Citirea unui vector cu ajutorul listelor

```
v=[]
x=1
while x!=0:
    x= int(input("Nr. intreg (0=gata): "))
    v.append(x)
print(v)
```

```
>>>================ RESTART: C:\Users\PSG\vector1.py
Nr. intreg (0=gata): 1
Nr. intreg (0=gata): 10
Nr. intreg (0=gata): 11
Nr. intreg (0=gata): 21
Nr. intreg (0=gata): 101
Nr. intreg (0=gata): 0
[1, 10, 11, 21, 101, 0]
>>>
```

Lista/Vector de siruri

```
w=[]
while True:
    x= input("Cuvant: ")
    if len(x)==0:
        break
    w += [x]
print(w)
```

```
>>>====================== RESTART: C:/Users/PSG/vector2.py
Cuvant: casa
Cuvant: masina
Cuvant: firma
Cuvant:
['casa', 'masina', 'firma']
>>>
```

Vector initializat cu o functie matematica

```
import math
w=[]
for x in range(0,10):
    w += [math.sqrt(x)]
print(w)
```

```
================= RESTART: C:/Users/PSG/vector2_1.py
[0.0, 1.0, 1.4142135623730951, 1.7320508075688772, 2.0,
2.23606797749979, 2.449489742783178, 2.6457513110645907,
2.8284271247461903, 3.0]
```

```
>>> import math
>>> math.ceil(1.0001)
2
>>> math.ceil(-2.0001)
-2
>>> math.floor(1.999)
1
>>> math.floor(-2.999)
-3
>>> math.fabs(-2.999)
2.999
>>> math.factorial(3)
6
>>> sum([.1, .1, .1, .1, .1, .1, .1, .1, .1, .1])
0.9999999999999999
>>> math.fsum([.1, .1, .1, .1, .1, .1, .1, .1, .1, .1])
1.0
>>> math.isfinite(0)
True
```

```
>>> math.modf(2.53)
(0.5299999999999998, 2.0)
>>> fract,intreg = math.modf(-3.75)
>>> print(fract,intreg)
-0.75 -3.0
>>>
>>> math.trunc(9.99)
9
>>> math.exp(1)
2.718281828459045
>>> math.e
2.718281828459045
>>> math.exp(0.5)
1.6487212707001282
>>> math.sqrt( math.e )
1.6487212707001282
>>> g = math.log(10)
>>> g
2.302585092994046
>>> math.exp(g)
10.000000000000002
>>> math.log(1024,2)
10.0
>>> 2**10
1024
```

```
>>> math.sin( math.pi/2 )
1.0
>>> math.asin(1)
1.5707963267948966
>>> gr = math.degrees (math.pi/4 )
>>> gr
45.0
>>> math.radians(gr)
0.7853981633974483
>>> math.radians(gr)*4
3.141592653589793
>>> math.acos(-1)
3.141592653589793
>>> math.tan( math.radians(90) )
1.633123935319537e+16
>>> t = math.tan( math.radians(270) )
>>> t
5443746451065123.0
>>> math.degrees ( math.atan(t) )
89.99999999999999
>>> math.cos( math.radians(180) )
-1.0
```

```
>>> abs(-10.2)
10.2
>>> c,r = divmod(19, 8)
>>> print(c,r)
2 3
>>> expresie = "c*8+r"
>>> eval ( expresie )
19
>>> eval("22/7")
3.142857142857143
>>> max(1,-2,3,-7)
3
>>> max([10,-2, 30, -4])
30
>>> m=[[1,2],[-3,4]]
>>> min(m)
[-3, 4]
>>> min([1,2,-3,4])
-3
>>> min(1,0,1,2,0)
0
>>> ord('0')
48
>>> hex ( ord('A'))
'0x41'
>>> chr( 0x42 )
'B'
```

```
>>> repr(3.14)
'3.14'
>>> reversed( [10,2,30,40] )
<list_reverseiterator object at 0x02354E50>
>>> print(reversed( [10,2,30,40] ))
<list_reverseiterator object at 0x02354ED0>
>>> for x in reversed( [10,2,30,40] ):
        print(x, end=' ')


40 30 2 10
>>> sorted( [10,2,30,40] )
[2, 10, 30, 40]
>>> round( math.pi, 5)
3.14159
>>> math.pi
3.141592653589793
>>> round( 3.245, 2)
3.25
>>> str(23.5)
'23.5'
>>> sum([10,20,30,40])
100
>>> sum([10,20,30,40], -1)
99
>>> sum(['a','b'])
Traceback (most recent call last):
```

```
>>> ''.join(['a','b'])
'ab'
>>> zip( [1,2], [3,4,5])
<zip object at 0x02359468>
>>> lz = zip( [1,2], [3,4,5])
>>> list( lz )
[(1, 3), (2, 4)]
>>> l1,l2 = zip ( *zip([1,2], [3,4,5]))
>>> print(list(l2))
[3, 4]
>>> print(list(l1))
[1, 2]
>>>
```

