1 Gridlayout

Das Layout mit dem “pack”-Mechanismus ist nicht immer leicht zu verstehen. Deshalb gibt es seit Jahren einen Nachfolger: das Grid-Layout.

Beachte:

- **Einmal Grid-Layout immer Grid-Layout!**

1.1 Aufbau

Der Aufbau muss natürlich etwas modifiziert werden.

```
import tkinter

class MyApp():

   def __init__(self, root):
      root.config(bg="yellow")
      self.setGUI()

   def setGUI(self):
      pass

root = tkinter.Tk()
root.title("Mein Fenster")
root.geometry("300x300")
app = MyApp(root)
root.mainloop()
```

Neu:

- Die Referenz des Dialogs „root“ wird nun per Parameter übergeben.
- Beim Erzeugen eines UI-Elementes muss man nun root statt self eingeben.
- Für „Ok“ und „Quit“ verwendet man gleichfalls root
  - `self.bnOk["command"] = root.quit`
- Für eigene Action-Methoden verwendet man self.
  - `self.bnRev["command"] = self.onReverse`
- Statt „pack“ fügt man nun ein Element mit ?.grid(…) ein
  - `.grid(row=0, column=0, columnspan=3, padx="10",pady="10")`
  - `cow`
  - `col`
  - `columnspan`
  - `rowspan`
- Die Anzahl der Zeilen und Spalten werden automatisch ermittelt.

1.2 Einfaches Beispiel:

```
import tkinter
from tkinter import messagebox

class MyApp():

   def __init__(self, root):
      root.config(bg="yellow")
      self.setGUI()

   def setGUI(self):
      n=5
      for rownr in range(0,n,1):
         for colnr in range(0,n,1):
            bn = tkinter.Button(root)
            bn["text"] = str(rownr)+' / '+str(colnr)
            #bn.grid(row=0, column=0, padx="10",pady="10")
            bn.grid(row=rownr, column=colnr, padx="10",pady="10")
            bn.config(foreground = "blue" )
            bn.config(background = "red") #"#FF0000"


root = tkinter.Tk()
root.title("Mein Fenster")
root.geometry("300x300")
app = MyApp(root)
root.mainloop()
```

Ergebnis:

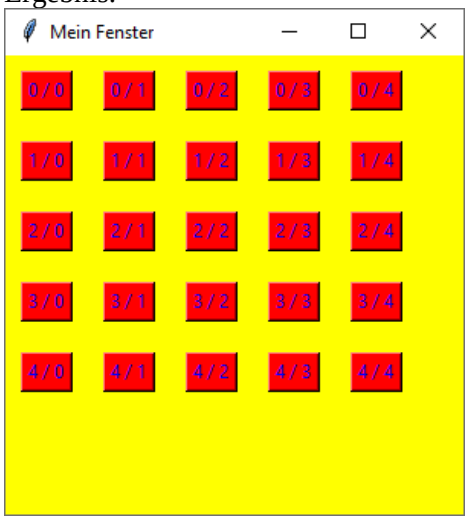

Abbildung 1 Grid-Layout mit Schaltern

1.3 Beispiel mit dem Vorgabegridlayout aus dem Wizzard:

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

import tkinter

class MyApp():

   def __init__(self, root):
      root.config(bg="yellow")
      self.setGUI()

   def setGUI(self):
      self.label1 = tkinter.Label(root)
      self.label1["text"]="Eingabe"
      self.label1.config(foreground="blue")
      self.label1.config(background="red")
      self.label1.grid(row=0, column=0, padx="10", pady="10")

      self.inputui = tkinter.Entry(root, background="blue",
             relief=tkinter.SUNKEN)
      self.inputui.grid(row=0, column=1, sticky="ew",
              padx="10", pady="10")

