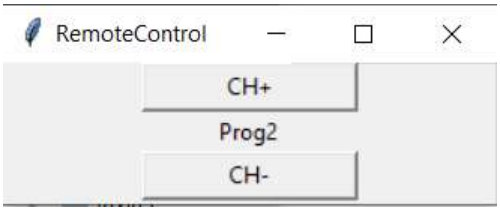
12.4. Benutzeroberfläche mit dem Modul tkinter

- mit dem Modul **tkinter** aus der Standardbibliothek werden grafische Benutzeroberflächen erstellt
- das Modul unterstützt
  - die Erstellung eines Dialogfensters
  - die Erzeugung und Anordnung der Steuerelemente (Widgets) innerhalb des Dialogfensters
  - mit dem Start einer Endlosschleife wird der Dialog gestartet und die Interaktionen des Benutzers können empfangen und verarbeitet werden

- Steuerelemente (Auswahl)
  - Schaltflächen Button
  - Beschriftung Label

- Ereignistyp (Auswahl)
  - Button clicked Aufruf der Handler-Funktion

Beispiel:



12.4.1. Grundsätzlicher Programmaufbau

- Einbinden Modul **tkinter**
- Dialogfenster erstellen **.Tk()**
- Dialogfenster aktivieren **.mainloop()**
- Hinweis: zunächst noch keine Behandlung von Ereignissen

```
# Einbinden tkinter (Modul zur Erstellung von Benutzeroberflächen / GUI)
import tkinter as tk

# Dialogfenster erstellen
root = tk.Tk()

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

12.4.2. Titel Dialogfenster anpassen

- der Titel in der Kopfzeile des Dialogfensters lässt sich über den Aufruf der Methode **.title()** ändern

```
# Einbinden tkinter (Modul zur Erstellung von Benutzeroberflächen / GUI)
import tkinter as tk

# Dialogfenster erstellen
root = tk.Tk()

# Titel Dialogfenster anpassen
root.title("RemoteControl")

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

12.4.3. Steuerelemente ergänzen

- Schaltfläche definieren mit dem Aufruf der Methode **.Button()**
  - Parameter
    - Zuordnung zum Dialogfenster
    - Beschriftung der Schaltfläche
    - z.B. btn_ch_plus = tk.Button(root, text="CH+")

- Anzeigefeld definieren mit dem Aufruf der Methode **.Label()**
  - Parameter
    - Zuordnung zum Dialogfenster
    - Text für das Anzeigefeld
    - z.B. lbl_program = tk.Label(root, text="Programm")

- alle Steuerelemente über den Layout-Manager mit dem Aufruf der Methode **.pack()** (automatisch) zusammengefügt und dem Dialogfenster zuordnet

```
# Einbinden tkinter (Modul zur Erstellung von Benutzeroberflächen / GUI)
import tkinter as tk

# Dialogfenster erstellen
root = tk.Tk()

# Titel Dialogfenster anpassen
root.title("RemoteControl")

# Schaltfläche für Programm+ erstellen
btn_ch_plus = tk.Button(root, text="CH+")
# Anzeigefeld für Programmname erstellen
lbl_program = tk.Label(root, text="Programm")
# Schaltfläche für Programm- erstellen
btn_ch_minus = tk.Button(root, text="CH-")

# alle Elemente über den Layout-Manager pack zusammenfügen
ch_plus.pack()
lbl_program.pack()
ch_minus.pack()

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

12.4.4. Ereignisbehandlung ergänzen

- das Ereignis „Button betätigt“ wird in einer Callback-Funktion behandelt
- die Callback-Funktion ist zu definieren und
- wird über den zusätzlichen Parameter **.command()** in die Methode **Button()** eingebunden
- Beispiel:
  - der Text im Anzeigefeld soll sich ändern, wenn eine der Schaltflächen (Button) betätigt wird
  - wird die Schaltfläche CH+ betätigt, soll „nächstes Programm“ angezeigt werden
    - mit dem Methoden-Aufruf **.config(text="nächstes Programm")** für das Anzeigefeld wird der Text entsprechend gesetzt
  - wird die Schaltfläche CH- betätigt, soll „vorheriges Programm“ angezeigt werden
    - mit dem Methoden-Aufruf **.config(text="nächstes Programm")** für das Anzeigefeld wird der Text entsprechend gesetzt

```
:
:
# Ereignisbehandlung für den Button ch_plus
def nextProgram():
    lbl_program.config(text="nächstes Programm")

# Ereignisbehandlung für den Button ch_minus
def lastProgram():
    lbl_program.config(text="vorheriges Programm")

# Fenster erstellen
root = tk.Tk()

# Fenstertitel erstellen
root.title("RemoteControl")

# Schaltfläche für Programm+ erstellen
btn_ch_plus = tk.Button(root, text="CH+", command=nextProgram)
# Anzeigefeld für Programmname erstellen
lbl_program = tk.Label(root, text="Programm")
# Schaltfläche für Programm- erstellen
btn_ch_minus = tk.Button(root, text="CH-", command=lastProgram)
:
:
```

12.4.5. Anwendung einbinden

- Klasse einbinden
- Objekt instanziieren
- Callback-Funktionen anpassen

```
# Einbinden tkinter (Modul zur Erstellung von Benutzeroberflächen / GUI)
import tkinter as tk

# Einbinden der Klasse RemoteControl (aus dem Python-Modul clRemoteControl.py)
from clRemoteControl import *

# Instanziieren eines Objektes auf der Grundlage der Klasse RemoteControl
# es werden 10 Programmspeicherplätze angelegt
rc = RemoteControl(10)

# Ereignisbehandlung für den Button ch_plus
def nextProgram():
    rc.nextProgram()
    lbl_program.config(text=rc.programs[rc.curr_program_number])

# Ereignisbehandlung für den Button ch_minus
def prevProgram():
    rc.prevProgram()
    lbl_program.config(text=rc.programs[rc.curr_program_number])

# Fenster erstellen
root = tk.Tk()
# Fenstertitel erstellen
root.title("RemoteControl")

# Schaltfläche für Programm+ erstellen
btn_ch_plus = tk.Button(root, text="CH+", width=15, command=nextProgram)
# Anzeigefeld für Programmname erstellen
lbl_program = tk.Label(root, text=rc.programs[rc.curr_program_number],
width=20)
# Schaltfläche für Programm- erstellen
btn_ch_minus = tk.Button(root, text="CH-", width=15, command=prevProgram)
# alle Elemente über den Layout-Manager pack zusammenfügen
btn_ch_plus.pack()
lbl_program.pack()
btn_ch_minus.pack()

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

Aufgabe

1. Lautstärkeregelung ergänzen
   Die Klasse RemoteControl ist um ein Attribut für die Lautstärke (volume) zu ergänzen;
   die Lautstärke soll nur über entsprechende Methodenaufrufe verändert werden können, das Attribut soll nur innerhalb der Klasse modifiziert werden können
   Die grafische Oberfläche ist entsprechend zu erweitern (Schaltfläche für lauter, Schaltfläche für leiser und die Anzeige der aktuellen Lautstärke)