

## TUTORIAT 7

Linkuri utile:

- <https://www.python.org/>
- <https://docs.python.org/3/>
- <https://www.codecademy.com/learn/learn-python-3>
- <https://www.learnpython.org/>
- <https://stanfordpython.com/#/>

Comentariu multiplu PyCharm: CTRL + /

Ce conține tutoriatul ?

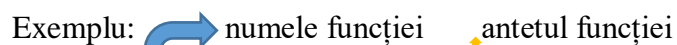
- I. Funcții
  - Definiere
  - Parametrii
  - Valoarea returnată
  - Transmiterea parametrilor
- II. Fișiere text
  - Citire din fișier
  - Scrierea în fișier
- III. Metoda Greedy
- IV. Exerciții

I.Funcții

### Definiere

**def** nume\_functie (**parametrii**):

// instructiuni

Exemplu: 

**def** suma ( a, b): a, b parametrii formali

sum = 0

for i in range(a, b+1):

sum += i

return sum

} instrucțiuni

Apel funcție:

result = suma(11,12) 11, 12 parametrii actuali



## Parametrii

➔ Funcțiile pot avea număr variabil de parametri

Cum sunt specificați?	sunt prefixați de operatorul * în antet
Pot exista parametri după cei prefixați cu * ?	<b>DA</b> , specificați prin doar <b>prin nume</b>

### !ATENȚIE!

În Python nu se pot defini mai multe funcții cu același nume și număr diferit de parametri(va fi luată în considerare doar ultima funcție declarată)

➔ Specificarea parametrilor

Parametrii **OBLIGATORII**(au valori asociate la apel)

Modul de specificare	Reguli la apel
prin poziție	se respectă numărul și ordinea din antet
prin nume(nume_param = expresie)	se respectă numărul, nu și ordinea
poziție și nume	întâi cei prin poziție, apoi cei prin nume

Parametrii cu valoare implicită (**DEFAULT**)

Pentru ce se folosesc ?	sunt utilizați în cazul în care parametrul nu se specifică la apel
Unde se specifică ?	în antet, după parametrii obligatorii (ulimii)

## Valoarea returnată

- ➔ **return** valoare, implicit întorc None
- ➔ pot fi returnate **mai multe valori** de orice tip ( **împachetate într-un tuplu**)
- ➔ rezultatul se poate despacheta în variabile (**număr variabile = număr valori returnate** ( numărul de elemente din tuplul întors))

## Transmiterea parametrilor

parametrii -> nume pentru obiecte definite în spațiul funcției

Modul de transmitere: prin referință la obiect(se transmit referințe de obiecte prin valoare);  
modificarea referinței respective în interiorul funcției nu este vizibilă în exteriorul funcției(**modificarea valorii, DA**)

### !ATENȚIE!

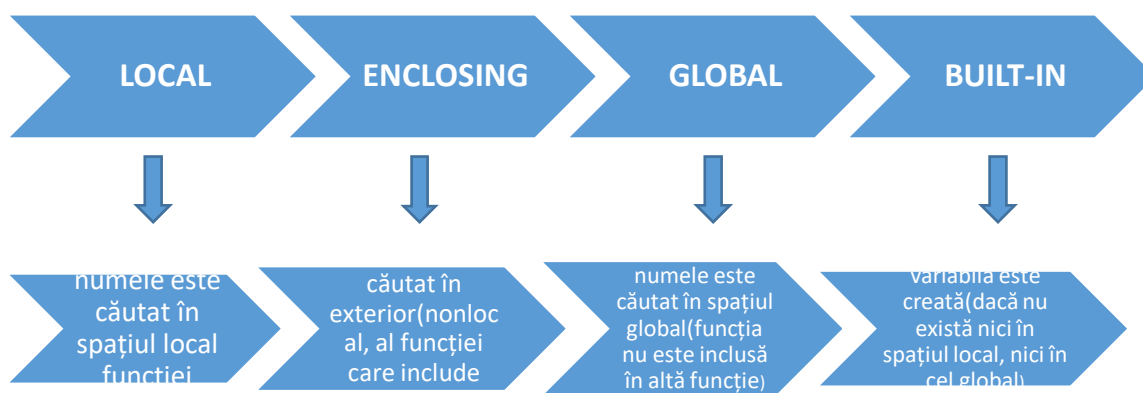
Dacă parametrul actual este un obiect mutabil, atunci modificarea valorii parametrului se face asupra obiectului transmis ca parametru (valoarea RĂMÂNE modificată după terminarea execuției funcției)

