Elemente de bază ale limbajului Python

Diferențe față de C/C++

- Variabilele în Python nu au tip de date static (nu se declară tipul lor, o variabilă este “declarantă” **când i se atribuie prima dată o valoare**).
- Tipul unei variabile se poate schimba pe parcursul execuției programului .

```
x = 7 #variabila x este "declarata”
print(x)
x = "abc"
print(x)
```

Diferențe față de C/C++

- **Nu sunt necesari delimitatori de blocuri** de tip {} sau begin/end etc, este **obligatorie indentarea** blocurilor de cod (și **suficientă** pentru delimitarea acestora)
- Nu este nevoie sa punem ; la finalul unei linii (dacă nu mai urmează alte linii de cod pe aceeași linie)

```
i = 1
while i<10:
      print(i)
      i = i + 1    #nu i++
print("gata afisarea")
```

Elemente de bază ale limbajului

1. Afișarea unei variabile + tipul acesteia (al valorii asignate)

Elemente de bază ale limbajului

```
print("mesaj")
x = 1
print("x=",x, type(x), id(x)) #pe acelasi rand cu spatiu, apoi linie noua
print("x=" + str(x))
x = "Sir"
print("x=", x, type(x), id(x)) #nu are acelasi id
y = 2
print(x, end=' ')#pentru a nu trece la linie noua modific parametrul end
print(y)
print(x, y, sep='*')
```

Elemente de bază ale limbajului

2. Citirea de la tastatură + funcții de conversie

- funcția **`input`**
  - parametru (opțional) mesajul care se va afișa pe ecran
  - returnează **șirul de caractere** introdus **până la sfârșitul de linie** (**de tip** str**, este necesară conversia** dacă vrem alt tip de date)

Elemente de bază ale limbajului

```
#citire-necesara conversie
x = input("x=")
print("x=", x, type(x))
x = int(input("intreg=")) #ValueError daca introducem gresit
print("x=", x, type(x))
x = float(input("real="))
print("x=", x, type(x))
x = complex(input("complex="))
print("x=", x, type(x))
```

Elemente de bază ale limbajului

3. Erori

```
i = 1  #i="ab", i=-1
print(i)
if i>0: #daca i nu este numar?
    print("ok")
else:
    print(i + " este negativ") #daca i nu este sir?
    print(y) #da eroare daca i=1?
```

Variabile

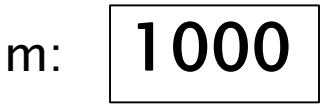
Variabile

- În C/C++ o variabilă se declară și are: tip, adresa, valoare
- În Python variabilele sunt **referințe spre obiecte (nume date obiectelor)**; orice valoare este un obiect
- Un obiect **`ob`** are asociat:
  - un număr de identificare: **`id(ob)`**
  - un tip de date: **`type(ob)`**
  - o valoare – poate fi convertită la șir de caractere **`str(ob)`**

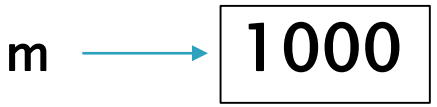
Variabila; obiect; alocare; atribuire

```
m = 1000
```

