INFOACADEMY.NET

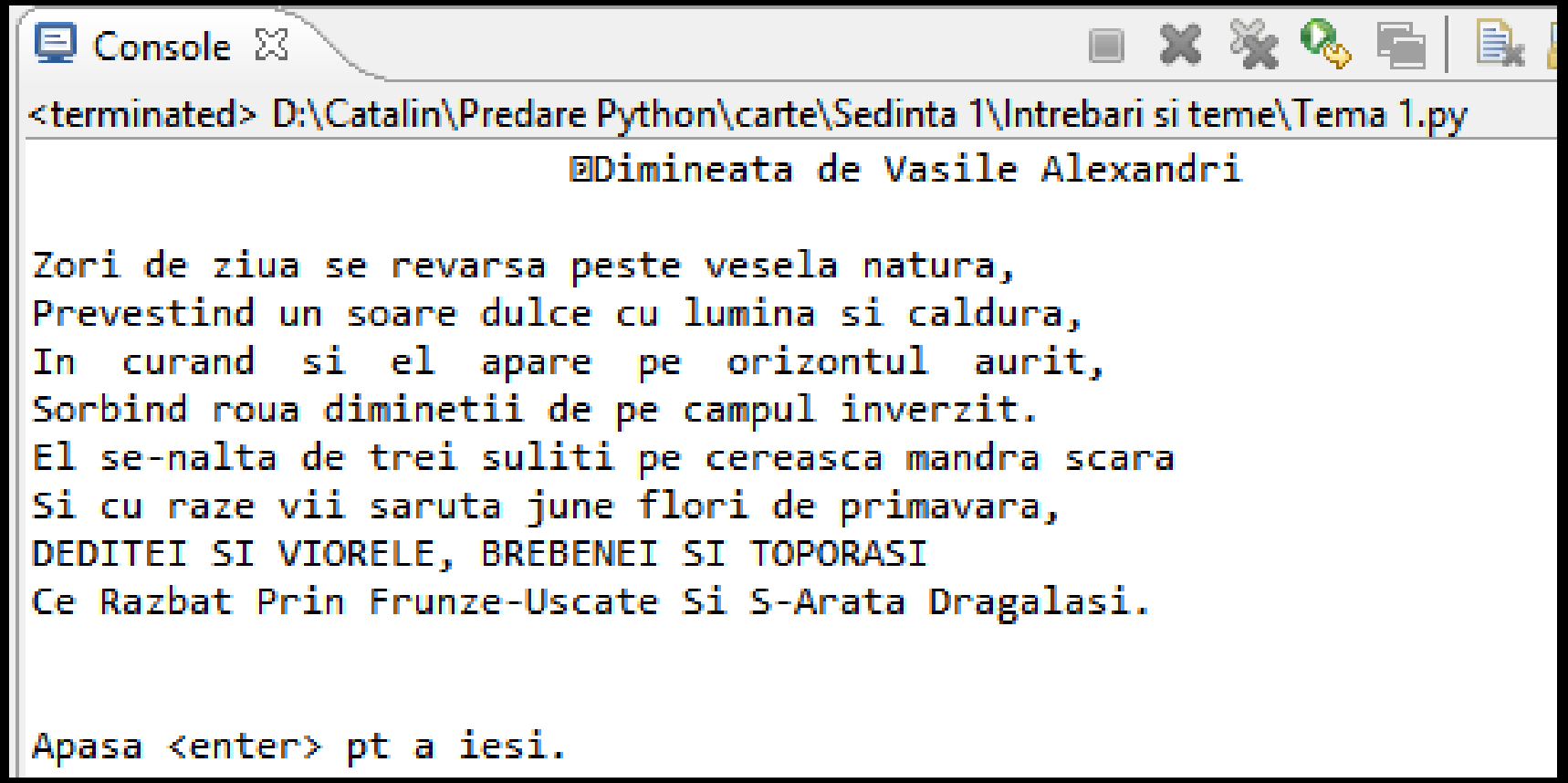
PYTHON FUNDAMENTALS

Curs interactiv de python

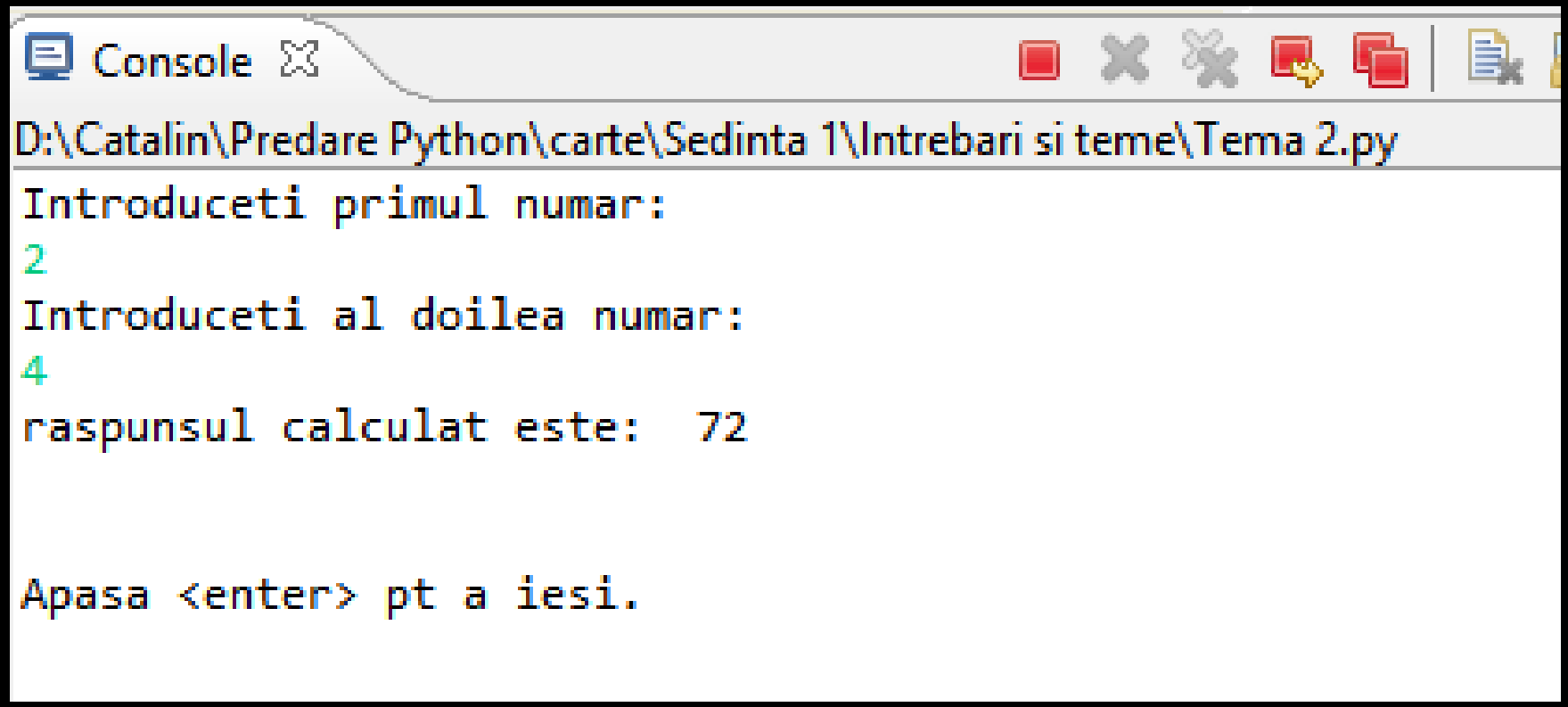
STRUCTURA SEDINTA 2 : BUCLE SI OPERATORI DECIZIONALI

- Tema sedinta anterioara
- Lucrul cu operatori decizionali și IF
- Lucrul cu bucle de calcul: while și for
- Lucrul avansat cu șiruri de caractere
- Tuplu
- Lista
- Dictionar

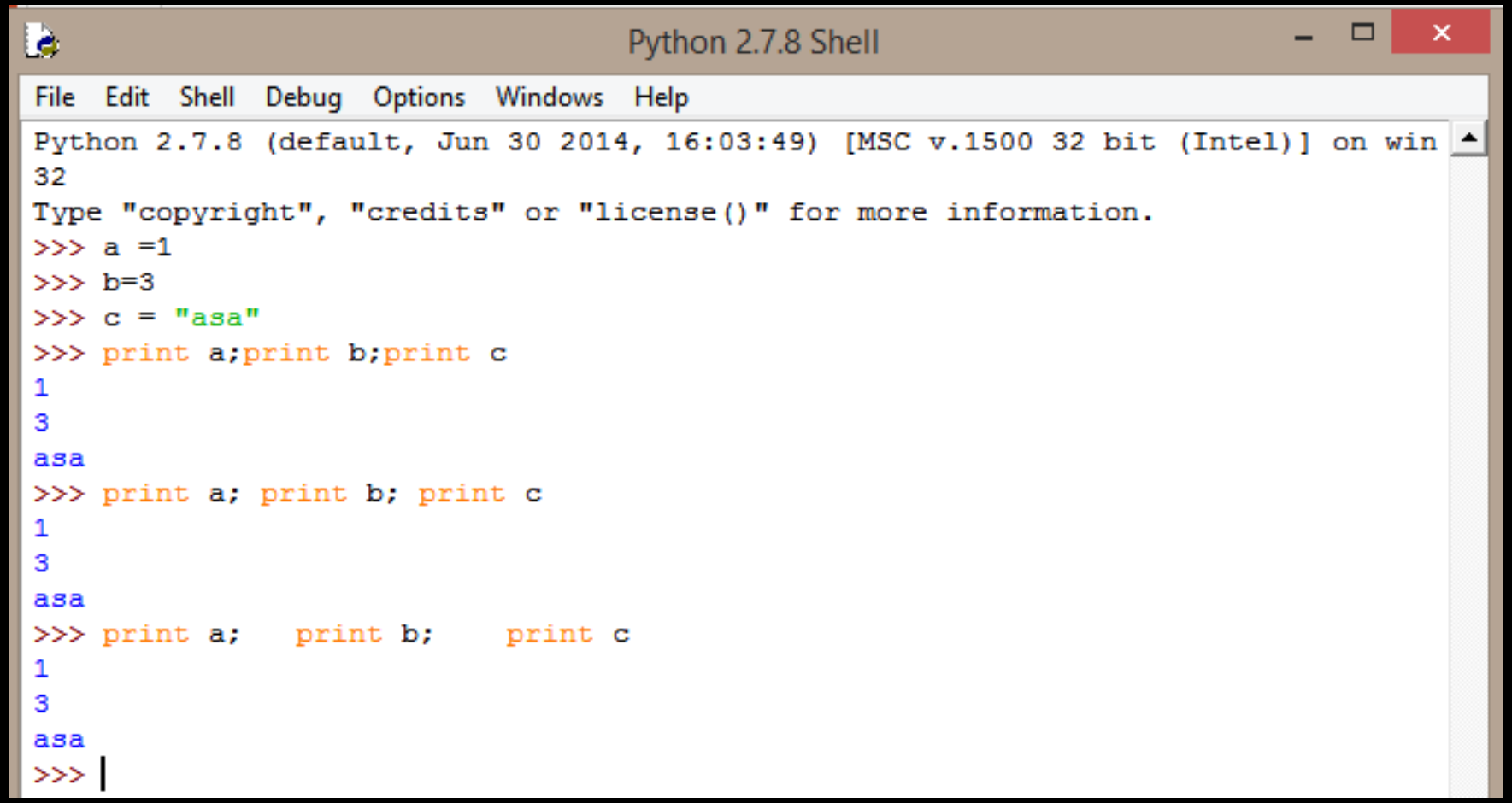
TEMA SEDINTA ANTERIOARA



TEMA SEDINTA ANTERIOARA



MULTIPLE COMENZI PE ACEASI LINIE



COMANDA PRINT SI DIFERENTE PYTHON2.X SI PYTHON 3.X

Până în prezent foloseam comanda print fără paranteze.

Acum putem vedea ca și comanda de afișare poate fi apelata cu paranteze. De fapt, aici ar fi una din schimbările majore și incompatibile intre versiunea Python 2.7.x și Python 3.x și anume Python 2.7.x recunoaște ambele moduri de a afișa un sir de caractere, pe când Python 3.x nu recunoaște decât print cu paranteze rotunde, adică print(‘string’).

```
>>> print "salut"
salut
>>> print("salut")
salut
>>>
```

COMANDA PRINT SI DIFERENTE PYTHON2.X SI PYTHON 3.X

Prin urmare, tot ce a fost scris până în prezent pentru 2.x acum trebuie portat la noua forma pentru 3.x. Portarea este un procedeu de translatare de la o forma la o alta prin schimbarea codului pentru a fi compatibil cu ambele versiuni sau cu noua versiune. Exista programe python care fac translatarea Python 2.x in Python 3.x automat , deci nu trebuie sa realizati voi aceasta translatare.
Nu se recomanda utilizarea funcției print() pentru afișarea unor caractere. În Python 2.7 utilizarea comenzii print cu paranteze nu se poate folosi corespunzător când vrem să afișam și variabile și șiruri de caractere despartite de virgula. Prin urmare va fi folosit cat de rar este cu putința. Figura de mai jos indica acest lucru, unde se poate vedea ca se afiseaza parantezele in cazul in care printam (x,”Salut”).

```
>>> print "salut"
salut
>>> print("salut")
salut
>>> x=23
>>> print(x,"Salut")
(23, 'Salut')
>>> print x,"Salut"
23 Salut
```

LUCRUL CU OPERATORI DECIZIONALI ȘI IF

- forma structurala a unui if

```
if (conditie):
        expresie1
```

Tradus mot-a-mot:

```
daca(conditie):
        expresie1
```

expresie1 va fi rulat numai cand conditia este adevarata.

Condițiile pot fi numerice, Boolean sau compararea șirurilor de caractere

OPERATORI DECIZIONALI DE TIP BOOLEAN

- forma structurala a unui if

```
if (True):
        expresie1
```

- Putem echivala acest caz cu următoarea sintaxa:

```
expresie1
```

OPERATORI DECIZIONALI DE TIP BOOLEAN

- forma structurala a unui if

```
if(False):
    expresie1
```

- Putem echivala acest caz cu lipsa unei sintaxe.

OPERATORI DECIZIONALI BOOLEAN

Avem nevoie de o metoda prin care indicam că, cand condiția este adevărată (True - boolean) , sintaxa if ar trebui să ruleze mai multe expresii.

Sa ne uitam un pic fragmentul de cod de mai jos:

```
if (False):
    print "x este mai mare ca 3"
print "Asa este!"
print “Test”
```

Cum se va comporta interpretorul? Condiția din cadrul if-ului este falsa astfel expresia print "x este mai mare ca 3." nu va fi rulata. În schimb expresia print "Asa este!" va fi rulata, ca de altfel și expresia print "Test". Dorim ca și a doua expresie să fie introdusa în if sub un singur bloc. Acest lucru este posibil folosind spațierea.

