LIST

List:

- It is a general purpose most widely used in data structures
- List is a collection which is ordered and changeable and allows duplicate members. (Grow and shrink as needed, sequence type, sortable).
- To use a list, you must declare it first. Do this using square brackets and separate values with commas.
- We can construct / create list in many ways.

Ex:

```
>>> list1=[1,2,3,'A','B',7,8,[10,11]]
>>> print(list1)
[1, 2, 3, 'A', 'B', 7, 8, [10, 11]]
----------
>>> x=list()
>>> x
[]
----------
>>> tuple1=(1,2,3,4)
>>> x=list(tuple1)
>>> x
[1, 2, 3, 4]
```

List operations:

These operations include indexing, slicing, adding, multiplying, and checking for membership

Basic List Operations:

Lists respond to the + and * operators much like strings; they mean concatenation and repetition here too, except that the result is a new list, not a string.

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

Indexing, Slicing, and Matrixes

Because lists are sequences, indexing and slicing work the same way for lists as they do for strings.

Assuming following input −

L = ['mrcet', 'college', 'MRCET!']

Python Expression	Results	Description
L[2]	MRCET	Offsets start at zero
L[-2]	college	Negative: count from the right
L[1:]	['college', 'MRCET!']	Slicing fetches sections

L>> list1[2:5]

[3, 4, 5]

```
>>> list1[:6]
[1, 2, 3, 4, 5, 6]
>>> list1[1:2:4]
[2]
>>> list1[1:8:2]
[2, 4, 6, 8]
```

List methods:

The list data type has some more methods. Here are all of the methods of list objects:

- Del()
- Append()
- Extend()
- Insert()
- Pop()
- Remove()
- Reverse()
- Sort()

Delete: Delete a list or an item from a list

```
>>> x=[5,3,8,6]
>>> del(x[1])            #deletes the index position 1 in a list
>>> x
[5, 8, 6]
--------
>>> del(x)
>>> x                    # complete list gets deleted
```

Append: Append an item to a list

```
>>> x=[1,5,8,4]
>>> x.append(10)
>>> x
[1, 5, 8, 4, 10]
```

Extend: Append a sequence to a list.

```
>>> x=[1,2,3,4]

>>> y=[3,6,9,1]

>>> x.extend(y)

>>> x

[1, 2, 3, 4, 3, 6, 9, 1]
```

Insert: To add an item at the specified index, use the insert () method:

```
>>> x=[1,2,4,6,7]
>>> x.insert(2,10)   #insert(index no, item to be inserted)

>>> x

[1, 2, 10, 4, 6, 7]

----------

>>> x.insert(4,['a',11])

>>> x

[1, 2, 10, 4, ['a', 11], 6, 7]
```

Pop: The pop() method removes the specified index, (or the last item if index is not specified) or simply pops the last item of list and returns the item.

```
>>> x=[1, 2, 10, 4, 6, 7]

>>> x.pop()

7

>>> x

[1, 2, 10, 4, 6]

----------

>>> x=[1, 2, 10, 4, 6]

>>> x.pop(2)
```

10

>>> x

[1, 2, 4, 6]

Remove: The **remove()** method removes the specified item from a given list.

>>> x=[1,33,2,10,4,6]

>>> x.remove(33)

>>> x
[1, 2, 10, 4, 6]

>>> x.remove(4)

>>> x

[1, 2, 10, 6]

Reverse: Reverse the order of a given list.

>>> x=[1,2,3,4,5,6,7]

>>> x.reverse()

>>> x

[7, 6, 5, 4, 3, 2, 1]

Sort: Sorts the elements in ascending order

>>> x=[7, 6, 5, 4, 3, 2, 1]

>>> x.sort()

>>> x

[1, 2, 3, 4, 5, 6, 7]

- - - - - - - - - - - - - - -

>>> x=[10,1,5,3,8,7]

>>> x.sort()

>>> x

[1, 3, 5, 7, 8, 10]

Dictionaries:

A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.

- Key-value pairs
- Unordered

We can construct or create dictionary like:

X={1:'A',2:'B',3:'c'}

X=dict([('a',3) ('b',4)] X=dict('A'=1,'B' =2)

Example:

```
>>> dict1 = {"brand":"mrcet","model":"college","year":2004}
>>> dict1
{'brand': 'mrcet', 'model': 'college', 'year': 2004}
```

Operations and methods:

Methods that are available with dictionary are tabulated below. Some of them have already been used in the above examples.

