Kapitel 41
Grafische Benutzeroberflächen

Nachdem wir uns bisher ausschließlich mit Konsolenanwendungen beschäftigt haben, also mit Programmen, die über eine rein textbasierte Schnittstelle zum Benutzer verfügen, kann das Kribbeln in den Fingern und damit der Schritt zur grafischen Benutzeroberfläche nicht mehr länger unterdrückt werden. Im Gegensatz zur textorientierten Oberfläche von Konsolenanwendungen sind Programme mit grafischer Oberfläche intuitiver zu bedienen, grafisch ansprechender und werden im Allgemeinen als moderner empfunden. Die grafische Benutzeroberfläche eines Programms, auch *GUI* (*Graphical User Interface*) genannt, besteht zunächst aus *Fenstern* (engl. *windows*). Innerhalb dieser Fenster lassen sich beliebige *Steuerelemente* platzieren, häufig auch *Widgets* oder *Controls* genannt. Unter Steuerelementen versteht man einzelne Bedieneinheiten, aus denen sich die grafische Benutzeroberfläche als Ganzes zusammensetzt. So ist beispielsweise eine Schaltfläche (engl. *button*) oder ein Textfeld ein Steuerelement.

Sowohl die Terminologie als auch die Implementierung einer grafischen Oberfläche hängen sehr stark davon ab, welche Bibliothek, auch *Toolkit* genannt, verwendet wird. Aus diesem Grund werden wir zunächst verschiedene Toolkits besprechen, die mit Python verwendet werden können, und erst im zweiten Abschnitt zur eigentlichen Programmierung einer grafischen Benutzeroberfläche kommen. Dort behandeln wir zunächst das Modul `tkinter`, das das Tk-Toolkit verwendet und in der Standardbibliothek enthalten ist. Danach präsentieren wir Ihnen einen projektorientierten Einstieg in das umfangreichere und zeitgemäßere Qt-Framework unter Verwendung von PySide6.

41.1 Toolkits

Unter einem Toolkit versteht man eine Bibliothek, mit deren Hilfe sich Programme mit grafischer Benutzeroberfläche erstellen lassen. Neben einigen plattformabhängigen Toolkits, beispielsweise den *MFC* (*Microsoft Foundation Classes*) oder *Windows Forms* für Windows, sind gerade im Zusammenhang mit Python plattformunabhängige Toolkits wie *Qt*, *Gtk* oder *wxWidgets* interessant. Diese Toolkits sind zumeist für C (Gtk) oder C++ (Qt, wxWidgets) geschrieben, lassen sich jedoch durch sogenannte *Bindings* auch mit Python ansprechen. Im Folgenden werden wir die wichtigsten Python-Bindings für GUI-Toolkits auflisten und kurz erläutern.

41.1.1 Tkinter (Tk)

Website: *http://wiki.python.org/moin/TkInter*

Das Toolkit *Tk* wurde ursprünglich für die Sprache Tcl (*Tool Command Language*) entwickelt und ist das einzige Toolkit, das in der Standardbibliothek von Python enthalten ist. Das Modul `tkinter` (*Tk interface*) erlaubt es, Tk-Anwendungen zu schreiben, und bietet damit eine interessante Möglichkeit, kleinere Anwendungen mit einer grafischen Benutzeroberfläche zu versehen, für die der Benutzer später keine zusätzlichen Bibliotheken installieren muss. Ein Beispiel für ein Tk-Programm ist die Entwicklungsumgebung IDLE, die jeder Python-Version beiliegt.

Im Anschluss an diese Vorstellung der im Zusammenhang mit Python gängigsten Toolkits finden Sie eine Einführung in die Programmierung grafischer Benutzeroberflächen mit `tkinter`.

41.1.2 PyGObject (Gtk)

Website: *https://pygobject.readthedocs.io*

Das Toolkit *Gtk* (*GIMP Toolkit*) wurde ursprünglich für das Grafikprogramm GIMP entwickelt und zählt heute neben Qt zu den am verbreitetsten plattformübergreifenden Toolkits. Sowohl das Toolkit selbst als auch die Python-Bindings *PyGObject* stehen unter der *GNU Lesser General Public License* und können frei heruntergeladen und verwendet werden.

Das Gtk-Toolkit ist die Grundlage der freien Desktop-Umgebung GNOME und erfreut sich daher gerade unter Linux-Anwendern einer großen Beliebtheit. Obwohl es eigentlich für C geschrieben wurde, ist Gtk von Grund auf objektorientiert und kann deshalb gut mit Python verwendet werden.

41.1.3 Qt for Python (Qt)

Website: *https://www.qt.io/qt-for-python*

Bei *Qt* handelt es sich um ein umfassendes Framework, das von der Firma *The Qt Company* entwickelt wird, die aus der norwegischen Firma *Trolltech* entstand. Qt enthält sowohl ein GUI-Toolkit als auch GUI-fremde Funktionalität. Das durch und durch objektorientierte C++-Framework ist die Basis der freien Desktop-Umgebung KDE (*K Desktop Environment*) und aus diesem Grund ähnlich verbreitet und beliebt wie das Gtk-Toolkit.

