FIȘĂ DE LUCRU · PYTHON · CLASA A IX-A · FIȘA 11 · LECȚIA PY9-11

Interfețe grafice cu Tkinter (1): fereastră, etichete, butoane, casete text

Durată: 100 min Nivel: mate-info · Python

CE VEI ȘTI DUPĂ LECȚIE

- Prezici ce arată fereastra după o serie de clicuri și operații pe Entry.
- Repari și completezi aplicații mici cu Label, Button, Entry și messagebox.
- Construiești aplicații cu logica într-o funcție separată, testată pe cazuri normale, limită și de eroare.

Ce trebuie să știi (o pagină)

Ce vrei	Cum scrii
bibliotecile	`import tkinter as tk` · `from tkinter import messagebox`
fereastra	`f = tk.Tk()` · `f.title("…")` · `f.geometry("300x200")` · la final `f.mainloop()`
etichetă	`et = tk.Label(f, text="…", font=("Arial", 14), fg="red", bg="white")`
buton	`tk.Button(f, text="OK", command=functie)`: **fără paranteze** după `functie`
casetă text	`cs = tk.Entry(f, width=20)`
așezare	`w.pack()` (unul sub altul, în ordinea apelurilor), cu `padx=…`, `pady=…`
citești din casetă	`cs.get()` dă **text**: `int(cs.get())`, `float(cs.get())`
golești / scrii în casetă	`cs.delete(0, tk.END)` · `cs.insert(0, "…")` sau `cs.insert(tk.END, "…")`
schimbi / citești o opțiune	`et.config(text="…")` sau `et["text"] = "…"` · `et.cget("text")`
mesaje	`messagebox.showinfo(titlu, mesaj)`, `showwarning`, `showerror`, `askyesno` (dă `True` / `False`)
verifici un natural	`t.strip().isdigit()` (fals pentru `""`, `"-5"`, `"2.5"`)
verifici un real	`try: x = float(t)` · `except ValueError: …`

Ce vrei	Cum scrii
modifici o variabilă globală dintr-o funcție	`global contor` la începutul funcției
organizare	calculul într-o funcție fără nimic grafic; funcția butonului: citește → verifică → calculează → afișează

> **SFAT**
>
> La problemele de nivel 3, testează **întâi funcția de calcul** în consolă (fără fereastră), apoi interfața, pe fiecare rând din tabelul de teste, inclusiv pe cazurile de eroare.

Nivelul 1 · Prezice

★ citește și prezice

P1. Ce etichete apar în fereastră și în ce ordine, de sus în jos?

```
a = tk.Label(f, text="A")
b = tk.Label(f, text="B")
c = tk.Label(f, text="C")
c.pack()
a.pack()
```

P2. Eticheta pornește cu textul `NU`. Ce text are după **trei** clicuri pe buton?

```
def schimba():
    if et.cget("text") == "DA":
        et.config(text="NU")
    else:
        et.config(text="DA")

et = tk.Label(f, text="NU")
bt = tk.Button(f, text="Schimbă", command=schimba)
```

P3. Caseta `e` este goală. Ce întoarce `e.get()` după rândurile de mai jos și ce tip are rezultatul?

```
e.insert(0, "23")
e.insert(tk.END, "4")
e.insert(1, "x")
```

P4. Utilizatorul apasă butonul de 3 ori. De câte ori apare `salut` în consolă, în total?

```
def f():
    print("salut")

b = tk.Button(fereastra, text="Salut", command=f())
b.pack()
```

P5. În casetă este `7`. Ce text arată eticheta după `eticheta.config(text=caseta.get() * 3)` ?

