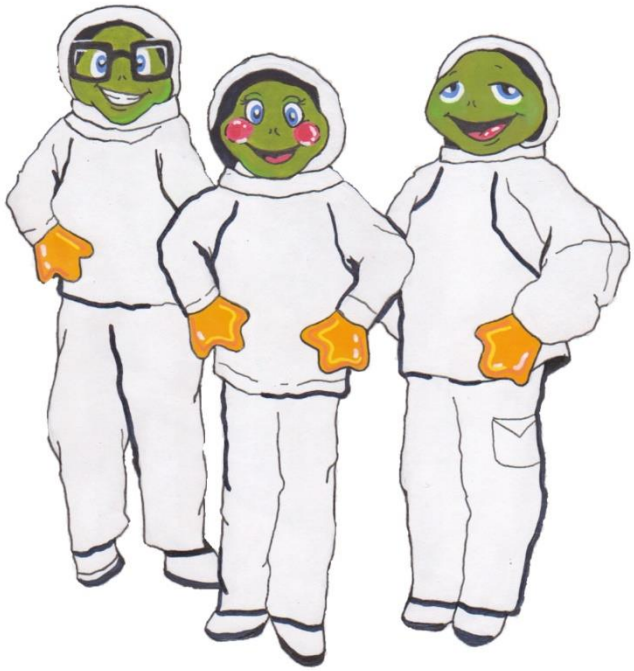
DOINA – LAVINIA PREDA

INTRODUCERE ÎN PROGRAMAREA ORIENTATĂ - OBIECT, PENTRU COPII, ÎN PYTHON



BUCUREȘTI, 2019

Această carte a fost înregistrată la Biblioteca Națională cu ISBN 978-973-0-30332-2.

CAPITOLUL I LISTE, TUPLURI ȘI DICȚIONARE

În acest capitol vom învăța:

- Noțiunile de listă, tuplu și dicționar;
- Să prelucrăm elementele listelor, tuplurilor și dicționarelor.

1.1 Liste în Python

De ce vorbim despre liste? Pentru ca le întâlnim peste tot. Lista cu exercițiile pe care le avem ca temă la matematică sau la limba română, lista cumpărăturilor pe care le avem de făcut, lista cadourilor pe care le dorim de Crăciun și multe altele.

Python ne ajută să reprezentăm liste și să le manipulăm conținutul așa că să vedem ce putem face cu ajutorul lui.

În majoritatea limbajelor de programare, o listă este o succesiune de elemente de același tip. În Python, elementele unei liste pot fi de același tip (liste omogene) sau de tipuri diferite (liste neomogene).

Exemple:

1. Presupunem ca dorim sa avem o lista a limbajelor de programare. În acest caz elementele listei vor avea același tip (lista omogenă).

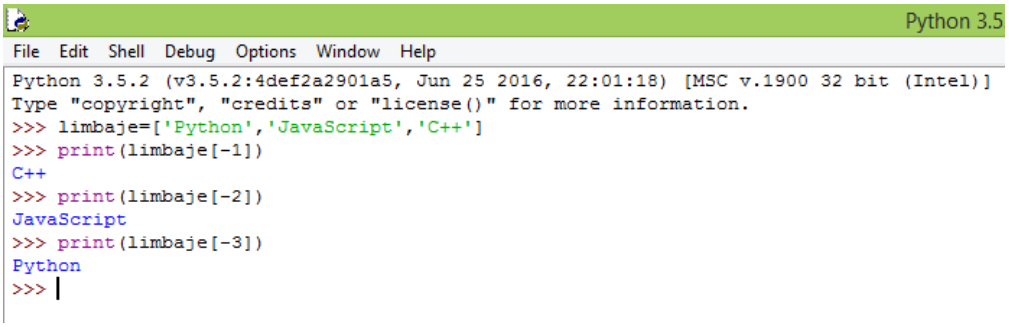
```
>>> limbaje=['Python','JavaScript','C++']
>>> print(limbaje)
['Python', 'JavaScript', 'C++']
>>>
```

Fiecare element al listei poate fi accesat printr-un indice numerotat de la 0.