## Vizibilitate variabile

- ➔ variabilele pot fi globale sau locale (variabila este creată în momentul în care i-a fost atribuită pentru prima dată o valoare, `nume_variabila = valoare`)

Variabile <b>GLOBALE</b>	Variabile <b>LOCALE</b>
i s-a atribuit prima dată o valoare <b>în afara</b> oricărei funcții	i s-a atribuit prima dată o valoare <b>în interiorul</b> unei funcții
vizibilă global	vizibilă DOAR local (în interiorul funcției)

În momentul în care este accesată valoarea unei variabile într-o funcție, procedeul are loc după regula LEGB:



Pentru a modifica valoarea unei variabile din domeniul global în interiorul unei funcții se utilizează sintagma **global nume\_variabila** (numele variabilei este căutat în domeniul global; necesar doar pentru atribuiri, **NU** și pentru accesarea unei variabile).

### Exemple

1. produsul pătratelor numerelor din intervalul a, b

```
def produs(a, b):
    prod = 1
    for i in range(a, b+1):
        prod *= i
    return prod
my_product = produs(1,5)
```

2. def inmultiri(a, b):

```
    return a * b, a *2, b*3
```

```
produs, dublu, triplu = inmultiri(5, 10)
```

```
produs, *altele = inmultiri(5, 10)
```

## 3. valoare default

```
def aduna(a, b = 0):
    return a + b
suma1 = aduna(10, 4)
suma2 = aduna(10)
```

## 4. număr variabil de parametrii

```
def inmultire(*numere):
    produs = 1
    for numar in numere:
        produs *= numar

    return produs
print(inmultire(2,4), inmultire(5,6,7))

def mai_mari(*numere, numar):
    nr = 0
    for i in numere:
        if i > numar:
            nr += 1

    return nr
print(mai_mari(1, 5, 12, 14, 18))
print(mai_mari(1, 5, 12, 14, numar = 18))
```

## 5. transmitere parametrului

```
def modific_valoare(a):
    a = 10
    print(a)
b = 12
modific(12)
print(b)

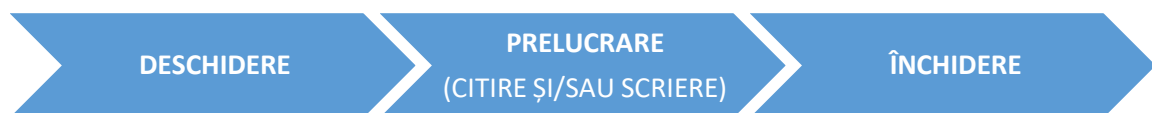
def modific_lista(ls, a):
    ls.append(a)
    ls = [15, 20]
    modific_lista(ls, 25)
    print(ls)
```

## 6. vizibilitate variabile

```
def modific_global():
    global numar
    print(numar)
    numar = 10
numar = 2
modific_global()
print(numar)
```

**II. Fișiere text**

Modul de lucru cu fișiere( secvențe de caractere organizate pe linii, stocate în memorie)

**1.Deschidere** (se asociază o variabilă unui fișier)

Metode de deschidere:

**open("cale\_fisier", "mod")** -> întoarce un obiect asociat fișierului având calea cale\_fisier(dacă fișierul este în directorul curent, atunci **cale\_fisier** = nume\_fisier)

Mod	Utilizare	Efect
"r"	read(citire)	EROARE dacă fișierul nu există
"w"	write(scriere)	dacă fișierul nu există, atunci el este creat, altfel suprascrie fișierul existent
"a"	append(scriere la sfârșit)	dacă fișierul nu există, atunci el este creat, altfel se va scrie la finalul fișierul existent

