

PYTHON FUNDAMENTALS

Curs interactiv de python

STRUCTURA SEDINTA 5 : MODULE

- ▤ Tema sedinta anterioara
- ▤ Module in Python
- ▤ Crearea unui modul in Python
- ▤ Module standard

Creati 3 clase ce vor reprezenta un catalog auto:

- Clasa1
 - La initializare trebuie sa oferim doi parametri de intrare marca si tip
 - Are o metoda ce accepta parametrul de intrare culoare. De asemenea o metoda numita AfisareCuloare pentru afisarea culorii. Folositi metoda pentru afisare.
- Clasa2 :
 - Mosteneste Clasa1 si avem o metoda care adauga argumentul scaune_incalzite ca parametru de intrare
- Clasa3 :
 - Mosteneste Clasa1 si avem o metoda care adauga argumentul Blocuri_Optice_LED ca parametru de intrare
- Creati un obiect al Clasei 2 (marca = ARO, Tip = M461) si folositi metoda de creare argum. scaune_incalzite cu valoarea <Da>;Creati argumentul culoare cu valoarea <rosu>
- Creati un obiect al Clasei 3 (marca = Dacia, Tip = 1310) si folositi metoda de creare argum. Blocuri_Optice_LED cu valoarea <Nu>; Creati argumentul culoare cu valoarea <negru>
- Afisati pe rand argumentele culoare, Blocuri_Optice_LED, scaune_incalzite marca si tip a **3** obiectelor create

TEMA SEDINTA ANTERIOARA

 Tema_in_clasa_slide66 ✕

```
1 # Program Tema_in_clasa_slide66
2 # Demonstreaza utilizarea obiectelor
3 # Ion Studentul - 1/26/13
4
```

TEMA PAG. 67

Tema in clasa:

Sa se creeze un fisier de tip ini numit Tema_Clasa ce va avea urmatoarele linii:

Linia1- Nume Prenume

Linia2 ini,text sau txt


Linia3 Citire

Creati un program care sa citeasca toate liniile sub forma unei liste si afisati lista.

Adaugati a patra linie la fisier cu textul <<EU SUNT 4>>

Cititi tot fisierul intr-un singur sir de caractere apoi afisati-l.

TEMA SEDINTA ANTERIOARA

 Tema_Clasa_Slide67 ✕

```
1 # Program Citire fisiere
2 # Explica accesarea fisierelor
3 # Ion Studentul -- 1/26/13
4
5 print("\nCitește toate liniile s
```

Creati un program format dintr-o clasa numita Adunare cu doua metode.

Prima metoda este cea de initializare si ia doi parametrii afisand daca se poate suma lor. In caz contrar returneaza <<Nu se poate adunare>>.

A doua metoda este cea de afisare (__str__) care va returna daca se poate produsul celor doua valori initiale. In caz contrar returneaza << Nu se poate inmultire>>.

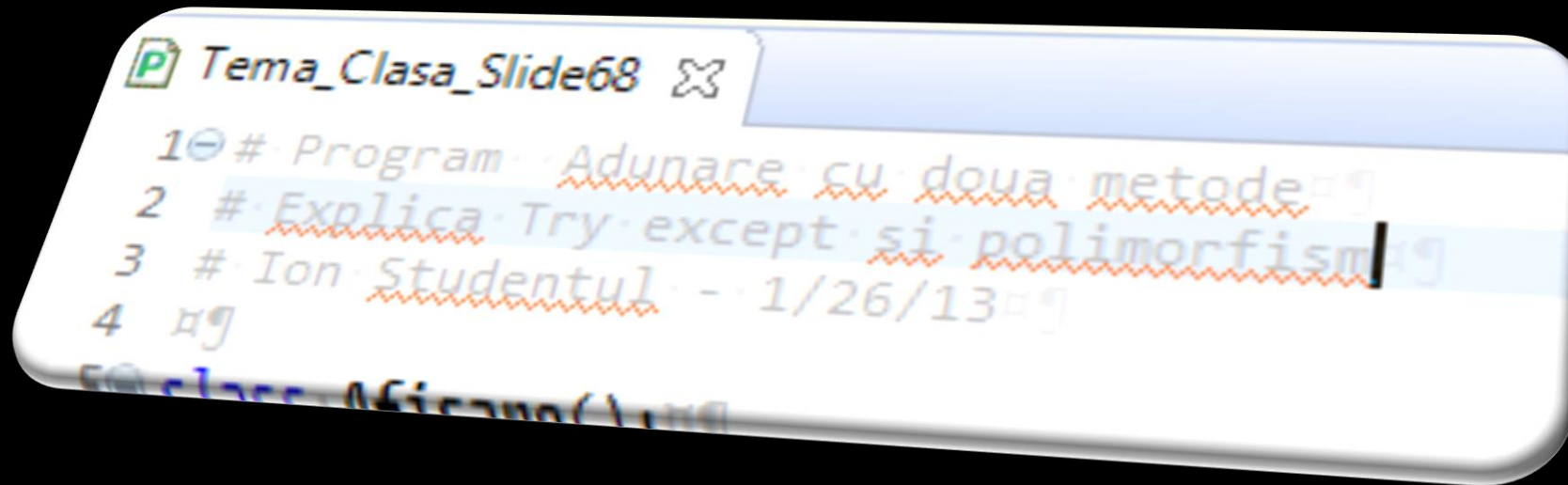
Creati un obiect ce are ca parametrii de intrare <<1>> si <<12>>. Printati obiectul

Creati un obiect ce are ca parametrii de intrare 1 si 12. Printati obiectul

Creati un obiect ce are ca parametrii de intrare <<1>> si 12. Printati obiectul

OBS. Folostiti peste tot try-except-else

TEMA SEDINTA ANTERIOARA

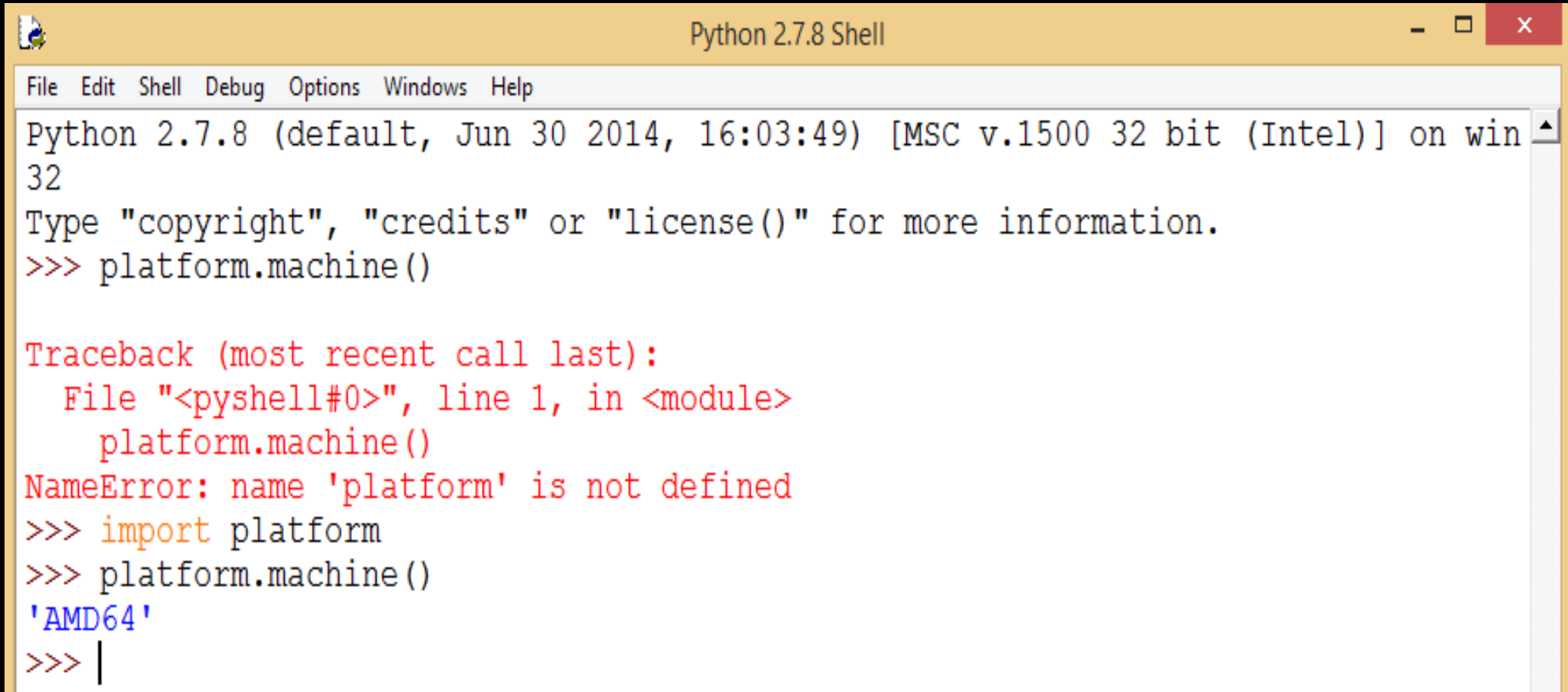


```
Tema_Clasa_Slide68 ✖  
1 # Program - Adunare cu doua metode  
2 # Explica Try-except si polimorfism  
3 # Ion Studentul - 1/26/13  
4  
5 class AfisareClasa
```


