CURS 4

FUNCȚII

- def denumire_functie (parametrii formali):
- apelare:
  - s = functie (param)
  - s = functie (param1 = 3 , param2 = 4)
- parametrii pot avea val. implicite:
  - s = functie () => s = 0
  - s = functie (nr) => s = nr (ex: 6)

ex: def suma_dif(x, y):
    return x+y, x-y

s, d = suma_dif(3,7) => $\begin{cases} s = 10 \\ d = -4 \end{cases}$

t = suma_dif(3,7) => t = (10, -4)
t[0] = 10

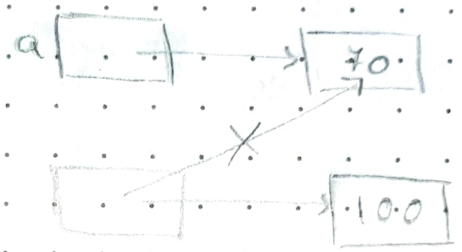
TRANSMITEREA PARAMETRILOR

- call by object reference

ex: def f(x):

    x = 100

a = 70
f(a)
print (a) => 70



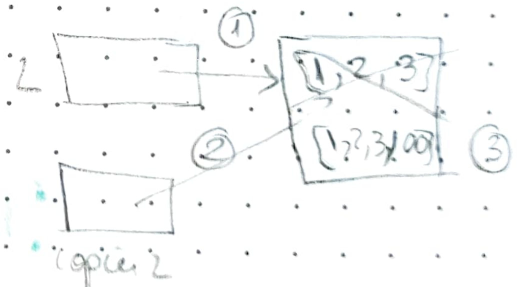
ex: def f(L):

    L.append(100) ③

L = [1,2,3] ①

f(L) ②

print (L) => [1,2,3,100]



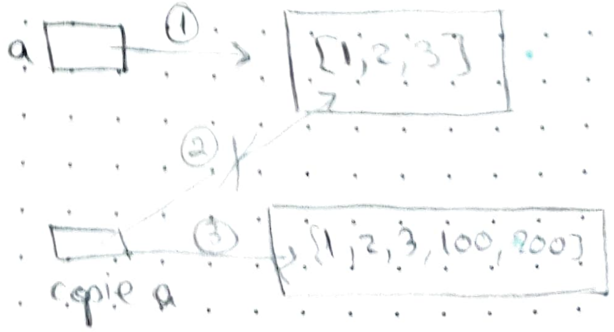
ex: def f(L): ③

    L = L + [100,200]

a = [1,2,3] ①

f(a) ②

print (a) => [1,2,3]



```
ex: def f(L):
        aux = [x for x in L if x>0]
        L.clear()          # L=aux nu merge.
        L.extend(aux)
```

Funcții cu nr variabil de parametrii

- suma (x, y): ?

```
s = suma (1,2)
s = suma (1,2,3)
s = suma (1,2,3,4)
def suma (*numere): -> funcția va primi nr. var. de param.
    s=0
    for x in numere:
        s=s+x
    return s
```

-

```
def suma (*numere, prag):  # sau (prag, *numere)
    s=0                        ~~s=suma (1,7,3,4)~~
    for x in numere:           s=suma (1,7,3,4, prag=4)
        if x>=prag:
            s=s+x
    return s
```

FUNCȚII ANONIME (LAMBDAS)

- lambda param: expresie

ex: lambda x,y : x+y

ex: lambda n : sum([int(c) for c in str(n)])

calc suma cfr. lui n

ex: s=(lambda x,y : x+y)(5,7)

s = 12 ; se apelează în ac. timp

ex: f = lambda x,y : x+y

s = f(5,7)

- funcții care returnează funcții (dispatchuri)

```
def disp (tip):
    if tip == "suma":
        return lambda x,y: x+y
    elif tip == "produs":
        return lambda x,y: x·y

f = disp(" suma ")
s = f(5,7)   ⟹ s = 12
```

MECANISMUL DE CALLBACK

- $S_k = \sum_{i=1}^{n} f_k(i)$

$S_1 = 1+2+\dots+n$ $\quad f_1(i) = i$

$S_2 = 1+\frac{1}{2}+\dots+1/n$ $\quad f_2(i) = 1/i$

$S_3 = e+e^2+\dots+e^n$ $\quad f_3(i) = e^i$

```
• def. suma_generică (n, fk):
      s = 0
      for i in range(1, n+1):
          s = s + fk(i)        → call back
      return s;
• def f1(i):
      return i

  s1 = suma_generica (10, f1)
       ||
  s2 = suma_generică (10, lambda i: i)
  s3 = suma_generica (10, lambda i: 1/i)
  s4 = suma_generica (10, lambda i: e**i)
```

SORTAREA DATELOR

- Element $\longrightarrow$ cheie de sortare
- Funcție predefinită: sorted (colecție) => L = sorted (L)

  ex: sorted ([5, 1, 7, 3]) = [1, 3, 5, 7]

  sorted ("test") = ["t", "e", "s", "t"]

Mereu îți întoarce o listă

- Clasa list $\longrightarrow$ sort() (metoda) => L.sort()

```
L = [154, 213, 12, 54, 133, 7, 10]
S = sorted (L, key = x : x % 10)
L = [10, 12, 213, 133, 154, 54, 7]
```