Python

FIESC, C+AIA, anul I

Operații de bază

- Citirea
- Atribuirea
- Scrierea
- Oprirea

Autorul limbajului Python - Guido van Rossum.

Variabilele

- Nu se declară
- Tipul variabilelor este stabilit în timpul execuției

Tipuri de date

Numerice – Numerele sunt inmutabile – odată create valoare nu se mai poate schimba (operațiile crează noi obiecte).

int (numere întregi):

- numerele întregi (pozitive și negative), dimensiune limitat doar de memoria disponibilă
- Operații: +, -, *, /, //, **, % comparare:==,!=,<, > operații pe biți: |, ^, &, <<, >>, ~
- Literali: 1, -3

bool (boolean):

- Valorile **True** și **False**.
- Operații: and, or, not
- Literali: False, True; 0, 1

float (numere reale):

- numerele reale (dublă precizie)
- Operations: +, -, *, / comparare:==,!=,<, >
- Literali: 3.14

Numere complexe

```
>>> (1-j)*(1+j)
Traceback (most recent call last):
  File "<pyshell#96>", line 1, in <module>
    (1-j)*(1+j)
NameError: name 'j' is not defined
>>> (1-1j)*(1+1j)
(2+0j)
>>>
```

Scriere – print (v3.x)

```
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

print(“Start”)
Start

print("Buna", "ziua")
Buna ziua

a=2
b=" lei"
print(a,b)
2 lei

Citirea – input([prompt]) -> string

```
a = input("a: ")
b = input("b: ")
print (a+b)
a = int(a)
b = int(b)
print(a + b)
------------------------------
a: 1
b: 2
12
3
```

Citirea - input

```
nume=input("Nume:")
prenume=input("Prenume:")
an =input("An admitere=")
medie =input("Medie=")
print("Veti absolvi facultatea in", int(an)+4, end=" ")
medieProbabila= float(medie)+0.5
print(", probabil cu", medieProbabila)
```

Nume:**Ionescu**
Prenume:**Ion**
An admitere=**2016**
Medie=**9.40**
Veti absolvi masteratul in 2020 , probabil cu 9.9

TEST

Fie programul:

```
x=input("x=")
y=input("y=")
print(x,y)
print(x+y)
x=float(x)
y=float(y)
print(x+y)
```

Se lanseaza in executie

x=2

y=3

Ce se afiseaza in continuare ?

Scriere – print (v3.x)

```
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

print(“Start”)
Start

print("Buna", "ziua")
Buna ziua

a=2
b=" lei"
print(a,b)
2 lei

Scrierea – print (v2.7 – v3.x)

v2.7: print “Rezultat=", 2*2

v3.x: print("Rezultat=", 2*2)

v2.7: print x, *# ultima virgula suprima \n*

v3.x: print(x, end=“.") *# Adauga un punct*

v2.7: print *# afisa un \n*

v3.x: print() *# Trebuie chemata functia!*

v2.7: print >>sys.stderr, "fatal error"

v3.x: print("fatal error", file=sys.stderr)

Comentarii

- încep cu # și țin până la sfârșitul liniei
- încep cu ’’’ și țin mai multe rânduri, până la un nou ’’’

Blocuri

- ● Parte a unui program care este executată ca o unitate
- ● Secvență de instrucțiuni
- ● Se realizează prin identarea liniilor

toate instrucțiunile identate

la același nivel aparțin aceluiași bloc

if

```
if conditie:
        bloc de instructiuni
elif conditie:
        bloc de instructiuni
else:
        bloc de instructiuni
```

```
If x>0:
        abs=x
else:
        abs=-x
```

while

```
while conditie:
        bloc de instructiuni
        [break]
        [continue]
else:
        bloc de instructiuni
-------------------------
a=0
while a<5:
        print (a)
        a=a+1
print("stop")
```

0
1
2
3
4
stop

Ciclu nesfârșit

```
while True:
        pass # instructiunea vida
```

for

```
for variabila in lista:
        bloc de instructiuni
        [break]
        [continue]
else:
        bloc de instructiuni
```

```
for(i=0; i<5; i++)
   printf("%d  ", i);
```

```
0 1 2 3 4
```

```
for i in range(5):
     print (i, end=' ')
```

```
0 1 2 3 4
```

