CAPITOLUL II INTRODUCERE ÎN PYTHON

În acest capitol vom învăța:

- Să citim, afișăm și prelucrăm variabile în Python
- Tipurile de date și operatorii specifici fiecărui tip de date
- Să identificăm tipurile de date care trebuie utilizate în funcție de problema concretă

2. 1 Constante și variabile în Python

Pentru procesarea informației într-un calculator se utilizează date. Datele pot fi de două feluri: constante sau variabile.
Următorul program calculează aria unui cerc.

```
PI=3.14159
raza=float(input('Dati raza cercului '))
aria=PI*raza*raza
print('Aria cercului este:', aria)
```

O **variabilă** este o dată care își poate schimba valoarea pe parcursul execuției unui program.

Noțiunea de variabilă din informatică trebuie asociată cu sensul cuvântului în limba română.

În dicționarul explicativ al limbii române, cuvântul **variabilă** este explicat astfel: *Cantități (sau mărimi) variabile = cantități (sau mărimi) susceptibile de a-și schimba valoarea față de altele, care rămân constante*.

Dacă dorim să afișăm temperatura existentă în fiecare oră într-o zi, într-un oraș, vom folosi o variabilă nu o constantă, deoarece pe parcursul unei zile temperatura își schimbă valoarea.

Ex. temperatura=10
temperatura=18

Dacă dorim să reținem scorul unui meci de fotbal avem nevoie de două variabile: goluri_gazde și goluri_oaspeți. Pe parcursul meciului, acestea își vor schimba valoarea.

O constantă este o dată care nu își schimbă valoarea pe parcursul execuției unui program.

Prin convenție, în Python constantele se scriu cu litere mari și sunt așezate la începutul programului.

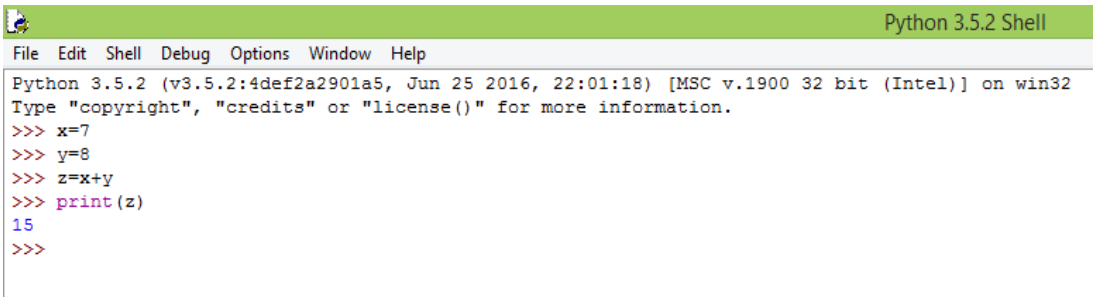
În exemplul prezentat anterior **PI** este o constantă iar raza și aria sunt variabile.

Alte exemple de constante și variabile:

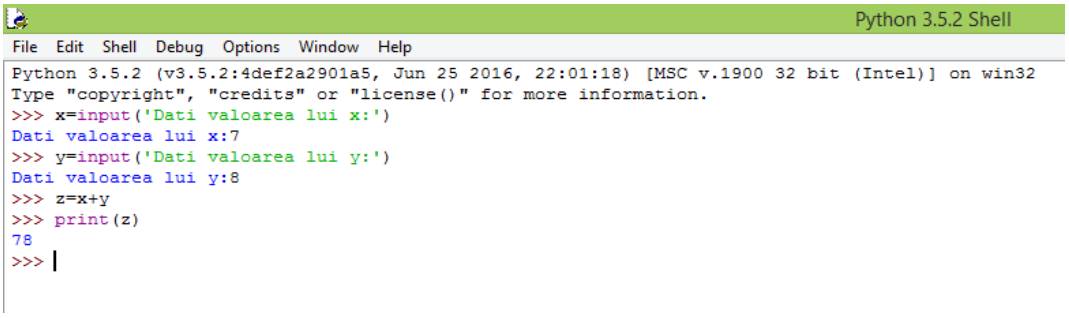
1. Dacă dorim să desenăm dreptunghiuri de dimensiuni diferite, lungimea și lățimea vor fi variabile în timp ce unghiul de 90° va fi o constantă.
2. Dacă dorim să reținem data nașterii și vârsta unei persoane, vom reține într-o constantă data nașterii și într-o variabilă vârsta.

În orice limbaj de programare datele au un nume, o valoare, un tip și o adresă de memorie. În Python, tipul variabilelor nu trebuie declarat înaintea utilizării lor, el este specificat în momentul inițializării sau este detectat automat în cazul în care nu se specifică la inițializare.

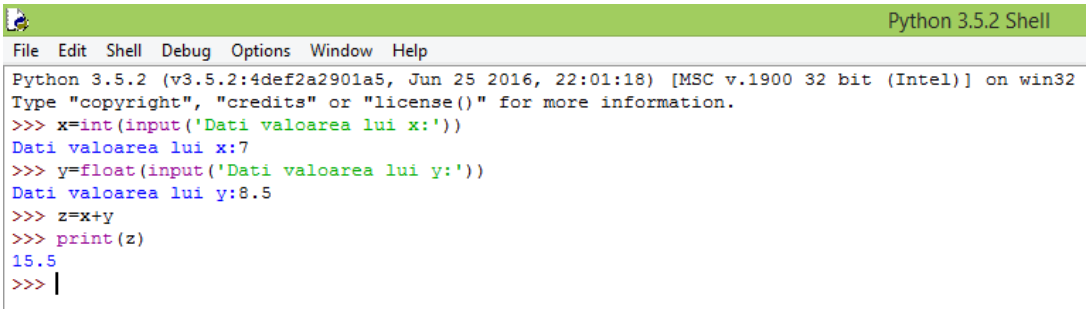
Exemplu:



În cazul în care valorile sunt citite de la tastatură folosind funcția input, tipul implicit de date este șir de caractere. În exemplul de mai jos operatorul + concatenează cele două șiruri de caractere.