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

      self.bnAction = tkinter.Button(root)
      self.bnAction["text"] = "Action"
      self.bnAction["command"] = self.onAction
      self.bnAction.grid(row=1, column=0, padx="10", pady="10")

      self.bnQuit = tkinter.Button(root)
      self.bnQuit["text"] = "Ende"
      self.bnQuit["command"] = root.quit
      self.bnQuit.grid(row=1, column=1, padx="10", pady="10")

   def onAction(self):
      self.var_name.set(self.var_name.get()[::- 1])
      messagebox.showinfo("Hello Python", "Hello World")

root = tkinter.Tk()
root.title("Mein Fenster")
root.geometry("350x200")
app = MyApp(root)
root.mainloop()
```

Anzeige

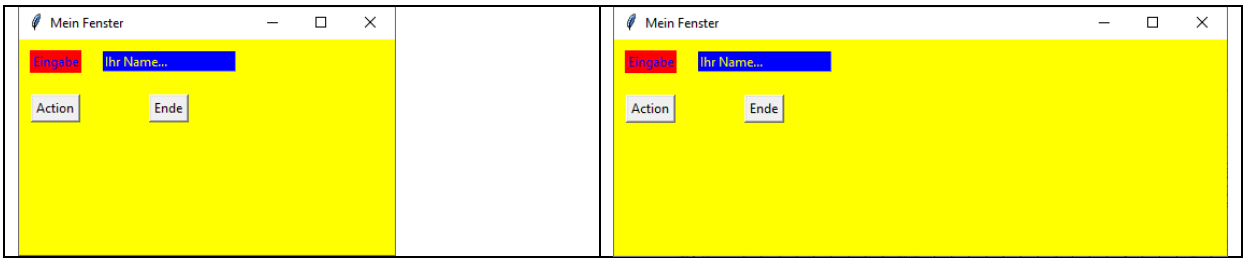

Abbildung 2 Anzeige der Vorgabe des GridLayouts

Nachteil:

- Das Eingabefeld wird nicht mit vergrößert resp. verkleinert.
- Die Schalter sind in jeweils einer Spalte eingetragen. Besser wäre ein Frame mit zwei Schaltern.

Umbau:

```
def setGUI(self):
    root.columnconfigure(1, weight=1)                    # setzt die 1. Spalte auf fill
```

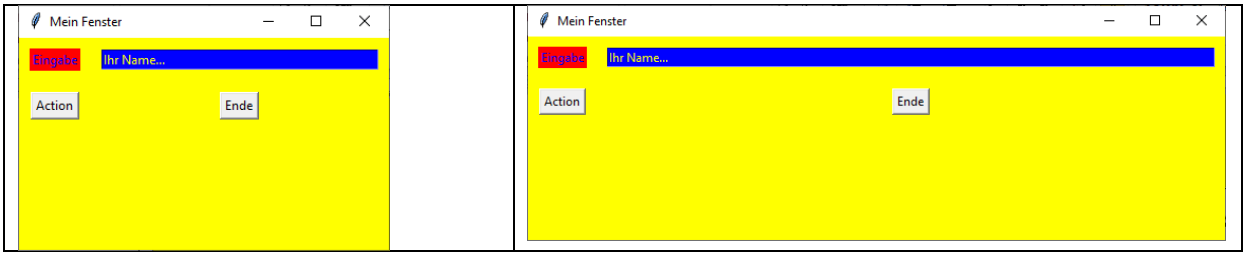

Abbildung 3 Anzeige der Vorgabe des GridLayouts

Umbau:

```
buttonframe = tkinter.Frame(root)
buttonframe.config(background = "blue") #"#FF0000"
buttonframe.grid(row=1, column=0, columnspan=2, padx="10",pady="10")

self.bnAction = tkinter.Button(buttonframe)
self.bnAction["text"] = "Action"
self.bnAction["command"] = self.onAction
self.bnAction.grid(row=0, column=0, padx="10", pady="10")

