Einführung in Tkinter

Einleitung

Viele Computeranwendungen besitzen eine grafische Benutzeroberfläche, über die ein Benutzer Menus aufrufen, Einstellungen vornehmen, Aktionen gezielt mit Mausklicks auslösen oder auch Eingaben in bestimmte Felder machen kann. Die grafische Benutzeroberfläche bildet so die Schnittstelle zwischen Mensch und Maschine.

Wie man ein Programm mit grafischer Benutzeroberfläche entwickelt, wird Gegenstand dieser Einführung sein. Wir werden hier nur einfache strukturierte Benutzeroberflächen betrachten und die Konzepte und Sprachelemente vorstellen, die zur Erstellung erforderlich sind.

GUI-Entwicklung mit tkinter

Worum geht es hier?

Grafische Benutzeroberflächen werden mit Hilfe geeigneter Programmierbausteine realisiert. Ziel dieser Einführung ist es, einige dieser Programmierbausteine vorzustellen. Du solltest nach einer kurzen Einarbeitung in der Lage sein, selbst einfache Benutzeroberflächen zu erstellen.

Hier lernst du ...

... wie man Komponenten einer grafischen Benutzeroberfläche mit Hilfe von GUI-Objekten erzeugt.
... wie man die Verarbeitung von Ereignissen implementiert.

Hinweis - tkinter

`tkinter` steht für Toolkit-Interface und ist eine Bibliothek, die all die Programmeinheiten enthält, die man zur Erzeugung von Komponenten einer grafischen Benutzeroberfläche mit Python benötigt. Dieses Modul gehört zur Standarddistribution von Python und kann daher direkt genutzt werden. In den folgenden Unterabschnitten wird exemplarisch gezeigt, wie man Komponenten grafischer Benutzeroberflächen mit Hilfe von `tkinter` realisiert.

Die folgenden Ausführungen stellen keine systematische Einführung in die Entwicklung grafischer Benutzeroberflächen mit `tkinter` dar. Viele Aspekte und Details bleiben ungeklärt. Wenn du mehr über `tkinter` wissen willst, dann musst du Dokumentationen und Tutorials nutzen.

Weitere Informationen zu `tkinter` findest du hier:

GUI-Programmierung mit Python: Python Tkinter Einführung
(https://www.python-kurs.eu/python_tkinter.php)

Python – GUI Programming (Tkinter) – Tutorialspoint
(https://www.tutorialspoint.com/python/python_gui_programming.htm)

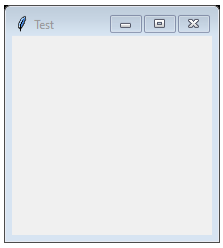
Erzeugung eines Fensters

Programm zur Erzeugung eines Fensters

Das folgende Programm erzeugt bereits ein funktionsfähiges Anwendungsfenster.

```
from tkinter import *
# Erzeugung des Fensters
root = Tk()
root.title('Test')
# Aktivierung des Fensters
root.mainloop()
```

Wenn man das Programm ausführt, dann ergibt sich dieses Anwendungsfenster:



Es macht eigentlich nichts weiter, als sich wie ein Fenster zu verhalten. Man kann es z. B. vergrössern oder auch schliessen.

Fenster als GUI-Objekt

Mit der Anweisung `from tkinter import *` werden alle Namen des Moduls `tkinter` importiert.
Die Anweisung `root = Tk()` erzeugt ein Objekt der Klasse `Tk` und bindet es an den Namen `root`. Mit der Anweisung `root.title('Test')` wird der im Fenster angezeigte Titel festgelegt. Mit der Anweisung `root.mainloop()` wird schliesslich die Ereignisschleife aktiviert.

Aufgabe 1

(a) Teste das oben gezeigte Programm. Speichere es hierzu geeignet ab (z.B. unter dem Dateinamen `gui_fenster.py`) und führe es aus

(b) Ergänze das Programm wie folgt. Welche Wirkung hat die `geometry`-Methode?

```
from tkinter import *
# Erzeugung des Fensters
root = Tk()
root.title('Test')
root.geometry('350x145')
# Aktivierung des Fensters
root.mainloop()
```

Anzeige von Text und Daten

Zielsetzung

Mit Hilfe von Textfeldern (Label) sollen Daten als Texte auf der Benutzeroberfläche dargestellt werden. Als Beispiel betrachten wir die Darstellung eines Kalenders.

