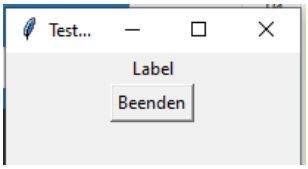
Fensterprogrammierung

Grundgerüst




```
from tkinter import *          # Bibliothek laden für alle Elemente
root = Tk()                    #Instands der Klasse in Variable speichern
root.title("Fenstertitel ")    #Titel für Fenster
root.geometry('300x200')       #Fenstergröße festlegen ( breit 300, hoch 200 Pixel)
label = Label(root, text= "Label ")    # Label vorbereiten, Text erzeugen
button = Button(root, text= "Beenden ", command=root.destroy)
# Button vorbereiten, Text erzeugen, Funktion (hier z.B. eine Standardfunktion)
label.pack()                   # Label anzeigen (Standardposition)
button.pack()                  #Button anzeigen
root.mainloop()                #Dauerschleife erzeugen
```

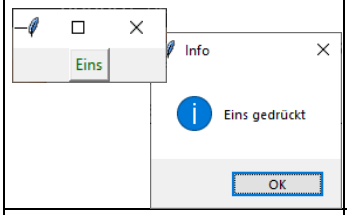
Label und Button Eigenschaften Schriftart, Farbe, Größe Position des Textes nachträglich ändern

```
label["bg"]="#FF0000"          # bg für background, Farbe als Hexwert (R,G,B)
label["fg"]="#FF00FF"          # fg für foreground(Text), Farbe als Hexwert (R,G,B)
label["font"]="Arial 16 italic"  # Schriftart, Größe, (oder z.B. Courier 12 bold)
label["height"]=2              # 2 = doppelte Zeichengröße
label["width"]=20              # 20 Zeichen
label["anchor"]="w"            # Ausrichtung Text im Button od. Label(n, s, w, e) sw
```

Funktionen für Button


```
from tkinter import *          # Bibliothek laden für alle Elemente
root = Tk()                    # Instands der Klasse in Variable speichern
def wechsleFarbe():            # Funktion definieren
  label["fg"]="#FF0000"        # Inhalt der Funktion eingerückt(hier Farbänderung)
root.title("Test")             #Titel für Fenster
root.geometry('300x200')       #Fenstergröße festlegen ( breit 500, hoch 200 Pixel)
label = Label(root, text="Label")    # Label vorbereiten, Text erzeugen
button = Button(root, text="Knopf", command=wechsleFarbe)  # Funktionsaufruf
label.pack()                   # Label anzeigen (Standardposition)
button.pack()                  #Button anzeigen
```

Messagebox

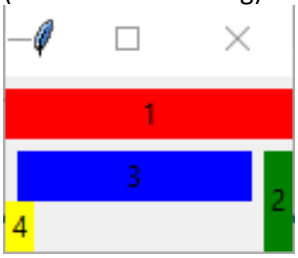



```
from tkinter import *          # Bibliothek laden für alle Elemente
from tkinter import messagebox  # Bibliothek für Messagebox
root = Tk()                    # Instands der Klasse in Variable speichern
def func1():                   # Funktion erstellen
   messagebox.showinfo("Info","Eins gedrückt") # Messagebox öffnen
button1 = Button(root, text="Eins", fg="dark green", command = func1)
button1.pack()                 # Button anzeigen
```

Bilder darstellen

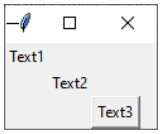
```
from tkinter import *          # Bibliothek laden für alle Elemente
root = Tk()                    #Instands der Klasse in Variable speichern
img1 = PhotoImage(file= "D:/blume.png ")#Bild laden als png
label = Label(root, image=img1)        # Bild zuweisen
label.pack()                   # Bild (label) anzeigen
root.mainloop()                # Dauerschleife erzeugen
```