P6. Ce text arată eticheta când apare fereastra? Vezi utilizatorul celelalte valori?

```
for i in range(5):
    eticheta.config(text=i)
fereastra.mainloop()
```

Nivelul 2 · Găsește și repară

★★ găsește și repară

P7. Programul trebuie să afișeze în etichetă pătratul numărului din casetă. Are **trei** greșeli. Găsește-le și scrie varianta corectă.

```
import tkinter

def calc():
    n = int(caseta.get)
    rez.config(text=n * n)

f = tk.Tk()
caseta = tk.Entry(f)
caseta.pack()
tk.Button(f, text="Pătrat", command=calc()).pack()
rez = tk.Label(f, text="")
rez.pack()
f.mainloop()
```

P8. La clic apare `AttributeError: 'NoneType' object has no attribute 'get'`. Explică eroarea și repar-o fără să schimbi felul în care arată fereastra.

```
caseta = tk.Entry(f).pack()
...
def citeste():
    text = caseta.get()
```

P9. *Completează.* Aplicația transformă grade Celsius în Fahrenheit (`F = C · 9 / 5 + 32`) și afișează rezultatul cu o zecimală. Pentru `100` trebuie să arate `212.0 °F`, iar pentru `abc` un mesaj de eroare.

```
import tkinter as tk
from tkinter import messagebox

def in_fahrenheit(c):
    return ________

def converteste():
    try:
        c = ________(caseta.________())
    except ________:
        messagebox.________("Eroare", "Temperatura nu este un număr.")
        return
    rezultat.config(text=f"{________:.1f} °F")

fereastra = tk.Tk()
caseta = tk.Entry(fereastra)
caseta.pack()
tk.Button(fereastra, text="Convertește", command=________).pack()
rezultat = tk.Label(fereastra, text="")
rezultat.pack()
fereastra.mainloop()
```

P10. *Rânduri amestecate.* Ordonează rândurile (indentarea o stabilești tu) ca aplicația să afișeze într-un `messagebox` cubul numărului din casetă. Un rând este în plus: care și de ce?

```
A)  f.mainloop()
B)  def arata():
C)  caseta = tk.Entry(f)
D)  n = int(input())
E)  messagebox.showinfo("Cub", f"{n} la cub este {n ** 3}")
F)  f = tk.Tk()
G)  n = int(caseta.get())
H)  tk.Button(f, text="Calculează", command=arata).pack()
I)  caseta.pack()
```

(Primele două rânduri, cele două `import`-uri, sunt deja scrise.)

P11. La primul clic pe `+1` apare `UnboundLocalError: cannot access local variable 'contor' …`. De ce? Repară-l adăugând un singur rând.

```
contor = 0
def creste():
    contor += 1
    et.config(text=contor)
```

P12. Verificarea de mai jos lasă să treacă și textul `abc`, iar programul se oprește cu `ValueError`. Unde este greșeala?

```
if caseta.get().isdigit:
    messagebox.showinfo("Pătrat", int(caseta.get()) ** 2)
```

P13. Un elev scrie doar `import tkinter as tk` și apoi `tk.messagebox.showinfo("a", "b")`. Obține `AttributeError: module 'tkinter' has no attribute 'messagebox'`. Ce lipsește?

Nivelul 3 · Construiește

★★ scrie cod

Pentru fiecare aplicație: o funcție de calcul fără elemente grafice + o funcție legată de buton. Verifică **toate** testele.

P14. Divizori comuni. Două casete, un buton. Un `messagebox` afișează câți divizori comuni au cele două numere (indiciu: divizorii comuni ai lui `a` și `b` sunt divizorii lui `math.gcd(a, b)`). Pornește de la aplicația din lecția py9-11, secțiunea 6.

caseta 1	caseta 2	mesaj
`12`	`18`	`12 și 18 au 4 divizori comuni.`
`7`	`13`	`7 și 13 au 1 divizori comuni.`
`36`	`36`	`36 și 36 au 9 divizori comuni.`
`a` sau `0`	`5`	mesaj de eroare

P15. Viteza medie. Casete pentru distanță (km) și timp (minute); eticheta afișează viteza medie în km/h, cu o zecimală.

distanță	timp	rezultat
`30`	`20`	`90.0 km/h`
`42.195`	`150`	`16.9 km/h`
`5`	`60`	`5.0 km/h`
`10`	`0`	eroare: timpul trebuie să fie pozitiv
`2,5`	`10`	eroare: nu este număr

