Python GUI

Meher Krishna Patel

Created on : Octorber, 2017

Last updated : October, 2018

Table of contents

Chapter 1

Python Qt5 (Under progress)

1.1 Basics

1.1.1 Creating widget

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import QApplication, QWidget

def main():
    app = QApplication(sys.argv)
    w = QWidget()

    # title for widget
    w.setWindowTitle("PyQt5")

    # left margin, top margin, width, height
    # w.setGeometry(250, 250, 200, 150)
    # or use below two lines
    w.resize(200, 150)
    w.move(250, 250)
    w.show()

    # wait to exit
    sys.exit(app.exec_())

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

1.1.2 Hello World

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import QApplication, QWidget, QLabel

def main():
    app = QApplication(sys.argv)
    w = QWidget()
```

(continues on next page)

(continued from previous page)

```
    # create label
    b = QLabel(w)
    b.setText("Hello World") # text value
    b.move(75, 75) # location of label

    # title for widget
    w.setWindowTitle("PyQt5")

    # left margin, top margin, width, height
    # w.setGeometry(250, 250, 200, 150)
    # or use below two lines
    w.resize(200, 150)
    w.move(250, 250)

    w.show()

    # wait to exit
    sys.exit(app.exec_())

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

1.1.3 Use classes for design

Codes are more manageable with classes. The above code can be rewritten as below.

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import QApplication, QWidget, QLabel

class MainPage(QWidget):
    def __init__(self, title=" "):
        super().__init__()  # inherit init of QWidget
        self.title = title
        self.left = 250
        self.top = 250
        self.width = 200
        self.height = 150
        self.widget()

    def widget(self):
        self.setWindowTitle(self.title)
        self.setGeometry(self.left, self.top, self.width, self.height)
        label1 = QLabel("Hello World", self)
        label1.move(75, 75)

        self.show()

def main():
    app = QApplication(sys.argv)
    w = MainPage(title="PyQt5")
    sys.exit(app.exec_())

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

1.2 Widgets

In this section, we will learn to add various widgets which are available in PyQt5.

1.2.1 Label

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import (
        QApplication, QWidget, QLabel
    )
from PyQt5.QtCore import QRect

class MainPage(QWidget):
    def __init__(self, title=" "):
        super().__init__()  # inherit init of QWidget
        self.title = title
        self.left = 250
        self.top = 250
        self.width = 200
        self.height = 150
        self.widget()

    def widget(self):
        # window setup
        self.setWindowTitle(self.title)
        # self.setGeometry(self.left, self.top, self.width, self.height)
        ## use above line or below
        self.resize(self.width, self.height)
        self.move(self.left, self.top)

        # add label
        self.label1 = QLabel(self, text="Hello World!\nWelcome to PyQt5 Tutorial")
        # margin: left, top; width, height
        self.label1.setGeometry(QRect(50, 5, 100, 50))
        self.label1.setWordWrap(True) # allow word-wrap

        self.show()


def main():
    app = QApplication(sys.argv)
    w = MainPage(title="PyQt5")
    sys.exit(app.exec_())


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

1.2.2 Push button

- In the below code, the label will be changed after pressing the 'submit' button.

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import (
```

(continues on next page)

(continued from previous page)

```
        QApplication, QWidget, QLabel, QPushButton
    )
from PyQt5.QtCore import pyqtSlot, QRect, QCoreApplication

class MainPage(QWidget):
    def __init__(self, title=" "):
        super().__init__()  # inherit init of QWidget
        self.title = title
        self.left = 250
        self.top = 250
        self.width = 200
        self.height = 150
        self.widget()

    def widget(self):
        # window setup
        self.setWindowTitle(self.title)
        # self.setGeometry(self.left, self.top, self.width, self.height)
        ## use above line or below
        self.resize(self.width, self.height)
        self.move(self.left, self.top)

        # add label
        self.label1 = QLabel(self, text="Hello World!\nWelcome to PyQt5 Tutorial")
        # margin: left, top; width, height
        self.label1.setGeometry(QRect(50, 5, 100, 50))
        self.label1.setWordWrap(True) # allow word-wrap

        # add button
        self.btn1 = QPushButton(self, text="Submit")
        self.btn1.setToolTip("Change value of label")
        self.btn1.move(5, 95)
        self.btn1.clicked.connect(self.change_label)

        self.btn2 = QPushButton(self, text="Close")
        self.btn2.setToolTip("Exit window")
        self.btn2.move(95, 95)
        self.btn2.clicked.connect(self.exit_window)

        self.show()

    @pyqtSlot()
    def change_label(self):
        self.label1.setText("Submit button is pressed ")

    @pyqtSlot()
    def exit_window(self):
        QCoreApplication.instance().quit()

def main():
    app = QApplication(sys.argv)
    w = MainPage(title="PyQt5")
    sys.exit(app.exec_())


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

1.2.3 Frame and Group box

We can create different frames and then arrange items inside it as below. Note that the location items inside the frame are relative to location of frame.

Note:

- To create a set of radio buttons or checkboxes, we need to define the radio buttons inside the Frame or groupBox.
- Replace 'QFrame' with 'QGroupBox' to create a groupbox.

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import (
        QApplication, QWidget, QLabel, QPushButton, QCheckBox, QFrame
    )
from PyQt5.QtCore import pyqtSlot, QRect

class MainPage(QWidget):
    def __init__(self, title=" "):
        super().__init__()  # inherit init of QWidget
        self.title = title
        self.left = 250
        self.top = 250
        self.width = 400
        self.height = 300
        self.widget()

    def widget(self):
        # window setup
        self.setWindowTitle(self.title)
        # self.setGeometry(self.left, self.top, self.width, self.height)
        ## use above line or below
        self.resize(self.width, self.height)
        self.move(self.left, self.top)

        # create frame for a set of checkbox
        self.frame1 = QFrame(self)
        self.frame1.setGeometry(QRect(40, 40, 250, 80))
        # push button to display output on label
        self.btn1 = QPushButton(self.frame1, text="Submit")
        # location of btn relative to frame2 i.e. (40+0, 40+20)
        self.btn1.move(0, 50)
        self.btn1.clicked.connect(self.btn1_click)
        # selected value will be displayed on label
        self.label1 = QLabel(self.frame1)
        self.label1.setGeometry(QRect(0, 20, 500, 20))


        # create frame for a set of checkbox
        self.frame2 = QFrame(self)
        self.frame2.setGeometry(QRect(40, 150, 250, 80))
        # push button to display output on label
        self.btn2 = QPushButton(self.frame2, text="Submit")
        self.btn2.move(0, 50) # location of btn relative to frame2
        self.btn2.clicked.connect(self.btn2_click)
        # selected value will be displayed on label
        self.label2 = QLabel(self.frame2)
        self.label2.setGeometry(QRect(0, 20, 500, 20))
```

(continues on next page)

(continued from previous page)

```
        self.show()

    @pyqtSlot()
    def btn1_click(self):
        self.label1.setText("Button is pressed in Frame 1")

    @pyqtSlot()
    def btn2_click(self):
        self.label2.setText("Button is pressed in Frame 2")


def main():
    app = QApplication(sys.argv)
    w = MainPage(title="PyQt5")
    sys.exit(app.exec_())


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

1.2.4 Radio button

- In the below code, the value of selected 'radio-button' will be shown on Lable (after pressing the submit button)

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import (
        QApplication, QWidget, QLabel, QPushButton, QRadioButton, QFrame
    )
from PyQt5.QtCore import pyqtSlot, QRect

class MainPage(QWidget):
    def __init__(self, title=" "):
        super().__init__()  # inherit init of QWidget
        self.title = title
        self.left = 250
        self.top = 250
        self.width = 400
        self.height = 300
        self.widget()

    def widget(self):
        # window setup
        self.setWindowTitle(self.title)
        # self.setGeometry(self.left, self.top, self.width, self.height)
        ## use above line or below
        self.resize(self.width, self.height)
        self.move(self.left, self.top)

        # create frame for a set of radio button
        self.frame1 = QFrame(self)
        self.frame1.move(40, 40)

        self.radioBtn1 = QRadioButton("Yes", self.frame1)
        self.radioBtn1.setChecked(True)  # select by default
        self.radioBtn1.move(0, 0)
        self.radioBtn2 = QRadioButton("No", self.frame1)
```

(continues on next page)

(continued from previous page)

```
        self.radioBtn2.move(0, 20)

