Optimizarea codului Python

Ciocan Irina
Facultatea de Matematică și Informatică

Profiler

Tema prezentării: rescrierea unor bucăți de cod comune într-un mod care să ruleze mai repede și eficient

Mod de verificare: am rulat codul inițial și cel optimizat comparând rezultatele cu ajutorul profilerului cProfile (https://docs.python.org/3/library/profile.html#module-cProfile)

Moduri de invocare:

În linia de comandă:

```
python -m cProfile myscript.py
```

În script:

```
import cProfile
cProfile.run('functie()')
```

Profiler

Semnificația identificatorilor din tabel:

- ncalls: numărul de apeluri
- tottime: timpul total (agregat) în care a fost executată funcția curentă
- percall: Raportul dintre timpul total și numărul de apeluri (cât a durat în medie o executare a acelei funcții\
- cumtime: Timpul cumulat al executării funcției, împreună cu funcțiile apelate de către ea
- percall: Se referă la al doilea percall din raport. Reprezintă raportul dintre timpul cumulat (cumtime) și numărul de apeluri (ncalls)
- filename_lineno(function): Punctual din program, care a fost evaluat (de exemplu un număr de linie din program sau un apel de funcție).

Concatenarea șirurilor

Folosirea metodei join() în locul unor concatenări repetate cu operatorul "+"

```
import cProfile
import random
import string
lista_sir_random=["".join([random.choice(string.ascii_lowercase)for x in range(random.randint(3,7))]) for x in range(1000000)]


def concat1():
    concatenare=""
    for sir in lista_sir_random:
        concatenare+= sir
    return concatenare

def concat2():
    concatenare="".join(lista_sir_random)
    return concatenare

cProfile.run('concat1()')
cProfile.run('concat2()')
```

Concatenarea șirurilor

```
PS D:\ordonate\politehnica\optimizari_python> python test.py
         4 function calls in 1.050 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    1.050    1.050 <string>:1(<module>)
        1    1.050    1.050    1.050    1.050 test.py:10(concat1)
        1    0.000    0.000    1.050    1.050 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}


         5 function calls in 0.017 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.017    0.017 <string>:1(<module>)
        1    0.000    0.000    0.016    0.016 test.py:16(concat2)
        1    0.000    0.000    0.017    0.017 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
        1    0.016    0.016    0.016    0.016 {method 'join' of 'str' objects}
```

Iterare prin lista

Folosirea operatorului in pentru iterarea printr-o listă

```
import cProfile
import random
lista_nr_random=[random.randint( 1,100) for x in
range(100000000)]

def iterare1():
    suma=0
    for i in range(len(lista_nr_random)):
        suma+=lista_nr_random[i]
    return suma

def iterare2():
    suma=0
    for i, x in enumerate(lista_nr_random):
        suma+=x
    return suma
```

```
def iterare3():
    suma=0
    for x in lista_nr_random:
        suma+=x
    return suma

def suma():
    return sum(lista_nr_random)

cProfile.run('iterare1()')
cProfile.run('iterare2()')
cProfile.run('iterare3()')
cProfile.run('suma()')
```

Iterare prin lista

```
PS D:\ordonate\politehnica\optimizari_python> python test.py
         5 function calls in 5.296 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    5.262    5.262 <string>:1(<module>)
        1    5.262    5.262    5.262    5.262 test.py:12(iterare1)
        1    0.034    0.034    5.296    5.296 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.len}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}


         4 function calls in 5.641 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    5.641    5.641 <string>:1(<module>)
        1    5.641    5.641    5.641    5.641 test.py:18(iterare2)
        1    0.000    0.000    5.641    5.641 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}


         4 function calls in 3.120 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    3.120    3.120 <string>:1(<module>)
        1    3.120    3.120    3.120    3.120 test.py:24(iterare3)
        1    0.000    0.000    3.120    3.120 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}


         5 function calls in 1.455 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    1.455    1.455 <string>:1(<module>)
        1    0.000    0.000    1.455    1.455 test.py:30(suma)
        1    0.000    0.000    1.455    1.455 {built-in method builtins.exec}
        1    1.455    1.455    1.455    1.455 {built-in method builtins.sum}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}


PS D:\ordonate\politehnica\optimizari_python>
```

Observăm că enumerate are cea mai slabă performanță. Are sens să fie folosit doar când avem nevoie să modificăm lista (ca obiect, nu elementele) și când indicele este necesar.