**with open("cale\_fisier", "mod") as nume\_variabila\_fisier:** (fișierul nu trebuie închis cu CLOSE)

## 2.Prelucrare

### i. Citire din fișier

Metoda	Apel	Valoare returnată
read()	variabila_fisier.read()	un șir de caractere cu tot conținutul fișierului
readline()	variabila_fisier.readline()	un șir de caractere cu linia curentă din fișier; șirul vid dacă s-a ajuns la sfârșitul fișierului
readlines()	variabila_fisier.readlines()	listă de șiruri de caractere cu toate liniile din fișier(inclusiv caracterul de sfârșit de linie, omis pentru ultima linie din fișier)

### ii. Scriere în fișier

Metoda	Apel	Valoare returnată
write	variabila_fisier.write(sir_caractere)	scrie sir_caractere în fișier, fără delimitator de linie
writelines	variabila_fisier.writelines(colectie_de_siruri)	scrie unul după altul elementele colecției, fără delimitator de linie

## 3.Închidere

**variabila\_fisier.close()** la finalizarea lucrului cu fișierul corespunzător variabilei nume\_fisier

### Exemple

- citire linie cu linie  

```
with open('fisier.in', 'r') as f:
    line = f.readline()
    while line != '':
        print(line, end=' ')
        line = f.readline()
```
- f = open('fisier.in', 'r')  
for line in f:
- scriere în fișier  

```
f = open('iesire.out', 'w')
nume = 'Ana'
varsta = 10
f.write(nume)
f.write(varsta) #EROARE, varsta nu este str
f.write("\n"+str(varsta))
f.close()
```

```
print(line, end = ' ')
f.close()
```

### III. Metoda Greedy

- ➔ utilizată în probleme de optimizare în care se poate obține optimul global(final) prin alegeri succesive ale optimului local(la un moment dat)
- ➔ este eficientă și rapidă, deoarece nu se fac reveniri la deciziile luate
- ➔ soluțiile trebuie demonstrate teoretic(demonstrarea corectitudinii, a faptului că modul de construcție a soluției conduce la soluția optimă; de obicei prin inducție matematică sau reducere la absurd)
- ➔ este o particularizare a metodei Backtracking(s-a renunțat la mecanismul de revenire)
- ➔ **Backtracking** obține **TOATE** soluțiile, **Greedy O** soluție( nu neapărat optimă)
- ➔ soluțiile obținute sunt sub forma unui vector
- ➔ timpul de lucru este polinomial ( $O(n^k)$ )

#### Model general de probleme ce se rezolvă cu tehnica Greedy

Se dă: o mulțime de candidați

Se cere: submulțime S, îndeplinirea unor condiții, optimizarea(maximizarea, minimizarea) unei funcții obiectiv

Metoda se aplică în două variante: cu prelucrarea inițială a mulțimii de candidați sau fără.

#### Elementele modelului Greedy

**Soluție:** se verifică dacă o mulțime de candidați reprezintă o soluție

**Acceptabil:** se verifică dacă o mulțime de candidați poate fi completată cu un candidat pentru a obține o soluție

**Selecție:** se alege dintre candidații nealeși cel mai promițător candidat(alegerea optimului local)

#### Descrierea generală a funcției Greedy

```
greedy(C) // C mulțimea candidaților
```

```
    S ← ∅ // S mulțimea în care se construiește soluția
```

```
    cât timp not solutie(S) and C ≠ ∅ execută
```

```
        x ← un element din C care maximizeaza/minimizeaza select(x)
```

```
        C ← C - {x}
```

```
        dacă acceptabil(S ∪ {x}) atunci S ← S ∪ {x}
```

```
    dacă solutie(S) atunci return S
```

```
    altfel return "nu exista solutie"
```

**Exemplul 1 (Problema numerelor naturale prime și neprime)**

Se dă un set de numere naturale și se cere alegerea a cel mult  $k$  (dat) numere prime și oricâte neprime, astfel încât suma celor prime să depășească suma celor neprime, iar numărul elementelor alese (prime și neprime) să fie maxim.