Alocare vector

```
import random
N=10
nr=[0]*N  #alocare + initializare

for i in range(10000*N):
```

```
    ind = random.randint(1,N)
    nr[ind-1] +=1
print(nr)
```

```
>>>
========== RESTART: C:/Users/PSG/vector3_rand.py =====
[10023, 9918, 10072, 9969, 10101, 9982, 9820, 10005, 10104, 10006]
>>>

========== RESTART: C:/Users/PSG/vector3_rand.py =====
[9930, 10120, 9987, 9956, 9819, 10022, 9942, 9981, 10144, 10099]
>>>
========== RESTART: C:/Users/PSG/vector3_rand.py =====
[9969, 10002, 10063, 10098, 10010, 10131, 9983, 9922, 9775, 10047]
>>>
========== RESTART: C:/Users/PSG/vector3_rand.py =====
[9948, 9832, 10109, 10123, 10034, 10031, 9815, 10033, 9954, 10121]
>>>
```

Citirea unui vector de note

```
note=[]
while True:
    x=input("nota=")
    if len(x)==0:
        break
    x = int(x)
    if x>=1 and x<=10:
        note.append(x)
    else:
        print("Nota incorecta")
print("Notele corect introduse:", note)
print("Media= ", round( sum(note)/len(note),2) )


import statistics
print(statistics.mean(note))
```

Citirea unei matrici

```
stud=[]
while True:
    nume=input("Nume:").rstrip()
    if nume=="":
        break
```

```
    nota=int(input("Nota="))
    stud.append([nume, nota])
print(stud)
```

```
Nume:Ionescu
Nota=10
Nume:Georgescu
Nota=3
Nume:Vasilescu
Nota=6
Nume:
[['Ionescu', 10], ['Georgescu', 3], ['Vasilescu', 6]]
```

Prelucrarea unei matrici (1 – studentul cu cea mai mare nota)

```
stud=[]
while True:
   nume=input("Nume:")  # s-a omis .rstrip()
   if nume=="":
      break
   nota=int(input("Nota="))
   stud.append([nume, nota])
print(stud)
#
imax=0
for i in range(len(stud)):
   if stud[i][1] > stud[imax][1]:
      imax = i

print("Studentul cu cea mai mare nota:")
print(stud[imax])
```

```
============ RESTART: D:/Python/curs_python/PSG/matriceNotaMax.py ============
Nume:Ionescu
Nota=9
Nume:Popescu
Nota=4
Nume:Georgescu
Nota=10
Nume:Vasilescu
Nota=8
Nume:
[['Ionescu      ', 9], ['Popescu', 4], ['Georgescu', 10], ['Vasilescu', 8]]
Studentul cu cea mai mare nota:
['Georgescu', 10]
>>>
```