C



Python

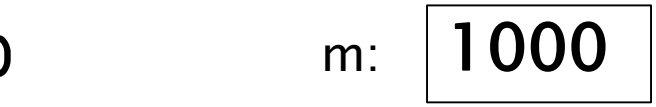


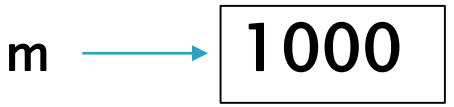
Variabila; obiect; alocare; atribuire

C

Python

```
m = 1000
```






```
m = m+1
```

Variabila; obiect; alocare; atribuire

C

```
m = 1000
```

m: 1001

```
m = m+1
```

m: 1001

Python

```
m = 1000
```

m → 1000

```
m = m+1
```

m → 1001

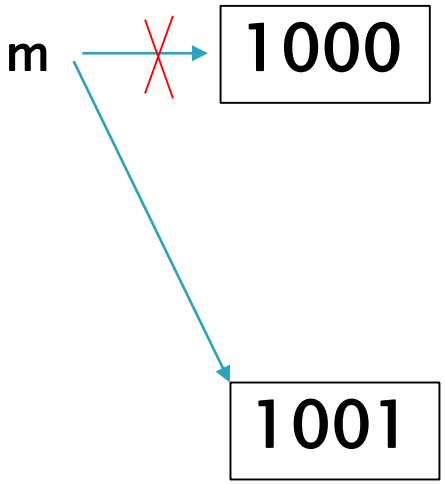
Variabila; obiect; alocare; atribuire

C

Python

```
m = 1000
```

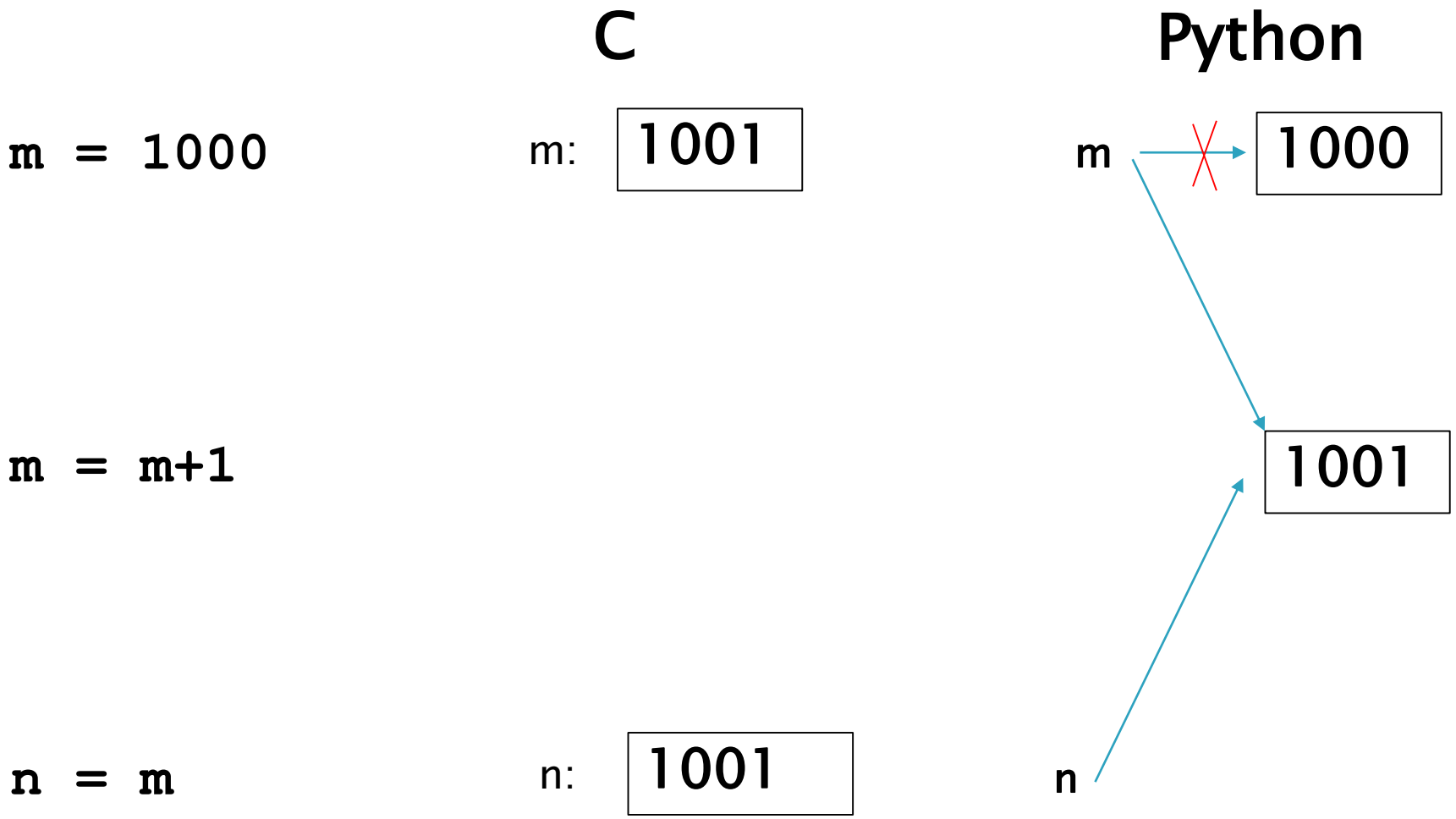
m: 1001




```
m = m+1
```

```
n = m
```

Variabila; obiect; alocare; atribuire



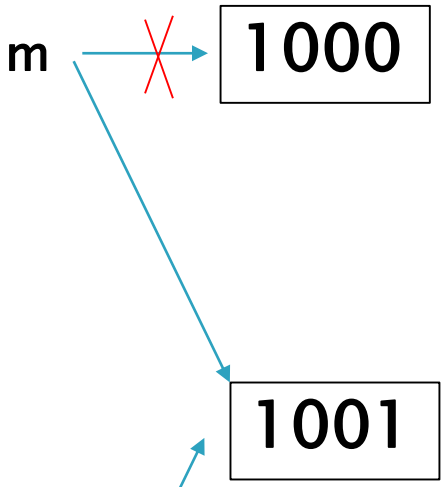
Variabila; obiect; alocare; atribuire

C

Python

```
m = 1000
```

m: 1001




```
m = m+1
```

```
n = m
```

n: 1001

```
n = n+1 ???
```

Variabile

- Tipul unei variabile se stabilește prin inițializare și se poate schimba prin atribuiri de valori de alt tip
- Numele unei variabile – identificatori
- Recomandare nume:

```
litere_mici_separate_prin_underscore
```