MODULE IN PYTHON

- Un modul este un un program care este conceput pt. a fi reutilizat.
- Reutilizarea codului (operatie numita importare modul) se realizeaza prin cuvantul cheie import
- Clasele sau functiile modulului importat nu exista in cod pana la importare

MODULE IN PYTHON



```
Python 2.7.8 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.8 (default, Jun 30 2014, 16:03:49) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> platform.machine()

Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    platform.machine()
NameError: name 'platform' is not defined
>>> import platform
>>> platform.machine()
'AMD64'
>>> |
```

MODULE IN PYTHON

`platform.machine()`

`<<AMD64>>` la masini de 64 biti

`<<i386>>` la masini de 32 biti

`<<armv7l>>` la Android

`<<>>` daca nu poate determina tipul

```
Python 2.7.2 (default, Oct 25 2014, 20:52:15)
```

```
[GCC 4.9 20140827 (prerelease)] on linux2
```

```
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> import platform
```

```
>>> platform.machine()
```

```
'armv7l'
```

```
>>> platform.processor()
```

```
..
```

```
>>> platform.platform()
```

```
'Linux-3.4.5-armv7l-with-libc'
```

```
>>> █
```

platform.system() – returneaza tipul sistemului de operare.

platform.win32_ver() - returneaza tipul de windows, tipul kernel si cum trateaza taskuri multi-Processor

platform.processor() –tipul procesor

platform.node() –nume dispozitiv in retea

platform.platform() - scurta descriere sistem operare.









```
>>> platform.win32_ver()
('8', '6.2.9200', '', u'Multiprocessor Free')
>>> platform.processor()
'Intel64 Family 6 Model 42 Stepping 7, GenuineIntel'
>>> platform.system()
'Windows'
>>> platform.node()
'FamPopescu'
>>> platform.platform()
'Windows-8-6.2.9200'
>>>
```

MODULE IN PYTHON

```
>>> import platform
>>> platform.machine()
'AMD64'
>>> print platform
<module 'platform' from 'C:\Python27\lib\platform.pyc'>
>>> |
```


MODULE IN PYTHON

▶ This PC ▶ Local Disk (C:) ▶ Python27 ▶ Lib ▶

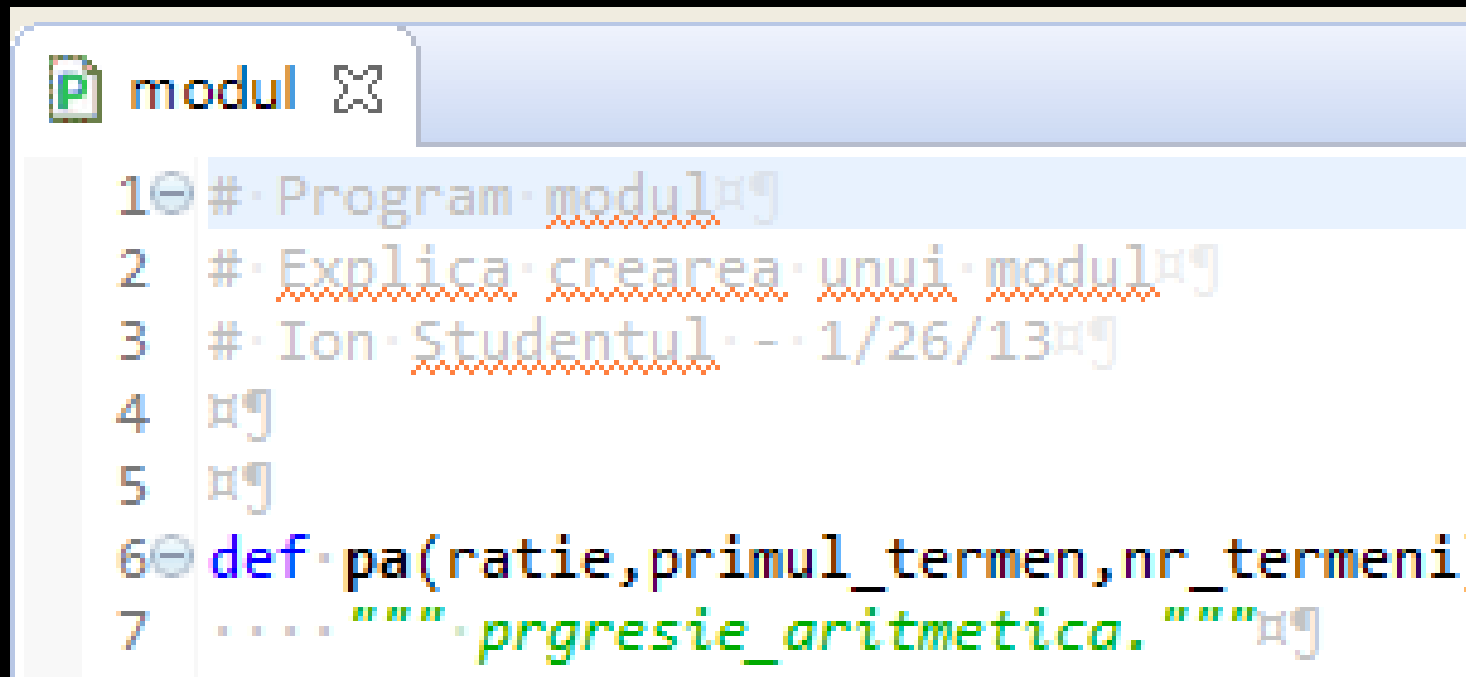
Name	Date modified	Type	Size
 pkgutil.py	4/30/2014 10:54 AM	Python File	20 KB
 platform.py	4/30/2014 10:54 AM	Python File	53 KB
 platform.pyc	1/9/2015 11:17 AM	Compiled Python ...	36 KB
 plistlib.py	4/30/2014 10:54 AM	Python File	15 KB
 popen2.py	4/30/2014 10:54 AM	Python File	9 KB
 poplib.py	4/30/2014 10:54 AM	Python File	13 KB
 posixfile.py	4/30/2014 10:54 AM	Python File	8 KB
 posixpath.py	4/30/2014 10:54 AM	Python File	14 KB

