



# Tag 9

## GUI with tkinter

21. April 2023

# Ablauf

- Rückblick achter Tag
- Prüfung
- Einstieg in GUIs
- Aufbau und Widgets
  - Übung
- Projektarbeit Themen- und Gruppenzuordnung
- Layouts
  - Übung
- Funktionalität / Handler
  - Übung

# Rückblick achter Tag

# Prüfung

- 60 min
- Open Book / mit Internet
- Kein Chat, weder Human noch Artifizuell

**Viel Erfolg!**

# tkinter

- Steht für TK-Interface
  - Anbindung an TCL/TK
- Auf den meisten Plattformen verfügbar
  - Mac, Windows, Linux
- Wird mit Python mitgeliefert
- Für kleine und mittlere Anwendungen geeignet
- Alternativen:
  - wxPython, PyQt, PySide, Pygame, Pyglet, and PyGTK



# GUI-Anwendungen im Allgemeinen

- UI / GUI heutzutage fast synonym
- Event-basiert
  - main()-loop nicht selber programmiert
  - Handler für Maus- und Keyboard-Events, Listen-selektion, alles...
  - UI Thread nicht blockieren
- Plattformunabhängigkeit vs. natives Look & Feel
- Stichwort «Responsive UI»
  - Anpassung an unterschiedliche Bildschirmgrößen

# tkinter «Hello World»

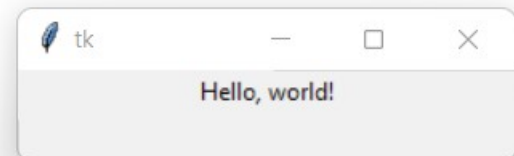
```
#!/usr/bin/env python3
# coding: utf8

import tkinter as tk

# create main window. This is usually called 'root'
root = tk.Tk()

# create a label and make it visible with pack
label = tk.Label(root, text="Hello, world!")
label.pack()

# run the program by executing the main loop
root.mainloop()
```



# Widgets

Label	This is a label
Button	<input type="button" value="Click me!"/>
Checkbox	<input checked="" type="checkbox"/> Choose me!
Radio buttons	<input checked="" type="radio"/> good choice <input type="radio"/> better choice
Textfield	<input type="text" value="input"/>
Spinbox	<input type="spin" value="0"/>
Combobox	<input type="text" value="option 2"/>
slider	<input type="range"/>

Verwende nach Möglichkeit die themed Widgets in ***tkinter.ttk***

Weitere:

- Liste
- Treeview
- (Label-)Frame
- Menues
- Images
- Dialogs & message boxes

Ressourcen:

- [pythontutorial](#)
- [tkdocs](#)



# Widgets

## **Aufgabe:**

- Studieren von *b1\_simple\_widgets\_demo.py*
- Herumstöbern in den angegebenen Ressourcen

## **Bitte beachten:**

Layouten und verknüpfen mit Funktionen/Events  
schauen wir nachher noch an

# Layouts

- **pack**
  - Von oben nach unten
  - Standardmässig vertikal oben & horizontal mittig ausgerichtet
- **grid**
  - Regelmässiges Gitter mit <auto> oder \* Spalten und Zeilen
- **place**
  - Absolutes positionieren
  - Selten verwendet

# pack layout

## Tutorial

### Wichtigste Optionen:

- ipadx, ipady
- padx, pady
- fill & expand
- anchor

internes padding (aka margin)  
padding

# grid layout

## Tutorial

### Wichtigste Optionen:

- column, row
- rowspan/columnspan
- sticky
- ipadx, ipady
- padx, pady

Zelle X/Y, obligatorisch

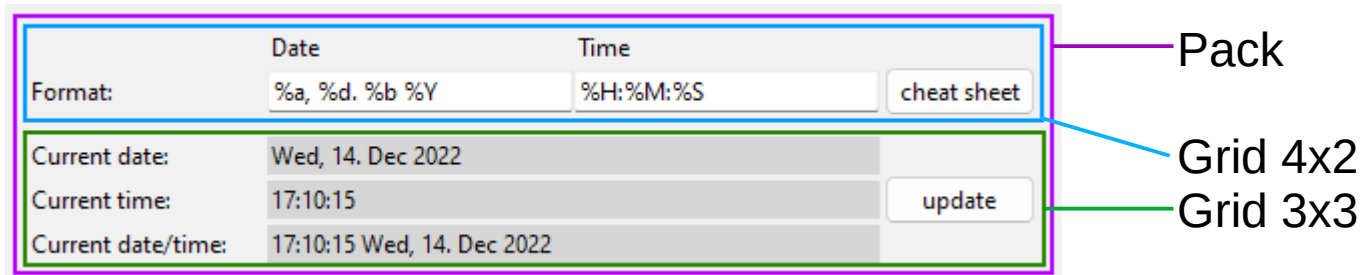
vergleichbar mit *anchor*  
internes padding (aka margin)  
padding

# Demo & Hands-on

- pack layout manager
- grid layout manager

# Verschachtelung

Reale Screens bestehen aus verschachtelten Frames



The screenshot shows a graphical user interface for a date and time application. It is annotated with three colored boxes and labels:

- Pack** (purple box): Points to the top section containing the 'Format:' label, 'Date' and 'Time' headers, input fields for '%a, %d. %b %Y' and '%H:%M:%S', and a 'cheat sheet' button.
- Grid 4x2** (blue box): Points to the bottom section containing three rows of date/time information and an 'update' button.
- Grid 3x3** (green box): Points to the bottom section, specifically the three rows of date/time information.

	Date	Time	
Format:	%a, %d. %b %Y	%H:%M:%S	cheat sheet
Current date:	Wed, 14. Dec 2022		update
Current time:	17:10:15		
Current date/time:	17:10:15 Wed, 14. Dec 2022		

# Events

- Einfach – command binding  
`widgets.Button(root, text="Click Me", command=button_clicked1)`
- Erweitert – event binding  
`btn = widgets.Button(root, text="Click Me")`  
`btn.bind("<Button>", button_clicked2)`
- Erweitertes Binding notwendig wenn kein Kommando zur Verfügung steht.
  - Listbox selection changes
  - Doppelklicks
  - Uvm.
- `command=x` und `bind(x)` erhalten die Funktion selber, nicht den Aufruf  
=> keine Klammern() am Schluss
- handler (Funktion) für command binding (hier: `button_clicked1`) hat kein Argument.  
Der handler für das erweiterte Binding (`button_clicked2`) hat ein Argument «event\_argument»

# Übung

- Demo-GUI (b1\_main.py und Abhängigkeiten (bX\_\_\_\_.py) durchgehen
- Debuggen, verstehen und testweise Änderungen vornehmen
- 3ter Tab einführen: Computerinformationen mit Modul «platform» darstellen