OPERATORI DECIZIONALI BOOLEAN

Aici am folosit patru spatii pentru indentare.

```
if (True):
    print "x este mai mare ca 3"
    print "Asa este!"
print “Test”
```

Fragmentul de cod de mai sus va rula ambele expresii de print doar în cazul în care conditia este adevărata. Deoarece conditia este adevărata va avea ca urmare rularea celor doua expresii print "x este mai mare ca 3." si print "Asa este!". Sintaxa print “Test” va rula oricum deoarece nu face parte din blocul de expresii ale if-ului.

Atât IDLE, cat și Eclipse ne oferă indentare automata la scrierea codului, detectând începerea unui bloc de expresii. Începerea blocului de comenzi se face prin indentarea unui tab, tab ce e configurat standard la 4 spatii.

OPERATORI DECIZIONALI BOOLEAN

Prin trecerea la nivelul anterior de număr de taburi se închide un bloc de expresii. In loc de tab se pot folosi și spatii, dar e mai facil să folosim un tab fiind mult mai vizibilă trecerea de la un bloc la altul. Indentarea trebuie sa fie consistenta altfel interpretorul returneaza erori.

```
>>> if (True):
        #bloc de expresii
        #Blocul tine cat timp am aceasi indentare
        #daca apas backspace pe un rand nou voi termina indentarea
        print "salut"
#aici am terminat indentarea


salut
>>>
```

OPERATORI DECIZIONALI BOOLEAN

- Sa presupunem ca avem următoarea expresie if:

```
if (conditie):
        expresie1
else:
        expresie2
```

- In cazul în care condiție este adevărata va fi rulata expresie1 , în caz contrar (condiția este falsa) expresie2 va fi rulata.

OPERATORI DECIZIONALI BOOLEAN

```
>>> if(True):
        print "Adevarat"
else:
        print "Fals"


Adevarat
>>> if(False):
        print "Adevarat"
else:
        print "Fals"


Fals
>>>
```

OPERATORI DECIZIONALI NUMERICI

- Condițiile numerice folosesc operatori decizionali.

Operator	Însemnătate	Exemplu de condiție	Ce returnează
==	Egal cu	5 == 5	True
!=	Nu este egal cu	8 != 5	True
>	Mai mare	3 > 10	False
<	Mai mic	5 < 8	True
>=	Mai mare sau egal cu	5 >= 10	False
<=	Mai mic sau egal cu	5 <= 5	True

OPERATORI DECIZIONALI NUMERICI

- Exemple: Definim x = 23 apoi rulam

```
>>> if(x>24):
        print "Salut"
        # Aceasta linie nu va fi rulata deoarece 23 nu este
        #mai mare decat 24


>>> if(x>22):
        print "Salut"
        # Aceasta linie va fi rulata deoarece 23 este
        #mai mare decat 22


Salut
```

- se verifica daca expresia x mai mare ca 24 este adevarata sau x mare decat 22. Cand conditia este adevarata atunci codul din blocul de comenzi de sub if va rula. Prin urmare, vom vedea afisat sirul de caractere „Salut” doar cand x este mai mic decat 22.

OPERATORI DECIZIONALI DE TIP ȘIR DE CARACTERE

- Comparare ASCII într-o ordine lexicografică caracter cu caracater
- Se utilizeaza functia ord() pentru a vedea numarul alocat ASCII pentru un caracter

```
>>> ord('a')
97
>>> ord('b')
98
>>> if ("asaMaiMerge"<"b"):
        print "a=>ASCII mai mare ca b=>ASCII"


a=>ASCII mai mare ca b=>ASCII
```

OPERATORI DECIZIONALI DE TIP ȘIR DE CARACTERE

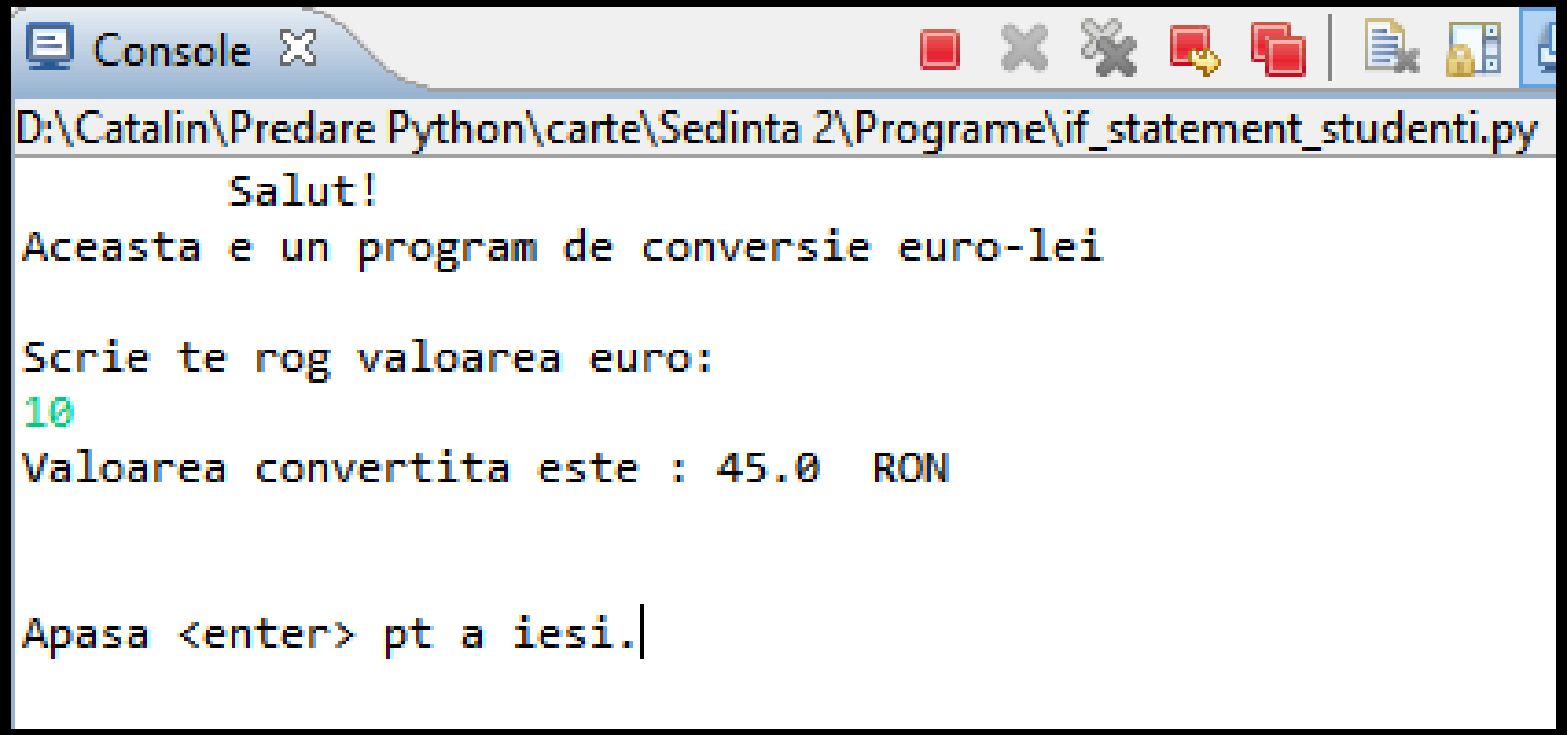
- Operația inversa a funcției ord() este chr().
- Funcția chr() nu acceptă decât numere de tip integer

```
>>> chr('a')

Traceback (most recent call last):
  File "<pyshell#38>", line 1, in <module>
    chr('a')
TypeError: an integer is required
>>> chr(97)
'a'
>>> chr(97.0)

Traceback (most recent call last):
  File "<pyshell#40>", line 1, in <module>
    chr(97.0)
TypeError: integer argument expected, got float
```

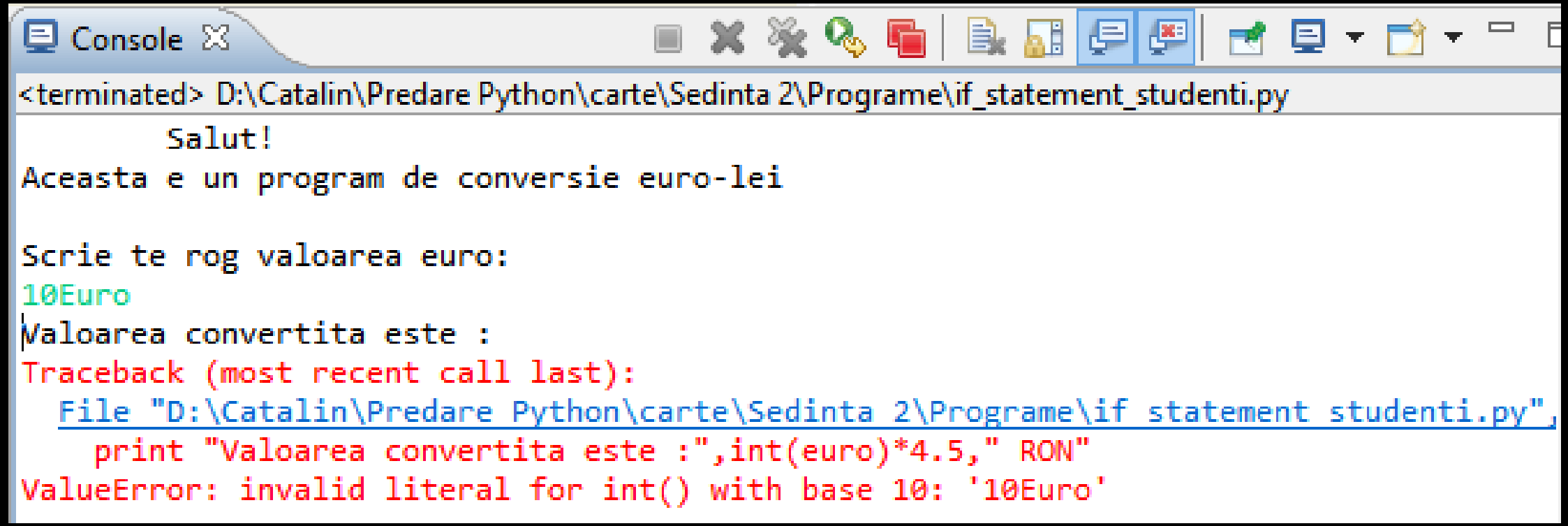
IF-ELSE

```
1  # Manipularea sir caractere  prin if statement
2  # Demonstreaza  utilizarea sintaxei if
3  # Ion Studentul 1/13/03
4
5  print("\tSalut! \nAceasta e un program de conversie euro-lei")
6
7  euro=raw_input("\nScrie te rog valoarea euro:\n")
8
9  print "Valoarea convertita este :",int(euro)*4.5," RON"
10
11
12 raw_input("\n\nApasa <enter> pt a iesi.")
```

IF-ELSE


Console
D:\Catalin\Predare Python\carte\Sedinta 2\Programe\if_statement_studenti.py
Salut!
Aceasta e un program de conversie euro-lei
Scrie te rog valoarea euro:
10
Valoarea convertita este : 45.0 RON
Apasa <enter> pt a iesi.

IF-ELSE

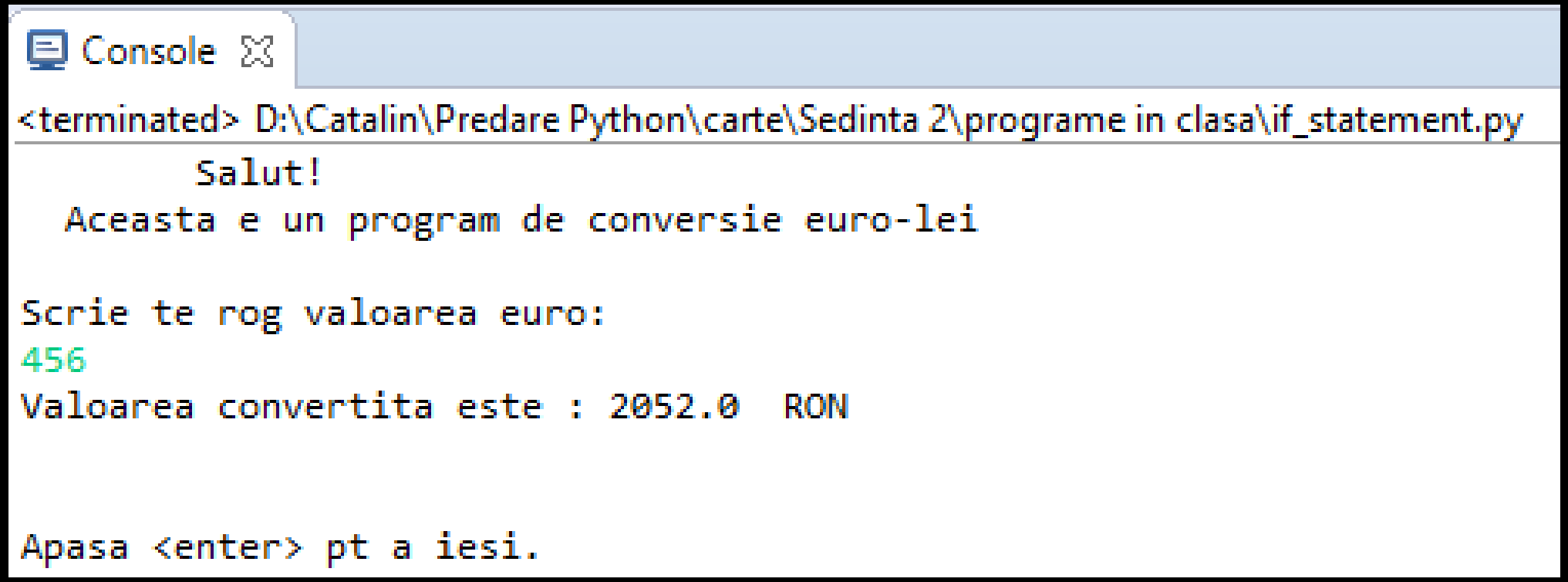

Console
<terminated> D:\Catalin\Predare Python\carte\Sedinta 2\Programe\if_statement_studenti.py
Salut!
Aceasta e un program de conversie euro-lei
Scrie te rog valoarea euro:
10Euro
Valoarea convertita este :
Traceback (most recent call last):
File "D:\Catalin\Predare Python\carte\Sedinta 2\Programe\if_statement_studenti.py",
print "Valoarea convertita este :",int(euro)*4.5," RON"
ValueError: invalid literal for int() with base 10: '10Euro'

LUCRUL CU OPERATORI DECIZIONALI ȘI IF

- Program If

```
# Manipularea sir caractere  prin if statement
# Demonstreaza  utilizarea sintaxei if
# Ion Studentul 1/13/03

print("\tSalut! \n  Aceasta e un program de conversie euro-lei")

euro=raw_input("\nScrie te rog valoarea euro:\n")

if (euro.isdigit()):
      euro=int(euro)
      print "Valoarea convertita este :",euro*4.5," RON"
else:
    print "Valoarea introdusa nu este un numar!"


raw_input("\n\nApasa <enter> pt a iesi.")
```

LUCRUL CU OPERATORI DECIZIONALI ȘI IF



LUCRUL CU OPERATORI DECIZIONALI ȘI IF

```
>>>
>>> a=1;b=2;c=3
>>> if(a==1):
        print "a = 1"
        if (b==2):
                print "b = 2"
        if (c == 4):
                print "c = 4"
        else:
                print "c = ",c


a = 1
b = 2
c =  3
>>>
```

In Python nu suntem limitati la a crea un singur if.Putem crea o ierarhie de tip if in if.

