



SNS COLLEGE OF ENGINEERING
Kurumbapalayam (Po), Coimbatore – 641 107



An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF CSE (IoT & CYBER SECURITY INCLUDING BLOCKCHAIN TECHNOLOGY)



19IT103 – COMPUTATIONAL THINKING AND PYTHON PROGRAMMING

- ❖ A readable, dynamic, pleasant, flexible, fast and powerful language

Session wise Agenda

- **session 1 - List (Operations, Slice, Methods)**
- **Session 2 - List (Loop, Mutability)**
- **Session 3 - List (Aliasing, Cloning, Parameters)**
- Session 4 - Tuples (Assignment, as return value)
- Session 5 - Dictionaries (operations and methods)
- Session 6 - Advance List processing, List Comprehension
- Session 7 - Simple Sort, Histogram
- Session 8 - Student Mark Statement
- Session 9 - Retail Bill preparation

Recap

- List is a sequence data type which can be traversed in 8 different ways
- List are Mutable → the values in the list can be changed. This process is called mutability

List Aliasing

- ❖ In this a single list object is created and modified using the subscript operator.
- ❖ When the first element of the list named “a” is replaced, the first element of the list named “b” is also replaced.
- ❖ This type of change is what is known as a **side effect**. This happens because after the assignment **b=a**, the variables **a** and **b** refer to the exact same list object.
- ❖ They are **aliases** for the same object. This phenomenon is known as **aliasing**.
- ❖ To prevent aliasing, a new object can be created and the contents of the original can be copied which is called **cloning**.

List Aliasing

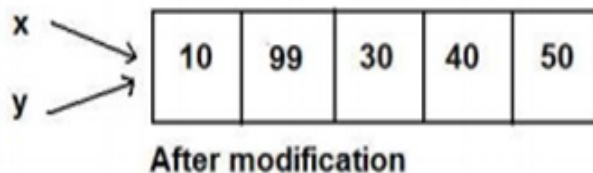
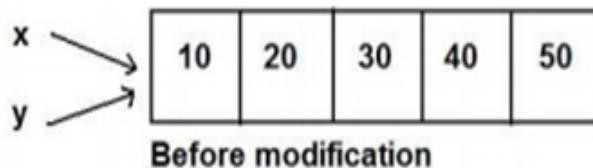
```
a=[1,2,3,4,5]
b=a
print(b)
print(b is a)
a[0]=100
print(a, "\n", b)
```

```
[1, 2, 3, 4, 5]
True
[100, 2, 3, 4, 5]
[100, 2, 3, 4, 5]
```

List Aliasing

```
x = [10,20,30,40,50]  
y = x #x is aliased as y  
print(x) will display [10,20,30,40,50]  
print(y) will display [10,20,30,40,50]
```

```
x[1] = 99 #modify 1st element in x  
print(x) will display [10,99,30,40,50]  
print(y) will display [10,99,30,40,50]
```



List Cloning

- To avoid the disadvantages of copying we are using cloning. creating a copy of a same list of elements with two different memory locations is called cloning.
- Changes in one list will not affect locations of another list.

Cloning using Slicing

```
a=[1,2,3,4,5]
b=a[:]
print(b)
print(b is a)
```

Shell

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

```
False
```


Cloning using List() method

main.py

```
1 a=[1,2,3,4,5]
2 b=list(a)
3 print(b)
4 print(a is b)
5 a[0]=100
6 print(a)
7 print(b)
```

Shell

```
[1, 2, 3, 4, 5]
False
[100, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
```

Cloning using copy() method

main.py

```
1 a=[1,2,3,4,5]
2 b=a.copy()
3 print(b)
4 print(a is b)
```

Shell

```
[1, 2, 3, 4, 5]
False
```

List Cloning

`y = x[:]` #x is cloned as y

```
x = [10,20,30,40,50]
```

```
y = x[:] #x is cloned as y
```

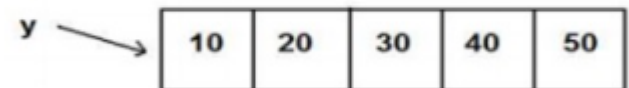
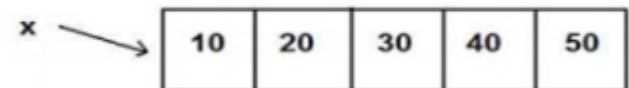
```
print(x) will display [10,20,30,40,50]
```

```
print(y) will display [10,20,30,40,50]
```

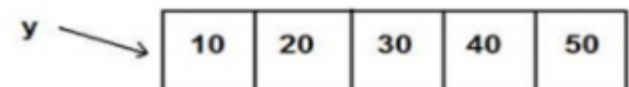
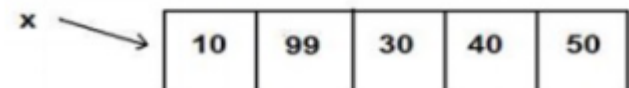
```
x[1] = 99 #modify 1st element in x
```

```
print(x) will display [10,99,30,40,50]
```

```
print(y) will display [10,20,30,40,50]
```



Before modification



After modification

List Cloning

8 Ways to Clone / Copy a List

Using Functions / Operations

1. Use slicing - `bl = al[:]` (apparently, the fastest technique)
2. Use list comprehension `bl = [elem for elem in al]`
3. Use `list()` function

Using Methods

4. Use the `.copy` method - `bl = al.copy()`
5. Using `.extend` method
6. Using `.append` method

Using Modules

7. Use `copy.copy()`
8. Use `copy.deepcopy()`

List Cloning

<https://www.geeksforgeeks.org/python-cloning-copying-list/>

Refer the above link for 8 types of cloning

List as Parameters

- In python, arguments are passed by reference.
- If any changes are done in the parameter which refers within the function, then the changes also reflects back in the calling function.
- Passing a list as an argument actually passes a reference to the list, not a copy of the list.
- Since lists are mutable, changes made to the elements referenced by the parameter change the same list that the argument is referencing.

List as Parameters

main.py

```
1 ▾ def inside(a):  
2 ▾     for i in range(0,len(a),1):  
3     |         a[i]=a[i]+10  
4     |         print("inside",a)  
5 a=[1,2,3,4,5]  
6 inside(a)  
7 print("outside",a)
```

Shell

```
inside [11, 12, 13, 14, 15]  
outside [11, 12, 13, 14, 15]
```

List as Parameters

```
def my_function(food):  
    for x in food:  
        print(x)  
  
fruits = ["apple", "banana", "cherry"]  
  
my_function(fruits)
```

```
def append_ten(a_list):  
    a_list.append(10)  
    return a_list  
  
print(append_ten([1, 2, 3]))
```

OUTPUT

```
[1, 2, 3, 10]
```


List as Parameters

```
def sum_list_elements(a_list):  
    sum = 0  
    for i in range(len(a_list)):  
        sum += a_list[i]  
    return sum  
  
print(sum_list_elements([1, 2, 3, 4, 5]))
```

OUTPUT

15

Summary

- Aliasing → copying the List i.e. the memory will be same for both the List variables. If any changes made in one list will affect other.
- Cloning → copying the List but the memory location is different. If any changes made in one list will not affect other.
- List as Parameter → List is passed as parameter to a function i.e. as Call by Reference (Address). If any changes made in the list inside function the change will occur in the calling function also



THANK YOU