Crearea unei liste (element de element)

Folosirea scrierii stil comprehension este mai rapidă decât alte moduri de creare a listelor

```
import cProfile
import random

lista_nr_random_1=[random.randint(1,100) for x in range(10000000)]

def creare_lista_append():
    l2=[]
    for x in lista_nr_random_1:
        l2.append(2*x)
    return l2

def creare_lista_compreh():
    return [2*x for x in lista_nr_random_1]

def creare_lista_plus():
    l2=[]
    for x in lista_nr_random_1:
        l2+=[2*x]
    return l2
```

```
def creare_lista_extend():
    l2=[]
    for x in lista_nr_random_1:
        l2.extend([2*x])
    return l2

cProfile.run('creare_lista_append()')
cProfile.run('creare_lista_compreh()')
cProfile.run('creare_lista_plus()')
cProfile.run('creare_lista_extend()')
```

Comprehensions (liste)

```
PS D:\ordonate\politehnica\optimizari_python> python test.py
         10000004 function calls in 2.135 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.013    0.013    2.134    2.134 <string>:1(<module>)
        1    1.238    1.238    2.120    2.120 test.py:11(creare_lista_append)
        1    0.001    0.001    2.135    2.135 {built-in method builtins.exec}
 10000000    0.882    0.000    0.882    0.000 {method 'append' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}


         5 function calls in 0.501 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.013    0.013    0.501    0.501 <string>:1(<module>)
        1    0.000    0.000    0.487    0.487 test.py:17(creare_lista_compreh)
        1    0.487    0.487    0.487    0.487 test.py:18(<listcomp>)
        1    0.000    0.000    0.501    0.501 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}


         4 function calls in 0.814 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.014    0.014    0.814    0.814 <string>:1(<module>)
        1    0.801    0.801    0.801    0.801 test.py:20(creare_lista_plus)
        1    0.000    0.000    0.814    0.814 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}


         10000004 function calls in 2.265 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.014    0.014    2.265    2.265 <string>:1(<module>)
        1    1.373    1.373    2.251    2.251 test.py:26(creare_lista_extend)
        1    0.000    0.000    2.265    2.265 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
 10000000    0.878    0.000    0.878    0.000 {method 'extend' of 'list' objects}
```

Operații pe elementele listelor

De exemplu, vrem să facem suma elementelor a două liste, element cu element

```
import cProfile
import random
import numpy

lista_nr_random_1=[random.randint(1,100) for x in range(10000000)]
lista_nr_random_2=[random.randint(1,100) for x in range(10000000)]



def suma_liste_1():
    lsum=[]
    for i in range(len(lista_nr_random_1)):
        lsum.append(lista_nr_random_1[i]+lista_nr_random_2[i])
    return lsum

def suma_liste_2():
    return list(numpy.array(lista_nr_random_1)+numpy.array(lista_nr_random_2))

cProfile.run('suma_liste_1()')
cProfile.run('suma_liste_2()')
```

Operații pe elementele listelor

```
PS D:\ordonate\politehnica\optimizari_python> python test.py
         10000005 function calls in 7.058 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.053    0.053    7.058    7.058 <string>:1(<module>)
        1    4.608    4.608    7.005    7.005 test.py:15(suma_liste_1)
        1    0.000    0.000    7.058    7.058 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.len}
 10000000    2.397    0.000    2.397    0.000 {method 'append' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}


         6 function calls in 4.936 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    1.747    1.747    4.936    4.936 <string>:1(<module>)
        1    0.991    0.991    3.189    3.189 test.py:21(suma_liste_2)
        1    0.000    0.000    4.936    4.936 {built-in method builtins.exec}
        2    2.198    1.099    2.198    1.099 {built-in method numpy.array}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

Ștergere secvență continuă dintr-o listă

Concatenarea părților rămase e mai rapidă decât multiple apeluri pop()

