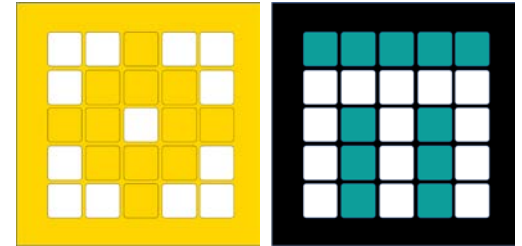


# PRIME LESSONS

By the Makers of EV3Lessons



## LISTE ȘI TUPLURI

DE SANJAY ȘI ARVIND SESHAN

# OBIECTIVELE LECȚIEI

- Învăță cum să creezi și să utilizezi listele unidimensionale.
- Învăță cum să creezi și să utilizezi tuplurile.
- Învăță cum să creezi și să utilizezi listele bidimensionale.

# INFORMAȚII DE BAZĂ

- Listele și tuplurile rețin seturi de date.
- Liste separate de virgulă
  - Liste între acolade
  - Tupluri între paranteze
- Fiecare element din listă sau tuplu este atribuit unui index, începând de la 0.
  - L=[index 0, index 1, index 2.....]
- Poți citi datele de la un index (pentru liste, tupluri și string-uri) prin apel.
  - L[index]

```
# List:
L = [1, 2, 3]
M = ["Hello", "bye"]
N = [1, True, "Hello"]
L[0] == 1 # True

# Tuple:
a = (1, 2, 3)
b = ("Hello", "bye")
c = (1, True, "Hello")
```

# METODELE LISTEI

- Toate metodele listelor editează lista originală și nu returnează niciun rezultat (cu excepția `pop()` care returnează elementul eliminat).

Metodă	Descrierea
<code>append(data)</code>	Adaugă un element la finalul listei
<code>count(data)</code>	Returnează numărul de elemente cu o valoare specifică.
<code>extend(L)</code>	Adaugă elementele unei liste la finalul listei curente.
<code>index(data)</code>	Returnează indexul primului element cu valoarea specificată.
<code>insert(i, data)</code>	Adaugă un element într-o poziție specifică.
<code>pop(i)</code>	Șterge elementul de la poziția specifică.
<code>remove(data)</code>	Șterge primul element de o valoare specifică.
<code>reverse()</code>	Înversează ordinea din listă.
<code>sort()</code>	Sortează lista.

# MUTABILITATEA

- Listele sunt tipuri de date mutabile.
  - Tuplurile, string-urile etc. nu sunt.
- Acestu lucru înseamnă că odată ce ai editat lista, se editează același obiect din memoria RAM și nu creează unul nou.
- Poți edita o listă prin atribuirea la un index a unui nou element. (vezi galben).
  - Acest lucru nu este adevărat pentru string-uri și tuple.

```
>>> s = "abc"
>>> s[0] = "b"
TypeError: 'str' object does not support item assignment
>>> t = (1,2,3)
>>> t[1] = 0
TypeError: 'tuple' object does not support item assignment
>>> L = [1,2,3]
>>> L[0] = 4
>>> L
[4, 2, 3]
>>>
```

# COPIEREA UNEI LISTE

- Este necesar să folosești o funcție de copiere dintr-un modul de copiere.
- Spre deosebire de string-uri și tuple-uri, obiectul din memorie este copiat, alte tipuri vor fi „copiate” doar prin „schimbarea” valorii.
  - De ex. nu poți face `a=b` pentru a copia o listă, dar poți realiza acest lucru la alte tipuri → vezi acest lucru în acțiune în porțiunea din dreapta (verde).
- Poți copia o listă (vezi galben).
  - `M = L.copy()`
  - Editările nu afectează lista originală.

```
>>> L = [1, 2, 3]
>>> M=L
>>> print(M, L)
[1, 2, 3] [1, 2, 3]
>>> L.append(5)
>>> print(M, L)
[1, 2, 3, 5] [1, 2, 3, 5]
>>> N = L.copy()
>>> N.append(4)
>>> print(M, L, N)
[1, 2, 3, 5] [1, 2, 3, 5] [1, 2, 3, 5, 4]
```

