Computer Science

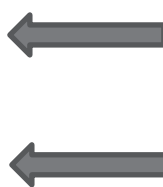## List, Tuple and Dictionaries

Notes

LIST – A list is a standard data type of python that can store a sequence of values belonging to any type. List is mutable. You can change elements of a list in place. Following are some lists in python:

[ ]   empty list

[1,2,3]   list of integers

['a',1,'b',2] list of mixed values

List in Python are stored as individual characters in contiguous locations,

with two-way index for each location.:

0	1	2	3	4	5
P	Y	T	H	O	N
-6	-5	-4	-3	-2	-1

LIST OPERATORS –The various operators that can be used to manipulate list in multiple ways :

Joining list(+) – The + operator is used to join two lists .

>>>l1=[1,2,3]

>>>l2=[4,5]

>>>l1+l2

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

List Replication operator (*) – This operator needs two type of operands. A list and a number . The number operand tells the number of times list is to be repeated.

>>>l1=[1,2]

>>>l1*2
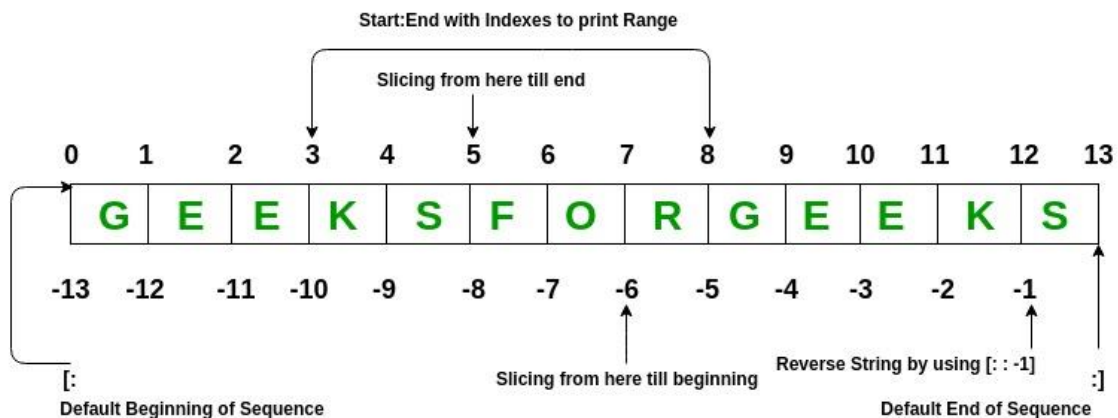
>>>[1,2,1,2]

Membership Operators –There are two membership operators for list. These are **in** and **not in**. Both membership operators require that both operands used with them are of list type.

Comparison Operator –All relational operators(>,<,>=,<=,==,!=) apply to list also.

List Slices – It refers to a part of the string containing some contiguous characters from the list same as string.

List Functions– Python also offers many built-in functions and methods for list manipulation. These can be applied to list as per following syntax :

<listobject> . <method name> ( )

- **APPEND()** – It adds a single item to the end of the list .

    Syntax :   list.append(<item>)

- **INDEX() -** It returns the index of first matched item from the list.

  Syntax :   list.index(<item>)

- **EXTEND() –**It is used for adding multiple elements to a list.

  Syntax :   list.extend(<list>)

- **INSERT() –** It is used to insert an item at a given position.

  Syntax :   list.insert(<pos> , <item>)

- **POP() -** It is used to remove the item from the list.

  Syntax :   list.pop(<index>)

- **REMOVE()**-It is used to remove the first occurrence of given item from the list.

  Syntax : list.remove(<value>)

- **CLEAR()-**It is used to remove all the items from the list and the list becomes empty.

  Syntax : list.clear()

- **COUNT()-T**his function returns the count of the item passed as a parameter.

  Syntax : list.count(<item>)

- **REVERSE()–**It will reverse the items of the list.

  Syntax: list.reverse()

- **SORT()-**It will sort the items of the list in increasing order by default.

