CloudCommand

Cyber Security

CloudCommand GmbH chr.schumacher@gmx.tm

CloudCommand

Grundlagen der Programmierung

CloudCommand GmbH chr.schumacher@gmx.tm

GUI Programmierung

Einführung in die GUI-Programmierung

Begriffserklärung:

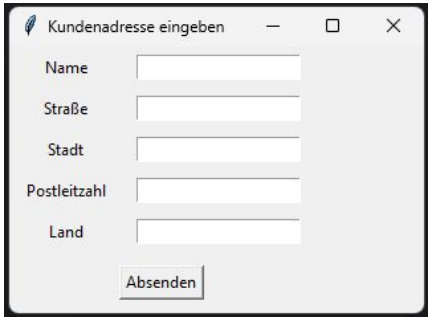
- Die Abkürzung GUI steht für: **Graphical User Interface**
- Zu deutsch: **Grafische Benutzeroberfläche**

Einführung in die GUI-Programmierung

- tkinter ist ein Modul, welches zur Benutzung importiert werden muss.
- **Tkinter** gehört zu den sogenannten **Standardlibraries** in Python
- Der Name setzt sich wie folgt zusammen:
  - **tk** ist die Abkürzung für Toolkit
  - **inter** ist die Abkürzung für Interface

Einführung in die GUI-Programmierung

- Das Aussehen einer Applikation, welche mit tkinter geschrieben wurde ist stark vom Betriebssystem abhängig.
- Die einzelnen grafischen Elemente heißen Steuerelemente bzw. Widgets.



Einführung in die GUI-Programmierung

- Widgets müssen nicht selbst programmiert werden
- Lediglich die Positionierung muss erfolgen.
- Dazu gibt es Layoutmanager:
  - **pack:** schmales, automatisiertes Layout
  - **grid:** Ausrichtung an gedachten Spalten und Zeilen
  - **place:** Explizites positionieren von Widgets

Die erste Applikation

- Ein tkinter-Programm beginnt üblicherweise mit:

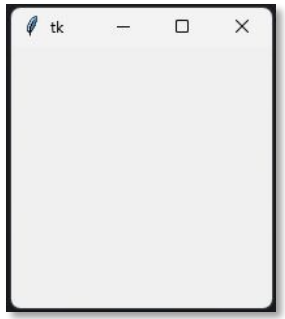
  import tkinter as tk
- Das Modul tkinter wird importiert und zusätzlich der alias **tk** darauf gesetzt. Alle Bestandteile der Bibliothek werden zukünftig über ein vorangestelltes **tk.** aufgerufen.

Die erste Applikation

- Ein tkinter-Programm beginnt üblicherweise mit:

  import tkinter as tk
  MeinFenster = tk.Tk()
  MeinFenster.mainloop()

- Diese obligatorischen Zeilen stellen ein leeres Fenster dar.



Die erste Applikation

- **MeinFenster = tk.Tk():** Erzeugt eine neue Instanz
- **MeinFenster.mainloop():** Startet die Hauptschleife und zeigt das Fenster an.

Die erste Applikation

Der nächste Schritt besteht darin das Hauptfenster mit Funktionalitäten zu füllen.

Schritt 1:

- Hinzufügen eines Buttons zum Beenden

Die erste Applikation

Schritt 1:

```python
import tkinter as tk


def beenden():
    root.destroy()


# Hauptfenster erstellen
root = tk.Tk()
root.title("Beenden Button Beispiel")
root.geometry("300x200")  # Fenstergröße festlegen

# Button erstellen
beenden_button = tk.Button(root, text="Beenden", command=beenden)
beenden_button.place(x=120, y=80, width=60, height=30)

# Hauptschleife starten
root.mainloop()
```

Die erste Applikation

Schritt 2:

- Hinzufügen eines weiteren Buttons, welcher bei Betätigung ein Label mit dem Wert eines Eingabefeldes anzeigt.