# MAI MULTE DESPRE LISTĂ

- Ai posibilitatea ...
- Să împarți în secțiuni.
- Să găsești lungimea listei.
- Să găsești suma listei.
- Să adaugi (vezi la metodele listei).
- Să sortezi o listă folosind metoda `sort()` (numeric, alfabetic etc).
- Inversează lista folosind metoda `reverse()`.

```
L = [1, 2, 3, 4, 5]
```

```
# Slices
```

```
L[1:3] == [2, 3]
```

```
L[1:5:2] == [2, 4]
```

```
# L[START:END:INTERVAL]
```

```
# Length (of list/tuple)
```

```
len(L) == 5
```

```
# Sum (of all items in the list/tuple)
```

```
sum(L) == 15
```

```
# Add to list
```

```
L.append(6)
```

```
print(L) # [1, 2, 3, 4, 5, 6]
```

# BUCLE „FOR” ÎN LISTE

- Poți itera (parcurge secvențial lista) prin lista sau tuplu folosind bucla „for”.
- Variabila buclă („item”-ul din exemplu) îi este atribuită următoarea variabilă din listă de fiecare dată când trecem prin buclă.
- Bucla se termină când nu mai sunt elemente.

```
L = [1, 2, 8, "hello"]  
for item in L:  
    print(item)
```

*Output:*

```
1  
2  
8  
hello
```



# STRING-URI ȘI LISTĂ

- Poți folosi funcția `list()` pentru a despărți string-ul în caractere.
- Poți folosi funcția `split()` pentru a converti un string într-o listă, împărțind după un criteriu stabilit.
- Poți anula conversia cu funcția `"".join(L)`.

```
>>> L = list("abcd")
>>> print(L)
['a', 'b', 'c', 'd']
>>> s = "a,b,c,de"

>>> M = s.split(",")
>>> print(M)
['a', 'b', 'c', 'de']
```

# PROVOCARE

- Fiind dată o listă de numere, adună pătratele numerelor și returnează răspunsul. Apoi printează răspunsul într-o matrice de lumini.
- Vei folosi listele unidimensionale, buclele „for” și, opțional, funcțiile.

# SOLUȚIA PROVOCĂRII

```
from spike import PrimeHub, LightMatrix
from spike.control import wait_for_seconds, wait_until, Timer
from math import *
import time

hub = PrimeHub()

def sumSquares(L):
    sum = 0
    for num in L:
        sum += num**2
    return sum

hub.light_matrix.write(sumSquares([1, 3, 9]))
```

# LISTELE BIDIMENSIONALE: LISTE ÎN LISTE

- În Python, o listă bidimensională este doar o listă de liste (fiecare element din listă este o altă listă).
- Poți avea 3 sau 4 dimensiuni.
- Lista bidimensională este o matrice.

```
L = [[ 2, 3, 5 ],  
      [ 1, 4, 7 ]]
```

# RETURNAREA UNUI ELEMENT

- Similar cu listele unidimensionale.
- Poți găsi un element din listă într-un element părinte.
- Adresarea unui element prin apel
  - `L[linie][coloană]`

```
L = [[ 2, 3, 5 ],  
      [ 1, 4, 7 ]]
```

```
L[0][1] == 3
```

```
L[1][2] == 7
```

# CICLAREA ÎNTR-O LISTĂ BIDIMENSIONALĂ

- Folosește buclă în buclă.
- Iterarea prin lista părinte și apoi prin lista copil.
- Ciclarea printre rânduri sau coloane.

```
L = [[ 2, 3, 5 ],  
      [ 1, 4, 7 ]]  
  
for row in L:  
    for col in row:  
        print(col)
```

Output:

```
2  
3  
5  
1  
4  
7
```

# COPIEREA UNEI LISTE BIDIMENSIONALE

- Similar cu problemele de mutabilitate de la listele unidimensionale dar mai mult
- Fiecare listă copilare propria referință de memorie.
- Ne trebuie un “deepcopy”.
- Din păcate, micropython nu implementează nativ biblioteca de copiere deci va trebui să creem noi propriul deepcopy.
- Funcția de mai jos folosește recursia (care va fi învățat într-o lecție de mai jos) pentru a crea o simplă copie a listei de elemente fără a utiliza lista originală.
- Folosește funcția de mai jos oricărei liste - `M=deepCopy(L)`