Prelucrarea unei matrici (2 – studentul cu cea mai mica nota de trecere)

```
stud=[]
while True:
    nume=input("Nume:").strip()
    if nume=="":
        break
    nota=int(input("Nota="))
    stud.append([nume, nota])
print(stud)
#
# Cea mai mica nota de trecere
#
notamin=11
for linie in stud:
    if linie[1] < 5:
        continue
    if linie[1] < notamin:
        numemin, notamin  = linie[0], linie[1]

if notamin < 11:
    print("Studentul cu cea mai mica nota de trecere:")
    print(numemin, notamin)
```

Functii aplicabile sirurilor

```
>>> "    abc  defgh    ".rstrip()
'    abc  defgh'
>>> "    abc  defgh    ".lstrip()
'abc  defgh    '
>>> "    abc  defgh    ".lstrip().rstrip()
'abc  defgh'

>>> "    abc  defgh    ".strip()
'abc  defgh'
>>>
>>> "ionescu ion".title()
'Ionescu Ion'
>>>
```

```
>>> "    abc  defgh    ".split()
['abc', 'defgh']
>>> '10,20,30,40'.split(',')
['10', '20', '30', '40']
>>>
>>> '10 20,30 40'.split(', ')
['10 20,30 40']
>>> '10 20, 30 40'.split(', ')
['10 20', '30 40']

>>> "ionescu ion".capitalize()
'Ionescu ion'
```

Functia split()

```
>>> "   abc  de  15   ".split()
['abc', 'de', '15']
>>>
>>> "10, 20, 30".split(",")
['10', ' 20', ' 30']
>>> "10, 20, 30".split(", ")
['10', '20', '30']
>>> "nu stiu nu pot nu inteleg".split('nu')
['', ' stiu ', ' pot ', ' inteleg']
>>>
```

```
stud=[]
while True:
    sir = input("Nume, nota:").strip()
    if len(sir)==0:
        break
    nume,nota = sir.split()
    stud.append([nume, int(nota)])
print(stud)
```

```
====================== RESTART: C:/Users/PSG/matrice4.py =====
Nume, nota:   Ionescu    10
Nume, nota:Popescu 5
Nume, nota: Georgescu   7
Nume, nota:
[['Ionescu', 10], ['Popescu', 5], ['Georgescu', 7]]
>>>
```

Analiza generarii de numere aleatoare

```
N=10
N_serii=5

mat=[None]*N_serii
for i in range (N_serii):
    mat[i]=[0]*N
print(mat)
print()

import random
for i in range (N_serii):
    for nr in range(10000*N):
        j = random.randint(1,N) - 1
        mat[i][j] += 1

for serie in mat:
    print(serie)

medii=[0]*N
```

```
for nr in range(0,N):
    for i in range(N_serii):
        medii[nr] += mat [i][nr]
    medii[nr] /= N_serii

print("\nNr. mediu de generari in cele %d serii" % N_serii)
print(medii)
```

```
====================== RESTART: C:/Users/PSG/matrice5.py
======================
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0,
0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0,
0, 0, 0, 0, 0, 0, 0, 0]]

[10035, 9960, 10075, 10022, 10047, 9977, 10153, 9891, 9900, 9940]
[10139, 10025, 10027, 9865, 9873, 10120, 9998, 9956, 10002, 9995]
[9918, 9841, 10104, 10050, 9891, 10111, 10085, 10025, 9982, 9993]
[9942, 9986, 10046, 10013, 9944, 10086, 10022, 9941, 9992, 10028]
[10006, 10083, 10060, 9948, 9899, 9963, 10171, 10027, 9920, 9923]
>>>
====================== RESTART: C:/Users/PSG/matrice5.py
======================
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0,
0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0,
0, 0, 0, 0, 0, 0, 0, 0]]

[9854, 9832, 10109, 10034, 9930, 9986, 10117, 10149, 9877, 10112]
[10097, 9822, 10014, 9956, 10202, 10030, 9848, 9957, 10025, 10049]
[10041, 9895, 10030, 10021, 9990, 10044, 9963, 9860, 10087, 10069]
[10099, 9900, 9922, 9999, 9987, 10079, 10092, 10020, 9920, 9982]
[10056, 9914, 10107, 10055, 9901, 9970, 10058, 10019, 10073, 9847]
>>>
```