MODULE IN PYTHON

- Pentru a gasi si accesa module Python standard puteti accesa pagina: <https://docs.python.org/2/py-modindex.html>
- Pentru a gasi si accesa module python non-standard puteti accesa pagina: <https://pypi.python.org/>

CREAREA UNUI MODUL IN PYTHON

- Vom crea doua fisiere. Primul fisier este numit “modul.py”



```
1 # Program modul.py
2 # Explica crearea unui modul.py
3 # Ion Studentul -- 1/26/13.py
4
5
6 def pa(ratie, primul_termen, nr_termeni):
7     """ prgresie aritmetica. """
```

CREAREA UNUI MODUL IN PYTHON

- Conditia sitaxei if (`__name__ == __main__`) este adevarata dacă acest program este rulat direct și devine fals dacă este importat ca modul. Prin urmare dacă fisierul `modul.py` este rulat direct vom afisa un mesaj prin care anuntam ca aceasta este un modul și trebuie importat.

```
>>>
```

```
Rulezi acest modul direct, deci nu va rula nimic. Importa acest modul.
```

```
Apasa <enter> pt a iesi.
```

```
>>> |
```

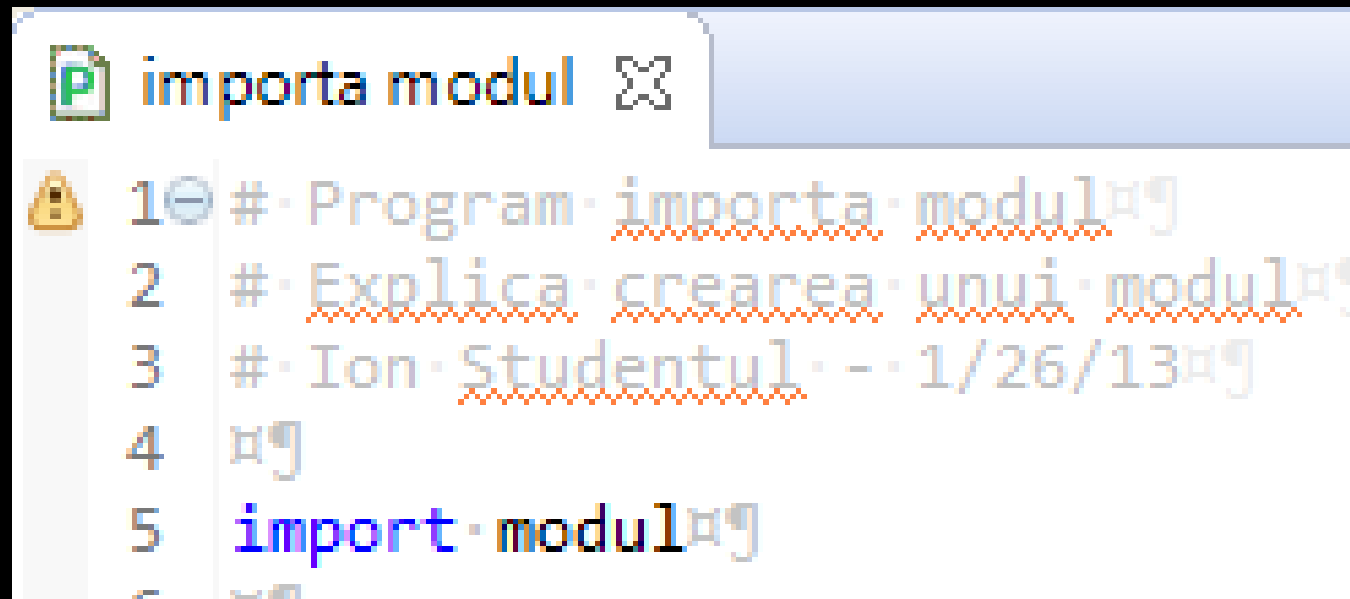
CREAREA UNUI MODUL IN PYTHON

- Formula Progresie aritmetica este (sursa Wikipedia):

$$S_n = \frac{(a_1 + a_n) \cdot n}{2} = \frac{(2 \cdot a_1 + (n - 1) \cdot r) \cdot n}{2} = a_1 \cdot n + r \cdot \frac{n \cdot (n - 1)}{2}$$

CREAREA UNUI MODUL IN PYTHON

- Al doilea fisier importa primul modul. Poate fi numit în orice maniera. Cele doua fisiere trebuie să fie în același director sau în C:\Python27\lib.



```
1 # Program: importa modul  
2 # Explica crearea unui modul  
3 # Ion Studentul - 1/26/13  
4  
5 import modul
```

CREAREA UNUI MODUL IN PYTHON

Dacă dorim să importam 20 de module acestea pot fi importate pe un singur rand cu virgule intre ele:

```
>>>  
>>> import platform, modul  
>>>
```

CREAREA UNUI PACKAGE IN PYTHON

O colectie de module se numeste package.

Pentru a crea un package :

1. Creeza un director ce va avea numele pachetului (package)
2. Pune modulele tale in fisiere separate.
3. Creeza un fisier `__init__.py` in director

RECOMANDARE. TOT MEREU UTILIZATI LITERE MICI PENTRU
NUMELE PACKAGE-ULUI (PENTRU A FI ACCEPTAT PE PYPI)

CREAREA UNUI PACKAGE IN PYTHON: EXEMPLU

structura de fisiere este:

importa_package.py

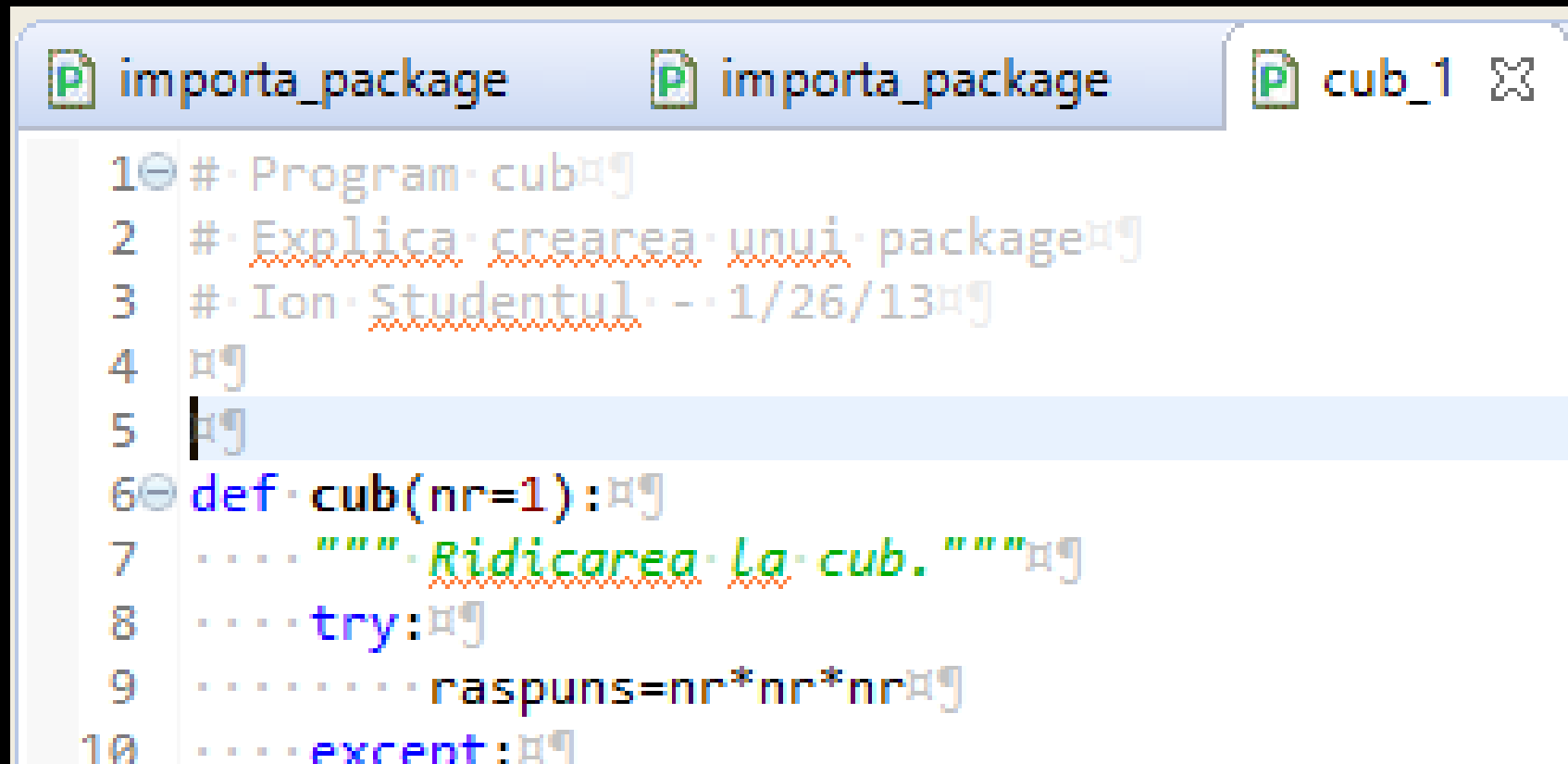
package_test\

 __init__.py

 prog_a.py

 cub_1.py

CREAREA UNUI PACKAGE IN PYTHON: EXEMPLU



```
1 # Program cub
2 # Explica crearea unui package
3 # Ion Studentul -- 1/26/13
4
5
6 def cub(nr=1):
7     """ Ridicarea la cub. """
8     try:
9         raspuns=nr*nr*nr
10    except:
```


CREAREA UNUI PACKAGE IN PYTHON: EXEMPLU

- METODA 1:

```
import package_test.cub_1
```

```
APELARE: package_test.cub_1.cub(3)
```

- METODA 2:

```
from package_test.cub_1 import cub
```

```
APELARE: cub(3)
```

- METODA3:

```
from package_test.cub_1 import cub as CeVreauEu
```

```
APELARE: CeVreauEu(3)
```

MODULELE SYS SI OS

Prin importarea modulului sys și os avem disponibile multe optiuni. Cele mai uzuale sunt:

- `sys.argv` – returneaza o lista de argumente date de utilizator dupa indicarea fisierului de rulare.

Daca vom rula un fisier din consola (command prompt în windows) astfel:

```
C:\Python27\python.exe C:\fisier.py argument1 argument2  
argument3
```

`sys.argv` va deveni o lista ce va avea 4 elemente:

```
["numele_program", "argument1", "argument2", "argument3"]
```

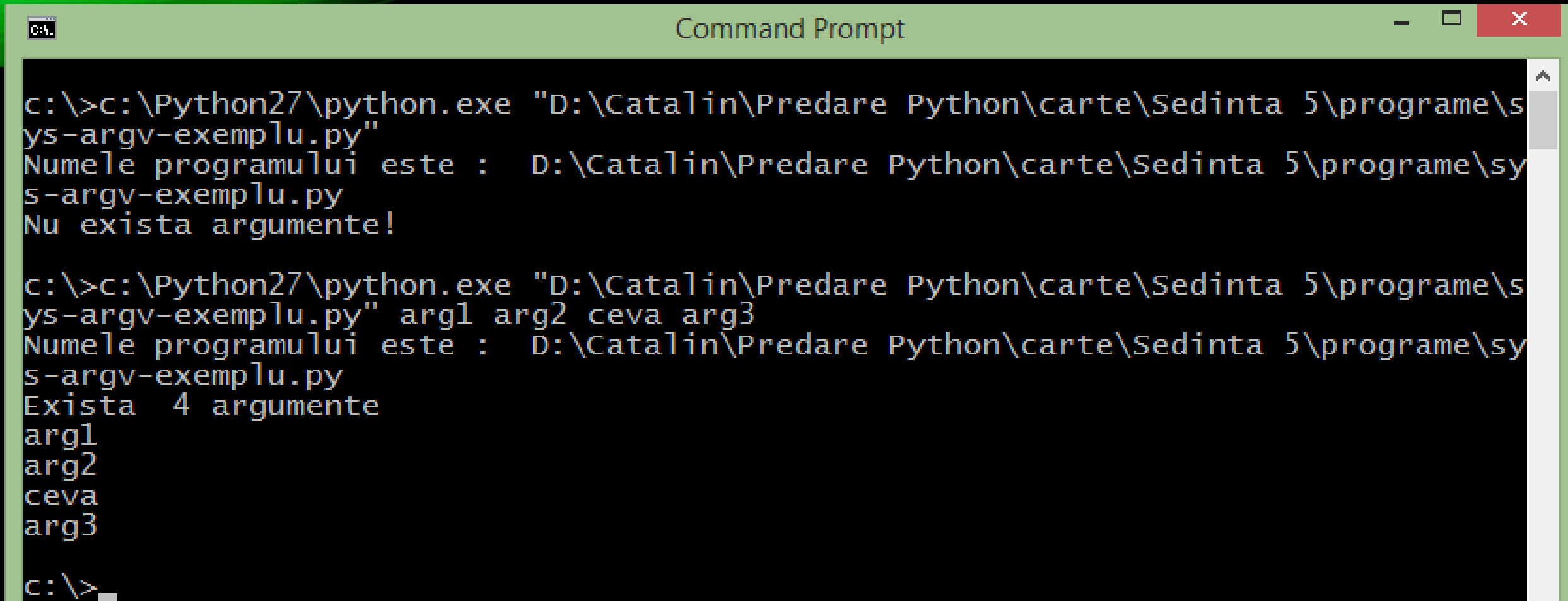
MODULELE SYS SI OS

```
# Fisier: sys-argv-exemplu.py

import sys

print "Numele programului este : ", sys.argv[0]

if len(sys.argv) > 1:
    print "Exista ", len(sys.argv)-1, "argumente"
    for arg in sys.argv[1:]:
        print arg
else:
    print "Nu exista argumente!"
```



```
c:\>c:\Python27\python.exe "D:\Catalin\Predare Python\carte\Sedinta 5\programe\sys-argv-exemplu.py"
Numele programului este : D:\Catalin\Predare Python\carte\Sedinta 5\programe\sys-argv-exemplu.py
Nu exista argumente!

c:\>c:\Python27\python.exe "D:\Catalin\Predare Python\carte\Sedinta 5\programe\sys-argv-exemplu.py" arg1 arg2 ceva arg3
Numele programului este : D:\Catalin\Predare Python\carte\Sedinta 5\programe\sys-argv-exemplu.py
Exista 4 argumente
arg1
arg2
ceva
arg3

c:\>_
```

MODULELE SYS SI OS

- `sys.path` – `sys.path` este o lista de directoare unde Python cauta dupa module. Cand pornesti Python aceasta lista este initializata cu continutul variabilei de mediu a sistemului de operare `PYTHONPATH` și alte cai de acces standardizate

Daca dorim să adaugam la aceasta lista un modul dintr-o cale, am putea adauga aceasta cale cu `sys.path.append("cale")`.