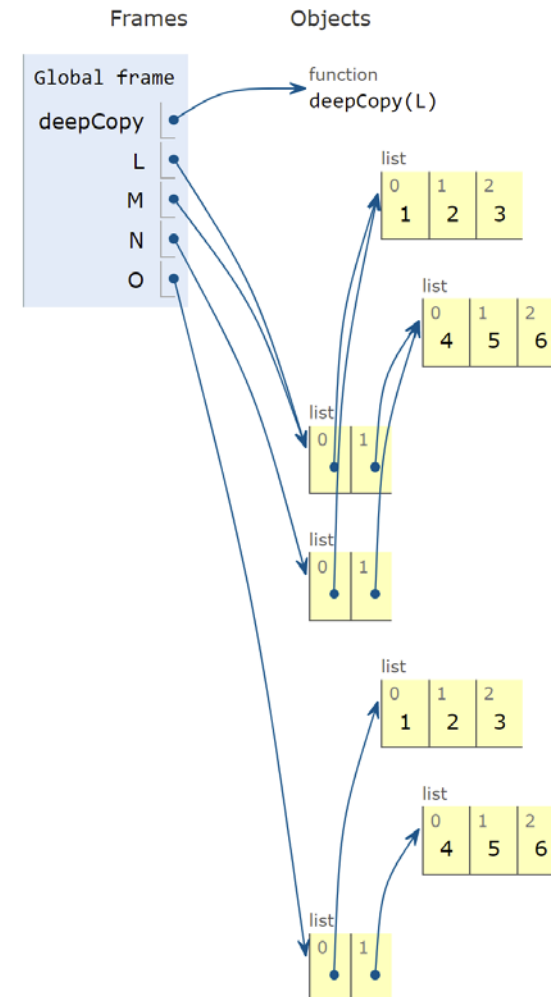
```
def deepCopy(L):  
    if (type(L)==list):  
        return [deepCopy(e) for e in L]  
    else: return L
```

# ANALIZA COPIERII LISTELOR BIDIMENSIONALE

- Să ne uităm la structura memoriei din următorul cod:

```
def deepCopy(L):  
    if (type(L)==list):  
        return [deepCopy(e) for e in L]  
    else: return L  
L = [ [ 1, 2, 3 ] , [ 4, 5, 6 ] ]  
M = L  
N = L.copy()  
O = deepCopy(L)
```

- Observă în diagrama obiectului (dreapta), punctele M și L din aceeași listă, realizând că este același obiect.
- Când N are lista proprie, elementele lui arată către aceleași liste ca L, arătând că nu sunt copiate ca o metodă normală de copiere.
- Elementul O, totuși, are toți copiii independenți la L, arătând că este copiat corect folosind deepcopy.
- Practic, dacă folosiți listele bidimensionale, folosește deepcopy.





# CONTROLUL PIXELILOR LA MATRICEA DE LUMINI

- Fiecare pixel al matricei de lumină este reprezentat de valorile x,y și o valoare de luminozitate.
- Metoda de controlare a pixelilor este `set_pixel(x, y, brightness)`.
  - Valoarea x este poziția pixelului numărând de la stânga (intervalul 1-5).
  - Valoarea y este poziția pixelului numărând de sus (intervalul 1-5).
- Valorile luminozității variază între 0-100

De exemplu:

```
hub.light_matrix.set_pixel(1, 4, brightness=100)
```

# PROVOCARE

- Fiind dată o listă bidimensională de coordonate, într-o buclă, pornește, așteaptă o secundă, oprește fiecare pixel secvențial.

- Lista arată așa:

```
L=[ [1, 1],  
    [2, 4],  
    [3, 5]]
```

- Fiecare listă copil este o pereche de coordonată  $[x,y]$ .

# SOLUȚIA PROVOCĂRII

```
from spike import PrimeHub, LightMatrix
from spike.control import wait_for_seconds, wait_until, Timer
from math import *
import time

hub = PrimeHub()

L=[[1, 1],
   [2, 4],
   [3, 5]]

for coord in L:
    x = coord[0]
    y = coord[1]
    # Note that the previous three lines can be replaced with a single 'for (x,y) in L:' instead
    hub.light_matrix.set_pixel(x, y, brightness=100)
    time.sleep(1)
    hub.light_matrix.set_pixel(x, y, brightness=0)
```

# CREDITE

- Această lecție de SPIKE Prime a fost realizată de Sanjay Seshan și Arvind Seshan.
- Mai multe lecții sunt disponibile pe [www.primelessons.org](http://www.primelessons.org)
- Această lecție a fost tradusă în limba română de echipa de robotică FTC – ROSOPHIA #21455 RO20



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).