Alta modalitate de alocare a matricilor

Utilizand List Comprehension

Exemplu Produsul a 2 matrici. Alocarea matricii produs, C, cu functia anterioara, calculul se poate programa astfel

```
    for i in range(nliniiA):
        for j in range(ncolB):
            C[i][j]=0
            for k in range(nliniiB):
                C[i][j] += A[i][k]*B[k][j]
```

Utilizand List Comprehension functia **produsMat(A,B)** arata astfel

```
def produsMat (a,b):
    nliniiA, ncolA = len(a), len(a[0])
    nliniiB, ncolB = len(b), len(b[0])
```

```
    if  ncolA != nliniiB:
        return None
    c = [ [0]*ncolB for _ in range(nliniiA)]
    for i in range(nliniiA):
        for j in range(ncolB):
            c[i][j] = sum ( [ a[i][k]*b[k][j] for k in range(nliniiB) ] )
    return c


if __name__ == '__main__':
    A=[[1,2],[3,4], [5,6]]
    B=[[1,0],[0,1]]
    C = produsMat(A,B)
    print(A)
    print(B)
    print(C)
```

Simbolul _ este considerat literă. Se folosetse atunci cand nu este necesară indicarea literei/simbolului, așa ca în expresia se mai jos:

```
[ [0]*ncolB for _ in range(nliniiA)]
```

În următoarea este însă necesară indicarea variabilei *k*

```
[ a[i][k]*b[k][j] for k in range(nliniiB) ]
```

Functii cu parametri impliciti

Sa se scrie un program care cu iesirea de mai jos utilizand o functie care returneaza True sau False

```
====================== RESTART: d:/Desktop/intrebari.py =
Ati realizat tema['D', 'N'] ?:D
Felicitari
>>>
====================== RESTART: d:/Desktop/intrebari.py =
Ati realizat tema['D', 'N'] ?:N
Cat timp ati alocat['putin', 'mult'] ?:putin
Reluati lucrul
>>>
====================== RESTART: d:/Desktop/intrebari.py =
Ati realizat tema['D', 'N'] ?:N
Cat timp ati alocat['putin', 'mult'] ?:mult
Cereti o consultatie
>>>
```

```
def intrebare(text, raspuns=['D','N']):
    '''
    Functia accepta doar variantele de
    raspuns din par2.
    Returneaza True daca este prima varianta
    '''
    while True:
        r = input ( text + str(raspuns) + ' ?:' ).strip()
```

```
        if r in raspuns:
            return r==raspuns[0]

if intrebare("Ati realizat tema"):
    print ("Felicitari")
elif intrebare("Cat timp ati alocat", ['putin','mult']):
    print('Reluati lucrul')
else:
    print('Cereti o consultatie')
```

Argumentele implicite se evalueaza la definitie

```
ind=0
def ins(l, x, i=ind):
    l.insert(i, x)
    print('S-a inserat %d inaintea elem. %d'
% (x,i))

l=[]
ins(l, 10)
ins(l, 20)
ind = 1
print("ind=", ind)
ins(l, 30)
print(l)

ins(l, 40, ind)
print(l)
```

```
==== RESTART: C:/fctParImplicit.py
===
S-a inserat 10 inaintea
elem. 0
S-a inserat 20 inaintea
elem. 0
ind= 1
S-a inserat 30 inaintea
elem. 0
[30, 20, 10]
S-a inserat 40 inaintea
elem. 1
[30, 40, 20, 10]

>>> ins(l, 50, -1)
S-a inserat 50 inaintea elem.
-1
>>> l
[30, 40, 20, 50, 10]
```

Indicele -1 este totdeauna indicele ultimei locatii.