SYS.PATH

```
>>> import modul
```

```
Traceback (most recent call last):
```

```
  File "<pyshell#0>", line 1, in <module>
```

```
    import modul
```

```
ImportError: No module named modul
```

```
>>> import sys
```

```
>>> sys.path
```

```
['', 'C:\\Python27\\Lib\\idlelib', 'C:\\Windows\\system32\\python27.zip', 'C:\\Python27\\DLLs', 'C:\\Python27\\lib', 'C:\\Python27\\lib\\plat-win', 'C:\\Python27\\lib\\lib-tk', 'C:\\Python27', 'C:\\Python27\\lib\\site-packages']
```

```
>>> print "Lungimea listei este ",len(sys.path)
```

```
Lungimea listei este  9
```

```
>>> sys.path.append("c:/Director")
```

```
>>> print "Lungimea listei este ",len(sys.path)
```

```
Lungimea listei este  10
```

```
>>> sys.path
```

```
['', 'C:\\Python27\\Lib\\idlelib', 'C:\\Windows\\system32\\python27.zip', 'C:\\Python27\\DLLs', 'C:\\Python27\\lib', 'C:\\Python27\\lib\\plat-win', 'C:\\Python27\\lib\\lib-tk', 'C:\\Python27', 'C:\\Python27\\lib\\site-packages', 'c:/Director']
```

```
>>> import modul
```

```
>>>
```

```
>>> modul.cub(2)
```

- `sys.platform` - indica sistemul de operare pe care ruleaza programul returnand:

“win32” – windows

“posix” – linux

“linux2” – android

“mac” – mac

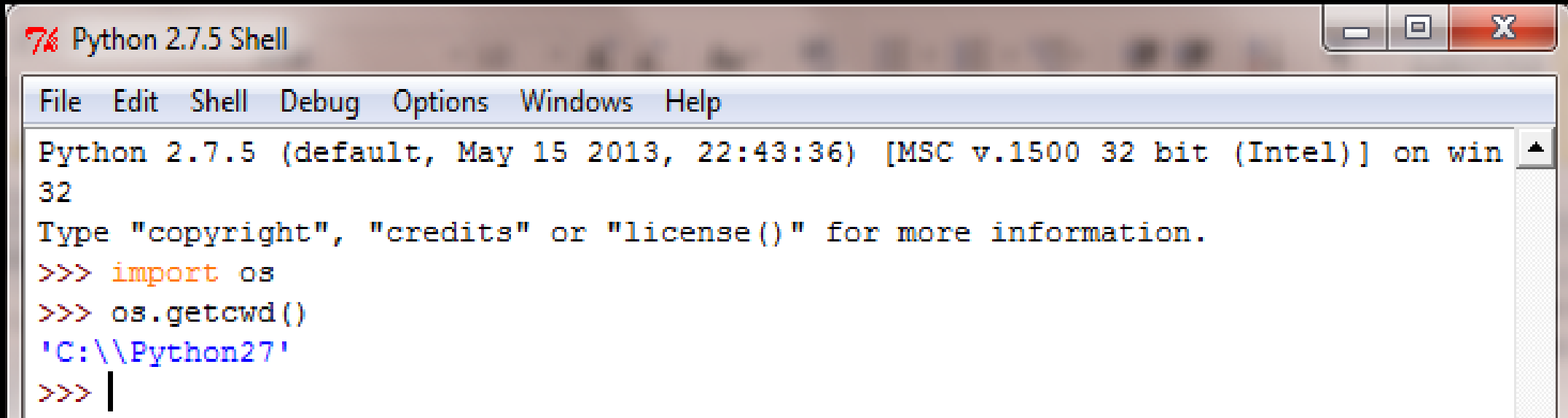
“sunos5” - solaris

```
>>> import sys
>>> if sys.platform == "win32" :
    print "rulam pe un windows de toata frumusetea!!!!"
```

```
rulam pe un windows de toata frumusetea!!!!
```

```
>>> |
```

- `sys.exit()` - are ca scop inchiderea aplicatiei curente
- `os.getcwd()` - directorul curent din care interpretorul ruleaza programul



```
Python 2.7.5 Shell
File Edit Shell Debug Options Windows Help
Python 2.7.5 (default, May 15 2013, 22:43:36) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
>>> import os
>>> os.getcwd()
'C:\\Python27'
>>> |
```

- `os.system()` - ruleaza o comanda în CMD/ Terminal. Observatie! Nu returneaza și outputul! Nu returneaza eroare daca nu exista fisierul sau fisierul ruleaza defectuos.

```
>>> os.system("D:\Catalin\Predare Python\carte\Sedinta 5\programe\sys-argv-exemplu.py")
1
>>> os.system("D:\Catalin\Predare Python\carte\Sedinta 5\programe\sys-argv-exemplu.p")
1
```

- `os.chdir("cale")` - schimba directorul curent

```
>>> import os
>>> os.getcwd()
'C:\\Python27'
>>> os.chdir("D:\Catalin\Predare Python\carte")
>>> os.getcwd()
'D:\\Catalin\\Predare Python\\carte'
```

MODULELE SYS SI OS

- `os.listdir("cale")` - returneaza fisierele dintr-un director

```
>>> os.listdir("c:\Director")  
['modul.py', 'modul.pyc']
```

- `os.name` – similar cu `sys.platform` : Returneaza 'nt' pentru Windows; 'posix' pentru Linux si Android ; 'mac' pentru mac.

```
>>> os.name  
'nt'
```

MODULELE SYS SI OS

- `os.path.exists("cale")` - returneaza True dacă exista calea.
- `os.path.isdir("cale")` - returneaza True dacă calea indica un director.
- `os.path.isfile("cale")` - returneaza True dacă calea indica un fisier.

```
>>> os.path.exists("C:/Director")
True
>>> os.path.isdir("C:/Director")
True
>>> os.path.isfile("C:/Director")
False
```

- `os.linesep` - un șir de caractere care face delimitarea între liniile unui fisier. Pentru linux acest șir de caractere este “\n”, iar pentru windows este “\r\n”. Pentru text files deschise în windows cu functia `open` putem utiliza și direct “\n” pe toate platformele.
- `os.sep` - indica calea prin directoare. Pentru Windows este “\\”, iar pentru linux este “/”. Se poate folosi din Python orice formă și se va converti automat în formă necesara.

```
>>> os.linesep
'\r\n'
>>> os.sep
'\\'
```


- `os.remove("cale_fisier")` - sterge fisierul indicat. Ridica eroare daca nu e fisier sau fisierul este folosit
- `os.rmdir("cale_director")` - sterge directorul indicat. Ridica eroare daca directorul nu este gol sau directorul este folosit.
- `os.removedirs("cale_director")` - sterge directorul indicat recursiv. Ridica eroare daca in calea recursiva exista fisiere (in sbdirectoare).

```
>>> os.listdir("C:\\test\\")
['test.txt']
>>> os.remove("C:\\test\\test.txt")
>>> os.listdir("C:\\test\\")
[]
>>> os.removedirs("C:\\test\\")
>>>
```

MODULUL RANDOM

- `random.choice(lista/dictionar/sir_caractere)`
alege un element aleatoriu
- `random.randrange(start=0,stop=None,step=1)`

`random.randrange(1,6,1)` ar genera un numar aleatoriu cuprins intre 1 și 6 inclusiv.

comanda `random.randrange(6)` ar genera un numar aleatoriu cuprins intre 0 și 5 inclusiv, deoarece functia are start default la 0, noi indicam stop-ul, iar pasul este la 1.

```
>>> x= {1:"111",2:"222"}
>>> random.choice(x)
'111'
>>> random.choice("salutare")
'r'
>>> random.choice(range(200))
17
```

```
>>> import random
>>> random.choice("avem mere frumoase")
'm'
>>> random.randrange(6)
3
```

