

# Funcții în Python

**Declarea unei funcții** se face folosind:

- cuvântul cheie **def**
- urmat de un nume prin care se poate identifica funcția în mod unic: se pot folosi caracterele alfa-numerice și caracterul "\_". Numele funcției nu poate începe cu o cifră
- o pereche de paranteze rotunde în care se pot enumera, după caz, parametri funcției, urmate de caracterul ":"
- tot blocul de instrucțiuni al funcției trebuie aliniat cu un tab (4 space-uri) la dreapta
- blocul poate începe cu un comentariu care să documenteze funcția. Acest comentariu poate fi accesat și afișat mai târziu
- setul de instrucțiuni al funcției
- opțional, cuvântul cheie **return**, urmat de o valoare sau expresie
- [Exemplul 5.1](#)

**Lista de parametri** a unei funcții:

- o funcție poate avea 0 sau mai mulți parametri.
- unii dintre aceștia pot avea valori implicite, iar apelul unor astfel de funcții se poate face fără a mai trimite o valoare pentru acel parametru.
- parametrii cu valori implicite pot fi doar ultimii din lista de parametri
- [Exemplul 5.2](#)

**Variabile locale vs variabile globale:**

- Aria de vizibilitate a unei variabile este blocul de instrucțiuni în care aceasta este inițializată. [Exemplul 5.3](#)
- O variabilă globală poate fi accesată în interiorul unei funcții, dar nu i se poate schimba referința (nu i se poate atribui un nou obiect/schimba id-ul) decât dacă ea este declarată mai întâi folosind cuvântul cheie **global**. [Exemplul 5.4](#)

Modul în care sunt trimiși parametri este asemănător (la o primă vedere) cu transmiterea parametrilor prin *valoare* din C++. Pe scurt, orice atribuire a unei valori pentru o variabilă transmisă ca parametru nu se va "vedea" în afara funcției. Asta se întâmplă deoarece la o atribuire se crează un nou obiect, cu un nou id, care are aria de vizibilitate doar blocul funcției. [Exemplul 5.5](#)

Putem crea funcții cu **număr nedeterminat de parametri**, declarând funcția ca având în loc de parametri enumerați o colecție de elemente "despachetată", folosind operatorul \*.

[Exemplul 5.6](#)

Alți parametri pot fi puși înaintea colecției în lista de parametri, sau după aceasta, dar în acest caz parametri trebuie apelați cu nume cu tot. [Exemplul 5.6](#)

**Funcții generator:** Cuvântul cheie **yield** se aseamănă cu *return*, în sensul că întoarce o valoare, dar spre deosebire de *return*, nu oprește funcția din parcurgere. Elementele

generate printr- astfel de procedură sunt utile mai ales că sunt produse pas cu pas, fără a fi reținute în o colecție, dar totuși sunt iterabile - pot fi folosite în un ciclu for. Deși ea însăși nu este un generator, comportamentul unui generator este asemănător funcției *range(n)*.

**yield** și **return** pot fi folosiți concomitent în aceeași funcție, dar valoarea obținută prin return nu se va număra printre valorile *generate* de către funcție. Altfel spus, există o diferență între valori generate și valoare returnată.

[Exemplul 5.7](#) - generarea șirului lui Fibonacci.