        # push button to display output on label
        self.btn1 = QPushButton(self.frame1, text="Submit")
        self.btn1.move(60, 20)
        self.btn1.clicked.connect(self.btn1_click)

        # selected value will be displayed on label
        self.label1 = QLabel(self.frame1)
        self.label1.setGeometry(QRect(60, 0, 100, 20))
        # self.label1.move(60, 0)


        # create another frame for other set of radio button
        self.frame2 = QFrame(self)
        self.frame2.move(40, 100)
        self.radioBtn3 = QRadioButton("College", self.frame2)
        self.radioBtn3.move(0, 0)
        self.radioBtn4 = QRadioButton("School", self.frame2)
        self.radioBtn4.move(0, 20)

        self.show()

    @pyqtSlot()
    def btn1_click(self):
        if self.radioBtn1.isChecked():
            self.label1.setText("You pressed Yes")
        elif self.radioBtn2.isChecked():
            self.label1.setText("You pressed No")
        else:
            self.label1.setText("Choose Yes or No")



def main():
    app = QApplication(sys.argv)
    w = MainPage(title="PyQt5")
    sys.exit(app.exec_())


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

1.2.5 Checkbox

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import (
        QApplication, QWidget, QLabel, QPushButton, QCheckBox, QFrame
    )
from PyQt5.QtCore import pyqtSlot, QRect

class MainPage(QWidget):
    def __init__(self, title=" "):
        super().__init__()  # inherit init of QWidget
        self.title = title
        self.left = 250
        self.top = 250
```

(continues on next page)

(continued from previous page)

```
        self.width = 400
        self.height = 300
        self.widget()

    def widget(self):
        # window setup
        self.setWindowTitle(self.title)
        # self.setGeometry(self.left, self.top, self.width, self.height)
        ## use above line or below
        self.resize(self.width, self.height)
        self.move(self.left, self.top)

        # create frame for a set of checkbox
        self.frame1 = QFrame(self)
        self.frame1.move(40, 40)

        self.checkbox1 = QCheckBox("C++", self.frame1)
        self.checkbox1.setChecked(True)  # select by default
        self.checkbox1.move(0, 0)
        self.checkbox2 = QCheckBox("Python", self.frame1)
        self.checkbox2.move(0, 20)

        # push button to display output on label
        self.btn1 = QPushButton(self.frame1, text="Submit")
        self.btn1.move(70, 20)
        self.btn1.clicked.connect(self.btn1_click)

        # selected value will be displayed on label
        self.label1 = QLabel(self.frame1)
        self.label1.setGeometry(QRect(60, 0, 500, 20))
        # self.label1.move(60, 0)


        # create another frame for other set of checkbox
        self.frame2 = QFrame(self)
        self.frame2.move(40, 100)
        self.checkbox3 = QCheckBox("College", self.frame2)
        self.checkbox3.move(0, 0)
        self.checkbox4 = QCheckBox("School", self.frame2)
        self.checkbox4.move(0, 20)

        self.show()

    @pyqtSlot()
    def btn1_click(self):
        selected_val = []
        if self.checkbox1.isChecked():
            selected_val.append(self.checkbox1.text())
        if self.checkbox2.isChecked():
            selected_val.append(self.checkbox2.text())

        val = "Preferred Languages: "
        for i in selected_val:
            val = val + i + ", "
        self.label1.setText(val)


def main():
    app = QApplication(sys.argv)
    w = MainPage(title="PyQt5")
    sys.exit(app.exec_())
```

(continues on next page)

(continued from previous page)

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

1.2.6 Spin box

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import (
        QApplication, QWidget, QLabel, QPushButton, QSpinBox, QFrame
    )
from PyQt5.QtCore import pyqtSlot, QRect

class MainPage(QWidget):
    def __init__(self, title=" "):
        super().__init__()  # inherit init of QWidget
        self.title = title
        self.left = 250
        self.top = 250
        self.width = 400
        self.height = 300
        self.widget()

    def widget(self):
        # window setup
        self.setWindowTitle(self.title)
        # self.setGeometry(self.left, self.top, self.width, self.height)
        ## use above line or below
        self.resize(self.width, self.height)
        self.move(self.left, self.top)

        # create frame for a set of checkbox
        self.frame1 = QFrame(self)
        self.frame1.setGeometry(QRect(40, 40, 250, 80))
        # create spin box
        self.spinbox1 = QSpinBox(self.frame1)
        self.spinbox1.setValue(3) # default value
        self.spinbox1.setMinimum(0) # minimum value
        self.spinbox1.setMaximum(6) # maximum value
        self.spinbox1.move(0, 50)
        self.spinbox1.valueChanged.connect(self.spinbox1_changed)
        # spinbox value will be displayed on label
        self.label1 = QLabel(self.frame1, text="Value in spin box is " + str(self.spinbox1.value()))
        self.label1.setGeometry(QRect(0, 20, 500, 20))


        self.show()

    @pyqtSlot()
    def spinbox1_changed(self):
        self.label1.setText("Value in spin box is "+str(self.spinbox1.value()))


def main():
    app = QApplication(sys.argv)
    w = MainPage(title="PyQt5")
    sys.exit(app.exec_())
```

(continues on next page)

(continued from previous page)

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

1.2.7 Slider

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import (
        QApplication, QWidget, QLabel, QPushButton, QSlider, QFrame
    )
from PyQt5.QtCore import pyqtSlot, QRect, Qt

class MainPage(QWidget):
    def __init__(self, title=" "):
        super().__init__()  # inherit init of QWidget
        self.title = title
        self.left = 250
        self.top = 250
        self.width = 400
        self.height = 300
        self.widget()

    def widget(self):
        # window setup
        self.setWindowTitle(self.title)
        # self.setGeometry(self.left, self.top, self.width, self.height)
        ## use above line or below
        self.resize(self.width, self.height)
        self.move(self.left, self.top)

        # create frame for a set of checkbox
        self.frame1 = QFrame(self)
        self.frame1.setGeometry(QRect(40, 40, 250, 180))
        # create spin box
        self.slider1 = QSlider(self.frame1)
        self.slider1.setOrientation(Qt.Horizontal) # Horizontal / Vertical
        self.slider1.setTickInterval(1)
        self.slider1.setTickPosition(QSlider.TicksBelow)
        self.slider1.setTickInterval(2)
        self.slider1.setValue(3) # default value
        self.slider1.setMinimum(0) # minimum value
        self.slider1.setMaximum(6) # maximum value
        self.slider1.move(0, 50)
        self.slider1.valueChanged.connect(self.slider1_changed)
        # slider value will be displayed on label
        self.label1 = QLabel(self.frame1, text="Slider is at " + str(self.slider1.value()))
        self.label1.setGeometry(QRect(0, 20, 500, 20))


        self.show()

    @pyqtSlot()
    def slider1_changed(self):
        self.label1.setText("Slider is at "+str(self.slider1.value()))
```

(continues on next page)

(continued from previous page)

```
def main():
    app = QApplication(sys.argv)
    w = MainPage(title="PyQt5")
    sys.exit(app.exec_())


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

1.2.8 Line edit box

Single line can be written in Line edit box

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import (
        QApplication, QWidget, QLabel, QPushButton, QLineEdit, QFrame
    )
from PyQt5.QtCore import pyqtSlot, QRect, Qt

class MainPage(QWidget):
    def __init__(self, title=" "):
        super().__init__()  # inherit init of QWidget
        self.title = title
        self.left = 250
        self.top = 250
        self.width = 400
        self.height = 300
        self.widget()

    def widget(self):
        # window setup
        self.setWindowTitle(self.title)
        # self.setGeometry(self.left, self.top, self.width, self.height)
        ## use above line or below
        self.resize(self.width, self.height)
        self.move(self.left, self.top)

        # create frame for a set of checkbox
        self.frame1 = QFrame(self)
        self.frame1.setGeometry(QRect(40, 40, 250, 180))
        # create line edit box
        self.line_edit1 = QLineEdit(self.frame1)
        self.line_edit1.move(0, 0)
        self.line_edit1.textChanged.connect(self.line_edit1_changed)
        # line_edit value will be displayed on label
        self.label1 = QLabel(self.frame1)
        self.label1.setGeometry(QRect(0, 20, 500, 20))

        # Password field
        self.line_edit2 = QLineEdit(self.frame1)
        self.line_edit2.move(0, 50)
        self.line_edit2.setEchoMode(QLineEdit.Password)


        self.show()

    @pyqtSlot()
```

(continues on next page)

(continued from previous page)