LUCRUL CU OPERATORI DECIZIONALI ȘI IF

Avem posibilitatea de a avea mai multe ramuri de if intr-un if, asa cum avem posibilitatea de a utiliza elif.

Acest elif se adauga cu scopul de a avea inca o ramura ce aplica o conditie in vederea rularii blocului de expresii de sub acel elif.

Operatiile se vor face de sus in jos.

LUCRUL CU OPERATORI DECIZIONALI ȘI IF

Prin urmare, daca conditia(conditie1) din if este adevarata, se va rula blocul de expresii de sub if (expresie1). In cazul in care nu este adevarata conditia din if, atunci se va verifica conditia din elif(conditie2). In cazul in care conditie2 este adevarata se va rula blocul de expresii de sub elif (expresie2). In cazul in care nici conditie2 nu este adevarata se ruleaza blocul de expresii de sub else (expresie3).

```
if (conditie1):
    expresie1
elif (conditie2):
    expresie2
else:
    expresie3
```

LUCRUL CU OPERATORI DECIZIONALI ȘI IF

```
If (conditie):
        expresie
elif (conditie2):
        expresie2
elif (conditie3):
        expresie3
....
elif (conditie n)
        expresien
else:
        expresie n+1
```

Asa cum probabil v-ati asteptat putem avea multiple conditii elif. Toate aceste conditii vor fi verificate pe rand, pana una din aceste conditii este adevarata. Pe ramura unde conditia este adevarata vom rula blocul de expresii de sub ramura respectiva.

LUCRUL CU OPERATORI DECIZIONALI ȘI IF

```
>>> if(x>2):
        print "x mai mare ca 2"
elif(x==2):
        print "x este egal cu 2"
elif(x>=2):
        print "Aceasta linie nu va fi afisata niciodata"
else:
        print "test"


x este egal cu 2
>>>
```

Atentie totusi la conditii si la logica din spate. Iata un exemplu in care a doua ramura de elif nu va fi accesata niciodata deoarece conditiile respective vor fi indeplinite de ramurile de dinainte. Avem un mic programel in care definim un numar stocat de variabila x. Acest numar este 2.

LUCRUL CU OPERATORI DECIZIONALI ȘI IF

```
If (conditie):
        expresie
elif (conditie2):
        expresie2
```

In primele exemple cu operatori decizionali de tip if nu aveam si un else implementat. Intrebarea este daca am putea avea si expresii de tip if-elif fara a avea un else la final. Raspunsul este da. Putem avea implementari de operatori decizionali de tip if-elif fara a implementa si ramura else. Ramane aceasi logica valabila: toate aceste conditii vor fi verificate pe rand, pana una din aceste conditii este adevarata. Pe ramura unde conditia este adevarata vom rula blocul de expresii de sub ramura respectiva. In cazul in care nici una din conditiile testate nu sunt adevarate nu vom aplica nici un bloc de expresii.

LUCRUL CU OPERATORI DECIZIONALI ȘI IF

```
>>> x=2;y=3
>>> if (x==3):
        print 3
elif (x == 2):
        print 2


2
>>>
```

LUCRUL CU OPERATORI DECIZIONALI ȘI IF

```
>>> x='4'
>>> if (x=='4'):
        print "not"
elif (x=='4'):
        print "ok"


not
>>>
```

Originalitatea conditiilor este importanta. Daca avem doua condiții identice programul nu ne va returna eroare, dar nu va ajunge niciodată pe ramura cu elif-ul ce se afla mai jos cu condiția ‘duplicat’. Retine ca python prelucrează fiecare sintaxă pe rând.

TRATAREA UNEI VARIABILE CA SI CONDITIE

```
>>> x=1
>>> if (x):
        print x
else:
        print "else"


1
>>> x=0
>>> if (x):
        print x
else:
        print "else"
```

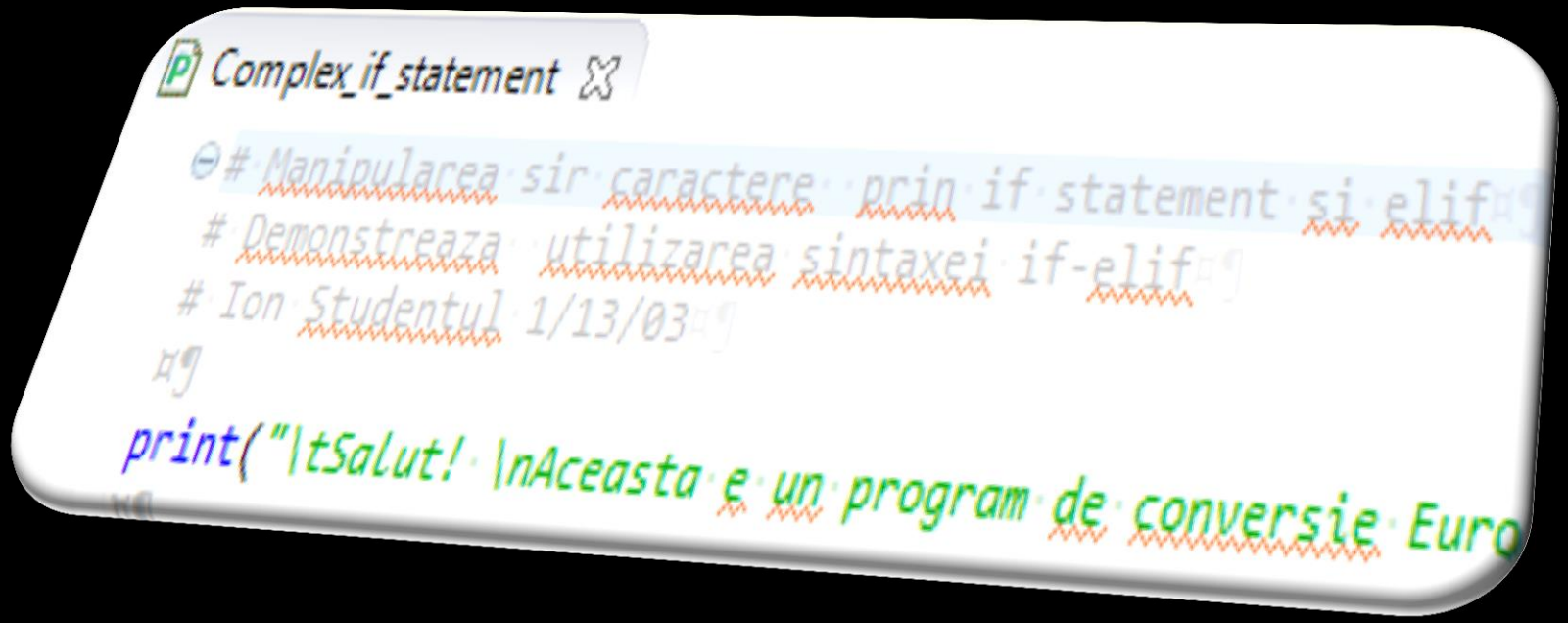
Tratarea unei variabile ca o condiție adevărata sau falsa este un alt mod de a scrie sintaxe if. In acest caz variabila x stocheaza valori numerice. Pentru orice valoare ce este diferita de 0 va rula blocul de sintaxe de sub if deoarece conditia este considerata a fi adevarata. In cazul in care variabila x stocheaza valoarea 0 atunci va rula blocul de expresii de sub else deoarece conditia este considerata a fi falsa.

TRATAREA UNEI VARIABILE CA SI CONDITIE

```
else
>>> x="orice"
>>> if (x):
        print x
else:
        print "else"


orice
>>> x="0"
>>> if (x):
        print x
else:
        print "else"


0
>>> x=''
>>> if (x):
        print x
else:
        print "else"


else
>>>
```

In cazul in care sirul de caractere stocat de variabila x reprezintă numărul zero sau în cazul în care sirul de caractere stocat de variabila x este un șir ce contine cuvantul orice conditia este tratata ca fiind adevarata. Șirul “0” nu este un șir vid, deci nu este un șir ce ar genera o condiție falsa. Orice sir de caractere ce nu este gol adica orice sir de caractere ce este diferit de sirul de caractere "" va fi tratat ca o conditie adevarata.

Sirul de caractere nul, adica sirul de caractere "" va fi tratat ca o conditie falsa.


Complex_if_statement
Manipularea sir caractere prin if statement si elif
Demonstreaza utilizarea sintaxei if-elif
Ion Studentul 1/13/03
print("\tSalut! \nAceasta e un program de conversie Euro

CONDITII MULTIPLE SIMULTAN

- In cazul în care avem mai multe condiții de făcut pentru a putea aplica anumite schimbări pentru o anumită variabilă avem la dispoziție trei soluții.
- Prima solutie se refera la if-uri separate. Spre exemplicare vom utiliza cazul in care avem de testat doua conditii (conditie1 si conditie2). In cazul in care amandoua sunt adevarate dorim sa rulam doua expresii (expresie1 si expresie2).

```
If (conditie):
        expresie


if (conditie2):
        expresie2
```

Prima soluție se refera la folosirea a doua if separate. Aceasta solutie trebuie adaptata si nu functioneaza pentru toate cazurile deoarece ar putea sa ruleze doar **expresie2** fara ca **expresie1** sa ruleze.

CONDITII MULTIPLE SIMULTAN

- A doua solutie se refera la constructii de tip IF in IF. Spre exemplicare vom utiliza cazul in care avem de testat doua conditii (conditie1 si conditie2). In cazul in care amandoua sunt adevarate dorim sa rulam doua expresii (expresie1 si expresie2).
- A doua solutie solutie nu trebuie adaptata si functioneaza pentru toate cazurile deoarece nu ar putea sa ruleze doar expresie2 fara ca expresie1 sa ruleze.

```
If (conditie):
        if (conditie2):
                expresie
                expresie2
```

CONDITII MULTIPLE SIMULTAN

- A treia soluție se refera la folosirea unui singur if, dar cu condiții multiple folosind and. A treia solutie are acelasi efect ca si a doua solutie. Cuvântul cheie and are rolul de a uni doua condiții. Pt. a rula blocul de expresii format din expresie1 si expresie2 trebuie ca ambele condiții să fie adevărate .

```
If (conditie and conditie2):
        expresie
        expresie2
```

CONDITII MULTIPLE SIMULTAN

- TABEL AND - Toate conditiile trebuie sa fie adevarate pt. a rezulta o conditie adevarata

a == "conditie1"	b == "conditie2"	a == "conditie1" and b == "conditie2"
true	**true**	true
true	**false**	false
false	**true**	false
false	**false**	false

CONDITII MULTIPLE SIMULTAN

- Daca doar una din ele trebuie sa fie adevarata.

```
If (conditie or conditie2):
        expresie
        expresie2
```

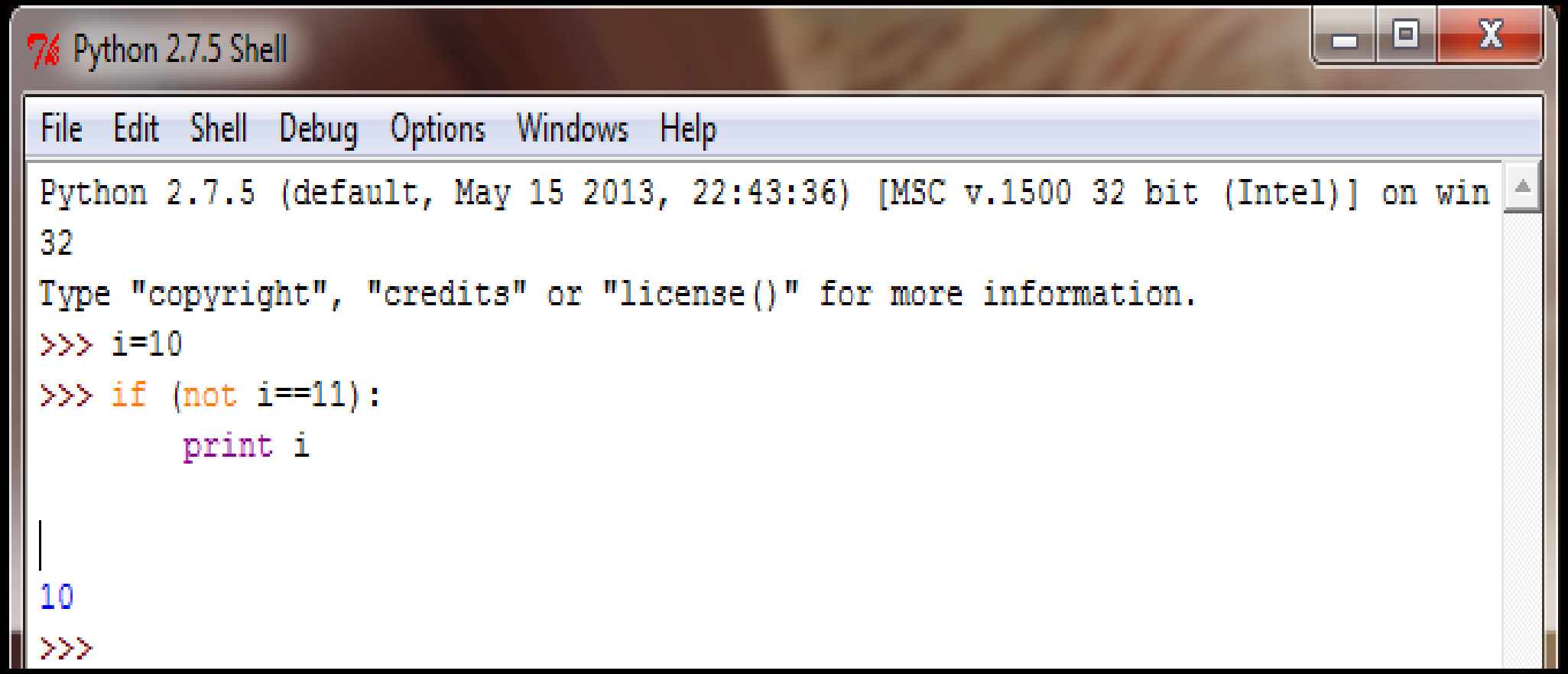
CONDITII MULTIPLE SIMULTAN

- Taber OR - Doar una din ele trebuie sa fie adevarata pt. a rezulta o conditie adevarata

a == "conditie1"	b == "conditie2"	a == "conditie1" or b == "conditie2"
true	true	true
true	false	true
false	true	true
false	false	false

OPERATORUL NOT

- negarea conditiei



OPERATORUL NOT

- negarea conditiei

i==10	not i==10
true	false
false	true

OPERATORUL NOT

- Not funcționează prin a nega expresia i==11. Prin urmare not i==11 este egal cu i!=11.

```
>>> x=100
>>> if(x!=10):
        print "Salut"


Salut
>>> if(not x==10):
        print "Salut"


Salut
>>>
```

WHILE SI FOR

- Buclarea cat timp condiția este adevărata. Rularea blocului de sintaxe se va opri cand conditia este falsa.

```
while (conditie):
        sintaxa1
        sintaxa2
```

WHILE SI FOR

- Buclarea pana cand conditia nu mai este indeplinita

```
>>> x = 10
>>> while (x>1):
        print "x este",x
        x = x-1


x este 10
x este 9
x este 8
x este 7
x este 6
x este 5
x este 4
x este 3
x este 2
>>>
```

WHILE SI FOR

```
>>> x = 12
>>> while (x > 1):
        print "ceva\n"



ceva

ceva

ceva

ceva
```

- O bucla infinita este atunci când condiția este mereu îndeplinita. Mare atenție la buclele infinite. Spre exemplu dacă omiteam in exemplu de mai sus sa decrementam valoarea lui x conditia este respectata mereu, deci cream o bucla infinita.
- Putem sa cream o bucla infinita si daca condiția este înlocuita cu variabilă boolean True.Si in acest caz sintaxele de sub while vor rula la infinit.