Funcția

range(stop) sau range(start, stop [,step])

Creează o listă

- [0, 1, … , stop-1]
- [start, start+1, … , stop-1]
- [start, start+step, … , *start+(n-1)*step*]

```
print(range(2))
for i in range(2):
    print (i, end=" ")
print()
for i in range(1,10,3):
    print (i, end=" ")
```

```
range(0, 2)
0 1
1 4 7
```

for

```
for i in range(10):
    print (i, end=' ')
print()

for i in range(2,100,7):
    print (i, end=' ')
print()
#
s = "abcde"
for c in s:
    print (c, end='  ')
print()
```

```
0 1 2 3 4 5 6 7 8 9

2 9 16 23 30 37 44 51 58 65 72 79 86 93

a b c d e
```

tip_date.py - D:/Python/curs_anul_I_licenta/Python/tip_date.py (3.4.4)

File Edit Format Run Options Window Help

```
for nr in range(2, 10):
    if nr % 2 == 0:
        print("Un numar par:", nr)
        continue
    print("Acesta numar e impar:", nr)
```

========= RESTART: D:/Python/curs_anul_I_licenta/Python/tip_date.py =========

Un numar par: 2
Acesta numar e impar: 3
Un numar par: 4
Acesta numar e impar: 5
Un numar par: 6
Acesta numar e impar: 7
Un numar par: 8
Acesta numar e impar: 9

```
>>> for n in range(2, 10):
...     for x in range(2, n):
...         if n % x == 0:
...             print(n, 'equals', x, '*', n//x)
...             break
...     else:
...         # loop fell through without finding a factor
...         print(n, 'is a prime number')
...
2 is a prime number
3 is a prime number
4 equals 2 * 2
5 is a prime number
6 equals 2 * 3
7 is a prime number
8 equals 2 * 4
9 equals 3 * 3
```

Tipuri de date standard

Secvențe:

- ● Mulțimi finite și ordonate, indexate prin numere ne-negative.
- ● Dacă a este o secvență atunci:
- ○ len(a) returnează numărul de elemente;
- ○ a[0], a[1], …, a[len(a)-1] elementele lui a.
- ● Exemple: [1, ‘a’], (“an”,1000)

String

- ● este o secvență inmutabilă;
- ● caractere Unicode .
- ● Literali: ‘abc’, “abc”

Liste

- • secvență mutabilă
- • ex: [] sau [1, ‘a’, [1, 3]]

Tuple

● este o secvență inmutabilă;

Ex (), (1,2), (1, 20, ’a’)

(tuple)

```
>>> t=  ( 1, 2 )
>>> t
(1, 2)
>>> t[1]
2
>>> t=t+3
... TypeError: can only concatenate tuple (not "int") to tuple
>>> t=t+(3)
... TypeError: can only concatenate tuple (not "int") to tuple
>>> t=t+ (3,)
>>> t
(1, 2, 3)
>>> t[2]=-3
...TypeError: 'tuple' object does not support item assignment
>>> len ( t )
3
>>> t5 = ( -1, ) * 5
>>> t5
(-1, -1, -1, -1, -1)
```

[liste]

```
>>> lst= [ 1, 2 ]
>>> lst
[1, 2]
>>> lst[1]
2
>>> lst = lst +3
... TypeError: can only concatenate list (not "int") to list
>>> lst = lst +(3,)
... TypeError: can only concatenate list (not "tuple") to list
>>> lst = lst +[3]
>>> lst
[1, 2, 3]
>>> lst[2]=-3
>>> lst
[1, 2, -3]
>>> len ( lst )
3
>>> l5 = [ 0 ] * 5
>>> l5
[0, 0, 0, 0, 0]
```

Exemplu. Tupla (a) si lista (b)

```
a=(1,2,'a')
b=[10,20,'b']
print(a,b,sep='\n')
print("a[0]=",a[0],"b[0]=",b[0])
# a[0]=11  # TypeError: 'tuple' object does not support item assignment
b[0]=100
print("Dupa modificare:",a,b,sep='\n')
```

```
(1, 2, 'a')
[10, 20, 'b']
a[0]= 1 b[0]= 10
Dupa modificare:
(1, 2, 'a')
[100, 20, 'b']
```

