Python -
GUI mit PyQt

Jan Popko
Python Grundkurs

Business Trends Academy (BTA) GmbH
Nestorstraße 36
D-10709 Berlin
Tel.: +49 (0) 30 894 087 57
Fax: +49 (0) 30 895 429 94
Geschäftsführer:
Gabriele Fleischmann-Hahn
Maxi-Marlen Fleischmann
Hauptsitz des Unternehmens:
Nestorstraße 36, D-10709 Berlin
HRB 115251 B / Amtsgericht Berlin Charlottenburg
Steuer-Nr. 27/248/31179

Benutzeroberflächen mit Qt

- PyQt ist eine Bibliothek mit der GUI's entwickelt werden können
- enthält über 620 Klassen und 6000 Funktionen und Methoden
- wurde von der Firma Riverbank entwickelt
- für alle gängigen Plattformen verfügbar

Business Trends Academy (BTA) GmbH
Nestorstraße 36
D-10709 Berlin
Tel.: +49 (0) 30 894 087 57
Fax: +49 (0) 30 895 429 94
Geschäftsführer:
Gabriele Fleischmann-Hahn
Maxi-Marlen Fleischmann
Hauptsitz des Unternehmens:
Nestorstraße 36, D-10709 Berlin
HRB 115251 B / Amtsgericht Berlin Charlottenburg
Steuer-Nr. 27/248/31179

Benutzeroberflächen mit Qt

Installation:

in die Konsole: **pip3 install pyqt5** eingeben und Enter drücken

Wenn Installation abgeschlossen ist, testen ob alles funktioniert:

interaktive Pythonshell starten

```
>>>from PyQt5 import QtCore

>>>dir(QtCore)
```

Eine Liste mit Namen von Konstanten, Klassen und Funktionen erscheint.

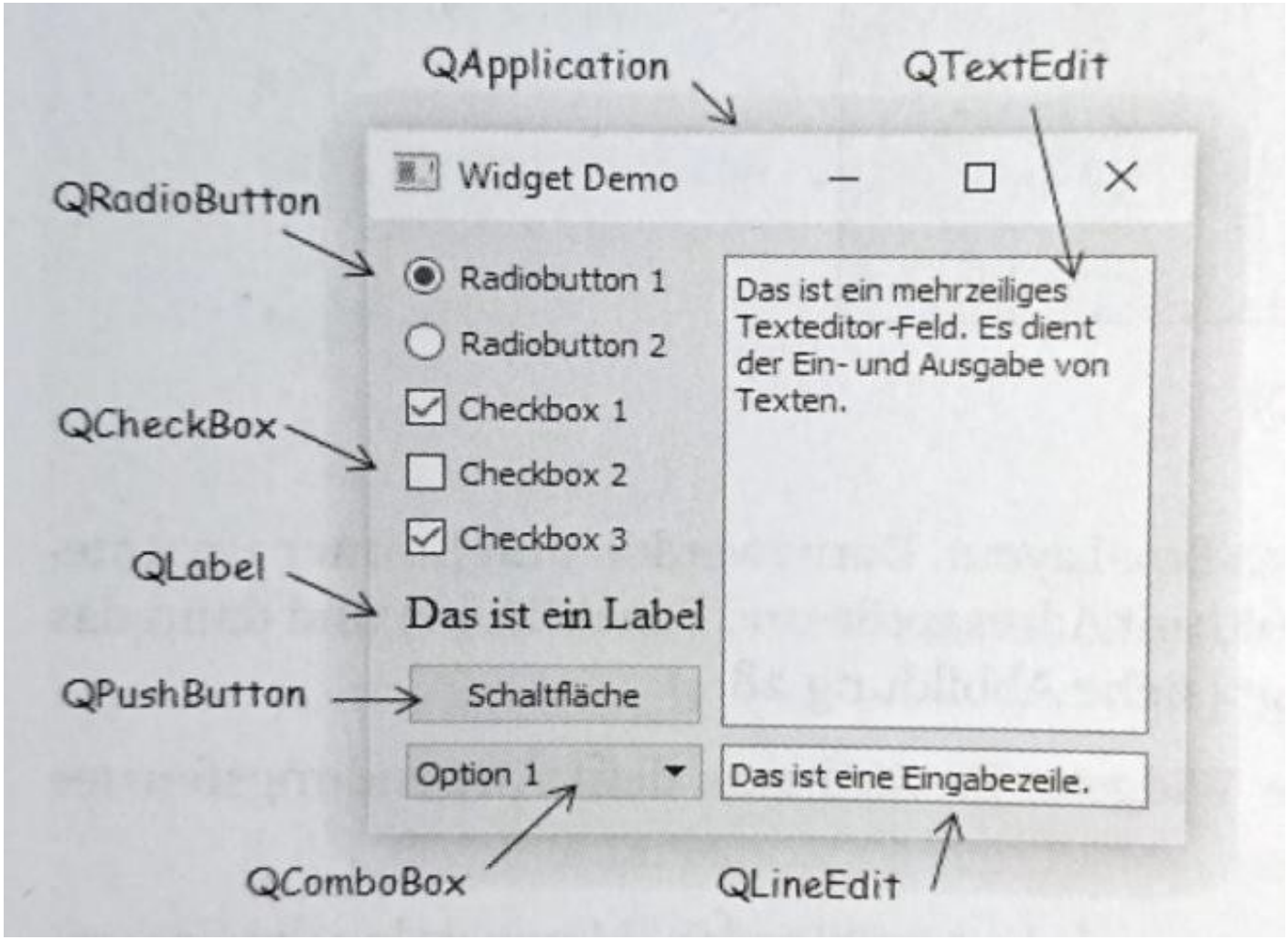
Benutzeroberflächen mit Qt

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget
from PyQt5.QtGui import QIcon