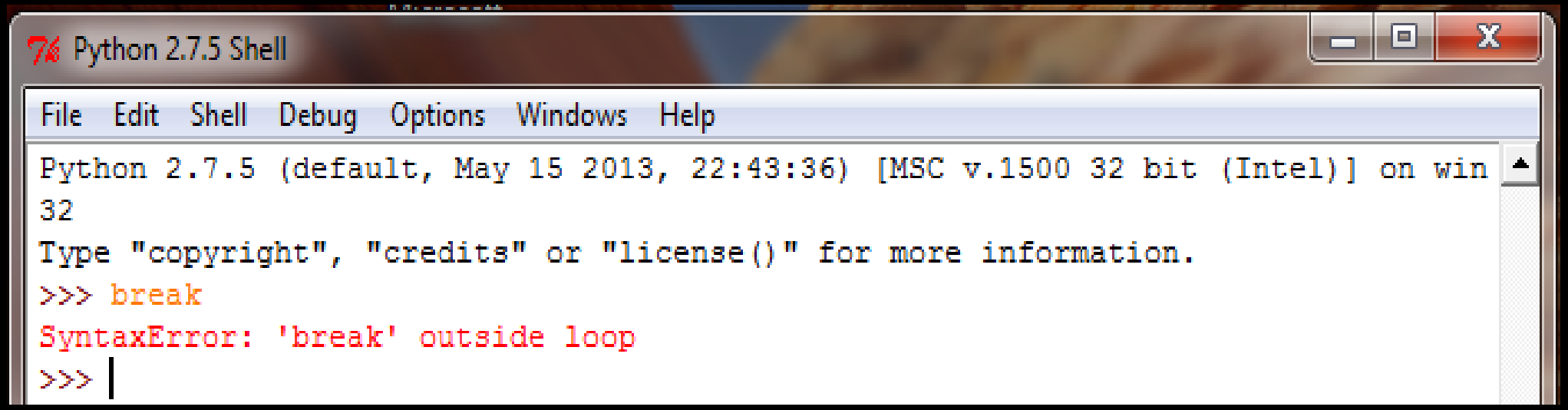
WHILE SI FOR

```
>>>
>>> x = 10
>>> while (True):
        print "1"
        if (x == 1):
                break
        x = x - 1


1
1
1
1
1
1
1
1
1
1
1
>>>
```

- Pentru a opri o bucla infinita avem nevoie de comanda speciala ce poarta numele de break ce are rolul de a implementa o terminare forțata a procesului de rulare.
- Putem utiliza True ca si conditie daca folosim cuvantul cheie break

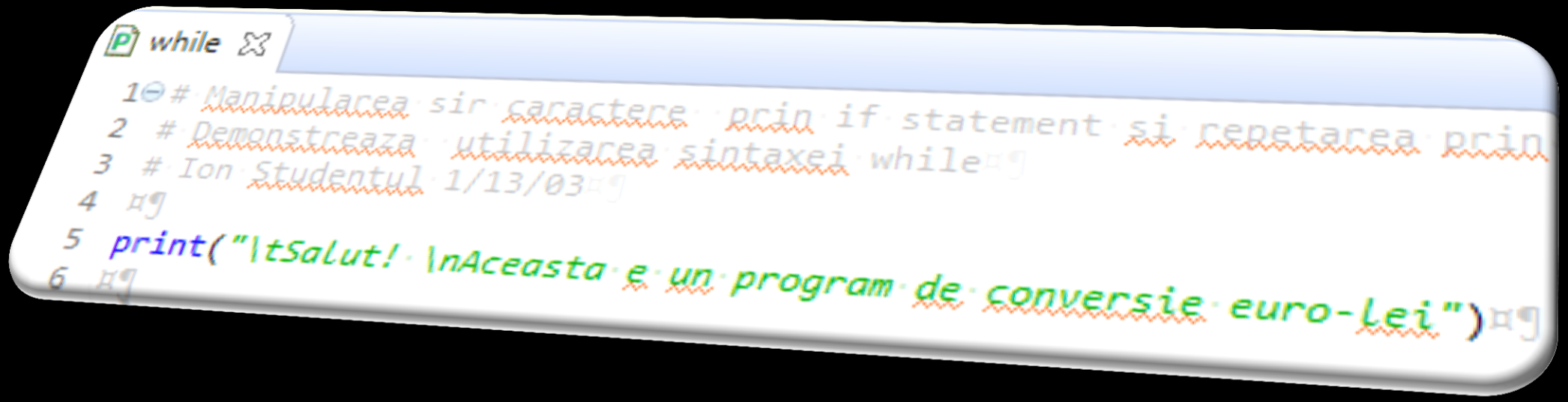
WHILE SI FOR

- Aveți grija cu utilizarea break deoarece poate genera erori când nu există o buclă care poate fi întreruptă.



WHILE SI FOR

- Exemplu



WHILE SI FOR

- Sintaxa pass = nu face nimic!
- Iata un alt exemplu de utilizare a pass. Acest exemplu a fost explicat anterior. Putem observa ca cuvantul cheie pass nu schimba cu nimic comportamentul anterior al sintaxei while.

```
>>> x = 10
>>> while (x>1):
        pass
        print "x este",x
        x = x-1


x este 10
x este 9
x este 8
x este 7
x este 6
x este 5
x este 4
x este 3
x este 2
>>>
```

WHILE SI FOR

- Comanda continue are rolul de a face un salt la rularea respectiva fara a intrerupe bucla. Prin urmare va întrerupe rularea blocului de sintaxe pentru rularea dată si se va incepe rularea de la inceput a blocului de expresii de sub while.

```
>>> x = 10
>>> while (x>1):
        print "x este",x
        continue
        x = x-1


x este 10
x este 10
x este 10
x este 10
x este 10
x este 10
x este 10
x este 10
x este 10
x este 10
x este 10
x este 10
x este 10
x este 10
x este 10
x este 10
x este 10
x este 10
x este 10
x este 10
x este 10
```

WHILE SI FOR

- FOR = bucleaza printr-o range dat; lista data; tuplu dat; dictionary dat sau sir de caractere date.

```
for variabila_iterare in lista:
    bloc_de_sintaxe #ce poate sau nu să foloseasca variabila iterare
```

Încă mai sunt disponibile cuvintele cheie continue, pass și break .

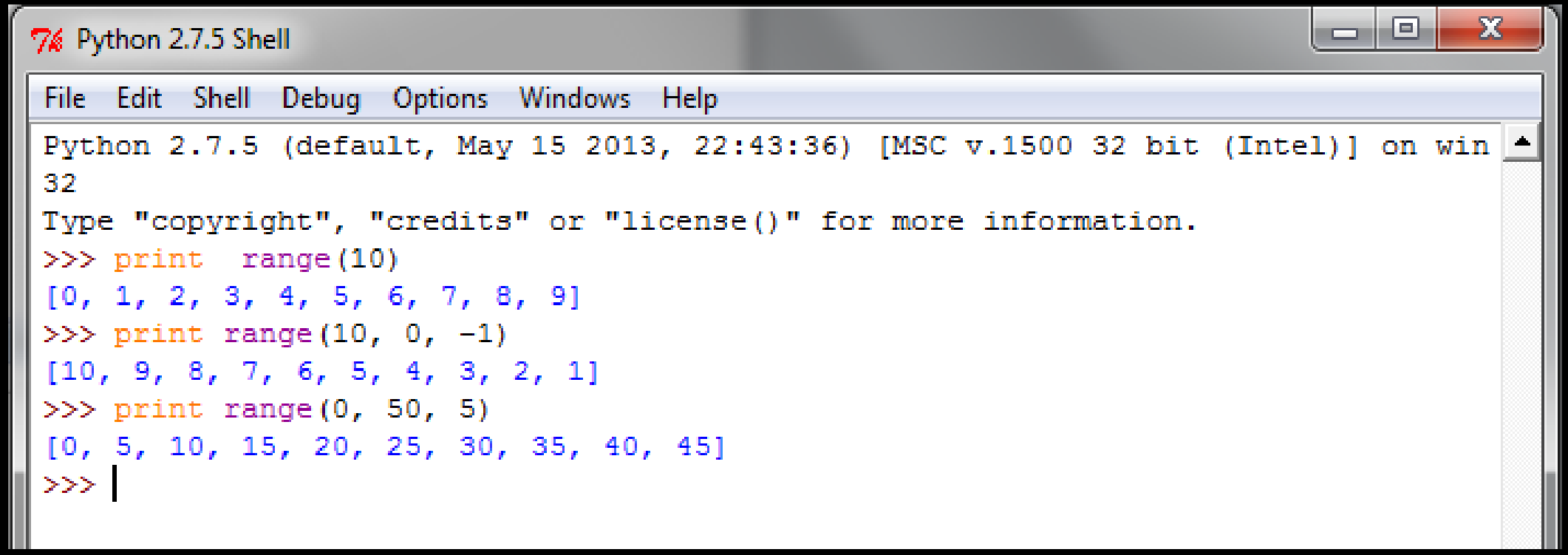
WHILE SI FOR

- FOR = bucleaza printr-un sir de caractere dat

```
>>>
>>> for var in "abcd":
        print var



a
b
c
d
```

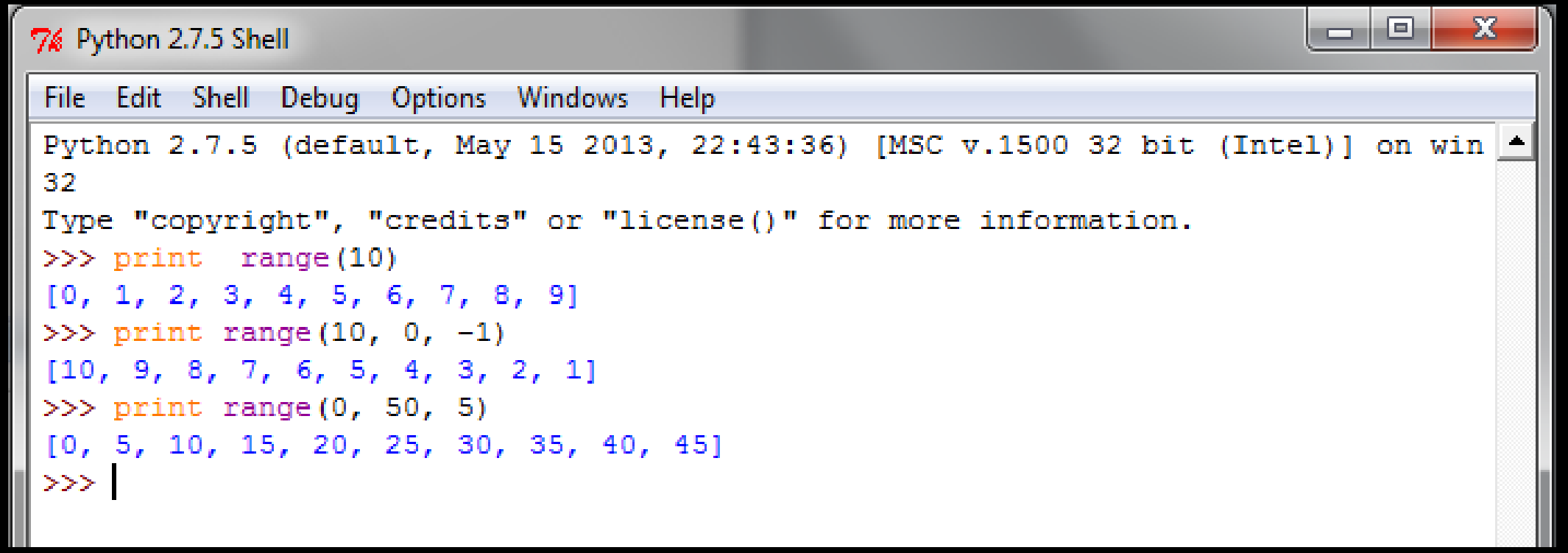
WHILE SI FOR

- Lista este un tip de date nestudiat inca.
- Pentru a crea liste vom utiliza sintaxa range()
- range (stop)
- range(start,stop, [pas])