```
import cProfile
import random

l1=[random.randint(1,100) for x in range(1000000)]
l2=list(l1)

def sterge_pop(l,ind1,ind2):
    for _ in range(ind2-ind1):
        l1.pop(ind1)
    return l1

def sterge_concat(l2,ind1,ind2):
    return l2[:ind1]+l2[ind2:]

cProfile.run('sterge_pop(l1,5,len(l1)-5)')
cProfile.run('sterge_concat(l2,5,len(l2)-5)')
```

Ștergere secvență continuă dintr-o listă

```
PS D:\ordonate\politehnica\optimizari_python> python test.py
         99995 function calls in 0.960 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.960    0.960 <string>:1(<module>)
        1    0.025    0.025    0.960    0.960 test.py:16(sterge_pop)
        1    0.000    0.000    0.960    0.960 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.len}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
    99990    0.935    0.000    0.935    0.000 {method 'pop' of 'list' objects}


         5 function calls in 0.000 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.000    0.000 <string>:1(<module>)
        1    0.000    0.000    0.000    0.000 test.py:21(sterge_concat)
        1    0.000    0.000    0.000    0.000 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.len}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

Sortarea elementelor dintr-o listă

Comparație între sort și sorted, cu cheie data ca expresie lambda și ca funcție obișnuită

```
import cProfile
import random


l1=[(random.randint(1,100), random.randint(1,100)) for x in range(10000000)]
l2=list(l1)
l3=list(l1)
l4=list(l1)

def sorteaza1(l):
    l.sort(key=lambda x: x[0]+x[1])
    return l

def sorteaza2(l):
    def f(elem):
        return elem[0]+elem[1]
    l.sort(key=f)
    return l
```

```
def sorteaza3(l):
    sorted(l, key= lambda x:
x[0]+x[1])
    return l

def sorteaza4(l):
    def f(elem):
        return elem[0]+elem[1]
    sorted(l, key=f)
    return l
cProfile.run('sorteaza1(l1)')
cProfile.run('sorteaza2(l2)')
cProfile.run('sorteaza3(l3)')
cProfile.run('sorteaza4(l4)')
```

Sortarea elementelor dintr-o listă (sort)

```
PS D:\ordonate\politehnica\optimizari_python> python test.py
         10000005 function calls in 3.386 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    3.386    3.386 <string>:1(<module>)
        1    0.000    0.000    3.386    3.386 test.py:18(sorteaza1)
 10000000    0.950    0.000    0.950    0.000 test.py:19(<lambda>)
        1    0.000    0.000    3.386    3.386 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
        1    2.435    2.435    3.386    3.386 {method 'sort' of 'list' objects}


         10000005 function calls in 3.335 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    3.335    3.335 <string>:1(<module>)
        1    0.000    0.000    3.335    3.335 test.py:22(sorteaza2)
 10000000    0.944    0.000    0.944    0.000 test.py:23(f)
        1    0.000    0.000    3.335    3.335 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
        1    2.391    2.391    3.335    3.335 {method 'sort' of 'list' objects}
```

Sortarea elementelor dintr-o listă (sorted)

```
         10000005 function calls in 3.760 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    3.760    3.760 <string>:1(<module>)
        1    0.218    0.218    3.760    3.760 test.py:28(sorteaza3)
 10000000    0.953    0.000    0.953    0.000 test.py:29(<lambda>)
        1    0.000    0.000    3.760    3.760 {built-in method builtins.exec}
        1    2.589    2.589    3.542    3.542 {built-in method builtins.sorted}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}


         10000005 function calls in 4.003 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    4.002    4.002 <string>:1(<module>)
        1    0.252    0.252    4.002    4.002 test.py:32(sorteaza4)
 10000000    0.958    0.000    0.958    0.000 test.py:33(f)
        1    0.000    0.000    4.003    4.003 {built-in method builtins.exec}
        1    2.792    2.792    3.750    3.750 {built-in method builtins.sorted}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

Crearea de liste cu elemente distincte

