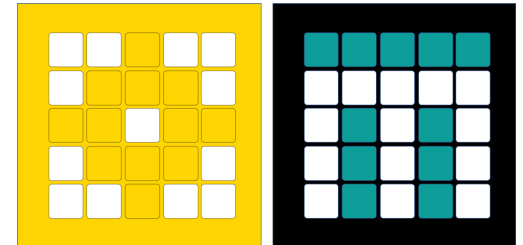


PRIME LESSONS

By the Makers of EV3Lessons



LISTS AND TUPLES

BY SANJAY AND ARVIND SESHAN

This lesson uses SPIKE 3 software

LESSON OBJECTIVES

- Learn to create and use 1D lists
- Learn to create and use tuples
- Learn to create and use 2D lists

BASICS

- Lists and Tuples store a set of data
- Comma separated lists
 - Lists inside brackets
 - Tuples inside parentheses
- Each entry in a list or tuple is assigned an index, starting at 0
 - L=[index 0, index 1, index 2.....]
- You can read data at an index (for lists, tuples, and strings) by calling
 - 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")
```

LIST METHODS

- All list methods edit the original list and do not return anything (except pop() which returns the removed data)

Method	Description
<u>append(data)</u>	Adds an element at the end of the list
<u>count(data)</u>	Returns the number of elements with the specified value
<u>extend(L)</u>	Add the elements of a list (or any iterable), to the end of the current list
<u>index(data)</u>	Returns the index of the first element with the specified value
<u>insert(i, data)</u>	Adds an element at the specified position
<u>pop(i)</u>	Removes the element at the specified position
<u>remove(data)</u>	Removes the first item with the specified value
<u>reverse()</u>	Reverses the order of the list
<u>sort()</u>	Sorts the list

MUTABILITY

- Lists are a mutable data type
 - Tuples, strings, etc. are not
- This means that when you edit a list, it edits that same memory (RAM) object instead of creating a new one
- You can edit a List by assigning an index's data to a new piece of data (see yellow)
 - This is not true for strings or tuples

```
>>> 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]
>>>
```

COPYING A LIST

- You must use the copy function from the copy module
- Unlike strings, tuples, etc., the memory object must be copied; other types will be “copied” simply by “changing” the value
 - I.e. you cannot do `a=b` to copy a list, but you can for other types → see this in action in the right (green)
- You can copy a list (see yellow)
 - `M = L.copy()`
 - Edits do not affect the original list

```
>>> 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]
```

MORE ABOUT LISTS

- You can....
- Get slices (sections)
- Length of list
- Sum of list
- Append, etc. (see list methods)
- Sort a list using `.sort()` (numerically, alphabetically, etc.) method
- Reverse a list using `.reverse()` method

```
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]
```

FOR LOOPS WITH LISTS

- You can iterate (i.e. sequentially go through) through a list or tuple using a “for” loop
- The loop variable (“item” in the example) is assigned the value of the next item in the list each time through the loop
- The loop ends when there are no more items

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

Output:

```
1  
2  
8  
hello
```


STRINGS TO LISTS

- You can use the `list()` function to split each character into an entry
- You can also use the `split()` method to convert the string into a list, splitting at the desired item
- You can undo the conversion with `"".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']
```

CHALLENGE

- Given a list of numbers, sum the squares of the numbers and return the answer. Then print the answer to the light matrix
- You will need to use ID lists, for loops, and optionally functions

CHALLENGE SOLUTION

```
from hub import light_matrix
import runloop, math, sys

# Function to stop the program using a system exception
def stopAndExitProgram():
    sys.exit("Stopping")

def sumSquares(L):
    sum = 0
    for num in L:
        sum += math.pow(num,2)
    return sum

async def main():
    sum = sumSquares([1,3,9])
    # Print the sum after converting it to an int, and then string
    await light_matrix.write(str(int(sum)))
    # Stop and exit the program. You should see the Program number on your hub.
    stopAndExitProgram()

runloop.run(main())
```

2D LISTS: LISTS WITHIN LISTS

- In Python, a 2D list is just a list of lists (i.e. each element of the list is another list)
- You can have 3D, 4D, etc.
- 2D list sometimes called a matrix

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

GETTING AN ELEMENT

- Similar to ID lists
- You get an element of a list within an element of the “parent” list
- Address an element by calling
 - `L[row][column]`

