

Tkinter Canvas - Offline-Referenz für dein Konvertierungsprogramm

1. Wozu Canvas in deinem Projekt?

- Vorschau von Bildern vor/nach der Konvertierung.
- Anzeige eines Fortschritts (visuell).
- Eigene GUI-Elemente wie Fortschrittsbalken oder Buttons.

2. Canvas erstellen

```
from tkinter import Tk, Canvas

root = Tk()

canvas = Canvas(root, width=400, height=300, bg='white')

canvas.pack()
```

3. Elemente zeichnen

- `canvas.create_rectangle(x1, y1, x2, y2, fill='blue')`
- `canvas.create_text(x, y, text='Konvertiere...')`
- `canvas.create_line(x1, y1, x2, y2)`
- `canvas.create_oval(x1, y1, x2, y2)`

4. Bilder anzeigen (Pillow)

```
from PIL import Image, ImageTk

img = Image.open('bild.jpg')

img = img.resize((300, 200))

tk_img = ImageTk.PhotoImage(img)

canvas.create_image(10, 10, anchor='nw', image=tk_img)

# Referenz behalten: self.tk_img = tk_img
```

5. Bewegung und Tags

- `canvas.create_rectangle(..., tags='balken')`
- `canvas.move('balken', dx, dy)`
- `canvas.itemconfig(id, text='Fertig!')`

Tkinter Canvas - Offline-Referenz für dein Konvertierungsprogramm

6. Fortschrittsbalken bauen

```
balken = canvas.create_rectangle(10, 10, 10, 30, fill='green')
```

```
def update_bar(pct):
```

```
    canvas.coords(balken, 10, 10, 10 + pct * 3, 30)
```

```
    canvas.update()
```

7. Maus-Events

- `canvas.bind('<Button-1>', callback)`
- `canvas.tag_bind('balken', '<Enter>', callback)`

8. Weitere Befehle

- `canvas.delete(item_id)` oder `canvas.delete('all')`
- `canvas.itemconfig(id, fill='red')`
- `canvas.find_withtag('name')`
- `canvas.coords(id)`

9. Tipps für dein Programm

- Nutze Canvas für Fortschritt, Vorschau und Statusanzeige
- Setze Vorschau klein mit `Image.thumbnail()`
- GUI-Aufteilung: links Canvas, rechts Optionen