

PYTHON FUNDAMENTALS

Curs interactiv de python

STRUCTURA SEDINTA 4 : CLASE

- ▤ Tema sedinta anterioara

- ▤ Clase avansate

- ▤ Exceptii

- ▤ Lucrul cu fisiere

CUNOSTINTE EXTRA LEGATE DE LISTE SI DICTIONARE

Creati un program ce imparte sirul de caractere “Avem ce face in Python” intr-o lista. Foloseste lista pentru a prelucra caracterele din interior astfel:

- Inlocuieste e cu i
- Inlocuieste spatiu cu _ (underline)

Uneste apoi lista intr-un sir de caractere ce trebuie afisata

CUNOSTINTE EXTRA LEGATE DE LISTE SI DICTIONARE

```
1 # Tema Sir_lista
2 # manipularea sir de caractere -- lista
3 # Ion Studentul vers 1
4
5 #Creati un program ce imparte sirul de caractere "Avem ce face in
6 #Foloseste lista pentru a prelucra caracterele din interior astfel
7 # --Inlocuieste e cu i
8 # --Inlocuieste spatiu cu _ (underline)
9 #Uneste apoi lista intr-un sir de caractere ce trebuie afisata
10
```

TEMA SEDINTA ANTERIOARA

Creati un dictionar cu elementele:

1-"Unu"

2-"Doi Doi"

3-"Trei Trei Trei"

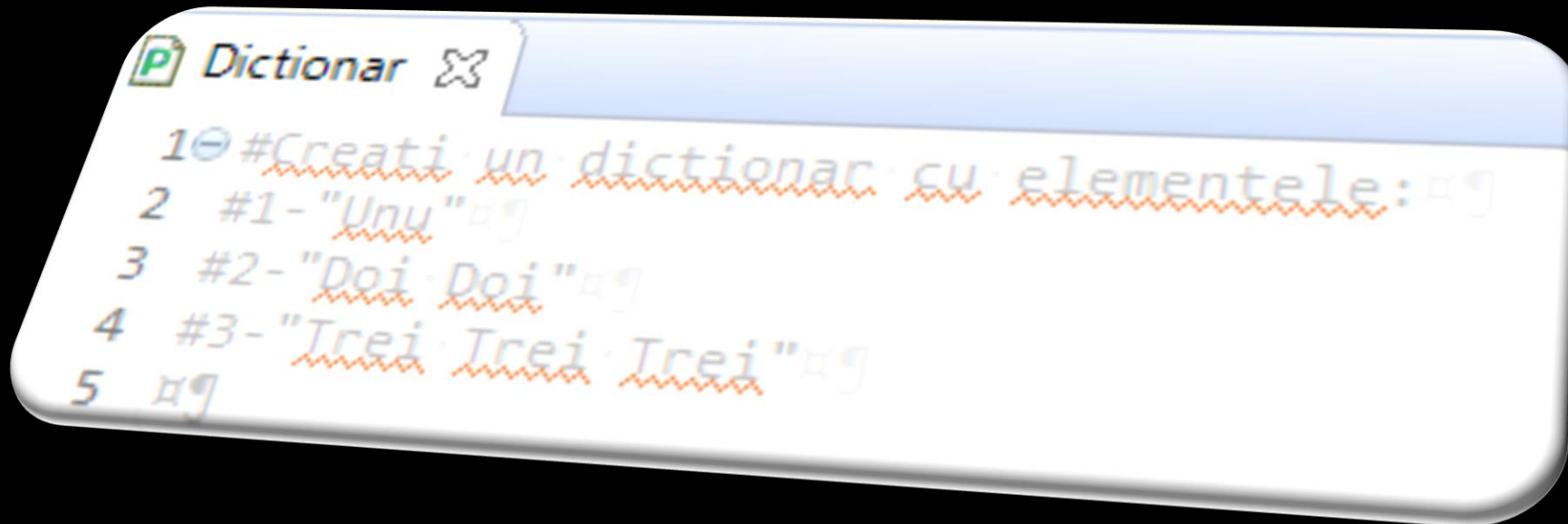
Printati fiecare pereche de tip cheie valoare din dictionar printr-un for.

Solicitati userului sa introduca text de la tastatura. Acest text devine valoarea unei noi intrari in dictionar ce va avea cheia 4.

Printati fiecare pereche de tip cheie valoare din dictionar printr-un for.

Stergeti intrarea ce are cheia cu numarul 2. Printati fiecare pereche de tip cheie valoare din dictionar printr-un for.

TEMA SEDINTA ANTERIOARA



```
1 #Creati un dictionar cu elementele:
2 #1- "Unu"
3 #2- "Doi Doi"
4 #3- "Trei Trei Trei"
5
```


FUNCTII

Exercitiu

- Creati o functie care sa calculeze si sa returneze operatia matematica de mai jos exclusiv pentru 3 numere. Daca nu sunt numere returnati sirul de caractere "Dati va rog 3 numere":

$$[a(a+3)/b]*c$$

Rulati functia pentru 1,2,3 si afisati rezultatul

Rulati functia pentru "1","2","3" si afisati rezultatul

FUNCTII

```
funcție exercitiu1_slide60 ✕  
1 # Testează cunoștințe funcții  
2 # Cum realizăm o funcție  
3 # Ion Studentul  
4  
5 # Creați o funcție care să calculeze și să  
6 # de mai jos exclusiv pentru 3 numere.  
7 # Dacă nu sunt numere returnați sirul de  
8 # ..... [a(a+3)]
```

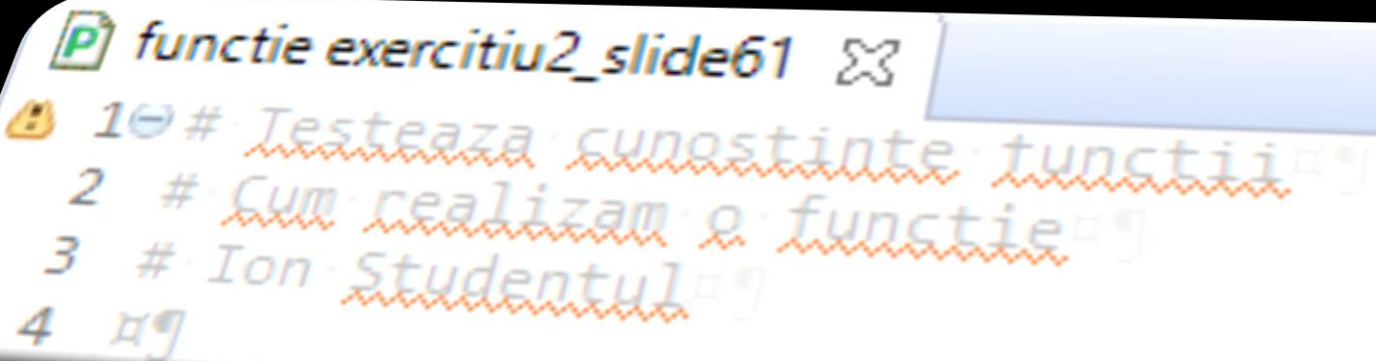

FUNCTII

Exercitiu

- Creati o functie care sa inlocuiasca caracterele de tip spatiu cu underline si sa inlocuiasca sirul de caractere "e" cu sirul de caractere "i" dintr-un sir de caractere primit ca parametru. Verificati ca este sir de caractere inainte de a face aceasta operatie. Returnati sirul de caractere modificat sau returnati sirul urmator daca nu a fost primit un sir de caractere:
"Dati va rog un sir de caractere"

Rulati codul pentru "Merele, perele si pestele nu au E-uri"

FUNCTII





```
functie exercitiu2_slide61 ✕  
1 # Testeaza cunostinte functii  
2 # Cum realizam o functie  
3 # Ion Studentul  
4
```

Creati o clasa ce va reprezenta un catalog:

- La initializare trebuie sa oferim doi parametrii de intrare nume si prenume
- Avem o metoda care afiseaza absente implementat cu `__str__`
- Avem o metoda care incrementeaza cu 1 nr. de absente
- Avem o metoda care sterge un nr. (exclusiv un numar - verifica) de absente dat (pentru cazurile in care avem o scutire medical) fara a deveni negativ
- Creati 1 student numit Ion Roata
- Modificati argumentul absente sa fie incrementat de 3 ori prin metoda creata
- Stergeti doua absente prin metoda specificata
- Creati al doilea student numit George Cerc
- Modificati argumentul absente sa fie incrementat de 4 ori prin metoda creata
- Stergeti doua absente prin metoda specificata
- Afisati absentele fiecarui student

INFOACADEMY.NET

CLASA IN PYTHON

 `clasa_exercitiu_slide 62` 

```
1 # Program Tema Sedinta3
2 # Demonstreaza utilizarea obiectelor
3 # Ion Studentul -- 1/26/13
4
```

METODE STATICE

O metoda statica este o variabila creata in exteriorul metodelor; deci structura este ca cea din imagine.

