Grafische Oberflächen

Mit Python lassen sich auch recht simpel grafische Oberflächen zu den Programmen erstellen. Dabei kann man auf die Funktionen der Bibliothek (so nennt man eine größere Menge von zusammengehörenden Klassen) *Tkinter* zurückgreifen. Das folgende Beispiel zeigt dir, wie du ein einfaches Fenster mit einem *Hallo Welt* - Text unter Verwendung von *Tkinter* erstellst und anzeigen lässt:

```
from Tkinter import *
fenster = Tk()
text = Label(fenster, text="Hella Welt")
text.pack()
fenster.mainloop()
```

In den folgenden Schritten wirst du dein erstes kleines Programm mit grafischer Oberfläche (GUI) programmieren und sollst die Grundlagen der GUI-Programmierung kennenlernen.

Schritt 1: Grundgerüst

Zunächst einmal musst du die Entwicklungsumgebung Geany starten und ein neues Python Programm öffnen. Anschließend kannst du den folgenden Quellcode abschreiben:

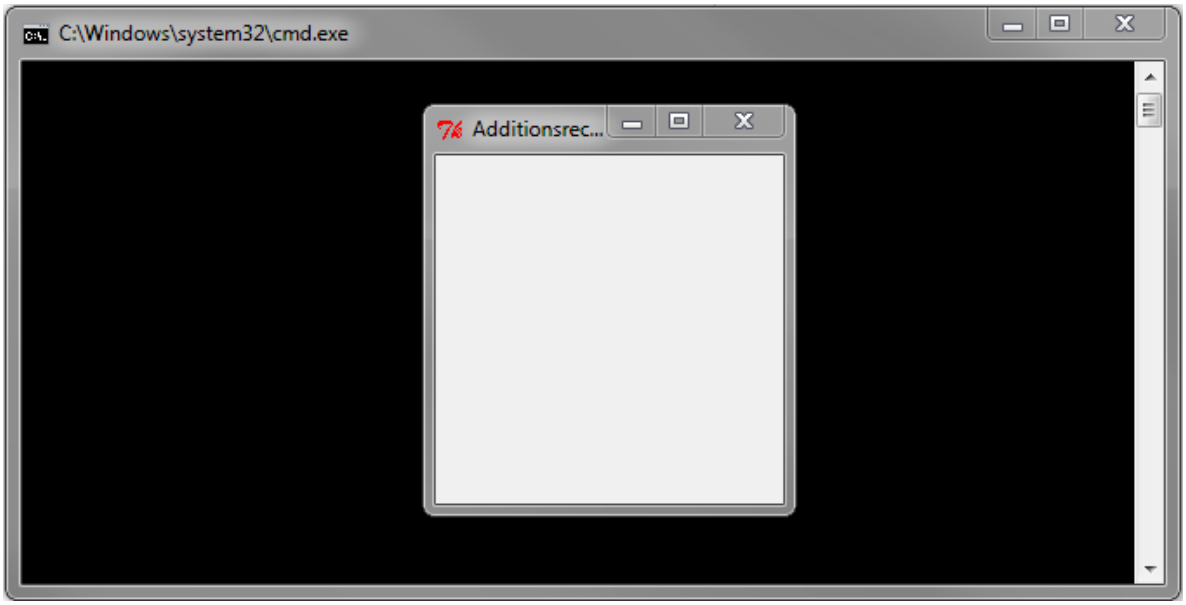
```
# Laden der Bibliothek
from Tkinter import *

# Fenster initialisieren und Titel setzen
fenster = Tk()
fenster.title("Additionsrechner")

# Fenster ausfuehren
fenster.mainloop()
```

Dies soll zunächst einmal das Grundgerüst für den Additionsrechner sein. Damit grafische Oberflächen erzeugt werden können, muss die Bibliothek *Tkinter* importiert werden (siehe Zeile 2).

Hinweis: Um uns etwas Aufwand zu sparen, importieren wir direkt alles aus dieser Bibliothek durch *from … import **.* Ansonsten müssten wir vor allen Klassen aus dieser Bibliothek immer *Tkinter.* schreiben.