Textfelder als GUI-Objekte

Das folgende Testprogramm zeigt, wie man Textfelder erzeugt und zur Anzeige von Texten und Daten nutzt.

```
from tkinter import *

# Erzeugung des Fensters
root = Tk()
root.title('Kalender')
root.geometry('130x145')

# Label für die Anzeige der Daten
labelMonat = Label(root, text='Januar', fg='white', bg='gray',
font=('Arial', 16))
labelMonat.place(x=5, y=5, width=120, height=20)
labelTag = Label(root, text='21', fg='red', bg='#FFCFC9',
font=('Arial', 72))
labelTag.place(x=5, y=30, width=120, height=80)
labelWochentag = Label(root, text='Montag')
labelWochentag.place(x=35, y=115, width=60, height=30)

# Aktivierung des Fensters
root.mainloop()
```

Textfelder werden als Objekte der Klasse `Label` implementiert.

Im Programm oben werden drei `Label`-Objekte erzeugt: Die Anweisungen `labelMonat = Label(...)`, `labelTag = Label(...)` und `labelWochentag = Label(...)` erzeugen jeweils ein Objekt der Klasse `Label` und binden sie an passende Bezeichner. Diese Objekte verwalten die Textfelder im Anwendungsfenster.

Bei der Erzeugung der `Label`-Objekte werden auch gleich Attribute initialisiert. Die Werte werden dabei mit Hilfe von Parametern direkt übergeben. Man erkennt am Beispiel bereits einige wichtige Attribute eines Labels:

Attribut	Bedeutung
text	Beschriftung
bg (bzw. background)	Hintergrundfarbe
fg (bzw. foreground)	Textfarbe
font	Schriftformat
padx, pady	Innenabstand in x- und y-Richtung (padding)

Zugriff auf die Attribute

Auf Attribute eines GUI-Objekts kann man lesend und schreibend zugreifen. Im folgenden Programm wird dies exemplarisch aufgezeigt.

```
from tkinter import *

# Erzeugung des Fensters
root = Tk()
root.title('Kalender')
root.geometry('130x85')

# Label für die Anzeige der Daten
labelWochentag = Label(root, text='Dienstag', fg='white',
bg='gray', font=('Arial', 16))
labelWochentag.place(x=5, y=5, width=120, height=20)
labelTag = Label(root, text='27', fg='red', bg='#FFCFC9',
font=('Arial', 24))
labelTag.place(x=5, y=30, width=55, height=50)

# lesender Zugriff auf Attribute von labelTag
schriftfarbe = labelTag.cget('fg')
hintergrundfarbe = labelTag.cget('bg')
schriftformat = labelTag.cget('font')

# schreibender Zugriff auf Attribute von labelMonat
labelMonat = Label(root, text='04')
labelMonat.config(fg=schriftfarbe)
labelMonat.config(bg=hintergrundfarbe)
labelMonat.config(font=schriftformat)
labelMonat.place(x=70, y=30, width=55, height=50)

# Aktivierung des Fensters
root.mainloop()
```

Das Programm erzeugt diese grafische Benutzeroberfläche.

Jedes `Label`-Objekt stellt die in der Übersicht gezeigten Methoden zur Verfügung.

Attribut	**Bedeutung**
cget(option)	Liefert den Wert des übergebenen Attributparameters.
config(option=wert)	Setzt den Wert eines Attributs.

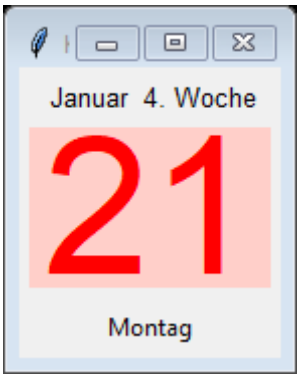
Beachte, dass die Methoden `cget(…)` und `config(…)` für alle GUI-Objekte zur Verfügung stehen.

