TECHNO INDIA
SILIGURI INSTITUTE
OF TECHNOLOGY
A Satyam Roychowdhury Initiative

List and Dictionary

List

- Lists are Python’s most flexible ordered collection object type.
- lists can contain any sort of object: numbers, strings, and even other lists.
- lists may be changed in-place by assignment to offsets and slices, list method calls, deletion statements, and more—they are mutable objects.

List

- Ordered collection of arbitray objects
- Accessed by offset
- Variable length, heterogenous, and arbitrarily nestable
- Of the category mutable sequence
- Arrays of object reference

Operation	Interpretation
`L = []`	An empty list
`L = [0, 1, 2, 3]`	Four items: indexes 0..3
`L = ['abc', ['def', 'ghi']]`	Nested sublists
`L = list('spam')`	Lists of an iterable's items, list of successive integers
`L = list(range(-4, 4))`	
`L[i]`	Index, index of index, slice, length
`L[i][j]`	
`L[i:j]`	
`len(L)`	

Operation	Interpretation
`L1 + L2`	Concatenate, repeat
`L * 3`	
`for x in L: print(x)`	Iteration, membership
`3 in L`	
`L.append(4)`	Methods: growing
`L.extend([5,6,7])`	
`L.insert(I, X)`	
`L.index(1)`	Methods: searching
`L.count(X)`	
`L.sort()`	Methods: sorting, reversing, etc.
`L.reverse()`	
`del L[k]`	Methods, statement: shrinking

Operation	Interpretation
`del L[i:j]`	
`L.pop()`	
`L.remove(2)`	
`L[i:j] = []`	
`L[i] = 1`	Index assignment, slice assignment
`L[i:j] = [4,5,6]`	
`L = [x**2 for x in range(5)]`	List comprehensions and maps (Chapters 14, 20)
`list(map(ord, 'spam'))`	

Basic list operation

```
>>> len([3,4,5])
3
>>> [1,2,3]+[5,6]
[1, 2, 3, 5, 6]
>>> [3,4]*3
[3, 4, 3, 4, 3, 4]
>>> [1,2]+list('MCA')
[1, 2, 'M', 'C', 'A']
>>> l=[0]*5
>>> l
[0, 0, 0, 0, 0]
>>> l=[0 for _ in range(5)]
```

Indexing, Slicing and Metrices

```
>>> l=[10,'MCA',2.5,[4,5]]
>>> l[0]
10
>>> l[1:]
['MCA', 2.5, [4, 5]]
>>> l[1:3]=[3,4]
>>> l
[10, 3, 4, [4, 5]]
>>> l[1:3]=20
```

List support operations that change a list object in-place

```
>>> list2=[[1,2,3],[6,7,8],[3,4,5]]
>>> list2
[[1, 2, 3], [6, 7, 8], [3, 4, 5]]
>>> list2[1]
[6, 7, 8]
>>> list[1][1]
```

```
>>> L=[3,4,5]
>>> L
[3, 4, 5]
>>> L.append(6)
>>> L
[3, 4, 5, 6]
```

```
>>> L.extend([1,1])
>>> L
[3, 4, 5, 6, 1, 1]
>>> L=[1,2,3]
>>> L
[1, 2, 3]
>>> L.append([9,10])
>>> L
[1, 2, 3, [9, 10]]
```

```
>>> l=[4,6,2,10]
>>> l
[4, 6, 2, 10]
>>> l.sort()
>>> l
[2, 4, 6, 10]
>>> l.sort(reverse=True)
>>> l
[10, 6, 4, 2]
```

```
>>> l=['kajal','Samjhana','susmita','sudip']
>>> l
['kajal', 'Samjhana', 'susmita', 'sudip']
>>> l.sort(reverse=True)
>>> l
['susmita', 'sudip', 'kajal', 'Samjhana']
```

- delete an item from the end of the list,
- pop method also accepts an optional offset of the item to be deleted
- remove method remove an item by value
- del statement to delete an item or section in-place

```
>>> l
[2, 3, 4, 3, 2]
>>> del l[3:]
>>> l
[2, 3, 4]
>>> del l
>>> l
Traceback (most recent call last):
  File "<pyshell#68>", line 1, in <module>
    l
NameError: name 'l' is not defined
```

```
>>> l=[1,2,3,5,4,3,2,1]
>>> l
[1, 2, 3, 5, 4, 3, 2, 1]
>>> l.remove(1)
>>> l
[2, 3, 5, 4, 3, 2, 1]
>>> l.remove(1)
>>> l
[2, 3, 5, 4, 3, 2]
>>> l.pop(2)
5
>>> l
[2, 3, 4, 3, 2]
```

- WAP to check whether first element of your list is occured in other place of your list or not.
- WAP to delete 2nd occurance of first elemnent of your list.
- WAP to delete all the occurance of list only keeping the first one.

- WAP to check whether first element of your list is occured in other place of your list or not.

- WAP to print 2nd occurance of 1st element(if any).
- WAP to remove one element from a list.

- WAP to delete all the occurance of list only keeping the first one.

Dictionaries

- Apart from lists, dictionaries are the most flexible buil-n datatype in Python.
- It can replace many of the searching algorithms and data structures.
  - ✓ Accessed by key, not offset: associative arrays or hashes. ot associate a set of values with keys
  - ✓ Unordered collections of arbitrary objects
  - ✓ Variable length, heterogenous, and arbitrarily nestable
  - ✓ Of the category "mutable mapping: can be changed in place by assigning to index
  - ✓ Table of object referneces (hash tables)

Dictionaries Operations

Operation	Interpretation
`D = {}`	Empty dictionary
`D = {'spam': 2, 'eggs': 3}`	Two-item dictionary
`D = {'food': {'ham': 1, 'egg': 2}}`	Nesting
`D = dict(name='Bob', age=40)`	Alternative construction techniques:
`D = dict(zip(keyslist, valslist))`	keywords, zipped pairs, key lists
`D = dict.fromkeys(['a', 'b'])`	
`D['eggs']`	Indexing by key
`D['food']['ham']`	
`'eggs' in D`	Membership: key present test
`D.keys()`	Methods: keys,
`D.values()`	values,

Operation	Interpretation
`del D[key]`	Deleting entries by key
`list(D.keys())`	Dictionary views (Python 3.0)
`D1.keys() & D2.keys()`	
`D = {x: x*2 for x in range(10)}`	Dictionary comprehensions (Python 3.0)
`D.items()`	keys+values,
`D.copy()`	copies,
`D.get(key, default)`	defaults,
`D.update(D2)`	merge,
`D.pop(key)`	delete, etc.
`len(D)`	Length: number of stored entries
`D[key] = 42`	Adding/changing keys

TECHNO INDIA
SILIGURI INSTITUTE
OF TECHNOLOGY