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

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

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

LOOPING ON A 2D LIST

- Use nested loops
- Iterate on the parent list then the child list
- Loop over rows then columns

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

Output:

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

COPYING A 2D LIST

- Similar mutability issues to 1D lists but even more
- Each “child” list has its own memory reference
- We need to do a “deepcopy”
- Unfortunately, micropython does not natively implement the copy library so we need to create our own deepcopy
- The function below uses recursion (which will be taught in a later lesson) to create a simple copy of list elements without using the original list
- Use this function on any list - i.e., `M=deepCopy(L)`

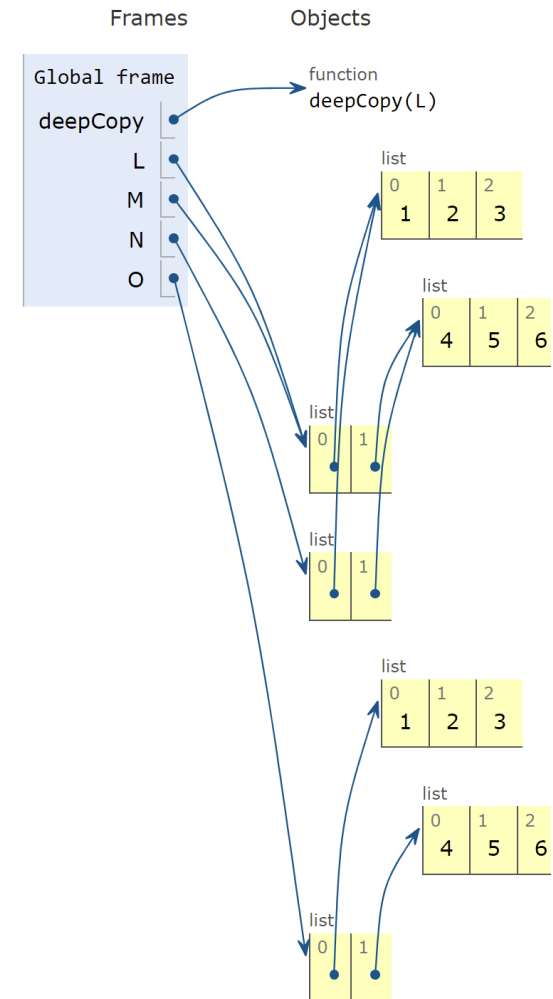
```
def deepCopy(L):  
    if (type(L)==list):  
        return [deepCopy(e) for e in L]  
    else: return L
```

2D LIST COPYING ANALYSIS

- Let's take a look at the memory structure of the following code:

```
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)
```

- Notice in the object diagram (right), M and L point to the same list, showing that it is really the same object
- While N has its own list, its elements point to the same lists as L, showing that they were not copied when using the normal copy method
- O, however, has all of its children independent of L, showing that it is copied correctly using deepcopy
- Basically, if you are working with 2D lists, use deepcopy.



LIGHT MATRIX PIXEL CONTROL

- Each pixel on the light matrix is represented by a `x,y` value and a brightness value
- The method to control the matrix pixel is `set_pixel(x, y, brightness)`.
 - The `x` value is the pixel position counting from the left (range 1-5)
 - The `y` value is the pixel position counting from the top (range 1-5)
 - The brightness value ranges from 0-100

For example:

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

CHALLENGE

- Given a 2D list of coordinates, in a loop, turn on, wait one second, and turn off each pixel sequentially
- The list will look like:

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

- Each child list is an $[x, y]$ coordinate. Note the ranges for the hub light matrix x and y values are 0-4. If you use numbers outside this range, it will be ignored.

CHALLENGE SOLUTION

```
from hub import light_matrix
import runloop, sys

# Function to stop the program using a system exception
def stopAndExitProgram():
    sys.exit("Stopping")

async def main():
    L = [[1,1],
         [2,3],
         [3,4]]
    for (x, y) in L:
        light_matrix.set_pixel(x, y, 100)
        await runloop.sleep_ms(1000)
        light_matrix.set_pixel (x, y, 0)
    stopAndExitProgram()

runloop.run(main())
```

CREDITS

- This lesson was created by Sanjay and Arvind Seshan for Prime Lessons
- Additional contributions by FLL Share & Learn community members.
- More lessons are available at www.primelessons.org



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