Să analizăm exemplul de mai jos.



Se observă că variabila z reține un număr real.

Numele variabilelor și constantelor

Am văzut deja că variabilele și constantele au nume care ne ajută să le folosim în cadrul unui program (raza, aria, PI) și ne spun câte ceva despre rolul lor în cadrul acestuia.

Pentru denumirea constantelor și variabilelor se pot folosi litere, cifre și caracterul _ (underscore).

Primul caracter nu poate fi cifra. De asemenea, nu se pot folosi spațiile între cuvinte și nici caracterele -,/,# și @.

Observații importante:

1. Deoarece Python face distincție între literele mari și mici, numele de variabile **suma** și **Suma** reprezintă două variabile diferite.

2. Cuvintele care reprezintă nume de funcții sau instrucțiuni Python (cum ar fi **input**) se numesc cuvinte rezervate și nu pot fi utilizate ca nume de variabile.

Lista cuvintelor rezervate în Python este următoarea:

```
'False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class',
'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for',
'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not',
'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield'
```

Exemple de nume de variabile și constante corecte: c, c1, raza, aria, pb5_pag2, _p1

Dintre numele de mai sus, unele, deși corecte, nu sunt recomandate. Numele c, c1, _p1 nu ne spun foarte multe despre rolul variabilei sau constantei respective. Este mult mai bine să folosim nume explicite cum ar fi *raza*, *aria* sau *PI*.

Exemple de nume de variabile și constante incorecte: 3km, km/ora, d#d, zece-numere, c@g.

2.2 Tipuri de date în Python

În micile noastre programe de până acum am folosit numere dar nu este greu să ne imaginăm programe care folosesc și alte tipuri de date. De exemplu, un program care caută jocuri pe Internet va folosi șiruri de caractere pentru a reprezenta genul jocului sau denumirea acestuia.

Iată câteva cuvinte despre tipurile de date pe care le putem folosi în programele noastre Python.

2.2.1 Tipul numeric

Python are două tipuri de date pentru a stoca numere: întreg și real.

Numerele întregi sunt numere fără virgulă iar numerele reale sunt numere cu virgulă.

Pentru numerele întregi vom folosi tipul int iar pentru cele reale tipul float.

2.2.2 Tipul șir de caractere

Un șir de caractere este o succesiune de litere, cifre și caractere speciale. În Python, șirurile de caractere sunt plasate între apostrofuri sau ghilimele. **În cazul în care nu specificăm la citirea unei variabile tipul ei, ea este implicit de tip șir de caractere.**

```
>>> sir='Programarea este simpla'
>>> print(sir)
Programarea este simpla
>>>
```

2.2.3 Tipul boolean

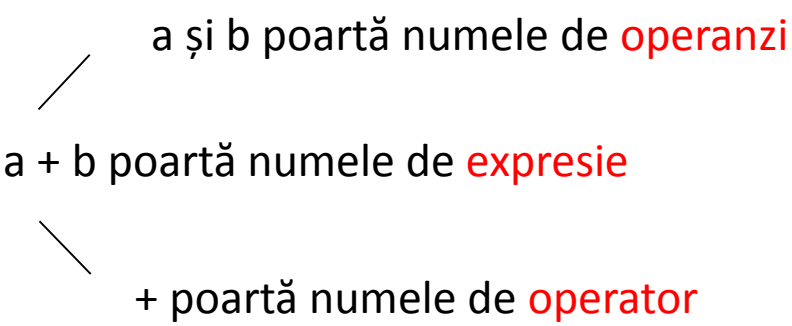
O variabilă sau constantă de tip boolean poate lua două valori True (adevărat) sau False (fals). În ambele cazuri cuvântul începe cu literă mare.

```
>>> a=True
>>> print(a)
True
>>> a=False
>>> print(a)
False
>>>
```

Observatie importantă: În Python, valorile True și False trebuie scrise cu literă mare.

2.3 Operatori și expresii în Python

2.3.1 Expresii



O expresie este alcătuită din operanzi conectați prin operatori. Un operand poate fi o constantă, o variabilă sau o expresie. Operatorii reprezintă operațiile care se execută asupra operanzilor. Operatorii care pot fi utilizați într-o expresie depind de tipul operanzilor (numerici întregi, numerici reali, șiruri de caractere sau logici).

OBSERVAȚIE IMPORTANTĂ

Evaluarea unei expresii aritmetice se face respectând prioritățile operatorilor. Prioritățile operatorilor sunt prezentate într-un alt subcapitol al acestui capitol. Pentru modificarea priorităților se pot folosi parantezele.

Operatorii pe care îi vom folosi în această carte sunt operatorii aritmetici, operatorii relaționali și operatorii logici.

2.3.2 Operatori aritmetici

Operatorii aritmetici pot fi: *unari* – se aplică unei singure date sau *binari* – se aplică pe două date.

Operatorii unari sunt + (plus), - (minus). Aceștia preced data și se folosesc pentru a stabili semnul valorii numerice.