Positionierung mit pack (Standarddarstellung)




```
from tkinter import *
root = Tk()
label1 = Label(root, text="1", bg="red")     # Label 1 erzeugen (rot)
label2 = Label(root, text="2", bg="green")  # Label 2 erzeugen (grün)
label3 = Label(root, text="3", bg="blue")    # Label 3 erzeugen (blau)
label4 = Label(root, text="4", bg="yellow") # Label 4 erzeugen (gelb)
label1.pack(side='top', fill='x', pady= '5')     # oben volle Breite danach 5 px frei
label2.pack(side='right', fill='y')              # rechts volle Höhe
label3.pack(side='top', fill='both', padx='5')
# oben volle Breiteund volle Höhedanach 5 px links und rechts frei
label4.pack(side='left')                         # Ausrichtung links
root.mainloop()
```

Positionierung mit grid (Raster)
0,0|0,1|0,2
1,0|1,1|1,2
2,0|2,1|2,2

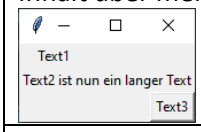



```
from tkinter import *                  # Bibliothek laden für alle Elemente
root = Tk()                            # Instands der Klasse in Variable speichern
lab1 = Label(root, text="Text1")       # Labels erzeugen lab1
lab2 = Label(root, text="Text2")       # Labels erzeugen lab2
but1 = Button(root, text="Text3") # Button erstellen
lab1.grid(row=0, column=0)             # bei Aktivierung - Positionierung im Gitternetz
lab2.grid(row=1, column=1)             # row für Reihe und column für Spalte
but1.grid(row=2, column=2)             # Gesamtbreite Summe der Textinhalte
root.mainloop()
```

Textpositionierung in einer Zelle von grid

```
lab2.grid(row=1, column=1, sticky="n")     # Ausrichtung erfolgt über n, s, w, e
                                           #auch möglich: sw, ne usw.
```

Inhalt über mehr Spalten

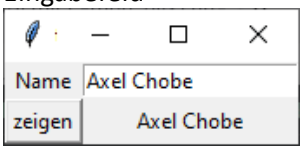



```
lab2.grid(row=1, column=0, columnspan=3)
#Label 2 beginnt jetzt in Spalte 0 und erstreckt sich über 3 Spalten
lab2.grid(row=0, column=0, rowspan=2)
#Label 2 erstreckt sich über 2 Reihen
```

Frame (unsichtbarer Container zur Formatierung)


```
from tkinter import *
root=Tk()
frame1 = Frame(root, relief=RIDGE, bd=2 )  #Container definieren, Relief mit Größe
frame2 = Frame(root, relief=RIDGE, bd=2 )  # auch GROOVE, SUNKEN, RAISED, FLAT
b1 = Button(frame1, text= "fb1 ")      # ein Button in den Frame1
b2 = Button(frame2, text= "fb1 ")      # ein Button in den Frame2
l1 = Label(frame1, text= "fl1 ")       # ein Label in den Frame1
l2 = Label(frame2, text= "fl2 ")       # ein Label in den Frame2
frame1.pack(side=LEFT)                 # frame1 aktivieren, Ausrichtung links
frame2.pack(side=RIGHT)                # frame2 aktivieren, Ausrichtung rechts
b1.pack(side=LEFT)                     # b1 aktivieren und im Frame ausrichten
l1.pack(side=RIGHT)                    # l1 aktivieren und im Frame ausrichten
b2.pack(side=LEFT)                     # b2 aktivieren und im Frame ausrichten
l2.pack(side=RIGHT)                    # l2 aktivieren und im Frame ausrichten
root.mainloop()                        # Dauerschleife
```

Eingabefeld




```
from tkinter import *                  # Bibliothek laden
def lese():                            #Funktion für die Ausgabe der Eingabe
   l2["text"] = nf.get()               # Änderung des Textfeldes l3 (vorher leer)
root = Tk()                            # Instanz der Klasse in Variable schreiben
l1 = Label(root, text= "Name ")        # Textfeld Name
l2 = Label(root, text= " ")            # Textfeld für Ausgabe
b1 = Button(root, text= "zeigen ", command=lese)  # Button mit Funktionsaufruf
nf = Entry(root)                       # Eingabefeld für Name erstellen
l1.grid(row=0, column=0)               # Anzeige Textfeld-Name Zeile 0 Spalte 0
l2.grid(row=1, column=1)               # Anzeige Textfeld v. Eingabe Zeile 1 Spalte 1
b1.grid(row=1, column=0)               # Anzeige Button Zeile 1 Spalte 0
nf.grid(row=0, column=1)               # Anzeigen Textfeld-nf Zeile 0 Spalte 1
root.mainloop()                        # Dauerschleife
```

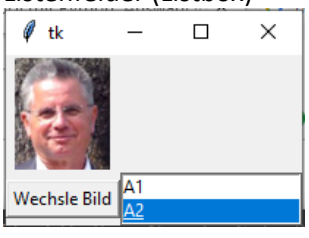
Eingabefeld Vordefiniert

```
nf.grid(row=0, column=1)               #Anzeige Textfeld-nf  (wie vorher)
nf.insert(0, "Name eingeben")          #nächste Zeile für Text, 0 = Position
```

Eingabefeld löschen nach Funktionsaufruf

```
def lese():                            #Funktion für die Ausgabe der Eingabe
   l2["text"] = nf.get()
   nf.delete(0, END)                   # Löscht Eingabe von Vorn bis Ende
```

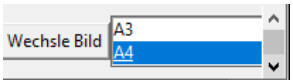
Listenfelder (Listbox)




```
from tkinter import *                  # Bibliothek laden
def wb():                              # Funktion zum Wechsel der Bilder
   lc = liste.get("active")            # übergibt den aktiven Namen der Liste an lc
   if lc == "A1":                      # ist der Wert A1 dann
      lab1["image"] = img1             # ändere lab1 auf img1 o. andere Funktionen
   elif lc == "A2":                    # Abfrage auf A2
      lab1["image"] = img2             # ändere lab1 auf img2 o. andere Funktionen
   else:                               # Abfrage des letzten Elementes
      lab1["image"] = img3             # ändere lab1 auf img3 o. andere Funktionen
root = Tk()
```

```
img1 = PhotoImage(file="D:/axel1.png")   # die einzelnen Bilder festlegen
img2 = PhotoImage(file="D:/axel2.png")
img3 = PhotoImage(file="D:/axel3.png")
liste = Listbox(root, height=2)          # Liste vorbereiten; 2 für 2 Eingaben sichtbar
liste.insert("end","A1", "A2", "A3")   # end für ans Ende anfügen dann die Texte
b1 = Button(root, text="Wechsle Bild", command=wb) # Button m. Funktionsaufruf
lab1 = Label(root, image=img1)           # Label für Bilddarstellung Init -> Bild1
lab1.grid(row=0, column=0)               # Bildlabel darstellen oben links
b1.grid(row=1, column=0)                 # Button darstellen unten links
liste.grid(row=1, column=1)              # Listenfeld darstellen unten rechts
root.mainloop()                          # Dauerschleife
```

Scrollbar für Listbox

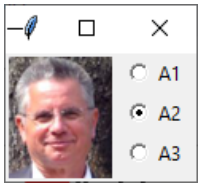



```
liste.insert …
scroll = Scrollbar(root, orient=VERTICAL) # Orientierung VERTIVAL /HORIZONTAL
liste["yscrollcommand"]=scroll.set # für die Liste , Verknüpfung mit der Scrollbar
scroll["command"]=liste.yview       # auch umgekehrt notwendig / auch in x-Achse
scroll.grid(row=1, column=1, sticky="e") # sticky für ganz rechts  besser:N+S+E
b1 = Button….
```

Vereinfachte Abfrage Listenbox

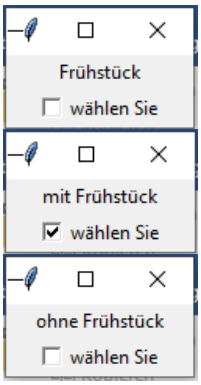
```
from tkinter import *
def wb():                                # Funktion zum Wechsel der Bilder
    lc = liste.index("active")           # index gibt Wert von 0 bis X aus
    lab1["image"]=img[lc]                # ändern lab1 mit akuellem lc-Wert
root = Tk()
img = []                                 # Liste der Bilder erstellen
img.append(PhotoImage(file= "D:/axel1.png "))  # Abruf z.B. mit img[2]
img.append(PhotoImage(file= "D:/axel2.png "))
img.append(PhotoImage(file= "D:/axel3.png "))
liste = Listbox(root, height=2)          # Liste vorbereiten; 2 für 2 Eingaben sichtbar
liste.insert("end","A1", "A2", "A3")   # end für ans Ende anfügen dann die Texte
b1 = Button(root, text="Wechsle Bild", command=wb) # Button m. Funktionsaufruf
lab1 = Label(root, image=img[0])         # Label für Bilddarstellung Init -> Bild1
lab1.grid(row=0, column=0)               # Bildlabel darstellen oben links
b1.grid(row=1, column=0)                 # Button darstellen unten links
liste.grid(row=1, column=1)              # Listenfeld darstellen unten rechts
root.mainloop()
```

Radiobutton



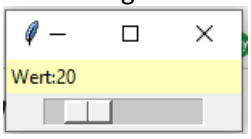

```
from tkinter import *                    # Bibliothek laden
def wb():                                # Funktion zum Wechseln der Bilder
   lab1["image"]=img[ikon.get()]         # Bild wechselt entsprechend der Variable
root = Tk()
img = []                                 # Liste der Bilder erstellen
img.append(PhotoImage(file="D:/axel1.png"))  # Abruf z.B. mit img[2]
img.append(PhotoImage(file="D:/axel2.png"))
img.append(PhotoImage(file="D:/axel3.png"))
lab1 = Label(root, image=img[0] )        # Label für die Bilder,hier mit 0 beginnend
frame = Frame(root)                      # einen Frame für die Button erstellen
ikon = IntVar()                          # Variable für Radiobutton festlegen
ikon.set(0)                              # Wert 0 setzen, RB 0 hat den Auswahlpunkt
rb0 = Radiobutton(frame, text="A1", variable=ikon, value = 0, command=wb)
rb1 = Radiobutton(frame, text="A2", variable=ikon, value = 1, command=wb)
rb2 = Radiobutton(frame, text="A3", variable=ikon, value = 2, command=wb)
# Radiobutton in Frame(1 - 4) mit Name, Variable 1 o. 0; Wert und Funktionsaufruf
lab1.pack(side=LEFT)                     # Label darstellen auf linker Seite
frame.pack()                             # Frame darstellen Standard rechte Seite
rb0.grid(row=0, sticky="w")              # Radiobutton 0 darstellen 1.Zeile linksbündig
rb1.grid(row=1, sticky="w")              # Radiobutton 1 darstellen 2.Zeile linksbündig
rb2.grid(row=2, sticky="w")              # Radiobutton 1 darstellen 3.Zeile linksbündig
root.mainloop()                          # Dauerschleife
```

Checkbutton



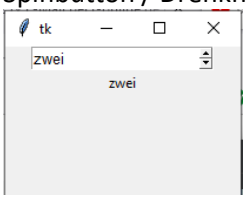

```
from tkinter import *                    # Bibliothek laden
def wt():                                # Funktion zum Wechseln des Textes
   if var.get():                         # ist Variablenwert = 1
      lab1["text"]="mit Frühstück"       # gib Text auf Label aus
   else:                                 # ist Variablenwert = 0
      lab1["text"]="ohne Frühstück"      # gib Text auf Label aus
   return                                # Rückkehr aus Funktion (Optional)
root = Tk()                              # Instands der Klasse
var = IntVar()                           # Variable für Checkbutton festlegen
lab1 = Label(root, text="Frühstück")  # Label erstellen
cb = Checkbutton(root, text="wählen Sie", variable=var, command=wt)
# Checkbutton mit Text, variable (1 oder 0) und Funktionsaufruf
lab1.pack()                              # Label darstellen
cb.pack()                                # Checkbutton darstellen
root.mainloop()                          #Dauerschleife
```

Schieberegler




```
from tkinter import *                    # Bibliothek laden
def sw(swert):                           # Funktion mit übergebenem Wert
   lab["text"]="Wert: " + swert          # Änderung des Wertes( !!swert ist ein String
root = Tk()                              # Instanz der Klasse          also keine Zahl)
lab = Label(root, text="0", width=20, bg="#FFFFBB", anchor="w")
# Label erstellen mit Text, Breite, Farbe, Ausrichtung Text links
regler = Scale(root, from_=17, to=34, command=sw)# zw. 17 und 34, Funktionsruf
#orient=HORIZONTAL, showvalue=0,  # Optional: Ausrichtung , keine Werteanzeige
#length=10, width=9, sliderlength=6)  # Optional: Breite, Höhe, Breite des Reglers
#resolution=0.1, tickinterval=2          # Optional: Intervall, Anzeige Schrittweite
#digits=3                                # Optional Anzahl von Zeichen (Stinglänge)
lab.pack()                               # Label darstellen
regler.pack()                            # Regler darstellen
root.mainloop()                          #Dauerschleife
```

Spinbutton / Drehknopf

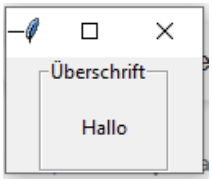



```
from tkinter import *                    # Bibliothek laden
def zw():                                # Funktion zur Ausgabe
   lab["text"]=sb.get()                  # gibt den geänderten Wert im Label aus
root = Tk()
lab = Label(root, text="null")           # Label erstellen mit Ausgangstext
v1 = ["eins", "zwei", "drei", "vier"]    #Liste für Werte definieren
sb = Spinbox(root, font="Arial 10", wrap=True, values=v1, command=zw)
# Spinbutton erstellen mit Schriftart, wrap -fängt am Ende von Vorne an, Werten,
sb.pack()                                # Spinbox darstellen          Kommando
lab.pack()                               # Label darstellen
root.mainloop()                          #Dauerschleife
```

Canvas / Leinwand
Grafikprogrammierung

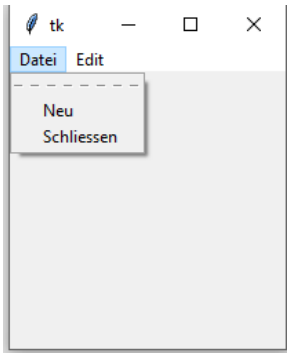

```
from tkinter import *                    # Bibliothek laden
root = Tk()
lw = Canvas(bg="black",width=200,height=200)
#Farbe mit Kurzwort oder Hex  =#000000, Breite und Höhe
lw.create_rectangle(80,80,120,120,fill="green") # von oben links nach unten rechts
lw.create_oval(0,0,50,50,fill="red" )    # Kreis
lw.create_line(100, 100, 200, 200,fill="blue",width=3) #Linie, Strichstärke
lw.create_text(10 ,120, text="Hallo Welt! ", fill="#FFFFFF",anchor="nw",
font="Blazed 20")     # Textposition von mitte aus,mit anchor (nw) ol von Position
img = PhotoImage(file="D:/ball1.png")    # Bild in Variable laden
lw.create_image(150, 40, image = img) # Bild darstellen #Position von Mitte aus
lw.pack()                                #Leinwand anzeigen
root.mainloop()                          #Dauerschleife
```

LabelFrame
(Gruppieren von Steuerelementen)



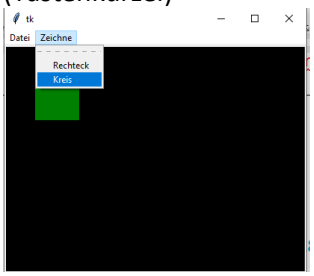

```
from tkinter import *                                   # Bibliothek laden
root = Tk()                                             # Instanz der Klasse
root.geometry("500x500")                                #Fenstergröße definieren
labFr = LabelFrame(root, text="Überschrift")            #LabelFrame definiert
lab = Label(labFr, text="Hallo", width=10, height=3) #Label im LabelFrame definiert
labFr.pack()                                            # Breite in Buchstabenbreite
lab.pack()                                              # Höhe in Zeile
root.mainloop()                                         #Dauerschleife
```