        Syntax : list.sort()

MEMORY MAP

LIST FUNCTIONS

index( )	append( )
extend( )	insert( )
pop( )	remove( )
clear( )	count( )
reverse( )	sort( )

List Manipulation :

1. Updating elements from the list – By assigning new value to the element of the list through its index will change an element. Ex:
   >>>l1 =[1,2,3]
   >>>l1[2] = 4
   >>>l1
   [1,2,4]

2. Deleting elements from the list – the def statement is used to remove an individual item, or to remove all items identified by a slice.
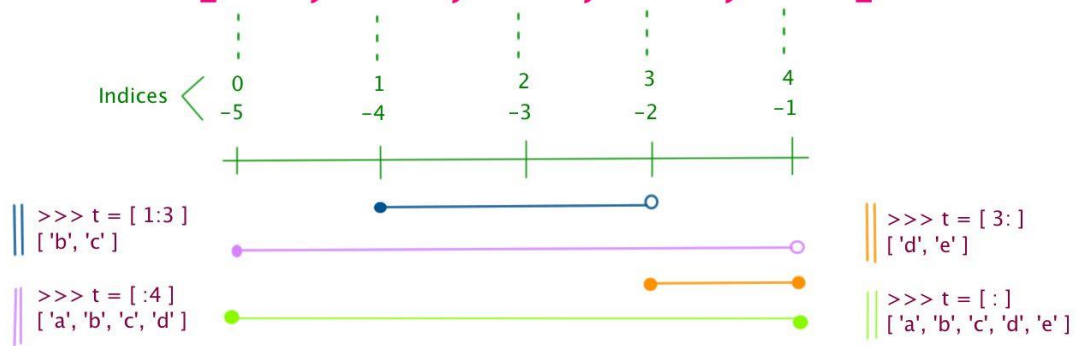   >>>l1=[10,12,13,14]
   >>>del l1[2]
   l1
   [10,12,14]

3. Slicing the lists – Like string a subpart of the list will be displayed.
   >>>l1=[10,12,14,20,22,24,30,32,34]

>>>s=l1[3 :-3]

s

[20,22,24]

Slicing

t = ['a', 'b', 'c', 'd', 'e']

TUPLE

The tuple are depicted through round brackets .Ex :

()   empty tuple

(3,)   tuple with one number

(1,2,3)   tuple of integers

(1,2.5,'a',2) tuple of mixed value types

Tuple Functions and Methods

1.The **len()** method – It returns length of the tuple, i.e. ,the count of elements in the tuple.Ex:

>>>tp=(1,2,3)

>>>len(tp)

3

2.The **max()** method – It returns the element from the tuple having maximum value.Ex:

\>\>\>tp=(10,5,78)

\>\>\>max(tp)

78

3.The **min()** method – It returns the element from the tuple having minimum value. Ex:

\>\>\>tp =(9,3,1)

\>\>\>min(tp)

1

4.The **index()** method – It returns the index of an existing element of a tuple.Ex:

\>\>\>t1=(3,4,5,6)

\>\>\>t1.index(5)

2

5.The **count()** function – This method returns the count of a member element in a given tuple. Ex:

\>\>\>t1=(2,3,4,5,6,2,7,8,2)

\>\>\>t1.count(2)

3

6.The tuple() method – It is constructor that can be used to create tuples from different types of values.

\>\>\>t= tuple("abc")

\>\>\>t

('a', 'b', 'c')

MEMORY MAP

TUPLE FUNCTIONS

len( )	max( )
index( )	min( )
count( )	tuple()

Tuple Operations :

1.Traversing a Tuple – Traversing a tuple means accessing and processing each element of the tuple.Ex:

```
>>>t=('p','u','r','e')
>>>for a in t:
        print(t[a])
p
u
r
e
```

2.Joining Tuple – The + operator, when used with two tuples, joins two tuples.Ex:

```
>>>t1=(1,2,3)
>>>t2=(4,5)
>>>t1 + t2
(1,2,3,4,5)
```

3.Replicating Tuple – The * operator is used to replicate a tuple specified number of times.Ex:

```
>>>t1=(1,2)
>>>t1*2
1212
```

4.Slicing the tuple – It gives a sub part of the tuple like list and string . Ex:

```
>>>t1=(10,12,14,20,22,24,30,32,34)
>>>seq=t1[3:-3]
>>>seq
(20,22,24)
```

PYTHON TUPLES VS LISTS

TUPLES		LISTS
The items are surrounded in paranthesis ().	Syntax	The items are surrounded in square brackets [ ].
Tuples are immutable in nature.	Mutability	Lists are mutable in nature.
There are 33 available methods on tuples.	Methods	There are 46 available methods on lists.
In dictionary, we can create keys using tuples.	Usability	In dictionary, we can't use lists as keys.