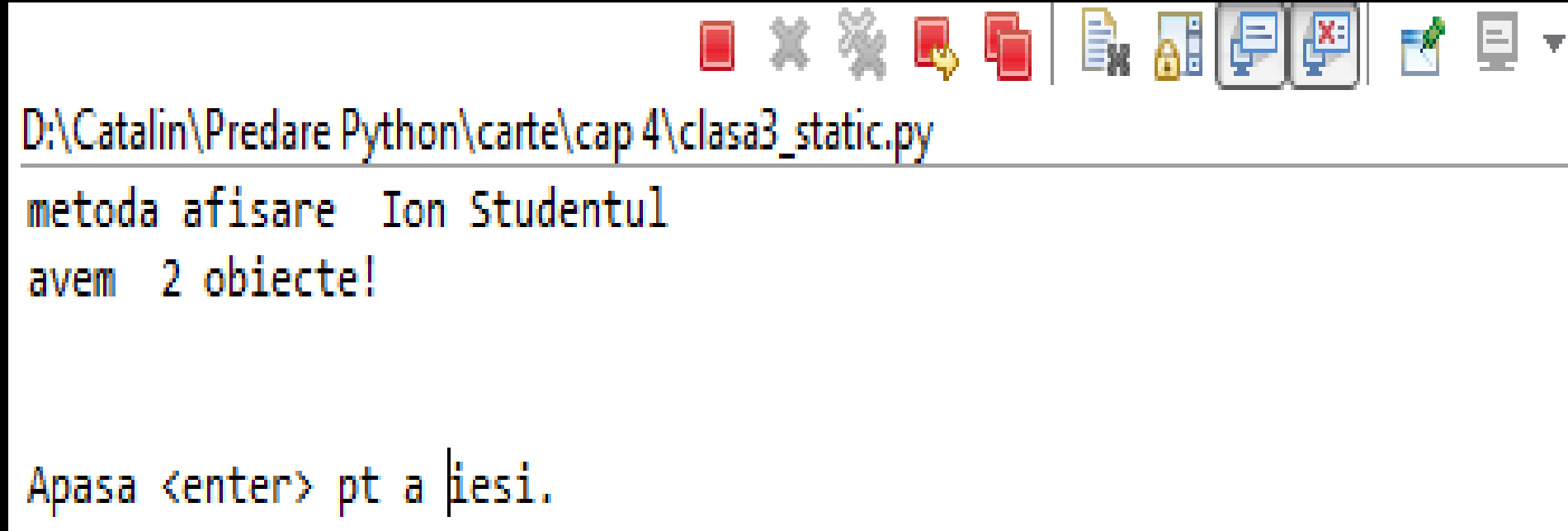
Metoda este foarte utila când avem nevoie de o referință fata de obiectele create.

```
>>> class NumeClasa(object):  
        """docstring"""  
        metoda_statica_1 = 0  
  
>>>
```

METODE STATICE

```
clasa3_static ✖  
1 # Program clasa3  
2 # Demonstreaza utilizarea static attributes  
3 # Ion Studentul -- 1/26/13  
4  
5 class NumeClasa(object):  
6     """clasa mea"""  
7     total=0
```


METODE STATICE



A screenshot of a Windows command prompt window. The title bar shows the file path `D:\Catalin\Predare Python\carte\cap 4\clasa3_static.py`. The command prompt displays the output of a Python script: `metoda afisare Ion Studentul` and `avem 2 obiecte!`. Below the output, the text `Apasa <enter> pt a iesi.` is shown, indicating the user should press Enter to exit the program. The taskbar at the bottom shows several open applications, including a web browser and a file explorer.

```
D:\Catalin\Predare Python\carte\cap 4\clasa3_static.py  
metoda afisare Ion Studentul  
avem 2 obiecte!  
  
Apasa <enter> pt a iesi.
```

METODE STATICE

Se poate vedea ca programul de mai sus definește un atribut în afara unei metode. Acest atribut poate fi accesat utilizand numele clasei prin apelarea `NumeClasa.total`(unde `total` este metoda statica definita in corpul clasei). Astfel programul propune numărarea obiectelor create.

Aceste attribute “statice” rămân neschimbate pe toata durata rularii, nu contează cate obiecte cream. Se poate vedea ca acest număr se incrementează cu fiecare obiect creat în metoda de inițializare `__init__` (`NumeClasa.total+=1`). Apoi de fiecare data când apelam metoda de afișare, programul ne va returna numărul stocat de metoda statica.

Am putea să cream pentru fiecare obiect acele attribute (si să le definim cu `self` in fata?

METODE STATICE

Am putea să cream pentru fiecare obiect acele attribute (și să le definim cu self. în față), dar ar deveni unice la nivel de obiect deoarece pentru fiecare obiect se va crea câte un atribut. Prin urmare nu am avea un atribut consistent peste toată clasa și nu am putea determina numărul de instanțe ale clasei (obiecte).

Acest lucru este testat și în cele ce urmează.

METODE STATICE

```
>>> class NumeClasa(object):  
    """clasa mea"""  
    total=0  
    def __init__(self,nume):  
        """nume student"""  
        self.nume= nume  
        total+=1
```

```
>>> x=NumeClasa("Cata")
```

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

```
  File "<pyshell#68>", line 1, in <module>
```

```
    x=NumeClasa("Cata")
```

```
  File "<pyshell#67>", line 7, in __init__
```

```
    total+=1
```

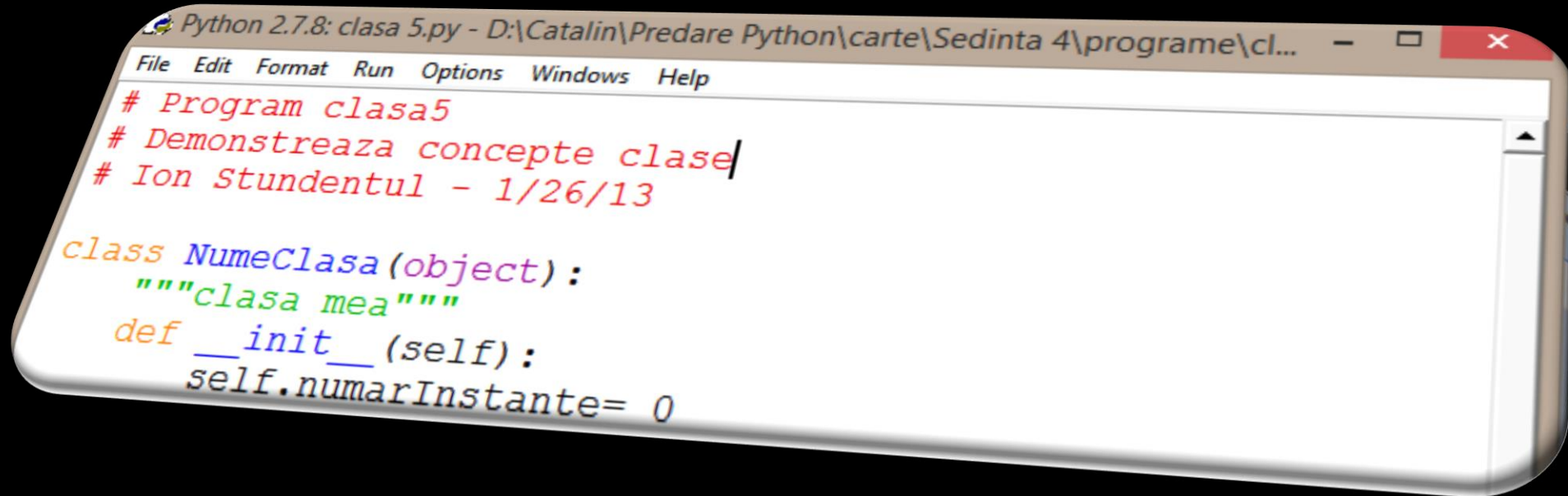
```
UnboundLocalError: local variable 'total' referenced before assignment
```