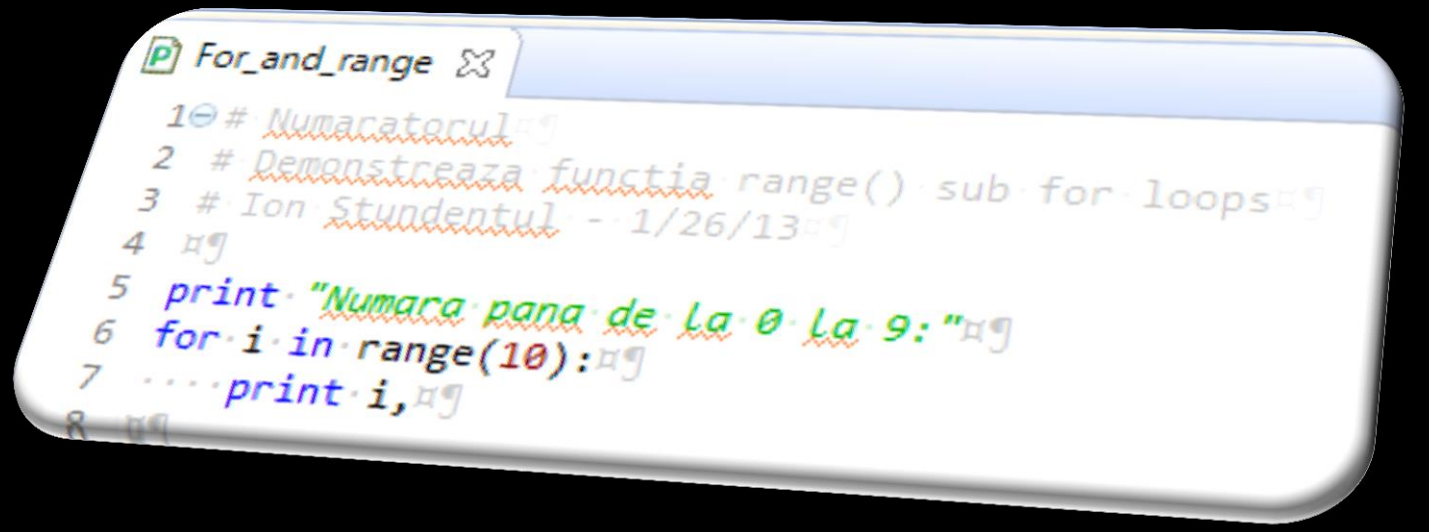
WHILE SI FOR

- Lista este un tip de date nestudiat inca.
- Pentru a crea liste vom utiliza sintaxa range()
- range (stop)
- range(start,stop, [pas])

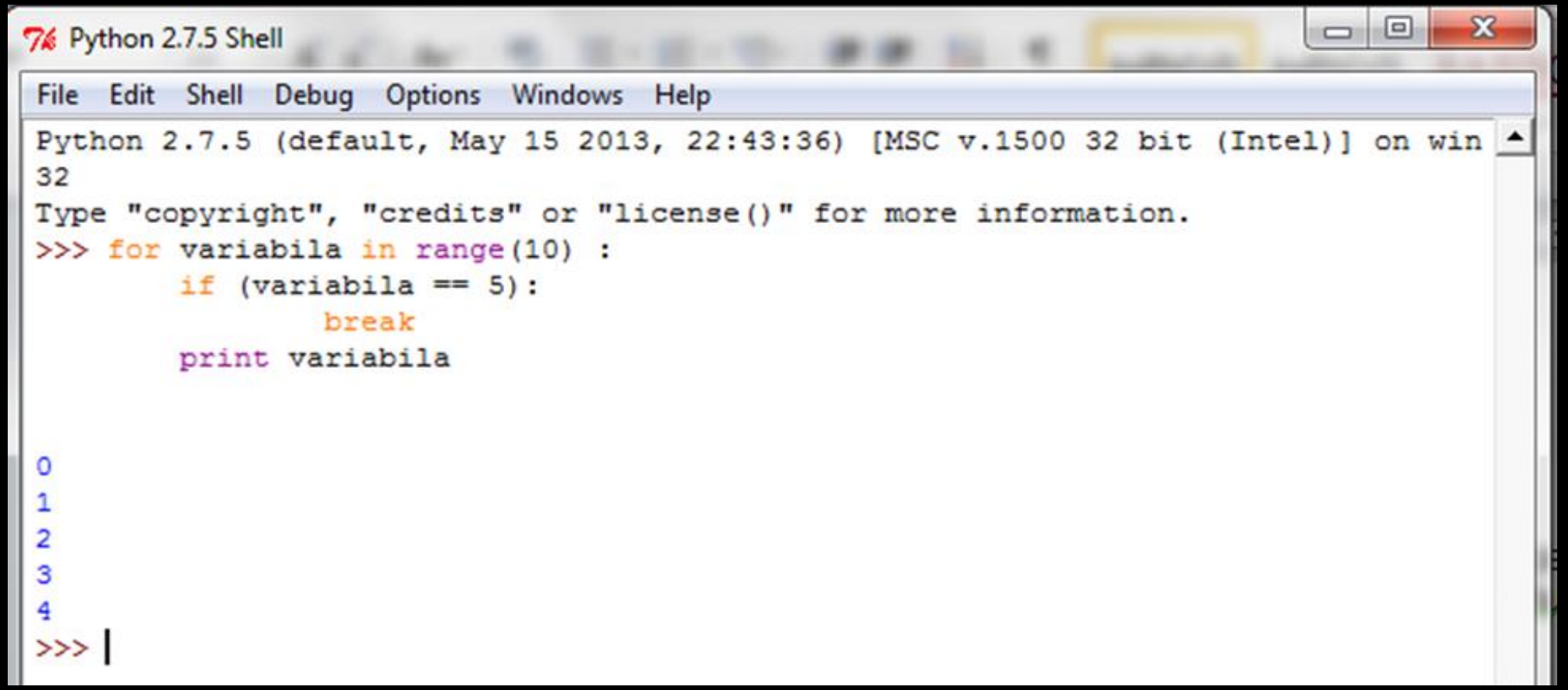


WHILE SI FOR

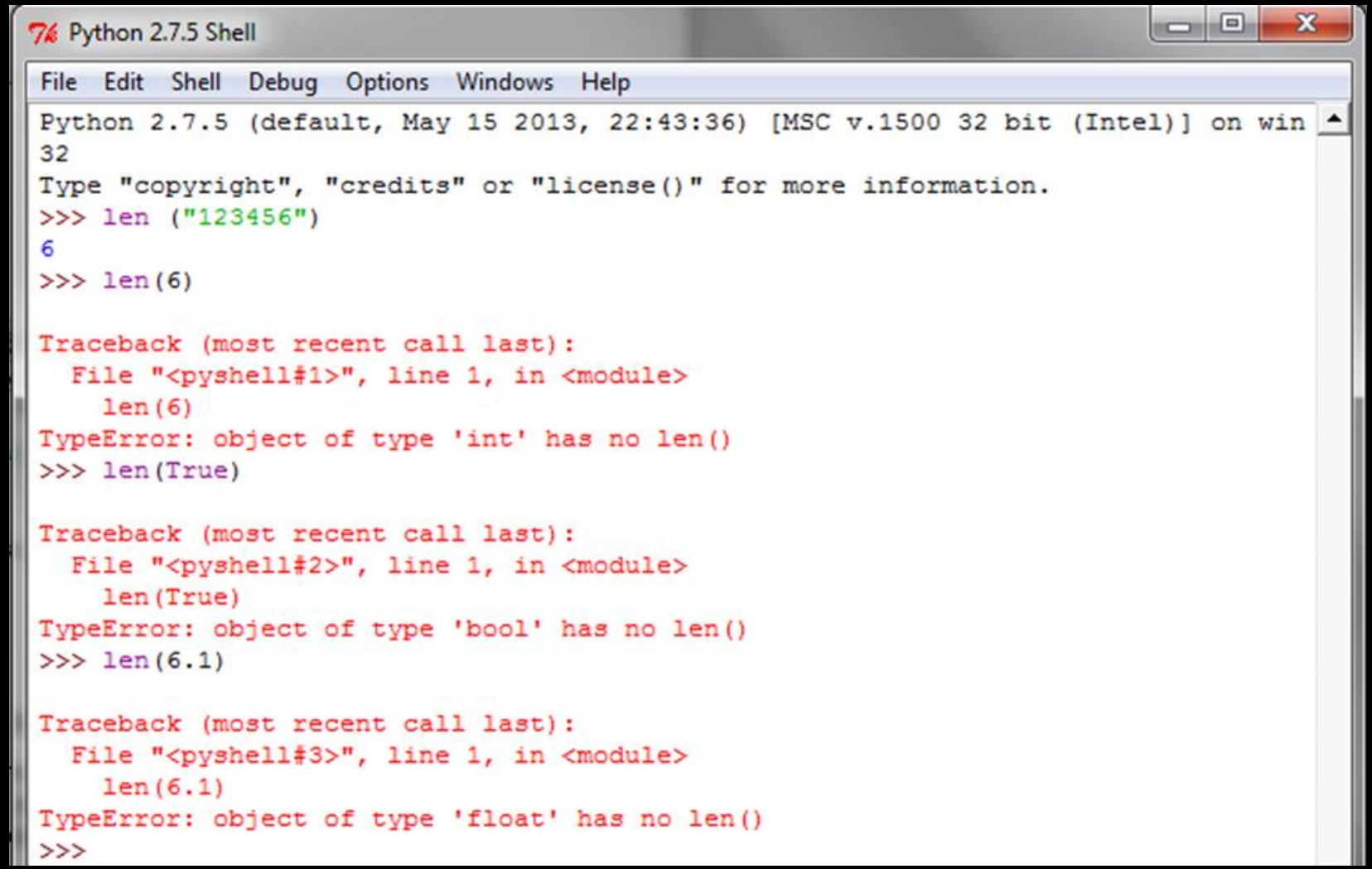
- Exemplu program For



WHILE SI FOR


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.
>>> for variabila in range(10) :
if (variabila == 5):
break
print variabila
0
1
2
3
4
>>>

SIRURI DE CARACTERE AVANSAT



len() ne ofera numarul de caractere dintr-un sir de caractere

WHILE SI FOR

- Exista o formă prescurtată de calcul matematic cand trebuie sa realizez o operatie matematica cu o varialabila si sa stochez rezultatul tot in aceasta variabila.

```
>>> x = 2
>>> x = x+2
>>> print x
4
>>> x = 2
>>> x+=2 # aceasta linie se poate traduce ca x=x+2 si reprezinta acelasi lucru
>>> print x
4
```

WHILE SI FOR

OBS.

- Putem scrie in loc de

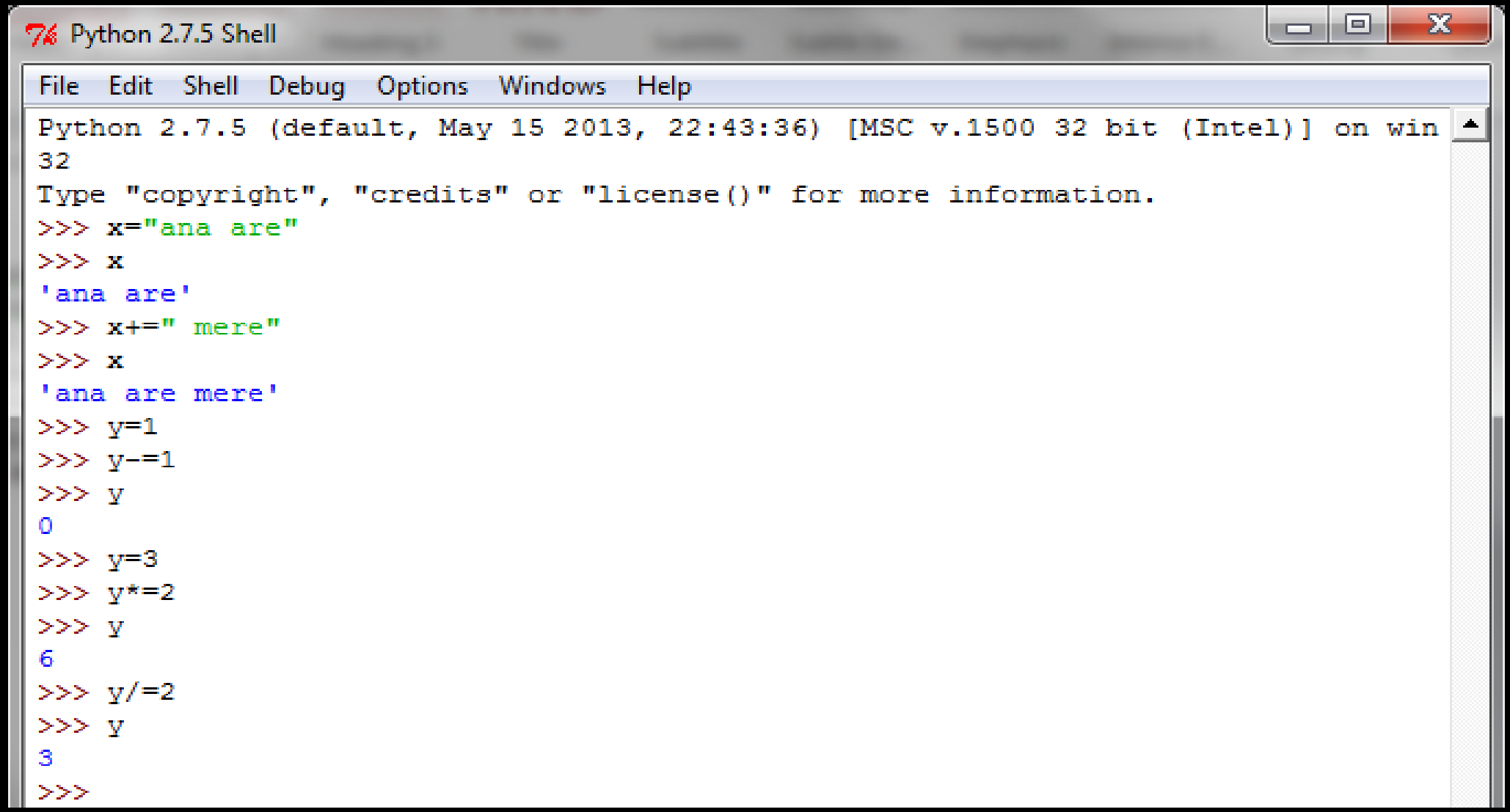
x = x +1 => x +=1

- Putem scrie in loc de

x = x -1 => x -=1

```
>>> a = 1
>>> a -=1
>>> print a
0
>>> a+=1
>>> print a
1
>>>
```

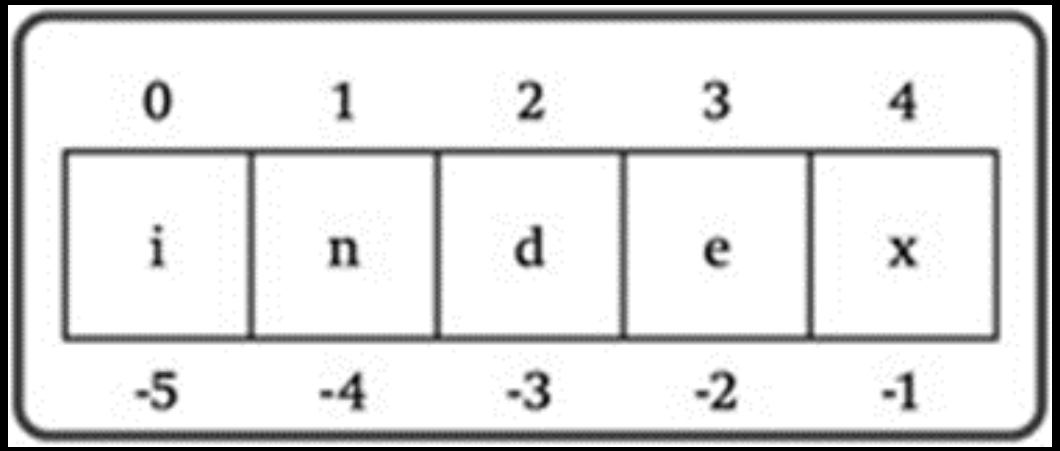
WHILE SI FOR


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="ana are"
>>> x
'ana are'
>>> x+=" mere"
>>> x
'ana are mere'
>>> y=1
>>> y-=1
>>> y
0
>>> y=3
>>> y*=2
>>> y
6
>>> y/=2
>>> y
3
>>>