Ex. -12, -14.7, +5

Operatorii binari sunt +, -, *, /, // (împărțire întreagă – div în algoritmi), % (modulo), ** (ridicarea la putere)

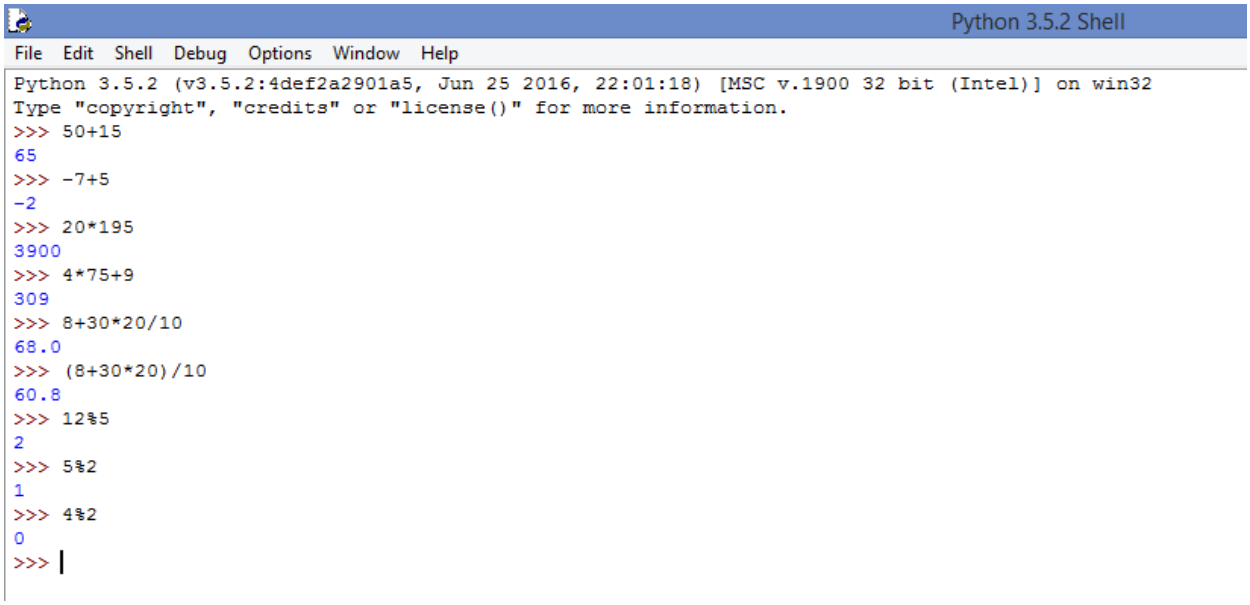
+ (adunare) – adună matematic două numere. Ex. 7+3=10

- (scădere) – scade matematic două numere. Ex. 9-2=7

* (înmulțire) – înmulțește două numere. Ex. 7*2=14

/ (împărțire) – împarte primul număr la al doilea. Ex. 7/2=3.5
// (împărțire întreagă) – calculează câtul împărțirii a două numere. Ex. 7//2=3
% (modulo) – calculează restul împărțirii întregi a două numere **întregi sau reale**.
Ex. 5 % 2=1, 9%2.5=1.5
** (ridicarea la putere) a**b calculează a la puterea b unde a și b pot fi numere **întregi sau reale**.
Ex. 2**3=8, 2.5**3=15.625, 2.5**2.5= 9.882117688026186

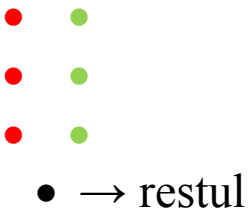
Exemple:



Explicații la operatorul % (modulo)

1)12%5=2

2) 7%3=1



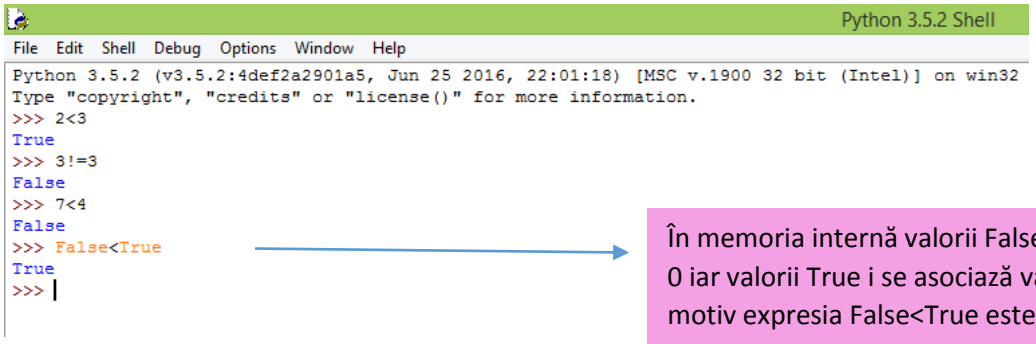
Observație importantă: Operatorii de incrementare și decrementare ++ și -- care există în limbaje precum C++ sau JavaScript, în Python nu există.

2.3.3 Operatori relaționali

Operatorii relaționali în Python se pot aplica asupra datelor de tip numeric, logic sau șir de caractere. Operatorii relaționali în Python sunt:

<	mai mic
<=	mai mic sau egal
>	mai mare
>=	mai mare sau egal
==	Egal
!=	Diferit

Prin aplicarea operatorilor relaționali se returnează o valoare True (adevărat) sau False (fals).



Aplicarea operatorilor relaționali asupra șirurilor de caractere este prezentată în subcapitolul 2.5.

2.3.4 Operatori logici

Python are trei operatori logici: and (conjuncție logică), or (disjuncție logică), not (negație logică). Operatorul not este operator unar iar operatorii and și or sunt operatori binari.

Operatorii logici se aplică asupra datelor de tip logic și produc un rezultat de tip logic (True sau False).