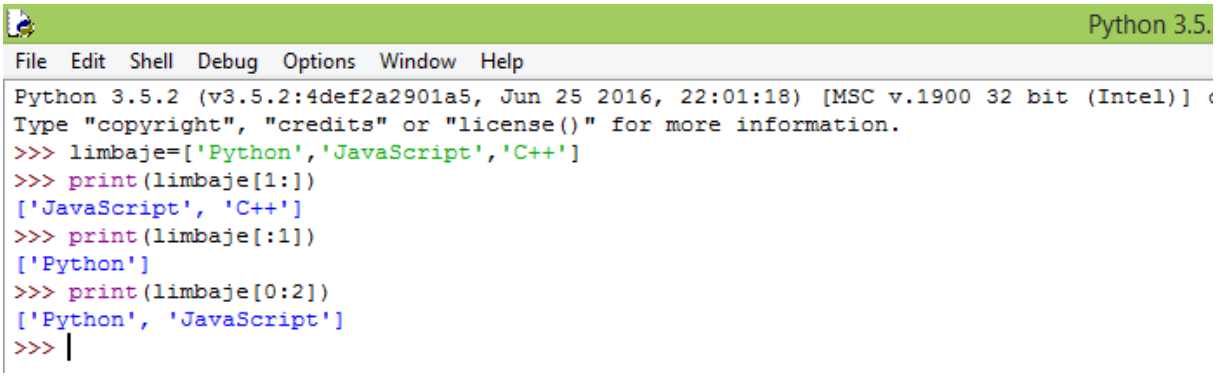
```
>>> print(limbaje[0])
Python
>>> print(limbaje[1])
JavaScript
>>> print(limbaje[2])
C++
>>>
```

Python permite utilizarea de indici negativi pentru accesarea elementelor unei liste. Iată pentru exemplul nostru cum se atașează indicii:

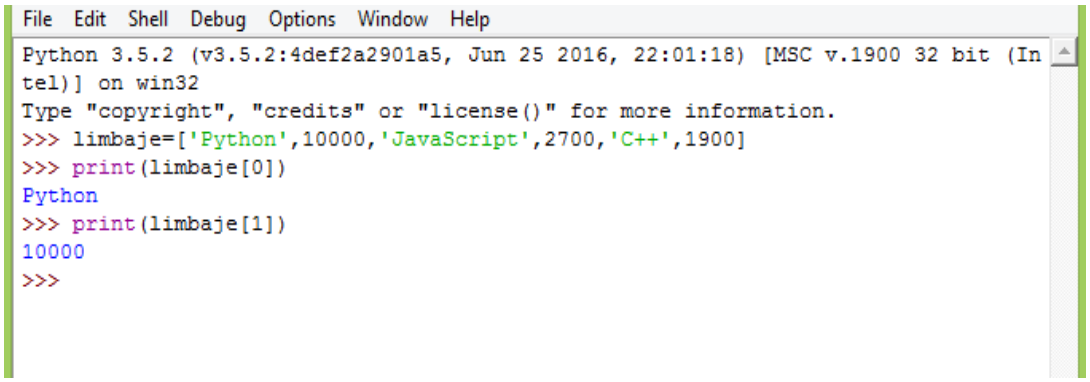
Python	JavaScript	C++
0	1	2
-3	-2	-1



La fel ca în cazul șirurilor de caractere se poate afișa o parte a listei folosind expresii de forma inceput:sfarsit. Analizând exemplul de mai jos, constatăm că se afișează elementele din pozițiile inceput, inceput+1, inceput+2, …, sfarsit-1.

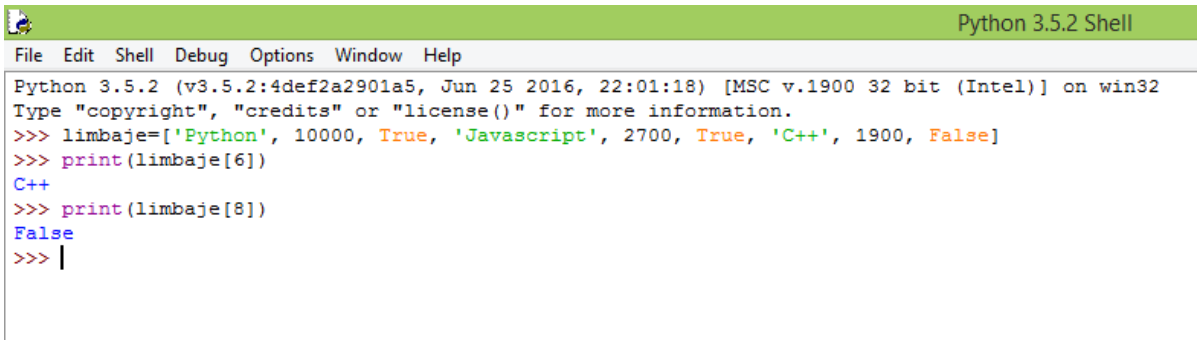


2. Dacă dorim să construim o listă cu cele mai populare limbaje de programare în care numele unui limbaj este urmat de un număr reprezentând numărul de voturi primite de la fanii limbajului respectiv vom folosi o lista neomogenă.