METODE STATICE

Apelarea unei metode statice se poate realiza si prin intermediul obiectelor, avand acelasi rezultat.

```
>>> class NumeClasa(object):  
    """clasa mea"""  
    total=0  
    def __init__(self,nume):  
        """nume student"""  
        self.nume= nume  
        NumeClasa.total+=1  
  
>>> NumeClasa.total  
0  
>>> x =NumeClasa("Ion")  
>>> NumeClasa.total  
1  
>>> x.total  
1  
>>> y=NumeClasa("Maria")  
>>> NumeClasa.total  
2  
>>> y.total  
2  
>>> x.total  
2  
>>>
```

METODE STATICE



A screenshot of a Python 2.7.8 IDE window. The title bar reads "Python 2.7.8: clasa 5.py - D:\Catalin\Predare Python\carte\Sedinta 4\programe\cl...". The menu bar includes "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The code editor contains the following Python code:

```
# Program clasa5
# Demonstreaza concepte clase
# Ion Studentul - 1/26/13

class NumeClasa(object):
    """clasa mea"""
    def __init__(self):
        self.numarInstante= 0
```


METODE STATICE

```
Console History
D:\Catalin\Predare Python\carte\Sedinta 4\Programe in clasa Sedinta 4\clasa 5.py
cream 4 obiecte!
vizualizam argumentul numarInstante!
0
0
0
0
modificam argumentul numarInstante!
vizualizam argumentul numarInstante modificat!
0
1
2
30

Apasa <enter> pt a iesi.
```

METODE STATICE

În programul de mai sus vedem că toate atributele `numarInstante` vor lua valoarea 0 la instantiere. Dacă dorim să modificăm acest atribut acesta va avea valori diferite pentru fiecare obiect în parte. Rețineți, fiecare obiect are un set unic de proprietăți (atribute). Prin urmare nu vom avea o referință față de obiectele create.

Dacă modificăm programul de mai sus pentru a solicita metoda statică `NumeClasa.numarInstante` aceasta va genera eroare deoarece nu a fost definită.

```

28
29 print("vizualizam argumentul numarInstante modificat!")
30 print(object1.numarInstante)
31 print(object2.numarInstante)
32 print(object3.numarInstante)
33 print(object4.numarInstante)
34 print(NumeClasa.numarInstante)
35
36 raw_input("\n\nApasa <enter> pt a iesi.")
37

```

Console History

<terminated> D:\Catalin\Predare Python\carte\Sedinta 4\Programe in clasa Sedinta 4\clasa 5.py

0

1

2

30

Traceback (most recent call last):

File "D:\Catalin\Predare Python\carte\Sedinta 4\Programe in clasa Sedinta 4\clasa 5.py", line 34, in <module>

print NumeClasa.numarInstante

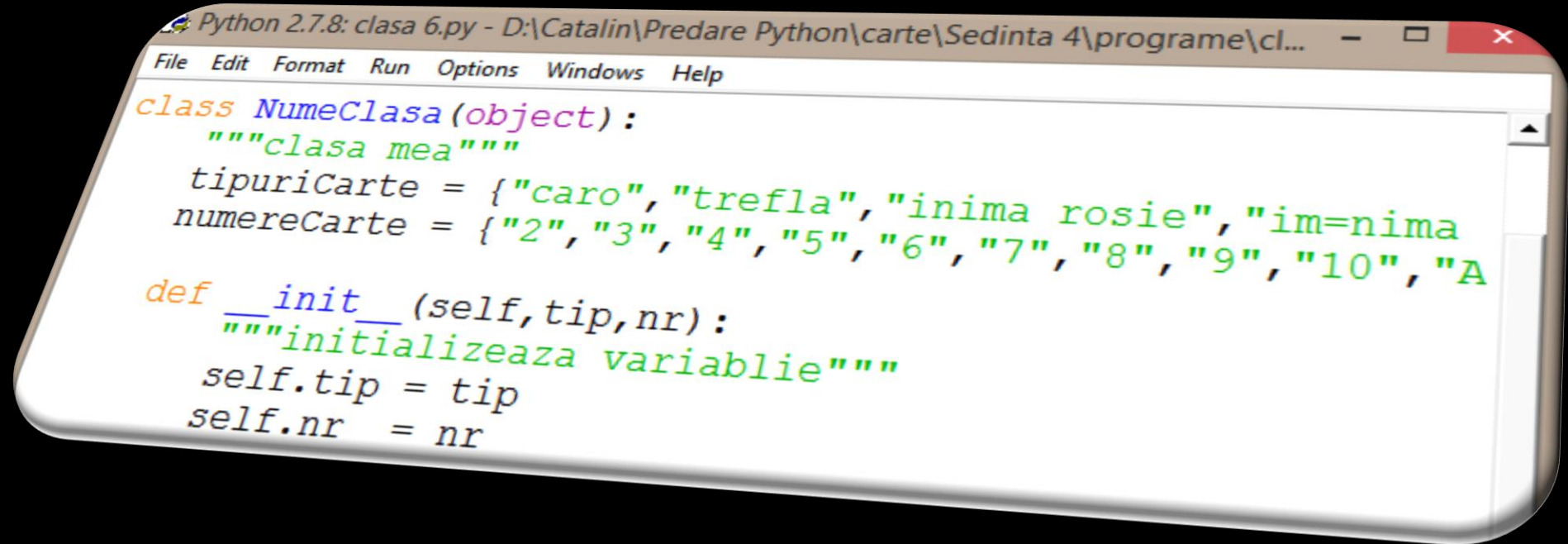
AttributeError: type object 'NumeClasa' has no attribute 'numarInstante'

Vom schimba valoarea stocata de total (metoda statica) la obiectele create. Vedem ca aceasta se updateaza, creandu-se attribute proprii ale obiectului x si y cu numele de total. Totusi metoda statica poate fi accesata inca prin intermediul clasei.

METODE STATICE

```
>>> class NumeClasa(object):  
    """clasa mea"""  
    total=0  
    def __init__(self,nume):  
        """nume student"""  
        self.nume= nume  
        NumeClasa.total+=1  
  
>>> x =NumeClasa("Ion")  
>>> y=NumeClasa("Maria")  
>>> NumeClasa.total  
2  
>>> y.total = 20  
>>> NumeClasa.total  
2  
>>> x.total=40  
>>> NumeClasa.total  
2  
>>> x.total  
40  
>>> y.total  
20  
>>>
```

METODE STATICE



```
Python 2.7.8: clasa 6.py - D:\Catalin\Predare Python\carte\Sedinta 4\programe\cl...
File Edit Format Run Options Windows Help

class NumeClasa(object):
    """clasa mea"""
    tipuriCarte = {"caro", "trefla", "inima rosie", "im=nima
    numereCarte = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "A

def __init__(self, tip, nr):
    """initializeaza variabile"""
    self.tip = tip
    self.nr = nr
```


INFOACADEMY.NET

METODE STATICE

```
Console X
D:\Catalin\Predare Python\carte\cap 4\clasa 5.py
Numarul si tipul ales de tine exista in packetul de carti!
Valori:caro,2

Numarul si tipul ales de tine nu exista in packetul de carti!
Valori:2,verde

Numarul si tipul ales de tine exista in packetul de carti!
Valori:caro,4

Numarul si tipul ales de tine exista in packetul de carti!
Valori:2,verde

Apasa <enter> pt a iesi.
```