Utilizarea unui modul separat care conține funcțiile

Atunci cand functiile sunt necesare și în alte programe

- Nu se va face copy/paste
- Se va crea un nou fișier (ex. fct.py) care să conțină numai funcțiile

Se lanseaza în execuție fișierul unde sunt apelate (aici se va face **import fct**.)

progMain.py

```
import fct

mat = fct.Matrice(3, 5)
print(mat)
```

fct.py

```
def Matrice(nlin, ncol):
    mat=[None]*nlin
    for i in range (nlin):
        mat[i]=[0]*ncol
    return mat
```

Exemplu. Alocare matrici cu functii utilizator (*fisierul aloca.py*)

```
def Matrice(nlin, ncol):
    mat=[None]*nlin
    for i in range (nlin):
        mat[i]=[0]*ncol
    return mat

def Vector(n):
    return [0]*n
```

Fisierul seriiAleat.py

```
import aloca

N=10
N_serii=15
mat = aloca.Matrice(N_serii, N)
print(mat)
print()

import random
for i in range (N_serii):
    for nr in range(10000*N):
        j = random.randint(1,N) - 1
        mat[i][j] += 1

for serie in mat:
    print(serie)

medii= aloca.Vector(N)
for nr in range(0,N):
    for i in range(N_serii):
        medii[nr] += mat [i][nr]
    medii[nr] = int (medii[nr]/N_serii)

print("\nNr. mediu de generari in cele %d serii" % N_serii)
print(medii)

========== RESTART: C:/Users/PSG/matrice5_fct.py
==========
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0,
0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0,
0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0,
0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0,
0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0,
0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0,
0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0,
0, 0, 0]]

[9978, 10100, 9967, 9857, 10043, 10013, 10057, 9988, 9991, 10006]
[9982, 9918, 9816, 10001, 9922, 9911, 10113, 10109, 10128, 10100]
[10056, 9935, 10035, 10049, 9871, 9877, 10328, 10015, 9929, 9905]
[9929, 10056, 10097, 10147, 10033, 9901, 10007, 10013, 9938, 9879]
[9929, 9990, 10103, 9930, 10107, 9936, 10014, 9976, 10078, 9937]
```

```
[9727, 10140, 10051, 9987, 9919, 9996, 10007, 10010, 10078, 10085]
[10056, 10056, 10059, 10007, 10031, 9925, 10177, 10137, 9815, 9737]
[9902, 9981, 10103, 9933, 9970, 10055, 9951, 10003, 10147, 9955]
[10098, 9839, 9995, 10054, 9902, 10093, 9978, 9861, 10123, 10057]
[10046, 9833, 10100, 10018, 9975, 9933, 9941, 10009, 10018, 10127]
[9979, 10107, 10058, 9789, 9992, 9815, 10135, 9919, 10144, 10062]
[9940, 9992, 9960, 10063, 9903, 9985, 10083, 10015, 10108, 9951]
[10197, 9888, 9997, 10196, 9988, 9862, 10092, 9764, 10013, 10003]
[9950, 10014, 10211, 9966, 9874, 9977, 10041, 9954, 10071, 9942]
[9990, 9946, 10014, 10032, 9833, 9971, 9988, 9989, 10144, 10093]

Nr. mediu de generari in cele 15 serii
[9983, 9986, 10037, 10001, 9957, 9950, 10060, 9984, 10048, 9989]
>>>
```