Die erste Applikation

Schritt 2:

```python
import tkinter as tk


# Erstelle die Hauptanwendungsklasse
class App:
    def __init__(self, root):
        self.root = root
        self.root.title("Leeres Fenster")
        self.root.geometry("400x300")  # Größe des Fensters (optional)

        # Erstelle einen Frame, um Label und Eingabefeld nebeneinander zu platzieren
        self.frame = tk.Frame(root)
        self.frame.pack(pady=20)

        # Füge ein Label hinzu, das am linken Rand ausgerichtet ist
        self.label1 = tk.Label(self.frame, text="Geben Sie Zahl 1 ein:")
        self.label1.pack(side='left')

        # Füge ein Eingabefeld rechts vom Label hinzu
        self.entry1 = tk.Entry(self.frame)
        self.entry1.pack(side='left', padx=10)

        # Füge einen "Label anzeigen"-Button in der Mitte am unteren Rand des Fenster
        self.show_label_button = tk.Button(root, text="Label anzeigen", command=self.
        self.show_label_button.pack(side='bottom', pady=10)

```

Die erste Applikation

Schritt 2:

```
        # Füge einen "Beenden"-Button in der Mitte am unteren Rand des Fensters hinzu
        self.quit_button = tk.Button(root, text="Beenden", command=root.quit)
        self.quit_button.pack(side='bottom', pady=10)

        # Initialisiere das zweite Label (noch nicht angezeigt)
        self.label2 = None

    def show_label(self):
        # Aktion für den "Label anzeigen"-Button
        eingabe = self.entry1.get()
        if self.label2:
            self.label2.destroy()  # Entfernt das alte Label, wenn es schon existiert
        self.label2 = tk.Label(self.root, text=eingabe)
        self.label2.pack()
        print("Label anzeigen Button gedrückt")
        print("Eingabe:", eingabe)


# Erstelle die Hauptfensterinstanz
root = tk.Tk()
app = App(root)

# Starte die Hauptschleife der Anwendung
root.mainloop()
```

Widgets bzw. weitere Steuerelemente

Folgend werden weitere Widgets vorgestellt:

Widget	Funktion
Button	Ein normaler Button / Schaltfläche
Canvas	Fläche, um Grafiken anzuzeigen
Checkbutton	Auswahlfeld (nicht Optionsfeld!)
Entry	Ein einzeiliges Eingabefeld
Label	Beschriftungsfeld
LabelFrame	Ein beschrifteter Rahmen

Widgets bzw. weitere Steuerelemente

Folgend werden weitere Widgets vorgestellt:

Widget	Funktion
Listbox	Eine Liste von Items mit Auswahl
Menubutton	Schaltfläche für Kontextmenü
OptionMenu	Auswahlliste mit einer Reihe Optionen
Radiobutton	Auswahlfeld, im Gegensatz zu Checkbutton kann nur eine innerhalb einer Gruppe gewählt sein
Scrollbar	Scrollleiste

Widgets bzw. weitere Steuerelemente

Folgend werden weitere Widgets vorgestellt:

Widget	Funktion
Spinbox	Zahlenwertauswahlfeld
Text	mehrzeiliges Texteingabefeld

Widgets bzw. weitere Steuerelemente

Label Widget:

- Das Label Widget zeigt i.d.R Text auf dem Fenster an Mittels Parameter ist es möglich ebenfalls Grafiken als Teil eines Labels anzuzeigen.

Widgets bzw. weitere Steuerelemente

Label Widget:

```python
import tkinter as tk

MeinFenster = tk.Tk()

# Textausgabe erzeugen
label1 = tk.Label(MeinFenster, text="Wlan-Status: ")

# in GUI Elemente einbetten
label1.pack(side="left")

# Grafik einbetten
bild1 = tk.PhotoImage(file="wlan_klein.png", height=32, width=32,)
label2 = tk.Label(MeinFenster, image=bild1).pack(side="right")

MeinFenster.mainloop()
```

Widgets bzw. weitere Steuerelemente

Label Widget, Farben und Schrift konfigurieren:

- Die Farben für ein Label werden mit den Parametern **fg** und **bg** konfiguriert
  fg='#e0e0e0', bg='orange'
- Einstellung der Schriftart durch den Parameter
  font=('arial', 25, 'bold')