Exemplul 1 - for si o lista

```
N=10
v=[-1]*  # alocare +init
for i in range(N):
    if i%2==1:
        a=0
    else:
        a=N
    v[i]=a
print(v)
```

[10, 0, 10, 0, 10, 0, 10, 0, 10, 0]

Exemplul - 2

```
for x in [2,-6,"a",5]:
        print (x, end=' ')
```

2 -6 a 5

for: . . . else:

```
int i, x, n, a[100];
//…
for(i=0; i<n; i++)
        if(x==a[i]) {
                printf("%d este in tablou la indicele %d", x, i);
                break;
        }
//

        printf("%d nu este in tablou", x)
```

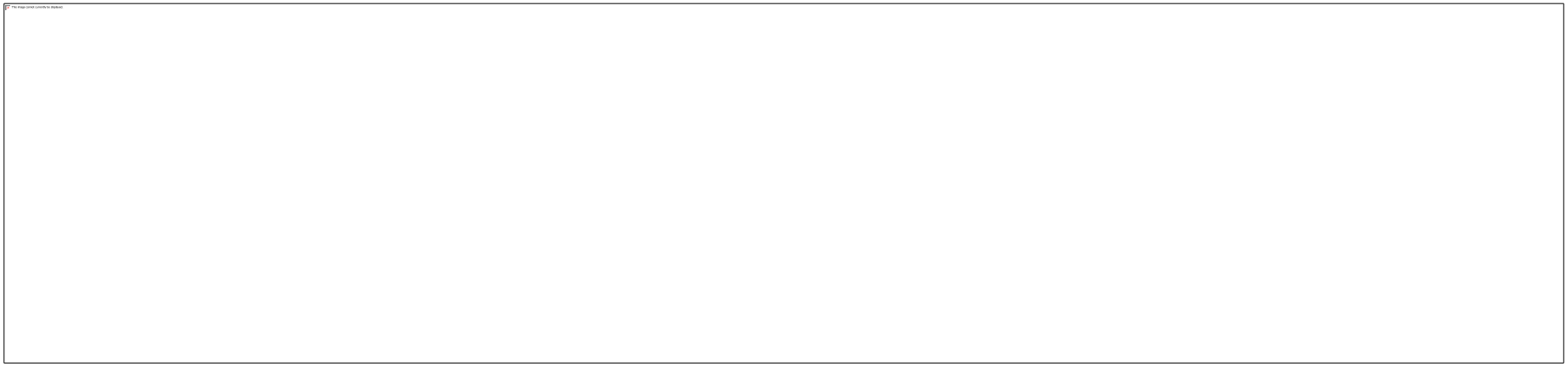
for: . . . else:

```
int i, x, n, a[100];
//…
for(i=0; i<n; i++)
        if(x==a[i]) {
                printf(“%d este in tablou la indicele %d”, x, i);
                break;
        }
//
if ( i == n )
        printf(“%d nu este in tablou”, x)
```

for: . . . else:

```
a= [ 10,20,-1, 40, 5 ]
x = int(input("x="))

for i in range(len(a)):
    if x == a[i]:
        print("%d se gaseste la indicele %d" % (x, i) )
        break;
else:
    print("%d nu se afla in tablou" % x)
```



Liste cu siruri (1)

```
>>> note=["ionescu",9, "Popescu",10]
>>> note
['ionescu', 9, 'Popescu', 10]
>>> note[0]
'ionescu'
>>> note[0][0]
'i'
>>> note[1]
9
>>> note[1][0]
TypeError: 'int' object is not subscriptable
>>>
```