```
    def line_edit1_changed(self):
        self.label1.setText("Text in Line edit: "+str(self.line_edit1.text()))


def main():
    app = QApplication(sys.argv)
    w = MainPage(title="PyQt5")
    sys.exit(app.exec_())


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

1.2.9 Text edit box

Multiple lines can be written in 'text edit box'

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import (
        QApplication, QWidget, QLabel, QPushButton, QTextEdit, QFrame
    )
from PyQt5.QtCore import pyqtSlot, QRect, Qt

class MainPage(QWidget):
    def __init__(self, title=" "):
        super().__init__()  # inherit init of QWidget
        self.title = title
        self.left = 250
        self.top = 250
        self.width = 400
        self.height = 300
        self.widget()

    def widget(self):
        # window setup
        self.setWindowTitle(self.title)
        # self.setGeometry(self.left, self.top, self.width, self.height)
        ## use above line or below
        self.resize(self.width, self.height)
        self.move(self.left, self.top)

        # create frame for a set of checkbox
        self.frame1 = QFrame(self)
        self.frame1.setGeometry(QRect(40, 40, 250, 180))
        # create text edit box
        self.text_edit1 = QTextEdit(self.frame1)
        self.text_edit1.move(0, 0)


        self.show()


def main():
    app = QApplication(sys.argv)
    w = MainPage(title="PyQt5")
    sys.exit(app.exec_())
```

(continues on next page)

(continued from previous page)

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

1.2.10 Combobox

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import (
        QApplication, QWidget, QLabel, QPushButton, QComboBox, QFrame
    )
from PyQt5.QtCore import pyqtSlot, QRect, Qt

class MainPage(QWidget):
    def __init__(self, title=" "):
        super().__init__()  # inherit init of QWidget
        self.title = title
        self.left = 250
        self.top = 250
        self.width = 400
        self.height = 300
        self.widget()

    def widget(self):
        # window setup
        self.setWindowTitle(self.title)
        # self.setGeometry(self.left, self.top, self.width, self.height)
        ## use above line or below
        self.resize(self.width, self.height)
        self.move(self.left, self.top)

        # create frame for a set of checkbox
        self.frame1 = QFrame(self)
        self.frame1.setGeometry(QRect(40, 40, 250, 180))
        # create text edit box
        self.combo1 = QComboBox(self.frame1)
        self.combo1.move(0, 0)
        self.combo1.addItem("Python")
        self.combo1.addItems(["C", "C++", "VHDL", "Verilog"])

        self.show()


def main():
    app = QApplication(sys.argv)
    w = MainPage(title="PyQt5")
    sys.exit(app.exec_())


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

1.3 Menu bar

1.3.1 Add menu bar

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import (
        QApplication, QWidget, QLabel, QPushButton, QComboBox, QFrame, QMainWindow
    )
from PyQt5.QtCore import pyqtSlot, QRect, Qt

class MainPage(QMainWindow):
    def __init__(self, title=" "):
        super().__init__()  # inherit init of QWidget
        self.title = title
        self.left = 250
        self.top = 250
        self.width = 400
        self.height = 300
        self.widget()

    def widget(self):
        # window setup
        self.setWindowTitle(self.title)
        # self.setGeometry(self.left, self.top, self.width, self.height)
        ## use above line or below
        # self.resize(self.width, self.height) # resizable
        self.setFixedSize(self.width, self.height)  # fixed size
        self.move(self.left, self.top)

        self.menubar= self.menuBar() # add menu bar
        self.helpMenu = self.menubar.addMenu("&Help")  # Add Help in menu bar
        self.about = self.helpMenu.addAction("&About")  # Add option in Help
        self.about.setShortcut("F11") # display F11 as shortcut
        self.credits = self.helpMenu.addAction("&Credits") # Add another option in Help

        self.show()


def main():
    app = QApplication(sys.argv)
    w = MainPage(title="PyQt5")
    sys.exit(app.exec_())


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

1.3.2 Add functionality to options

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import (
        QApplication, QWidget, QLabel, QPushButton, QComboBox, QFrame, QMainWindow,
        QMessageBox

    )
```

(continues on next page)

(continued from previous page)

```
from PyQt5.QtCore import pyqtSlot, QRect, Qt

class MainPage(QMainWindow):
    def __init__(self, title=" "):
        super().__init__()  # inherit init of QWidget
        self.title = title
        self.left = 250
        self.top = 250
        self.width = 400
        self.height = 300
        self.widget()

    def widget(self):
        # window setup
        self.setWindowTitle(self.title)
        # self.setGeometry(self.left, self.top, self.width, self.height)
        ## use above line or below
        # self.resize(self.width, self.height) # resizable
        self.setFixedSize(self.width, self.height)  # fixed size
        self.move(self.left, self.top)

        self.menubar= self.menuBar() # add menu bar
        self.helpMenu = self.menubar.addMenu("&Help")  # Add Help in menu bar
        self.about = self.helpMenu.addAction("&About")  # Add option in Help
        self.about.setShortcut("F11") # display F11 as shortcut
        self.about.triggered.connect(self.aboutDef)

        self.credits = self.helpMenu.addAction("&Credits") # Add another option in Help

        self.messageBox = QMessageBox(self)
        self.messageBox.setFixedSize(self.width, self.height)  # fixed size

        self.show()

    @pyqtSlot()
    def aboutDef(self):
        message = "Parameters:\n  i = integer  d = double"
        self.messageBox.about(self, "Help ", message)
        # self.messageBox.question(self, "Parameter help", message, QMessageBox.Ok)

def main():
    app = QApplication(sys.argv)
    w = MainPage(title="PyQt5")
    sys.exit(app.exec_())


if __name__ == '__main__':
    main()



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

1.4 Qt designer

We can design the GUI using Qt-designer and then add functionality as shown in this section,

1.4.1 Installation

- Install qt5

```
(install qt5, if required)
sudo apt-fast install qt5-default
```

Install any one of two,

- qt4-designer is a part of IDE which creates only Form.
- qtcreator is complete IDE.

```
sudo apt-fast install qt4-designer
(run below to start qt4-designer)
designer

or

sudo apt-fast install qtcreator
(run below to start qtcreator)
qtcreator
```

1.4.2 Create design

First create design `mainwindow.ui` using Qt-designer or qt-creator as shown in Fig. 1.1,

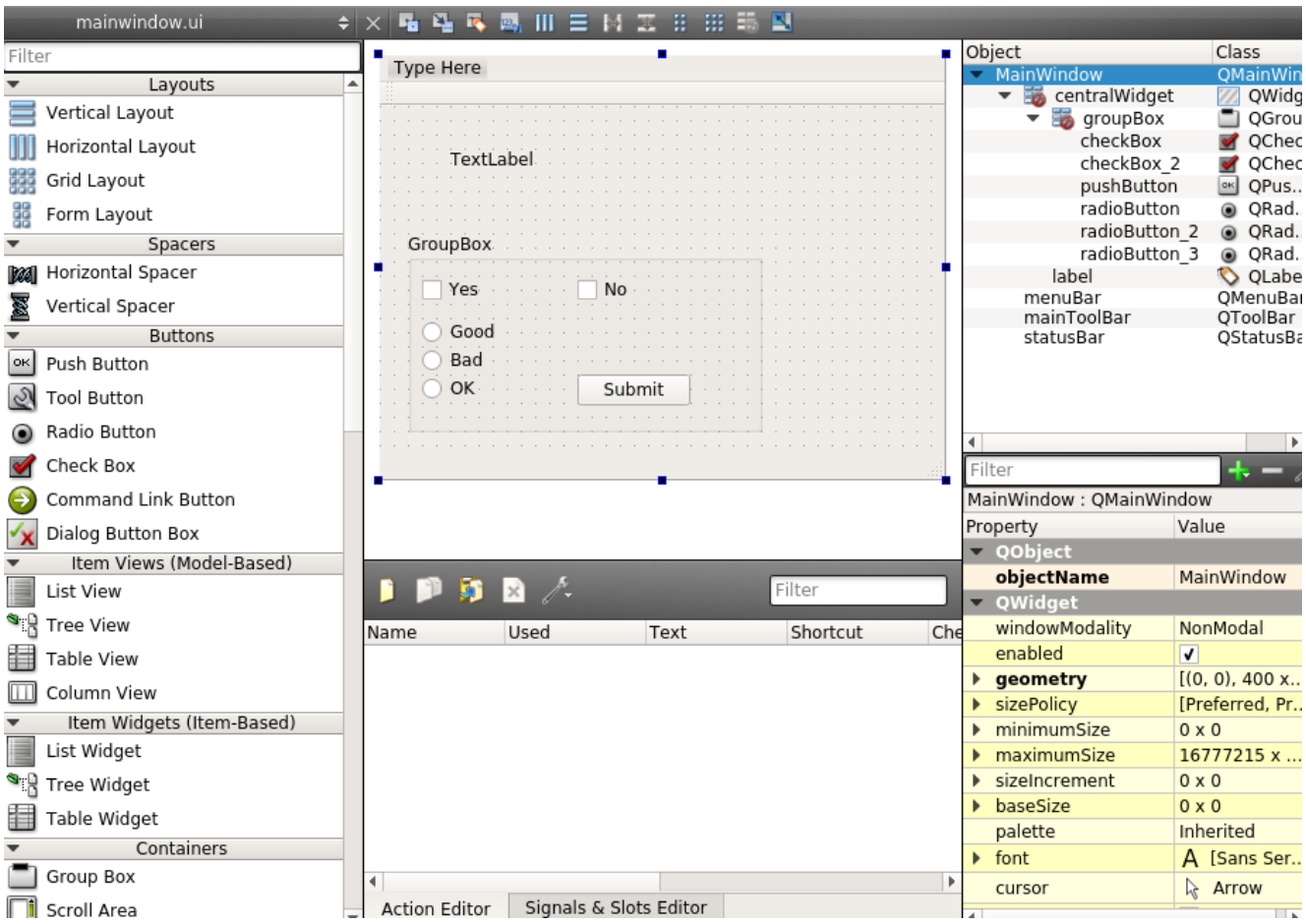

Fig. 1.1: Design using qtcreator

1.4.3 Convert code into python

The design file is saved as 'mainwindow.ui' file