self.bnQuit = tkinter.Button(buttonframe)
self.bnQuit["text"] = "Ende"
self.bnQuit["command"] = root.quit
self.bnQuit.grid(row=0, column=1, padx="10", pady="10")
```

Ergebnis (Layout_grid_bsp3.py):

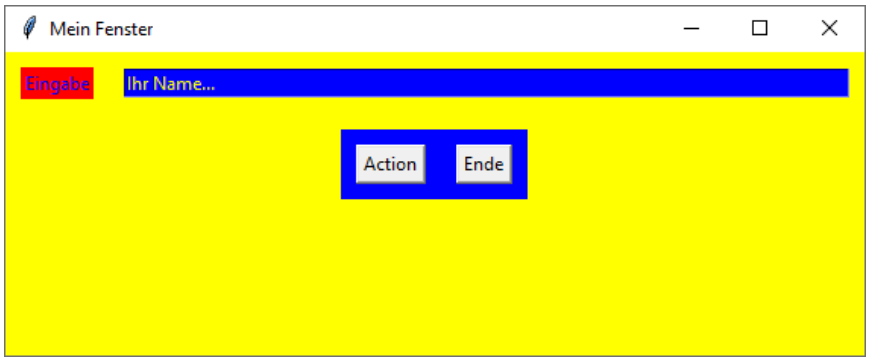

Abbildung 4 Anzeige der Vorgabe des GridLayouts mit einem „buttonframe“

Vollständiger Quellcode:

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

import tkinter
from tkinter import messagebox

class MyApp():

  def __init__(self, root):
    root.config(bg="yellow")
    self.setGUI()

  def setGUI(self):
    root.columnconfigure(1, weight=1)  # setzt die 1. Spalte auf fill

    self.label1 = tkinter.Label(root)
    self.label1["text"]="Eingabe"
    self.label1.config(foreground="blue")

    self.label1.config(background="red")
    self.label1.grid(row=0, column=0, padx="10", pady="10")
    self.inputui = tkinter.Entry(root, background="blue",
foreground="yellow", relief=tkinter.SUNKEN)
    self.inputui.grid(row=0, column=1, sticky="ew", padx="10",
pady="10")

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

    buttonframe = tkinter.Frame(root)
    buttonframe.config(background = "blue") #"#FF0000"
    buttonframe.grid(row=1, column=0, columnspan=2, padx="10",pady="10")


    self.bnAction = tkinter.Button(buttonframe)
    self.bnAction["text"] = "Action"
    self.bnAction["command"] = self.onAction
    self.bnAction.grid(row=0, column=0, padx="10", pady="10")
```

```
        self.bnQuit = tkinter.Button(buttonframe)
        self.bnQuit["text"] = "Ende"
        self.bnQuit["command"] = root.quit
        self.bnQuit.grid(row=0, column=1, padx="10", pady="10")

    def onAction(self):
        self.var_name.set(self.var_name.get()[::- 1])
        messagebox.showinfo("Hello Python", "Hello World")

root = tkinter.Tk()
root.title("Mein Fenster")
root.geometry("350x200")
app = MyApp(root)
root.mainloop()
```

1.4 Komplexes Beispiel (Layout_grid_bsp4.py)

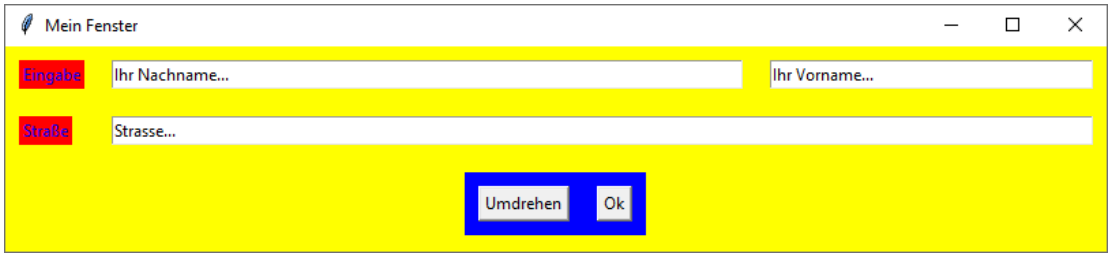

Abbildung 5 Komplexes Beispiels (3 Eingabefelder)

1.4.1 Grid-Layout

Columns:	auto	weight=3	weight=1	
Rows	auto	auto	auto	# braucht man nicht einzutragen

```
root.columnconfigure(1, weight=3)  # setzt die 1. Spalte auf fill
root.columnconfigure(2, weight=1)  # setzt die 2. Spalte auf fill
```

1.4.2 Labels

```
self.label1 = tkinter.Label(root)
self.label1["text"] = "Eingabe"
self.label1.config(foreground = "blue" )
self.label1.config(background = "red")
self.label1.grid(row=0, column=0, sticky="W", padx="10",pady="10")

self.label3 = tkinter.Label(root)
self.label3["text"] = "Straße"
self.label3.config(foreground = "blue" )
self.label3.config(background = "red")
self.label3.grid(row=1, column=0,sticky="W", padx="10",pady="10")
```

Bei Label sollte man den „sticky“-Parameter setzen. Diese platziert das Labelelement innerhalb der zelle. Der Westparameter ist dabei die sinnvollste Variante.

