INFORMATICĂ · PYTHON · CLASA A IX-A · MEMORAREA DATELOR ȘI ORGANIZAREA CODULUI · LECȚIA 12

Interfețe grafice cu Tkinter (2): grid, place, Frame, Text, Canvas

Durată: 100 min Nivel: mate-info · Python Înainte: lecțiile py9-09, py9-11

CE VEI ȘTI DUPĂ LECȚIE

- Așezi elementele unei ferestre în tabel cu grid() sau la coordonate fixe cu place() și le grupezi cu Frame.
- Citești și scrii text pe mai multe rânduri cu Text, folosind indici de forma „linie.coloană”.
- Desenezi pe un Canvas linii, dreptunghiuri, cercuri și texte, în coordonatele ferestrei.
- Construiești, pas cu pas, o aplicație cu logica separată de interfață și o testezi.
- Reacționezi la apăsarea unei taste cu bind() și folosești Listbox și Menu.

1. grid: fereastra ca un tabel

`pack()` (lecția py9-11) pune elementele unul sub altul. Pentru formulare, unde vrei etichetă și casetă **pe același rând**, folosești `grid(row=…, column=…)`. Rândurile și coloanele încep de la 0.

```
import tkinter as tk

def calculeaza():
    l = float(caseta_l.get())
    L = float(caseta_L.get())
    rezultat.config(text=f"Aria: {l * L:.2f} m²")

f = tk.Tk()
f.title("Aria camerei")
tk.Label(f, text="Lățime (m):").grid(row=0, column=0, sticky="e", padx=5, pady=5)
caseta_l = tk.Entry(f, width=10)
caseta_l.grid(row=0, column=1)
tk.Label(f, text="Lungime (m):").grid(row=1, column=0, sticky="e", padx=5, pady=5)
caseta_L = tk.Entry(f, width=10)
caseta_L.grid(row=1, column=1)
tk.Button(f, text="Calculează", command=calculeaza).grid(row=2, column=0, columnspan=2)
rezultat = tk.Label(f, text="")
rezultat.grid(row=3, column=0, columnspan=2)
f.mainloop()
```

Pentru `3.5` și `4`, eticheta arată `Aria: 14.00 m²`.

Opțiune `grid`	Efect
`row`, `column`	rândul și coloana celulei
`columnspan=2`	elementul ocupă două coloane alăturate

Opțiune `grid`	Efect
`sticky="e"`	lipit de marginea din dreapta a celulei (`n`, `s`, `e`, `w` = nord, sud, est, vest; `"we"` = întins pe lățime)
`padx`, `pady`	spațiu în jurul elementului

2. place și regula celor trei metode

`place(x=…, y=…)` așază elementul la coordonate exacte, în pixeli, măsurate din **colțul stânga-sus** al ferestrei. E simplu, dar elementele nu se mai reașază când se schimbă textul, fontul sau mărimea ferestrei.

Pe scurt: `pack()` pentru câteva elemente unul sub altul (sau alăturate, cu `side="left"`), `grid()` pentru formulare și butoane în rânduri și coloane, `place()` rar, pentru poziții exacte (un desen, un joc).

> **ATENȚIE — GREȘELI FRECVENTE**
>
> În **același container** (fereastră sau `Frame`) nu amesteca `pack()` cu `grid()`. Python se oprește cu o eroare de tipul `TclError: cannot use geometry manager "grid" inside "." …`. Dacă ai nevoie de ambele, pune o parte într-un `Frame`.

3. Frame: grupuri de elemente

Un `Frame` este un dreptunghi gol, în care pui alte elemente. Pentru fereastră el este **un singur element**, așezat cu `grid` sau `pack`; în interiorul lui alegi altă metodă de așezare.

```
butoane = tk.Frame(f)                      # părintele este fereastra
butoane.grid(row=2, column=0, columnspan=2)
tk.Button(butoane, text="Calculează", command=calculeaza).pack(side="left", padx=5)
tk.Button(butoane, text="Șterge", command=sterge).pack(side="left", padx=5)
```

Cele două butoane stau alăturate, pe rândul 2 al grilei. Observă că părintele lor este `butoane`, nu `f`: de aceea `pack` nu intră în conflict cu `grid`.

4. Text: casetă cu mai multe rânduri

`Entry` ține un singur rând; `Text` ține oricâte. Pozițiile se scriu ca text, `"linie.coloană"`: liniile încep de la **1**, coloanele de la **0**, iar `"end"` înseamnă „după ultimul caracter”.

```
t = tk.Text(f, width=30, height=5)
t.insert("end", "prima linie\n")
t.insert("end", "a doua linie")
print(repr(t.get("1.0", "end")))      # 'prima linie\na doua linie\n'
print(repr(t.get("2.0", "2.5")))      # 'a dou'
print(repr(t.get("1.0", "1.end")))    # 'prima linie'
t.delete("1.0", "end")                # golește caseta
```

Comparativ cu `Entry`: `get("1.0", "end")` în loc de `get()`, `insert("1.0", …)` sau `insert("end", …)` în loc de `insert(0, …)`, `delete("1.0", "end")` în loc de `delete(0, tk.END)`.