```
pyuic5 -x mainwindow.ui -o qtdesign.py
```

Below is the code generated by above command,

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

# Form implementation generated from reading ui file 'mainwindow.ui'
#
# Created by: PyQt5 UI code generator 5.6
#
# WARNING! All changes made in this file will be lost!

from PyQt5 import QtCore, QtGui, QtWidgets

class Ui_MainWindow(object):
    def setupUi(self, MainWindow):
        MainWindow.setObjectName("MainWindow")
        MainWindow.resize(400, 300)
        self.centralWidget = QtWidgets.QWidget(MainWindow)
        self.centralWidget.setObjectName("centralWidget")
        self.groupBox = QtWidgets.QGroupBox(self.centralWidget)
        self.groupBox.setGeometry(QtCore.QRect(20, 90, 251, 141))
        self.groupBox.setObjectName("groupBox")
        self.radioButton = QtWidgets.QRadioButton(self.groupBox)
        self.radioButton.setGeometry(QtCore.QRect(10, 60, 100, 20))
        self.radioButton.setObjectName("radioButton")
        self.radioButton_2 = QtWidgets.QRadioButton(self.groupBox)
        self.radioButton_2.setGeometry(QtCore.QRect(10, 80, 100, 20))
        self.radioButton_2.setObjectName("radioButton_2")
        self.radioButton_3 = QtWidgets.QRadioButton(self.groupBox)
        self.radioButton_3.setGeometry(QtCore.QRect(10, 100, 100, 20))
        self.radioButton_3.setObjectName("radioButton_3")
        self.pushButton = QtWidgets.QPushButton(self.groupBox)
        self.pushButton.setGeometry(QtCore.QRect(120, 100, 80, 22))
        self.pushButton.setObjectName("pushButton")
        self.checkBox = QtWidgets.QCheckBox(self.groupBox)
        self.checkBox.setGeometry(QtCore.QRect(10, 30, 85, 20))
        self.checkBox.setObjectName("checkBox")
        self.checkBox_2 = QtWidgets.QCheckBox(self.groupBox)
        self.checkBox_2.setGeometry(QtCore.QRect(120, 30, 85, 20))
        self.checkBox_2.setObjectName("checkBox_2")
        self.label = QtWidgets.QLabel(self.centralWidget)
        self.label.setGeometry(QtCore.QRect(50, 30, 241, 16))
        self.label.setObjectName("label")
        MainWindow.setCentralWidget(self.centralWidget)
        self.menuBar = QtWidgets.QMenuBar(MainWindow)
        self.menuBar.setGeometry(QtCore.QRect(0, 0, 400, 19))
        self.menuBar.setObjectName("menuBar")
        MainWindow.setMenuBar(self.menuBar)
        self.mainToolBar = QtWidgets.QToolBar(MainWindow)
        self.mainToolBar.setObjectName("mainToolBar")
        MainWindow.addToolBar(QtCore.Qt.TopToolBarArea, self.mainToolBar)
        self.statusBar = QtWidgets.QStatusBar(MainWindow)
        self.statusBar.setObjectName("statusBar")
        MainWindow.setStatusBar(self.statusBar)

        self.retranslateUi(MainWindow)
        QtCore.QMetaObject.connectSlotsByName(MainWindow)
```

(continues on next page)

(continued from previous page)

```
    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.groupBox.setTitle(_translate("MainWindow", "GroupBox"))
        self.radioButton.setText(_translate("MainWindow", "Good"))
        self.radioButton_2.setText(_translate("MainWindow", "Bad"))
        self.radioButton_3.setText(_translate("MainWindow", "OK"))
        self.pushButton.setText(_translate("MainWindow", "Submit"))
        self.checkBox.setText(_translate("MainWindow", "Yes"))
        self.checkBox_2.setText(_translate("MainWindow", "No"))
        self.label.setText(_translate("MainWindow", "TextLabel"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

Now, we can excute the python file as below, which will open the design as shown in Fig. 1.2

```
(run file)
python qtdesign.py
```

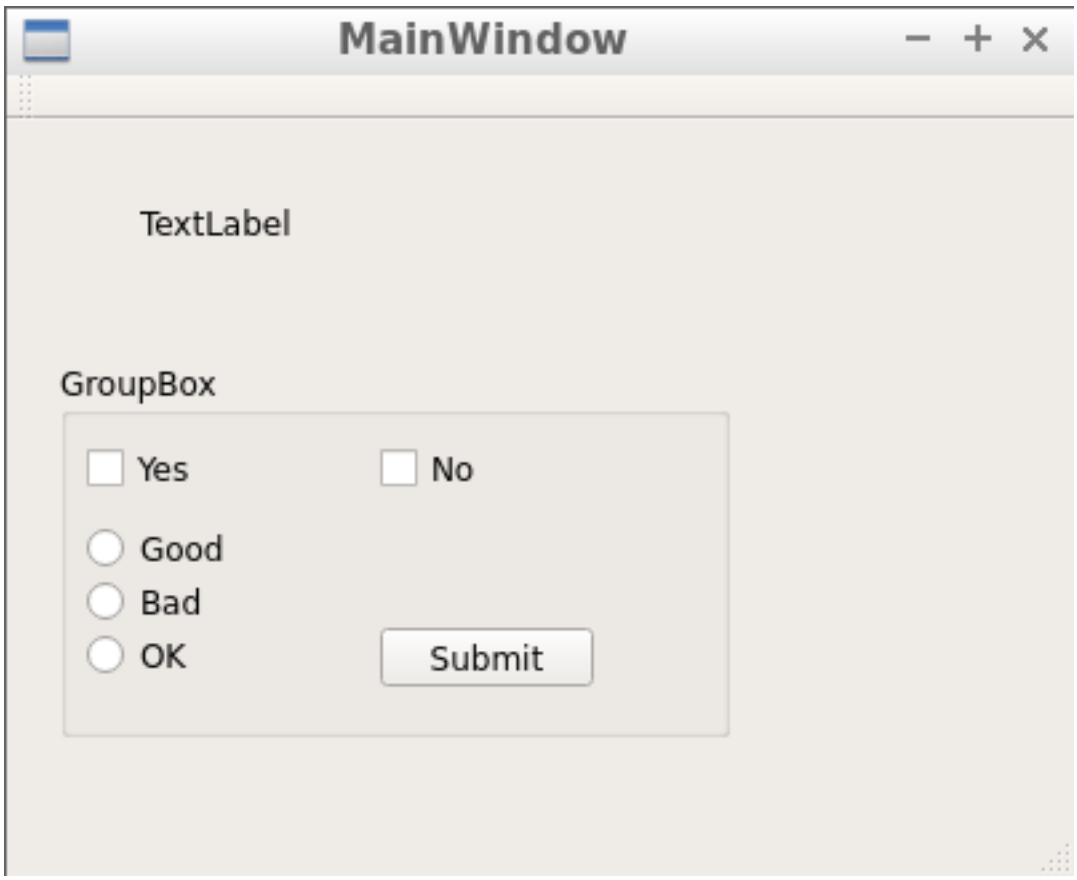

Fig. 1.2: GUI after conversion

1.4.4 Add functionality to design

It is better to create a new file and import 'qtdesign.py' file to it as shown below,

Note: Since we did not modify the qtdesign.py file, therefore we can add more widgets to 'mainwindow.ui' without modifying the code in 'myactions.py' (as long as the name of the widgets are not changed).