Operatorul logic and

Sintaxa este: expresie1 and expresie2

expresie1	expresie2	expresie1 and expresie2
False	False	False
False	True	False
True	False	False
True	True	True

Pentru a reține mai ușor tabelul de valori corespunzător operatorului and imaginați-vă o ușă cu două lacăte. Putem intra în cameră dacă ambele lacăte sunt deschise. Vom asocia lacătului închis valoarea False și lacătului deschis valoarea True.

lacăt 1	lacăt 2	ușa
închis (False)	închis (False)	închisă (False)
închis (False)	deschis (True)	închisă (False)
deschis (True)	închis (False)	închisă (False)
deschis (True)	deschis (True)	deschisă (True)

Operatorul logic or

Sintaxa este: expresie1 or expresie2

expresie1	expresie2	expresie1 or expresie2
False	False	False
False	True	True
True	False	True
True	True	True

Pentru a reține mai ușor tabelul de valori corespunzător operatorului or, imaginați-vă o cameră cu două uși. Putem intra în cameră dacă una din uși este deschisă. Ca și la operatorul and, vom asocia ușii închise valoarea False și ușii deschise valoarea True.

ușa 1	ușa 2	camera
închisă (False)	închisă (False)	închisă (False)
închisă (False)	deschisă (True)	deschisă (True)
deschisă (True)	închisă (False)	deschisă (True)
deschisă (True)	deschisă (True)	deschisă (True)

Operatorul logic not

Sintaxa este not expresie

not True = False

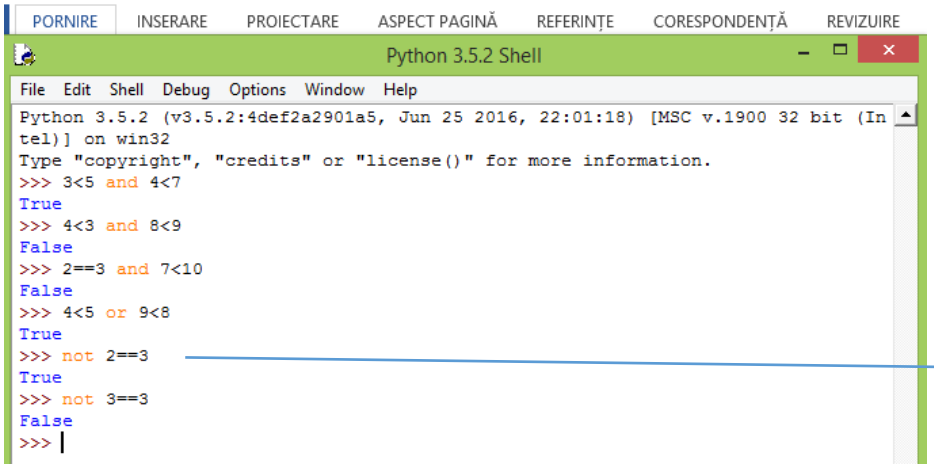
not False = True

LEGILE LUI DE MORGAN

not(A and B)=not A or not B

not(A or B)=not A and not B

Exemple:



Deoarece operatorii relaționali au prioritate mai mare decât operatorul not, nu este nevoie de paranteze.

2.3.5 Prioritățile operatorilor aritmetici, relaționali, logici

Prioritate	Operatori	Simbol	Asociativitate
1	Aritmetici unari	+,-	De la stânga la dreapta
2	Ridicarea la putere	**	De la dreapta la stânga
3	Aritmetici multiplicativi	*, /,//, %	De la stânga la dreapta
4	Aritmetici aditivi	+, -	De la stânga la dreapta
5	Relaționali	<, <=, >, >=	De la stânga la dreapta
6	Relaționali	==, !=	De la stânga la dreapta
7	De atribuire	=, +=, -=, *=, /=, %=, **=	De la dreapta la stânga
8	Negația logică	NOT	De la stânga la dreapta
9	Conjuncție logică	AND	De la stânga la dreapta
10	Disjuncție logică	OR	De la stânga la dreapta

Pentru schimbarea priorității operatorilor, în Python se folosesc paranteze rotunde.

Observație importantă: prioritățile operatorilor în Python sunt diferite de prioritățile operatorilor în C++.

2.4 Inițializarea și prelucrarea variabilelor

2.4.1 Instrucțiunea pentru atribuire

Are sintaxa

variabila=expresie

Efectul: se evaluează expresia iar valoarea ei este atribuită variabilei din membrul stâng.
În cazul operatorilor aritmetici, instrucțiunea de atribuire mai poate să aibă forma
variabila op=expresie

Exemple:

a+=b ⇔ a=a+b

a-=b ⇔ a=a-b

a*=b ⇔ a=a*b

a/=b ⇔ a=a/b

a%=b ⇔ a=a%b

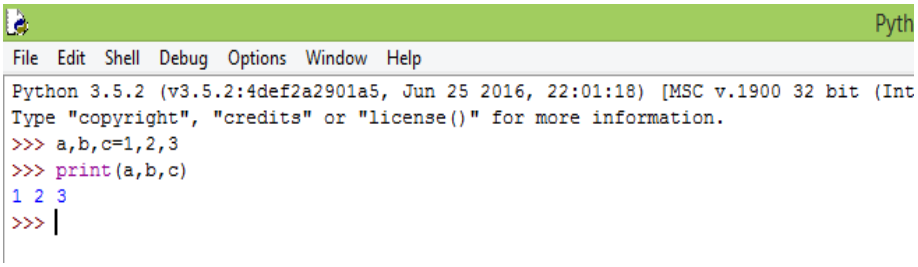
a//=b ⇔ a=a//b

a**=b ⇔a=a**b

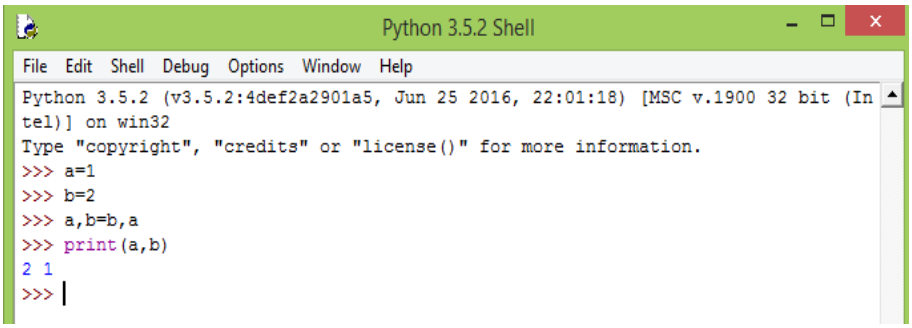
```
>>> a=5
>>> a%=2
>>> print(a)
1
>>>
```

Observații importante

1. Python permite atribuirea de forma **a,b,c=1,2,3**. În urma acestei atribuiri variabila a va lua valoarea 1, variabila b va lua valoarea 2 iar variabila c va lua valoarea 3.