P16. Ghicește numărul. Calculatorul alege un număr între 1 și 100. O casetă, un buton „Verifică” și o etichetă care arată `mai mare`, `mai mic` sau `Ai ghicit în k încercări!`. Caseta se golește după fiecare încercare. Funcția de logică: `verdict(secret, incercare)`. Test cu `secret = 37`:

încercări, în ordine	eticheta după fiecare
`50`, `20`, `abc`, `37`	`mai mic`, `mai mare`, mesaj de eroare (nu se numără), `Ai ghicit în 3 încercări!`

P17. Contor de persoane. O etichetă mare cu un număr (pornește de la 0) și trei butoane: `+1`, `−1`, `Reset`. Contorul **nu scade sub 0**.

clicuri	eticheta
`+1` de 3 ori	`3`
apoi `−1` de 5 ori	`2`, `1`, `0`, `0`, `0`
apoi `+1`, apoi `Reset`	`1`, apoi `0`

P18. Palindromul. O casetă și un buton. Eticheta arată `palindrom` (cu verde, `fg="green"`) sau `nu este palindrom` (cu roșu). Spațiile și majusculele nu contează.

text	rezultat
`Ele fac cafele`	`palindrom`
`Capac`	`palindrom`
`a`	`palindrom`
`Python`	`nu este palindrom`

P19. Următorul termen. Fereastra are fundal colorat (`fereastra.config(bg="lightyellow")`), o etichetă mare și butonul „Următorul termen". La fiecare clic, eticheta afișează următorul termen al șirului Fibonacci. După 8 clicuri s-au afișat, pe rând: `1 1 2 3 5 8 13 21` .

Nivelul 4 · Pune mintea la contribuție

★★★ pune mintea la contribuție

P20. Consolă sau fereastră? Scrie varianta de consolă a aplicației P14 (folosind aceeași funcție de calcul). Dă două avantaje ale fiecărei variante. Care se poate testa mai ușor pe 1000 de perechi de numere? De ce?

P21. „User-friendly". Aplicația unui coleg are: titlul ferestrei `tk` , un buton cu textul `Buton1` , o casetă fără nicio etichetă lângă ea, rezultatul afișat cu `print` în consolă, iar la o literă scrisă în casetă programul se oprește cu eroare. Propune **cinci** îmbunătățiri concrete și scrie, pentru fiecare, instrucțiunea Python care o realizează.

P22. Plan de teste. Pentru aplicația P15, completează un tabel cu cel puțin 7 teste: 2 normale, 2 la limită, 3 de eroare. Pentru fiecare, scrie ce **ar trebui** să se întâmple. Ce dă `float()` pentru `" 3 "` , `"-3"` , `"1e3"` , `"2,5"` și `""` ?

P23. Teste automate. Scrie cinci rânduri `assert` care verifică funcția `nr_divizori_comuni` din P14 (de exemplu `assert nr_divizori_comuni(12, 18) == 4`). De ce nu poți scrie la fel de ușor un test automat pentru funcția legată de buton?

P24. Mini-proiect: casa de marcat. Casete pentru preț și suma primită (lei, numere naturale). Eticheta arată restul descompus în bancnote de 50, 10, 5 și 1 leu, folosind cât mai puține bancnote. Dacă suma nu ajunge, un `showwarning` spune câți lei mai trebuie. Logica: `rest(pret, dat)` întoarce numărul de bancnote din fiecare valoare.

preț	primit	rezultat
`37`	`100`	`Rest 63 lei: 1 x 50 lei, 1 x 10 lei, 3 x 1 lei`
`88`	`200`	`Rest 112 lei: 2 x 50 lei, 1 x 10 lei, 2 x 1 lei`
`50`	`50`	`Fără rest.`
`120`	`100`	avertisment: `Mai trebuie 20 lei.`
`12x`	`20`	mesaj de eroare