1.4.3 Eingabeelemente

```
self.inputui1    =    tkinter.Entry(root,    background    =    "white",
relief=tkinter.SUNKEN)
self.inputui1.grid(row=0, column=1,sticky="NSEW",padx="10",pady="10")
self.var_name1 = tkinter.StringVar()
self.var_name1.set("Ihr Nachname...")
self.inputui1["textvariable"] = self.var_name1

self.inputui2 = tkinter.Entry(root, background = "white",
       relief=tkinter.SUNKEN)
self.inputui2.grid(row=0, column=2,sticky="NSEW",padx="10",pady="10")
self.var_name2 = tkinter.StringVar()
self.var_name2.set("Ihr Vorname...")
self.inputui2["textvariable"] = self.var_name2

self.inputui3 = tkinter.Entry(root, background = "white",
       relief=tkinter.SUNKEN)
self.inputui3.grid(row=1, column=1, columnspan=2,
       sticky="NSEW",padx="10",pady="10")

self.var_name3 = tkinter.StringVar()
self.var_name3.set("Strasse...")
self.inputui3["textvariable"] = self.var_name3
```

Wichtig:

- Ohne den sticky-Eintrag wird nur die Zelle „gezoomt“!
- Also immer eintragen „**sticky="NSEW**"

1.4.4 Schalter

```
    buttonframe = tkinter.Frame(root)
    buttonframe.config(background = "blue") #"#FF0000"
    #buttonframe.pack(fill="x", side="bottom" )
    buttonframe.grid(row=2, column=0, columnspan=3, padx="10",pady="10")

# der buttonframe hat nun für die Schalter einj eigenes Koordinatensystem
    self.bnAction = tkinter.Button(buttonframe)
    self.bnAction["text"] = "Umdrehen"
    self.bnAction["command"] = self.onReverse
    self.bnAction.grid(row=0, column=0, padx="10",pady="10")

    self.bnOk = tkinter.Button(buttonframe)
    self.bnOk["text"] = "Ok"
    self.bnOk["command"] = root.quit
    self.bnOk.grid(row=0, column=1, padx="10",pady="10")
```