```
# myactions.py

from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtCore import pyqtSlot, QRect

from qtdesign import Ui_MainWindow


class MyActions(Ui_MainWindow):
    def __init__(self, title=" "):
        self.title = title
        self.left = 250
        self.top = 250
        self.width = 200
        self.height = 150

    # update setupUi
    def setupUi(self, MainWindow):
        super().setupUi(MainWindow)
        # MainWindow.resize(400, 300) # do not modify it
        MainWindow.move(self.left, self.top)  # set location for window
        MainWindow.setWindowTitle(self.title) # change title

        self.myactions() # add actions for different buttons

    # define actions here
    def myactions(self):
        self.pushButton.clicked.connect(self.change_label)


    @pyqtSlot()
    def change_label(self):
        self.label.setText("Submit button is pressed ")

if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = MyActions("PyQt5")
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

1.5 Database

Note: See MySQL with python guide for more MySQL commands.

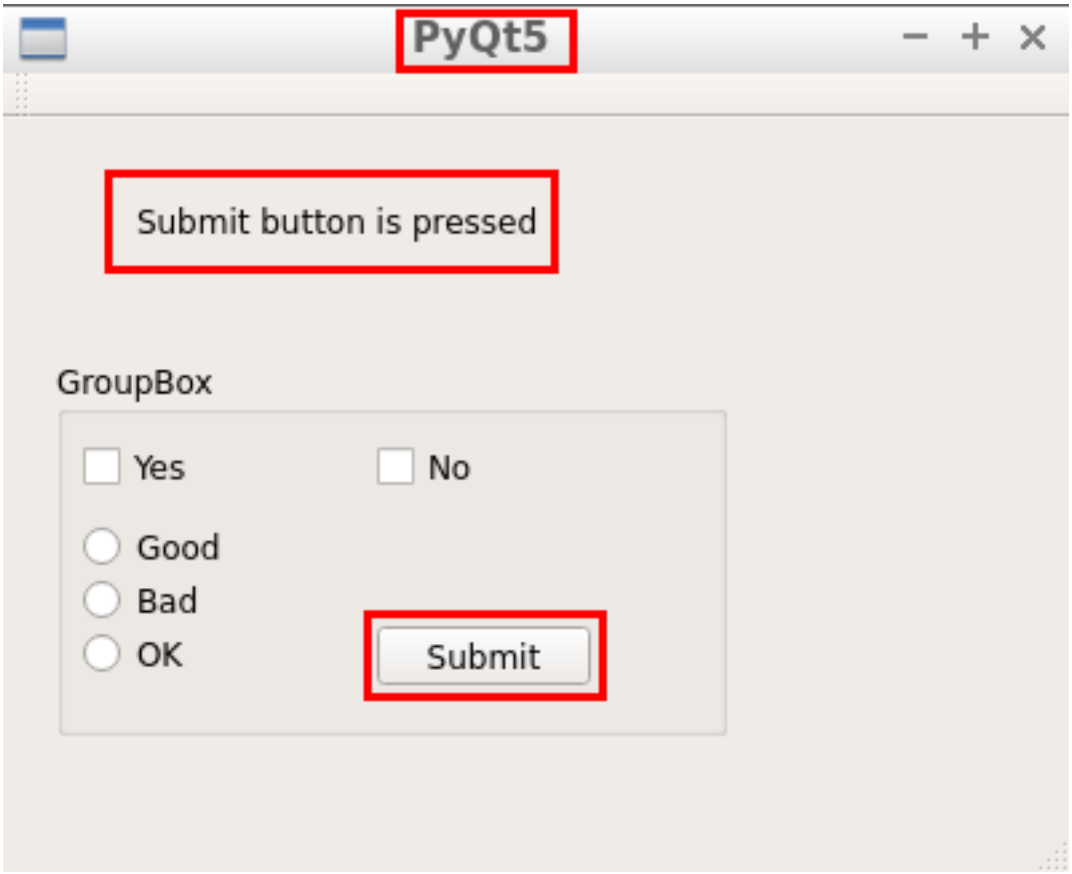

Fig. 1.3: Perform action on 'submit'

1.5.1 Connect to Database

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import (
        QApplication, QWidget, QLabel, QPushButton, QComboBox, QFrame, QMainWindow,
        QMessageBox

    )
from PyQt5.QtCore import pyqtSlot, QRect, Qt

import MySQLdb as mq

class MainPage(QMainWindow):
    def __init__(self, title=" "):
        super().__init__()  # inherit init of QWidget
        self.title = title
        self.left = 250
        self.top = 250
        self.width = 400
        self.height = 300
        self.widget()

    def widget(self):
        # window setup
        self.setWindowTitle(self.title)
        # self.setGeometry(self.left, self.top, self.width, self.height)
```

(continues on next page)

(continued from previous page)

```
        ## use above line or below
        # self.resize(self.width, self.height) # resizable
        self.setFixedSize(self.width, self.height)  # fixed size
        self.move(self.left, self.top)

        # create frame for a set of checkbox
        self.frame1 = QFrame(self)
        self.frame1.setGeometry(QRect(40, 40, 250, 80))

        self.btn1 = QPushButton(self.frame1, text="Connect")
        self.btn1.move(50, 50)
        self.btn1.clicked.connect(self.connect_db)

        self.show()

    def connect_db(self):
        try:
            conn = mq.connect(host='localhost', user='root', password='root', db='qtdb')
            print("Connected")
        except mq.Error as err:
            print(err)
        else:
            print("Connection closed")
            conn.close()

def main():
    app = QApplication(sys.argv)
    w = MainPage(title="PyQt5")
    sys.exit(app.exec_())


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

1.5.2 Create table and insert data

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import (
        QApplication, QWidget, QLabel, QPushButton, QComboBox, QFrame, QMainWindow,
        QMessageBox, QLineEdit

    )
from PyQt5.QtCore import pyqtSlot, QRect, Qt

import MySQLdb as mq

class MainPage(QMainWindow):
    def __init__(self, title=" "):
        super().__init__()  # inherit init of QWidget
        self.title = title
        self.left = 250
        self.top = 50
        self.width = 400
        self.height = 250
        self.widget()

    def widget(self):
```

(continues on next page)

(continued from previous page)

```
        # window setup
        self.setWindowTitle(self.title)
        # self.setGeometry(self.left, self.top, self.width, self.height)
        ## use above line or below
        # self.resize(self.width, self.height) # resizable
        self.setFixedSize(self.width, self.height)  # fixed size
        self.move(self.left, self.top)

        # create frame for a set of checkbox
        self.frame1 = QFrame(self)
        self.frame1.setGeometry(QRect(40, 40, 250, 250))

        # create table
        self.tbl_name = QLineEdit(self.frame1)
        self.tbl_name.setPlaceholderText("Table name")
        self.tbl_name.setGeometry(0, 0, 100, 30)

        self.btn1 = QPushButton(self.frame1, text="Create Table")
        self.btn1.move(0, 50)
        self.btn1.clicked.connect(self.create_table)

        # insert data
        self.tbl_name2 = QLineEdit(self.frame1)
        self.tbl_name2.setPlaceholderText("Table name")
        self.tbl_name2.setGeometry(150, 0, 100, 30)

        self.name = QLineEdit(self.frame1)
        self.name.setPlaceholderText("Name")
        self.name.setGeometry(150, 50, 100, 30)

        self.age = QLineEdit(self.frame1)
        self.age.setPlaceholderText("Age")
        self.age.setGeometry(150, 100, 100, 30)

        self.btn2 = QPushButton(self.frame1, text="Insert")
        self.btn2.move(150, 150)
        self.btn2.clicked.connect(self.insert_data)


        self.show()

    def connect_db(self):
        try:
            self.conn = mq.connect(host='localhost', user='root', password='root', db='qtdb')
            self.cursor = self.conn.cursor()
            print("Connected")
        except mq.Error as err:
            print(err)

    def disconnect_db(self):
        """ commit changes to database and close connection """
        self.conn.commit()
        self.cursor.close()
        self.conn.close()
        print("Disconnected")

    def insert_data(self):
        self.connect_db()
        self.cursor.execute("INSERT INTO %s (name, age) VALUES ('%s',%s)" % (
            self.tbl_name2.text(),
            self.name.text(),
```

(continues on next page)

(continued from previous page)