Așa cum era de așteptat de la o carte în care învățăm să programăm în limbajul Python, acesta este cel mai popular limbaj și a reușit să strângă 10000 de voturi.

3. În listă putem adăuga și elemente de tip logic. De exemplu, la lista noastră de limbaje de programare, pentru fiecare limbaj putem adăuga o valoare True dacă acel limbaj este studiat în cadrul unui curs opțional și o valoare False în caz contrar.



Observăm că C++ nu este studiat în cadrul cursurilor opționale.

Verificarea apartenenței unui element la o lista

Verificarea apartenenței unui element la o lista se realizează folosind operatorul in.

Exemple:

1)

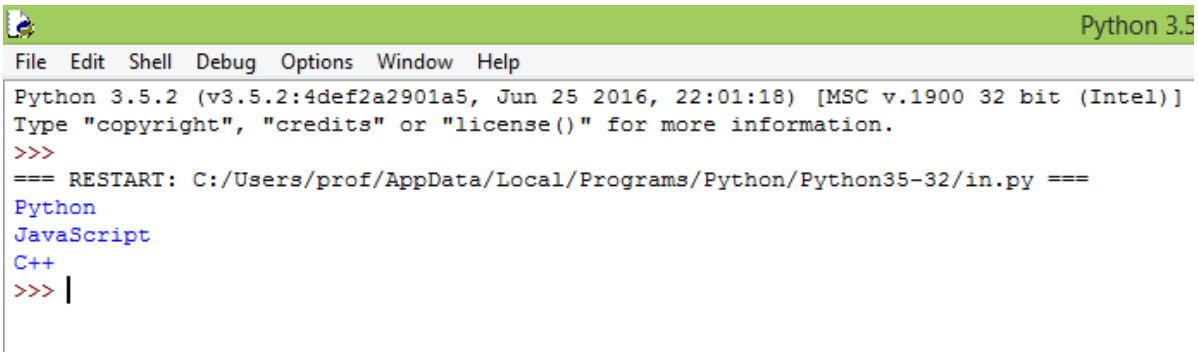
```
limbaje=['Python','JavaScript','C++']
if 'Java' in limbaje:
   print("Limbajul a fost gasit")
else:
   print("Limbajul nu a fost gasit")
```

Evident, în urma rulării programului, pe ecran ne apare mesajul *Limbajul nu a fost gasit*.

2)

```
limbaje=['Python','JavaScript','C++']
for l in limbaje:
   print(l)
```

În urma execuției programului, pe ecran ne apare rezultatul următor:



1.2 Manipularea elementelor listelor

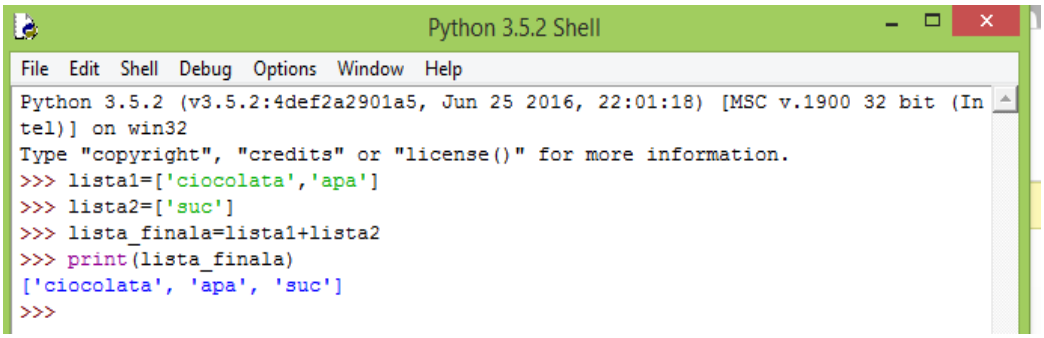
1.2.1 Concatenarea a două liste

Ce facem dacă suntem într-o excursie cu clasa, mergem într-un magazin iar un coleg ne roagă să îi cumpărăm o ciocolată și o sticlă de apă iar un alt coleg ne roagă sa îi cumpărăm un suc? Vom avea doua liste de cumpărături.

lista1 = ['ciocolata', 'apa']

lista2 = ['suc']

În magazin vom dori să avem o singură listă așa că le vom lipi (termenul folosit în programare este cel de concatenare) și vom folosi o singură listă cu ajutorul operatorului +.