```
import cProfile
import random

l1=[(random.randint(1,100), random.randint(1,100)) for x in range(100000)]
l2=list(l1)
l3=list(l1)


def multime1():
    lrez=[]
    for e in l1:
        if e not in lrez:
            lrez.append(e)
    return lrez
```

```
def multime_set():
    return list(set(l2))

def multime_dictionar():
    return list(dict.fromkeys(l3)


cProfile.run('multime1()')

cProfile.run('multime_set()')
cProfile.run('multime_dictionar()')
```

```
>>> list(set([7,3,2,2,2,3,7]))
[2, 3, 7]
>>> list(dict.fromkeys([7,3,2,2,2,3,7]))
[7, 3, 2]
>>>
```

Crearea de liste cu elemente distincte

```
PS D:\ordonate\politehnica\optimizari_python> python .\creare_multime.py
         10004 function calls in 11.502 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000   11.502   11.502 <string>:1(<module>)
        1   11.501   11.501   11.502   11.502 creare_multime.py:10(multime1)
        1    0.000    0.000   11.502   11.502 {built-in method builtins.exec}
    10000    0.001    0.000    0.001    0.000 {method 'append' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}


         4 function calls in 0.010 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.009    0.009 <string>:1(<module>)
        1    0.009    0.009    0.009    0.009 creare_multime.py:21(multime_set)
        1    0.000    0.000    0.010    0.010 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}


         5 function calls in 0.011 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.011    0.011 <string>:1(<module>)
        1    0.000    0.000    0.011    0.011 creare_multime.py:24(multime_dictionar)
        1    0.000    0.000    0.011    0.011 {built-in method builtins.exec}
        1    0.011    0.011    0.011    0.011 {built-in method fromkeys}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

Crearea dicționarelor

Folosirea lui zip este mai rapidă decât alte moduri de creare a dicționarelor, urmată de comprehensions

```
import cProfile
import random
import math
import string

l1=[random.randint(1,100) for x in
range(int(math.pow(len(string.ascii_lowercase),5)))]
lista_sir_random=["".join([x,y,z,t,w])
                for x in string.ascii_lowercase
                for y in string.ascii_lowercase
                for z in string.ascii_lowercase
                for t in string.ascii_lowercase
                for w in string.ascii_lowercase]


def creeaza_dictionar():
    d1={}
    for i in range(len(l1)):
        d1[lista_sir_random[i]]=l1[i]
    return d1
```

```
def creeaza_dictionar_compreh():
    d1={lista_sir_random[i]:l1[i]  for i in range(len(l1))}
    return d1

def creeaza_dictionar_zip():
    d1=dict(zip(lista_sir_random, l1))
    return d1


cProfile.run('creeaza_dictionar()')
cProfile.run('creeaza_dictionar_compreh()')
cProfile.run('creeaza_dictionar_zip()')
```

Crearea dicționarelor

```
PS D:\ordonate\politehnica\optimizari_python> python test.py
         5 function calls in 8.940 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    2.502    2.502    8.939    8.939 <string>:1(<module>)
        1    6.437    6.437    6.437    6.437 test.py:20(creeaza_dictionar)
        1    0.001    0.001    8.940    8.940 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.len}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}


         6 function calls in 7.424 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.408    0.408    7.424    7.424 <string>:1(<module>)
        1    0.001    0.001    7.016    7.016 test.py:27(creeaza_dictionar_compreh)
        1    7.015    7.015    7.015    7.015 test.py:28(<dictcomp>)
        1    0.000    0.000    7.424    7.424 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {built-in method builtins.len}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}


         4 function calls in 5.929 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.379    0.379    5.929    5.929 <string>:1(<module>)
        1    5.549    5.549    5.549    5.549 test.py:31(creeaza_dictionar_zip)
        1    0.001    0.001    5.929    5.929 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

Iterare printr-un dicționar

Concatenarea părților rămase e mai rapidă decât multiple apeluri pop()