```
            self.age.text()
            )
        )
        self.disconnect_db()
        QMessageBox.about(self, "Insert", "Data inserted successfully")

    def create_table(self):
        """ Create table in the database """

        self.connect_db()
        # optional: drop table if exists
        self.cursor.execute('DROP TABLE IF EXISTS %s' % self.tbl_name.text())
        self.cursor.execute('CREATE TABLE %s \
                (                    \
                  id    INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY, \
                  name  VARCHAR(30) NOT NULL,  \
                  age   int \
                )' % self.tbl_name.text()
        )
        print("Table created")
        QMessageBox.about(self, "Create", "Table created successfully")
        self.disconnect_db()

def main():
    app = QApplication(sys.argv)
    w = MainPage(title="PyQt5")
    sys.exit(app.exec_())


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

1.5.3 View data

```
# qt5_ex.py

import sys
from PyQt5.QtWidgets import (
        QApplication, QWidget, QLabel, QPushButton, QComboBox, QFrame, QMainWindow,
        QMessageBox, QLineEdit, QTextEdit

    )
from PyQt5.QtCore import pyqtSlot, QRect, Qt
from PyQt5.QtGui import QTextCursor

import MySQLdb as mq

class MainPage(QMainWindow):
    def __init__(self, title=" "):
        super().__init__()  # inherit init of QWidget
        self.title = title
        self.left = 250
        self.top = 50
        self.width = 600
        self.height = 250
        self.widget()

    def widget(self):
        # window setup
```

(continues on next page)

(continued from previous page)

```
        self.setWindowTitle(self.title)
        # self.setGeometry(self.left, self.top, self.width, self.height)
        ## use above line or below
        # self.resize(self.width, self.height) # resizable
        self.setFixedSize(self.width, self.height)  # fixed size
        self.move(self.left, self.top)

        # create frame for a set of checkbox
        self.frame1 = QFrame(self)
        self.frame1.setGeometry(QRect(40, 40, 250, 250))

        # create table
        self.tbl_name = QLineEdit(self.frame1)
        self.tbl_name.setPlaceholderText("Table name")
        self.tbl_name.setGeometry(0, 0, 100, 30)

        self.btn1 = QPushButton(self.frame1, text="Create Table")
        self.btn1.move(0, 50)
        self.btn1.clicked.connect(self.create_table)

        # insert data
        self.tbl_name2 = QLineEdit(self.frame1)
        self.tbl_name2.setPlaceholderText("Table name")
        self.tbl_name2.setGeometry(150, 0, 100, 30)

        self.name = QLineEdit(self.frame1)
        self.name.setPlaceholderText("Name")
        self.name.setGeometry(150, 50, 100, 30)

        self.age = QLineEdit(self.frame1)
        self.age.setPlaceholderText("Age")
        self.age.setGeometry(150, 100, 100, 30)

        self.btn2 = QPushButton(self.frame1, text="Insert")
        self.btn2.move(150, 150)
        self.btn2.clicked.connect(self.insert_data)


        self.frame2 = QFrame(self)
        self.frame2.setGeometry(QRect(350, 40, 250, 250))
        # show data
        self.txtbox1 = QTextEdit(self.frame2)
        self.txtbox1.setGeometry(0, 0, 200, 140)

        self.tbl_name3 = QLineEdit(self.frame2)
        self.tbl_name3.setText("writer")
        self.tbl_name3.setGeometry(0, 150, 100, 25)

        self.btn4 = QPushButton(self.frame2, text="Show data")
        self.btn4.move(110, 150)
        self.btn4.clicked.connect(self.show_data)
        self.show()

    def connect_db(self):
        try:
            self.conn = mq.connect(host='localhost', user='root', password='root', db='qtdb')
            self.cursor = self.conn.cursor()
            print("Connected")
        except mq.Error as err:
            print(err)
```

(continues on next page)

(continued from previous page)

```
    def disconnect_db(self):
        """ commit changes to database and close connection """
        self.conn.commit()
        self.cursor.close()
        self.conn.close()
        print("Disconnected")

    def insert_data(self):
        self.connect_db()
        self.cursor.execute("INSERT INTO %s (name, age) VALUES ('%s',%s)" % (
            self.tbl_name2.text(),
            self.name.text(),
            self.age.text()
            )
        )
        self.disconnect_db()
        QMessageBox.about(self, "Insert", "Data inserted successfully")

    def create_table(self):
        """ Create table in the database """

        self.connect_db()
        # optional: drop table if exists
        self.cursor.execute('DROP TABLE IF EXISTS %s' % self.tbl_name.text())
        self.cursor.execute('CREATE TABLE %s \
                (                   \
                  id    INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY, \
                  name  VARCHAR(30) NOT NULL,  \
                  age   int \
                )' % self.tbl_name.text()
        )
        print("Table created")
        QMessageBox.about(self, "Create", "Table created successfully")
        self.disconnect_db()

    def show_data(self):
        self.connect_db()
        self.cursor.execute("SELECT * FROM %s" % self.tbl_name3.text())
        data = self.cursor.fetchall()
        self.disconnect_db()

        self.txtcursor = QTextCursor(self.txtbox1.document())
        text = "{0:<2s} {1:<10s} {2:<3s}".format("Id", "Name", "Age")
        self.txtcursor.insertText(text + "\n")
        for d in data:
            text = "{0:<2d} {1:<10s} {2:<3d}".format(d[0], d[1], d[2])
            self.txtcursor.insertText(text + "\n")
            print(text)


def main():
    app = QApplication(sys.argv)
    w = MainPage(title="PyQt5")
    sys.exit(app.exec_())


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

Chapter 2

Tkinter (Under progress)

2.1 pack vs grid

- 'pack()' command automatically fit the items e.g. in the below code, the "Hello World" will be align to center if we increase the size of the window as shown in Fig. 2.1.

```
# pythongui.py

import tkinter as tk

tk.Label(text="Hello World").pack()
tk.mainloop()
```

```
$ python pythongui.py
```

Fig. 2.1: Hello World using pack()

- we can specify the exact location using 'grid command' i.e. in the below code the "Hello World" will be displayed at location '(0, 0)' as shown in Fig. 2.2.

```
# pythongui.py

import tkinter as tk

tk.Label(text="Hello World").grid(row=0, column=0)
tk.mainloop()
```

Fig. 2.2: "Hello world" using grid()

Warning: Note that if we have one element at (0,0) and other element at (0, 5) with 'no elements between them', the (0, 5) will be placed beside the (0, 0). In the other words, Tkinter does not keep empty spaces as shown in Fig. **??**.

```
# pythongui.py

import tkinter as tk

tk.Label(text="Hello World").grid(row=0, column=0)
tk.Label(text="Hello World").grid(row=0, column=5)
tk.mainloop()
```

2.2 Widgets

2.2.1 Button

- In the below code, the message will be changed after pressing the button 'Hi'.

```
# pythongui.py

import tkinter as tk

# display message on button click
def click_button():
    msg.configure(text="Hi Meher")

# obj of Tkinter
obj = tk.Tk()

# initial message will be changed after pressing the button
msg = tk.Label(text="Press button")
msg.grid(row=0, column=0)

# run click_button on click
btn_hi = tk.Button(text="Hi", command=click_button)
btn_hi.grid(row=1, column=0)

obj.mainloop()
```

2.2.2 Textbox

- In the below code, name is read from textbox and the message "Hello + name" is displayed on pressing the button.

Note: The 'obj' is used with all the Widgets, which is the correct way of coding.

```
# pythongui.py

import tkinter as tk

# display message on button click
def click_button():
    txt = "Hi {}".format(name.get())  # read message from textbox
    msg.configure(text=txt)
```

(continues on next page)

(continued from previous page)

```
# obj of Tkinter
obj = tk.Tk()

# initial message will be changed after pressing the button
msg = tk.Label(obj, text="Press button")
msg.grid(row=0, column=0)

# run click_button on click
btn_hi = tk.Button(obj, text="Hi", command=click_button)
btn_hi.grid(row=1, column=0)

# Textbox
name = tk.StringVar() # name is string variable
txtBox = tk.Entry(obj, width=12, textvariable=name)
txtBox.grid(row=1, column=1)

obj.mainloop()
```

- In the below code, 'state' and 'focus' commands are shown. The 'focus' command keeps the cursor on the 'textbox' whereas state='disabled' will disable the button after the first press.