Se dă: o mulțime de candidați  $C$  (mulțimea numerelor date)

Se cere: o submulțime  $S \subseteq C$  (submulțimea numerelor prime și neprime) care:

- îndeplinește condiții interne (acceptabil)
- este optimă (selecție)

Soluție:

- ➔ se ordonează numerele prime descrescător, apoi cele neprime crescător
- ➔ se alege cel mai mare număr prim, dacă există, altfel cel mai mic
- ➔ se verifică dacă sunt cel mult  $k$  numere prime și suma lor este mai mare

**Exemplul 2 (Medici)**

O asociație caritabilă asigură consultații medicale gratuite pentru cei fără posibilități materiale. Există un singur cabinet dotat cu aparatură medicală, din acest motiv la un moment dat un singur medic poate face consultații. Asociația apelează la  $n$  medici de diverse specialități, care își oferă benevol serviciile. Fiecare prezintă un singur interval  $[s_i, f_i]$  de-a lungul aceleiași zile, în care este disponibil. Ajuțați asociația să realizeze o programare a consultațiilor în cabinet, astfel încât numărul de medici să fie maxim.

**Exemplu:**

4 # numărul medicilor

2 6 20 4 # orele de început pentru cei 4 medici

3 8 23 7 # orele de sfârșit pentru cei 4 medici

Soluție:

3 # număr maxim medici

1 4 3 # indicii medicilor selectați

**Modalitate de rezolvare**

Fie mulțimea  $M = \{1, 2, \dots, n\}$  mulțimea medicilor. Corespunzător acestora se cunosc cele  $n$  intervale  $[s_1, f_1], [s_2, f_2], \dots, [s_n, f_n]$  ce conțin ora de început și ora de final pentru fiecare medic.

Dacă un medic  $i$  va fi selectat, cabinetul va fi ocupat în intervalul  $[s_i, f_i]$ .

Doi medici  $i$  și  $j$  au orele compatibile dacă  $[s_i, f_i] \cap [s_j, f_j] = \emptyset$ . Deci, trebuie să determinăm o submulțime maximă  $S \subseteq M$  de medici cu orele compatibile două câte două.

**PASUL1:** schimbăm ordinea medicilor obținând  $M = \{i_1, i_2, \dots, i_n\}$ , astfel încât  $f_{i_1} \leq f_{i_2} \leq \dots \leq f_{i_n}$  (ordonăm medicii în  $M$  după timpul de final, capătul din dreapta al intervalului).

**PASUL 2:** selectăm un medic astfel:

selectie-greedy(sel,s,f):

sel =  $\{i_1\}$  # se selectează primul medic

ultim =  $i_1$  # el este în același timp ultimul medic selectat

pentru  $k = 2, n$  execută: # se caută următorul medic din  $M$  care are orarul compatibil cu  $i_1$

dacă  $s_{i_k} > f_{\text{ultim}}$  atunci

sel = sel  $\cup \{i_k\}$  # dacă este compatibil se adaugă la soluție

ultim =  $i_k$  # cel adăugat devine ultimul

### Demonstrație corectitudine

Se consideră că s-a ordonat mulțimea medicilor după timpul final al fiecărui medic:

$M = \{i_1, i_2, \dots, i_n\}$ , prin urmare  $f_{i_1} \leq f_{i_2} \leq f_{i_3} \leq \dots \leq f_{i_n}$ . (1)

Astfel, primul medic din  $M$  (cel cu indicele  $i_1$  va avea valoarea  $f_{i_1}$  minimă dintre toate valorile  $f_{i_k}$ ,  $k = 1, 2, \dots, n$ .

Demonstrăm că există întotdeauna o soluție optimă care îl conține pe  $i_1$ .

Fie  $A \subseteq M$  o soluție optimă a problemei, unde  $A = \{j_1, j_2, \dots, j_k\}$ .

Presupunem că această soluție nu îl conține pe  $i_1$ . Atunci, are loc relația  $f_{j_1} \leq s_{i_2}$ . (2)

Fie  $A' = \{i_1, j_2, \dots, j_k\}$ .

Din (1) și (2)  $\Rightarrow f_{i_1} \leq s_{i_2}$ , deci medicul  $i_1$  va avea orarul compatibil cu  $j_2, j_3, \dots, j_k$ . Deci toți medicii din  $A'$  au orare compatibile reciproc. Cum  $|A'| = |A| \Rightarrow A'$  este o soluție optimă a problemei.

Am demonstrat că există o soluție optimă a problemei care conține medicul  $i_1$ . În continuare în soluție vor putea fi selectați în soluția optimă doar acei medici care au orele compatibile cu  $i_1$ .