- Creare si manipularea de fisier temporare

```
>>> import tempfile
>>> tempfile = tempfile.mktemp()

>>> print "tempfile", "=>", tempfile
tempfile => c:\users\popescu\appdata\local\temp\tmpbt1b91
>>>
>>> import os
>>> T_file= open(tempfile,"w")
>>> T_file.write("Test infoacademy\n")
>>> T_file.close()
>>> os.remove(tempfile)
>>> |
```

- cream și deschidem un fisier pentru eventuale adaugari putem utiliza în mod direct functia TemporaryFile()

```
>>> import tempfile
>>> fisier = tempfile.TemporaryFile()
>>> fisier = tempfile.TemporaryFile("w+")
>>> # asta este modul default pentru TemporaryFile => w+
>>>
```

```
>>> fisier.write("test test test")
>>> fisier.close()
>>>
```

- De asemenea dacă doriți să știți ce director este utilizat în momentul de față atunci trebuie să apelați la funcția `gettempdir()` din cadrul modulului `tempfile`. Cum puteți crea un fișier trebuie să aveți posibilitatea de a crea și un director temporar; iar această operațiune este disponibilă cu ajutorul funcției `mkdtemp()`

```
>>> print tempfile.gettempdir()
c:\users\popescu\appdata\local\temp
>>>
>>> tempfile.mkdtemp()
'c:\\users\\popescu\\appdata\\local\\temp\\tmpu1xsfb'
>>> |
```

STOCAREA ÎN FISIERE CU CPICKLE

- Stocarea datelor în fișiere de tip text este convenabilă dar totul se face manual. cPickle automatizează totul.
- În Python se numește Pickling procesul de conversie a datelor complexe și de mai multe tipuri cu scopul de a fi stocate în fișiere, dar în alte limbaje de programare acest proces se numește serialization sau marshaling
- Există două variante de Pickle. Prima variantă este Pickle, iar cea a doua este cPickle. Varianta cPickle este scrisă în C, fiind un pic mai rapidă decât cea scrisă în Python. Din acest motiv se regăsește doar varianta cPickle ca modul standard.

STOCAREA ÎN FISIERE CU CPICKLE

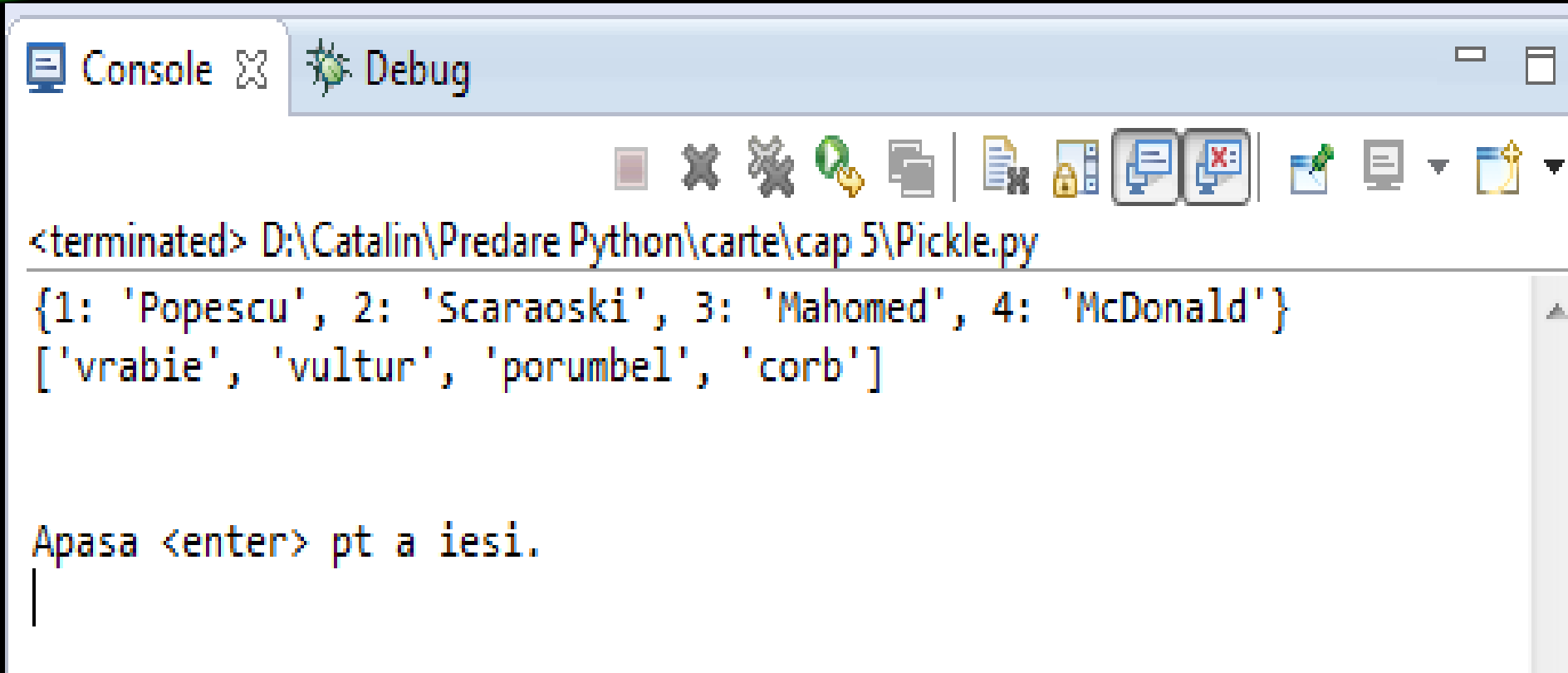
- Pentru a scrie o variabila vom utiliza `cPickle.dump(variabila, nume fisier)`, iar pentru a citi o variabila vom utiliza `cPickle.load(nume fisier)`.
- dacă avem doar doua variabile salvate în fisier și încercăm să apelăm `load` a treia oara, atunci va da eroare.
- variabilele sunt adaugate și citite incremental (adica de la prima la ultima).

STOCAREA ÎN FISIERE CU CPICKLE

Pickle

```
1 # Program Pickle
2 # Explica utilizarea modulului
3 # Ion Studentul - 1/26/13
4
5 import cPickle
```