Widgets bzw. weitere Steuerelemente

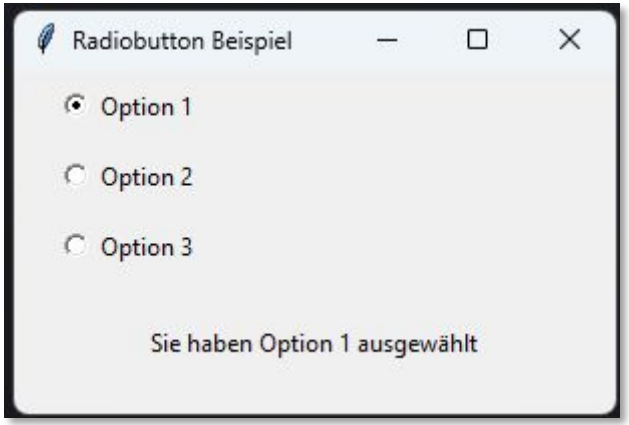
Label Widget, Farben und Schrift konfigurieren:

```python
import tkinter as tk

MeinFenster = tk.Tk()
MeinFenster.title("Buntes Label")
MeinFenster.geometry("320x40")

# Label erzeugen:
label1 = tk.Label(MeinFenster,
                  text='Buntes Label',
                  fg='#e0e0e0',
                  bg='orange',
                  font=('arial', 25, 'bold'))

# in GUI Elemente einbetten
label1.pack(side="top")

MeinFenster.mainloop()
```

Widgets bzw. weitere Steuerelemente

Radiobutton-Widget:

- Dieses Widget erlaubt es immer nur eine Option auszuwählen.



Widgets bzw. weitere Steuerelemente

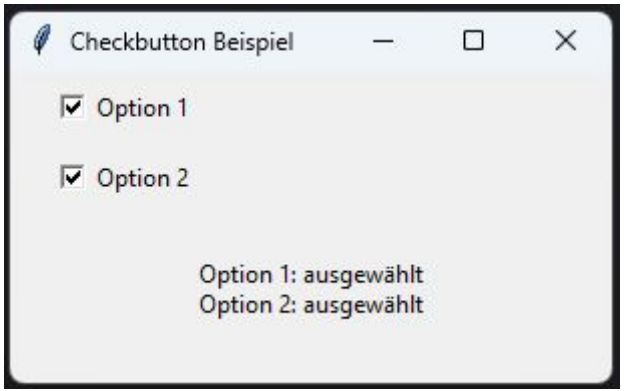
Radiobutton-Widget:

```python
import tkinter as tk

# Erstelle die Hauptanwendungsklasse
class App:
    def __init__(self, root):
        self.MeinFenster = MeinFenster
        self.MeinFenster.title("Radiobutton Beispiel")
        self.MeinFenster.geometry("300x180")  # Größe des Fensters (optional)

        # Erstelle eine StringVar für die ausgewählte Option
        self.selected_option = tk.StringVar()
        self.selected_option.set("Option 1")  # Standardmäßig ausgewählte Option

        # Erstelle die Radiobuttons
        self.radio1 = tk.Radiobutton(root, text="Option 1", variable=self.selected_op
        self.radio1.pack(anchor='w', padx=20, pady=5)

        self.radio2 = tk.Radiobutton(root, text="Option 2", variable=self.selected_op
        self.radio2.pack(anchor='w', padx=20, pady=5)

        self.radio3 = tk.Radiobutton(root, text="Option 3", variable=self.selected_op
        self.radio3.pack(anchor='w', padx=20, pady=5)

        # Label zur Anzeige der ausgewählten Option
        self.selection_label = tk.Label(root, text="Sie haben Option 1 ausgewählt")
        self.selection_label.pack(pady=20)

    def show_selection(self):
        # Methode zur Anzeige der ausgewählten Option
        selection = f"Sie haben {self.selected_option.get()} ausgewählt"
        self.selection_label.config(text=selection)

# Erstelle die Hauptfensterinstanz
MeinFenster = tk.Tk()
app = App(MeinFenster)

# Starte die Hauptschleife der Anwendung
MeinFenster.mainloop()
```

Widgets bzw. weitere Steuerelemente

Checkbutton-Widget:

- Dieses Widget erlaubt es mehrere Optionen auszuwählen.