class Window(QWidget): # Klasse Window erbt von QWidget, hat keine Eltern
  def __init__(self):
    super().__init__()
    self.resize(250, 100) # Fenstergröße wird festgelegt
    self.setWindowTitle('Fenster') # Fenstertitel
    self.setWindowIcon(QIcon("welt.png")) # Icon in die linke obere Ecke
    self.show() # Fenster wird sichtbar gemacht

app = QApplication(sys.argv) # für jede PyQt-App muss ein Objekt der
                                    Klasse QApplication erzeugt werden
w = Window() # Ein Objekt der Klasse Window wird erzeugt und sofort angezeigt
sys.exit(app.exec_()) # hier wird die GUI aktiviert und in die Hauptschleife
        gegeben, die Ausführung wird beendet, wenn das Fenster geschlossen wird
```

Interaktive Widgets

QPushButton	Schaltfläche, mit setText() kann die Beschriftung dynamisch gesetzt werden
QCheckBox	besteht aus einem Label und einem Kästchen zur Selektion, mit isChecked() kann der Zustand abgefragt werden (True bedeutet selektiert)
QComboBox	Auswahlliste von Items
QLabel	ist eine Fläche auf der Text angezeigt werden kann, mit setText() kann dieser Text gesetzt werden
QRadioButton	besteht aus einem Label und einem Kreis zur Selektion, Ein Radiobuttons gehören immer zu einer Gruppe (QbuttonGroup)
QButtonGroup	Gruppe von Radiobuttons, checkedButton() liefert die angeklickte Schaltfläche
QLineEdit	Eingabezeile, Methode text() liefert den Inhalt, setText() wird der Text neu gesetzt, append() hängt neuen Text an
QTextEdit	mehrzeiliges Eingabefeld, gleiche Methoden wie QLineEdit

Interaktive Widgets



Buttons horizontal

```
import sys
from PyQt5.QtWidgets import QApplication, QHBoxLayout, QWidget, QPushButton

class Window(QWidget):
  def __init__(self):
     super().__init__()
     self.resize(200,200)
     self.setWindowTitle('Button horizontal')

     #Widgets
     self.cancelButton = QPushButton("Cancel")
     self.okButton = QPushButton("OK")

     #horizontales Layout
     hBox = QHBoxLayout()
     hBox.addWidget(self.okButton)
     hBox.addWidget(self.cancelButton)
     self.setLayout(hBox)
     self.show()

#Hauptprogramm
app = QApplication(sys.argv)
w = Window()
sys.exit(app.exec_())
```

Buttons vertikal

```
import sys
from PyQt5.QtWidgets import QApplication, QVBoxLayout, QWidget, QPushButton

class Window(QWidget):
  def __init__(self):
     super().__init__()
     self.resize(200,200)
     self.setWindowTitle('Button vertikal')

     #Widgets
     self.cancelButton = QPushButton("Cancel")
     self.okButton = QPushButton("OK")

     #vertikales Layout
     vBox = QVBoxLayout()
     vBox.addWidget(self.okButton)
     vBox.addWidget(self.cancelButton)
     self.setLayout(vBox)
     self.show()

#Hauptprogramm
app = QApplication(sys.argv)
w = Window()
sys.exit(app.exec_())
```

horizontal/vertikal kombiniert

```
class Window(QWidget):
  def __init__(self):
     super().__init__()
     self.resize(200,200)
     self.setWindowTitle('Button kombiniert')

     #Widgets
     self.cancelButton = QPushButton("Cancel")
     self.okButton = QPushButton("OK")
     self.cancelButton2 = QPushButton("Cancel2")
     self.okButton2 = QpushButton("OK2")

     #horizontales Layout
     hBox = QHBoxLayout()
     hBox.addWidget(self.okButton)
     hBox.addWidget(self.cancelButton)

     #vertikales Layout
     vBox = QVBoxLayout()
     vBox.addWidget(self.okButton2)
     vBox.addWidget(self.cancelButton2)
     vBox.addLayout(hBox)
     self.setLayout(vBox)
```

Grid-Layout

```
self.b1 = QPushButton("Cancel")
self.b2 = QPushButton("OK")
self.b3 = QPushButton("Cancel")
self.b4 = QPushButton("OK")
self.b5 = QPushButton("Cancel")
self.b6 = QPushButton("OK")