Mit der Methode `cget(…)` kann man lesend auf Attributwerte eines GUI-Objekts zugreifen. Beispielsweise liefert die Anweisung `labelTag.cget('fg')` den aktuellen Wert des Attributs `fg` des `Label`-Objekts `labelTag`.

Mit der Methode `config(…)` kann man Attributwerte eines GUI-Objekts festlegen. So bewirkt beispielsweise die Anweisung `labelMonat.config(fg='red')`, dass das Attribut `fg` des `Label`-Objekts `labelMonat` den Wert `'red'` erhält.

Aufgabe 2

Ändere das weiter oben gezeigte Programm (im Abschnitt "Textfelder als GUI-Objekte") so ab, dass folgende GUI erzeugt wird.



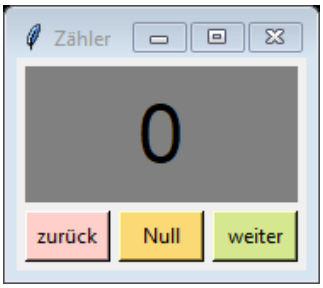
Aufgabe 3

Erstelle ein Programm zur Erzeugung der folgenden Benutzeroberfläche:

Anklicken von Schaltflächen

Zielsetzung

Mit Hilfe von Schaltflächen soll ein Zähler aktiviert werden. Wenn man z.B. die Schaltfläche mit der Beschriftung "weiter" anklickt, dann soll der Zähler um einen Schritt hochgezählt werden.



Schaltflächen als GUI-Objekte

Das folgende Programme zeigt, wie man eine Schaltfläche erzeugt und zur Verarbeitung von Daten nutzt. Beachte, dass das Programm noch nicht ganz fertig ist. Wenn man die Schaltflächen mit den Beschriftungen "zurück" und "Null" anklickt, geschieht hier gar nichts.

```
from tkinter import *

# Ereignisbehandlung
def buttonWeiterClick():
    stand = int(labelZahl.cget('text'))
    stand = stand + 1
    labelZahl.config(text=str(stand))

def buttonZurueckClick():
    pass

def buttonNullClick():
    pass

# GUI-Objekte
# Fenster
root = Tk()
root.title('Zähler')
root.geometry('170x125')
```

```
# Label
labelZahl = Label(root, text='0', bg='gray', font=('Arial', 36))
labelZahl.place(x=5, y=5, width=160, height=80)

# Button
buttonWeiter = Button(root, text='weiter', bg='#D5E88F',
command=buttonWeiterClick)
buttonWeiter.place(x=115, y=90, width=50, height=30)
buttonZurueck = Button(root, text='zurück', bg='#FFCFC9',
command=buttonZurueckClick)
buttonZurueck.place(x=5, y=90, width=50, height=30)
buttonNull = Button(root, text='Null', bg='#FBD975',
command=buttonNullClick)
buttonNull.place(x=60, y=90, width=50, height=30)

# Aktivierung des Fensters
root.mainloop()
```

Schaltflächen werden durch Objekte der Klasse `Button` dargestellt. Durch die Anweisung `buttonWeiter = Button(…)` wird im Programm oben ein Objekt der Klasse `Button` erzeugt und an den Namen `buttonWeiter` gebunden.

Behandlung von Ereignissen

Wenn man eine Schaltfläche anklickt, werden (in der Regel) bestimmte Reaktionen ausgelöst. Reaktionen auf das Ereignis Anklicken einer Schaltfläche werden in einer Ereignisbehandlungsprozedur festgelegt. Damit beim Anklicken einer bestimmten Schaltfläche die Ausführung der zu dieser Schaltfläche zugehörige Ereignisbehandlungsprozedur ausgelöst wird, muss die Ereignisbehandlungsprozedur an die betreffende Schaltfläche angebunden werden. Dies wird hier mit dem Setzen des `command`-Attributs realisiert.

Im Programm oben wird durch die Parameterübergabe `command=buttonWeiterClick` bei der Erzeugung des Objekts `buttonWeiter` festgelegt, dass die Prozedur `buttonWeiterClick` ausgeführt wird, wenn die Schaltfläche zum Objekt `buttonWeiter` angeklickt wird.