Dacă  $A$  este o soluție optimă pentru  $M$ , vom demonstra că  $A'' = A - \{i_1\}$  construită cu algoritmul nostru este o soluție optimă pentru  $M' = M - \{j \mid s_j \leq f_{i_1}\}$ .

Presupunem că  $A''$  nu este soluția optimă. Deci există o soluție  $B$ , astfel încât  $|B| > |A''|$ .

Dar atunci,  $|B \cup \{i_1\}| > |A|$ . Am ajuns la o contradicție cu presupunerea că  $A$  este o soluție optimă.

### Complexitate

Programul presupune citirea datelor din fișier, sortarea, parcurgerea șirului pentru construirea mulțimii  $A$ , deci  $O(n) + O(n \lg n) = O(n \lg n)$





**Exemplul 3 (Cutii)**

Al Bundy este în dificultate! Are  $n+1$  cutii de pantofi și  $n$  perechi de pantofi identificate prin  $1, 2, \dots, n$  ( $n$  perechi sunt așezate în  $n$  cutii, iar o cutie este liberă). Din păcate pantofii nu se află la locurile lor. Până să vină Gary (șefa lui), Bundy trebuie să potrivească pantofii în cutii. Pentru că Gary poate să apară în orice moment, aranjarea pantofilor trebuie făcută rapid și în așa fel încât să nu trezească bănuiele; prin urmare, dacă scoate o pereche de pantofi dintr-o cutie, trebuie să o pună imediat în cutia liberă. Ajuțați-l pe Al Bundy să aranjeze pantofii la locurile lor printr-un număr minim de mutări.

Exemplu:

4 # numărul  $n$  de cutii

3 4 1 0 2 # dispunerea inițială a perechilor de pantofi în cutii ( $n+1$  numere distincte, 0 – cutia goală)

4 0 2 1 3 # dispunerea corectă a perechilor de pantofi (0-poziția căreia îi corespunde cutia goală)

Soluție:

4 # numărul minim de mutări

**Modalitate de rezolvare**

Există două cazuri posibile:

1. Cutia în care trebuie pusă perechea  $p$  este goală
2. Cutia perechii  $p$  este ocupată de altă pereche  $q$

În cazul 1, vom muta perechea  $p$  în cutia ei, fără a mai face alte mutări. (o singură mutare)

În cazul 2, eliberăm mai întâi cutia ocupată de  $q$  și doar pe urmă vom putea muta perechea  $p$  în cutia ei, acum eliberată. (două mutări).

Deci, trebuie să identificăm în rezolvarea problemei cazul 1 cât timp este posibil. În celelalte cazuri, vom muta perechea de pantofi în cutia potrivită prin două mutări (cazul 2).

**Pseudocod pentru determinarea numărului de mutări**

ci – configurația inițială a perechilor de pantofi în cutii

cf – configurația corectă a perechilor de pantofi în cutii

cigol – indicele cutiei goale în configurația inițială

cfgol – indicele cutiei goale în configurația finală

cauta( $p$ ) – funcție ce returnează numărul cutiei care conține perechea  $p$



cazul 1 apare dacă indicele cutiei goale în configurația inițială este diferit de indicele cutiei goale în configurația corectă

numar\_mutari(n, ci, cf):

repetă

schimb = fals # nu s-a realizat niciun schimb de cutii

cât timp  $cigol \neq cfgol$  execută: # se rezolvă cazul 1

ci[cigol] = cf[cigol] # mutăm în cutia goală perechea nepotrivită

cigol = cauta(cf[cigol])

ci[cigol] = 0 # se eliberează o nouă cutie

mutari = mutari + 1 # s-a efectuat doar o mutare

schimb = adevarat # s-a efectuat un schimb

sfârșit cât timp

cât timp ( $ci[i] = cf[i]$ ) și ( $i \leq n + 1$ ) execută: # se caută o cutie nearanjată

i = i + 1

sfârșit cât timp

dacă  $i \leq n$  atunci # se rezolvă cazul 2

ci[cigol] = ci[i] # se golește cutia nearanjată și se aduce perechea potrivită

ci[i] = cf[i]

cigol = cauta(ci[i])