> **SFAT**
>
> `t.get("1.0", "end")` adaugă mereu un `\n` la final. Când citești numere, `split()` elimină singur spațiile și sfârșiturile de rând: `t.get("1.0", "end").split()` pentru textul `9 10` și `7` pe rândul următor dă `['9', '10', '7']`.

5. Canvas: o pânză pentru desen

Pe un `Canvas` desenezi figuri. Originea `(0, 0)` este în **colțul stânga-sus**, iar `y` crește **în jos** (invers față de axa de la matematică).

```
panza = tk.Canvas(f, width=200, height=100, bg="white")
panza.pack()
panza.create_line(0, 0, 200, 100)                        # din (0,0) în (200,100)
d = panza.create_rectangle(10, 20, 60, 80, fill="blue")  # colțurile opuse
panza.create_oval(100, 10, 140, 50, fill="red")          # cercul din pătratul dat
panza.create_text(100, 90, text="Salut")                 # textul centrat în (100,90)
print(d, panza.coords(d))         # 2 [10.0, 20.0, 60.0, 80.0]
panza.move(d, 5, 0)               # mută dreptunghiul 5 pixeli la dreapta
panza.delete("all")               # șterge tot desenul
```

Fiecare `create_…` întoarce un **număr de identificare** al figurii (1, 2, 3, … în ordinea desenării), pe care îl folosești ca să o muți (`move`), să-i afli coordonatele (`coords`) sau s-o ștergi (`delete`).

6. Mini-proiect ghidat: calculatorul de medii

Cerința. Utilizatorul scrie numele unui elev și notele lui (întregi de la 1 la 10, separate prin spații sau pe rânduri). La „Calculează” apare media cu două zecimale, media rotunjită și un grafic cu bare (roșu pentru notele sub 5). „Șterge” golește totul.

Pasul 1: logica, fără ferestre, testată în consolă. (`note` este o listă, ca în mini-introducerea din lecția py9-09: `note.append(x)` adaugă o valoare, `note[i]` este elementul de pe poziția `i`.)

```
def citeste_note(text):
    """Lista notelor (întregi 1..10) sau None dacă textul este greșit."""
    note = []
    for t in text.split():
        if not t.isdigit() or not 1 <= int(t) <= 10:
            return None
        note.append(int(t))
    if len(note) == 0:
        return None
    return note

def medie(note):
    return sum(note) / len(note)

def rotunjeste(m):
    return int(m + 0.5)             # ,50 se rotunjește în sus
```

Test	Rezultat
`citeste_note("9 10 7")`	`[9, 10, 7]`
`citeste_note("8\n9\n")`	`[8, 9]`
`citeste_note("")`, `("9 11")`, `("9 abc")`, `("7.5 8")`	`None`
`medie([9, 10, 7])`, `rotunjeste(...)`	`8.666666666666666`, `9`

> **ATENȚIE — GREȘELI FRECVENTE**
>
> `round(8.5)` dă `8`, iar `round(9.5)` dă `10`: Python rotunjește valorile aflate exact la jumătate spre numărul **par**. Pentru regula „de la ,50 în sus" scrie tu funcția, ca `rotunjeste` de mai sus.

Pasul 2: schița interfeței pe grilă. Rândul 0: „Elev:" + `Entry`; rândul 1: „Note:" + `Text`; rândul 2: un `Frame` cu cele două butoane; rândul 3: eticheta rezultat; rândul 4: `Canvas`.

Pasul 3: funcțiile butoanelor (doar citesc, apelează logica și afișează):

```
def calculeaza():
    note = citeste_note(text_note.get("1.0", "end"))
    if note is None:
        messagebox.showerror("Eroare", "Notele trebuie să fie întregi de la 1 la 10.")
        return
    m = medie(note)
    nume = caseta_nume.get().strip() or "Elevul"     # text gol → "Elevul"
    rezultat.config(text=f"{nume}: media {m:.2f}, rotunjită {rotunjeste(m)}")
    deseneaza(note)

def deseneaza(note):
    panza.delete("all")
    for i in range(len(note)):
        x = 10 + i * 30                    # bara i, lată de 25 de pixeli
        y = 125 - note[i] * 10             # nota 10 → bară înaltă de 100 de pixeli
        culoare = "tomato" if note[i] < 5 else "steelblue"
        panza.create_rectangle(x, y, x + 25, 125, fill=culoare)
        panza.create_text(x + 12, y - 8, text=note[i])

def sterge():
    caseta_nume.delete(0, tk.END)
    text_note.delete("1.0", "end")
    rezultat.config(text="")
    panza.delete("all")
```

Pasul 4: fereastra. Construiești elementele după schița de la pasul 2 (`caseta_nume`, `text_note`, cadrul `butoane` din secțiunea 3, `rezultat` și `panza = tk.Canvas(f, width=320, height=130, bg="white")`), cu `grid`, apoi `f.mainloop()`.