```
# pythongui.py

import tkinter as tk

# display message on button click
def click_button():
    txt = "Hi {}".format(name.get())  # read message from textbox
    msg.configure(text=txt)
    btn_hi.configure(state="disabled")

# obj of Tkinter
obj = tk.Tk()

# initial message will be changed after pressing the button
msg = tk.Label(obj, text="Press button")
msg.grid(row=0, column=0)

# run click_button on click
btn_hi = tk.Button(obj, text="Hi", command=click_button)
btn_hi.grid(row=1, column=0)

# Textbox
name = tk.StringVar() # name is string variable
txtBox = tk.Entry(obj, width=12, textvariable=name)
txtBox.grid(row=1, column=1)
txtBox.focus()

obj.mainloop()
```

2.2.3 Combobox

Combobox is available in 'ttk'.

```
# pythongui.py

import tkinter as tk
from tkinter import ttk
```

(continues on next page)

(continued from previous page)

```
# display message on button click
def click_button():
    txt = "Hi {}".format(name.get())  # read message from textbox
    msg.configure(text=txt)

# obj of Tkinter
obj = tk.Tk()
obj.resizable(0, 0) # switch off resizable

# initial message will be changed after pressing the button
msg = tk.Label(obj, text="Press button")
msg.grid(row=0, column=0)

# run click_button on click
btn_hi = tk.Button(obj, text="Hi", command=click_button)
btn_hi.grid(row=1, column=0)

# Combobox
name = tk.StringVar() # name is string variable
cmbBox = ttk.Combobox(obj, width=12, textvariable=name)
cmbBox['values'] = ["Meher", "Krishna", "Patel"]
cmbBox.current(1) # choose second element
cmbBox.grid(row=1, column=1)

obj.mainloop()
```

2.2.4 Checkbox

```
# pythongui.py

import tkinter as tk
from tkinter import ttk

# obj of Tkinter
obj = tk.Tk()
obj.resizable(0, 0) # switch off resizable

# display message on button click
def click_button():
    lang_known = []
    if check1Var.get() == 1:
        lang_known.append("C++") # if selected then add to lang_known
    if check2Var.get() == 1:
        lang_known.append("Python")
    txt = "You know {0}".format(lang_known) #print as list
    msg.configure(text=txt)

check1Var = tk.IntVar()
check1 = tk.Checkbutton(obj, text="C++", variable=check1Var)
check1.deselect()  # do not check it
check1.grid(row=0, column=0)

check2Var = tk.IntVar()
check2 = tk.Checkbutton(obj, text="Python", variable=check2Var)
check2.select() # check this box
check2.grid(row=0, column=1)

# initial message will be changed after pressing the button
```

(continues on next page)

(continued from previous page)

```
msg = tk.Label(obj, text="Press button")
msg.grid(row=2, column=0)

# run click_button on click
btn_hi = tk.Button(obj, text="Hi", command=click_button)
btn_hi.grid(row=1, column=0)

obj.mainloop()
```

2.2.5 Radio button

```
# pythongui.py

import tkinter as tk
from tkinter import ttk

# obj of Tkinter
obj = tk.Tk()
obj.resizable(0, 0) # switch off resizable

# display message on button click
def click_button():
    lang_known = []
    if radioVar.get() == 0:
        lang_known.append("C++") # if selected then add to lang_known
    if radioVar.get() == 1:
        lang_known.append("Python")
    txt = "You know {0}".format(lang_known) #print as list
    msg.configure(text=txt)

radioVar = tk.IntVar()
radio1 = tk.Radiobutton(obj, text="C++", value=0, variable=radioVar)
radio1.deselect()  # do not radio it
radio1.grid(row=0, column=0)

radio2 = tk.Radiobutton(obj, text="Python", value=1, variable=radioVar)
radio2.select() # radio this box
radio2.grid(row=0, column=1)

# initial message will be changed after pressing the button
msg = tk.Label(obj, text="Press button")
msg.grid(row=2, column=0)

# run click_button on click
btn_submit = tk.Button(obj, text="Submit", command=click_button)
btn_submit.grid(row=1, column=0)

obj.mainloop()
```

2.2.6 Scrolled text

```
# pythongui.py

import tkinter as tk
from tkinter import ttk
from tkinter import scrolledtext
```

(continues on next page)

(continued from previous page)

```
# obj of Tkinter
obj = tk.Tk()
obj.resizable(0, 0) # switch off resizable

# initial message will be changed after pressing the button
msg = tk.Label(obj, text="Write message")
msg.grid(row=0, column=0)

width = 15
height = 3
scr = scrolledtext.ScrolledText(obj, width=width, height=height, wrap=tk.WORD)
scr.grid(row=1, column=0)

obj.mainloop()
```

2.2.7 Spinbox

```
# pythongui.py

import tkinter as tk
from tkinter import ttk


# obj of Tkinter
obj = tk.Tk()
obj.geometry("500x200+50+50")
obj.resizable(0, 0) # switch off resizable

# relief : GROOVE, FLAT, SUNKEN, RAISED, RIDGE
spinbox = tk.Spinbox(obj, from_=0, to=5, width=3, bd=5, relief=tk.GROOVE)
spinbox.grid()

obj.mainloop()
```

2.3 Frame

```
# pythongui.py

import tkinter as tk
from tkinter import ttk
from tkinter import scrolledtext

# obj of Tkinter
obj = tk.Tk()
obj.resizable(0, 0) # switch off resizable

frame1 = tk.LabelFrame(obj, text="Frame 1")
frame1.grid(row=0, column=0)
tk.Label(frame1, text="Lable1").grid(row=0, column=0)
tk.Label(frame1, text="Lable2").grid(row=0, column=1)
tk.Label(frame1, text="Lable3").grid(row=0, column=2)
tk.Button(frame1, text="Submit").grid(row=0, column=3)

frame2 = tk.LabelFrame(obj, text="Frame 2")
frame2.grid(row=0, column=1)
tk.Label(frame2, text="Lable1").grid(row=1, column=0)
```

(continues on next page)

(continued from previous page)

```
tk.Label(frame2, text="Lable2").grid(row=1, column=1)
tk.Label(frame2, text="Lable3").grid(row=1, column=2)
tk.Button(frame2, text="Submit").grid(row=1, column=3)

frame3 = tk.LabelFrame(obj, text="Frame 3")
frame3.grid(row=1, column=1)
tk.Label(frame3, text="Lable1").grid(row=2, column=0)
tk.Label(frame3, text="Lable2").grid(row=2, column=1)
tk.Label(frame3, text="Lable3").grid(row=2, column=2)
tk.Button(frame3, text="Submit").grid(row=2, column=3)

obj.mainloop()
```

2.4 Menu

```
# pythongui.py

import tkinter as tk
from tkinter import ttk
from tkinter import scrolledtext

# obj of Tkinter
obj = tk.Tk()
obj.geometry("500x200+50+50")
obj.resizable(0, 0) # switch off resizable

# create menu bar
menu_bar = tk.Menu(obj)
obj.config(menu=menu_bar)
# add item to menu bar
file_menu = tk.Menu(menu_bar, tearoff=0) # create file menu
# add file-menu to menu bar with label
menu_bar.add_cascade(label="File", menu=file_menu)
# add commands to File menu
file_menu.add_command(label="New") # add new to file menu
file_menu.add_command(label="Open")
file_menu.add_command(label="Save")
file_menu.add_separator()
file_menu.add_command(label="Exit")

help_menu = tk.Menu(menu_bar, tearoff=0) # create help menu
# add help-menu to menu bar with label
menu_bar.add_cascade(label="Help", menu=help_menu)
# add commands to File menu
help_menu.add_command(label="Help") # add new to help menu
help_menu.add_separator()
help_menu.add_command(label="Credits") # add new to help menu
help_menu.add_command(label="About")

obj.mainloop()
```

- Now add the functionality to File->Exit button,

```
# pythongui.py

import tkinter as tk
from tkinter import ttk
from tkinter import scrolledtext
```

(continues on next page)

(continued from previous page)

```
def exit_qk():
    obj.quit()
    obj.destroy()
    exit()


# obj of Tkinter
obj = tk.Tk()
obj.geometry("500x200+50+50")
obj.resizable(0, 0) # switch off resizable


# create menu bar
menu_bar = tk.Menu(obj)
obj.config(menu=menu_bar)
# add item to menu bar
file_menu = tk.Menu(menu_bar, tearoff=0) # create file menu
# add file-menu to menu bar with label
menu_bar.add_cascade(label="File", menu=file_menu)
# add commands to File menu
file_menu.add_command(label="New") # add new to file menu
file_menu.add_command(label="Open")
file_menu.add_command(label="Save")
file_menu.add_separator()
file_menu.add_command(label="Exit", command=exit_qk)

help_menu = tk.Menu(menu_bar, tearoff=0) # create help menu
# add help-menu to menu bar with label
menu_bar.add_cascade(label="Help", menu=help_menu)
# add commands to File menu
help_menu.add_command(label="Help") # add new to help menu
help_menu.add_separator()
help_menu.add_command(label="Credits") # add new to help menu
help_menu.add_command(label="About")

obj.mainloop()
```

2.5 Message box

Message box is added for “Exit” and “About” options.

