



Lists in Python

Estimated time needed: **15** minutes

Objectives

After completing this lab you will be able to:

- Perform list operations in Python, including indexing, list manipulation and copy/clone list.

Table of Contents

- [About the Dataset](#)
- [Lists](#)
 - [Indexing_\(index\)](#)
 - [List Content \(content\)](#)
 - [List Operations \(op\)](#)
 - [Copy and Clone List \(co\)](#)
- [Quiz on Lists](#)

About the Dataset

Index	
0	Element 1
1	Element 2
2	Element 3
3	Element 4
4	Element 5
Element	

[Element 1 , Element 2 , Element 3 , Element 4, Element 5]

Index 0 1 2 3 4

To create a list, type the list within square brackets [], with your content inside the parenthesis and separated by commas. Let's try it!

In [1]:

```
# Create a List
```

```
L = ["Michael Jackson", 10.1, 1982]
L
```

Out[1]:

```
['Michael Jackson', 10.1, 1982]
```

We can use negative and regular indexing with a list :

L = ["Michael Jackson", 10.1, 1982]

-3	0	"Michael Jackson"	L[-3]: "Michael Jackson"
-2	1	10.1	L[-2]: 10.1
-1	2	1982	L[-1]: 1982

In [2]:

Print the elements on each index

```
print('the same element using negative and positive indexing:\n Postive:',L[0],
'\n Negative:', L[-3] )
print('the same element using negative and positive indexing:\n Postive:',L[1],
'\n Negative:', L[-2] )
print('the same element using negative and positive indexing:\n Postive:',L[2],
'\n Negative:', L[-1] )
```

```
the same element using negative and positive indexing:
Postive: Michael Jackson
Negative: Michael Jackson
the same element using negative and positive indexing:
Postive: 10.1
Negative: 10.1
the same element using negative and positive indexing:
Postive: 1982
Negative: 1982
```

List Content

Lists can contain strings, floats, and integers. We can nest other lists, and we can also nest tuples and other data structures. The same indexing conventions apply for nesting:

In [3]:

Sample List

```
["Michael Jackson", 10.1, 1982, [1, 2], ("A", 1)]
```

Out[3]:

```
['Michael Jackson', 10.1, 1982, [1, 2], ('A', 1)]
```

List Operations

We can also perform slicing in lists. For example, if we want the last two elements, we use the following command:

In [4]:

Sample List

```
L = ["Michael Jackson", 10.1,1982,"MJ",1]
L
```

Out[4]:

```
['Michael Jackson', 10.1, 1982, 'MJ', 1]
```

L = ["Michael Jackson", 10.1, 1982, "MJ", 1]

	0	1	2	3	4
--	----------	----------	----------	----------	----------

In [5]:

List slicing

```
L[3:5]
```

Out[5]:

```
['MJ', 1]
```

We can use the method `extend` to add new elements to the list:

In [6]:

```
# Use extend to add elements to list

L = [ "Michael Jackson", 10.2]
L.extend(['pop', 10])
L
```

Out[6]:

```
['Michael Jackson', 10.2, 'pop', 10]
```

Another similar method is `append`. If we apply `append` instead of `extend`, we add one element to the list:

In [7]:

```
# Use append to add elements to list

L = [ "Michael Jackson", 10.2]
L.append(['pop', 10])
L
```

Out[7]:

```
['Michael Jackson', 10.2, ['pop', 10]]
```

Each time we apply a method, the list changes. If we apply `extend` we add two new elements to the list. The list `L` is then modified by adding two new elements:

In [8]:

```
# Use extend to add elements to list

L = [ "Michael Jackson", 10.2]
L.extend(['pop', 10])
L
```

Out[8]:

```
['Michael Jackson', 10.2, 'pop', 10]
```

If we append the list `['a', 'b']` we have one new element consisting of a nested list:

In [9]:

```
# Use append to add elements to list

L.append(['a', 'b'])
L
```

Out[9]:

```
['Michael Jackson', 10.2, 'pop', 10, ['a', 'b']]
```

As lists are mutable, we can change them. For example, we can change the first element as follows:

In [10]:

```
# Change the element based on the index

A = ["disco", 10, 1.2]
print('Before change:', A)
A[0] = 'hard rock'
print('After change:', A)
```

```
Before change: ['disco', 10, 1.2]
After change: ['hard rock', 10, 1.2]
```

We can also delete an element of a list using the `del` command:

In [11]:

```
# Delete the element based on the index

print('Before change:', A)
del(A[0])
print('After change:', A)
```

```
Before change: ['hard rock', 10, 1.2]
After change: [10, 1.2]
```