2. Pentru interschimbarea conținutului a două variabile nu este nevoie de o a treia variabila, este suficientă atribuirea **a, b=b, a .**



2.4.2 Funcția pentru afișare

Are sintaxa:

print(variabila)

Efect: se afișează pe ecran conținutul variabilei. Funcția print este utilizată și pentru afișarea textelor pe ecran. În acest caz textul trebuie inclus între ghilimele sau între apostrofuri.

2.4.3 Funcția pentru citire

Are sintaxa:

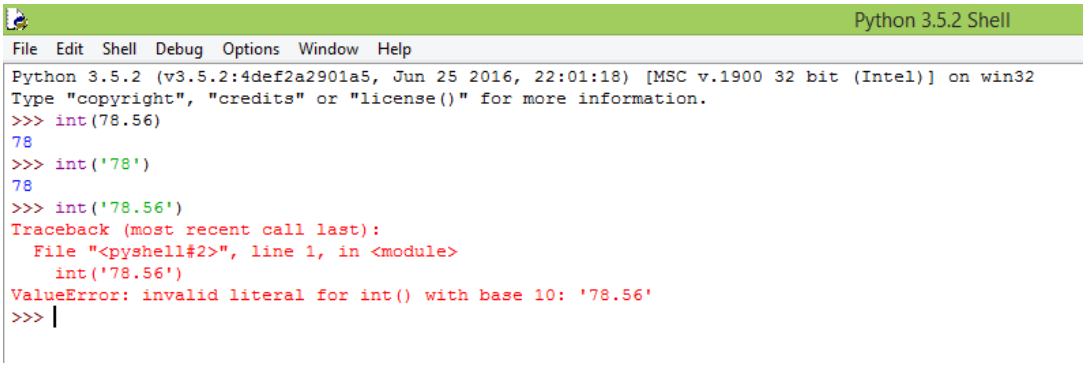
variabila=input(text)

Efect: Se introduce de la tastatura un număr sau un șir de caractere care se memorează în variabilă.

2.4.4 Funcții pentru lucrul cu variabile numerice

1) Funcția int

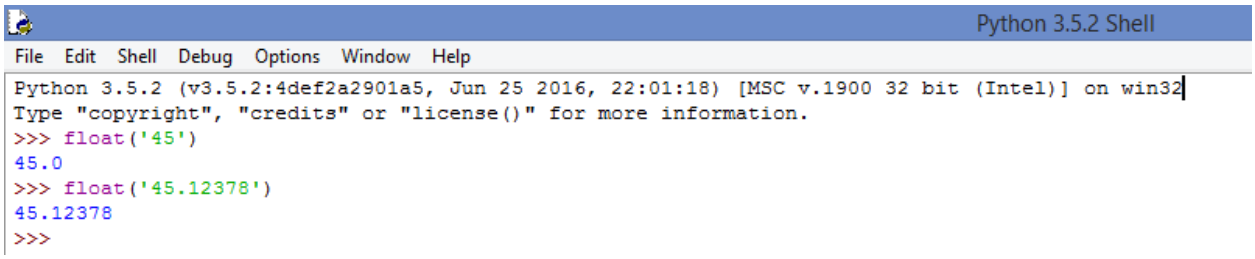
Convertește un număr real la un număr întreg sau un șir de caractere la un număr.



În cazul în care argumentul este un șir de caractere ce conține un număr real, funcția va returna eroare.

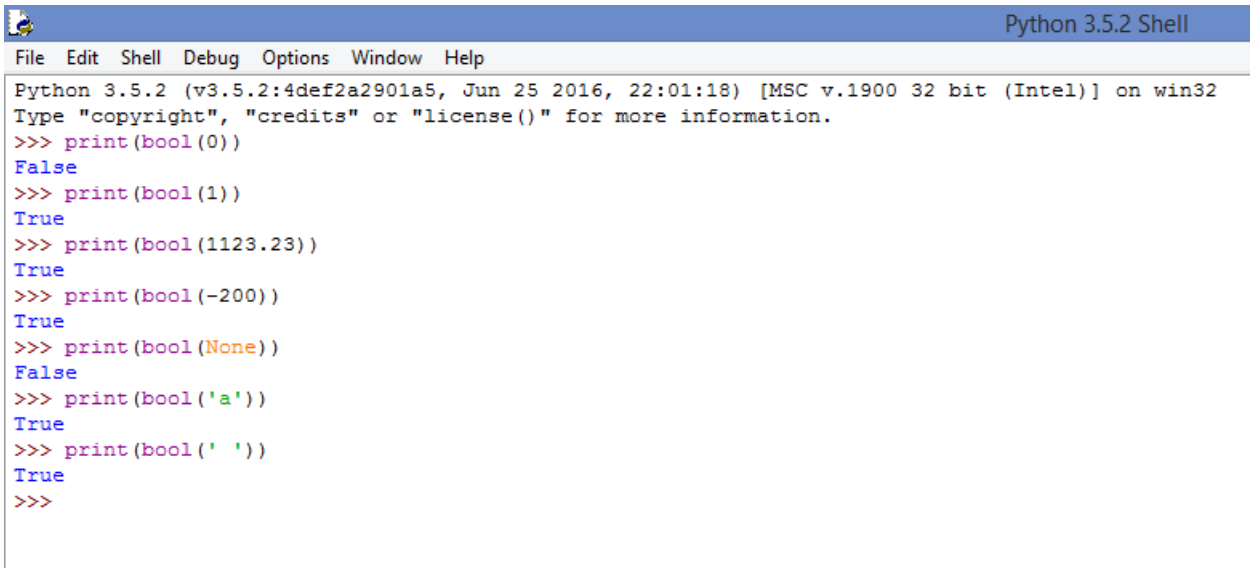
2) Funcția float

Funcția float convertește un șir de caractere la un număr real.
Un număr real este un număr cu virgulă.
Numărul matematic 12,45 devine în Python 12.45.