1.5 Komplexes Beispiel (Layout_grid_bsp5.py)

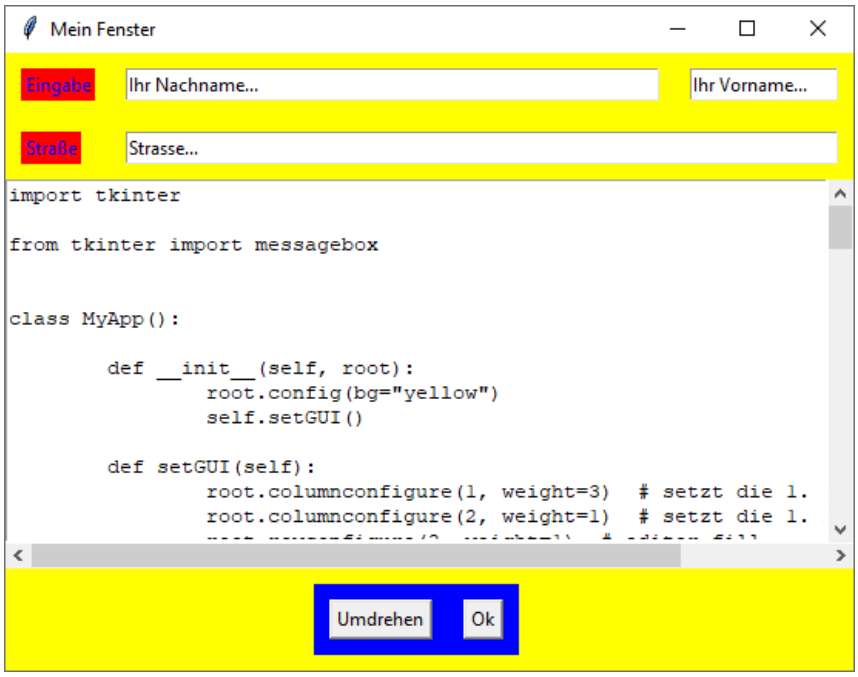

Abbildung 6 Komplexes Beispiels (3 Eingabefelder)

1.5.1 Grid-Layout

Columns:	auto Label	weight=3 entry	weight=1 entry	auto y-scrollbar
Rows	auto	auto	weight=1	auto

```
root.columnconfigure(1, weight=3)  # setzt die 1. Spalte auf fill
root.columnconfigure(2, weight=1)  # setzt die 2. Spalte auf fill
root.rowconfigure(2, weight=1)  # editor fill
```

1.5.2 Labels

```
self.label1 = tkinter.Label(root)
self.label1["text"] = "Eingabe"
self.label1.config(foreground = "blue" )
self.label1.config(background = "red")
self.label1.grid(row=0, column=0, sticky="W", padx="10",pady="10")

self.label3 = tkinter.Label(root)
self.label3["text"] = "Straße"
self.label3.config(foreground = "blue" )
self.label3.config(background = "red")
self.label3.grid(row=1, column=0,sticky="W", padx="10",pady="10")
```

Bei Label sollte man den „sticky“-Parameter setzen. Diese platziert das Labelelement innerhalb der zelle. Der Westparameter ist dabei die sinnvollste Variante.

1.5.3 Einzeilige Eingabeelemente

```
self.inputui1    =    tkinter.Entry(root,    background    =    "white",
relief=tkinter.SUNKEN)
self.inputui1.grid(row=0, column=1, sticky="NSEW", padx="10", pady="10")
self.var_name1 = tkinter.StringVar()
self.var_name1.set("Ihr Nachname...")
self.inputui1["textvariable"] = self.var_name1

self.inputui2 = tkinter.Entry(root, background = "white",
       relief=tkinter.SUNKEN)
self.inputui2.grid(row=0, column=2, columnspan=2,   sticky="NSEW",
           padx="10",pady="10")
self.var_name2 = tkinter.StringVar()
self.var_name2.set("Ihr Vorname...")
self.inputui2["textvariable"] = self.var_name2

self.inputui3 = tkinter.Entry(root, background = "white",
       relief=tkinter.SUNKEN)
self.inputui3.grid(row=1, column=1, columnspan=3,   sticky="NSEW",
           padx="10", pady="10")

self.var_name3 = tkinter.StringVar()
self.var_name3.set("Strasse...")
self.inputui3["textvariable"] = self.var_name3
```

Wichtig:

- Ohne den sticky-Eintrag wird nur die Zelle „gezoomt“!
- Also immer eintragen „**sticky="NSEW**"

1.5.4 Editor

Als erstes wird der Editor und danach die beiden Scrollbar erstellt und eingefügt. Danach werden die drei Elemente verknüpft.

```
self.editor = tkinter.Text(root)
self.editor.config(wrap="none")  # wrap="word"  word char
self.editor.grid(row=2, column=0, columnspan=3, sticky="NSEW",
  padx="0", pady="0")

sbx = tkinter.Scrollbar(root, orient="horizontal")
sbx.grid(row=3, column=0, columnspan=4, sticky="NSEW", pady="0")

sby = tkinter.Scrollbar(root)
sby.grid(row=2, column=3, columnspan=1, sticky="NSEW", pady="0")
```

```
   self.editor["xscrollcommand"] = sbx.set
   sbx["command"] = self.editor.xview
   self.editor["yscrollcommand"] = sby.set
   sby["command"] = self.editor.yview
```