SIRURI DE CARACTERE AVANSAT

Indexarea unui sir de caractere:

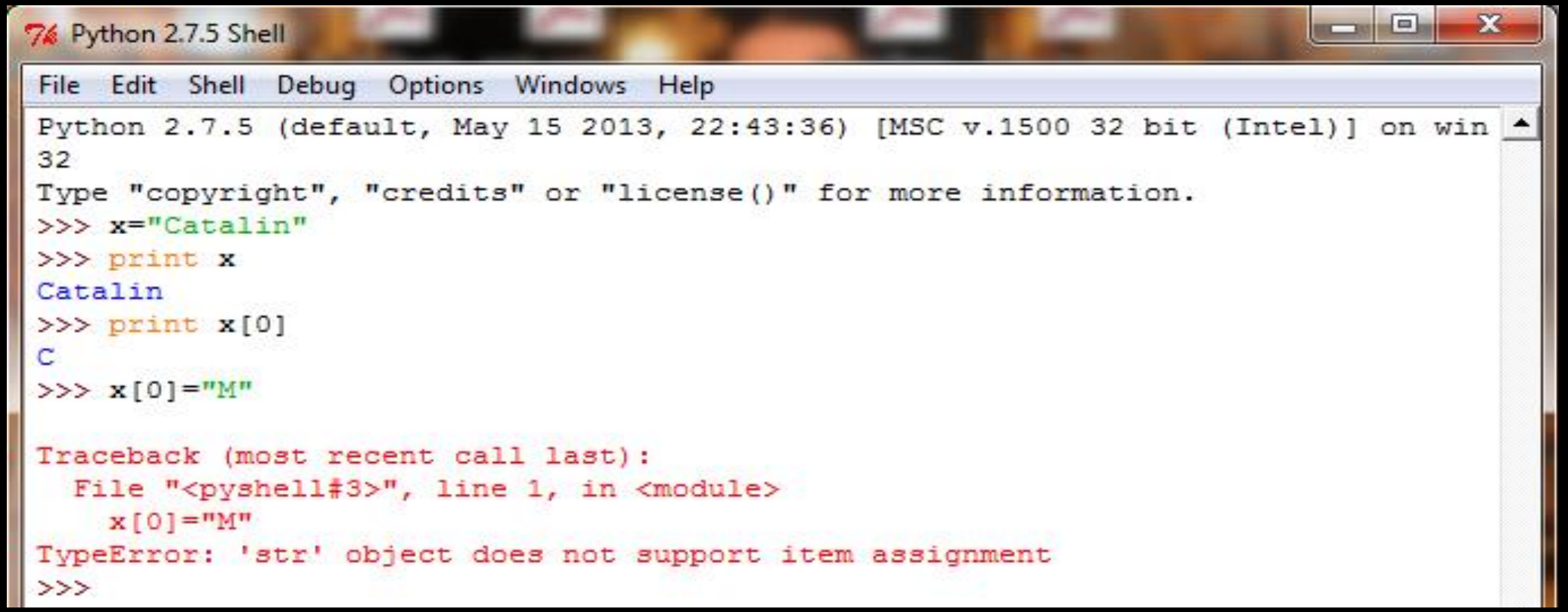
```
>>> cuvant = "12345"
>>> cuvant[0]
'1'
>>> cuvant[1]
'2'
>>> cuvant[2]
'3'
>>> cuvant[3]
'4'
```




```
>>> cuvant[10]

Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    cuvant[10]
IndexError: string index out of range
>>>
```

SIRURI DE CARACTERE AVANSAT

Un sir de caractere este un element imutabil. Nu pot folosi indexarea pentru a modifica sirul.

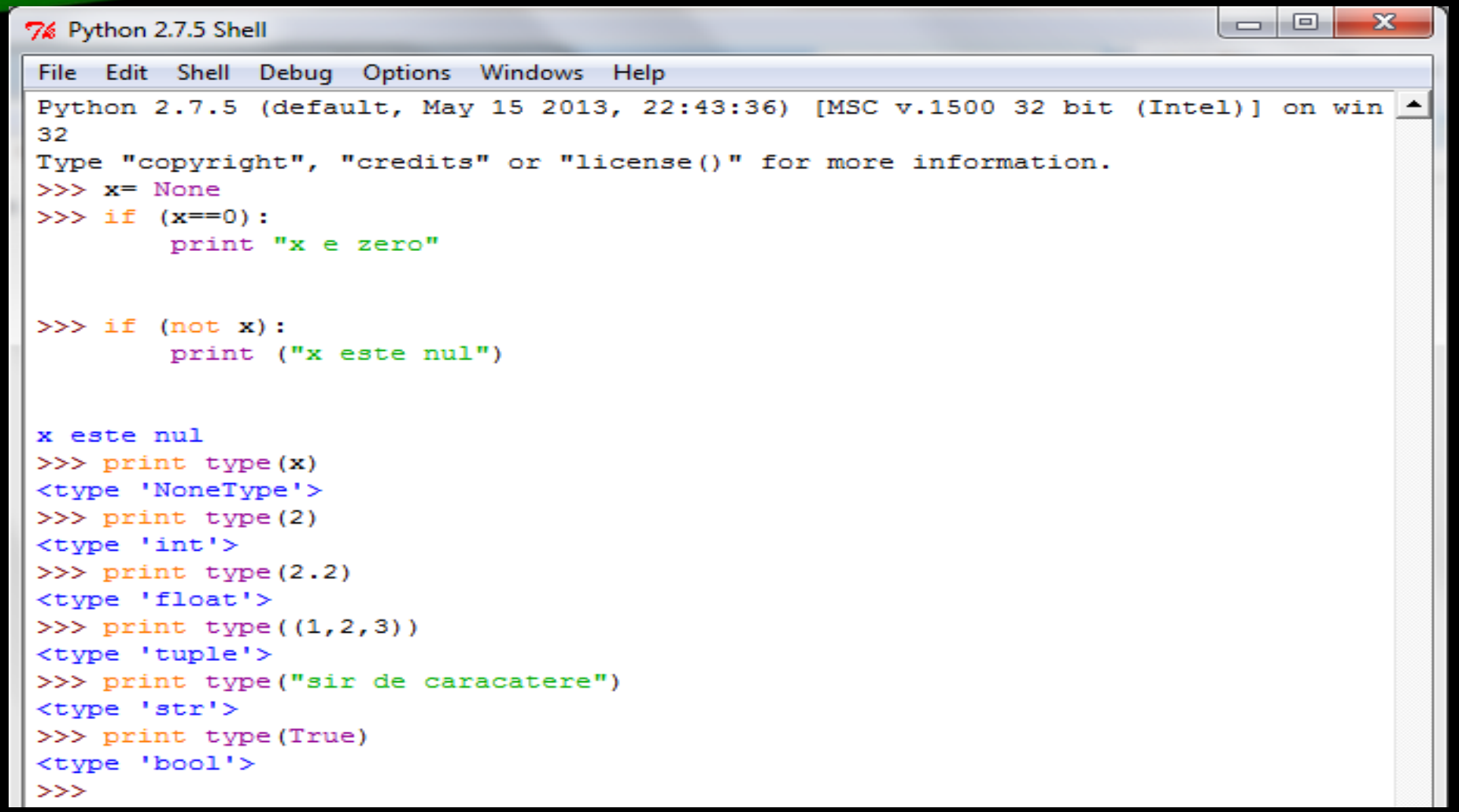


SIRURI DE CARACTERE AVANSAT

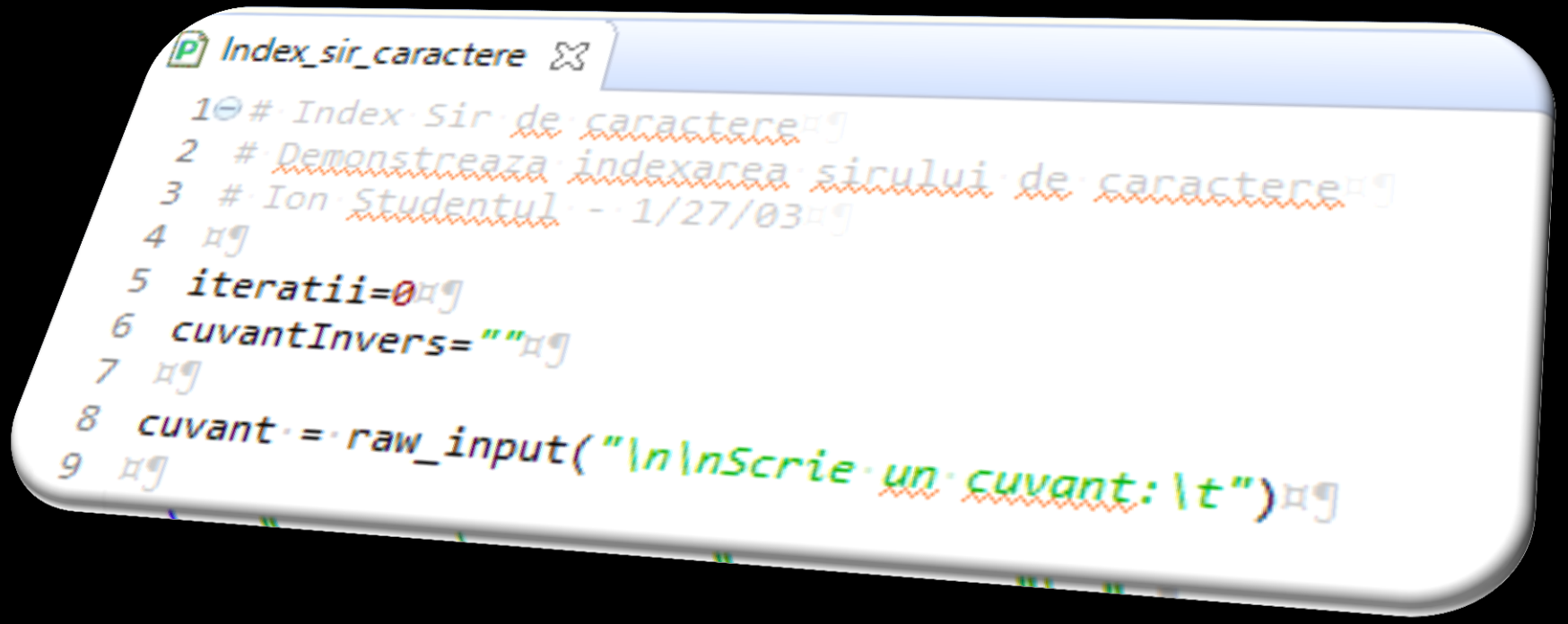
None are scopul de a putea inițializata o variabila, dar fără a susține nici o valoare. La testarea ei ca si condiție va returna fals.

```
>>> x = None
>>> x-2

Traceback (most recent call last):
  File "<pyshell#47>", line 1, in <module>
    x-2
TypeError: unsupported operand type(s) for -: 'NoneType' and 'int'
>>> x+"2"

Traceback (most recent call last):
  File "<pyshell#48>", line 1, in <module>
    x+"2"
TypeError: unsupported operand type(s) for +: 'NoneType' and 'str'
>>> float(x)

Traceback (most recent call last):
  File "<pyshell#49>", line 1, in <module>
    float(x)
TypeError: float() argument must be a string or a number
>>>
```

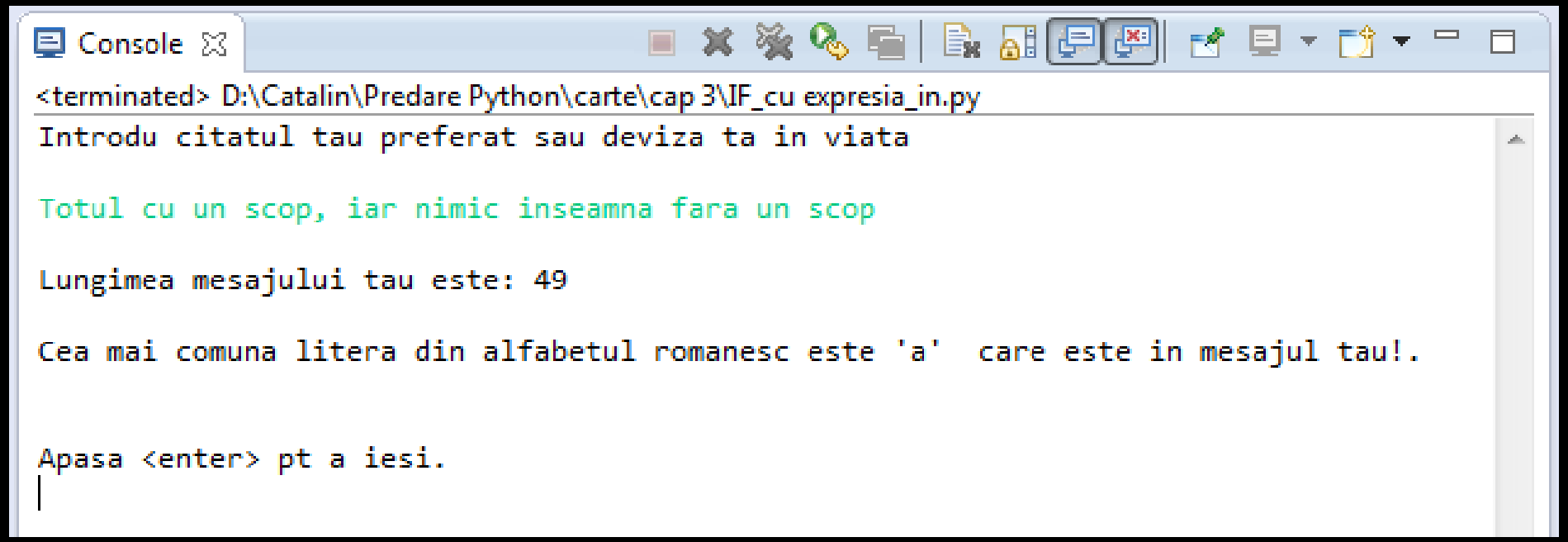
SIRURI DE CARACTERE AVANSAT



SIRURI DE CARACTERE AVANSAT



SIRURI DE CARACTERE AVANSAT



SIRURI DE CARACTERE AVANSAT

```
>>> x = "sir de caractere"
>>> x.replace("r","c")
'sic de cacactece'
>>> print x
sir de caractere
>>>
```