Mit *Tk()* wird das Fenste initialisiert und mit der Methode *title*() lässt ein ein Titel für das Fenster festlegen. Die Methode *mainloop()* sorgt dafür, dass das Fenster angezeigt wird und alle damit verbundenen Programmlogiken ausführbar sind.

Wenn du das Programm ausführst, sollest du folgendes angezeigt bekommen. (Die Console, das Fenster mit dem schwarzen Hintergrund, wird immer mitgeöffnet.)



Schritt 2: Die ersten Komponenten

Komponenten wie Labels (Textbausteine), Eingabefelder oder Buttons müssen vor dem Aufruf von *mainloop()* zum Fenster hinzugefügt werden. Zunächst wollen wir das erste Eingabefeld hinzufügen. Füge dafür folgenden Quellcode nach der Initialisierung des Fenster ein:

```
# Erstes Eingabefeld hinzufuegen
zahl1_entry = Entry(fenster, width=5, justify="center")
zahl1_entry.grid(row=0, column=0)
```

Die Klasse Entry steht für typische Eingabefelder. Bei der Initialisierung Entry(…) muss der erste Parameter immer die Komponente sein, in die das Eingabefeld eingebunden wird (in diesem Fall das Fenster). Anschließend können weitere Parameter frei übergeben werden (in diesem Fall die Breite (width) und Textausrichtung (justify)).

Zum Schluss muss noch angegeben werden, wo genau das Eingabefeld positioniert werden soll. Hierfür gibt es 3 verschiedene Methoden: *pack(), grid()* und *place()*. Mit der *pack*-Methode können Elemente grob durch Angabe einer Richtung (left, right, top, down) angeordnet werden. Mit der *grid*-Methode unterteilt man die grafische Oberfläche in eine tabellenartige Struktur. Hier lässt sich ein Element unter Angabe von Zeile (row) und Spalte (column) anordnen. Mit der *pack*-Methode können Elemente pixelgenau auf der Oberfläche angeordnet werden. Der einfachhaltshalber verwenden wir hier nur die *grid*-Methode.

Würdest du das Programm nun ausführen, solltest du folgende Oberfläche angezeigt bekommen:

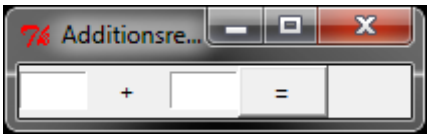
Schritt 3: Weitere Komponenten

Nun sollen weitere Komponenten hinzugefügt werden. Füge dazu den folgenden Quellcode nach der Platzierung des ersten Eingabefeldes hinzu:

```
# Plus-Label hinzufuegen
plus_label = Label(fenster, width=5, text="+")
plus_label.grid(row=0, column=1)

# Erstes Eingabefeld hinzufuegen
zahl2_entry = Entry(fenster, width=5, justify="center")
zahl2_entry.grid(row=0, column=2)

# Gleich-Button hinzufuegen
gleich_button = Button(fenster, width=5, text="=", command=addiere)
gleich_button.grid(row=0, column=3)

# Ergebnis-Label hinzufuegen
ergebnis_label = Label(fenster, width=5, text="")
ergebnis_label.grid(row=0, column=4)
```

Im obigen Quellcode werden nun vier weitere Komponenten hinzugefügt. Ein Label mit einem Pluszeichen als Text, ein weiteres Eingabefeld, ein Button mit einem Gleichheitszeichen als Text und ein Label ohne Text. Der Quellcode der neuen Komponenten ist ähnlich zu dem aus dem letzten Schritt. Einzige Besonderheit ist der Parameter *command* des Buttons. Dieser gibt an, welche Methode/Funktion ausgeführt wird, sobald auf diesen Button geklickt wird.

Die grafische Oberfläche würde nun wie folgt aussehen:



Ausführen lässt sich das Programm aber noch nicht, da die Methode/Funktion *addiere* noch nicht definiert wurde.