Inserarea unei secvente de test în modulul cu funcţii

principal.py	functii.py

`print("Start principal")`
`import functii`
 	
`print("Din principal")`
`functii.g()`
	`def f():`
	`  print ('f()')`
	
	`def g():`
	`  print ('g()')`
	
	`f()`
	
	`if __name__ == '__main__':`
	`  print('Din test functii')`
	`  g()`

```
================ RESTART: D:/Python/curs_python/principal.py ==
Start principal
f()
Din principal
g()
>>>
>>>
================== RESTART: D:\Python\curs_python\functii.py ===
f()
Din test functii
g()
>>>
```

Reluare exemplu precedent (*Fişierul aloca2.py*)

```
def Matrice(nlin, ncol, valInit=0):
    mat=[None]*nlin
    for i in range (nlin):
        mat[i]=[valInit]*ncol
    return mat

def Vector(n, valInit=0):
    return [valInit]*n
```

```
if __name__ == "__main__":
    v=Vector(10)
    print("v=",v)

    w=Vector(3,"Anul I")
    print("w=",w)

    m=Matrice(2,3)
    print("m=",m)

    m1=Matrice(3,4, 1)
    print("m1=",m1)

========== RESTART: C:/Users/PSG/aloca2.py ==========
v= [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
w= ['Anul I', 'Anul I', 'Anul I']
m= [[0, 0, 0], [0, 0, 0]]
m1= [[1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1]]
>>>
```

Fisierul *principalAleat.py*

```
import aloca2

N=10
N_serii=15
mat = aloca2.Matrice(N_serii, N)
print(mat)
print()

import random
for i in range (N_serii):
    for nr in range(10000*N):
        j = random.randint(1,N) - 1
        mat[i][j] += 1

for serie in mat:
    print(serie)

medii= aloca2.Vector(N,10000) #intentionat
for nr in range(0,N):
    for i in range(N_serii):
        medii[nr] += mat [i][nr]
    medii[nr] = int (medii[nr]/(N_serii+1))  #s-a initializat cu
10000

print("\nNr. mediu de generari in cele %d serii" % N_serii)
print(medii)
'''
>>>
>>>
================ RESTART: C:/Users/PSG/matrice5_fct_aloca2.py
================
```

```
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0,
0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0,
0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0,
0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0,
0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0,
0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0,
0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], [0, 0, 0, 0, 0, 0, 0,
0, 0, 0]]

[9912, 9856, 10016, 10089, 10095, 10077, 10008, 9962, 9999, 9986]
[9956, 9807, 9898, 10146, 9948, 9977, 9892, 10132, 10129, 10115]
[9842, 9940, 10115, 10155, 9839, 10292, 9943, 9958, 9997, 9919]
[10112, 10081, 9909, 9893, 10079, 9990, 9880, 10062, 10002, 9992]
[10085, 9918, 9945, 9999, 9991, 9993, 10034, 10076, 9934, 10025]
[9997, 9990, 10267, 9986, 9958, 9809, 9964, 10056, 10031, 9942]
[9833, 10112, 9960, 10068, 9928, 9968, 9998, 10120, 9978, 10035]
[10140, 9780, 10052, 10016, 9739, 10052, 10037, 10119, 10043, 10022]
[9974, 9930, 9970, 10020, 10081, 9936, 10053, 9895, 10177, 9964]
[9917, 10057, 10049, 10172, 10091, 9981, 10012, 9951, 9901, 9869]
[10023, 10016, 10017, 9927, 9974, 9990, 10083, 10017, 9952, 10001]
[9970, 10053, 10073, 10036, 9821, 9962, 9929, 10111, 10020, 10025]
[9920, 10301, 9957, 9952, 10094, 9886, 9719, 10128, 10118, 9925]
[9976, 9967, 10090, 9953, 9963, 10161, 9923, 9972, 10093, 9902]
[9973, 9971, 9974, 9959, 9975, 10008, 9992, 9983, 10029, 10136]

Nr. mediu de generari in cele 15 serii
[9976, 9986, 10018, 10023, 9973, 10005, 9966, 10033, 10025, 9991]
>>>
'''
```