We can convert a string to a list using `split`. For example, the method `split` translates every group of characters separated by a space into an element in a list:

In [12]:

```
# Split the string, default is by space

'hard rock'.split()
```

Out[12]:

```
['hard', 'rock']
```

We can use the `split` function to separate strings on a specific character. We pass the character we would like to split on into the argument, which in this case is a comma. The result is a list, and each element corresponds to a set of characters that have been separated by a comma:

In [13]:

```
# Split the string by comma

'A,B,C,D'.split(',')
```

Out[13]:

```
['A', 'B', 'C', 'D']
```

Copy and Clone List

When we set one variable **B** equal to **A**; both **A** and **B** are referencing the same list in memory:

In [14]:

```
# Copy (copy by reference) the List A

A = ["hard rock", 10, 1.2]
B = A
print('A:', A)
print('B:', B)
```

```
A: ['hard rock', 10, 1.2]
B: ['hard rock', 10, 1.2]
```



Initially, the value of the first element in **B** is set as hard rock. If we change the first element in **A** to **banana**, we get an unexpected side effect. As **A** and **B** are referencing the same list, if we change list **A**, then list **B** also changes. If we check the first element of **B** we get banana instead of hard rock:

In [15]:

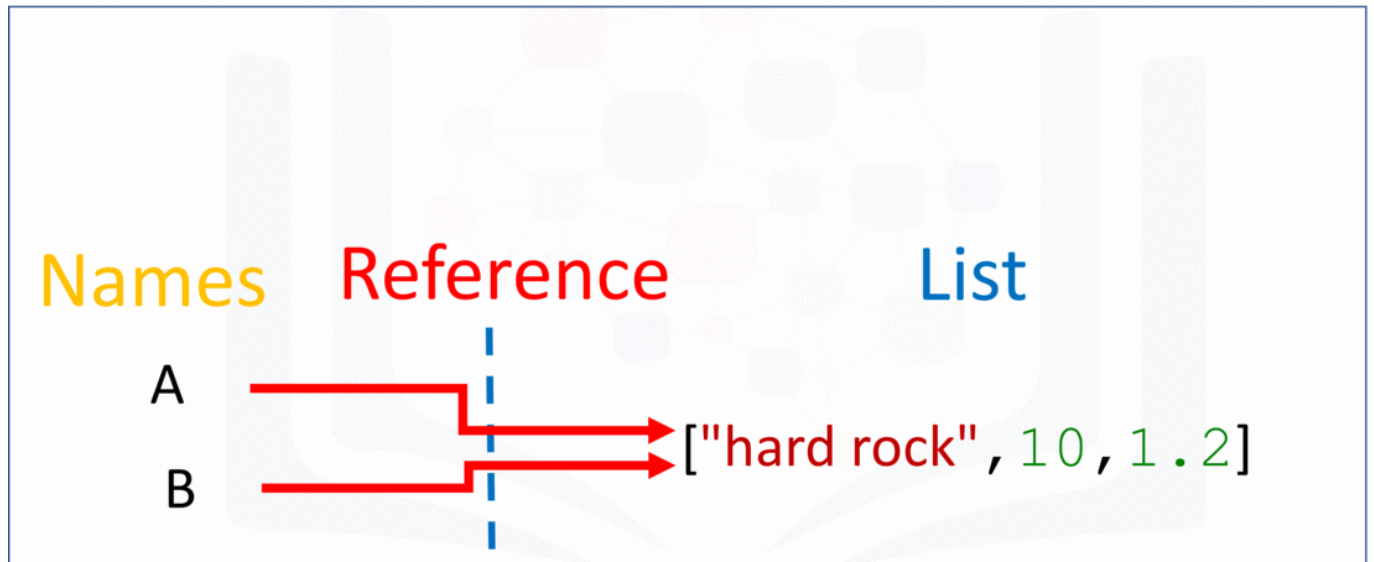
```
# Examine the copy by reference
```

```
print('B[0]:', B[0])  
A[0] = "banana"  
print('B[0]:', B[0])
```

B[0]: hard rock

B[0]: banana

This is demonstrated in the following figure:



You can clone list **A** by using the following syntax:

In [16]:

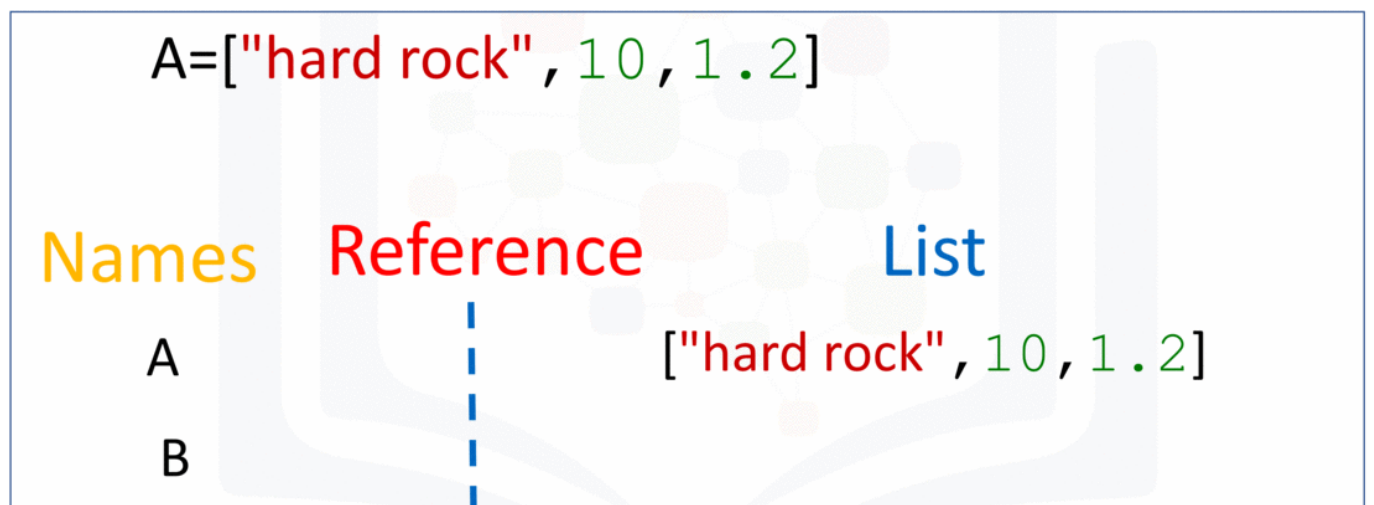
```
# Clone (clone by value) the list A
```

```
B = A[:]  
B
```

Out[16]:

```
['banana', 10, 1.2]
```

Variable **B** references a new copy or clone of the original list; this is demonstrated in the following figure:



Now if you change **A**, **B** will not change:

In [17]:

```
print('B[0]:', B[0])
A[0] = "hard rock"
print('B[0]:', B[0])
```

B[0]: banana
B[0]: banana

Quiz on List

Create a list `a_list` , with the following elements `1` , `hello` , `[1,2,3]` and `True` .

In [20]:

```
# Write your code below and press Shift+Enter to execute
a_list=[1,"hello",[1,2,3],True]
a_list
```

Out[20]:

[1, 'hello', [1, 2, 3], True]

► Click here for the solution

Find the value stored at index 1 of `a_list` .

In [21]:

```
# Write your code below and press Shift+Enter to execute
a_list[1]
```

Out[21]:

'hello'

► Click here for the solution

Retrieve the elements stored at index 1, 2 and 3 of `a_list` .

In [23]:

```
# Write your code below and press Shift+Enter to execute
a_list[1:4]
```

Out[23]:

['hello', [1, 2, 3], True]

► Click here for the solution

Concatenate the following lists `A = [1, 'a']` and `B = [2, 1, 'd']` :

In [34]:

```
# Write your code below and press Shift+Enter to execute
A=[1,'a']
B=[2,1,'d']
A+B
#A.extend([2,1,'d'])
A
```

Out[34]:

[1, 'a', 2, 1, 'd']

► Click here for the solution

The last exercise!

Congratulations, you have completed your first lesson and hands-on lab in Python. However, there is one more thing you need to do. The Data Science community encourages sharing work. The best way to share and showcase your work is to share it on GitHub. By sharing your notebook on GitHub you are not only building your reputation with fellow data scientists, but you can also show it off when applying for a job. Even though this was your first piece of work, it is never too early to start building good habits. So, please read and follow [this article \(https://cognitiveclass.ai/blog/data-scientists-stand-out-by-sharing-your-notebooks/\)](https://cognitiveclass.ai/blog/data-scientists-stand-out-by-sharing-your-notebooks/) to learn how to share your work.

Author

[Joseph Santarcangelo \(https://www.linkedin.com/in/joseph-s-50398b136/\)](https://www.linkedin.com/in/joseph-s-50398b136/)

Other contributors

[Mavis Zhou \(www.linkedin.com/in/jiahui-mavis-zhou-a4537814a\)](https://www.linkedin.com/in/jiahui-mavis-zhou-a4537814a)

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2020-08-26	2.0	Lavanya	Moved lab to course repo in GitLab

© IBM Corporation 2020. All rights reserved.