1.5.5 Schalter

```
      buttonframe = tkinter.Frame(root)
      buttonframe.config(background = "blue") #"#FF0000"
      #buttonframe.pack(fill="x", side="bottom" )
      buttonframe.grid(row=2, column=0, columnspan=3, padx="10",pady="10")

# der buttonframe hat nun für die Schalter einj eigenes Koordinatensystem
      self.bnAction = tkinter.Button(buttonframe)
      self.bnAction["text"] = "Umdrehen"
      self.bnAction["command"] = self.onReverse
      self.bnAction.grid(row=0, column=0, padx="10",pady="10")

      self.bnOk = tkinter.Button(buttonframe)
      self.bnOk["text"] = "Ok"
      self.bnOk["command"] = root.quit
      self.bnOk.grid(row=0, column=1, padx="10",pady="10")
```

1.5.6 Vollständiger Quellcode

```
import tkinter

from tkinter import messagebox


class MyApp():

   def __init__(self, root):
      root.config(bg="yellow")
      self.setGUI()

   def setGUI(self):
      root.columnconfigure(1, weight=3)  # setzt die 1. Spalte auf fill
      root.columnconfigure(2, weight=1)  # setzt die 1. Spalte auf fill
      root.rowconfigure(2, weight=1)  # editor fill

      self.label1 = tkinter.Label(root)
      self.label1["text"] = "Eingabe"
      self.label1.config(foreground = "blue" )
      self.label1.config(background = "red")
      self.label1.grid(row=0, column=0, sticky="W", padx="10",pady="10")

      self.inputui1 = tkinter.Entry(root, background = "white",
             relief=tkinter.SUNKEN)
      self.inputui1.grid(row=0, column=1,sticky="NSEW",padx="10",pady="10")
      self.var_name1 = tkinter.StringVar()
      self.var_name1.set("Ihr Nachname...")
```

```
        self.inputui1["textvariable"] = self.var_name1

        self.inputui2 = tkinter.Entry(root, background = "white",
               relief=tkinter.SUNKEN)
        self.inputui2.grid(row=0, column=2, columnspan=2,sticky="NSEW",
               padx="10",pady="10")
        self.var_name2 = tkinter.StringVar()
        self.var_name2.set("Ihr Vorname...")
        self.inputui2["textvariable"] = self.var_name2

#-----------------------------------------

        self.label3 = tkinter.Label(root)
        self.label3["text"] = "Straße"
        self.label3.config(foreground = "blue" )
        self.label3.config(background = "red")
        self.label3.grid(row=1, column=0, sticky="W",  padx="10",pady="10")

        self.inputui3    =    tkinter.Entry(root,    background    =    "white",
relief=tkinter.SUNKEN)
        self.inputui3.grid(row=1, column=1, columnspan=3, sticky="NSEW",
               padx="10",pady="10")
        self.var_name3 = tkinter.StringVar()
        self.var_name3.set("Strasse...")
        self.inputui3["textvariable"] = self.var_name3

# ----------------------------------------


        self.editor = tkinter.Text(root)
        self.editor.config(wrap="none")  # wrap="word"  word char
        self.editor.grid(row=2, column=0,columnspan=3, sticky="NSEW",
               padx="0",pady="0")

        sbx = tkinter.Scrollbar(root, orient="horizontal")
        sbx.grid(row=3, column=0,columnspan=4, sticky="NSEW",pady="0")

        sby = tkinter.Scrollbar(root)
        sby.grid(row=2, column=3,columnspan=1, sticky="NSEW",pady="0")

        self.editor["xscrollcommand"] = sbx.set
        sbx["command"] = self.editor.xview
        self.editor["yscrollcommand"] = sby.set
        sby["command"] = self.editor.yview


# ----------------------------------------

        buttonframe = tkinter.Frame(root)
        buttonframe.config(background = "blue") #"#FF0000"
        buttonframe.grid(row=4, column=0, columnspan=3, padx="10",pady="10")

        self.bnAction = tkinter.Button(buttonframe)
        self.bnAction["text"] = "Umdrehen"
        self.bnAction["command"] = self.onReverse
        self.bnAction.grid(row=0, column=0, padx="10",pady="10")
```

```
        self.bnOk = tkinter.Button(buttonframe)
        self.bnOk["text"] = "Ok"
        self.bnOk["command"] = root.quit
        self.bnOk.grid(row=0, column=1, padx="10",pady="10")

# ----------------------------------------

    def onReverse(self):
        self.var_name1.set( self.var_name1.get()[::-1] )
        #messagebox.showinfo( "Hello Python", "Hello World")


root = tkinter.Tk()
root.title("Mein Fenster")
root.geometry("550x400")
app = MyApp(root)
root.mainloop()
```