Widgets bzw. weitere Steuerelemente

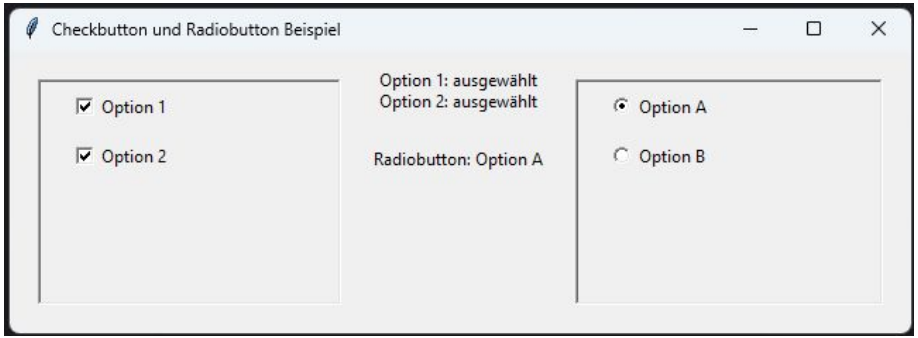
Checkbutton-Widget:

```python
import tkinter as tk

# Erstelle das Hauptfenster
root = tk.Tk()
root.title("Checkbutton Beispiel")
root.geometry("300x180")  # Größe des Fensters (optional)

# Erstelle BooleanVars für die Checkbutton-Status
var1 = tk.BooleanVar()
var2 = tk.BooleanVar()

# Funktion zum Aktualisieren des Statuslabels
def update_status():
    status = f"Option 1: {'ausgewählt' if var1.get() else 'nicht ausgewählt'}\n"
    status += f"Option 2: {'ausgewählt' if var2.get() else 'nicht ausgewählt'}"
    status_label.config(text=status)

# Erstelle die Checkbuttons
check1 = tk.Checkbutton(root, text="Option 1", variable=var1, command=update_status)
check1.pack(anchor='w', padx=20, pady=5)

check2 = tk.Checkbutton(root, text="Option 2", variable=var2, command=update_status)
check2.pack(anchor='w', padx=20, pady=5)

# Label zur Anzeige des Status
status_label = tk.Label(root, text="Option 1: nicht ausgewählt\nOption 2: nicht ausgew
status_label.pack(pady=20)

# Starte die Hauptschleife der Anwendung
root.mainloop()
```

Widgets bzw. weitere Steuerelemente

Frame-Widget:

- Dieses Widget erlaubt es optisch und funktional Steuerelemente voneinander zu trennen.



Widgets bzw. weitere Steuerelemente

Frame-Widget:

```python
import tkinter as tk

# Erstelle das Hauptfenster
MeinFenster = tk.Tk()
MeinFenster.title("Checkbutton und Radiobutton Beispiel")
MeinFenster.geometry("600x200")  # Größe des Fensters (optional)

# Erstelle BooleanVars für die Checkbutton-Status
var1 = tk.BooleanVar()
var2 = tk.BooleanVar()

# Funktion zum Aktualisieren des Statuslabels für Checkbuttons
def update_status():
    status = f"Option 1: {'ausgewählt' if var1.get() else 'nicht ausgewählt'}\n"
    status += f"Option 2: {'ausgewählt' if var2.get() else 'nicht ausgewählt'}"
    status_label.config(text=status)

# Funktion zum Aktualisieren des Statuslabels für Radiobuttons
def update_radio_status():
    radio_status = f"Radiobutton: {radio_var.get()}"
    radio_status_label.config(text=radio_status)

# Erstelle einen Frame um die Checkbuttons
frame1 = tk.Frame(MeinFenster, relief=tk.FLAT, borderwidth=1)
frame1.pack(side='left', pady=20, padx=20, fill="both", expand=True)

# Erstelle die Checkbuttons innerhalb des Frames
check1 = tk.Checkbutton(frame1, text="Option 1", variable=var1, command=update_status)
check1.pack(anchor='w', padx=20, pady=5)
```