Variabile

- optimizare: numerele întregi din intervalul [-5,256] sunt prealocate (în cache) – **toate obiectele care au o astfel de valoare sunt identice (au același id)**

- variabile cu aceeași valoare **pot avea** același id (dacă este o valoare prealocată, atunci sigur da)

Variabile

- Exemplu

```
x = 1
y = 0
y = y + 1
z = x
print(x,y,z,x*x)
print(id(x),id(y),id(z),id(x*x))
#toate expresiile cu valoare 1 au acelasi id
```

Variabile

- Exemplu

```
x = 1000
y = 999
y = y+1
z = x
print(x,y,z,10*x//10)
print(id(x),id(y),id(z),id(10*x//10))
```

Variabile

- **`del x`** – șterge o variabilă din memorie

```
m = input()
del m
print(m)#eroare
```

- Garbage collector – șterge obiecte către care nu mai sunt referințe

Tipuri de date

Tipuri de date

- **`int`**
  - numere întregi cu oricât de multe cifre (limita dată doar de performanța sistemului pe care se rulează)

Tipuri de date

- **`int`**
  - numere întregi cu oricât de multe cifre (limita dată doar de performanța sistemului pe care se rulează)
  - memorate ca vectori de "cifre" din reprezentarea în baza 2^{30} (cu cifre de la 0 la $2^{30}-1 = 1073741823$)

Exemplu: Reprezentarea pentru 23425464654983427349 8:

ob_size	3		
ob_digit	462328538	197050268	203

deoarece $23425464654983427349 8 = 462328538 \times (2^{30})^0 + 197050268 \times (2^{30})^1 + 203 \times (2^{30})^2$

Tipuri de date

- **`int`**
  - constante în baza 10 (implicit), dar și in bazele 2 (prefix 0b,0B), 8 (prefix 0o, 0O), 16 (prefix 0x,0X):

```
print(0b101, 0o10, 0xAb)
```

Tipuri de date

- **`int`**
  - constante în baza 10 (implicit), dar și in bazele 2 (prefix 0b,0B), 8 (prefix 0o, 0O), 16 (prefix 0x,0X):

```
print(0b101, 0o10, 0xAb)
```

  - putem folosi **`int(sir)`** pentru creare/conversie (exista si varianta **`int(sir, base=baza)`**)

```
print(int(9.7) + int("101", base = 2) + int("101",2))
```

Tipuri de date

- **`float`**
  - Constante: `3.5`, `1e-2` (notație științifică)
  - float([x]):

```
float("inf"); float("infinity"); float("nan")
```

Tipuri de date

- **`float`**
  - IEEE-754 double precision
  - **operațiile aritmetice cu tipul de date *float* nu au precizie absolută:**

    NU: `0.1*0.1 == 0.01`

    DA: `abs(0.1*0.1-0.01) < 1e-9`

Tipuri de date

- **`complex`**
  - de forma a + bj **(!!! nu i**, merge și J**)**

Tipuri de date

- **complex**

```
z = complex(-1, 4)
print("Numarul complex:", z)
print("Partea reala:", z.real)
print("Partea imaginara:", z.imag)
print("Conjugatul:", z.conjugate())
print("Modul:", abs(z))
```

Tipuri de date

- **bool**
  - True, False
  - putem folosi `bool()` pentru conversie
  - În context boolean – **conversia oricărei valori la bool**

    Context boolean – condiție if, while; operand pentru operatori logici – conversii

Tipuri de date

- bool
  - Se consideră **`False`**:
    - **`None, False`**
    - **`0, 0.0, 0j, Decimal(0), Fraction(0,1)`**
    - **Colecții și secvențe vide** (+obiecte în care __bool__() returneaza False sau __len__() returnează 0)

```
print(bool(0), bool(-5))
print(bool(""), bool(" "))
print(bool(None), bool([]))
```

Tipuri de date

- NoneType
  - **`None`**

- Nu exista char

```
ord("a")

chr(97)
```

Tipuri de date

Secvențe:

Mutabile (le putem modifica) și imutabile

- liste `list`: a = [3, 1, 4, 7] – mutabile
- tupluri `tuple`: a = (3, 1, 4, 7)
- șiruri de caractere `str`: a = "3147sir"

Tipuri de date

Mulțimi:

- set: a={1, 4, 5}
- frozenset: fa = `frozenset`(a) – nu se poate modifica

Dicționare

Comentarii

- Prefixat de # => comentariu pe o linie
- Pentru mai multe linii – # pe fiecare linie sau delimitatori de șiruri de caractere
  - Încadrat de ' ' ' => pe mai multe linii
  - Încadrat de " " " => docstring – comentariu pe mai multe linii, folosit în mod special pentru documentare