Liste cu siruri (2)

```
>>> note
['ionescu', 9, 'Popescu', 10]
>>> note[0][0]='I'
Traceback (most recent call last):
  File "<pyshell#107>", line 1, in <module>
    note[0][0]='I'
TypeError: 'str' object does not support item assignment
>>> note[0]='Ionescu'
>>> note
['Ionescu', 9, 'Popescu', 10]
>>>
```

```
lst=[]
i=0
while True :
    i+=1
    c=input("cuvantul "+ str(i) +"=")
    if len(c)==0:
        break
    lst.append(c)
print(lst)
for w in lst:
    print(w," - lungime ",len(w))
```

```
======= RESTART: D:/Python/curs_anul_I_licenta/Python/listaCuvinte.py =======
cuvantul 1=Anul
cuvantul 2=I
cuvantul 3=licenta
cuvantul 4=
['Anul', 'I', 'licenta']
Anul  - lungime  4
I  - lungime  1
licenta  - lungime  7
```

Functii

```
#definitia functiei
def fact ( n ):
    f=1
    for i in range(2,n+1):
        f *= i
    return f

#programul principal
n=int ( input("n=") )
print ( n, '! =', fact(n) )
```

```
def factr ( n ):
    '''
        factr( n )
        Functia factorial implementata recursiv'''
    if n<=1:
        return 1
    else:
        return factr(n-1)*n

#programul principal
n=int ( input("n=") )
print ( n, '! =', factr(n) )
print()
print("Valoare calculata cu",factr.__doc__)
```

```
========== RESTART: C:/Python34/factrecursiv.py ===
n=19
19 ! = 121645100408832000

Valoare calculata cu
        factr( n )
        Functia factorial implementata recursiv
>>>
```

Atribuirea multiplă

```
a, b = 1, 3
a, b = b, a+b
print("a=",a,"b=",b)
```

Atribuirea multiplă

```
a, b = 1, 3
a, b = b, a+b
print("a=",a,"b=",b)
```

```
a= 3 b= 4
```

Functia Fibonacci

```
def fib(n):
"""
Print a Fibonacci series up to n."""
        a, b = 0, 1
        while a < n:
                print(a, end=' ')
                a, b = b, a+b
        print()
fib(2000)
```

```
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597
```

Utilizarea unei functii din alt modul

```
import random
u = random.randint(1,6)
print("Zarul dv. :", str(u))
c = random.randint(1,6)
print("Al meu    :" + str(c))

if u > c:
    print("Ati castigat")
elif u < c:
    print("Am castigat")
else:
    print("Egalitate")
```

```
============= RESTART: D:/Python/curs_python/PSG/zar_simplu.py =============
Zarul dv.: 4
Al meu   :2
Ati castigat
```

Utilizarea unei functii a unei clase

```
>>> sir="AbCdEf"
>>> sir.upper()
'ABCDEF'
>>> sir.lower()
'abcdef'
```

Observatie. **sir** ramane nemodificat

Utilizarea unei functii a unei clase

```
>>> sir="AbCdEf"
>>> sir.upper()
'ABCDEF'
>>> sir.lower()
'abcdef'

>>> sir=sir.upper()
>>> sir
'ABCDEF'
```

Utilizarea unei functii a unei clase

```
import random

while input("Arunc zarul ?").upper() == "D":
    u = random.randint(1,6)
    print("Zarul dv.:", str(u))
    c = random.randint(1,6)
    print("Al meu   :" + str(c))

    if u > c:
        print("Ati castigat")
    elif u < c:
        print("Am castigat")
    else:
        print("Egalitate")
```

Utilizarea unei functii a unei clase

```
import random

while input("Arunc zarul ?").upper() == "D":
    u = random.randint(1,6)
    print("Zarul dv.:", str(u))
    c = random.randint(1,6)
    print("Al meu   :" + str(c))

    if u > c:
        print("Ati castigat")
    elif u < c:
        print("Am castigat")
    else:
        print("Egalitate")
```

```
========== RESTART: D:/Python
Arunc zarul ?d
Zarul dv.: 3
Al meu   :1
Ati castigat
Arunc zarul ?d
Zarul dv.: 5
Al meu   :1
Ati castigat
Arunc zarul ?n
>>>
```

import ... as ... (păstrarea scorului)

```
import random as rnd

pctUtil = pctCalc =0
while input("Arunc zarul ?").upper() == "D":
    u = rnd.randint(1,6)
    print("Zarul dv.:", str(u))
    c = rnd.randint(1,6)
    print("Al meu   :" + str(c))
    if u > c:
        pctUtil += 1
    if c > u:
        pctCalc += 1
    print("Scorul Utilizator-Calculator:",pctUtil,"-",pctCalc, "\n")
```

```
========== RESTART: D:/Python/curs_python/PSG/zar_cuScor_while.py
Arunc zarul ?d
Zarul dv.: 5
Al meu   :2
Scorul Utilizator-Calculator: 1 - 0

Arunc zarul ?d
Zarul dv.: 1
Al meu   :1
Scorul Utilizator-Calculator: 1 - 0

Arunc zarul ?n
>>>
```

