

CURS 2

Instrucțiuni

Pentru a putea controla fluxul unui program (ordinea în care se vor executa operațiile dorite), majoritatea limbajelor de programare folosesc *instrucțiuni de control*. Aceste instrucțiuni pot fi, de exemplu, *instrucțiuni de decizie* (cu ajutorul cărora se stabilește dacă o anumită operație se efectuează sau nu în funcție de o anumită condiție), *instrucțiuni repetitive* (cu ajutorul cărora se efectuează de mai multe ori o anumită operație) etc.

În limbajul Python nu există delimitatori pentru *blocurile de instrucțiuni* (cum sunt acoladele în limbajele C/C++), ci gruparea mai multor instrucțiuni se realizează prin indentarea lor în raport de instrucțiunea căreia i se subordonează.

În limbajul Python sunt definite următoarele instrucțiuni de control:

1. *instrucțiunea de atribuire*

- Spre deosebire de limbajele C/C++, atribuirea nu este un operator, ci este o instrucțiune!
- Instrucțiunea de atribuire poate avea următoarele forme:
 - *atribuire simplă* ($x = 100$);
 - *atribuire multiplă* ($x = y = 100$);
 - *atribuire compusă* ($x, y, z = 100, 200, 300$).
- Două variabile se pot interschimba prin atribuirea compusă $x, y = y, x$!
- O atribuire de forma $x = x \text{ operator expresie}$ poate fi scrisă prescurtat sub forma $x \text{ operator} = \text{expresie}$, unde operator este un operator aritmetic sau pe biți binar. De exemplu, instrucțiunea $x = x + y * 10$ poate fi scrisă prescurtat sub forma $x += y * 10$.
- În limbajul Python nu sunt definiți operatorii $++/--$ din limbajele C/C++!

2. *instrucțiunea de decizie / alternativă if*

- Instrucțiunea de decizie este utilizată pentru a executa o instrucțiune (sau un bloc de instrucțiuni) doar în cazul în care o expresie logică este adevărată:

```
if expresie_logică:
    instrucțiune
```

Exemplu (maximul dintre două numere):

```
a = int(input("a = "))
b = int(input("b = "))
maxim = a
if b > maxim:
    maxim = b
print("Maximul dintre", a, "si", b, "este", maxim)
```

- Instrucțiunea alternativă este utilizată pentru a alege executarea unei singure instrucțiuni (sau a unui bloc de instrucțiuni) dintre două posibile, în funcție de valoarea de adevăr a unei expresii logice:

```
if expresie_logică:
    instrucțiune_1
else:
    instrucțiune_2
```

Exemplu (maximul dintre două numere):

```
a = int(input("a = "))
b = int(input("b = "))
if a > b:
    maxim = a
else:
    maxim = b
print("Maximul dintre", a, "si", b, "este", maxim)
```

- Instrucțiunile alternative imbricate se pot scrie mai concis folosind instrucțiunea `elif`, așa cum se poate observa din exemplul următor:

<pre>a = int(input("a = ")) if a < 0: print("Strict negativ") else: if a == 0: print("Zero") else: print("Strict pozitiv")</pre>	<pre>a = int(input("a = ")) if a < 0: print("Strict negativ") elif a == 0: print("Zero") else: print("Strict pozitiv")</pre>
---	---

- Până la varianta 3.10 a limbajului Python nu era definită o instrucțiune alternativă multiplă, cum este, de exemplu, instrucțiunea `switch` din limbajele C/C++! În varianta 3.10 a fost introdusă o astfel de instrucțiune, denumită `match` ([The Match-Case In Python 3.10 Is Not That Simple | by Christopher Tao | Towards Data Science](#)).

3. instrucțiunea repetitivă `while`

- Instrucțiunea `while` este o instrucțiune repetitivă cu test inițial, fiind utilizată pentru a executa o instrucțiune (sau un bloc de instrucțiuni) cât timp o expresie logică este adevărată:

```
while expresie_logică:
    instrucțiune
```

Exemplu (suma cifrelor unui număr natural):

```
n = int(input("n = "))
aux = n
sc = 0
while aux != 0:
    sc = sc + aux % 10
```

```

    aux = aux // 10
print("Suma cifrelor numarului", n, "este", sc)

```

- În limbajul Python nu este definită o instrucțiune repetitivă cu test final, cum este, de exemplu, instrucțiunea `do...while` din limbajele C/C++!

