PYQT6 Course

7. Event Handling in PyQt6

We have already learned about some basic components in PyQt6, now let's talk about event handling in PyQt. Event handling is an important mechanism in every GUI application. The application should not only recognize the event, but must take the respective action to serve the event, too. In PyQt, the event handling mechanism is also known as signals and slots. An event can be in the form of clicking on a widget, or pressing the Enter key, or selecting an option from a radio button, checkbox, and so on. Every widget emits a signal when any event is applied on it and, that signal needs to be connected to a method, also known as a slot, there are different built in signals in pyqt for every widget, for example for button we have *clicked* signal, for menu items we have *triggered* signal and for lineedit we have *returnPressed* signal.

In this code we have created a QHBoxLayout, QPushButton and also a QLabel. Also we have added our QPushButton and QLabel in the QHBoxLayout. Now we have already learned about these concepts and there is no problem.

```
hbox = QHBoxLayout()
btn = QPushButton("Change Text")
self.label = QLabel("Default Text")

hbox.addWidget(btn)
hbox.addWidget(self.label)
```

This is the important point of this lesson, you can see that we have used the clicked signal of the QPushButton, and we have connected that signal to the *clicked_btn()* slot, now slots are just regular python functions.

```
btn.clicked.connect(self.clicked_btn)
```

This is our slot or method, in this method we want whenever we click on the button, we change the text, color and the font of the QLabel.

```
def clicked_btn(self):
    self.label.setText("Another Text")
    self.label.setFont(QFont("Times", 15))
    self.label.setStyleSheet('color:red')
```