Formatare siruri

```
>>> x=3.25
>>> print("x=%f" % x)
x=3.250000

>>> s="ziua"
>>> print("Buna %s!" % s)
Buna ziua!


>>> sir = "n=%d n/2=%f n//2=%d n/3=%.2f" % (14, 14/2, 14//2, 14/3)
>>> print(sir)
n=14 n/2=7.000000 n//2=7 n/3=4.67
>>>
```

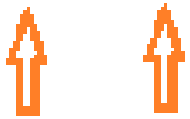
Formatare siruri

```
>>> x=3.25
>>> print("x=%f" % x)
x=3.250000

>>> s="ziua"
>>> print("Buna %s!" % s)
Buna ziua!
```

```
>>> sir = "n=%d n/2=%f n//2=%d n/3=%.2f" % (14, 14/2, 14//2, 14/3)
>>> print(sir)
n=14 n/2=7.000000 n//2=7 n/3=4.67
>>>
```

```
>>> print("x=%.2f  s=%s"  %  (x,s) )
x=3.25  s=ziua
>>>
```



Functie citire optiuni utilizator

```
def citesteMaj(mesaj, acceptate):
    while True:
        r = input(mesaj+str(acceptate)+"? ")
        r=r.lstrip()
        r=r.rstrip()
        r=r.upper()
        # r = r.lstrip().rstrip().upper()
        if r in acceptate:
            return r

opt = citesteMaj("Optiunea dv. in FIESC", ['C','AIA','EA','SE','ME','IE'])
print("Ati ales programul %s. Succes!" % opt)
```

```
============= RESTART: D:/Python/curs_python/PSG/OptiuneFIESC.py =====
Optiunea dv. in FIESC['C', 'AIA', 'EA', 'SE', 'ME', 'IE']? f
Optiunea dv. in FIESC['C', 'AIA', 'EA', 'SE', 'ME', 'IE']?     c
Ati ales programul C. Succes!
>>>
```

import modul utilizator

Fisierul citeste.py

```
def citesteMaj(mesaj, acceptate):
    while True:
        r = input(mesaj+str(acceptate)+"? ")
        r = r.lstrip().rstrip().upper()
        if r in acceptate:
            return r
```

Fisierul zaruri5.py

```
import random, citeste

pctUtil = pctCalc =0
while citeste.citesteMaj("Arunc zarul", ["D","N"]) == "D":
    u = random.randint(1,6)
    print("Zarul dv.:", str(u))
    c = random.randint(1,6)
    print("Al meu   :" + str(c))
    if u > c:
        pctUtil += 1
    if c > u:
        pctCalc += 1
    print("Scorul Utilizator-Calculator:",pctUtil,"-",pctCalc, "\n")
```

Scrieti un program care sa realizeze jocul

```
========== RESTART: C:\Users\PSG\fph.py
Ce alegeti['F', 'P', 'H', 'S']? f
Eu aleg F
La fel, reluam! Ce alegeti['F', 'P', 'H', 'S']? f
Eu aleg P
FP  - Ati pierdut
Scorul Utilizator-Calculator: 0 - 1

Ce alegeti['F', 'P', 'H', 'S']? f
Eu aleg F
La fel, reluam! Ce alegeti['F', 'P', 'H', 'S']? f
Eu aleg F
La fel, reluam! Ce alegeti['F', 'P', 'H', 'S']? f
Eu aleg P
FP  - Ati pierdut
Scorul Utilizator-Calculator: 0 - 2

Ce alegeti['F', 'P', 'H', 'S']? f
Eu aleg H
FH  - Ati castigat
Scorul Utilizator-Calculator: 1 - 2

Ce alegeti['F', 'P', 'H', 'S']? h
Eu aleg P
HP  - Ati castigat
Scorul Utilizator-Calculator: 2 - 2

Ce alegeti['F', 'P', 'H', 'S']? s
La revedere!
>>>
```

