

## Contents

# CustomTkinter Cheat Sheet

Semester Project & Exam Revision

## 1. Basics

- **Library:** Modern UI layer built on Tkinter
- **Python:** 3.7+
- **Import:** `import customtkinter as ctk`

```
import customtkinter as ctk

ctk.set_appearance_mode("dark")      # "dark" | "light" | "system"
ctk.set_default_color_theme("blue")  # "blue" | "dark-blue" | "green"

app = ctk.CTk()                    # Main window
app.geometry("500x300")
app.title("My App")
app.mainloop()
```

## 2. Core Widgets

Widget	Purpose	Key Options
CTkLabel	Display text	text, font, text_color
CTkButton	Clickable button	text, command, fg_color
CTkEntry	Text input	placeholder_text, show
CTkFrame	Container	fg_color, corner_radius
CTkSwitch	ON / OFF toggle	variable, command
CTkCheckBox	Checkbox	onvalue, offvalue
CTkSlider	Slider (range)	from_, to, command
CTkProgressBar	Progress indicator	set(0-1)

## 3. Widget Examples

### Buttons & Entry

```
btn = ctk.CTkButton(app, text="Click Me", command=lambda: print("Clicked"))
btn.pack(pady=10)

entry = ctk.CTkEntry(app, placeholder_text="Email")
entry.pack(pady=10)
```

### Switch & Checkbox

```
mode_var = ctk.StringVar(value="off")
switch = ctk.CTkSwitch(app, text="Dark Mode",
                      variable=mode_var, onvalue="on", offvalue="off")
switch.pack()
```

### Slider + ProgressBar

```
def update(val):
    progress.set(float(val)/100)

slider = ctk.CTkSlider(app, from_=0, to=100, command=update)
slider.pack()

progress = ctk.CTkProgressBar(app)
progress.pack()
```

## 4. Layout Managers

Manager	Use Case
pack	Simple vertical/horizontal layouts
grid	Form-like layouts (rows/columns)
place	Absolute positioning (rare)

```
widget.pack(pady=10)
widget.grid(row=0, column=1, padx=5)
```

## 5. Frames & Organization

```
sidebar = ctk.CTkFrame(app, width=200)
sidebar.pack(side="left", fill="y")

main = ctk.CTkFrame(app)
main.pack(side="right", expand=True, fill="both")

scroll = ctk.CTkScrolledFrame(main, width=300, height=200)
scroll.pack()
```

## 6. Themes & Appearance

- Appearance mode: dark | light | system
- Color themes: blue, dark-blue, green
- Colors customizable per widget

```
ctk.set_appearance_mode("light")
btn = ctk.CTkButton(app, fg_color="#9b59b6", hover_color="#8e44ad")
```

## 7. Windows & Dialogs

```
top = ctk.CTkToplevel(app)
top.title("Settings")
top.grab_set() # Modal window
```

## 10. Complete Layout Example (EXPLAINED)

```
# Main window
app = ctk.CTk()
app.title("Layout Avancé")
app.geometry("700x500")

# Sidebar
sidebar = ctk.CTkFrame(app, width=200, corner_radius=0)
sidebar.pack(side="left", fill="y")
ctk.CTkLabel(sidebar, text="Menu", font=("Arial", 20, "bold")).pack(pady=20)
ctk.CTkButton(sidebar, text="Accueil", width=180).pack(pady=10)
ctk.CTkButton(sidebar, text="Paramètres", width=180).pack(pady=10)
ctk.CTkButton(sidebar, text="Propos", width=180).pack(pady=10)

# Main Frame
main_frame = ctk.CTkFrame(app)
main_frame.pack(side="right", fill="both", expand=True, padx=20, pady=20)
ctk.CTkLabel(main_frame, text="Contenu Principal", font=("Arial", 24, "bold")).pack(pady=20)

# Scrollable content
scrollable = ctk.CTkScrolledFrame(main_frame, width=400, height=300)
scrollable.pack(pady=10)
for i in range(20):
    ctk.CTkLabel(scrollable, text=f"Element {i+1}").pack(pady=5)

app.mainloop()
```

