------------------------------------------------------------
#:: To create a Menu with options this are the libraries and components that
#:: requiered. For each new option we will be o adding new components
#::----------------------------------------------------------------
import sys
from PyQt5.QtWidgets import QMainWindow, QAction, QMenu, QApplication
from PyQt5.QtGui import QIcon
from PyQt5.QtWidgets import QMessageBox    # No.2

from PyQt5.QtCore import pyqtSlot          # No. 3
from PyQt5.QtCore import pyqtSignal        # No. 3
from PyQt5.QtWidgets import  QWidget,QLabel, QVBoxLayout, QHBoxLayout, QGridLayout #
No. 3
```

```
#::-------------------------------------------------------------------------------
---
#:: Class Vertical Layout   # No. 3
#::-------------------------------------------------------------------------------
---
class VLayoutclass(QMainWindow):  ## All the class was added in No. 3 Section
    send_fig = pyqtSignal(str)  # To manage the signals PyQT manages the
communication

    def __init__(self):
        #::-----------------------------------------------------------
        # Initialize the values of the class
        # Here the class inherits all the attributes and methods from the
QMainWindow
        #::-----------------------------------------------------------
        super(VLayoutclass, self).__init__()

        self.Title = 'Vertical Layout'
        self.initUi()

    def initUi(self):
        #::-----------------------------------------------------------------
        #  We create the type of layout QVBoxLayout (Vertical Layout )
        #  This type of layout comes from QWidget
        #::-----------------------------------------------------------------
        self.setWindowTitle(self.Title)
        self.main_widget = QWidget(self)
        self.layout = QVBoxLayout(self.main_widget)    # Creates vertical layout
        self.label1 = QLabel("This is Line 1")         # Creates label1
        self.label2 = QLabel("This is Line 2")         # Creates label2
        self.layout.addWidget(self.label1)             # Add label 1 to  layout
        self.layout.addWidget(self.label2)             # Add label 2 to layout
        self.setCentralWidget(self.main_widget)        # Creates the window with all
the elements
        self.resize(300, 100)                          # Resize the window


#::-------------------------------------------------------------------------------
---
#:: Class Horizontal Layout   # No. 3
#::-------------------------------------------------------------------------------
---
class HLayoutclass(QMainWindow):  ## All the class was added in No. 3 Section
    send_fig = pyqtSignal(str)  # To manage the signals PyQT manages the
communication

    def __init__(self):
        #::-----------------------------------------------------------
        # Initialize the values of the class
        # Here the class inherits all the attributes and methods from the
QMainWindow
        #::-----------------------------------------------------------
        super(HLayoutclass, self).__init__()

        self.Title = 'Horizontal Layout'
        self.initUi()

    def initUi(self):
        #::-----------------------------------------------------------------
        #  We create the type of layout QHBoxLayout (Horizontal Layout )
        #  This type of layout comes from QWidget
        #::-----------------------------------------------------------------
        self.setWindowTitle(self.Title)
```

```
        self.main_widget = QWidget(self)
        self.layout = QHBoxLayout(self.main_widget)    # Creates horizontal layout
        self.label1 = QLabel("This is Column 1")         # Creates label1
        self.label2 = QLabel("This is Column 2")         # Creates label2
        self.layout.addWidget(self.label1)             # Add label 1 to  layout
        self.layout.addWidget(self.label2)             # Add label 2 to layout
        self.setCentralWidget(self.main_widget)        # Creates the window with all
the elements
        self.resize(300, 100)                          # Resize the window


#::-------------------------------------------------------------------------------
---
#:: Class Horizontal Layout   # No. 3
#::-------------------------------------------------------------------------------
---
class GLayoutclass(QMainWindow):  ## All the class was added in No. 3 Section
    send_fig = pyqtSignal(str)  # To manage the signals PyQT manages the
communication

    def __init__(self):
        #::-----------------------------------------------------------
        # Initialize the values of the class
        # Here the class inherits all the attributes and methods from the
QMainWindow
        #::-----------------------------------------------------------
        super(GLayoutclass, self).__init__()

        self.Title = 'Grid Layout'
        self.initUi()

    def initUi(self):
        #::---------------------------------------------------------------
        #  We create the type of layout QGridLayout (Horizontal Layout )
        #  This type of layout comes from QWidget
        #::---------------------------------------------------------------
        self.setWindowTitle(self.Title)
        self.main_widget = QWidget(self)
        self.layout = QGridLayout(self.main_widget)    # Creates horizontal layout
        self.label1 = QLabel("This is line 1 col 1")        # Creates label1
        self.label2 = QLabel("This is line 1 col 2")        # Creates label2
        self.label3 = QLabel("This is line 2 col 1")        # Creates label3
        self.label4 = QLabel("This is line 2 col 2")        # Creates label4
        self.layout.addWidget(self.label1,0,0)            # Add label 1 to  layout
        self.layout.addWidget(self.label2,0,1)            # Add label 2 to layout
        self.layout.addWidget(self.label3,1,0)            # Add label 3 to layout
        self.layout.addWidget(self.label4,1,1)            # Add label 4 to layout

        self.setCentralWidget(self.main_widget)        # Creates the window with all
the elements
        self.resize(300, 100)                          # Resize the window