SIRURI DE CARACTERE AVANSAT

```
>>> #x.replace(old,new)
>>> x.find("a")
8
>>> x[8]
'a'
>>> #prima oara unde a fost gasit elementul
>>>
>>>
>>> x.find("r")
2
>>>
#x este neschimbat pt a salva x=x.replace(old,new)
>>> print x
sir de caractere
>>> x.find("z")
-1
>>> #elementul nu este gasit
>>> #deci  returneaza -1
>>> x.split(" ")
['sir', 'de', 'caractere']
>>> #returneaza o lista impartind dupa elementul din interior
>>> x.split("r")
['si', ' de ca', 'acte', 'e']
>>> # se vede ca elimina acel caracter
```

TUPLU

Tuplu reprezintă o serie de secvențe care pot conține numere, șiruri , sau alte tuple.

x=( )

Iată mai jos un tuplu care susține alte tipuri de variabile:

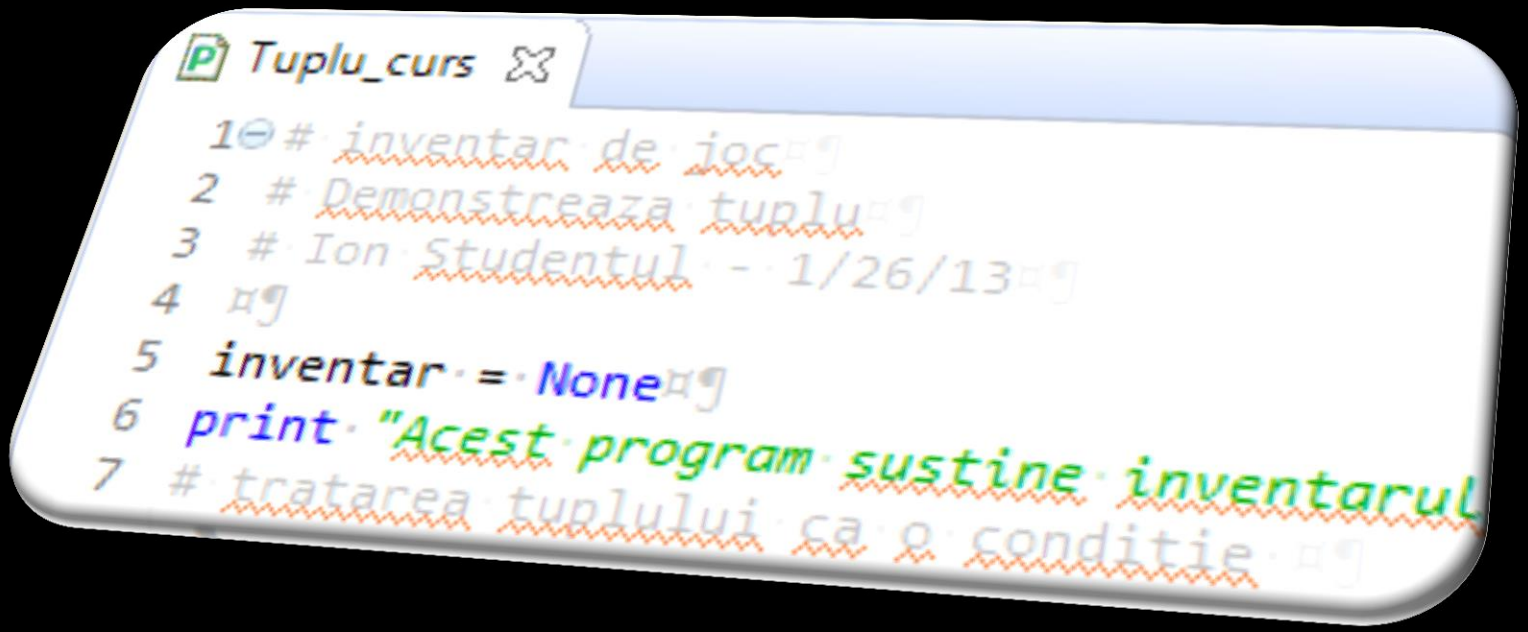
x=(True, False, 4,5,6,7,"sir de caractere",("tuplu 2",2))

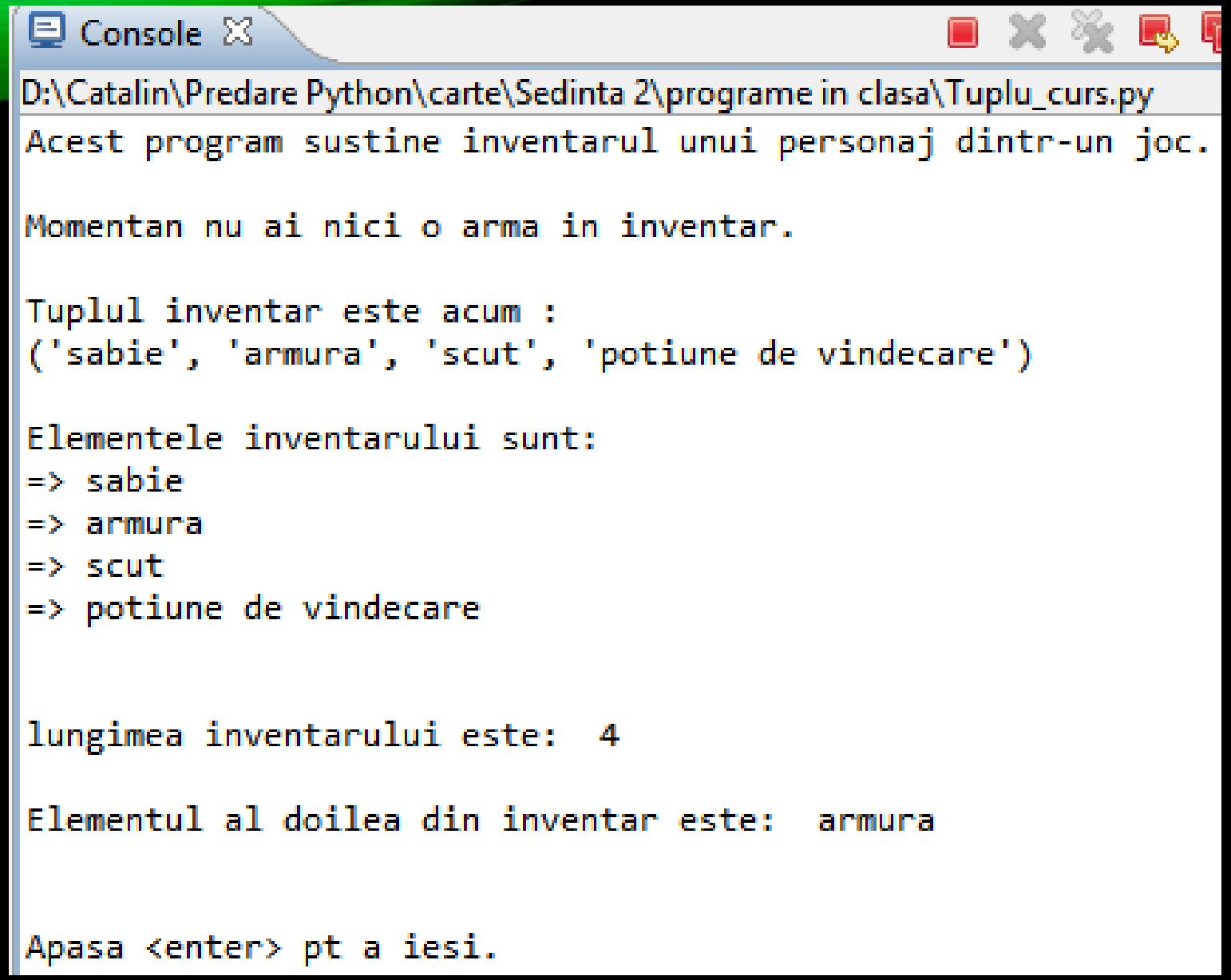
TUPLU

Tuplu este un tip de variabilă imutabila.

```
>>> x=("a","b","d")
>>> x[2]
'd'
>>> x[2]="c"

Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    x[2]="c"
TypeError: 'tuple' object does not support item assignment
>>>
```

Exemplu tuplu:




Console
D:\Catalin\Predare Python\carte\Sedinta 2\programe in clasa\Tuplu_curs.py
Acest program sustine inventarul unui personaj dintr-un joc.
Momentan nu ai nici o arma in inventar.
Tuplul inventar este acum :
('sabie', 'armura', 'scut', 'potiune de vindecare')
Elementele inventarului sunt:
=> sabie
=> armura
=> scut
=> potiune de vindecare
lungimea inventarului este: 4
Elementul al doilea din inventar este: armura
Apasa <enter> pt a iesi.

LISTA

Tuplu reprezintă o serie de secvențe care pot conține numere, șiruri , liste sau tuple.

x=[]

Iată mai jos o lista care susține alte tipuri de variabile:

x=[True, False, 4,5,6,7,"sir de caractere",("tuplu 2",2),[1,2,43]]

LISTA

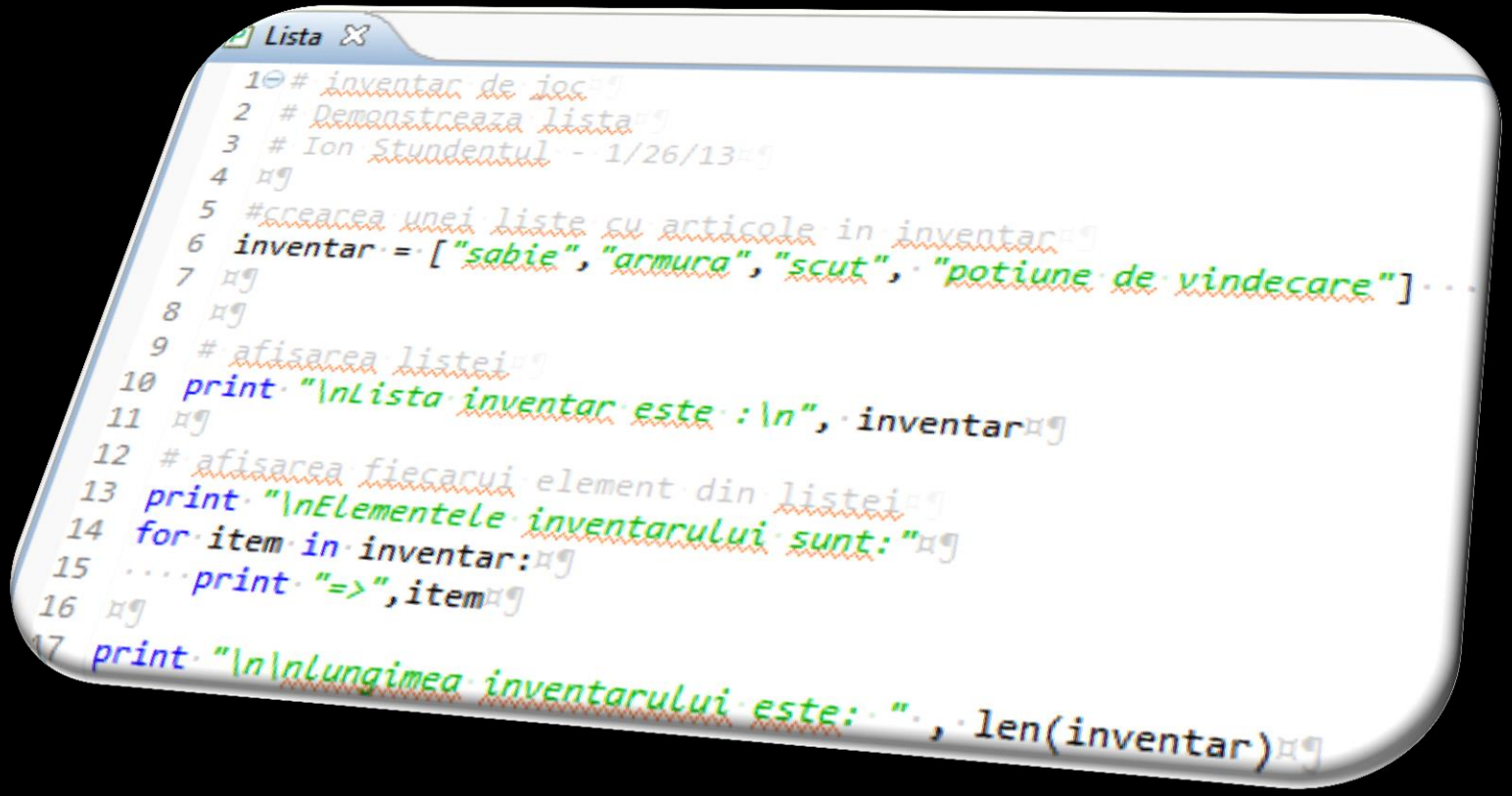
O lista este un alt tip de structura de date

```
>>> cufar=["lotiune magica"]
>>> type(cufar)
<type 'list'>
>>> inventar = ["sabie","armura","scut", "potiune de vindecare"]

>>> type(inventar)
<type 'list'>
```

Are proprietatea de mutabilitate si indexare:

```
>>>
>>> x = [1,2,"3",4.0]
>>> x
[1, 2, '3', 4.0]
>>> x[1] = "2.0"
>>> x
[1, '2.0', '3', 4.0]
>>>
```

LISTA


Lista
inventar de joc
Demonstreaza lista
Ion Stundentul - 1/26/13
#creeacea unei liste cu articole in inventar
inventar = ["sabie","armura","scut", "potiune de vindecare"]
# afisarea listei
print "\nLista inventar este :\n", inventar
afisarea fiecarui element din listei
print "\nElementele inventarului sunt:"
for item in inventar:
print "=>",item
print "\n\nlungimea inventarului este: ", len(inventar)

LISTA

Metoda	Descriere
append(value)	adauga value la finalul listei.
sort()	Sorteaza elementele dupa ASCII.
reverse()	Reinverseaza ordinea listei. Primul element devine ultimul, al doilea devine penultimul etc...
count(value)	Returneaza nr. de dati cand o valoare (value) este intalnita.
index(value)	Returneaza prima pozitie cand value apare în lista. Returneaza eroare daca nu este intalnita.
insert(i, value)	Insereaza value la pozitia i.
pop([i])	Returneaza value pozitia i și sterge valoarea value din lista. E optionala numarul pozitie i și poate fi omis ; în acest caz ultimul element va fi sters și returnat.
remove(value)	Sterge prima intrare pe care o intalneste cu valoarea value din lista.