from MODUL import FUNCTIE

```
import random as rnd
from citeste import citesteMaj

optiuni=["F","P","H","S"]
castigaUtilizator=['FH', 'HP', "PF"]
pctUtil = pctCalc =0
while True:
    utilizator = citesteMaj("Ce alegeti", optiuni)
    if utilizator != optiuni[-1]:
        # calculator = str(random.choices(optiuni[:-1])[0])
        calculator = optiuni [  rnd.randint(0,2) ]
        print("Eu aleg " + calculator)
        if utilizator == calculator:
            print("La fel, reluam!", end=" ")
            continue
        jucate = utilizator + calculator
        if jucate in castigaUtilizator:
            rez= "castigat"
            pctUtil += 1
        else:
            rez="pierdut"
            pctCalc+=1
        print(jucate," - Ati " + rez)
        print("Scorul Utilizator-Calculator:",pctUtil,"-",pctCalc, "\n")
    else:
        print("La revedere!")
        break
```

Tipuri de date

String

- ● este o secvență inmutabilă;
- ● caractere Unicode .
- ● Literali: ‘abc’, “abc”

```
x="12"
# x=x+1 -> TypeError: Can't convert 'int' object to str implicitly
x=x+str(1)
print(x)
```

Rezultat: **121**

‘Siruri’ “String”

```
>>> s='abcde'
>>> s
'abcde'
>>> s [ 1 ]
'b'
>>> s = s + 5
... TypeError: Can't convert 'int' object to str implicitly
>>> s = s +  str ( 5 )
>>> s
'abcde5'
>>> s  =  s +  'g'
>>> s
'abcde5g'
>>> s [ 5 ] = 'f'
... TypeError: 'str' object does not support item assignment
>>> len(s)
7
>>> s5 = "x-“  * 5
>>> s5
'x-x-x-x-x-'
```

OPERATII CU LISTE

```
>>> lst=[10]
>>> lst = lst + [20,30]
>>> lst
[10, 20, 30]
>>> lst.append(40)
>>> lst
[10, 20, 30, 40]
>>> lst.extend([50,60])
>>> lst
[10, 20, 30, 40, 50, 60]
>>> lst.insert(6, 70)
>>> lst
[10, 20, 30, 40, 50, 60, 70]
>>> lst.insert(-1, 80)
>>> lst
[10, 20, 30, 40, 50, 60, 80, 70]
>>> lst.insert(0, 0)
>>> lst
[0, 10, 20, 30, 40, 50, 60, 80, 70]
```

OPERATII CU LISTE
comparatie **append()** cu **extend()**

```
>>> stud=[]
>>> stud.append("Ionescu")
>>> stud.append("Georgescu")
>>> stud
['Ionescu', 'Georgescu']
>>> doiAmici =["Popescu", "Vasilescu"]
>>> stud.extend( doiAmici )
>>> stud
['Ionescu', 'Georgescu', 'Popescu', 'Vasilescu']
>>> a=["Zamfir","Teodorescu"]
>>> stud.append(a)
>>> stud
['Ionescu', 'Georgescu', 'Popescu', 'Vasilescu',
  ['Zamfir', 'Teodorescu']]
```

```
>>> def fib2(n): # return Fibonacci series up to n
...     """Return a list containing the Fibonacci series up to n."""
...     result = []
...     a, b = 0, 1
...     while a < n:
...         result.append(a)    # see below
...         a, b = b, a+b
...     return result
...
>>> f100 = fib2(100)     # call it
>>> f100                 # write the result
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

OPERATII CU LISTE - căutare în liste

```
>>> lit = [ "a", "b", "c", "a", "d"]
>>> lit.count("a")
2
>>> "d" in lit
True
>>> "e" in lit
False
>>> lit.index("c")
2
>>> lit.index("e")
Traceback (most recent call last):
  File "<pyshell#158>", line 1, in <module>
    lit.index("e")
ValueError: 'e' is not in list
>>> ind= lit.index("a")
>>> ind
0
>>> lit.index("a", ind+1 )
3
```

OPERATII CU LISTE – eliminare elemente