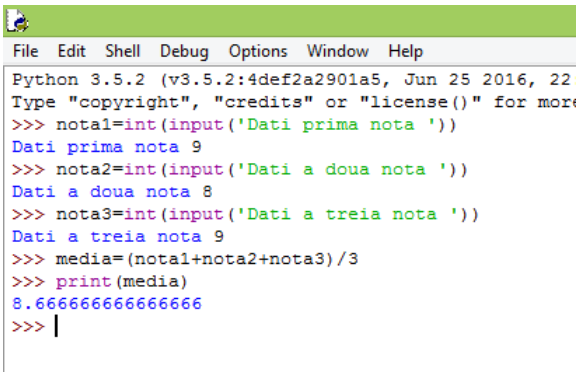
3) Funcția bool

Returnează False în cazul în care argumentul său este zero sau șirul vid și True în caz contrar.



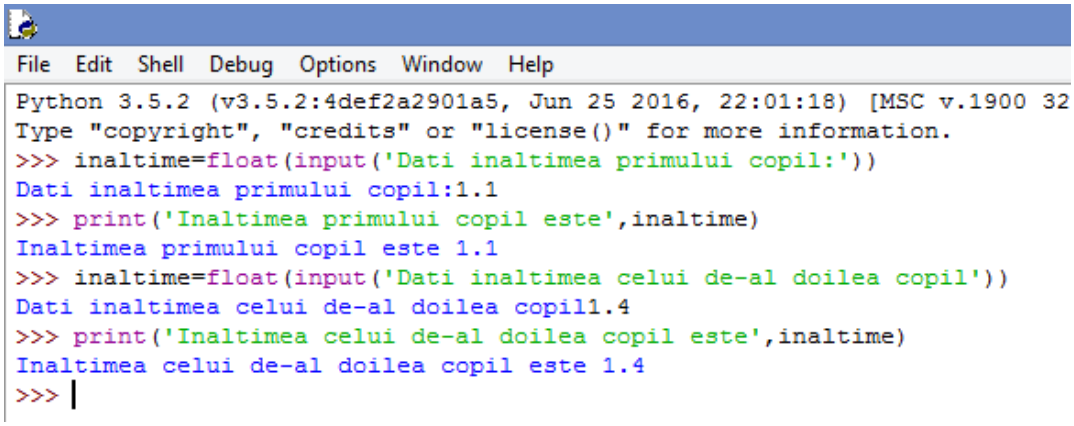
Exemple:

1. Calculul mediei aritmetice.



Deoarece notele sunt numere întregi, am folosit pentru variabilele nota1, nota2, nota3 tipul int. Observăm că notele sunt numere întregi iar rezultatul este un număr real. În cazul în care dorim să obținem partea întreagă a rezultatului împărțirii, folosim operatorul // (div) sau funcția int.

2. Presupunem că dorim să citim și să afișăm înălțimea a doi copii utilizând o singură variabilă înălțime.



Deoarece înălțimea este un număr real, vom folosi pentru variabila inaltime tipul float.

2.5 Șiruri de caractere în Python

Un șir de caractere este o succesiune de litere, cifre sau caractere speciale. Șirurile de caractere sunt memorate în variabile.

Șirurile de caractere în Python pot fi create în două moduri:

1) **Prin inițializare directă**

sir1='Planeta'

sir2='programatorilor'

Șirurile de caractere în Python trebuie introduse între apostrofuri sau între ghilimele.

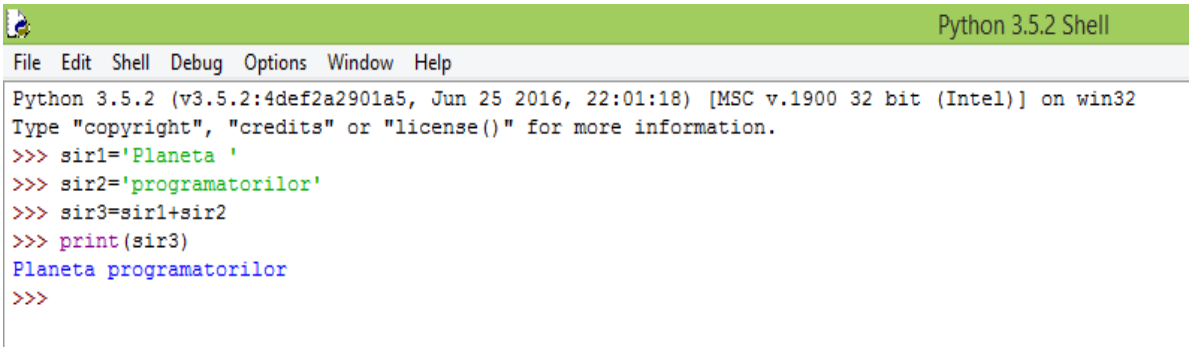
2) Prin citire de la tastatură

sir1=input('Dati un sir de caractere')

print(sir1)

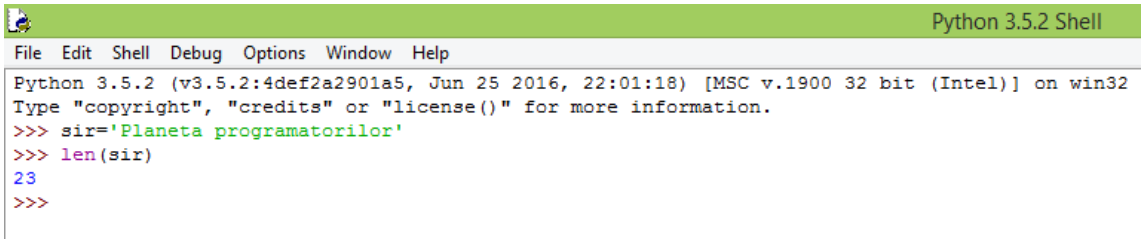
2.5.1 Concatenarea a două șiruri de caractere

Concatenarea a două șiruri de caractere este realizată de operatorul +. În urma execuției programului de mai jos pe ecran se va afișa **Planeta programatorilor**.