Datenverarbeitung bei der Ereignisbehandlung

Die oben im Programm vorkommende Ereignisverarbeitungsprozedur kann als Muster für viele Datenverarbeitungsvorgänge dienen, bei denen die Daten von GUI-Objekten verwaltet werden.

```
def buttonWeiterClick():
    # Übernahme der Daten
    stand = int(labelZahl.cget('text'))
    # Verarbeitung der Daten
    stand = stand + 1
    # Aktualisierung der Anzeige der Daten
    labelZahl.config(text=str(stand))
```

Schritt 1: Erst werden die benötigten Daten von Attributen betreffender GUI-Objekte übernommen und in Hilfsvariablen zwischengespeichert. Beachte, dass hier oft Typanpassungen erforderlich sind.

Schritt 2: Die jetzt mit Hilfe von Variablen erfassten Daten werden mit Hilfe geeigneter Anweisungen verarbeitet.

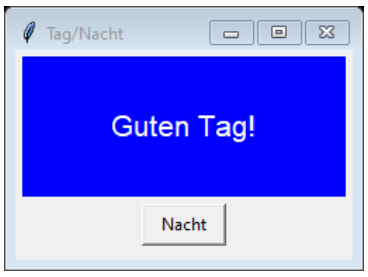
Schritt 3: Die Ergebnisse der Verarbeitung werden zur Anzeige auf dem Bildschirm an geeignete GUI-Objekte weitergegeben. Auch hier können Typumwandlungen erforderlich sein.

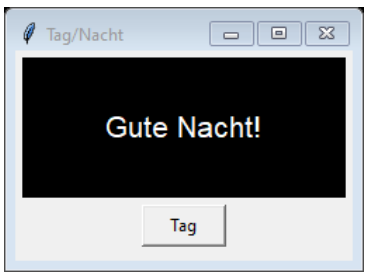
Aufgabe 4

(a) Ergänze das oben gezeigte Programm so, dass der Zählerstand beim Anklicken der Schaltfläche mit der Beschriftung "Null" auf 0 gesetzt wird und beim Anklicken der Schaltfläche mit der Beschriftung "zurück" um 1 verringert wird, sofern der Zählerstand hierdurch nicht unter 0 gerät.

Aufgabe 5

Entwickle eine GUI, mit der man zwischen Tag und Nacht umschalten kann. Beachte, dass sich hier auch die Beschriftung der Schaltfläche ändern soll.





Gestaltung des Layouts

Zielsetzung

Das Layout einer grafischen Benutzeroberfläche wird wesentlich durch die Anordnung und Gestaltung der GUI-Komponenten bestimmt. Tkinter stellt drei verschiedene Layout-Manager zur Verfügung, die das Layout ausgehend von den im Programm getroffenen Festlegungen automatisiert erzeugen.

- Der Place-Manager benutzt Koordinatensysteme, um GUI-Komponenten zu platzieren.
- Der Pack-Manager packt alle GUI-Komponenten in ein System aus ineinander geschachtelten Rahmen.
- Der Grid-Manager benutzt ein Raster, in das alle GUI-Komponenten eingepasst werden.

Man muss im Einzelfall entscheiden, welches Verfahren zweckmäßig ist. Wir haben in den vorherigen Abschnitten den Place-Manager verwendet, weil seine Darstellungslogik einfach und leicht zu durchschauen ist.

Folgende Parameter können beim Platzieren der GUI-Komponenten benutzt werden.

Parameter	**Bedeutung**
x, y	Position der GUI-Komponente im zugehörigen Koordinatensystem.
width, height	Breite und Höhe der GUI-Komponente

Beachte, dass es bei komplexeren Benutzeroberflächen recht mühsam ist, alle Koordinaten und Ausmaße der GUI-Komponenten zu bestimmen.

Lösungen zu den Aufgaben

https://github.com/przl-lab/tkinter

Verwendete Quellen

- https://www.inf-schule.de/software/gui/entwicklung_tkinter (26. April 2021)
- https://de.wikipedia.org/wiki/Tkinter (26. April 2021)
- https://www.python-kurs.eu/python_tkinter.php (26. April 2021)
- https://www.tutorialspoint.com/python/python_gui_programming.htm (26. April 2021)