Widgets bzw. weitere Steuerelemente

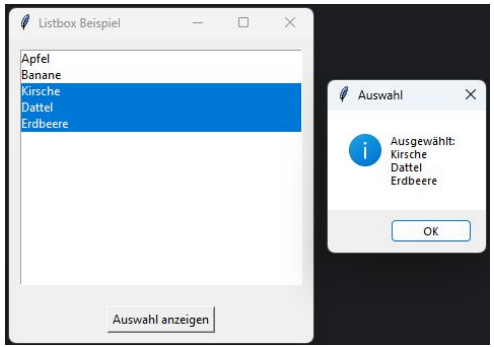
Frame-Widget:

```
check2 = tk.Checkbutton(frame1, text="Option 2", variable=var2, command=update_status)
check2.pack(anchor='w', padx=20, pady=5)

# Erstelle einen zweiten Frame um die Radiobuttons
frame2 = tk.Frame(MeinFenster, relief=tk.FLAT, borderwidth=1)
frame2.pack(side='right', pady=20, padx=20, fill="both", expand=True)

# Erstelle eine StringVar für die Radiobutton-Option
radio_var = tk.StringVar()
radio_var.set("Option A")  # Standardmäßig ausgewählte Option

# Erstelle die Radiobuttons innerhalb des zweiten Frames
radio1 = tk.Radiobutton(frame2, text="Option A", variable=radio_var, value="Option A", command=update_radio_status)
radio1.pack(anchor='w', padx=20, pady=5)

radio2 = tk.Radiobutton(frame2, text="Option B", variable=radio_var, value="Option B", command=update_radio_status)
radio2.pack(anchor='w', padx=20, pady=5)

# Label zur Anzeige des Status der Checkbuttons
status_label = tk.Label(MeinFenster, text="Option 1: nicht ausgewählt\nOption 2: nicht ausgewählt")
status_label.pack(pady=10)

# Label zur Anzeige des Status der Radiobuttons
radio_status_label = tk.Label(MeinFenster, text="Radiobutton: Option A")
radio_status_label.pack(pady=10)

# Starte die Hauptschleife der Anwendung
MeinFenster.mainloop()
```

Widgets bzw. weitere Steuerelemente

Listbox-Widget:

- Dieses Widget erlaubt es eine Liste anzuzeigen und ggf. mehrere Elemente daraus auszuwählen.
- In dem Beispiel wird die **Messagebox** benutzt, um die gewählten Elemente anzuzeigen.



Widgets bzw. weitere Steuerelemente

Listbox-Widget:

```python
import tkinter as tk
from tkinter import messagebox


# Funktion zum Anzeigen der aktuell ausgewählten Elemente in einer Messagebox
def show_selection():
    # Hole die aktuell ausgewählten Indizes
    selected_indices = listbox.curselection()

    # Hole die aktuell ausgewählten Elemente
    selected_items = [listbox.get(i) for i in selected_indices]

    if selected_items:
        # Zeige die ausgewählten Elemente in einer Messagebox an
        items_text = "\n".join(selected_items)
        messagebox.showinfo( title: "Auswahl",  message: f"Ausgewählt:\n{items_text}")
    else:
        # Zeige eine Warnung an, wenn keine Auswahl getroffen wurde
        messagebox.showwarning( title: "Keine Auswahl",  message: "Bitte wählen Sie mindestens ein Element aus.")

```

Widgets bzw. weitere Steuerelemente

Listbox-Widget:

```
21
22  # Erstelle das Hauptfenster
23  root = tk.Tk()
24  root.title("Listbox Beispiel")
25  root.geometry("300x300")  # Größe des Fensters
26
27  # Erstelle eine Listbox mit Mehrfachauswahl und füge einige Elemente hinzu
28  listbox = tk.Listbox(root, height=8, selectmode=tk.MULTIPLE)
29  listbox.pack(pady=10, padx=10, fill=tk.BOTH, expand=True)
30
31  # Füge einige Elemente zur Listbox hinzu
32  items = ["Apfel", "Banane", "Kirsche", "Dattel", "Erdbeere"]
33  for item in items:
34      listbox.insert(tk.END, *elements: item)
35
36  # Erstelle einen Button zum Anzeigen der Auswahl
37  show_button = tk.Button(root, text="Auswahl anzeigen", command=show_selection)
38  show_button.pack(pady=10)
39
40  # Starte die Hauptschleife der Anwendung
41  root.mainloop()
```

CloudCommand

CloudCommand GmbH chr.schumacher@gmx.tm