2.5.2 Calcularea lungimii unui șir de caractere

Funcția len(sir) este folosită pentru a afla lungimea unui șir. Python numără toate caracterele, inclusiv spațiile, pentru a calcula numărul total de caractere dintr-un șir.



2.5.3 Accesarea elementelor unui șir de caractere

Fiecare caracter dintr-un șir poate fi accesat prin numele șirului urmat de un număr. Acest număr al poziției poartă numele de indice. Primul caracter dintr-un șir în Python se află în poziția 0, al doilea în poziția 1 ș.a.m.d.

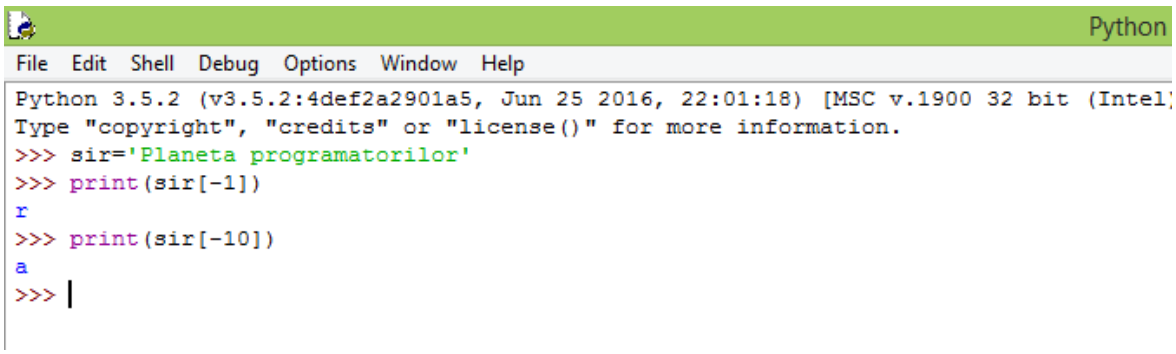
P	L	a	n	e	t	a		p	r	o	g	r	a	m	a	t	o	r	i	l	o	r
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

Observăm că cele 23 de caractere sunt dispuse în pozițiile 0 - 22.

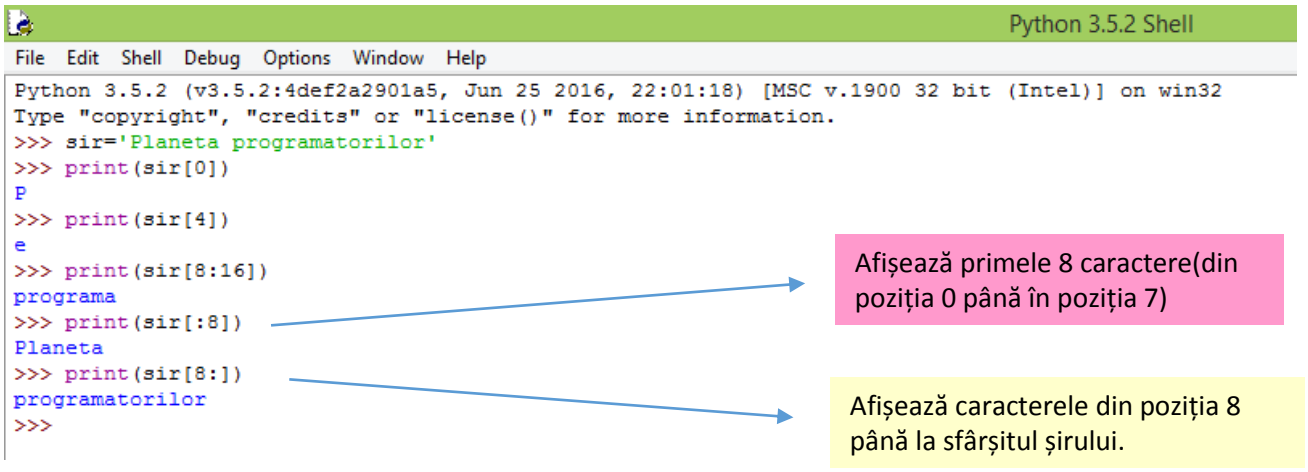
Observație importantă

Python permite și indici negativi, intervalul indicilor fiind [-len(sir), len(sir)). Completarea cu elemente în zona indicilor negativi se face automat.

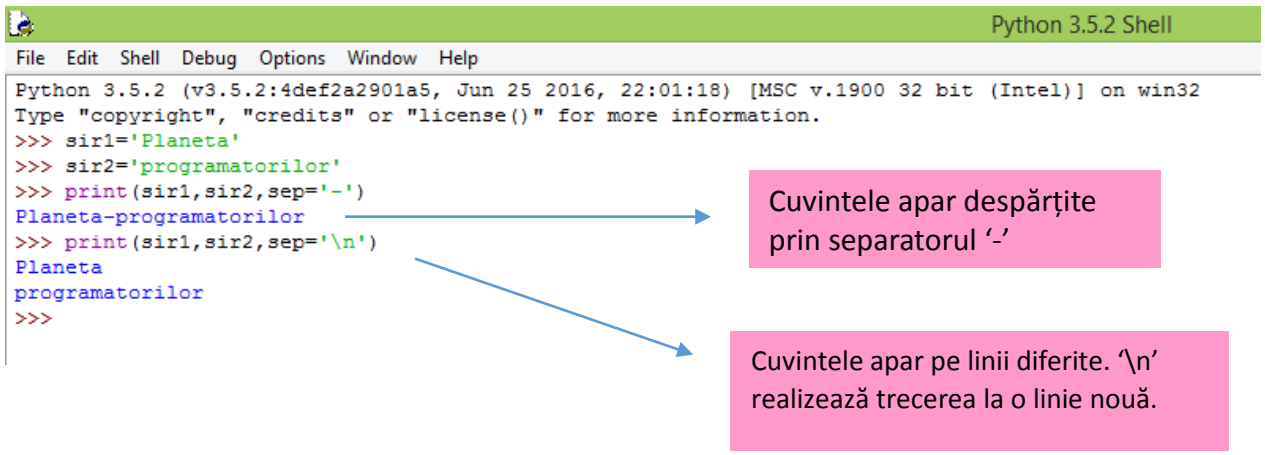
P	L	a	n	e	t	a		p	r	o	g	r	a	m	a	t	o	r	i	l	o	r
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
-23	-22	-21	-20	-19	-18	-17	-16	-15	-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1