METODE STATICE

```
Python 2.7.8: clasa 6_imbunatatita.py - D:\Catalin\Predare Python\carte\Sedinta 4...  
File Edit Format Run Options Windows Help  
# Program clasa6 imbunatatita  
# Demonstreaza utilizarea obiectelor  
# Ion Studentul - 1/26/13  
  
class NumeClasa(object):  
    """clasa mea"""  
    tipuriCarte = {"caro", "trefla", "inima rosie", "im=nima  
    numereCarte = {"2", "3", "4", "5", "6", "7", "8", "9", "10", "A
```

METODE STATICE

```
>>> ===== RESTART =====  
>>>  
Numarul si tipul ales de tine exista in packetul de carti!/n Valori:caro,2  
  
Numarul si tipul ales de tine nu exista in packetul de carti!/n Valori:2,verde  
  
Numarul si tipul ales de tine exista in packetul de carti!/n Valori:caro,4  
  
Numarul si tipul ales de tine nu exista in packetul de carti!/n Valori:2,verde  
  
Apasa <enter> pt a iesi.|
```

APELARE PRIN CLASA

```
>>> class ClasaMea (object):  
    x=1  
    y=3  
    def __init__(self):  
        print self.x  
        print ClasaMea.y  
  
>>> print ClasaMea.y  
3  
>>> print obj1.y  
3  
>>>
```

METODE STATICE OBIECT VS CLASA

Setarea lui test.b la 20 duce la o rescriere a metodei statice b pentru ambele obiecte.

Setarea lui test.a la 40 duce la o rescriere a metodei statice a doar pentru obiectul y.

Obiectul z a ramas neschimbat deoarece valoarea data unui atribut prin obiect precede (are prioritate) valorii data unor attribute prin numele clasei.

```
>>> class test(object):  
    """ clasa cu obiecte statice """  
    a = 1  
    b = 1  
  
>>> z=test()  
>>> y=test()  
>>> z.a  
1  
>>> y.a  
1  
>>> z.a= 10  
>>> z.a  
10  
>>> y.a  
1  
>>> test.b=20  
>>> z.b  
20  
>>> y.b  
20  
>>> test.a=40  
>>> z.a  
10  
>>> y.a  
40
```

RECURSIVITATE

Globals este un dictionar ce mentine toate variabilele globale (din namespaceul global). Acesta se apeleaza prin globals(). Se poate vedea ca variabilele definite anterior precum x sau y se regasesc in dictionar.

```
>>> x= 1
>>> y=2
>>> z=4
>>> globals()
{'__builtins__': <module '__builtin__' (built-in)>, '
__package__': None, 'x': 1, 'y': 2, '__name__': '__ma
in__', 'z': 4, '__doc__': None}
>>> .
```


GLOBALS()

```
>>> globals()
{'__builtins__': <module '__builtin__' (built-in)>, '__name__':
'__main__', '__doc__': None, '__package__': None}
>>> a = "test"
>>> globals()
{'__builtins__': <module '__builtin__' (built-in)>, '__name__':
'__main__', '__doc__': None, 'a': 'test', '__package__': None}
>>> if "a" in globals():
    print "Este"

Este
>>> globals()["a"]
'test'
>>> globals["b"] = "test2"

Traceback (most recent call last):
  File "<pyshell#19>", line 1, in <module>
    globals["b"] = "test2"
TypeError: 'builtin_function_or_method' object does not support
item assignment
>>>
```

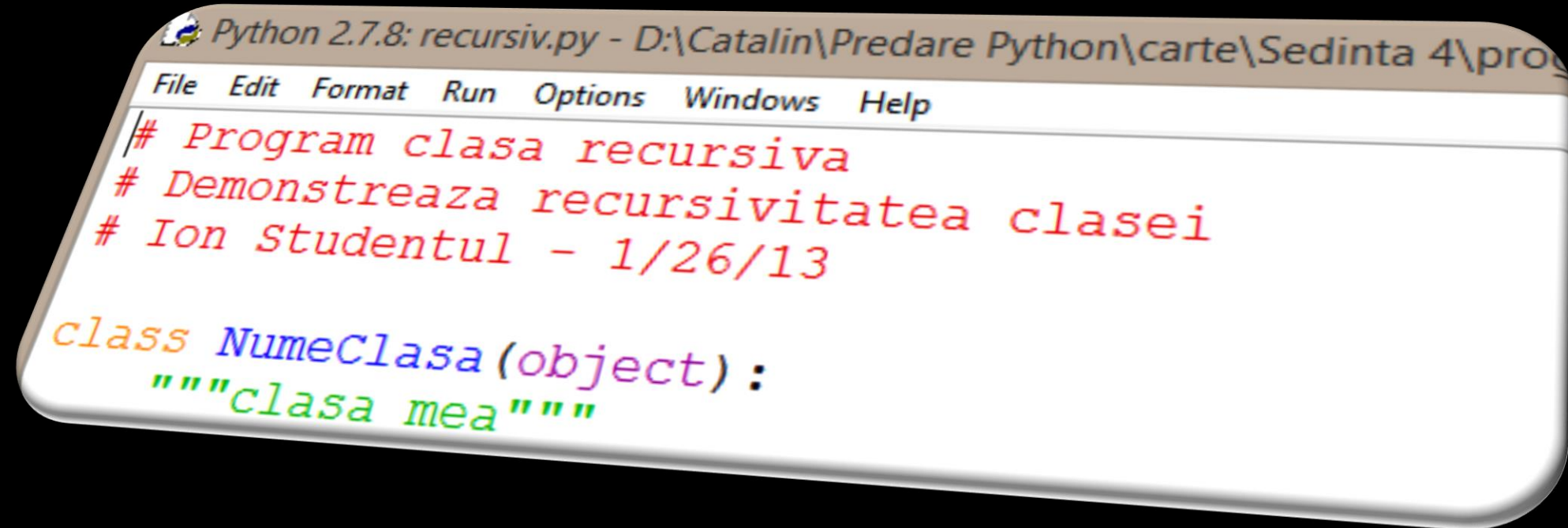
GLOBALS()

```
>>> class T(object):
        pass

>>> globals()["x"] = T()
>>>
>>> def TT():
        return "Test"

>>> globals()["y"] = TT()
>>> y
'Test'
>>> x
<__main__.T object at 0x027F7970>
>>> |
```