```
>>> lista = ["Ion", 9, "Georgeta", 10]
>>> lista[1]
9
>>> del lista[1]
>>> lista.remove("Georgeta")
>>> lista
['Ion', 10]
>>> lista.remove("Georgeta")
Traceback (most recent call last):
  File "<pyshell#172>", line 1, in <module>
    lista.remove("Georgeta")
ValueError: list.remove(x): x not in list
>>> lista
['Ion', 10]
```

OPERATII CU LISTE – eliminare elemente

```
>>> lista
['Ion', 10]
>>> lista +=[7, 8]
>>> lista
['Ion', 10, 7, 8]
>>> lista.pop()
8
>>> lista
['Ion', 10, 7]
>>> lista.pop(1)
10
>>> lista
['Ion', 7]
>>>
```

OPERATII CU LISTE – eliminare elemente pop() in lista vida

```
>>> lista
['Ion', 7]
>>> lista.pop()
7
>>> lista.pop()
'Ion'
>>> lista
[]
>>> lista.pop()
Traceback (most recent call last):
  File "<pyshell#183>", line 1, in <module>
    lista.pop()
IndexError: pop from empty list
>>>
```

OPERATII CU LISTE – copiere referință

```
>>> c=stud
>>> c
['Avram', 'Georgescu', 'Popescu', 'Vasilescu']
>>> del c[0]
>>> c
['Georgescu', 'Popescu', 'Vasilescu']
>>> stud
['Georgescu', 'Popescu', 'Vasilescu']
```

OPERATII CU LISTE – copiere listă

```
>>> w = stud.copy()
>>> w
['Georgescu', 'Popescu', 'Vasilescu']
>>> stud.append("Zamfirescu")
>>> stud
['Georgescu', 'Popescu', 'Vasilescu', 'Zamfirescu']
>>> w
['Georgescu', 'Popescu', 'Vasilescu']
>>> w.insert(0,"Nimeni")
>>> w
['Nimeni', 'Georgescu', 'Popescu', 'Vasilescu']
>>> stud
['Georgescu', 'Popescu', 'Vasilescu', 'Zamfirescu']
>>>
```

OPERATII CU LISTE - Slicing

```
>>> stud
['Ionescu', 'Georgescu', 'Popescu', 'Vasilescu']
>>> primii2= stud[0:2]
>>> primii2
['Ionescu', 'Georgescu']
>>> stud.insert(0,"Avram")
>>> stud
['Avram', 'Ionescu', 'Georgescu', 'Popescu', 'Vasilescu']
>>> primii2
['Ionescu', 'Georgescu']
>>> stud.remove("Ionescu")
>>> stud
['Avram', 'Georgescu', 'Popescu', 'Vasilescu']
>>> primii2
['Ionescu', 'Georgescu']
>>>
```

OPERATII CU LISTE - **Slicing**

```
>>> lst=[0,10,20,30,40]
>>> lst[1:3]
[10, 20]
>>> lst[0:]
[0, 10, 20, 30, 40]
>>> lst[:]
[0, 10, 20, 30, 40]
>>> lst[:-1]
[0, 10, 20, 30]
>>> lst[:3]
[0, 10, 20]
>>> lst[3:]
[30, 40]
>>> a = lst[:3]+lst[3:]
>>> a
[0, 10, 20, 30, 40]
```

OPERATII CU LISTE - **Slicing**

```
>>> lst
[0, 10, 20, 30, 40]
>>> del lst[2:4]
>>> lst
[0, 10, 40]
>>> lst.insert(2,[2,3])
>>> lst
[0, 10, [2, 3], 40]
>>> lst[2:-1]
[[2, 3]]
>>> lst[2][0]
2
>>> lst[2][:]
[2, 3]
>>> lst[2]
[2, 3]
```

OPERATII CU LISTE