2.5.4 Afișarea elementelor unui șir de caractere

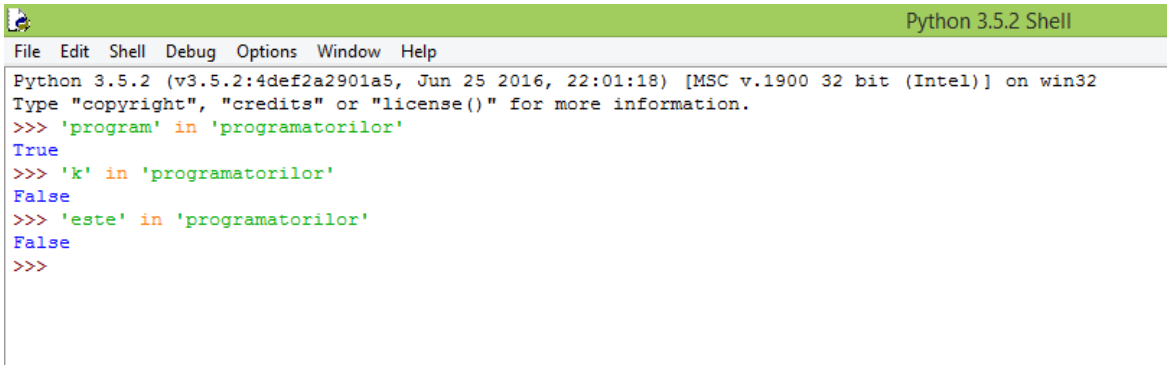


2.5.5 Separarea șirurilor de caractere



2.5.6 Operatorul in

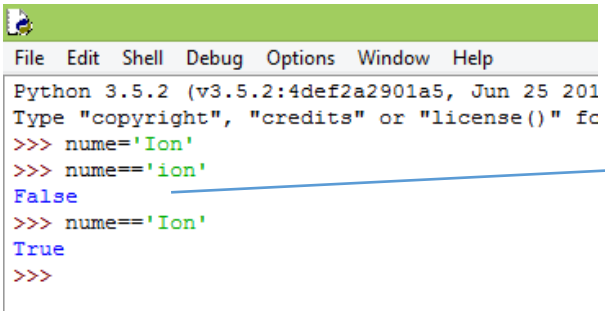
Operatorul in este utilizat pentru a vedea daca un șir de caractere se găsește în alt șir de caractere (dacă este subșir al șirului inițial).



2.5.7 Compararea șirurilor de caractere

Șirurile de caractere pot fi comparate utilizând operatorii <, <=,>,>=,== (egal) și != (diferit).

În cazul operatorului ==, șirurile trebuie să fie identice pentru ca rezultatul returnat să fie True.



Rezultatul este False deoarece Python face distincție între literele mari și mici.

2.5.8 Funcțiile ord, chr și str

Fiecare caracter are un cod numeric asociat, numit codul ASCII al caracterului. Pentru a determina codul numeric asociat unui caracter folosim funcția ord.

```
>>> ord('A')
65
>>> ord('B')
66
>>> ord('a')
97
>>> ord('b')
98
>>>
```

Literele au coduri numere consecutive, A are codul 65, B are codul 66 s.a.m.d. Literele mici au coduri diferite de literele mari. Astfel, a are codul 97, b are codul 98 s.a.m.d.

Pentru a determina caracterul corespunzător unui număr folosim funcția chr. Funcția chr poate primi ca argument un număr cuprins între 0 și 255.

```
>>> chr(65)
'A'
>>> chr(66)
'B'
>>> chr(97)
'a'
>>> chr(98)
'b'
>>>
```

Compararea șirurilor de caractere se realizează prin compararea codurilor ASCII corespunzătoare caracterelor. Caracterele se compară de la stânga la dreapta.

```
>>> 'a'<'b'
True
>>> 'Ionescu'<'Ionicescu'
True
>>> 'ab'<'abc'
True
>>>
```

Se compară codurile celor două caractere.

Primele trei caractere din cele două șiruri sunt egale. La al patrulea caracter avem relația ‘e’<‘i’. În acest moment se oprește compararea și se concluzionează că primul șir este mai mic decât al doilea.

În acest caz primele două caractere sunt identice. În cazul în care toate caracterele primului șir sunt identice cu primele caractere din al doilea șir dar lungimea primului șir este mai mică decât lungimea celui de-al doilea șir se consideră primul șir mai mic decât al doilea.

Dacă dorim să citim valori numerice și să le convertim la șiruri de caractere, folosim funcția str. Pentru a înțelege efectul acestei funcții, urmăriți exemplul de mai jos.

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> nota1=int(input('Dati prima nota '))
Dati prima nota 7
>>> nota2=int(input('Dati a doua nota '))
Dati a doua nota 8
>>> sir1=str(nota1)+','+str(nota2)
>>> print('Notele obtinute sunt ',sir1)
Notele obtinute sunt  7,8
>>>
```