TECHNISCHE UNIVERSITÄT DRESDEN

PYTHONKURS FÜR INGENIEUR:INNEN

Einführung GUI Programmierung mit PyQT

Folien: Sebastian Voigt, Carsten Knoll, TU Dresden

Dresden, WiSe 2020/21

„GUI“ - Was ist das?

- GUI = Graphical User Interface
- Benutzerfreundlichkeit, insbesondere wenn Programm für Dritte entwickelt wird
- Gebrauchstauglichkeit (ISO 9241-11)
- Vielfältige Möglichkeiten der Interaktion (Maus, Gesten, etc)
- Kontrolle paralleler Programmabläufe durch den Nutzer
- GUI Toolkits für Python:
  - Tkinter
  - PyGtk
  - wxPython
  - PyQt4 (alt)
  - PyQt5 (aktuell)
  - PySide
  - ...
- Ziel: Kennenlernen der Basiselemente und einfache graphische Oberfläche zur Simulationssteuerung der Laufkatze

Hintergrund und Installation

- Qt (engl. wie „cute“, dt. oft „Kuh-Tee“) ist umfangreiche GUI-Bibliothek (C++)
- Es gibt eine automatisch erzeugte Python-Anbindung („Bindings“) namens PyQt
  - $\Rightarrow$ Es gibt keine separate Referenz-Doku für PyQt
    $\rightarrow$ C++-Doku ist maßgeblich
  - $\Rightarrow$ Namenskonvention (z.B. CamelCase für Methoden) ist für Python eher unüblich

Grundlagen

- Zentraler Begriff in der GUI-Programmierung: **Widget**
  - Wortherkunft: **Wi**ndow + Ga**dget** „Fenster-Spielerei“
  - Ursprüngliche Bedeutung: Steuerelement eines Fensters
    z.B.: Button, Bildlaufleiste, Eingabefeld, ...
  - Zusammenfassungen von Widgets ergeben wieder ein Widget
    Bsp: Dialogfeld mit zwei Buttons „OK“ und „Abbrechen“
- Wesentlicher Bestandteil der GUI-Programmierung:
  - Anordnung und Interaktion von Widgets bestimmen

Anordnung (1)

- Fenster mit Textelement

Listing: example-code/gui-example1.py

```
import PyQt5.QtWidgets as QtWidgets

app = QtWidgets.QApplication([])
dialog = QtWidgets.QDialog()
mass_label = QtWidgets.QLabel('Masse', dialog)
dialog.exec()
```

- Button und Eingabefeld hinzufügen

Listing: example-code/gui-example2.py (7-10)

```
# ... wie gui-example1.py + zwei neue Widgets
mass_edit = QtWidgets.QLineEdit('2.5', dialog)
exit_button = QtWidgets.QPushButton('Exit', dialog)
dialog.exec()
```

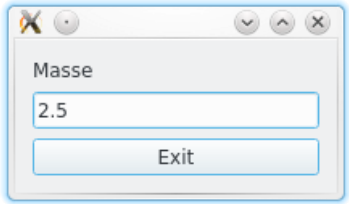
- Problem: Positionierung übereinander → Verdeckung
- Lösung: Layout

Anordnung (2)

- Layouts passen Größe und Anordnung der Widgets automatisch an
- Typen: vertical, horizontal, grid
- Hier: vertikales `QVBoxLayout` (Elemente vertikal untereinander „stapeln“)

Listing: example-code/gui-example3.py (11-16)

```
layout.addWidget(mass_label)
layout.addWidget(mass_edit)
layout.addWidget(exit_button)
dialog.setLayout(layout)
dialog.exec()
```



Anordnung (3)

- Zweites `Label` und `LineEdit` anlegen

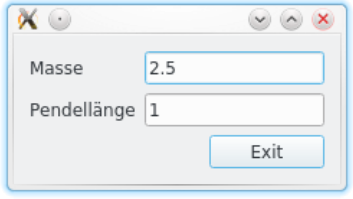
Listing: example-code/gui-example4.py (12-13)

```
len_label = QtWidgets.QLabel('Pendellänge', dialog)
len_edit = QtWidgets.QLineEdit('1', dialog)
```

- Gridlayout für Rasteranordnung:

Listing: example-code/gui-example4.py (15-20)

```
layout = QtWidgets.QGridLayout()
layout.addWidget(mass_label, 0, 0)   # Widget, Zeile, Spalte
layout.addWidget(mass_edit, 0, 1)
layout.addWidget(len_label, 1, 0)
layout.addWidget(len_edit, 1, 1)
layout.addWidget(exit_button, 2, 1, QtCore.Qt.AlignRight)
```



- Letzte Zeile: Ausrichtung innerhalb des Layouts anpassen
- Weitere Infos: `https://doc.qt.io/qt-5/qgridlayout.html`
  - Achtung: C++-Doku → Polymorphie beachten
    (d. h. Methodennamen treten mehrfach mit unterschiedlichen Signaturen auf)

