



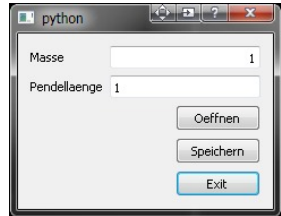
TECHNISCHE
UNIVERSITÄT
DRESDEN

GUI PROGRAMMIERUNG MIT PYQT4 TEIL II

Sebastian Voigt, TU Dresden

Dresden, Wintersemester 2013/14

- Widget = rechteckiger Zeichenbereich auf dem Bildschirm
 - viele Widgets sind als Steuerelemente bekannt (Buttons etc.)
- Layout = passt Größe und Anordnung der Widgets dynamisch an
- Arten von Layouts: horizontal, vertikal, grid
- Widgets und Layouts stehen in Eltern-Kind-Beziehungen zueinander
- Anwendungsfenster in Form eines Dialogs (Nutzung der Klasse QDialog)



aber: viele Elemente graphischer Benutzerschnittstellen stehen bei Verwendung von QDialog nicht zur Verfügung

Für „echte“ Anwendungen wird QMainWindow verwendet:

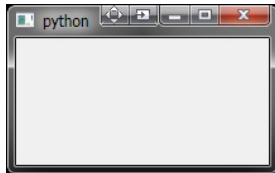
```
# -*- coding: utf-8 -*-
```

```
import PyQt4.QtGui as QtGui
import PyQt4.QtCore as QtCore
import sys
```

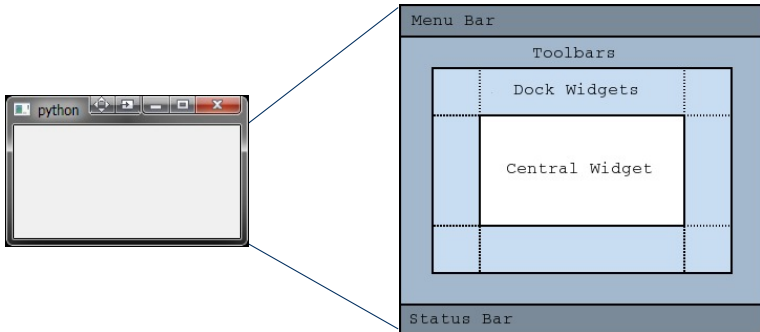
```
class Gui(QtGui.QMainWindow):
```

```
    def __init__(self):
        QtGui.QMainWindow.__init__(self)
```

```
app = QtGui.QApplication(sys.argv)
gui = Gui()
gui.show()
app.exec_()
```



QMainWindow sieht bereits Platzhalter bzw. definierte Bereiche innerhalb des Anwendungsfensters für Menüs, Toolbars usw. vor:



Quelle:

<http://qt-project.org/doc/qt-4.8/QMainWindow.html>

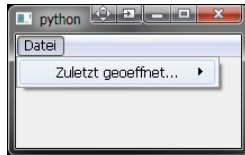
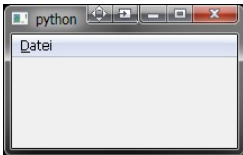
- Menüs können direkt über die Menu Bar von QMainWindow angelegt werden:

```
self.menuFile = self.menuBar().addMenu('&Datei')
```

- Menüs können auch ineinander geschachtelt werden:

```
self.menuRecent = self.menuFile.addMenu(  
    'Zuletzt geöffnet...')
```

- Menüs definieren nur die Struktur, keine klickbaren Einträge
- das „&“ im Namensstring definiert Shortcut (Alt+D)

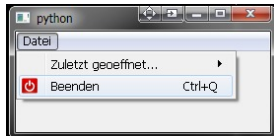


- sind Instanzen der Klasse QAction
- großer Vorteil: können in verschiedener Gestalt in Erscheinung treten
- Action zum Beenden des Programms:

```
self.actExit = QtGui.QAction(self)
self.actExit.setText('Beenden')
self.actExit.setIcon(QtGui.QIcon('../data/exit.png'))
self.actExit.setShortcut('Ctrl+Q')
```

- Actions stellen eine abstrakte Interaktionsmöglichkeit mit den Benutzer dar
- taucht nirgendwo auf, also noch zum Menü hinzufügen:

```
self.menuFile.addAction(self.actExit)
```



- Wenn Actions „gettriggered“ werden, kann darauf mit einem Funktionsaufruf reagiert werden:

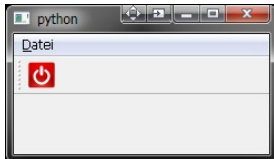
```
self.actExit.triggered.connect(self.close)
```

- Actions können aber ebenso in Toolbars auftauchen. Dazu legen wir eine neue Toolbar an:

```
self.toolbar = QtGui.QToolBar('Datei')  
self.toolbar.setIconSize(QtCore.QSize(24,24))  
self.addToolBar(self.toolbar)
```

- und fügen die Action hinzu:

```
self.toolbar.addAction(self.actExit)
```



- Actions können auch im Kontextmenü auftauchen:

```
self.cw = QtGui.QWidget()
self.setCentralWidget(self.cw)

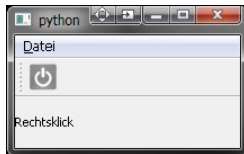
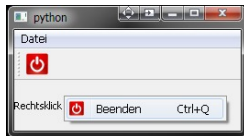
self.vBox = QtGui.QVBoxLayout(self.cw)

self.label = QtGui.QLabel('Rechtsklick')
self.label.setContextMenuPolicy(
    QtCore.Qt.ActionsContextMenu)
self.label.addAction(self.actExit)

self.vBox.addWidget(self.label)
```

- Actions können deaktiviert werden:

```
self.actExit.setDisabled(True)
```



- Kommunikationsmechanismus innerhalb der Anwendung:
Widgets emittieren Signale und (beliebig viele) Widgets oder Funktionen reagieren darauf
- Bsp. Ausgabe des Wertes eines Sliders:

```
self.slider = QtGui.QSlider(self)
self.slider.setMinimum(-10)
self.slider.setMaximum(10)
self.slider.setOrientation(QtCore.Qt.Horizontal)

self.connect(self.slider, QtCore.SIGNAL('valueChanged(int)'),
             self.label,   QtCore.SLOT('setNum(int)'),

self.connect(self.slider, QtCore.SIGNAL('valueChanged(int)'),
             self.printValue)

self.vBox.addWidget(self.slider)
```

ParameterMaske
4 x EingabeZahl

StartwertSlider
StartwertSlider