STOCAREA ÎN FISIERE CU CPICKLE

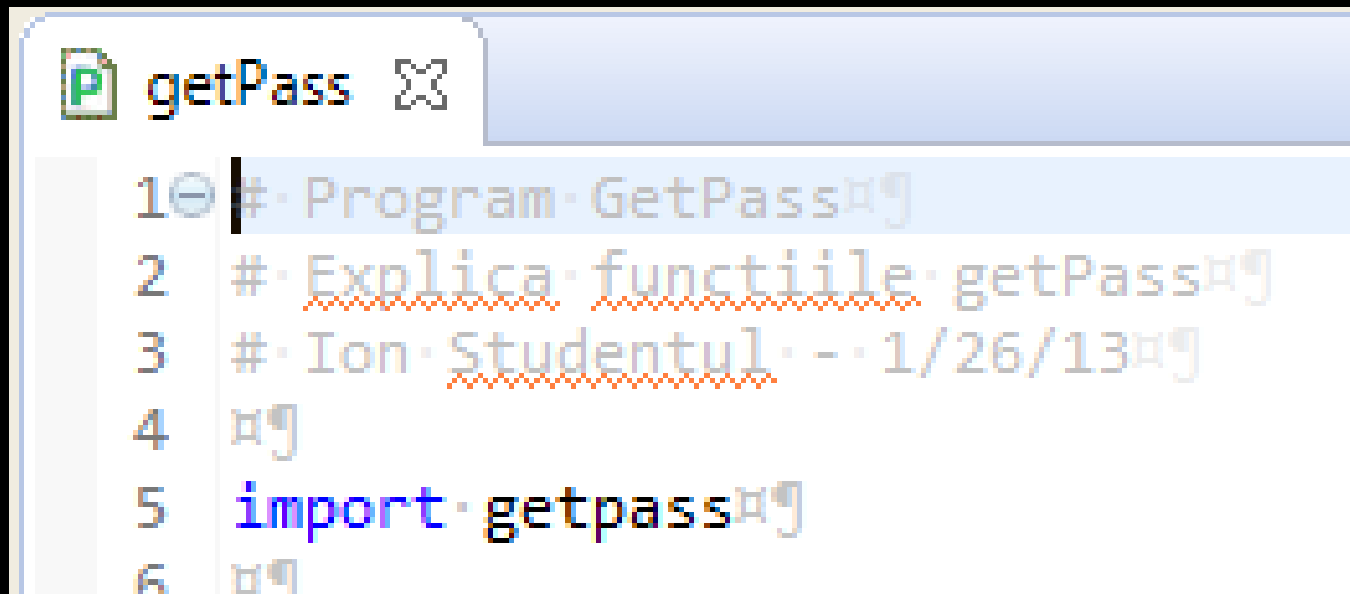


The screenshot shows a Python IDE window with a 'Console' tab selected. The console displays the output of a script that has terminated. The output shows a dictionary and a list of strings. Below the output, there is a prompt for the user to press the enter key to exit the console.

```
<terminated> D:\Catalin\Predare Python\carte\cap 5\Pickle.py  
{1: 'Popescu', 2: 'Scaraoski', 3: 'Mahomed', 4: 'McDonald'}  
['vrabie', 'vultur', 'porumbel', 'corb']  
  
Apasa <enter> pt a iesi.  
|
```

MODULUL GETPASS

- `getpass.getuser()` - returneaza userul curent ce s-a logat la masina
- `Getpass.getpass()` - poate fi folosita doar în cadrul consolei deoarece permite să nu se vada parola scrisa în consola(rulare prin dublu click a fisierului python).



```
1 # Program GetPass
2 # Explica functiile getPass
3 # Ion Studentul - 1/26/13
4
5 import getpass
6
```

MODULUL MATH

```
>>> var1 = 1j
>>> var2 = 3+3j
>>> var3 = 3
>>> type(var1)
<type 'complex'>
>>> type(var2)
<type 'complex'>
>>> type(var3)
<type 'int'>
>>>
```

- Dupa importarea modulului math putem utiliza:

```
>>> print "numarul natural e: ",math.e
numarul natural e:  2.71828182846
>>> print "numarul pi:", math.pi
numarul pi: 3.14159265359
>>> print "2 la puterea a treia este:", math.pow(2,3)
2 la puterea a treia este: 8.0
>>> print "radical din 8 este:",math.sqrt(8)
radical din 8 este: 2.82842712475
>>> print "radical din 9 este:",math.sqrt(9)
radical din 9 este: 3.0
>>> |
```

- Intervalele de timp sunt numere ce sunt flotante (curgatoare) în secunde. Anumite instante de timp sunt exprimate în secunde de la 12:00am ,1 Ianuarie 1970 (numit pe scurt epoch). Exista un modul foarte utilizat numit time ce ofera functii pentru lucrul cu timpul și pentru a converi timpul.

```
>>> import time
>>>
>>>
>>> # Aceasta comanda va afisa timpul epoch
>>> print time.time()
1403297810.48
```

- Multe functii ale modulului Python time sunt reprezentate de un tuplu de 9 numere. Poarta numele de **struct_time**

Ind ex	Camp	Valori	Atribut
0	Anul	2008	tm_year
1	Luna	1 to 12	tm_mon
2	Ziua	1 to 31	tm_mday
3	Ora	0 to 23	tm_hour
4	Minutul	0 to 59	tm_min
5	Secunda	0 to 61 (60 or 61 are leap-seconds)	tm_sec
6	Ziua saptamanii	0 to 6 (0 is Monday)	tm_wday
7	Ziua anului	1 to 366 (Julian day)	tm_yday
8	Ora de Vara	-1, 0, 1, -1 means library determines DST	tm_isdst

- Pentru a obtine **struct_time** :

```
>>> import time
>>> time.localtime()
time.struct_time(tm_year=2015, tm_mon=1, tm_mday=9, tm_hour=15, t
m_min=47, tm_sec=5, tm_wday=4, tm_yday=9, tm_isdst=0)
>>>
```

- Exista metode de a formata timpul din epoch la data curenta, dar cea mai simpla este cea integrata în modulul time : `time.asctime()`..

```
>>> import time
>>> localtime = time.asctime( time.localtime(time.time()) )
>>>
>>>
>>> print localtime
Sat Jun 21 00:35:38 2014
>>>
>>> print time.localtime(time.time())
time.struct_time(tm_year=2014, tm_mon=6, tm_mday=21, tm_hour=0, tm_min=36, tm_sec=15, tm_wday=5, tm_yday=172, tm_isdst=1)
>>> time.localtime()
time.struct_time(tm_year=2014, tm_mon=6, tm_mday=21, tm_hour=0, tm_min=36, tm_sec=36, tm_wday=5, tm_yday=172, tm_isdst=1)
>>> time.asctime( time.localtime())
'Sat Jun 21 00:37:46 2014'
```

- Exista metode de a formata timpul din epoch la data curenta, dar cea mai simpla este cea integrata în modulul time : `time.asctime()`

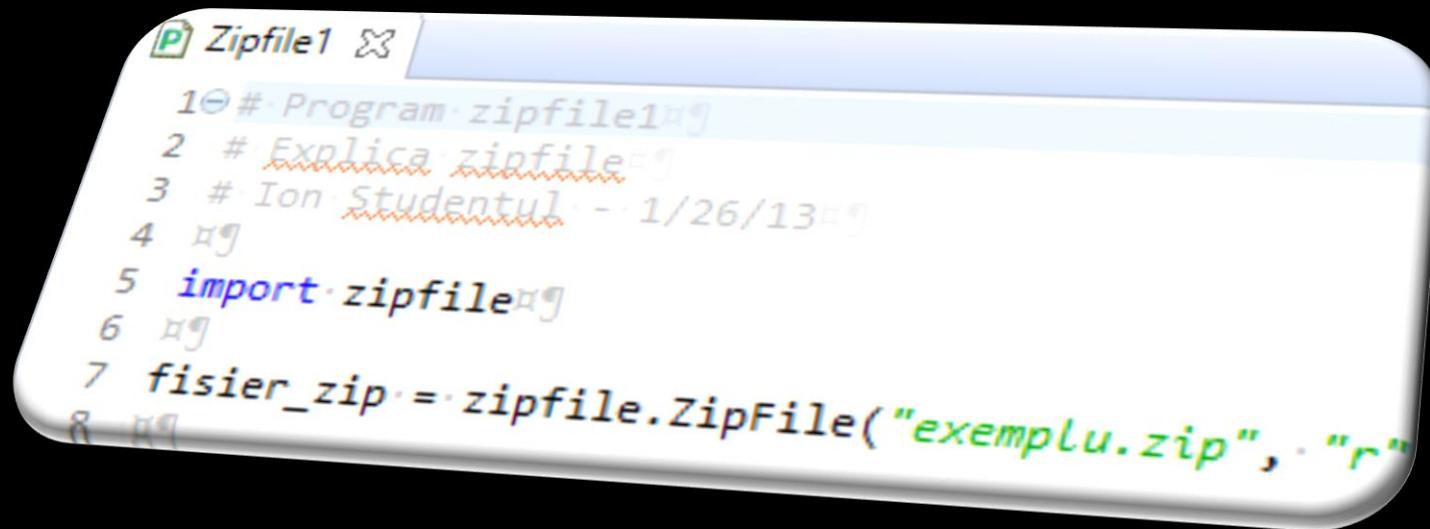
```
>>> time.asctime()  
'Fri Jan 09 15:50:55 2015'  
>>>
```

```
>>> import calendar
>>>
>>> Decembrie= calendar.month(2015,12)
>>> print Decembrie
    December 2015
Mo Tu We Th Fr Sa Su
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