Putem concatena și liste de numere

```
>>> lista1=[1,2,3]
>>> lista2=[4,5,6,7]
>>> lista3=lista1+lista2
>>> print(lista3)
[1, 2, 3, 4, 5, 6, 7]
>>>
```

Ceea ce trebuie să reținem este că operația de concatenare este o operație care implică două liste și are ca rezultat construirea unei noi liste care conține elementele din prima listă urmate de elementele din a doua listă.

1.2.2 Adăugarea unui element la o lista

1.2.2.1 Adăugarea la sfârșit

Adăugarea la sfârșitul listei este realizată de funcția **append.**

Avem o lista de limbaje de programare pe care ne dorim să le învățăm și tocmai am aflat că Java este un limbaj de programare foarte popular și dorim sa îl adăugăm la lista noastră.

```
>>> limbaje=['Python','JavaScript','C++']
>>> limbaje.append('Java')
>>> print(limbaje)
['Python', 'JavaScript', 'C++', 'Java']
>>>
```

1.2.2.2 Adăugarea în interiorul listei

Adăugarea în interiorul listei este realizată de funcția **insert.**

Ceea ce am aflat despre limbajul Java ne-a impresionat atât de mult încât am decis să îl învățăm înaintea limbajului C++. Poate și pentru faptul ca este considerat mai ușor de învățat.

```
>>> limbaje=['Python','JavaScript','C++']
>>> limbaje.insert(2,'Java')
>>> print(limbaje)
['Python', 'JavaScript', 'Java', 'C++']
>>>
```

Se realizează inserarea pe poziția 2.

1.2.3 Ștergerea unui element dintr-o listă

1.2.3.1 Ștergerea unui element dintr-o anumită poziție

În cazul în care dorim totuși să fim mai realiști și să reducem lista limbajelor pe care le învățăm renunțând la C++ putem face asta cu ușurință folosind funcția **pop**.

```
>>> limbaje=['Python','JavaScript','C++']
>>> limbaje.pop(2)
'C++'
>>> print(limbaje)
['Python', 'JavaScript']
>>>
```

Se elimina elementul din pozitia 2.

1.2.3.2 Ștergerea unui element cu o anumită valoare

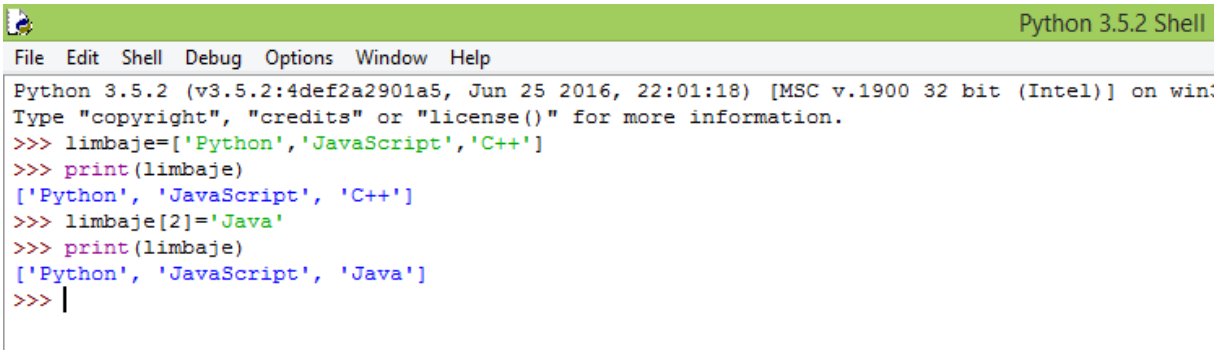
În cazul în care dorim ștergerea limbajului JavaScript vom proceda astfel:

```
>>> limbaje=['Python','JavaScript','C++']
>>> limbaje.remove('JavaScript')
>>> print(limbaje)
['Python', 'C++']
>>>
```