#::-------------------------------------------------------------
#:: Definition of a Class for the main manu in the application
#::-------------------------------------------------------------
class Menu(QMainWindow):

    def __init__(self):

        super().__init__()
        #::-----------------------
        #:: variables use to set the size of the window that contains the menu
```

```
        #::-----------------------
        self.left = 100
        self.top = 100
        self.width = 500
        self.height = 300

        #:: Title for the application

        self.Title = 'Demo PyQt5 - No.1 : Menus'

        #:: The initUi is call to create all the necessary elements for the menu

        self.initUI()

    def initUI(self):

        #::-------------------------------------------------
        # Creates the manu and the items
        #::-------------------------------------------------
        self.setWindowTitle(self.Title)
        self.setGeometry(self.left, self.top, self.width, self.height)
        self.statusBar()
        #::------------------------------
        # 1. Create the menu bar
        # 2. Create an item in the menu bar
        # 3. Creaate an action to be executed the option in the  menu bar is choosen
        #::------------------------------
        mainMenu = self.menuBar()
        fileMenu = mainMenu.addMenu('File')

        #:: Add another option to the Menu Bar

        exampleWin = mainMenu.addMenu ('Examples')    # No. 2

        #::--------------------------------------
        # Exit action
        # The following code creates the the da Exit Action along
        # with all the characteristics associated with the action
        # The Icon, a shortcut , the status tip that would appear in the window
        # and the action
        #  triggered.connect will indicate what is to be done when the item in
        # the menu is selected
        # These definitions are not available until the button is assigned
        # to the menu
        #::--------------------------------------

        exitButton = QAction(QIcon('enter.png'), '&Exit', self)
        exitButton.setShortcut('Ctrl+Q')
        exitButton.setStatusTip('Exit application')
        exitButton.triggered.connect(self.close)

        #:: This line adds the button (item element ) to the menu

        fileMenu.addAction(exitButton)

        #::-----------------------------------------------------
        #::Add Example 1 We create the item Menu Example1
        #::This option will present a message box upon request
        #::-----------------------------------------------------

        example1Button = QAction("Example1", self)     # No. 2
        example1Button.setStatusTip("Print Hello World 1")    # No. 2
        example1Button.triggered.connect(self.printhello)     # No. 2
```

```
        #:: We addd the example1Button action to the Menu Examples
        exampleWin.addAction(example1Button)     # No. 2

        #::-------------------------------------------------------------
        #:: Add button for Vertical Layout    # No.3
        #::-------------------------------------------------------------

        example2Button = QAction("Vertical", self)     # No. 3
        example2Button.setStatusTip("Example of vertical layout")   # No. 3
        example2Button.triggered.connect(self.VLayout)   # No. 3

        #:: We addd the example2Button to the menu examples
        exampleWin.addAction(example2Button)     # No. 3

        #::-------------------------------------------------------------
        #:: Add button for Horizontal Layout    # No.3
        #::-------------------------------------------------------------

        example3Button = QAction("Horizontal", self)   # No. 3
        example3Button.setStatusTip("Example of horizontal layout")   # No. 3
        example3Button.triggered.connect(self.HLayout)   # No. 3

        #:: We addd the example2Button to the menu examples
        exampleWin.addAction(example3Button)   # No. 3

        #::-------------------------------------------------------------
        #:: Add button for Grid Layout    # No.3
        #::-------------------------------------------------------------

        example4Button = QAction("Grid", self)   # No. 3
        example4Button.setStatusTip("Example of Grid layout")   # No. 3
        example4Button.triggered.connect(self.GLayout)   # No. 3

        #:: We addd the example2Button to the menu examples
        exampleWin.addAction(example4Button)   # No. 3


        #:: Creates an empty list of dialogs to keep track of
        #:: all the iterations

        self.dialogs = list()

        #:: This line shows the windows
        self.show()

    def printhello(self):     # No. 2
        QMessageBox.about(self, "Results Example1", "Hello World!!!")   # No. 2

    def VLayout(self):  # No. 3
        dialog = VLayoutclass()  #  Creates an object with Vertical class
        self.dialogs.append(dialog)  # Appends the list of dialogs
        dialog.show()      # Show the window

    def HLayout(self):  # No. 3
        dialog = HLayoutclass()     # Creates an object with the Horizontal class
        self.dialogs.append(dialog) # Appeds the list of dialogs
        dialog.show()    # Show the window

    def GLayout(self):  # No. 3
        dialog = GLayoutclass()     # Creates an object with the Horizontal class
        self.dialogs.append(dialog) # Appeds the list of dialogs
        dialog.show()    # Show the window
```

```
#::------------------------
#:: Application starts here
#::------------------------

def main():
    app = QApplication(sys.argv)  # creates the PyQt5 application
    mn = Menu()  # Cretes the menu
    sys.exit(app.exec_())  # Close the application

if __name__ == '__main__':
    main()
```

The figure below shows the organization of the new methods.