## Why This Layout Is Important

- Frames separate UI sections (clean structure)
- Sidebar for navigation (professional apps)
- ScrollableFrame replaces Listbox
- pack() is ideal for vertical sections
- expand=True allows responsive resizing

## 11. CRUD Project Essentials (Procedural)

- Build a CRUD app (Create, Read, Update, Delete)
- MySQL database backend
- Forms + Buttons + Scrollable list

## 12. Database Connection (MySQL)

```
import mysql.connector

def connect_db():
    return mysql.connector.connect(
        host="localhost",
        user="root",
        password="",
        database="crud_db"
    )
```

## 13. Form Layout (Label + Entry + Grid)

```
form = ctk.CTkFrame(app)
form.pack(pady=20)

ctk.CTkLabel(form, text="Name").grid(row=0, column=0, padx=10, pady=10)
ctk.CTkLabel(form, text="Email").grid(row=1, column=0, padx=10, pady=10)

name_entry = ctk.CTkEntry(form, width=250)
email_entry = ctk.CTkEntry(form, width=250)
name_entry.grid(row=0, column=1)
email_entry.grid(row=1, column=1)
```

## 14. Validation + Error Messages

```
error_label = ctk.CTkLabel(app, text="", text_color="red")
error_label.pack()

def validate_inputs():
    if name_entry.get() == "" or email_entry.get() == "":
        error_label.configure(text="All fields are required")
        return False
    if "@" not in email_entry.get():
        error_label.configure(text="Invalid email address")
        return False
    error_label.configure(text="")
    return True
```

## 15. CREATE / INSERT

```
def add_record():
    if not validate_inputs():
        return

    db = connect_db()
    cursor = db.cursor()
```

```

query = "INSERT INTO users (name,email) VALUES (%s,%s)"
cursor.execute(query, (name_entry.get(), email_entry.get()))
db.commit()
db.close()
fetch_data()

```

## 16. READ / FETCH

```

list_frame = ctk.CTkScrollableFrame(app, width=400, height=300)
list_frame.pack(pady=20)

def fetch_data():
    for widget in list_frame.winfo_children():
        widget.destroy()
    db = connect_db()
    cursor = db.cursor()
    cursor.execute("SELECT * FROM users")
    rows = cursor.fetchall()
    db.close()
    for row in rows:
        btn = ctk.CTkButton(list_frame, text=f"{row[1]} - {row[2]}",
                            command=lambda r=row: select_record(r))
        btn.pack(fill="x", pady=5)

```

## 17. UPDATE & DELETE (With Selected Item)

```

selected_id = None

def select_record(row):
    global selected_id
    selected_id = row[0]
    name_entry.delete(0, "end")
    email_entry.delete(0, "end")
    name_entry.insert(0, row[1])
    email_entry.insert(0, row[2])

def update_record():
    if selected_id is None:
        error_label.configure(text="Select a record first")
        return
    db = connect_db()
    cursor = db.cursor()
    cursor.execute("UPDATE users SET name=%s,email=%s WHERE id=%s",
                  (name_entry.get(), email_entry.get(), selected_id))
    db.commit()
    db.close()
    fetch_data()

def delete_record():
    if selected_id is None:
        error_label.configure(text="Select a record first")
        return
    db = connect_db()
    cursor = db.cursor()
    cursor.execute("DELETE FROM users WHERE id=%s", (selected_id,))
    db.commit()
    db.close()
    fetch_data()

```

## 18. Exam Question → Answer

**Q: Why CustomTkinter?**

A: Modern UI, dark mode, better UX, same logic as Tkinter.

**Q: Why Frames?**

A: Organize UI into sections (sidebar, content, forms).

**Q: Why grid() for forms?**

A: Ensures proper alignment of labels and entry widgets.

**Q: CRUD flow?**

A: Create → Insert into MySQL

Read → Fetch and display

Update → Modify selected row

Delete → Remove selected row

**Q: How to connect UI and DB?**

A: Using mysql-connector and button-triggered functions.