Se elimina elementul JavaScript

1.2.4 Modificarea elementelor unei liste

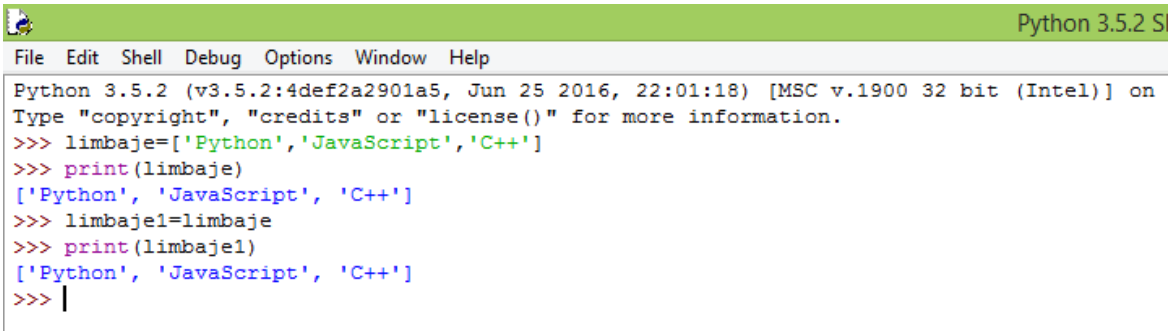
Dacă dorim în lista noastră de limbaje de programare pe ultima poziție să apară limbajul Java, facem modificarea prin următoarea atribuire: limbaje[2]='Java'.



1.2.5 Copierea elementelor unei liste

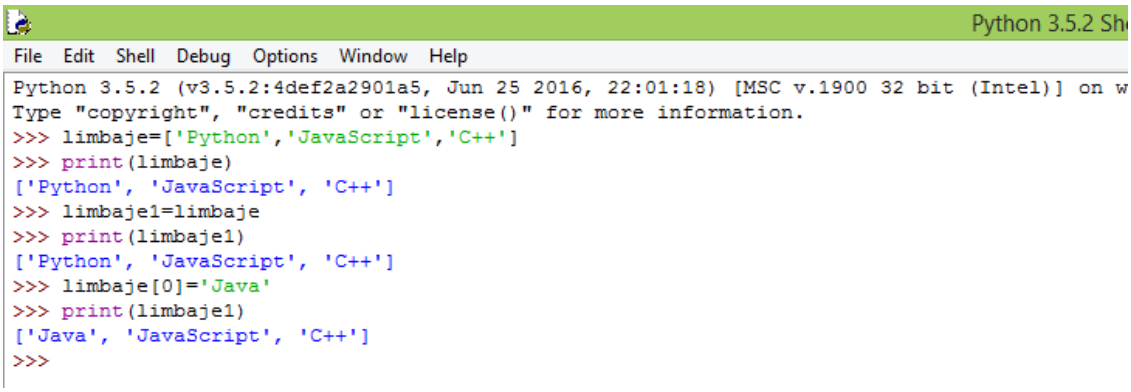
Copierea elementelor unei liste într-o altă listă se poate realiza prin o atribuire de forma:

lista_noua=lista_veche

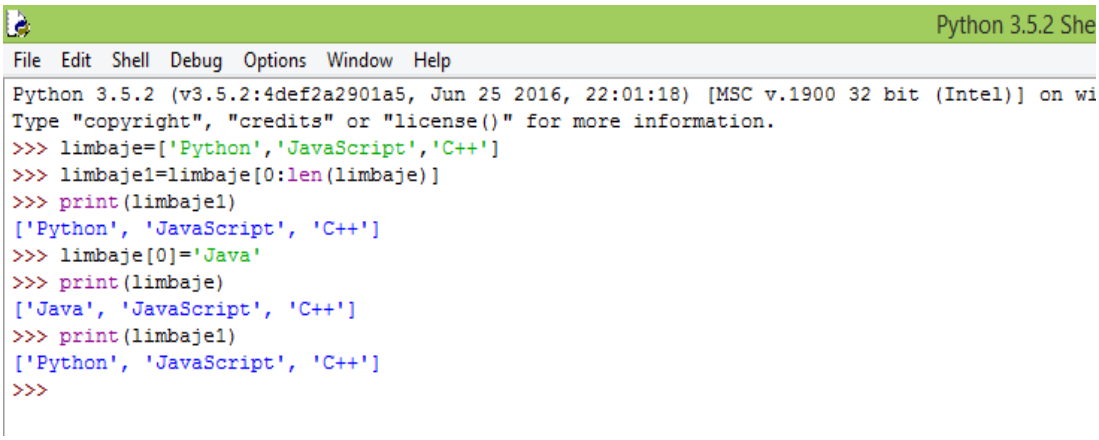


Observație importantă: Atunci când modificăm un element din lista inițială, modificarea are loc automat și în lista obținută prin copiere.