LISTA

Funcția sort aplicata listei în sintaxă lista.sort() are rolul de a sorta elementele listei într-o ordine alfanumerica. Aranjarea unei liste se realizează astfel mai întâi numerele, apoi caractere speciale sau numere de tip șir de caractere, apoi celelalte șiruri de caractere ce încep cu litere în ordine alfabetica.

```
>>>
>>> lista =['a',"c",'b',1,"1","@","!"]
>>> lista.sort()
>>> print lista
[1, '!', '1', '@', 'a', 'b', 'c']
>>>
```

LISTA

```
>>> x=["0",1,"2",None,["4",5],(6,7)]
>>> x.sort()
>>> x
[None, 1, ['4', 5], '0', '2', (6, 7)]
>>> x.reverse()
>>> x
[(6, 7), '2', '0', ['4', 5], 1, None]
>>> x.append(None)
>>> x
[(6, 7), '2', '0', ['4', 5], 1, None, None]
>>> x.pop(6)
>>> x
[(6, 7), '2', '0', ['4', 5], 1, None]
>>> x.insert(3,None)
>>> x
[(6, 7), '2', '0', None, ['4', 5], 1, None]
```

```
>>> x
[(6, 7), '2', '0', None, ['4', 5], 1, None]
>>> x.remove(None)
>>> x.remove("4")

Traceback (most recent call last):
  File "<pyshell#83>", line 1, in <module>
    x.remove("4")
ValueError: list.remove(x): x not in list
```

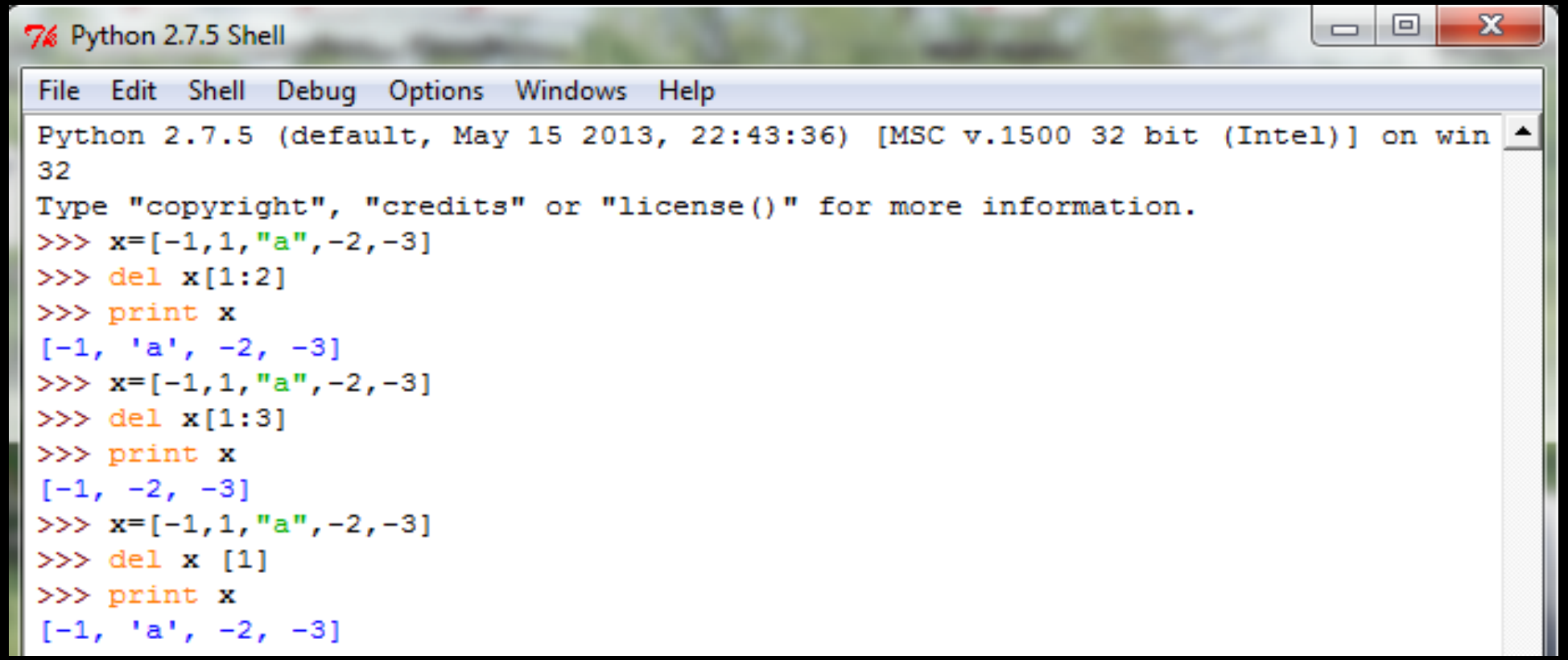
LISTA – NESTED SEQUENCE

secvențe imbricate (imbricat=suprapus parțial)
Accesul se face prin indexare repetata – nu se recomanda mai mult de un nivel.

```
>>> lista=[1,["al doilea",["infoacademy"]]]
>>> len(lista)
2
>>> lista[1]
['al doilea', ['infoacademy']]
>>> lista[0]
1
>>> lista[1][0]
'al doilea'
>>> lista[1][1]
['infoacademy']
>>> lista[1][1][0]
'infoacademy'
```

LISTA

Alta modalitate de a sterge un element : del - nu returneaza nimic



DICTIONAR

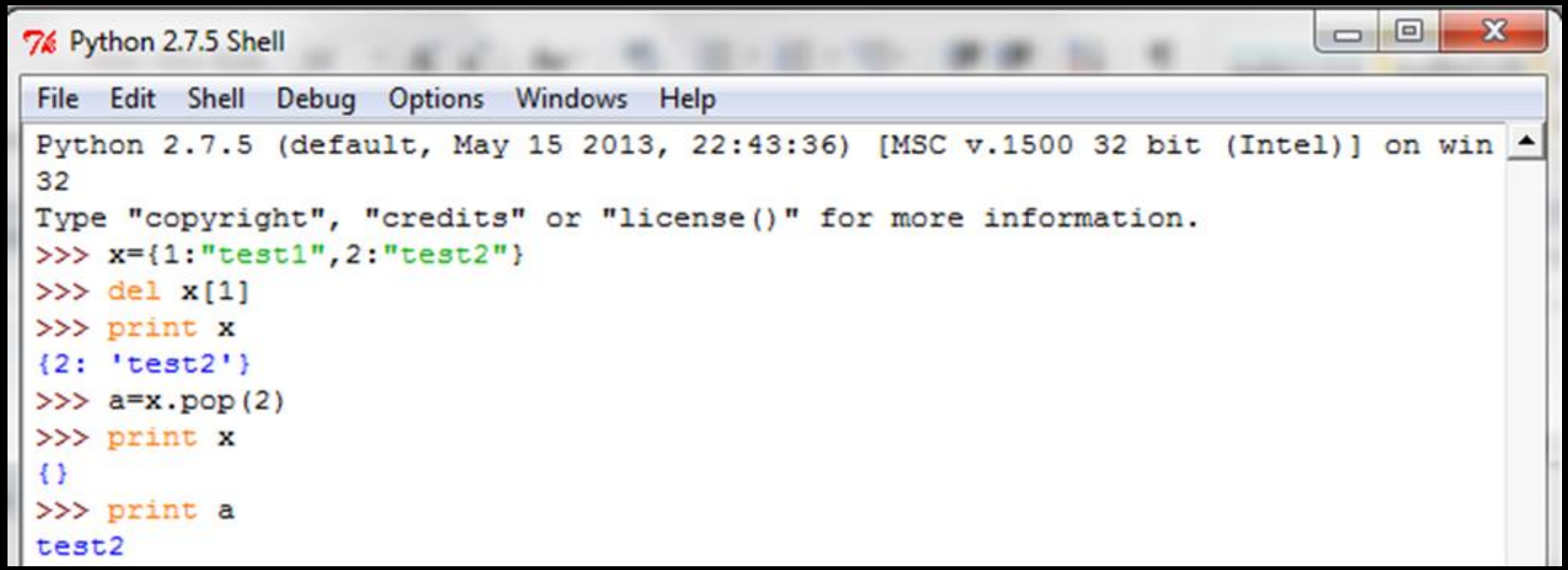
Structureaza mai bine datele

```
>>>
>>> a = {1:"abc","2":2}
>>> #nume={key:value}
>>>
```

DICTIONAR

Metoda	Descriere
has_key(key)	Returneaza true daca key se gaseste în dictionar ca și cheie, altfel returneaza false.
get(key, [default])	Returneaza valoarea cheii key.daca cheia nu este gasita atunci se returneaza cuvantul optional default. Daca cheia nu exista și cuvantul default nu este specificat, atunci se va returna None.
keys()	Returneaza o lista cu toate cheile din dictionar.
values()	Returneaza o lista cu toate valorile din dictionar.
items()	Returneaza o lista cu toate elementele din dictionar Fiecare element este un tuplu de doua elemnete de tip (cheie,valoare)..
pop(key)	Sterge elementul key:value din dictionar daca key exista. In cazul în care nu exista va returna eroare. Returneaza value.
del dictionar[key]	Sterge perechea key:value din dictionar daca key exista. In cazul în care nu exista va returna eroare. Nu returneaza nimic.

DICTIONAR DIFERENTA DINTRE POP SI DEL



DICTIONAR EXEMPLU

Dictionar

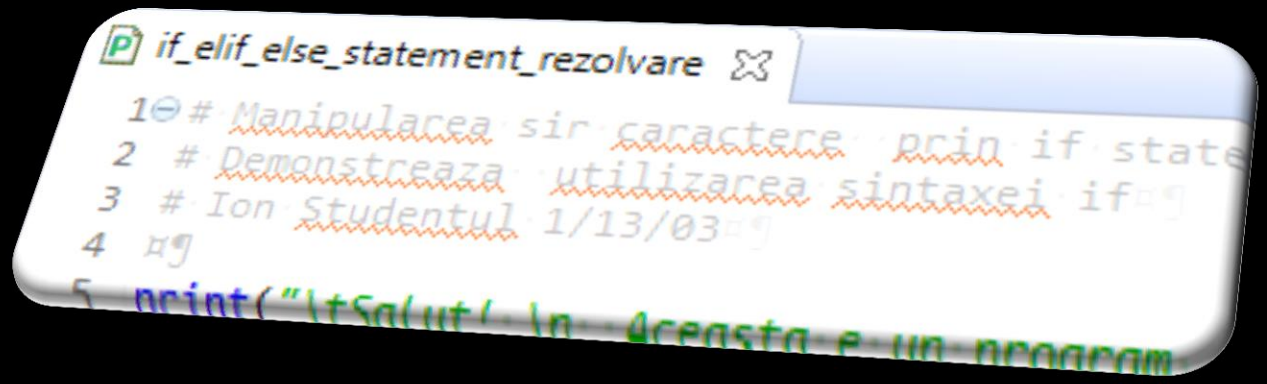
```
1  # Dictionar de facebook
2  # Demonstreaza dicitonar
3  # Ion Stundentul - 1/26/13
4  dictionar={"ASAP": "As Soon As Possible-Cat mai
5  "ASL": "Age Sex Location-Varsta, sex, locatie",
6  "BRB": "Be Right Back-Revin imediat",
7  "FYI": "For Your Info-Pentru informatia ta",
8  "LOL": "Laugh out Loud-Rad tare",
9  "PM": "Private Message-mesaj pe privat",
10 "FRUMI":"Frumos"}
11
12 print"\nAccesam dictionarul pentru a vedea ce cu
13 print dictionar.keys()
```

TEMA IN CLASA

Creati un progam care sa verifice daca textul introdus
de un utilizator de la tastatura este un sir de tip numere sau litere.

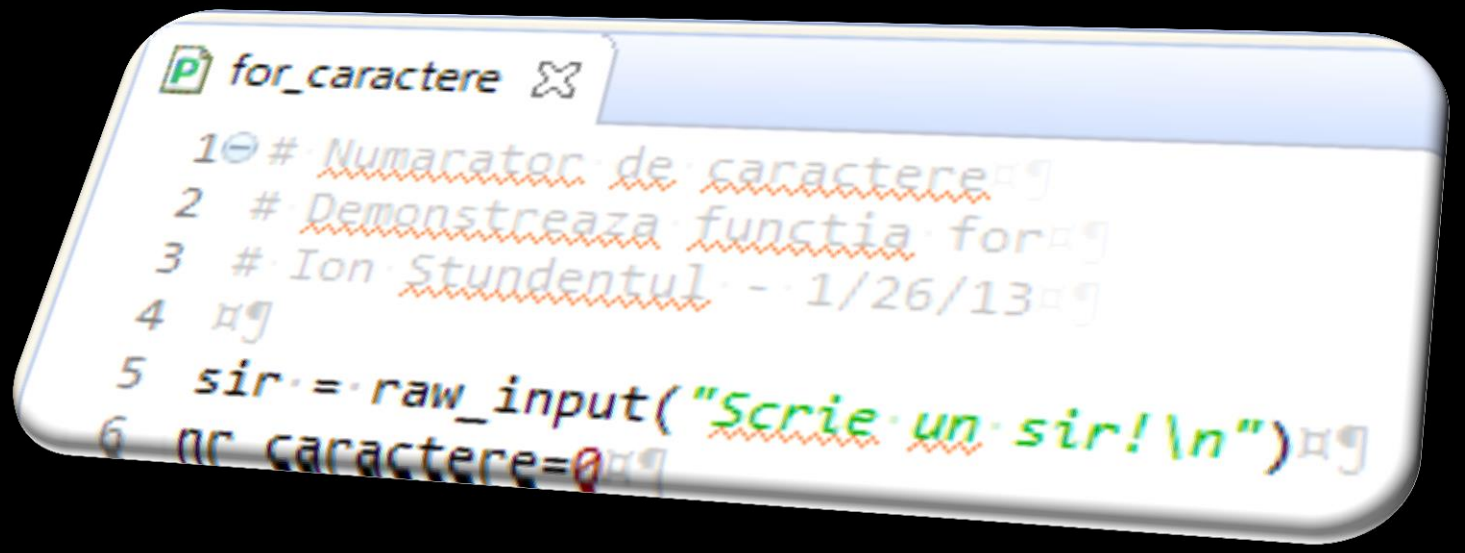
Va rog utilizati if-elif-else.
Trebue sa afisati urmatoarele siruri de caractere:

- Numar: “Sirul de caractere este format din numare”
- Litere: “Sirul de caractere este format din litere”
- Orice altceva: “Sirul de caractere este format din diferite elemente”



FOR

Scrieti un program ce va numara cate caractere are un sir de caractere dat de utilizator. Aceata numarare sa se realizeze cu ajutorul unui for. La final afisati rezultatul.



VA MULTUMESC PENTRU PARTICIPARE

La revedere!