```
# pythongui.py

import tkinter as tk
from tkinter import ttk
from tkinter import messagebox


def exit_qk():
    ans = tk.messagebox.askyesnocancel("Do you want to quit")
    print(ans) # True/False/None
    if ans:
        obj.quit()
        obj.destroy()
        exit()
```

(continues on next page)

(continued from previous page)

```
def about_box(): # info, warning, error, askyesnocancel
    # tk.messagebox.showinfo("Help-Title", "This is help message")
    # tk.messagebox.showwarning("Help-Title", "This is help message")
    tk.messagebox.showerror("Help-Title", "This is help message")

# obj of Tkinter
obj = tk.Tk()
obj.geometry("500x200+50+50")
obj.resizable(0, 0) # switch off resizable


# create menu bar
menu_bar = tk.Menu(obj)
obj.config(menu=menu_bar)
# add item to menu bar
file_menu = tk.Menu(menu_bar, tearoff=0) # create file menu
# add file-menu to menu bar with label
menu_bar.add_cascade(label="File", menu=file_menu)
# add commands to File menu
file_menu.add_command(label="New") # add new to file menu
file_menu.add_command(label="Open")
file_menu.add_command(label="Save")
file_menu.add_separator()
file_menu.add_command(label="Exit", command=exit_qk)

help_menu = tk.Menu(menu_bar, tearoff=0) # create help menu
# add help-menu to menu bar with label
menu_bar.add_cascade(label="Help", menu=help_menu)
# add commands to File menu
help_menu.add_command(label="Help") # add new to help menu
help_menu.add_separator()
help_menu.add_command(label="Credits") # add new to help menu
help_menu.add_command(label="About", command=about_box)

obj.mainloop()
```

2.6 Tabs

```
# pythongui.py

import tkinter as tk
from tkinter import ttk
from tkinter import scrolledtext


def exit_qk():
    obj.quit()
    obj.destroy()
    exit()


# obj of Tkinter
obj = tk.Tk()
obj.geometry("500x200+50+50")
obj.resizable(0, 0) # switch off resizable

tabs = ttk.Notebook(obj) # create tab object
# tab 1
```

(continues on next page)

(continued from previous page)

```
tab1 = tk.Frame(tabs)
tabs.add(tab1, text="Tab 1")
# add item to tab 1
label1 = tk.Label(tab1, text="Hello Tab1")
label1.grid()

# tab 2
tab2 = tk.Frame(tabs)
tabs.add(tab2, text="Tab 2")
# add item to tab 2
label2 = tk.Label(tab2, text="Hello Tab1")
label2.grid()

tabs.grid()

obj.mainloop()
```

2.7 OOPs

2.7.1 Hello World with OOPs

- Inherit from 'Frame'

```
# tkoops.py

import tkinter as tk

# inherit from tk.Frame
class HomePage(tk.Frame):
    def __init__(self):
        tk.Frame.__init__(self)
        self.grid()
        self.createWidgets() # call user defined createWidgets

    def createWidgets(self):
        # create Label
        self.myLabel = tk.Label(self, text="Hello World! This is Home page")
        # set grid parameters
        self.myLabel.grid(
                row=0, column=0,
                rowspan=3, columnspan=2,
                padx=10, pady=10,
                ipadx=3, ipady=3, # internal pad

                # use this or below
                # sticky="nsew"  # north south east west
                sticky=tk.EW
            )

def main():
    app = HomePage()
    app.master.title("Plot graphs")
    app.mainloop()

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

- Inherit for Tk

```
# tkoops.py

import tkinter as tk

# inherit from tk.Frame
class HomePage(tk.Tk):
    def __init__(self):
        super().__init__()  # inherit parent class init

        # set size of window
        self.geometry("500x400+10+10") # width, height, left margin, top margin

        # frame
        self.frame = tk.Frame(self) # create frame
        self.frame.master.title("Tkinter") # title of frame
        self.frame.grid(padx=20, pady=20) # add some margin from top and bottom

        self.createWidgets() # call user defined createWidgets

    def createWidgets(self):
        # create Label
        self.myLabel = tk.Label(self.frame, text="Hello World! This is Home page")
        # set grid parameters
        self.myLabel.grid(row=0, column=0)

def main():
    app = HomePage()
    app.mainloop()

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

2.7.2 Seperate window setup and HomePage

```
# tkoops.py

import tkinter as tk

# inherit from tk.Frame
class WindowSetup(tk.Tk):
    def __init__(self):
        super().__init__()  # inherit parent class init
        # set size of window
        self.geometry("500x400+10+10") # width, height, left margin, top margin

class HomePage(WindowSetup):
    def __init__(self, title=" "):
        super().__init__()
        # frame
        self.frame = tk.Frame(self) # create frame
        self.frame.master.title(title) # title of frame
        self.frame.grid(padx=20, pady=20) # add some margin from top and bottom

        self.createWidgets() # call user defined createWidgets

    def createWidgets(self):
        # create Label
        self.myLabel = tk.Label(self.frame, text="Hello World! This is Home page")
        # set grid parameters
        self.myLabel.grid(row=0, column=0)
```

(continues on next page)

(continued from previous page)

```
def main():
    app = HomePage("Home Page")
    app.mainloop()

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

- Modfied with frame created in the Tk

```
# tkoops.py

import tkinter as tk

# inherit from tk.Frame
class WindowSetup(tk.Tk):
    def __init__(self, title):
        super().__init__()  # inherit parent class init
        # set size of window
        self.geometry("500x400+10+10") # width, height, left margin, top margin

        # frame
        self.frame = tk.Frame(self) # create frame
        self.frame.master.title(title) # title of frame
        self.frame.grid(padx=20, pady=20) # add some margin from top and bottom


class HomePage(WindowSetup):
    def __init__(self, title=" "):
        super().__init__(title)
        self.createWidgets() # function call : createWidgets

    def createWidgets(self):
        # create Label
        self.myLabel = tk.Label(self.frame, text="Hello World! This is Home page")
        # set grid parameters
        self.myLabel.grid(row=0, column=0)

def main():
    app = HomePage("Home Page")
    app.mainloop()

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

2.7.3 Matplotlib

```
# tkoops.py

import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.backends.tkagg as tkagg
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

import numpy as np

import tkinter as tk
from tkinter import ttk # ttk changes the look of widgets
```

(continues on next page)

(continued from previous page)

```
# inherit from tk.Frame
class HomePage(tk.Frame):
    def __init__(self):
        tk.Frame.__init__(self)
        self.grid()
        self.createWidgets() # call user defined createWidgets

    def createWidgets(self):
        # create Label
        self.myLabel = ttk.Label(self, text="Hello World! This is Home page")
        # set grid parameters
        self.myLabel.grid(
                row=0, column=0,
                rowspan=3, columnspan=2,
                padx=10, pady=10,
                ipadx=3, ipady=3, # internal pad

                # use this or below
                # sticky="nsew"  # north south east west
                # sticky=tk.E
            )

        x = np.random.rand(100)
        fig = mpl.figure.Figure(figsize=(5,5))
        sf = fig.add_subplot(111)
        sf.plot(x)
        canvas = FigureCanvasTkAgg(fig, self)
        canvas.get_tk_widget().grid(row=5, column=1)
        canvas.show()

def main():
    app = HomePage()
    app.master.title("Plot graphs")
    app.mainloop()

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