Unter dem Namen *Qt for Python* entwickelt *The Qt Company* mittlerweile die offizielle Python-Anbindung *PySide6* für das Framework. Zuvor erfreuten sich die Bindings

PyQt des Drittanbieters *Riverbank Computing*[1] großer Beliebtheit. Im Gegensatz zu PyQt wird PySide6 von den Qt-Entwicklern *The Qt Company* offiziell unterstützt und ist unter den Bestimmungen der LGPL erhältlich.

PyQt und PySide6 sind in hohem Maße kompatibel, sodass es häufig ausreicht, die `import`-Anweisungen zu ändern, um ein PyQt-Programm unter PySide6 lauffähig zu machen. Diese API-Kompatibilität ist aber nicht vollständig und auch für die Zukunft nicht garantiert.

Sowohl Qt selbst als auch die Python-Bindings PySide6 sind unter einem dualen Lizenzsystem erhältlich. Für Software, die unter der GPL bzw. LGPL veröffentlicht wird, können auch Qt und PySide6 unter den Bestimmungen von GPL und LGPL verwendet werden. Zusätzlich gibt es ein Lizenzmodell für den kommerziellen Gebrauch. Qt wird für viele kommerzielle Projekte verwendet, darunter zum Beispiel Google Earth, Mathematica oder die Linux-Versionen von Skype und Spotify.

41.1.4 wxPython (wxWidgets)

Website: *http://www.wxpython.org*

Bei *wxPython*, den Python-Bindings für *wxWidgets*, handelt es sich um ein freies GUI-Toolkit, dessen erste Version bereits 1992 erschienen ist. Das Toolkit ist dementsprechend weit entwickelt und für alle gängigen Plattformen verfügbar. Gerade in Bezug auf Python ist wxWidgets beliebt und gut dokumentiert. Eines der Ziele des wxWidgets-Projekts ist es, das Look & Feel der verschiedenen Plattformen, auf denen das Toolkit lauffähig ist, bestmöglich abzubilden. Dies wird insbesondere dadurch erreicht, dass wxWidgets die Steuerelemente nicht selbst zeichnet, sondern dafür auf die Routinen der jeweiligen Plattform zurückgreift. Sowohl wxWidgets als auch wxPython befinden sich ständig in aktiver Entwicklung.

41.2 Einführung in tkinter

Nachdem wir Ihnen die verschiedenen GUI-Toolkits vorgestellt haben, für die Python-Bindings existieren, möchten wir uns der Programmierung grafischer Benutzeroberflächen widmen. Dazu ist in der Standardbibliothek das Modul `tkinter` enthalten, über das sich grafische Oberflächen mit dem Tk-Toolkit programmieren lassen. Das Modul `tkinter` ist die einzige Möglichkeit, ohne die Installation von Drittanbieterbibliotheken eine grafische Benutzeroberfläche in Python zu schreiben.

Deshalb lohnt es sich, einen Blick auf `tkinter` und seine Möglichkeiten zu werfen.

1 *https://riverbankcomputing.com*

Hinweis

Wenn Sie die folgenden Beispielprogramme in einer Anaconda-Umgebung unter Linux ausführen, kann es vorkommen, dass die üblicherweise verwendeten Systemschriftarten nicht zur Verfügung stehen und stattdessen sogenannte *Bitmap-Fonts* verwendet werden. Dies führt insbesondere auf hochauflösenden Monitoren zu einer unschönen und zu kleinen Darstellung der Dialoge.

Da es keine einfache Lösung für dieses Problem gibt, raten wir in diesen Fällen, Ihre Python-Systeminstallation für die Ausführung der Beispielprogramme einzusetzen statt Anaconda.

41.2.1 Ein einfaches Beispiel

Zum Einstieg in die Verwendung von `tkinter` möchten wir ein einfaches Beispielprogramm präsentieren und anschließend besprechen. Das Programm bringt einen Dialog auf den Bildschirm, der den Benutzer dazu auffordert, seinen Namen einzugeben. Durch einen Klick auf einen Button wird der Name umgedreht, die Buchstaben erscheinen also in umgekehrter Reihenfolge. Ein weiterer Button beendet den Dialog. Die Grafik in Abbildung 41.1 zeigt, wie die Oberfläche später aussehen wird. Links sehen Sie die GUI vor dem Anklicken des UMDREHEN-Buttons und rechts danach.

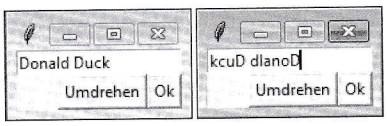

Abbildung 41.1 Die erste grafische Oberfläche

Diesem Beispielprogramm liegt der folgende Quelltext zugrunde:

```
import tkinter
class MyApp(tkinter.Frame):
    def __init__(self, master=None):
        super().__init__(master)
        self.pack()
        self.createWidgets()
    def createWidgets(self):
        self.nameEntry = tkinter.Entry(self)
        self.nameEntry.pack()
        self.name = tkinter.StringVar()
        self.name.set("Ihr Name...")
        self.nameEntry["textvariable"] = self.name
        self.ok = tkinter.Button(self)
```