```
import string
import math
import cProfile
import random

l1=[random.randint(1,100) for x in range(int(math.pow(len(string.ascii_lowercase),5)))]
lista_sir_random=["".join([x,y,z,t,w])
                for x in string.ascii_lowercase
                for y in string.ascii_lowercase
                for z in string.ascii_lowercase
                for t in string.ascii_lowercase
                for w in string.ascii_lowercase]


dd=dict(zip(lista_sir_random, l1))

def itereaza1(d):
    sum=0
    for k in d:
        sum+=d[k]
    return sum
```

```
def itereaza2(d):
    sum=0
    for k,v in d.items():
        sum+=v
    return sum

def itereaza3(d):
    sum=0
    for v in d.values():
        sum+=v
    return sum

#strict pe cazul acesta
def suma(d):
    return sum(d.values())

cProfile.run('itereaza1(dd)')
cProfile.run('itereaza2(dd)')
cProfile.run('itereaza3(dd)')
cProfile.run('suma(dd)')
```

Iterare printr-un dicționar

```
PS D:\ordonate\politehnica\optimizari_python> python test.py
         4 function calls in 2.898 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    2.898    2.898 <string>:1(<module>)
        1    2.898    2.898    2.898    2.898 test.py:24(itereaza1)
        1    0.000    0.000    2.898    2.898 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}


         5 function calls in 0.499 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.499    0.499 <string>:1(<module>)
        1    0.499    0.499    0.499    0.499 test.py:30(itereaza2)
        1    0.000    0.000    0.499    0.499 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
        1    0.000    0.000    0.000    0.000 {method 'items' of 'dict' objects}


         5 function calls in 0.431 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.431    0.431 <string>:1(<module>)
        1    0.431    0.431    0.431    0.431 test.py:36(itereaza3)
        1    0.000    0.000    0.431    0.431 {built-in method builtins.exec}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
        1    0.000    0.000    0.000    0.000 {method 'values' of 'dict' objects}


         6 function calls in 0.114 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.114    0.114 <string>:1(<module>)
        1    0.000    0.000    0.114    0.114 test.py:43(suma)
        1    0.000    0.000    0.114    0.114 {built-in method builtins.exec}
        1    0.114    0.114    0.114    0.114 {built-in method builtins.sum}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
        1    0.000    0.000    0.000    0.000 {method 'values' of 'dict' objects}
```

Modulul pickle

Când avem cantități mari de date care trebuie procesate și aduse într-un format diferit folosit mai departe în algoritmi putem să salvăm datele într-un fișier temporar
Datele sunt memorate in format binar.
Se pot încărca din fișier în același format.

```
import pickle

fis = open("date.pkl",'wb')
pickle.dump(date_multe,fis)

f= open("date.pkl",'rb')
date = pickle.load(f)
f.close()
```

Observație: Jupyter notebook

Combinare python cu C

O altă strategie de a îmbunătăți timpul de rulare pentru anumite zone de cod, este să fie scrise niște funcții ajutătoare în C care să fie apelate în Pyhton.
Se realizeaza cu ajutorul intermediului modulului distutils.
Un exemplu de fișier (exemplu_c.py) de setare este cel de mai jos:

```
from distutils.core  import  setup, Extension
setup(name='exemplu_c', version='1.0',  \
      ext_modules=[Extension( 'exemplu_c',
['exemplu_c.c' ])])
```

Funcțiile sunt definite în fișierul exemplu_c.

```
python exemplu_c.py install
```

```
#include <Python.h>

static PyObject* exemplu_c(PyObject* self) {
   return Py_BuildValue("s", "Text exemplu");
}

static char exemplu_c_docs[] =
   "exemplu_c( ): Se apelează fara argumente. Afi□ează
un text\n";

static PyMethodDef functii[] = {
   {"exemplu_c", (PyCFunction)exemplu_c,
      METH_NOARGS, exemplu_c_docs},
      {NULL}
};

void initexemplu_c(void) {
   Py_InitModule3("exemplu_c", functii,"");
}
```