Daten-Verarbeitung und Buttons

- Wert aus `LineEdit` holen und wieder setzen

Listing: example-code/gui-example4.py (22-29)

```
dialog.exec() # Dialog zum ersten Mal ausführen

m = float(mass_edit.text())
# Text auf der Kommandozeile ausgeben:
print("Im Dialog wurde die Masse", m, "eingegeben.")

mass_edit.setText(str(m*2))
dialog.exec()  # Dialog zum zweiten Mal ausführen
```

- Eingabefeld anpassen: Ausrichtung und Eingabefilter:
  → Mit dem `QDoubleValidator` sind nur Zahlen als Eingabe erlaubt

```
mass_edit.setAlignment(QtCore.Qt.AlignRight)
mass_edit.setValidator(QtGui.QDoubleValidator(mass_edit))
```

- Button mit **Funktionsobjekt** verknüpfen (`dialog.close` ohne Klammern):

```
exit_button.clicked.connect(dialog.close)
```

Buttons (2)

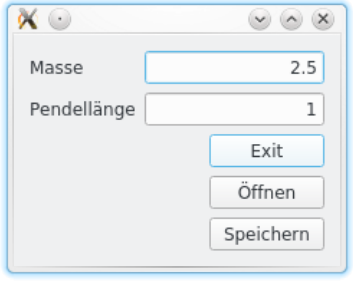
- Buttons mit eigener Funktion verknüpfen (am Beispiel Laden/Öffnen einer Datei)

Listing: example-code/gui-example5.py (25-31)

```
def openfile():
    path, type_filter = QtWidgets.QFileDialog.getOpenFileName()
    print(path, type_filter)

def savefile():
    path, type_filter = QtWidgets.QFileDialog.getSaveFileName()
    print(path, type_filter)
```

Listing: example-code/gui-example5.py (33-44)

```
open_button = QtWidgets.QPushButton('Öffnen', dialog)
save_button = QtWidgets.QPushButton('Speichern', dialog)

layout.addWidget(open_button, 3, 1, QtCore.Qt.AlignRight)
layout.addWidget(save_button, 4, 1, QtCore.Qt.AlignRight)

open_button.clicked.connect(openfile)
save_button.clicked.connect(savefile)
```

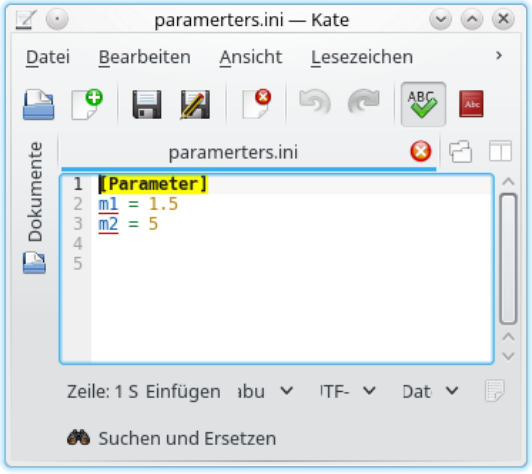


Konfigurationsdateien erstellen

- Dafür existiert in der Python Standardbibliothek das Modul `configparser`

Listing: example-code/cfgparser-example1.py

```
import configparser

c = configparser.SafeConfigParser()
c.add_section('Parameter')
c.set('Parameter', 'm1', '1.5')
c.set('Parameter', 'm2', '5')

with open('paramerters.ini', 'w') as fid:
    c.write(fid)
```



- Es können beliebige Sections (erscheinen in eckigen Klammern in der Datei) und der Schlüssel-Wert-Paare angelegt werden
- Achtung: Es müssen Strings geschrieben werden!

Konfigurationsdateien lesen

- Konfigurationsdateien lesen: genau so einfach
- Werte: mittels `get()`-Methode der `SafeConfigParser`-Klasse

Listing: example-code/cfgparser-example2.py

```
import configparser

c = configparser.SafeConfigParser()
c.read('paramerters.ini')
print(c.sections())

m1 = c.get('Parameter', 'm1')
m2 = c.get('Parameter', 'm2')

print(m1)
print(m2)
```

- Achtung: Auch hier werden die Werte als Strings gelesen und müssen nach Bedarf konvertiert werden!

Links

Links:

- PyQt5 Übersicht:
  http://pyqt.sourceforge.net/Docs/PyQt5/index.html
- PyQt5 Module:
  http://pyqt.sourceforge.net/Docs/PyQt5/modules.html
- PyQt5 Widgets-Modul (das wichtigste Modul):
  http://pyqt.sourceforge.net/Docs/PyQt5/QtWidgets.html
  verweist auf
- Referenz des QtWidgets-Moduls (C++):
  https://doc.qt.io/qt-5/qtwidgets-module.html
- Verschiedene kurze Qt5-Tutorials:
  https://pythonspot.com/en/gui/
- Referenz des configparser-Moduls:
  https://docs.python.org/3/library/configparser.html