Urmăriți exemplul de mai jos. Modificarea realizată pentru elementul limbaje[0] s-a realizat automat și pentru elementul limbaje1[0]. Ce s-a întâmplat de fapt? În urma atribuirii limbaje1=limbaje, cele două liste vor pointa către aceeași zonă de memorie.



O modalitate de a copia o lista într-o lista care sa ocupe altă zonă de memorie este următoarea: folosim expresia inceput:sfarsit pentru precizarea indicilor elementelor care se vor copia.



Observăm că pentru marginea din dreapta a intervalului s-a folosit funcția len care returnează numărul de elemente din listă.

1.2.6 Sortarea elementelor unei liste

Uneori este bine sa fim ordonați. Putem sorta elementele listei in ordine alfabetică (lexicografică) dacă lista conține șiruri de caractere sau putem sorta o lista de numere. Sortarea este realizată de funcția **sort**.

```
>>> limbaje=['Python','JavaScript','C++']
>>> limbaje.sort()
>>> print(limbaje)
['C++', 'JavaScript', 'Python']
>>> numere=[7,3,9,-5]
>>> numere.sort()
>>> print(numere)
[-5, 3, 7, 9]
>>>
```

În cazul listei limbaje s-a făcut ordonare lexicografică.

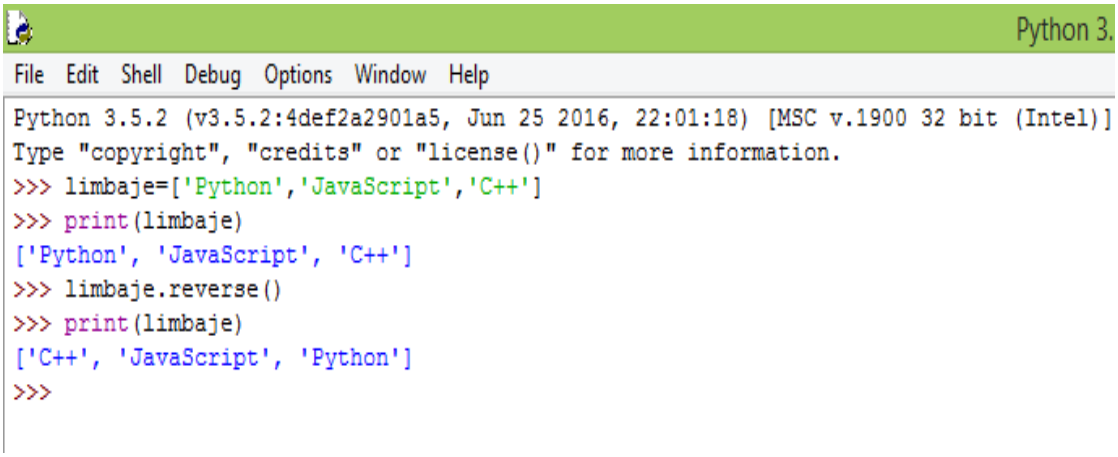
1.2.7 Calculul minimului și maximului elementelor unei liste

Calculul minimului și maximului elementelor unei liste este realizat de funcțiile min și max.

```
>>> limbaje=['Python','JavaScript','C++']
>>> print(min(limbaje))
C++
>>> print(max(limbaje))
Python
>>> numere=[7,3,9,-5]
>>> print(min(numere))
-5
>>> print(max(numere))
9
>>>
```

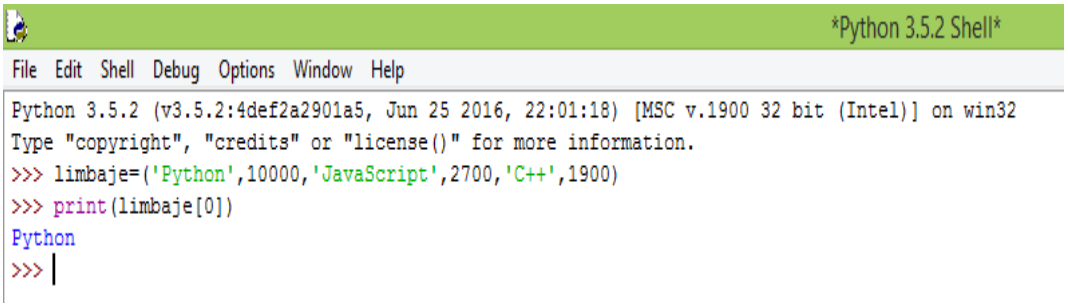
1.2.8 Inversarea elementelor unei liste

Inversarea elementelor unei liste se realizează folosind funcția reverse.