Method	**Description**
clear()	Remove all items form the dictionary.

copy()	Return a shallow copy of the dictionary.
fromkeys(seq[, v])	Return a new dictionary with keys from seq and value equal to v (defaults to None).
get(key[,d])	Return the value of key. If key doesnot exit, return d (defaults to None).
items()	Return a new view of the dictionary's items (key, value).
keys()	Return a new view of the dictionary's keys.
pop(key[,d])	Remove the item with key and return its value or d if key is not found. If d is not provided and key is not found, raises KeyError.
popitem()	Remove and return an arbitary item (key, value). Raises KeyError if the dictionary is empty.
setdefault(key[,d])	If key is in the dictionary, return its value. If not, insert key with a value of d and return d (defaults to None).
update([other])	Update the dictionary with the key/value pairs from other, overwriting existing keys.
values()	Return a new view of the dictionary's values

Below are some dictionary operations:

To access specific value of a dictionary, we must pass its key,

>>> dict1 = {"brand":"mrcet","model":"college","year":2004}

>>> x=dict1["brand"]

>>> x

'mrcet'

– – – – – – – – – – – – – –

To access keys and values and items of dictionary:

>>> dict1 = {"brand":"mrcet","model":"college","year":2004}

>>> dict1.keys() dict_keys(['brand', 'model', 'year'])

>>> dict1.values() dict_values(['mrcet', 'college', 2004])

>>> dict1.items()

dict_items([('brand', 'mrcet'), ('model', 'college'), ('year', 2004)])

– –

>>> for items in dict1.values(): print(items)

mrcet college 2004

>>> for items in dict1.keys(): print(items)

brand model year

>>> for i in dict1.items(): print(i)

('brand', 'mrcet')

('model', 'college')

('year', 2004)

Some more operations like:

- Add/change
- Remove
- Length
- Delete

Add/change values: You can change the value of a specific item by referring to its key name

>>> dict1 = {"brand":"mrcet","model":"college","year":2004}

```
>>> dict1["year"]=2005
>>> dict1
{'brand': 'mrcet', 'model': 'college', 'year': 2005}
```

Remove(): It removes or pop the specific item of dictionary.

```
>>> dict1 = {"brand":"mrcet","model":"college","year":2004}
>>> print(dict1.pop("model")) college
>>> dict1
{'brand': 'mrcet', 'year': 2005}
```

Delete: Deletes a particular item.

```
>>> x = {1:1, 2:4, 3:9, 4:16, 5:25}
>>> del x[5]
>>> x
```

Length: we use len() method to get the length of dictionary.

```
>>>{1: 1, 2: 4, 3: 9, 4: 16}
{1: 1, 2: 4, 3: 9, 4: 16}
>>> y=len(x)
>>> y 4
```

Iterating over (key, value) pairs:

```
>>> x = {1:1, 2:4, 3:9, 4:16, 5:25}
>>> for key in x:
print(key, x[key])
```

```
1 1
2 4
```

3 9

4 16

5 25

>>> for k,v in x.items(): print(k,v)

1 1

2 4

3 9

4 16

5 25

List of Dictionaries:

>>> customers = [{"uid":1,"name":"John"},

{"uid":2,"name":"Smith"},

{"uid":3,"name":"Andersson"},

]

>>> >>> print(customers)

[{'uid': 1, 'name': 'John'}, {'uid': 2, 'name': 'Smith'}, {'uid': 3, 'name': 'Andersson'}]

Print the uid and name of each customer

>>> for x in customers: print(x["uid"], x["name"])

1 John

2 Smith

3 Andersson

Modify an entry, This will change the name of customer 2 from Smith to Charlie

>>> customers[2]["name"]="charlie"

>>> print(customers)

[{'uid': 1, 'name': 'John'}, {'uid': 2, 'name': 'Smith'}, {'uid': 3, 'name': 'charlie'}] ## Add a new field to each entry

>>> for x in customers:

x["password"]="123456" # any initial value

```
>>> print(customers)
[{'uid': 1, 'name': 'John', 'password': '123456'}, {'uid': 2, 'name': 'Smith', 'password': '123456'}, {'uid': 3, 'name': 'charlie', 'password': '123456'}]

## Delete a field
>>> del customers[1]
>>> print(customers)
[{'uid': 1, 'name': 'John', 'password': '123456'}, {'uid': 3, 'name': 'charlie', 'password': '123456'}]

>>> del customers[1]
>>> print(customers)
[{'uid': 1, 'name': 'John', 'password': '123456'}] ## Delete all fields

>>> for x in customers: del x["uid"]
>>> x
{'name': 'John', 'password': '123456'}
```