RECURSIVITATE

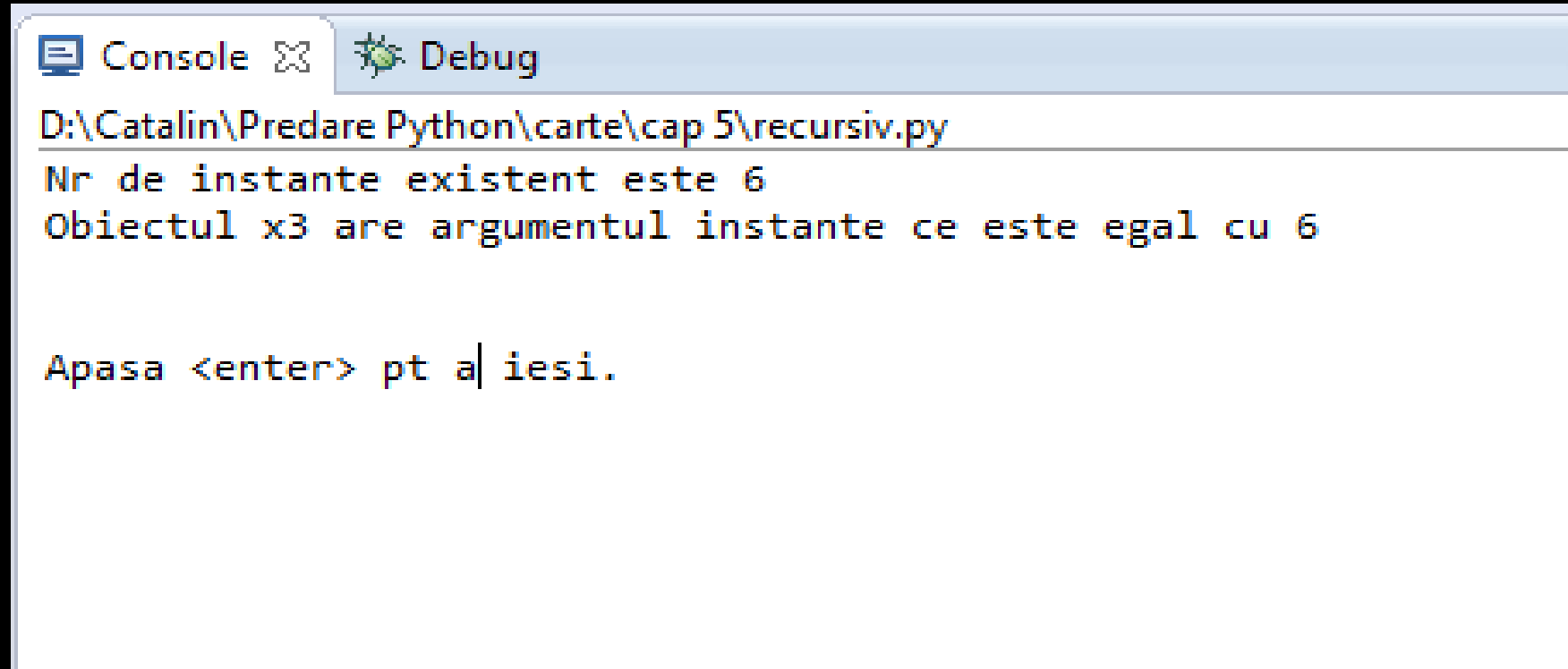


A screenshot of a Python 2.7.8 IDE window titled "Python 2.7.8: recursiv.py - D:\Catalin\Predare Python\carte\Sedinta 4\pro". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The code is written in a monospaced font with syntax highlighting: comments are red, the class name is blue, and the docstring is green. The code defines a class named "NumeClasa" that inherits from "object". The docstring for the class is "clasa mea".

```
# Program clasa recursiva  
# Demonstreaza recursivitatea clasei  
# Ion Studentul - 1/26/13  
  
class NumeClasa(object):  
    """clasa mea"""
```

METODE STATICE

OBIECT VS CLASA



The screenshot shows a Python IDE window with two tabs: 'Console' and 'Debug'. The 'Console' tab is active, displaying the output of a program. The file path 'D:\Catalin\Predare Python\carte\cap 5\recursiv.py' is shown at the top. The output consists of two lines: 'Nr de instante existent este 6' and 'Obiectul x3 are argumentul instante ce este egal cu 6'. Below the output, there is a prompt 'Apasa <enter> pt a| iesi.'.

```
Console [X] [X] Debug
D:\Catalin\Predare Python\carte\cap 5\recursiv.py
Nr de instante existent este 6
Obiectul x3 are argumentul instante ce este egal cu 6

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

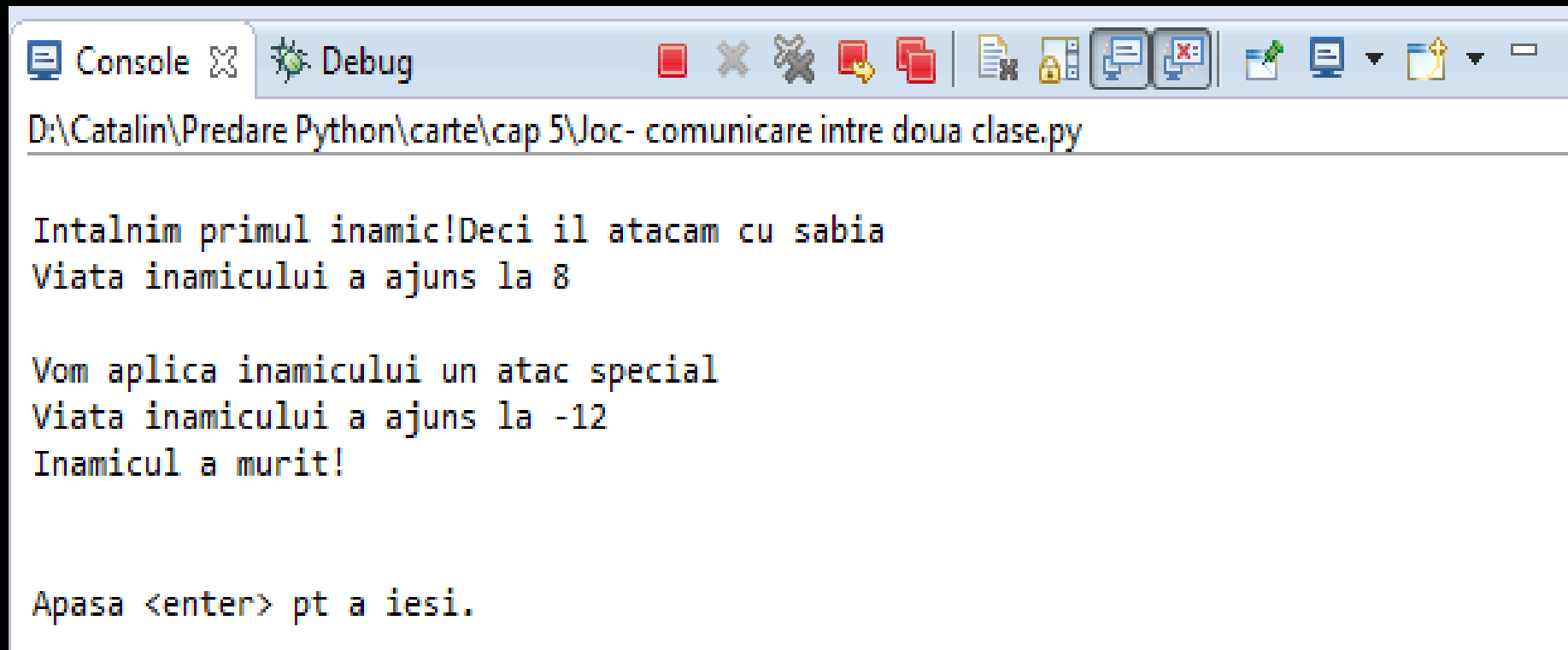
COMUNICAREA DINTRE CLASE

```
Python 2.7.8: Joc- comunicare intre doua clase.py - D:\Catalin\Predare Python\car...
File Edit Format Run Options Windows Help
# Program joc
# Demonstreaza apelarea unei clase in alta clasa
# Ion Studentul - 1/26/13

class Inamic(object):
    """Inamic"""
    def __init__(self):
        """initalizarea inamicului"""
        self.viata=10

def Danilo(self, rana):
```

COMUNICAREA DINTRE CLASE



The screenshot shows a Python IDE window with a 'Console' tab selected. The title bar indicates the file path is 'D:\Catalin\Predare Python\carte\cap 5\Joc- comunicare intre doua clase.py'. The console output displays the following text:

```
Intalnim primul inamic!Deci il atacam cu sabia  
Viata inamicului a ajuns la 8  
  
Vom aplica inamicului un atac special  
Viata inamicului a ajuns la -12  
Inamicul a murit!  
  
Apasa <enter> pt a iesi.
```


MOȘTENIREA

Utilizarea moștenirilor în programarea OOP este un subiect foarte discutat. Acesta presupune crearea unei clase care moștenește toate proprietățile altei clase (atribute și metode). Nu mosteneste obiectele.