i = i + 1

mutari = mutari + 2 # s-au făcut două mutări

schimb = adevarat # s-a efectuat o schimbare

până când nu schimb # până când nu a mai fost necesară nicio schimbare

#### IV.Exerciții

1. Spiridușii lui Moș Crăciun au fiecare asociat câte un cod format din litere și cifre. Până în luna noiembrie spiridușii au avut la dispoziție un fișier în care să completeze ce jucării pot face până

la Crăciun și câte bucăți. O linie din acest fișier conține codul spiridușului, numărul de bucăți (număr natural) și numele jucăriei (numele este format din cuvinte separate prin câte un spațiu). Un spiriduș poate adăuga de mai multe ori o linie în fișier, chiar și cu aceeași jucărie, dacă se hotărăște că poate face mai multe. Un exemplu de fișier este: S1 1 papusa S2 1 papusa S3 1 masinuta S1 10 trenulet S2 1 papusa S2 2 masinuta S1 10 ponei S3 15 ponei

a) Memorați datele din fișier astfel încât Moș Crăciun să poată afla cât mai repede informațiile cerute la punctele următoare.

b) Dat codul unui spiriduș, care sunt jucăriile pe care le poate face și ce cantitate din fiecare? Pentru aceasta scrieți o funcție despre\_spiridus cu 2 parametri: în primul parametru se transmite structura în care s-au memorat datele la punctul a) iar al doilea este codul unui spiriduș. Funcția returnează o lista cu elementele tupluri de 2 elemente – primul fiind numele jucăriei, iar al doilea cantitatea – ordonată descrescător după cantitate și, în caz de egalitate, crescător după nume. Apelați funcția pentru codul S1 și afișați lista returnată de funcție. Pentru datele din fișierul exemplu se va afișa [('ponei', 10), ('trenulet', 10), ('papusa', 1)]

c) Care este mulțimea jucăriilor pe care spiridușii le pot produce? Pentru aceasta scrieți o funcție jucarii care primește ca parametru structura în care s-au memorat datele la punctul a) și returnează o mulțime cu numele jucăriilor care pot fi produse de spiriduși. Apelați funcția și afișați pe ecran elementele mulțimii returnate (pe o linie, separate prin virgula). Pentru datele din fișierul exemplu o posibilă ieșire este (jucăriile se pot afișa în orice ordine):  
ponei,trenulet,masinuta,papusa

d) Care este lista spiridușilor harnici: ordonați descrescător după numărul de jucării diferite pe care le pot face și, în caz de egalitate, descrescător după cantitatea de jucării pe care o vor produce și, în caz de egalitate, crescător după cod? Pentru aceasta scrieți o funcție spiridusi care primește ca parametru structura în care s-au memorat datele și returnează o lista cu elementele tupluri de 3 elemente – primul fiind codul spiridușului, al doilea numărul de jucării diferite pe care le poate produce, iar al treilea numărul total de bucăți de jucării care le poate produce spiridușul – ordonată după criteriile cerute de Moș Crăciun (precizate anterior). Apelați funcția și afișați pe ecran elementele listei obținute, fiecare tuplu din listă fiind afișat pe o linie separată. Pentru datele din fișierul exemplu se va afișa ('S1', 3, 21) ('S3', 2, 16) ('S2', 2, 4)

e) În caz că un spiriduș nu mai poate produce un tip de jucărie, să se actualizeze informațiile asociate spiridușului. Pentru aceasta scrieți o funcție actualizare care primește 3 parametri (în aceasta ordine): structura în care s-au memorat datele, codul spiridușului și numele jucăriei. Dacă spiridușul dat ca parametru produce cel puțin două tipuri de jucării, funcția va șterge din structura de date informațiile despre jucăria cu numele dat ca parametru atașate spiridușului cu codul dat ca parametru și va returna True. Altfel funcția va returna False . Să se apeleze funcția pentru spiridușul cu cod S1 și jucăria trenulet, apoi să se afișeze informațiile rămase despre S1 folosind funcția despre\_spiridus de la punctul b. Pentru datele din fișierul exemplu se va afișa [('ponei', 10), ('papusa', 1)] ( problemă preluată recapitulare pentru test laborator, seria 24, an universitar 2020-2021, prof.Marinescu-Ghemeci Ruxandra)