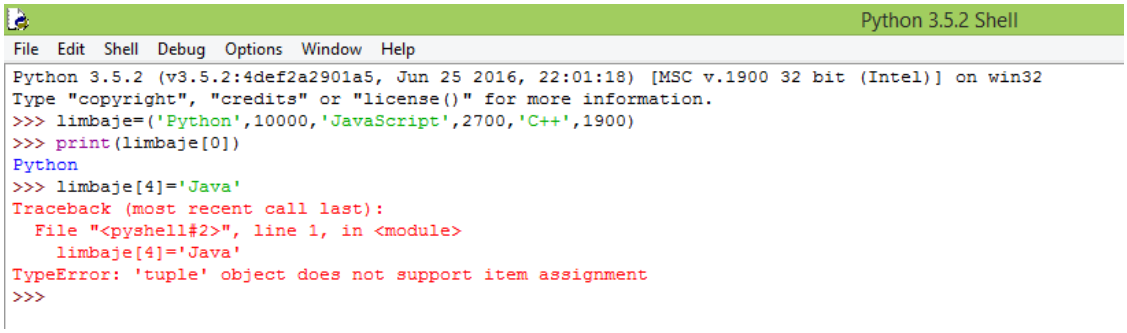
1.3 Tupluri

În Python, un tuplu sau n-uplet este asemănător unei liste, singura diferență fiind dată de faptul că elementele lui nu pot fi modificate. Elementele unui tuplu se introduc între paranteze rotunde. Ca și la liste, elementele unui tuplu pot fi accesate prin numele tuplului urmat de un indice. Indicii sunt numerotați de la 0.



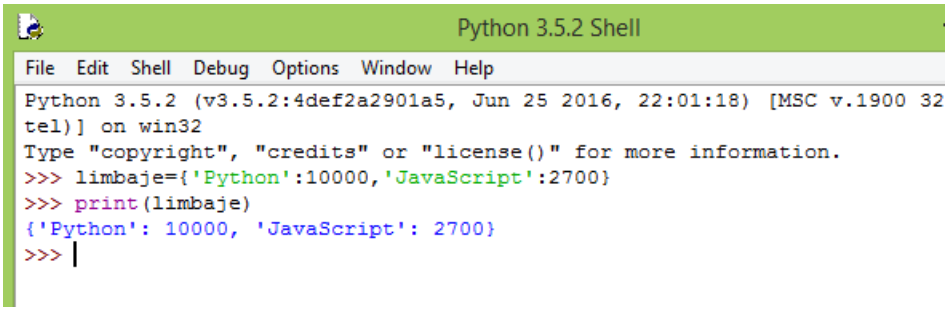
În exemplul de mai jos, încercarea de modificare a tuplului a eșuat.

1.4 Dicționare



Dicționarele sunt asemănătoare listelor numai că ele sunt formate din perechi de forma cheie:valoare. În locul indicelui, pentru accesul la un element se folosește cheia.

Revenim la exemplul nostru cu limbajele de programare.



Adăugarea unui element la un dicționar se realizează astfel:

```
>>> limbaje['C++']=1900
>>> print(limbaje)
{'C++': 1900, 'Python': 10000, 'JavaScript': 2700}
>>>
```

Spre deosebire de tupluri, elementele unui dicționar pot fi modificate. Iată un exemplu:

```
>>> limbaje['JavaScript']=2900
>>> print(limbaje)
{'C++': 1900, 'Python': 10000, 'JavaScript': 2900}
>>>
```

Ștergerea unui obiect dintr-un dicționar se realizează astfel:

```
>>> del limbaje['C++']
>>> print(limbaje)
{'Python': 10000, 'JavaScript': 2900}
>>>
```

Remarcăm faptul că într-un dicționar elementele sunt identificate prin cheia lor nu prin indice.