4. *instrucțiunea repetitivă for*

- Instrucțiunea `for` este utilizată pentru a accesa, pe rând, fiecare element dintr-o secvență (de exemplu, un șir de caractere, o listă etc.), elementele fiind considerate în ordinea în care apar în secvență:

for *variabilă în secvență:*
 instrucțiune

Exemple:

<pre> sir = "test" for c in sir: print(c, end=" ") #Se va afișa: t e s t </pre>	<pre> lista = [1, 2, 3] for x in lista: print(x, end=" ") #Se va afișa: 1 2 3 </pre>
--	---

- Pentru a genera secvențe numerice de numere întregi asemănătoare unor progresii aritmetice se poate utiliza funcția `range([min], max, [pas])`, care va genera, pe rând, numerele întregi cuprinse între valorile `min` (inclusiv) și `max` (exclusiv!!!) cu rația `pas`. Parametrii scriși între paranteze drepte sunt opționali, iar parametrul opțional `pas` se poate specifica doar dacă se specifică și parametrul opțional `min`. Dacă pentru parametrul `min` nu se specifică nicio valoare, atunci el va fi considerat în mod implicit ca fiind 0.

Exemple:

```

range(6)           => 0, 1, 2, 3, 4, 5
range(2, 6)        => 2, 3, 4, 5
range(2, 11, 3)     => 2, 5, 8
range(2, 12, 3)     => 2, 5, 8, 11
range(7, 2)         => secvență vidă (deoarece 7 > 2)
range(7, 2, -1)     => 7, 6, 5, 4, 3

```

5. *instrucțiunea continue*

- Instrucțiunea `continue` este utilizată în cadrul unei instrucțiuni repetitive pentru a termina forțat iterația curentă (dar nu și instrucțiunea repetitivă!), continuându-se direct cu următoarea iterație.

Exemplu:

```

for i in range(1, 11):
    if i%2 == 0:
        continue
    print(i, end=" ")

#Se va afișa: 1 3 5 7 9

```

6. instrucțiunea break

- Instrucțiunea `break` este utilizată în cadrul unei instrucțiuni repetitive pentru a termina forțat executarea instrucțiunii respective.

Exemplu: Se citește un șir de numere care se termină cu valoarea 0 (care se consideră că nu face parte din șir, ci este doar un marcaj al sfârșitului său). Să se afișeze suma numerelor citite.

```
s = 0
while True:
    x = int(input("x = "))
    if x == 0:
        break
    s += x
print("Suma numerelor citite: ", s)
```

Atenție, în acest program instrucțiunea `s += x` ar fi putut fi scrisă și înaintea instrucțiunii `if` fără a-i afecta corectitudinea! Totuși, în alte cazuri (de exemplu, dacă s-ar fi cerut produsul numerelor citite), acest lucru ar fi dus la afișarea unui rezultat eronat!

7. instrucțiunea else

- Instrucțiunea `else` poate fi adăugată la sfârșitul unei instrucțiuni repetitive, instrucțiunile subordonate ei fiind executate doar în cazul în care instrucțiunea repetitivă se termină natural (condiția dintr-o instrucțiune `while` devine falsă sau o instrucțiune `for` a parcurs toate elementele unei secvențe), ci nu din cauza întreruperii sale forțate (utilizând o instrucțiune `break`).

Exemple:

- a) Se citește un șir format din n numere întregi. Să se verifice dacă toate numerele citite au fost pozitive sau nu.

```
n = int(input("n = "))
for i in range(n):
    x = int(input("x = "))
    if x < 0:
        print("A fost citit un număr negativ!")
        break
else:
    print("Toate numerele citite au fost pozitive!")
```

- b) Să se afișeze cel mai mic număr prim cuprins între două numere naturale a și b sau un mesaj corespunzător în cazul în care nu există niciun număr prim cuprins între a și b .

```
a = int(input("a = "))
b = int(input("b = "))
```

```

for x in range(a, b+1):
    for d in range(2, x//2+1):
        if x % d == 0:
            break
    else:
        #instrucțiunea for d in ... s-a terminat natural,
        #deci numărul x nu are divizori proprii,
        #ceea ce înseamnă că x este un număr prim
        print("Cel mai mic numar prim cuprins între", a,
              "și", b, "este", x)

        #numărul x este cel mai mic număr prim cuprins între
        #a și b, deci nu are rost să mai continuăm căutarea
        #altuia și oprim forțat instrucțiunea for x in ...
        break
else:
    #instrucțiunea for x in ... s-a terminat natural, deci
    # nu a fost găsit niciun număr prim cuprins între a și b
    print("Nu exista niciun numar prim cuprins între", a,
          "și", b)

```

8. instrucțiunea pass

- Instrucțiunea **pass** este o instrucțiune care nu are niciun efect în program (este similară unei instrucțiuni vide). Această instrucțiune se utilizează în cazurile în care sintactic ar fi necesară o instrucțiune vidă, deoarece în limbajul Python aceasta nu este definită.

Exemplu:

```

varsta = 10
if varsta <= 18:
    print("Junior")
elif varsta < 60:
    #nu prelucrăm informațiile despre persoanele
    #cu vârste cuprinse între 19 și 59 de ani
    pass
else:
    print("Senior")

```