1.5.7 Gridlayoutbeispiel mit zwei vertikalen Editoren

Abbildung 7 Layout_grid_bsp6.py

1.5.8 Gridlayoutbeispiel mit zwei horizontalen Editoren

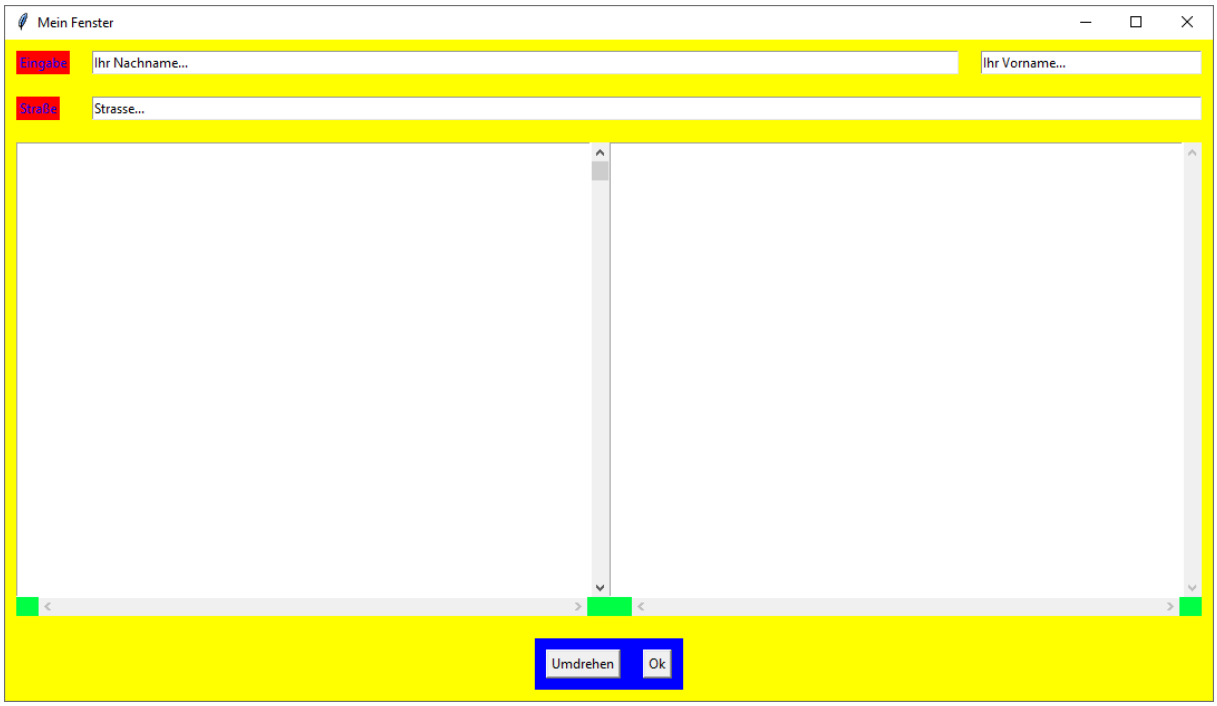

Abbildung 8 Layout_grid_bsp7.py

Besonderheit:

```
editorframe = tkinter.Frame(root)
editorframe.config(background = "#00FF44") #"#FF0000"
editorframe.grid(row=2, column=0, columnspan=3,sticky="NSEW",
       padx="10",pady="10")
editorframe.columnconfigure(0, weight=1)
editorframe.columnconfigure(2, weight=1)
editorframe.rowconfigure(0, weight=1)
```

Der „editorframe“ ist ein Regal mit zwei Fächern. Man muss aber die Gewichtung in der horizontalen und vertikalen Richtung **neu** eintragen!