Pasul 5: testarea interfeței. Pentru `Ana` și notele `9 10 7` / `4 8` apare `Ana: media 7.60, rotunjită 8` și cinci bare, a patra roșie. Pentru `9 x` apare mesajul de eroare. După „Șterge", caseta, textul, eticheta și pânza sunt goale.

7. Tastă apăsată, Listbox și meniu

Trei elemente des întâlnite, într-o singură aplicație: o listă de sarcini. Scrii o sarcină în casetă și apeși **Enter** (sau butonul), sarcina apare într-un `Listbox`, iar din meniul **Fișier** ștergi sarcina aleasă.

```
import tkinter as tk

f = tk.Tk()
f.title("Sarcini")

def adauga(event=None):                  # Enter trimite și un obiect „eveniment”
    text = caseta.get().strip()
    if text != "":
        lista.insert("end", text)        # adaugă la sfârșitul listei vizuale
        caseta.delete(0, tk.END)

def sterge_selectia():
    sel = lista.curselection()           # pozițiile alese; este goală dacă nu e nimic ales
    if len(sel) > 0:
        lista.delete(sel[0])

caseta = tk.Entry(f, width=24)
caseta.grid(row=0, column=0)
caseta.bind("<Return>", adauga)          # tasta Enter în casetă apelează adauga
tk.Button(f, text="Adaugă", command=adauga).grid(row=0, column=1)
lista = tk.Listbox(f, height=6, width=30)
lista.grid(row=1, column=0, columnspan=2)

meniu = tk.Menu(f)
f.config(menu=meniu)                     # bara de meniu a ferestrei
m_fisier = tk.Menu(meniu, tearoff=0)
meniu.add_cascade(label="Fișier", menu=m_fisier)
m_fisier.add_command(label="Șterge selecția", command=sterge_selectia)
m_fisier.add_command(label="Ieșire", command=f.destroy)
f.mainloop()
```

Ce vezi	Ce face
`w.bind("<Return>", functie)`	apelează `functie` când e apăsat Enter și `w` are focusul; `functie` primește un parametru (evenimentul), de aceea `event=None`
alte evenimente	`"<Key>"` (orice tastă; `event.keysym` e numele ei), `"<Button-1>"` (clic stânga)
`Listbox`: `insert("end", t)`, `curselection()`, `delete(i)`, `get(i)`	adaugă, află poziția aleasă, șterge, citește elementul de la poziția `i`
`Menu`: `add_cascade`, `add_command(label=…, command=…)`	un meniu principal cu submeniuri; fiecare comandă apelează o funcție, ca un buton

DE ȘTIUT

Același `command=` pentru buton și pentru meniu înseamnă că logica rămâne într-o singură funcție, iar interfața doar o apelează (ca la calculatorul de medii). Alte clase (`tk.Checkbutton` , metoda `f.after(500, functie)` pentru animații) le găsești în documentația modulului *tkinter*.

8. Verifică-te

1. Ce scrii ca un buton să ocupe rândul 3, pe lățimea a două coloane?
2. Ce se întâmplă dacă în aceeași fereastră scrii `a.pack()` și apoi `b.grid(row=0, column=0)` ? Cum rezolvi?
3. Caseta `Text` conține `Ana\nare mere` . Ce întoarce `t.get("2.0", "2.3")` ?
4. Pe un `Canvas` de 200 × 100, ce coordonate are colțul din dreapta-jos?
5. Desenează (cu o singură instrucțiune) un cerc cu centrul în `(50, 50)` și raza 20.
6. De ce funcția `citeste_note` întoarce `None` în loc să afișeze ea mesajul de eroare?

Răspunsuri

1. `b.grid(row=3, column=0, columnspan=2)` .
2. `TclError` : un container nu acceptă două metode de așezare. Folosești doar una sau pui unul dintre elemente într-un `Frame` .
3. `'are'` .
4. `(200, 100)` : `x` crește spre dreapta, `y` în jos.
5. `panza.create_oval(30, 30, 70, 70)` .
6. Ca să rămână logică pură, testabilă în consolă și refolosibilă; decizia cum anunți eroarea (mesaj, etichetă roșie) aparține interfeței.