```
>>> stud
['Georgescu', 'Adamescu', 'Popescu', 'Vasilescu', 'Zamfirescu']
>>> stud.reverse()
>>> stud
['Zamfirescu', 'Vasilescu', 'Popescu', 'Adamescu', 'Georgescu']
>>> stud.sort()
>>> stud
['Adamescu', 'Georgescu', 'Popescu', 'Vasilescu', 'Zamfirescu']
>>> stud.reverse()
>>> stud
['Zamfirescu', 'Vasilescu', 'Popescu', 'Georgescu', 'Adamescu']
>>>
```

Operatii cu liste - **zip**

```
l1=['a', 'b' ,'c']
l2=[10, 20, 30]
iterZip=zip(l1,l2)
print("iterZip=",iterZip)
for x in iterZip:
    print(x)

lzip=list( zip( l1, [0,1]))
print("lzip=",lzip)
```

```
iterZip= <zip object at 0x0233FC88>

('a', 10)
('b', 20)
('c', 30)

lzip= [('a', 0), ('b', 1)]
```

Operatii cu liste - **zip**

```
nume = ['Ionescu', 'Popescu',
'Georgescu']
note = [10, 9, 8, 7, 6]
objZip = zip ( nume, note )
lzip = list ( objZip )
print("lzip=", lzip)

# x,y=(zip( *objZip ) ) #### need
more than 0 values to unpack

st, n = zip ( *zip(nume,note) )
print("st=",st)
print ("n=",n)
```

lzip= [('Ionescu', 10), ('Popescu', 9), ('Georgescu', 8)]

st= ('Ionescu', 'Popescu', 'Georgescu')
n= (10, 9, 8)

```
>>> lzip[0]
('Ionescu', 10)
>>> lzip[0][1]
10
>>> lzip[0][1]=9
TypeError: 'tuple' object does not
support item assignment
>>> n[0]=5
TypeError: 'tuple' object does not ...
```

OPERATII CU LISTE – **Vectori si matrici**

List Comprehension - 1

```
Lst = [ ]
for x in range (5000):
    Lst.append ( 2*x + 1 )
print ( Lst )
```

```
>>> lst_impare = [ 2*x+1 for x in range(5000) ]
>>> lst_impare
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33,
35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63,
65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93,
95, 97, 99, 101, 103, 105, 107, 109, 111, 113, 115, 117, ...
...
9991, 9993, 9995, 9997, 9999]
>>>
```

List Comprehension 2

```
import math
alfa = [ i*math.pi/4  for  i  in  range(9) ]
valsin=[]
for  x  in  alfa:
   valsin.append( math.sin(x) )
print(valsin)
```

```
>>> alfa = [ i * math.pi / 4  for  i  in  range(9) ]
>>> alfa
[0.0, 0.7853981633974483, 1.570796326794896,
2.356194490192345, 3.141592653589793, 3.926990816987241,
4.712388980384689, 5.497787143782138, 6.283185307179586]
>>> [ math.sin(x)  for  x  in  alfa ]
[0.0, 0.7071067811865475, 1.0, 0.7071067811865476,
1.2246467991473532e-16, -0.7071067811865475, -1.0, -
0.7071067811865477, -2.449293598294706e-16]
```

List Comprehension - 3

```
cuv = []
for x in ["a", "e", "c"]:
    for y in ["a", "b", "c"]:
        if x != y:
            cuv.append(x+y)
print(cuv)
```

['ab', 'ac', 'ea', 'eb', 'ec', 'ca', 'cb']

```
cc = [ x+y for x in ["a", "e", "c"] for y in ["a", "b", "c"] if x!=y ]
print(cc)
```

['ab', 'ac', 'ea', 'eb', 'ec', 'ca', 'cb']

Dictionar

```
dict={}
dict[0] = "primul"
dict["unu"] = 2
print dict          # {0: 'primul', 'unu': 2}
dict2 = dict
dict2[3] = "ceva"
del dict2["unu"]
print dict                  # {0: 'primul', 3: 'ceva'}
print len(dict)             # 2
print dict.has_key("5")     # False
if 0 in dict:
        dict["3"] = 2
print dict                  # {0: 'primul', '3': 2, 3: 'ceva'}
print dict.keys()           # [0, '3', 3]
```