Schritt 4: Das Addieren

Wichtig ist, dass die Funktion addiere vor dem Button definiert werden muss, da dieser auf die Funktion verweist. Es empfiehlt sich, solche Funktionen immer am Anfang, direkt nach dem Importieren der Bibliotheken, zu definieren. Füge folgenden Quellcode vor der Initialisierung des Fenster hinzu:

```
# Methode zum Addieren definieren
def addiere():
	# Hole Eingaben aus Eingabefeldern
	zahl1 = zahl1_entry.get()
	zahl2 = zahl2_entry.get()

	# Pruefe ob Eingaben auch Zahlen sind
	if (zahl1.isdigit() and zahl2.isdigit()):
		# Wandle Eingaben in Integer um
		zahl1 = int(zahl1)
		zahl2 = int(zahl2)

		# Aendere Text des Ergebnis-Labels in berechnetes Ergebnis
		ergebnis_label.config(text = str(zahl1 + zahl2))
```

An sich sollte der obrige Quellcode anhand der Kommentare selbsterklärend sein. Wichtige Stellen sind hier nur die Verwendung der *get()* Methode, um den Inhalt der Eingabefelder zu erhalten, das Prüfen auf Integer-Zahlen mittels *isdigit()* und das Ändern des *text*-Attributes des Ergebnislabels mittels *config()*.

Schritt 5: Ungültige Eingaben

Was nun jedoch noch fehlt ist, eine Nachricht oder ein Hinweis, sofern keine ordentlichen Eingaben getätigt wurden. Hierfür muss die *if*-Anweisung der *addiere*-Funktion mit einer *else*-Anweisung erweitert werden. Als kleine Besonderheit soll jedoch die Fehlermeldung in Form eines keinen Hinweisfenster angezeigt werden. Solche Nachrichtenfenster können mit einer zusätzlichen Klasse ganz einfach angezeigt werden.

Importiere dazu die Klasse *tkMessageBox*. Füge dazu folgenden Quellcode nach dem Import von Tkinter ein:

```
import tkMessageBox
```

Anschließend kannst du folgenden Quellcode an das Ende der addiere Funktion hinzufügen:

```
# Wenn Eingaben keine Zahl, dann Hinweis-Fenster anzeigen
else:
	tkMessageBox.showinfo("Fehler", "Fehlerhafte Eingabe!")
```

Die Methode *showinfo* der Klasse *tkMessageBox* sorgt dafür, dass ein Nachrichtenfenster mit dem Titel *Fehler* und der Nachricht *Fehlerhafte Eingabe!* angezeigt wird.

Hiermit ist dein erstes Programm mit grafischer Oberfläche fertig.

Überblick

Der Quellcode des gesamten Programms sollte, wenn du alles richtig gemacht hast, wie folgt aussehen:

```
# Laden der Bibliothek
from Tkinter import *
import tkMessageBox

# Methode zum Addieren definieren
def addiere():
	# Hole Eingaben aus Eingabefeldern
	zahl1 = zahl1_entry.get()
	zahl2 = zahl2_entry.get()

	# Pruefe ob Eingaben auch Zahlen sind
	if (zahl1.isdigit() and zahl2.isdigit()):
		# Wandle Eingaben in Integer um
		zahl1 = int(zahl1)
		zahl2 = int(zahl2)

		# Aendere Text des Ergebnis-Labels in berechnetes Ergebnis
		ergebnis_label.config(text = str(zahl1 + zahl2))

	# Wenn Eingaben keine Zahl, dann Hinweis-Fenster anzeigen
	else:
		tkMessageBox.showinfo("Fehler", "Fehlerhafte Eingabe!")

# Fenster initialisieren und Titel setzen
fenster = Tk()
fenster.title("Additionsrechner")

# Erstes Eingabefeld hinzufuegen
zahl1_entry = Entry(fenster, width=5, justify="center")
zahl1_entry.grid(row=0, column=0)

# Plus-Label hinzufuegen
plus_label = Label(fenster, width=5, text="+")
plus_label.grid(row=0, column=1)
```

```
# Erstes Eingabefeld hinzufuegen
zahl2_entry = Entry(fenster, width=5, justify="center")
zahl2_entry.grid(row=0, column=2)

# Gleich-Button hinzufuegen
gleich_button = Button(fenster, width=5, text="=", command=addiere)
gleich_button.grid(row=0, column=3)

# Ergebnis-Label hinzufuegen
ergebnis_label = Label(fenster, width=5, text="")
ergebnis_label.grid(row=0, column=4)

# Fenster ausfuehren
fenster.mainloop()
```

Ein Berechnung auf der grafischen Oberfläche kann so aussehen: