1.1 Widgets

1.1.1 Button

Erzeugen:
```
bn = tkinter.Button(self)
```

Beschriftung:
```
self.bn["text"] = "Ok"
```

Event:
```
self.bn["command"] = self.onClickTheButton
```

Layout-Manager:
```
self.bn.pack oder self.bn.grid()
```

ActiveBackground:
```
self.bn.config(activebackground= "yellow")
```

ActiveForeground:
```
self.bn.config(activeforeground = "green")
```

Background:
```
self.bn. config(background = "green")
self.bn. config(background = "#FF0000")
```

Foreground:
```
self.bn.config(foreground="red")
```

Borderwidth:
```
self.bn.config(borderwidth=="2")  #pixel
```

Borderwidth:
```
self.bn.config(bd=pixel="2")
```

Height:
```
self.bn.config(height="2")          # Textzeilen
```

Justify:
```
self.bn.config(justify("left")
self.bn.config(justify("right")
self.bn.config(justify("justify")
```

overrelief:
```
self.bn.config(overrelief="raised")    Mouse Hover
sunken
flat
ridge
solid
groove
```

relief:
```
self.bn.config(relief="raised")
sunken
flat
ridge
solid
groove
```

state:	`self.bn.config(state="normal")`
	`active`
	`enabled`
	`disabled`

Takefokus:	`self.bn.config(takefocus=bool)` `Kann Fokus erhalten?`
Text:	`self.bn["text"] = Caption`
Textvariable:	`self.bn["textvariable"] = Caption` Property für den Schalter

1.1.2 Checkbutton

Checkbutton mit Property:

```
chk1 = tkinter.Checkbutton(self)
chk1["text"] = "Montag, 08:00 Uhr"

chkProperty = tkinter.BooleanVar()
chkProperty.set(True)
chk1["variable"] = chkProperty
```

Beispieldialog:

```
self.group = tkinter.LabelFrame(self, text="Noten")
self.group.pack(fill="both", expand="yes")
self.chk1= tkinter.Checkbutton(self)
self.chk1["text"] = "Eins"
self.chk1.pack()

self.chk2= tkinter.Checkbutton(self)
self.chk2["text"] = "Zwei"
self.chk2.pack()

self.chk3= tkinter.Checkbutton(self)
self.chk3["text"] = "Drei"
self.chk3.pack()

self.chkProperty1 = tkinter.BooleanVar()
self.chkProperty1.set(True)
self.chk1["variable"] = self.chkProperty1

self.chkProperty2 = tkinter.BooleanVar()
self.chkProperty2.set(False)
self.chk2["variable"] = self.chkProperty2

self.chkProperty3 = tkinter.BooleanVar()
self.chkProperty3.set(False)
self.chk3["variable"] = self.chkProperty3
```

```
print(str(self.chkProperty1.get()) )
print(str(self.chkProperty2.get()) )
print(str(self.chkProperty3.get()) )
# output =
   "Wert der CheckBox ist {}".format(self.chkProperty1.get())
print(output)
```

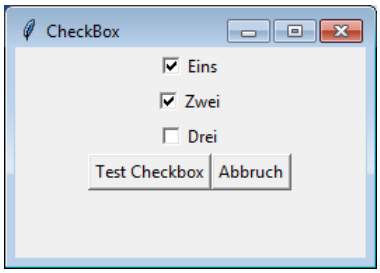

Abbildung 1 Beispieldialog mit drei Checkbutton und der Abfrage

1.1.3 Radiobutton

Es gibt eine Groupbox à la Delphi oder die Klasse Group aus Java. Alle RadioButton in einem Widget gehören derselben Gruppe an.

Beispiel:

```
rb1= tkinter.Radiobutton(self, text="Eins", value=" Eins ")
rb2= tkinter.Radiobutton(self, text="Zwei", value="Zwei")
rbProperty = tkinter.StringVar()
rbProperty.set("Zwei")
rb1["variable"] = rbProperty
rb2["variable"] = rbProperty
```

Mit GroupBox
self.group = tkinter.LabelFrame(self, text="Noten")

```
self.group.pack(fill="both", expand="yes")
self.rb1= tkinter.Radiobutton(self.group, text="Eins", value="one")
self.rb1.pack()
self.rb2= tkinter.Radiobutton(self.group, text="Zwei", value="two")
self.rb2.pack()
self.rb3= tkinter.Radiobutton(self.group, text="Drei", value="three")
self.rb3.pack()

self.rbProperty = tkinter.StringVar()
self.rbProperty.set("Drei")
self.rb1["variable"] = self.rbProperty
self.rb2["variable"] = self.rbProperty
self.rb3["variable"] = self.rbProperty
```

Abfrage:

```
    self.rbProperty.get()     # one,two, three
```

1.1.4 Entry (TextField)

Property eines Entry-Elements:

```
self.inputui = tkinter.Entry(self)
self.inputui.pack()

self.propertyInputui = tkinter.StringVar()
self.propertyInputui.set("Ihr Name...")
self.inputui["textvariable"] = self.propertyInputui

self.propertyInputui = "abc"
print(self.propertyInputui)
```

Eingabezeichen für Paßworteingabe:

```
self.inputui = tkinter.Entry(self)
self.inputui["show"] = "*"
```

Löschen des Inhalts:

```
self.inputui.delete(0,END)
```

Ausgabe des Inhalts:

```
print("Inhalt: %s" % (self.inputui.get()) )
print( self.inputui.get() )
```

1.1.5 Label

```
self.labelui = tkinter.Label(self)
self.labelui["text"] = "Vorname"
self.labelui.pack()     oder grid(…)
```

1.1.6 LabelFrame

Entspricht dem GroupBox in verschiedenen Frameworks.

Beispiel:

```
self.group = tkinter.LabelFrame(self, text="Noten")
self.group.pack(fill="both", expand="yes")
self.rb1= tkinter.Radiobutton(self.group, text="Eins", value="Eins")
self.rb1.pack()
self.rb2= tkinter.Radiobutton(self.group, text="Zwei", value="Zwei")
self.rb2.pack()
self.rb3= tkinter.Radiobutton(self.group, text="Drei", value="Drei")
self.rb3.pack()

self.rbProperty = tkinter.StringVar()
self.rbProperty.set("Drei")
self.rb1["variable"] = self.rbProperty
self.rb2["variable"] = self.rbProperty
self.rb3["variable"] = self.rbProperty
```

Hinweise:

- Bitte beachten Sie, dass die Radiobutton auch die Methode “pack” aufrufen müssen.
- Mit dem Befehl “`self.rbProperty.set("Drei")`” kann man ein RadioButton setzen bzw. abfragen.
  - `print(self.rbProperty.get())`

Beispieldialog:

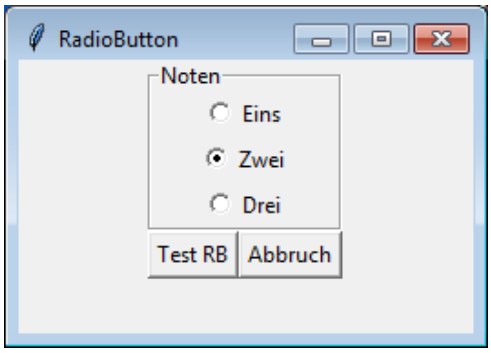

Abbildung 2 Dialog mit drei RadioButton und einer Abfrage

1.1.7 Listbox

```
listbox = tkinter.Listbox(self)
listbox.pack(fill="both", expand=True)

for i in range(20):
  listbox.insert("end", str(i))
for i in range(10):
  listbox.insert(0, str(i))
```

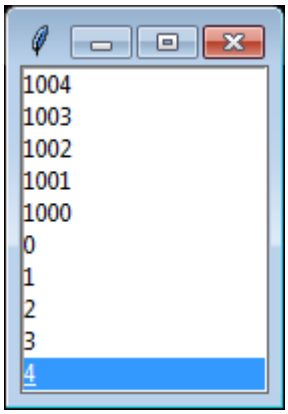

Abbildung 3 Listbox ohne Scrollbar

Weitere wichtige Methoden:

Methode	Bedeutung
curselection()	getSelectedItems
delete(first[,last])	Löschen von first bis last.
get(first[,last])	Rückgabe der Werte der Listbox von first bis last.
insert(index, *elements)	Einfügen eines oder mehrerer Elemente. • listbox.insert(0, "value") • listbox.insert("end", "value")
itemcget(index,option)	Gibt den Wert der Option “option” des Eintrags mit dem Index “index” zurück.
itemconfig(index,**option)	Setzt die Optionen
selection_clear(first[,last])	Löscht die Selection in der Liste die Einträge von first bis last.
selection_includes(index)	Ist index gesetzt? Rückgabe: True oder False
selection_set(first[,last])	Setzt die Selection in der Liste die Einträge von first bis last.
size()	Anzahl der Listenelemente.
selectmode	"single", "browse", "multiple", "extended". Single und Browse erlauben nur einen Wert zu markieren.
xscrollcommand	Anbindung einer Scrollbar
yscrollcommand	Anbindung einer Scrollbar

Events SelectedChange

```
self.listbox = tkinter.Listbox(self)
self.listbox.pack(fill="both", expand=True)

for i in range(20):
  self.listbox.insert("end", str(i))

self.listbox.bind("<<ListboxSelect>>", self.selectionChanged)
self.listbox.selection_set(0)

def selectionChanged:
  print( self.listbox.curselection())
```

Löschen aller Einträge einer Listbox:

```
self.listbox.delete(0,"end")
```

Einfügen eines Eintrags in eine Listbox:

```
self.listbox.insert("end", "Hallo")
```

1.1.8 Spinbox

Die Spinbox ist unter vielfätigem Name bekannt:

- Delphi: SpinEdit
- Java: JSpinner
- Winforms: NumericUpDown

Wie in Java, so kann man mit der Spinbox einen ganzzahligen Zahlenbereich auswählen; aber auch ihn wie eine Listbox benutzen.

Methoden:

Methode	**Wertebereich**	**Erläuterung**
format	"%pad1.pad2"	Bekannte C-Format
from	float	Untere Grenze
increment	float	Increment beim Klick der kleinen Pfeile
to	float	Obere Grenze
values	tuple oder list	Feste Werte
xscrollcommand	Delegate	Für eine Scrollbar

Beispiele:

```
sp = tkinter.Spinbox(self)
s["from"] = 0
s["to"] = 100
s.pack

sp = tkinter.Spinbox(self)
s["values"] = (0,1,1,2,3,5,8,13,21,34,55,89)
s["values"] = ("A","B","C")
s.pack
```

1.1.9 Text

Die Textkomponente entspricht eher einer RTF-Komponente. In ihr kann man formatierten und farbigen Text darstellen, aber auch Bilder platzieren. Es ist leider nicht möglich, eine Steuerelementvariable resp. Property für das Textelement einzurichten. Man kann aber den Inhalt abfragen.

Einfügen von Texten:

- "insert"
- "current"
- "end"

Parameter:

- height=2 Zwei Zeilen
- width=30 Dreißig Buchstaben Breite

Beispiel:

```
self.editor = tkinter.Text(self)
self.editor.grid(expand=True, fill="both", padx="5",pady="5")
self.editor.insert("end","\r\nonAction\r\n")
```

Textwrapping:

- self.editor.config(wrap="none")
- self.editor.config(wrap="word") Umbrechen beim letzten Wort
- self.editor.config(wrap="char")

Löschen aller Einträge eines Text-Elementes:

```
self. editor.delete(0,"end")
```

Einfügen eines Eintrags in eine Text-Elementes:

```
self.editor.insert("end", "Hallo\r\n")
```

Abfragen des Inhaltes:

```
input = self.myText_Box.get("1.0",END)
        1.0     1. Linie bis zum Ende
input = self.myText_Box.get("1.0",'end-1c')
        1.0     1. Linie bis zum Ende abzüglich eines Zeichens
input = self.myText_Box.get("1.0",'end-2c')
        1.0     1. Linie bis zum Ende abzüglich zweier Zeichens
```

1.1.10 Tree

Der Tree funktioniert genauso wie andere UI-Tree-Elemente. Beim Einfügen wird ein Parent-Konto benötigt, und als Rückgabewert erhält man einen Knoten.

Besonderheit:

- Man kann Spaltenüberschriften (heading) oben einfügen.

Beispiel ohne Spalten:

```
self.tree = tkinter.ttk.Treeview(frame1)
self.tree.pack(fill="both",expand=True)
self.tree["xscrollcommand"] = xsb.set    #scrolling
xsb["command"]=self.tree.xview           #scrolling
ysb["command"]=self.tree.yview           #scrolling

# 1. Knoten anlegen mit Spalten(Values)
root_node = self.tree.insert('', '1', text='Ordner',
```

```
            values=("13-Jun-19 11:28","Verz","-") open=True)

    # subKnoten anlegen
    node=self.tree.insert(root_node , 'end', text='abc', open=False)
```

Beispiel mit Spalten:

```
    self.tree = tkinter.ttk.Treeview(frame1)
    self.tree.pack(fill="both",expand=True)
    self.tree["xscrollcommand"] = xsb.set
    xsb["command"]=self.tree.xview
    ysb["command"]=self.tree.yview

    self.tree["columns"]=("one","two","three")
    self.tree.column("#0", width=70, minwidth=27, stretch=tkinter.NO)
    self.tree.column("one", width=90, minwidth=50, stretch=tkinter.NO)
    self.tree.column("two", width=60, minwidth=45)
    self.tree.column("three", width=80, minwidth=50, stretch=tkinter.NO)

    self.tree.heading("#0",text="Name",anchor=tkinter.W)
    self.tree.heading("one", text="Date modified",anchor=tkinter.W)
    self.tree.heading("two", text="Type",anchor=tkinter.W)
    self.tree.heading("three", text="Size",anchor=tkinter.W)

    # 1. Knoten anlegen mit Spalten(Values)
    root_node = self.tree.insert('', '1', text='Ordner',
            values=("13-Jun-19 11:28","Verz","-") open=True)

    # subKnoten anlegen
    node=self.tree.insert(root_node , 'end', text='abc',
            values=("13-Jun-19 11:29","jpg","12.444") open=False)
```

Kompletes Beispiel:

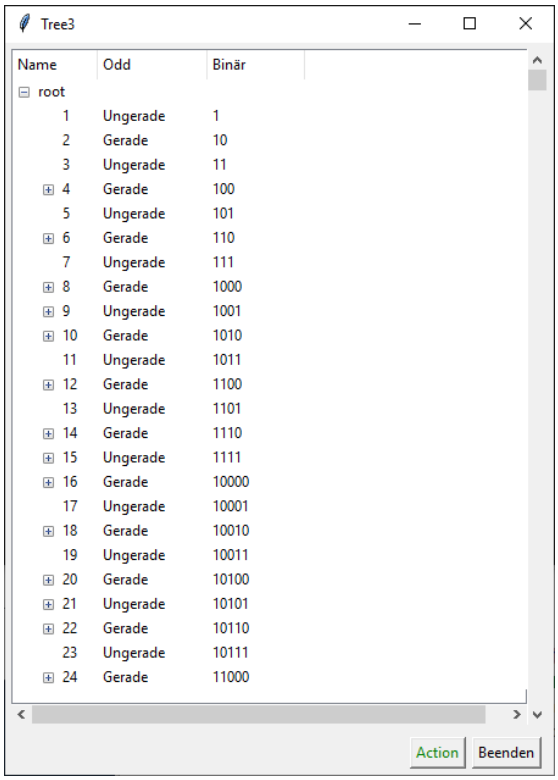

Abbildung 4 Tree-Beispiel mit Spalten

Quellcode:

```
#! /usr/bin/env python3
# coding=utf8

import os
import tkinter
from tkinter.ttk import *
from tkinter import messagebox

class MyApp(tkinter.Frame):

   def __init__(self, master):
      tkinter.Frame.__init__(self, master)
      self.pack(expand=True, fill="both") #
      self.setGUI()

   def setGUI(self):
      # pack ist wie das DockPanel in WPF
      # erst muessen die "normalen" UI-Elemente eintgetragen werden
      # am Schluss werden die UI-Elemente eingetragen, die fill="both" haben
      buttonframe = tkinter.Frame(self)
      buttonframe.pack(fill="x", side="bottom", padx="5", pady="5")

      self.bnEsc = tkinter.Button(buttonframe)
      self.bnEsc["text"] = "Beenden"
      self.bnEsc["command"] = self.quit
      self.bnEsc.pack(padx="5", side="right")
```

```
        self.bnAction = tkinter.Button(buttonframe)
        self.bnAction["text"] = "Action"
        self.bnAction["command"] = self.onAction
        self.bnAction.pack(side="right")
        self.bnAction.config(foreground= "green")
        self.bnAction.config(activebackground= "yellow")

        frame1 = tkinter.Frame(self)
        frame1.pack(expand=True, fill="both", side="top", padx="5", pady="5")

        ysb = tkinter.Scrollbar(frame1) # , command=self.tree.yview
        ysb.pack(fill="y",side="right")

        xsb = tkinter.Scrollbar(frame1, orient="horizontal")
        xsb.pack(fill="x",side="bottom")

        self.tree = tkinter.ttk.Treeview(frame1)
        self.tree.pack(fill="both",expand=True)
        self.tree["xscrollcommand"] = xsb.set
        xsb["command"]=self.tree.xview
        ysb["command"]=self.tree.yview

        self.tree["columns"]=("one","two","three")
        self.tree.column("#0", width=70, minwidth=27, stretch=tkinter.NO)
        self.tree.column("one", width=90, minwidth=45)
        self.tree.column("two", width=80, minwidth=50, stretch=tkinter.NO)

        self.tree.heading("#0",text="Name",anchor=tkinter.W)
        self.tree.heading("one", text="Odd",anchor=tkinter.W)
        self.tree.heading("two", text="Binär",anchor=tkinter.W)

        root_node = self.tree.insert('', 'end', text='root', open=True)
        self.insertTree(root_node)
        #self.tree.insert(root_node, 'end', text='Line 1', open=False)

    def insertTree(self,parent):
        n=100
        for i in range(1,n+1,1):
            if i%2==0:
                typ="Gerade"
            else:
                typ="Ungerade"
            binaer=int(bin(i)[2:])
            node=self.tree.insert(parent , 'end',values=(typ,binaer) ,
                        text=str(i), open=False)
            for j in range(2,i,1):
                if i%j==0:
                    self.tree.insert(node , 'end', text=str(j), open=False)


    def onAction(self):
        messagebox.showinfo( "Hello Python", "action")

root = tkinter.Tk()
root.title("Tree3")
root.geometry("450x600")
app = MyApp(root)
```

```
app.mainloop()
```

1.1.11 Menu

- Das Menü besteht wie in Java aus einem MenuBar, hier aber Menu benannt.
- Die Menüs werden über das Widget “Menu” dargestellt.
- Die Menüeinträge warden über “add_command” realisiert.
- Eingetragen warden die Menüeinträge mitttels des Befehls “add_cascade”.

Option	**Beschreibung**
add_command (options)	Hinzufügen eines Menüeintrags
add_radiobutton(options)	Hinzufügen eines Menüeintrags mit Radiobutton
add_checkbutton(options)	Hinzufügen eines Menüeintrags mit check button
add_cascade(options)	Einfügen eines Hauptmenüs oder Submenüs.
add_separator()	Einfügen eines Separators
add(type, options)	Hinzufügen von Typen von Menüs.
delete(startindex [, endindex])	Löschen von Menüeinträge
entryconfig(index, options)	Möglichkeit zur Modifizierung.
index(item)	Ausgabe des Index eines Menüeintrags.
insert_separator (index)	Einfügen eines Separator
invoke (index)	Aufruf der Event-Methode. Setzt die Checkbox oder Radiobutton.
type (index)	Rückgabe des Types: • "cascade" • "checkbutton" • "command" • "radiobutton" • "separator" • "tearoff".

Beispiel:

```
self.menuBar = tkinter.Menu(self)
master.config(menu=self.menuBar)
self.menuFile = tkinter.Menu(self.menuBar, tearoff=False)
self.menuFile.add_command(label="Öffnen", command=self.openFile)
self.menuFile.add_command(label="Speichern", command=self.saveFile)
self.menuFile.add_command(label="Speichern unter", command=self.saveasFile)
self.menuFile.add_separator()

self.menuSubFile = tkinter.Menu(self.menuBar, tearoff=False)
self.menuSubFile.add_command(label="Sub1")
self.menuSubFile.add_command(label="Sub2")
self.menuSubFile.add_command(label="Sub3")
self.menuFile.add_cascade(label="Sub1-3", menu=self.menuSubFile)
```

```
self.menuFile.add_command(label="Beenden", command=self.quit)
self.menuBar.add_cascade(label="Datei", menu=self.menuFile)

self.menuEdit = tkinter.Menu(self.menuBar, tearoff=False)
self.menuEdit.add_command(label="Einfügen", command=self.insert)
self.menuEdit.add_command(label="Kopieren", command=self.copy)
self.menuBar.add_cascade(label="Edit", menu=self.menuEdit)
```

Erzeugen der MenuBar:
```
self.menuBar = tkinter.Menu(self)
```

Einfügen in das Dialog:
```
master.config(menu=self.menuBar)
```

Erzeugen eines Hauptmenüs:
```
self.menuFile = tkinter.Menu(self.menuBar, tearoff=False)
```
Mit tearoff kann man das Menü vom fenster lösen.

Erzeugen eines Menüeintrags mit onClick-Event:
```
self.menuFile.add_command(label="Öffnen", command=self.openFile)
```

Einfügen in den Menübar
```
self.menuBar.add_cascade(label="Datei", menu=self.menuFile)
```

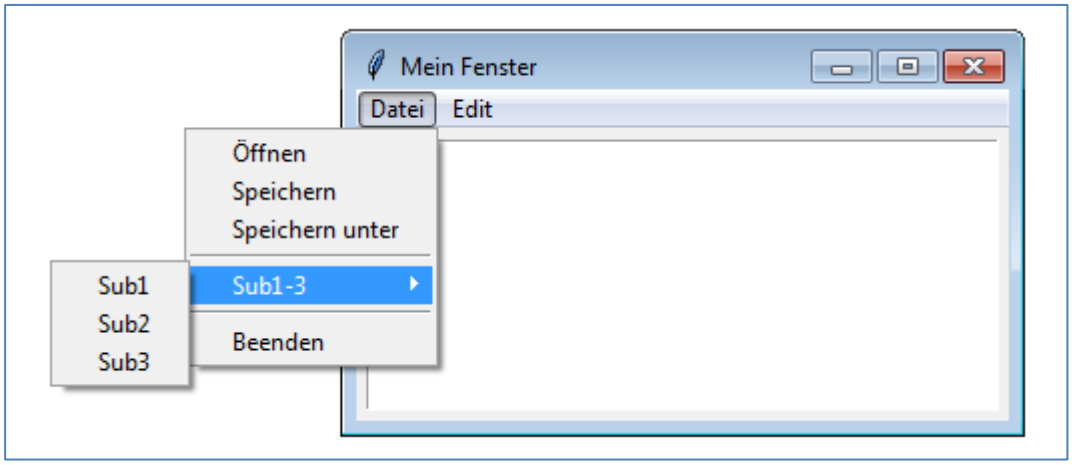

Abbildung 5 Menü mit Submenüs unter TKinter

Kompletter Quellcode:

```
import tkinter

class MyApp(tkinter.Frame):

   def __init__(self, master=None):
      tkinter.Frame.__init__(self, master)
      self.menuBar = tkinter.Menu(self)
      master.config(menu=self.menuBar)
      self.setGUI()
      self.pack()
```

```
   def setGUI(self):
      self.menuFile = tkinter.Menu(self.menuBar, tearoff=False)
      self.menuFile.add_command(label="Öffnen", command=self.openFile)
      self.menuFile.add_command(label="Speichern", command=self.saveFile)
      self.menuFile.add_command(label="Speichern                    unter",
command=self.saveasFile)
      self.menuFile.add_separator()

      self.menuSubFile = tkinter.Menu(self.menuBar, tearoff=False)
      self.menuSubFile.add_command(label="Sub1")
      self.menuSubFile.add_command(label="Sub2")
      self.menuSubFile.add_command(label="Sub3")
      self.menuFile.add_cascade(label="Sub1-3", menu=self.menuSubFile)

      self.menuFile.add_separator()
      self.menuFile.add_command(label="Beenden", command=self.quit)
      self.menuBar.add_cascade(label="Datei", menu=self.menuFile)

      self.menuEdit = tkinter.Menu(self.menuBar, tearoff=False)
      self.menuEdit.add_command(label="Einfügen", command=self.openFile)
      self.menuEdit.add_command(label="Kopieren", command=self.saveFile)
      self.menuBar.add_cascade(label="Edit", menu=self.menuEdit)

      self.editor = tkinter.Text(self)
      self.editor.pack(padx="5",pady="5")

   def openFile(self):
      self.editor.insert("end","openFile\r\n")

   def saveFile(self):
      self.editor.insert("end","saveFile\r\n")

   def saveasFile(self):
      self.editor.insert("end","saveasFile\r\n")


root = tkinter.Tk()
root.title("Mein Fenster")
root.geometry("250x300")
app = MyApp(root)
app.mainloop()
```

Um Checkboxen in einem Menü zu integrieren, benutzt man folgenden Code:

```
self.menuFont = tkinter.Menu(self.menuBar, tearoff=False)
self.menuFont.add_checkbutton(label="Fett", command=self.bold)
self.menuFont.add_checkbutton(label="Kursiv", command=self.italic)
self.menuBar.add_cascade(label="Schrift", menu=self.menuFont)
```

Um Radiobuttons in ein Menü zu integrieren, benutzt man folgenden Code:

```
   self.menuFont.add_separator()
   self.menuFont.add_radiobutton(label="Arial", command=self.arial)
   self.menuFont.add_radiobutton(label="Helvitica", command=self.helvitica)
   self.menuFont.add_radiobutton(label="Verdana", command=self.verdana)
```

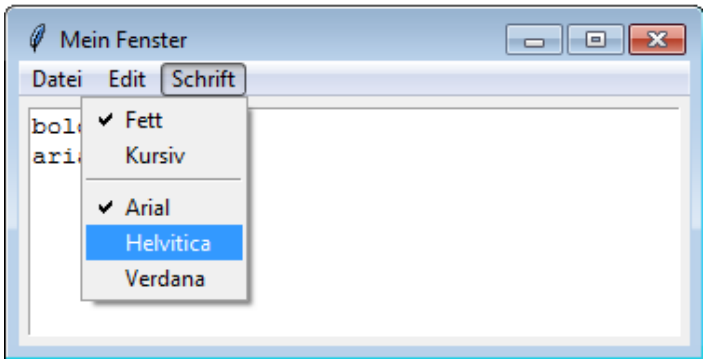

Abbildung 6 Menü mit zwei Checkbox und drei RadioButtons

1.1.12 Scrollbar

```
sb = tkinter.Scrollbar(self)
sb.pack(fill="y", side="right")

listbox = tkinter.Listbox(self)
listbox.pack(fill="both", expand=True)
listbox["yscollcrommand"] = sb.set
sb["command"] = listbox.yview

for i in range(20):
   listbox.insert("end", str(i))
for i in range(10):
   listbox.insert(0, str(i))
```

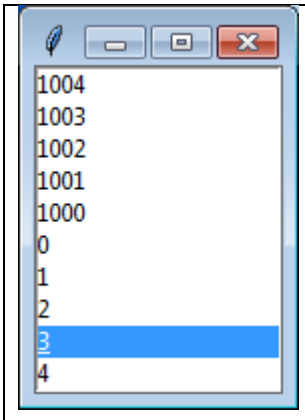

Abbildung 7 Listbox ohne Scrollbar

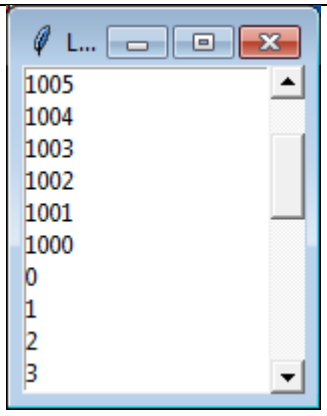

Abbildung 8 Listbox mit Scrollbar

Erzeugen einer Scrollbars:

```
sbx = tkinter.Scrollbar(frame, orient="horizontal")
sbx.pack(fill="x", side="bottom")

sby = tkinter.Scrollbar(frame)
sby.pack(fill="y", side="right")
```

1.2 Standarddialoge

Mit dem Modul „filedialog“ werden die von anderen Frameworks bekannten Standarddialoge zur Verfügung gestellt.

1.2.1 Meldungen

Um die unteren Funktionen aufzurufen, benötigen Sie folgende Import-Anweisung:

```
from tkinter import messagebox
```

1.2.1.1 showinfo

Beispiel:

```
messagebox.showinfo( "Meldung",
                "Sie erhalten eine 1,0 als Note",icon='info')
```

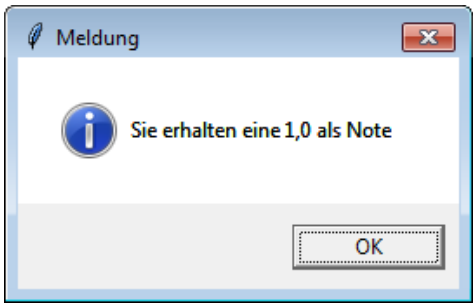

Abbildung 9 ShowInfo unter Python

Folgende Symbole sind möglich:

- error
- warning
- info
- question

1.2.1.2 showwarning

Diese Variante ist eigentlich nur showinfo mit einem festen Symbol (warning)
Beispiel:

```
messagebox.showwarning( "Hello Python", "Hello World")
```

Abbildung 10 ShowWarning unter Python

Folgende Symbole sind möglich:

- error
- warning
- info
- question

1.2.1.3 showerror

Diese Variante ist eigentlich nur showinfo mit einem festen Symbol (error)
Beispiel:

```
messagebox.showerror( "Hello Python", "Hello World")
```

Abbildung 11 ShowError unter Python

Folgende Symbole sind möglich:

- error
- warning
- info
- question

1.2.2 Abfragen

Um die unteren Funktionen aufzurufen, benötigen Sie folgende Import-Anweisung:

```
from tkinter import messagebox
```

1.2.2.1 askokcancel

Beispiel:

```
messagebox.askokcancel( "Note", "Sie erhalten eine 1,0")
```

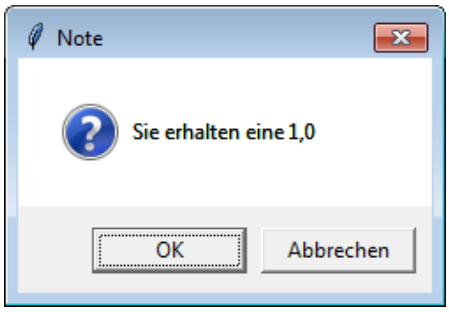

Abbildung 12 AskOkCancel unter Python

Folgende Symbole sind möglich:

- error
- warning
- info
- question

1.2.2.2 askquestion

Entspricht eigentlich dem yesno.

Beispiel:

```
messagebox.askquestion ( "Note", "Wollen Sie eine 1,0?")
```

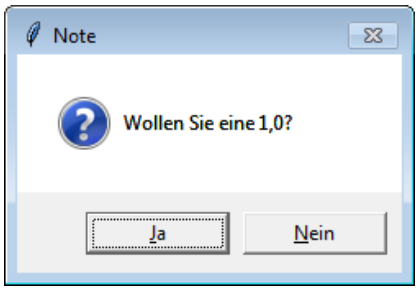

Abbildung 13 AskQuestion unter Python

Folgende Symbole sind möglich:

- error
- warning
- info
- question

Mit default kann man den Standardschalter definieren:

- `messagebox.askokcancel( "Note", "Sie erhalten eine 1,0", default='yes')`
- `messagebox.askokcancel( "Note", "Sie erhalten eine 1,0", default='no')`

1.2.2.3 askyesno

Beispiel:

```
result = messagebox.askyesno("Delete", "Are You Sure?",
                             icon='question', default='no')
if result == True:
  messagebox.showinfo("Deleted","Deleted")
else:
  messagebox.showinfo("Not Deleted","Not Deleted")
```

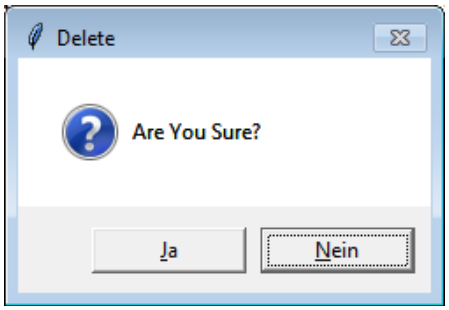

Abbildung 14 AskYesNo unter Python

Folgende Symbole sind möglich:

- error
- warning
- info
- question

Mit default kann man den Standardschalter definieren:

- `messagebox.askokcancel( "Note", "Sie erhalten eine 1,0", default='yes')`
- `messagebox.askokcancel( "Note", "Sie erhalten eine 1,0", default='no')`

1.2.2.4 askyesnocancel

Beispiel:

```
result = messagebox.askyesnocancel( "Beenden",
                                    "Änderungen speichern?")
```

```
if result == True:
  messagebox.showinfo("Beenden","Save")
elif result == False:
  messagebox.showinfo("Beenden","No Save")
elif result == None:
  messagebox.showinfo("Beenden","No Close")
else:
  messagebox.showinfo("Beenden","Fehlerhafte Abfrage")
```

Eine switch/Case-Anweisung gibt es in Python nicht.

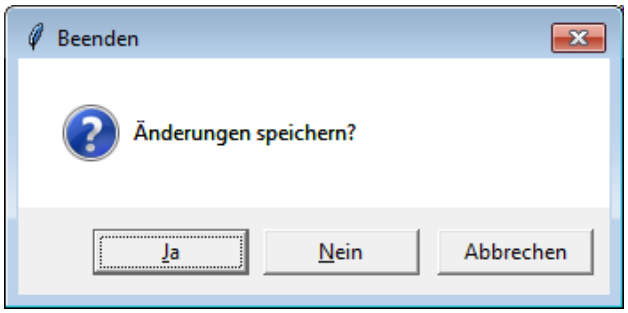

Abbildung 15 AskYesNoCancel unter Python

Folgende Symbole sind möglich:

- error
- warning
- info
- question

Mit default kann man den Standardschalter definieren:

- `messagebox.askokcancel( "Note", "Sie erhalten eine 1,0", default='yes')`
- `messagebox.askokcancel( "Note", "Sie erhalten eine 1,0", default='no')`

1.2.2.5 askretrycancel

Beispiel:

```
result = messagebox.askretrycancel( "Datei öffnen",
                  "Fehler beim Öffnen, noch einmal versuchen?")
if result == True:
  messagebox.showinfo("Datei öffnen","Noch einmal")
else:
  messagebox.showinfo("Datei öffnen",
                                "Ende, kein weiterer Versuch")
```

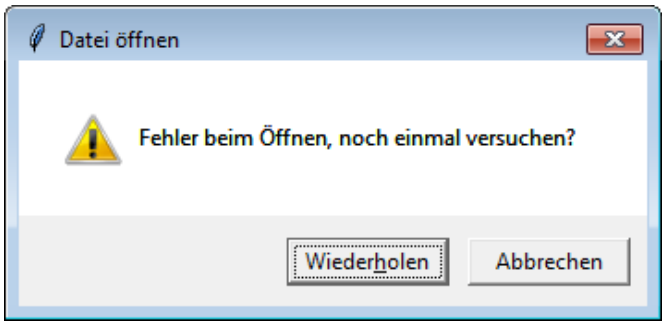

Abbildung 16 AskYesNoCancel unter Python

Folgende Symbole sind möglich:

- error
- warning
- info
- question

Mit default kann man den Standardschalter definieren:

- `messagebox.askokcancel( "Note", "Sie erhalten eine 1,0", default='yes')`
- `messagebox.askokcancel( "Note", "Sie erhalten eine 1,0", default='no')`

1.2.3 Eingabe-Dialoge

Um die unteren Funktionen aufzurufen, benötigen Sie folgende Import-Anweisung:

```
from tkinter import simpledialog
```

1.2.3.1 askstring

Beispiel:

```
result = simpledialog.askstring( "Hello Python", "Hello World")
if result == None:
  messagebox.showinfo("Eingabe","keine")
else:
  messagebox.showinfo("Eingabe",result)
```

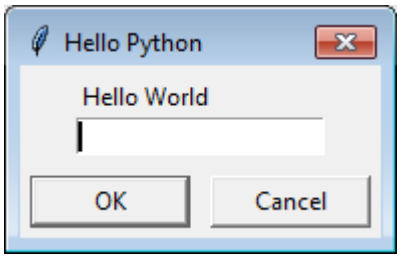

Abbildung 17 AskString unter Python

1.2.3.2 askint

Beispiel:

```
result = simpledialog.askinteger( "Hello Python", "Hello World",
                      initialvalue=42, minvalue=1, maxvalue=99)
if result == None:
  messagebox.showinfo("Eingabe","keine")
else:
  messagebox.showinfo("Eingabe",result)
```

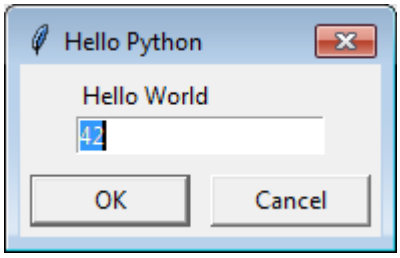

Abbildung 18 AskInt unter Python

1.2.3.3 askfloat

Beispiel:

```
result = simpledialog.askfloat( "Hello Python", "Hello World",
              initialvalue=42, minvalue=1, maxvalue=99)
if result == None:
  messagebox.showinfo("Eingabe","keine")
else:
  messagebox.showinfo("Eingabe",result)
```

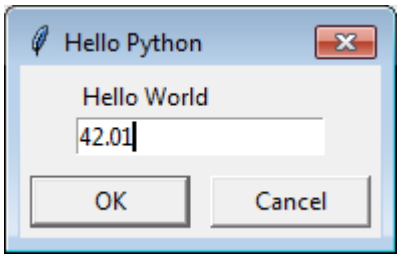

Abbildung 19 AskFloat unter Python

1.2.4 File-Dialoge

Um die unteren Funktionen aufzurufen, benötigen Sie folgende Import-Anweisung:

```
from tkinter import filedialog
```

1.2.4.1 OpenFile-Dialog

Aufruf:

- filedialog.askopenfilename

Optionen:

- title
  - title="Dateien laden"
- initialdir
  - initialdir="D:\\progs\\Python\\UI\\"
- multiple
  - Mehrfachauswahl
  - multiple=True
- filetypes
  - Dateimasken
  - filetypes=(("Pythondateien",".py") ,("Alle Dateien","*.*")))

Beispiel:

```
filename = filedialog.askopenfilename(title="Dateien laden",
   initialdir="D:\\progs\\Python\\UI\\", multiple=True,
   filetypes=( ("Pythondateien",".py" ),("Alle Dateien","*.*" ) ) )
if filename:
  messagebox.showinfo("Auswahl",filename)
else:
  messagebox.showinfo("Auswahl","Abbruch")
```

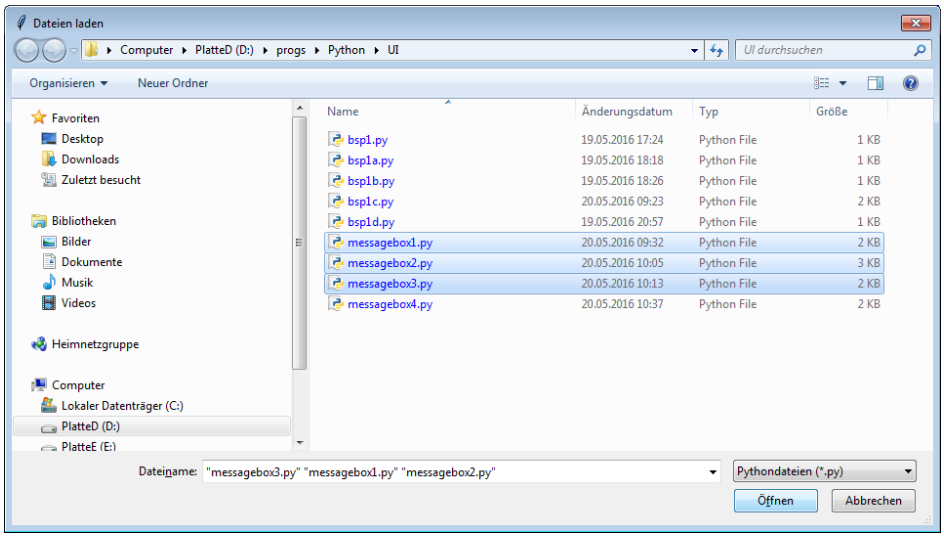

Abbildung 20 Auswahl von Dateien

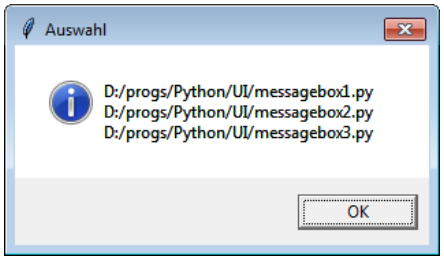

Abbildung 21 Ausgabe der ausgewählten Dateien

1.2.4.2 SaveAsFile-Dialog

Aufruf:

- filedialog.asksavefilename

Optionen:

- title
  - title="Dateien speichern"
- initialdir
  - initialdir="D:\\progs \\Python\\UI\\"
- multiple
  - Mehrfachauswahl
  - multiple=True
- filetypes
  - Dateimasken
  - filetypes=(("Pythondateien",".py") ,("Alle Dateien","*.*")))

Beispiel:

```
filename = filedialog.asksaveasfilename(initialfile="Beispiel.py",
    title="Dateien laden",
   initialdir="D:\\progs\\Python\\UI\\",
   filetypes=( ("Pythondateien",".py" ),("Alle Dateien","*.*" ) ) )
if filename:
  messagebox.showinfo("Auswahl",filename)
else:
  messagebox.showinfo("Auswahl","Abbruch")
```

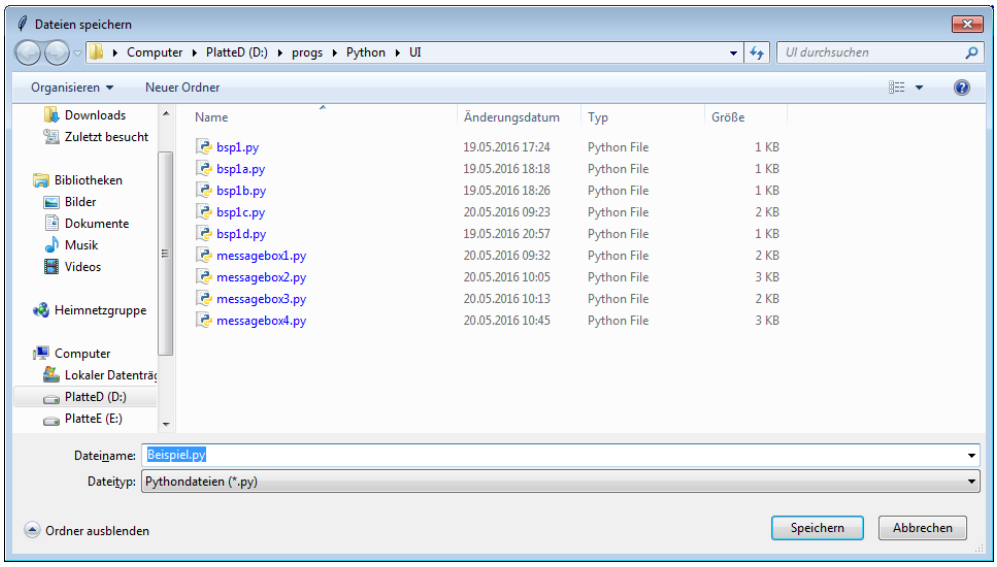

Abbildung 22 Speichern einer Dateien

1.2.4.3 Directory-Dialog

Aufruf:

- filedialog.askdirectory

Optionen:

- title
  - title="Dateien speichern"
- initialdir
  - initialdir="D:\\progs\\Python"
- mustexists
  - wenn true, muss das Verzeichnis existieren

Beispiel:

```
filename = filedialog.askdirectory(title="Verzeichnis auswählen",
                                   initialdir="D:\\progs\\Python" )
```

```
if filename:
   messagebox.showinfo("Auswahl",filename)
else:
   messagebox.showinfo("Auswahl","Abbruch")
```

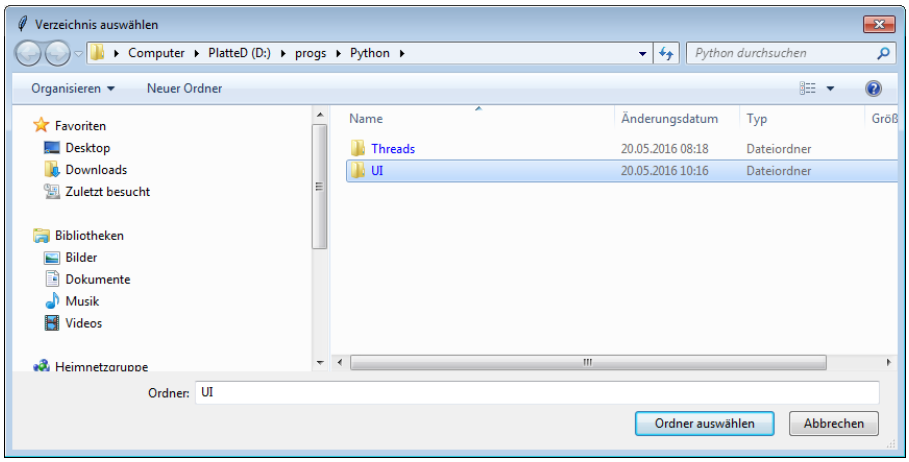

Abbildung 23 Auswahl eines Verzeichnisses

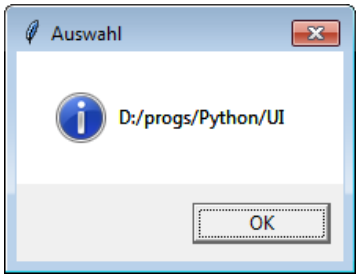

Abbildung 24 Anzeige des ausgewählten Verzeichnisses