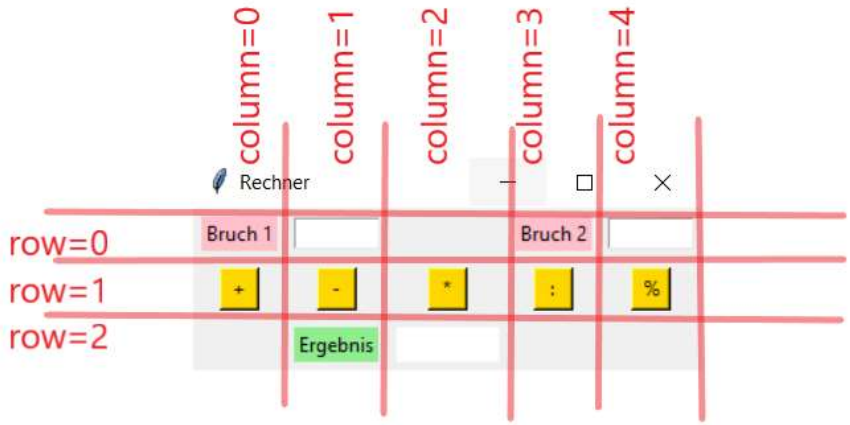
19.1.1. Layout-Manager grid()

- der grid-Manager benutzt ein Raster, in das alle GUI-Komponenten eingepasst werden.



Parameter	Bedeutung
row, column	Zeile bzw. Spalte, in der die GUI-Komponente angeordnet wird.
padx, pady	Leerer Platz rechts und links bzw. ober- und unterhalb der GUI-Komponente

19.4.1.2. Benutzeroberfläche mit Layout-Manager grid()

- Beispiel einer Benutzeroberfläche für die Klasse Bruch (ohne Frame)

```
from tkinter import *

# Einbinden der Klasse Bruch
from clBruch import Bruch

# Instanziieren der Objekte der Klasse Bruch
bruch1 = Bruch()
bruch2 = Bruch()

def btnPlusClick():
    # Übernahme der Daten .get()
    h_bruch = entBruch1.get().split('/')
    bruch1.zaehler = int(h_bruch[0])
    bruch1.nenner = int(h_bruch[1])

    h_bruch = entBruch2.get().split('/')
    bruch2.zaehler = int(h_bruch[0])
    bruch2.nenner = int(h_bruch[1])

    # Verarbeitung der Daten
    ergebnis = bruch1 + bruch2

    # Anzeige der Daten
```

```
        lblErgebnis.config(text=str(ergebnis))

    def btnMinusClick():
        # Übernahme der Daten
        bruch1.zaehler, bruch1.nenner = tuple(int(i) for i in
    entBruch1.get().split("/"))
        bruch2.zaehler, bruch2.nenner = tuple(int(i) for i in
    entBruch2.get().split("/"))

        # Verarbeitung der Daten
        ergebnis = bruch1 - bruch2

        # Anzeige der Daten
        lblErgebnis.config(text=str(ergebnis))

    def btnMultClick():
        # Übernahme der Daten
        bruch1.zaehler, bruch1.nenner = tuple(map(int, entBruch1.get().split("/")))
        bruch2.zaehler, bruch2.nenner = tuple(map(int, entBruch2.get().split("/")))

        ergebnis = bruch1 * bruch2

        # Anzeige der Daten
        lblErgebnis.config(text=str(ergebnis))

    def btnDivClick():
        # Übernahme der Daten
        # Verarbeitung der Daten
        # Anzeige der Daten
        pass

    def btnRestClick():
        # Übernahme der Daten
        # Verarbeitung der Daten
        # Anzeige der Daten
        pass

    # Fenster/Window
    tkFenster = Tk()
    tkFenster.title('Rechner')

    # lbl mit Text Bruch 1
    lblBruch1 = Label(master=tkFenster, bg='pink', text='Bruch 1')
    lblBruch1.grid(row=0, column=0, padx='5', pady='5')
    # Entry für Bruch 1
    entBruch1 = Entry(master=tkFenster, bg='white', width='8')
    entBruch1.grid(row=0, column=1, padx='5', pady='5')

    # lbl mit Text Bruch 2
    lblbruch2 = Label(master=tkFenster, bg='pink', text='Bruch 2')
    lblbruch2.grid(row=0, column=3, padx='5', pady='5')
    # Entry für Bruch 2
    entBruch2 = Entry(master=tkFenster, bg='white', width='8')
    entBruch2.grid(row=0, column=4, padx='5', pady='5')

    # btn zum Addieren
    btnPlus = Button(master=tkFenster, text='+', width='2', bg='gold',
    command=btnPlusClick)
    btnPlus.grid(row=2, column=0, padx='5', pady='5')

    # btn zum Subtrahieren
    btnMinus = Button(master=tkFenster, text='-', width='2', bg='gold',
```

```
command=btnMinusClick)
btnMinus.grid(row=2, column=1, padx='5', pady='5')

# btn zum Multiplizieren
btnMult = Button(master=tkFenster, text='*', width='2', bg='gold',
command=btnMultClick)
btnMult.grid(row=2, column=2, padx='5', pady='5')

# btn zum Dividieren ohne Rest
btnDiv = Button(master=tkFenster, text=':', width='2', bg='gold',
command=btnDivClick)
btnDiv.grid(row=2, column=3, padx='5', pady='5')

# btn zum Rest bei der Division
btnRest = Button(master=tkFenster, text='%', width='2', bg='gold',
command=btnRestClick)
btnRest.grid(row=2, column=4, padx='5', pady='5')

# lbl mit Text Ergebnis
lblTextErgebnis = Label(master=tkFenster, bg='lightgreen', text='Ergebnis')
lblTextErgebnis.grid(row=3, column=1, padx='5', pady='5')

# lbl für das Ergebnis
lblErgebnis = Label(master=tkFenster, bg='white', width='8', text='')
lblErgebnis.grid(row=3, column=2, padx='5', pady='5')

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

Aufgaben

1. GUI RemoteControl
   für die Klasse RemoteControl ist eine grafische Benutzeroberfläche zu erweitern:
   Die direkte Auswahl des Programmspeicherplatzes soll mithilfe von Button ergänzt werden.
   Die Klasse RemoteControl ist dazu um eine geeignete Methode zu erweitern.