```
>>> class Unu(object):  
    """prima clasa"""  
    #argumente statice  
    a = 1  
    b = 2  
    def __init__(self):  
        """initializare arg"""  
        self.c=3
```

```
>>> class Mosteneste_unu(Unu):  
    """a doua clasa"""  
    def __init__(self):  
        """initializare arg"""  
        self.x=25
```

```
>>> obj1=Unu()  
>>> print obj1.a  
1  
>>> print obj1.b  
2  
>>> print obj1.c  
3  
>>> obj2=Mosteneste_unu()  
>>> print obj2.a  
1  
>>> print obj2.b  
2  
>>> print obj2.x  
25
```

MOȘTENIREA

```
>>> print obj2.c
```

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

```
  File "<pyshell#12>", line 1, in <module>
```

```
    print obj2.c
```

```
AttributeError: 'Mosteneste_unu' object has no attribute 'c'
```

```
>>>
```

```
>>>
```

```
>>>
```

```
>>>
```

```
>>> print obj1.x
```

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

```
  File "<pyshell#17>", line 1, in <module>
```

```
    print obj1.x
```

```
AttributeError: 'Unu' object has no attribute 'x'
```

La printare obj2.c nu ar trebui să ne dea eroare dacă Mosteneste_unu() moștenește și metoda `__init__` din clasa Unu() și ar trebui să dea eroare dacă nu se moștenesc metodele. Dar clasa Mosteneste_unu() deja are o metoda `__init__`, prin urmare nu ar trebui să ne ridice o eroare compilatorul dacă metodele se moștenesc?

MOȘTENIREA

- Oare și metodele se moștenesc?
- Ce se întâmplă dacă și în clasa child și în clasa parent exista același argumente definite?
- Ce se întâmplă dacă și în clasa child și în clasa parent exista același metode?

MOȘTENIREA

Raspunsuri:

- Si metodele se mostenesc.
- Dacă și în clasa child și în clasa parent exista aceleași attribute definite, doar in clasa child vor fi rescrise valorile atributelor mostenite.
- Dacă și în clasa child și în clasa parent exista același metode, doar metodele definite in clasa child vor rescrie metodele mostenite din clasa parent.

MOȘTENIREA

```
Python 2.7.8: Mostenire2.py - D:\Catalin\Predare Python\carte\Sedinta 4\program... -  
File Edit Format Run Options Windows Help  
# Program mostenire -joc2  
# Explica mostenirea  
# Ion Studentul - 1/26/13  
  
class Fiinta(object):  
    """creaza o serie de proprietati ale unei fiinte"""  
    def __init__(self):  
        """proprietati mostenite de toate fiintele"""
```

MOȘTENIREA

Console Debug

D:\Catalin\Predare Python\carte\cap 5\Mostenire2.py

```
Bine ati venit la jocul "Cavalerul"
Iata proprietatile fiintei noastre( erou ):
viata : 100 puncte
sabie : 4 puncte vatamate din viata

Intalnim primul inamic !Deci il atacam cu sabia
Iata proprietatile fiintei intalnite( inamic ):
viata : 10 puncte
sabie : 2 puncte vatamate din viata
Viata a ajuns la 6 ( inamic )
Vom aplica inamicului un atac special!

Viata a ajuns la -14 ( inamic )
Inamicul a murit!

Intalnim al doilea inamic !Deci il atacam cu sabia
Iata proprietatile fiintei intalnite( inamic ):
viata : 10 puncte
sabie : 2 puncte vatamate din viata
Viata a ajuns la 6 ( inamic )
Vom aplica inamicului un atac special!

Viata a ajuns la -14 ( inamic )
Inamicul a murit!

Apasa <enter> pt a iesi.
```


OBJECT ENCAPSULATION

Pentru a încuraja încapsularea, în Python se poate regăsi și crea attribute și metode private, ceea ce înseamnă ca doar alte metode ale obiectului pot accesa acea metoda sau atribut.

O metoda privata se creează prin utilizarea a doua caractere underline __ în fata numelui metodei, iar un atribut privat se creează folosind tot doua caractere underline __ in fata numelui metodei.

INFOACADEMY.NET

OBJECT ENCAPSULATION

```
>>>
class Clasa_privat(object):
    """test clasa privata"""
    def test_public(self):
        """o metoda publica cu un argument privat"""
        self.argumentPublic = "public"
        self.__argumentPrivat = "privat"

    def acceseazaArgPrivat(self):
        """o metoda publica ce acceseaza un argument privat"""
        print self.__argumentPrivat, " <= argument privat "

    def __MetodaPrivata(self):
        """o metoda privata"""
        print self.__argumentPrivat, " <= argument privat "
        print self.argumentPublic, " <= argument public"

    def acceseazaMetodaPrivata(self):
        """o metoda publica ce acceseaza o metoda privata"""
        print self.__argumentPrivat, " <= argument privat "

>>> obj1=Clasa_privat()
```

INFOACADEMY.NET

OBJECT ENCAPSULATION

```
>>> obj1.test_public()
>>> obj1.__argumentPrivat

Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    obj1.__argumentPrivat
AttributeError: 'Clasa_privat' object has no attribute '__argumentPrivat'
>>> obj1.argumentPrivat

Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    obj1.argumentPrivat
AttributeError: 'Clasa_privat' object has no attribute 'argumentPrivat'
>>>
```

INFOACADEMY.NET

OBJECT ENCAPSULATION

```
>>>  
>>> obj1.accesseazaArgPrivat()  
privat <= argument privat  
>>>
```

```
>>> obj1._Clasa_privat__argumentPrivat  
'privat'  
>>> |
```

INFOACADEMY.NET

OBJECT ENCAPSULATION

```
>>> obj1.__MetodaPrivata()

Traceback (most recent call last):
  File "<pyshell#22>", line 1, in <module>
    obj1.__MetodaPrivata()
AttributeError: 'Clasa_privat' object has no attribute '__MetodaPrivata'
>>> obj1.MetodaPrivata()

Traceback (most recent call last):
  File "<pyshell#23>", line 1, in <module>
    obj1.MetodaPrivata()
AttributeError: 'Clasa_privat' object has no attribute 'MetodaPrivata'
>>>
```

INFOACADEMY.NET

OBJECT ENCAPSULATION

```
>>> obj1.accesseazaMetodaPrivata()  
privat <= argument privat  
>>> obj1._Clasa_privat__MetodaPrivata()  
privat <= argument privat  
public <= argument public  
>>> |
```


INFOACADEMY.NET

OBJECT ENCAPSULATION

```
Python 2.7.8: Private_Methods_and_arg.py - D:\Catalin\Predare Python\carte\Sedinta 4\
File Edit Format Run Options Windows Help
# Program Private methods and arguments
# Explica mostenirea
# Ion Studentul - 1/26/13

class Clasa_privat(object):
    """test clasa privata"""
    def test_public(self):
        """o metoda publica cu un argument privat"""
        self.argumentPublic = "public"
        self.__argumentPrivat = "privat"
```

```
>>> len("Cat ani crezi ca am?")  
20  
>>> len((1, 2, 3, 4, 5))  
5  
>>> len(["a", "b", "c"])  
3
```

Polimorfismul este o calitate de a putea fi capabil să utilizezi același tip de funcție sau clasă cu tipuri diferite.

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 win32

Type "copyright", "credits" or "license()" for more information.

```
>>> class Animal:
    def __init__(self, name):      # metoda de initializare a clasei
        self.name = name
    def vorbeste(self):           # Metoda abstracta
        pass
```

```
>>> class Pisica(Animal):
    def vorbeste (self):
        return 'Miau!'
```

```
>>> class Caine(Animal):
    def vorbeste (self):
        return 'Ham!'
```

```
>>> obj1 = Pisica ('Lola')
>>> obj1.vorbeste()
'Miau!'
>>> obj2 = Caine ('Lassie')
>>> obj2.vorbeste()
'Ham!'
>>> |
```

LUCRUL CU FIȘIERELE

Exista doua tipuri de fisiere:

- plain text - fișier ce este creat doar cu caractere ASCII
- fișier binar - un fișier ce conține și alte elemente non-ASCII. Fișierul binar poate fi procesat doar de aplicația ce cunoaște structura acelui fișier

LUCRUL CU FIȘIERELE

`fisier_object = open(numeFisier, mod)` unde `fisier_object` este variabila care susține obiectul fișier returnat

Modurile pot fi:

- 'r' când vrem să citim din fișier (exclusiv citire). Fișierul trebuie să existe!
- 'w' când dorim să scriem în fișier (exclusiv scriere). Dacă fișierul nu exista va fi creat. Dacă fișierul deja exista conținutul vechiului fișier va fi șters.
- 'a' deschide fișierul pentru scriere(exclusiv scriere). Dacă fișierul nu exista va fi creat. Orice data scrisa cu a va fi adăugată la final.
- 'r+' când vrem să citim și să scriem fișierul. Fișierul trebuie să existe!
- 'w+' când vrem să citim și să scriem fișierul. Dacă fișierul nu exista va fi creat. Dacă fișierul deja exista conținutul vechiului fișier va fi șters.
- 'a+' când vrem să citim și să scriem fișierul. Dacă fișierul nu exista va fi creat. Orice data scrisa cu ,a' va fi adăugată la final.

LUCRUL CU FIȘIERELE