#Grid Layout
grid = QGridLayout()
grid.addWidget(self.b1, 0, 0)
grid.addWidget(self.b2, 0, 1)
grid.addWidget(self.b3, 0, 2)

grid.addWidget(self.b4, 1, 0)
grid.addWidget(self.b5, 1, 1)
grid.addWidget(self.b6, 1, 2)

self.setLayout(grid)
```

Radiobutton

```
#Widgets
self.rb1 = QRadioButton("Option1")
self.rb1.setChecked(True)
self.rb2 = QRadioButton("Option2")
self.radioButtons = QButtonGroup()
self.radioButtons.addButton(self.rb1)
self.radioButtons.addButton(self.rb2)


#Layout
hBox = QHBoxLayout()
hBox.addWidget(self.rb1)
hBox.addWidget(self.rb2)
self.setLayout(hBox)
```

Checkbox

```
#Widgets
self.cb1 = QCheckBox("Option1")
self.cb2 = QCheckBox("Option2")
self.cb3 = QCheckBox("Option3")

#Layout
vBox = QVBoxLayout()
vBox.addWidget(self.cb1)
vBox.addWidget(self.cb2)
vBox.addWidget(self.cb3)
self.setLayout(vBox)
```

Combobox

```
#Widgets
self.combo = QComboBox()
self.combo.addItem("Option1")
self.combo.addItem("Option2")
self.combo.addItem("Option3")

#Layout
vBox = QVBoxLayout()
vBox.addWidget(self.combo)
self.setLayout(vBox)
```

LineEdit, TextEdit, Label

```
#Widgets
self.labelLine = QLabel("Text Line")
self.labelText = QLabel("Text Text")
self.line = QLineEdit()
self.text = QTextEdit()

#Layout
vBox = QVBoxLayout()
vBox.addWidget(self.labelLine)
vBox.addWidget(self.line)
vBox.addWidget(self.labelText)
vBox.addWidget(self.text)
self.setLayout(vBox)
```

Stylesheets

Globales Stylesheet:

```
STYLE = """
    QComboBox{
    color: white;
    border: 1px solid gray;
    border-radius: 3px;
    padding: 1px 18px 1px 3px;
    background: black;
    }
"""

app = QApplication(sys.argv)
app.setStyleSheet(STYLE)
```

http://doc.qt.io/qt-5/stylesheet-examples.html#style-sheet-usage)

Stylesheets

Stylesheet für einzelne Widgets.
Es wird die Methode setStyleSheet() verwendet:

```
self.searchButton = QPushButton("Suchen")
self.searchButton.setStyleSheet("background: red")
```

http://doc.qt.io/qt-5/stylesheet-examples.html#style-sheet-usage)

Signale

Signal	… wird gesendet, wenn …
activated[str]	… eine Option der Liste ausgewählt wurde, liefert auch den Text (QcomboBox)
ButtonClicked	… in einer Gruppe von Radiobuttons einer angeklickt wurde (QButtonGroup)
clicked	… das Widget angeklickt wurde (QButtonGroup, QCheckbox, QPushButton, QRadioButton)
textChanged	… in einem Eingabefeld der Text geändert wurde (QLineEdit, QTextEdit)

Jedes Signal kann mit einer Methode oder Funktion verbunden werden!

Diese Funktionen werden in Qt Slots genannt.

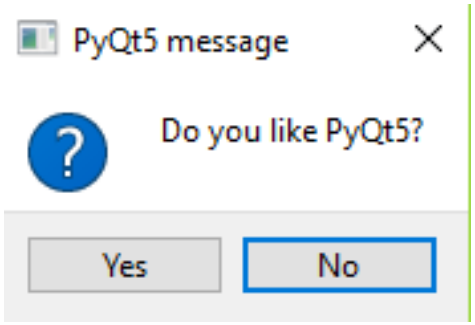
Signale und Slots

```
    Clicked:

    self.textfeld = QLineEdit()
    self.ergebnis = QLabel()
    self.senden = QPushButton("Senden")
    #Signal clicked des Button senden wird mit Slot(Methode)
    dosomething verknüpft
    self.senden.clicked.connect(self.dosomething)


def dosomething(self):
    text = self.textfeld.text()
    if text == "w":
        self.ergebnis.setText("weiblich")
    else:
        self.ergebnis.setText("nicht weiblich")
```

Signale und Slots

```
    activated[str]:
    self.label = QLabel()
    self.combo = QComboBox()
    self.combo.addItem("Option1")
    self.combo.addItem("Option2")
    self.combo.addItem("Option3")
    #Signal activated[str] der Combobox combo wird mit
    Slot(Methode) change verknüpft
    self.combo.activated[str].connect(self.change)


def change(self, str):
    self.label.setText(str)
```

Messageboxen




```
def showdialog(self):
    buttonReply = QMessageBox.question(self, 'PyQt5 message',
    "Do you like PyQt5?", QMessageBox.Yes | QMessageBox.No,
    QMessageBox.No)

if buttonReply == QMessageBox.Yes:
    self.ergebnis.setText("Yes")
else:
    self.ergebnis.setText("No")
```