```
>>>  
>>> zi = calendar.weekday(2020,11,24)  
>>> print "Ziua 24 din luna 11 a anului 2020 este a ",zi+1," a saptamanii"  
Ziua 24 din luna 11 a anului 2020 este a  2  a saptamanii  
>>> #se pleaca de la 0 cu luni iar 6 reprezinta duminica  
>>>
```

ARHIVAREA ȘI DEZARHIVAREA DE FISIERE: ZIPFILE

- Citirea de informații ale unei arhive:



```
1 # Program: zipfile1.py
2 # Explica: zipfile.py
3 # Ion Studentul - 1/26/13
4
5 import zipfile
6
7 fisier_zip = zipfile.ZipFile("exemplu.zip", "r")
8
```

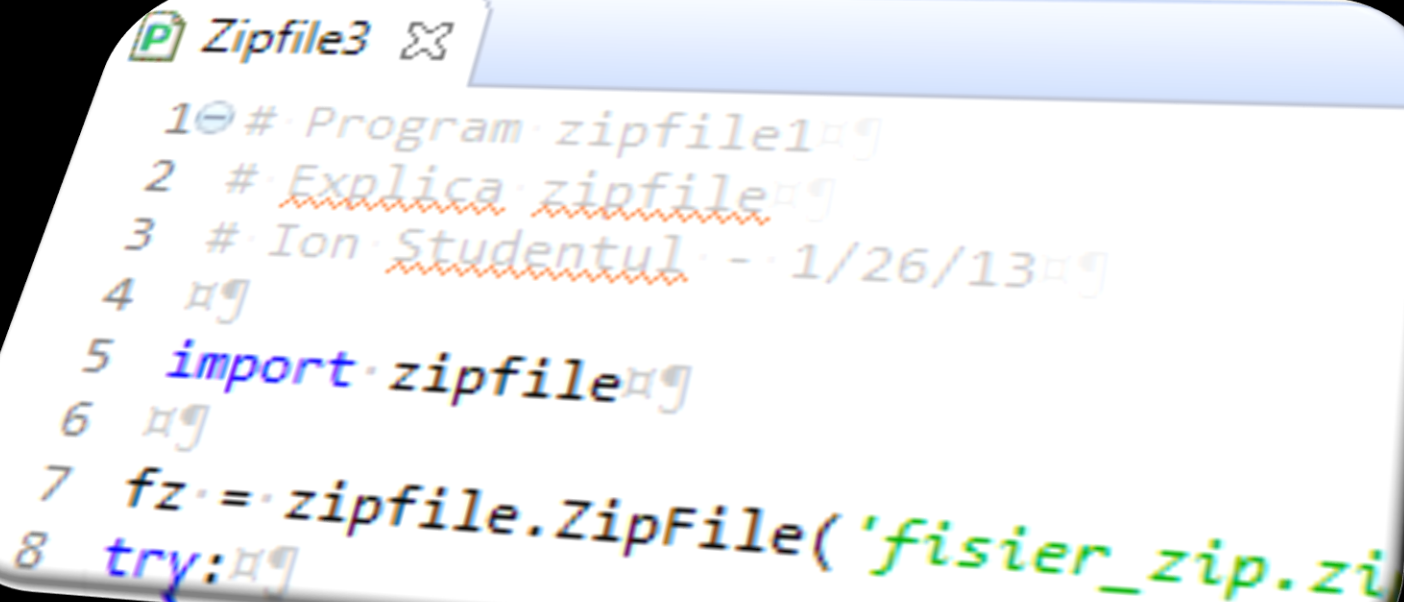
ARHIVAREA ȘI DEZARHIVAREA DE FISIERE: ZIPFILE

- Extragerea de fișiere ale unei arhive:

```
Zipfile2 ✖  
1 # Program zipfile2  
2 # Explica zipfile  
3 # Ion Studentul -- 1/26/13  
4  
5 import zipfile  
6  
7 fisier_zip = zipfile.ZipFile("exemplu.zip", "r")  
8
```


ARHIVAREA ȘI DEZARHIVAREA DE FISIERE: ZIPFILE

- Scrierea de fișiere într-o arhivă:



```
1 # Program zipfile1
2 # Explica zipfile
3 # Ion Studentul -- 1/26/13
4
5 import zipfile
6
7 fz = zipfile.ZipFile('fisier_zip.zip')
8 try:
```

ARHIVAREA ȘI DEZARHIVAREA DE FISIERE:ZIPFILE

- Dacă comparăm dimensiunea fișierelor comprimate cu dimensiunile fișierelor ce formează fișierul zip putem vedea că dimensiunile sunt egale. Dacă dorim să adăugăm și o modalitate de compresie trebuie să adăugăm la crearea obiectului zip expresia `compression=zipfile.ZIP_DEFLATED`. `ZIP_DEFLATED` are nevoie de modulul `zlib`, modul ce este instalat standard. Dacă nu specifici nimic, standard consideră că metoda de compresie este `zipfile.ZIP_STORED` care doar stochează fișierele fără compresie.

ARHIVAREA ȘI DEZARHIVAREA DE FISIERE: ZIPFILE

```
Zipfile4  ⌵  
16  ⌵  
17  for info in fz_necompresat.infolist():  
18      ...print info.filename, "Marime fisier"  
19  ⌵  
20  print '\n\tCream o arhiva compresata'  
21  fz = zipfile.ZipFile('fisier_zip.zip',  
22  try:⌵  
23      ...print '\tAdauga fisiere'⌵  
24      ...fz.write('setup.py')⌵  
25      ...fz.write('Python_ico.ico')⌵  
26  finally:⌵  
27      ...print '\tInchiderea arhivei'⌵  
28      ...fz.close()
```

- Tema :

Creati un modul ce sa contina o functie numita Adunare ce va primi doi parametri si ii va aduna daca poate (try-else) returnand rezultatul. In cazul in care nu poate sa adune/concateneze cele doua elemente sa returneze un text explicativ. Adaugati un if `__name__ == "__main__"` cu un text explicativ la rularea directa.

Creati un fisier care sa importe sys si os.

In cazul in care sistemul de operare este windows (sys.platform) sa importe modulul altfel sa returneze un mesaj si iasa din program (if-else)

Acest program trebuie sa afiseze calea curenta apoi sa o schimbe calea curenta la calea "C:\Program Files". Confirmati schimbarea. Rulati apoi functia Adunare pentru "2" si "3" (sir de caractere).

- Tema:

Creati un program care sa aiba o functie numita dataAleatoriu ce returneaza o lista din trei elemente. Primul element al listei sa fie un an aleatoriu intre 2000 si 2020. Al doilea element al listei sa fie o luna aleatorie returnata ca numar intre 1 si 12. Al treilea element al listei sa fie o zi intre 1 si 28.

Apelati de trei ori functia data si stocati rezultatul intr-o variabila. Aflati ziua din saptamana a datelor returnate.

VA MULTUMESC PENTRU PARTICIPARE

La revedere!