Menüleiste



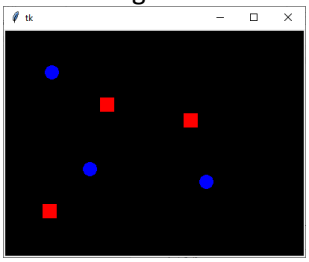

```
from tkinter import *                                   # Bibliothek laden
def leer():                                             # Testfunktion definieren
   return                                               # Rückgabe Testfunktion
def schliessen(event=None):                             # Funktion um Hauptfenster
   root.destroy()                                       # zu schließen(event für Alt-q)
root = Tk()                                             # Hauptfenster
root.bind("<Alt-q>",schliessen)                         # Eventhandler für  Alt- q
leiste = Menu(root)                                     # Menüleiste definieren
root.config(menu=leiste)
dateiMenu = Menu(leiste)                                # 1. Menüelement definieren
leiste.add_cascade(label="Datei",menu=dateiMenu)# 1. Menüelement zuordnen
dateiMenu.add_command(label="Neu",command=leer)      # 1. Untermenü
dateiMenu.add_command(label="Schliessen",command=schliessen,accelerator="Alt-q")
# 2. Untermenü mit Aufruf der Funktion zum Schließen des Hauptfensters mit Info
editMenu = Menu(leiste)                                 # 2. Menüelement definieren
leiste.add_cascade(label="Edit",menu=editMenu)   # 2. Menüelement zuordnen
root.mainloop()                                         # Programmschleife
```

Evanthandler
(Tastenkürzel)




```
from tkinter import *                                   # Bibliothek laden
def leer():                                             # Testfunktion definieren
   return                                               # Rückgabe Testfunktion
def schliessen(event=None):                             # Funktion um Hauptfenster
   root.destroy()                                       # zu schließen(event für Alt-q)
def RE(event=None):                                     # Funktion Rechteck
   can.create_rectangle(40,40,80,80,fill="green")       # Erzeugung Rechteck
def OV(event=None):                                     # Funktion Oval
   can.create_oval(40,40,80,80,fill="red")              # Erzeugung Kreis
root = Tk()                                             # Hauptfenster
can=Canvas(root,width=400,height=300,bg="black")# Erzeugung Zeichnungsebene
root.bind("<Alt-q>", schliessen)                        # Tastenkürzel für schliessen
root.bind("<r>",RE)                                     # Tastenkürzel für Recheck
root.bind("<o>",OV)                                     # Tastenkürzel für Kreis
leiste = Menu(root)                                     # Menüleiste definieren
root.config(menu=leiste)
dateiMenu = Menu(leiste)                                # 1. Menüelement definieren
leiste.add_cascade(label="Datei", menu=dateiMenu)# 1. Menüelement zuordnen
dateiMenu.add_command(label="Schliessen",command=schliessen,accelerator="Alt-q")
# 1. Untermenü mit Aufruf der Funktion zum Schließen des Hauptfensters mit Info
editMenu = Menu(leiste)                                 # 2. Menülelement definieren
leiste.add_cascade(label="Zeichne",menu=editMenu)# 2. Menüelemt zuordnen
editMenu.add_command(label="Rechteck",command=RE)
#  1. Untermenü mit Aufruf Funktion RE
editMenu.add_command(label="Kreis",command=OV)
#  2. Untermenü mit Aufruf Funktion OV
can.pack()                                              # Zeichenfläche sichtbar
root.mainloop()                                         # Endlosschleife
```

Mausabfrage



```
from tkinter import *                                   # Bibliothek laden
def maleKR(event):                                      # Funktion für Kreis zeichnen
   can.create_oval(event.x-10,event.y-10,event.x+10,event.y+10,fill="blue")
# Zeichne Kreis 10 Pixel um die aktuelle x und y-Position des Mauszeigers
def maleRE(event):                                      # Funktion f. Rechteck
   can.create_rectangle(event.x-10,event.y-10,event.x+10,event.y+10,fill="red")
# Zeichne Rechteck 10 Pixel um die aktuelle x und y-Position des Mauszeigers
root = Tk()                                             # Hauptfenster
can=Canvas(root,width=400,height=300,bg="black")  # Erzeugung Zeichenfläche
can.bind("<Button-1>", maleKR)                          # Abfrage auf linken Mausbutton
can.bind("<Button-3>", maleRE)                          # Abfrage auf rechten Mausbutton
can.pack()                                              # Zeichenfläche sichtbar
root.mainloop()                                         # Endlosschleife
```