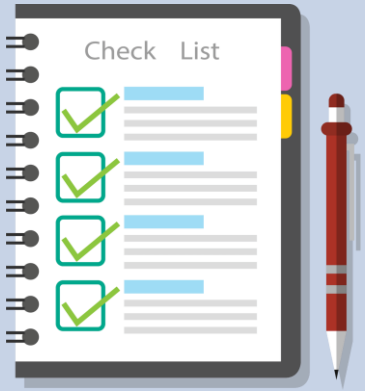
Python – Basics

Digital Lync

INNOVATION - EDUCATION - INCUBATION

Shah Ayub Quadri
aquadri@digital-lync.com

Index



Python Basics – Lists

- Introduction
- Accessing List
- Operations
- List Comprehension
- Function and Methods

Lists

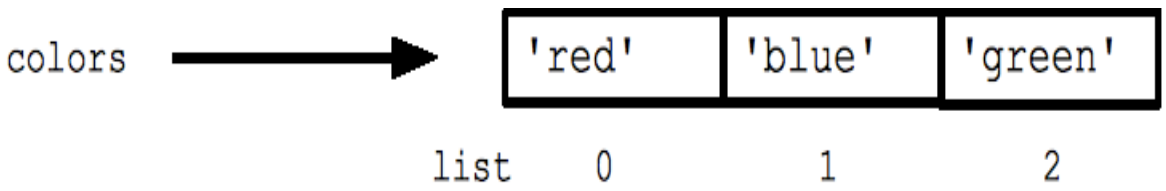
- Python has a great built-in sequential data type called "**list**".
- List literals are written within square brackets [ ].
- Lists can contain any data type.
- Lists work similarly to strings -- use the **len()** function and square brackets [ ] to access data, with the first element at index **0**.

Examples of lists :

- numbers = [10, 20, 30, 40, 50]
- names = ["Sara", "David", "Warner", "Sandy"]
- student = ["Sara", 1, "Chemistry"]

colors = ['red', 'blue', 'green']

- print colors[0] ## red
- print colors[2] ## green
- print len(colors) ## 3



Updating Lists:

- You can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator, and you can add to elements in a list with the append() method.

For example:

```
list = ['ABC', 'DEF', 19, 20];
print ("Value available at index 2 : ")
print (list[2])
list[2] = 21;
print ("New value available at index 2 : ")
print (list[2])
```

```
C:\Users\ravikiran\Desktop\python>a.py
Value available at index 2 :
19
New value available at index 2 :
21
```

Lists

Delete List Elements:

- To remove a list element, you can use either the del statement if you know exactly which element(s) you are deleting.
- Else use remove() method if you do not know.

For example :

```
list = ['ABC', 'DEF', 19, 20];
print (list)
del (list[2])
print ("After deleting value at index 2 : ")
print(list)
```

```
C:\Users\ravikiran\Desktop\python>a.py
['ABC', 'DEF', 19, 20]
After deleting value at index 2 :
['ABC', 'DEF', 20]
```

Lists

Digital Lync
INNOVATION - EDUCATION - INCUBATION

Basic List Operations:

Python Expression	Results	Description
len([1, 2, 3])	3	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership
for x in [1, 2, 3]: print x,	1 2 3	Iteration

List Methods:

Here are some other common list methods.

- **list.append(elem):** adds a single element to the end of the list.
- **list.insert(index, elem):** inserts the element at the given index, shifting elements to the right.
- **list.extend(list2):** adds the elements in list2 to the end of the list. Using + or += on a list is similar to using extend().
- **list.index(elem) :**searches for the given element from the start of the list and returns its index.
- **list.remove(elem) :** searches for the first instance of the given element and removes it.
- **list.sort() :** sorts the list in place (does not return it). (The sorted() function shown below is preferred.)
- **list.reverse() :**reverses the list in place (does not return it)
- l**ist.pop(index) :** removes and returns the element at the given index. Returns the rightmost element if index is committed.

Lists

List Methods:
EXAMPLES:

```
list = [14, 22, 19, 20];
print(list)
list.append(30)
print(list)
list.insert(0,43)
list1=[1,2,3,4]
list.extend(list1)
print(list)
print(list.index(14))
list.remove(22)
print(list)
list.sort()
print("sort::",list)
list.remove(1)
print("remove::",list)
list.reverse()
print("reverse::",list)
print(list.pop())#it will last index only
print(list.pop(2))
```

```
C:\Users\ravikiran\Desktop\python>a.py
[14, 22, 19, 20]
[14, 22, 19, 20, 30]
[43, 14, 22, 19, 20, 30, 1, 2, 3, 4]
1
[43, 14, 19, 20, 30, 1, 2, 3, 4]
sort:: [1, 2, 3, 4, 14, 19, 20, 30, 43]
remove:: [2, 3, 4, 14, 19, 20, 30, 43]
reverse:: [43, 30, 20, 19, 14, 4, 3, 2]
2
20
```

Lists

List Slices:

- Slices work on lists just as with strings, and can also be used to change sub-parts of the list.

```
list = ['a', 'b', 'c', 'd']
print (list[1:-1])
list[0:2] = 'z'
print (list )
```

```
C:\Users\ravikiran\Desktop\python>a.py
['b', 'c']
['z', 'c', 'd']
```

Lists

Lists are Mutable:

- Items in the list are mutable i.e. after creating a list you can change any item in the list.

```
color_list=["A", "B", "C", "D"]
print("color_list[0]",color_list[0])
color_list[0]="Z" # Change the value of first item "A" to "Z"
print(color_list)
print("color_list[0]",color_list[0])
```

```
C:\Users\ravikiran\Desktop\python>a.py
color_list[0] A
['Z', 'B', 'C', 'D']
color_list[0] Z
```

Convert a list to a tuple:

```
tupl=(1, 2, 3, 4)
print("tuple:",tupl)
listx=list(tupl)
print("list:",listx)
```

```
C:\Users\ravikiran\Desktop\python>a.py
tuple: (1, 2, 3, 4)
list: [1, 2, 3, 4]
```

Digital Lync
INNOVATION - EDUCATION - INCUBATION

List Comprehensions:

- List comprehensions provides a concise way of creating a list, in a very natural and easy way.
- It consists of brackets containing expression followed by a for clause, then Zero or more if clause

Eg:

```
# normal way of writing for loop to print odd numbers
for x in (range(10)):
    if x%2 != 0:
        print x
```

```
# using Comprehensions print odd numbers
odd = [i for i in range(10) if i%2 != 0]
odd
```

```
S = [x**2 for x in range(10)]
V = [2**i for i in range(13)]
print("S",S)
print("V",V)
```

```
C:\Users\ravikiran\Desktop\python>a.py
S [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
V [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096]
```

Assignment - 7

Digital Lync
INNOVATION - EDUCATION - INCUBATION

1. Write a Python program to sum all the items in a list.

2. Write a Python program to get the largest number from a list.

3. Write a Python program to remove duplicates from a list.

4. Write a Python program to check a list is empty or not.

5. Write a Python program to print a specified list after removing the 0th, 4th and 5th elements.
   Sample List : [‘A', ‘B', ‘C', ‘D', ‘E', ‘F']
   Expected Output : [‘B', ‘C', ‘D]

6. Write a Python program to find the second smallest number in a list.