```
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.
>>> x=open("c:/test.ini","r+")

Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    x=open("c:/test.ini","r+")
IOError: [Errno 2] No such file or directory: 'c:/test.ini'
>>> x=open("c:/test.ini","w+")
>>>
>>> |

>>> f=open("C:/test.ini","r")
>>> print f
<open file 'C:/test.ini', mode 'r' at 0x02359D88>
>>> |
```

Calea absoluta este calea prin sistemul de fișiere de la rădăcina sau drive pana la fișierul dorit. Calea relativa se raportează la directorul curent unde acel program rulează. Argumentul mod este de tip șir de caractere și este opțional; dacă îl omitem se va consider ca fiind 'r'.

Citirea unui fișier 4 soluții:

`file.read()` –aceasta metoda de citire va returna tot conținutul fișierului într-un singur șir de caractere.

`file.readline()` – aceasta metoda de citire citește linie cu linie. De fiecare data când este apelata va returna o linie.

`file.readlines()` – aceasta metoda citește toate liniile, iar fiecare linie este un element dintr-o lista.

A patra soluție este să facem o buclare a fișierului. Mai jos regăsim un exemplu:

```
file = open('newfile.txt', 'r')
for line in file:
    print line,
```

LUCRUL CU FIȘIERELE

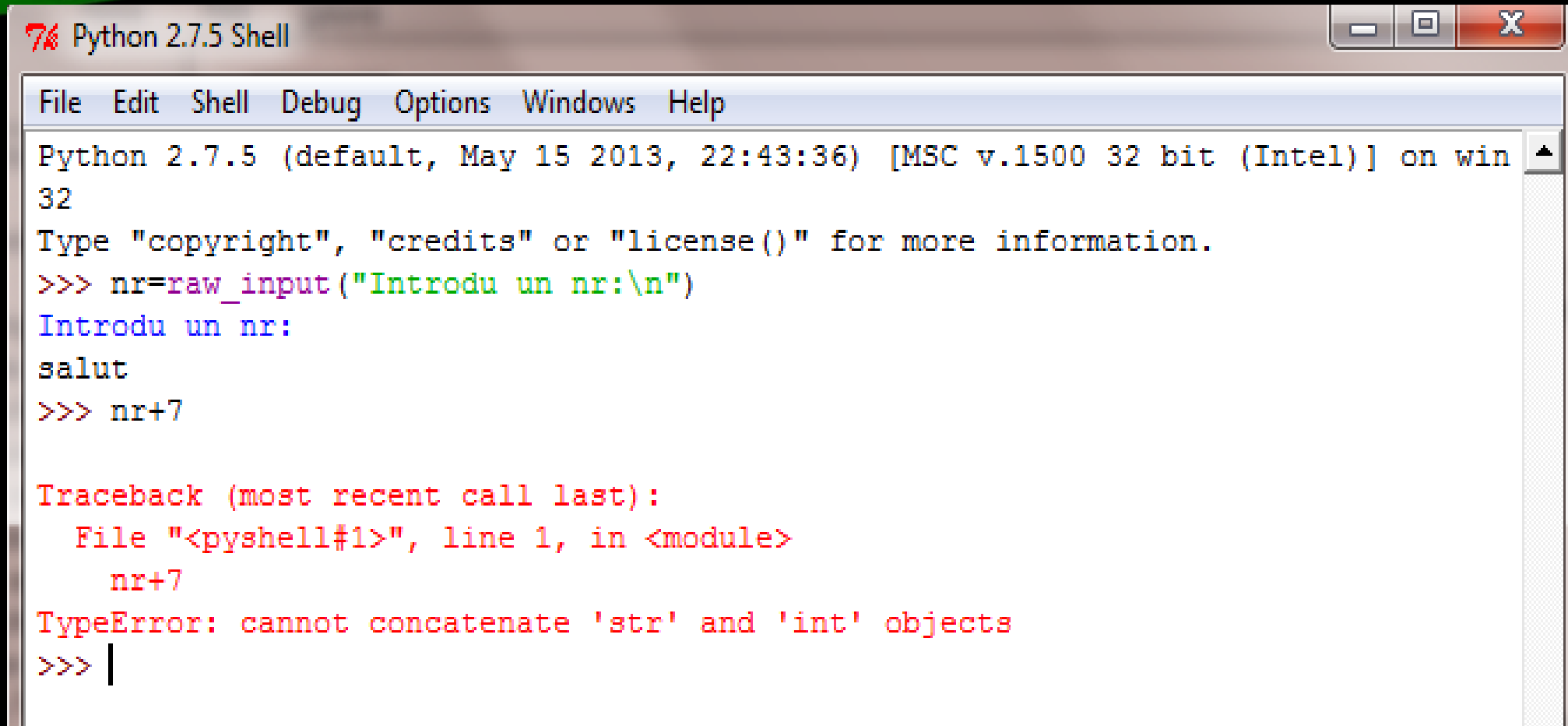
```
Citire_fisier ✖  
1 # Program Citire fisiere  
2 # Explica accesarea fisierelor  
3 # Ion Studentul - 1/26/13  
4  
5 print("\nDeschiderea si inchiderea fisierului.")  
6 text_file = open("text_importat1.txt", "r")  
7 text_file.close()
```

LUCRUL CU FIȘIERELE

Scrierea fișierului se realizează cu write(). Orice fișier utilizat trebuie închis pentru a permite utilizarea ulterioară sau pentru a salva ce ai scris

```
fișier = open("nou.txt", "w")  
fișier.write("Aici e prima Linie.\n")  
fișier.write("Aici e a doua Linie\n")  
fișier.write("Aici e a treia Linie.")  
fișier.write("inca e a treia Linie.")  
fișier.close()
```

EXCEPT II

A screenshot of a Python 2.7.5 Shell window. The window has a title bar with the text 'Python 2.7.5 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Windows', and 'Help'. The main text area shows the following content:

```
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.
>>> nr=raw_input("Introdu un nr:\n")
Introdu un nr:
salut
>>> nr+7

Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    nr+7
TypeError: cannot concatenate 'str' and 'int' objects
>>> |
```

EXCEPT II

```
>>> int("salut")
```

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

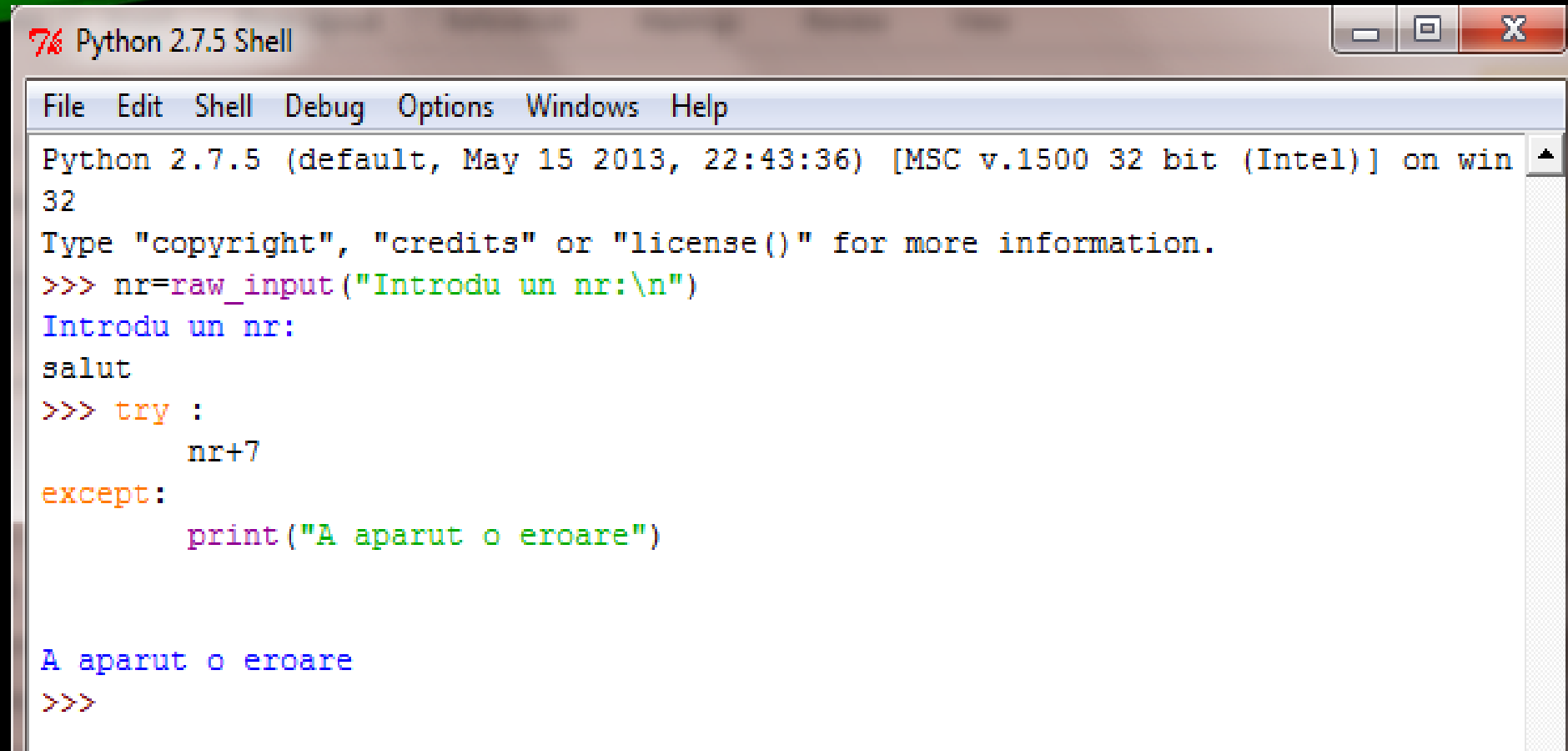
```
  File "<pyshell#5>", line 1, in <module>
```

```
    int("salut")
```

```
ValueError: invalid literal for int() with base 10: 'salut'
```

```
>>> |
```

EXCEPTII

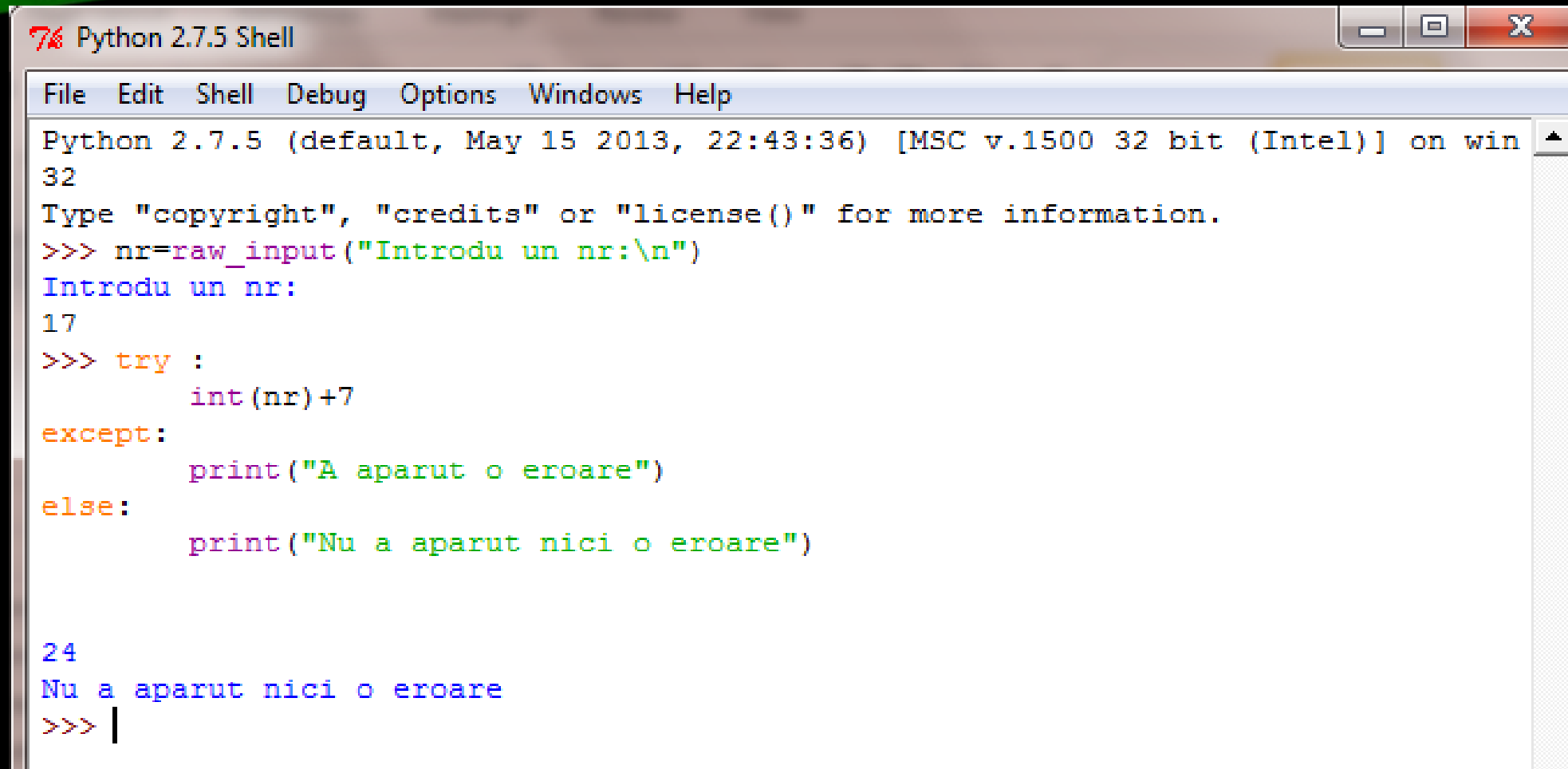


The screenshot shows a Python 2.7.5 Shell window with a menu bar (File, Edit, Shell, Debug, Options, Windows, Help) and a title bar (Python 2.7.5 Shell). The console output shows the Python version and build information, followed by a prompt to type "copyright", "credits", or "license()". The user enters "Introdu un nr:" and "salut". The user then enters a try-except block. The try block contains "nr+7". The except block contains "print('A aparut o eroare')". The output shows "A aparut o eroare" and the prompt ">>>".

```
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.
>>> nr=raw_input("Introdu un nr:\n")
Introdu un nr:
salut
>>> try :
        nr+7
except:
        print("A aparut o eroare")

A aparut o eroare
>>>
```

EXCEPTII- TRY/EXCEPT/ELSE



```
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.
>>> nr=raw_input("Introdu un nr:\n")
Introdu un nr:
17
>>> try :
        int(nr)+7
except:
        print("A aparut o eroare")
else:
        print("Nu a aparut nici o eroare")

24
Nu a aparut nici o eroare
>>> |
```


EXCEPTII- TRY/EXCEPT/ELSE

Tipul Excepției	Descriere
IOError	Eroare ridicata când încercăm să deschidem un fișier inexistent.
IndexError	Eroare ridicata când o secvență este indexata cu un număr inexistent.
KeyError	Eroare ridicata când o cheie a unui dictionar nu este găsită.
NameError	Eroare ridicata când un nume de variabila sau funcție nu este găsit.
SyntaxError	Eroare ridicata când o eroare de sintaxa apare.
TypeError	Eroare ridicata când o funcție incorporata este aplicata la un obiect/variabila neadecvata.
ValueError	Eroare ridicata când o funcție incorporata este aplicata la un obiect/variabila adecvata, dar o valoare greșită.
ZeroDivisionError	Eroare ridicata când al doilea argument al unei diviziuni(numitor) este zero.

EXCEPTII- TRY/EXCEPT/ELSE

```
>>> try:
        int("salut")
except (ValueError), e:
        print "nu da eroare"

nu da eroare
>>> print e
invalid literal for int() with base 10: 'salut'
>>> # variabila e a capturat mesajul de eroare
>>> |
```

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 obiectelor create

LUCRUL CU FIȘIERELE

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.

EXCEPTII TEMA IN CLASA

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

VA MULTUMESC PENTRU PARTICIPARE

La revedere!