DICTIONARIES :

Dictionaries are a collection of some key-value pairs. .Dictionaries are mutable,unordered collections with elements in the form of a key: value pairs that associates keys to values.The syntax to create a dictionary is :

<dictionary-name> = {<key> : <value> , <key> :<value>……………}

Python Dictionary

```
              key    value    key    value
               ↓      ↓        ↓      ↓
py_dict = { 1: 'Apple', 2: 'OnePlus' }
              └───┬───┘      └───┬───┘
                Item 1          Item 2
```

Ex: emp = {'salary' : 10000 , 'age' :24 , 'name' : 'john' }
>>>emp.keys()
['salary','age','name']
>>>emp.values()
[10000,24,'john']

Dictionary Functions and Methods :

1.The len() method – This method returns length of the dictionary (the count of elements) .Ex:
>>> emp = {'salary' : 10000 , 'age' :24 , 'name' : 'john' }
>>>len(emp)
3

2.The clear() method – This method removes all items from the dictionary and the dictionary becomes empty.
>>> emp = {'salary' : 10000 , 'age' :24 , 'name' : 'john' }
>>>emp.clear()
{ }
3.The get() method – With this method we can get the item with the given key.
>>> emp = {'salary' : 10000 , 'age' :24 , 'name' : 'john' }
>>>emp.get('age')
24

4.The items() method – This method returns all of the items in the dictionary as a sequence of(key,value) tuples .
emp = {'salary' : 10000 , 'age' :24 , 'name' : 'john' }
mylist=emp.items()
for x in mylist :
        print(x)
the output will be like this:
('salary' ,10000)
('age' ,24)

('name', 'john')

5.The keys() method – This method returns all of the keys in the dictionary as a sequence of keys.
>>> emp = {'salary' : 10000 , 'age' :24 , 'name' : 'john' }
>>>emp.keys()
>>>['salary' ,'age' ,'name']

6.The values() method - This method returns all of the values in the dictionary as a sequence (a list).
>>>emp.values()
>>>[10000, 24, 'john']

7.The update() method – This method merges key:value pairs from the new dictionary into the original dictionary, adding or replacing as needed . The items in the new dictionary are added to the old one and override any items already there with the same keys.
This method returns all of the keys in the dictionary as a sequence of keys.
>>> emp1 = {'salary' : 10000 , 'age' :24 , 'name' : 'john' }
>>> emp2 = {'salary' : 20000 , 'age' :25 , 'name' : 'riya' }
>>>emp1.update(emp2)
>>>emp1
{'salary' : 20000 , 'age' :25 , 'name' : 'riya' }

MEMORY MAP

DICTIONARY FUNCTIONS

len( )	clear( )
keys( )	get( )
values( )	update( )

Dictionary Operations :

1.Traversing a Dictionary : Traversal of a collection means accessing and processing each element of it.

```
>>> emp1 = {'salary' : 10000 , 'age' :24 , 'name' : 'john' }
>>>for key in emp1:
        print(key, emp1[key])
salary : 10000
age  : 24
name : john
```

2.Adding elements to a dictionary: A new element can be added in the dictionary but it must not exist in dictionary and must be unique . if the key already exists the this statement will change the value of existing key and no new entry will be added.

```
>>> emp1 = {'salary' : 10000 , 'age' :24 , 'name' : 'john' }
>>>emp1['dept'] = 'sales'
>>>emp1
{'salary' : 10000 , 'age' :24 , 'name' : 'john' , 'dept' : 'sales'}
```

3.Deleting elements from a dictionary :There are two methods fo deleting elements from a dictionary.

(a) To delete a dictionary element or a dictionary entry, i.e., a key:value pair

```
>>>emp1 ={'salary' : 10000 , 'age' :24 , 'name' : 'john' }
>>>del emp1['age']
>>>emp1
 {'salary' : 10000 , 'name' : 'john' }
```

(b) To delete elements using pop() method

```
>>>emp1 ={'salary' : 10000 , 'age' :24 , 'name' : 'john' }
>>>emp1.pop('age')
>>>24
```

{'salary' : 10000 , 'name' : 'john' }

4.Checking for a existence of a key:The membership operator in and not in work with dictionary to check the existence of a element.

>>>emp1={'salary' : 10000 , 'name' : 'john' }
>>>'age' in emp1
True
>>> 'riya' not in emp1
True

MEMORY MAP

```
                          ┌───────────┐
                          │ DATA TYPE │
                          └───────────┘
                        ╱               ╲
              ┌───────────┐         ┌─────────┐
              │ IMMUTABLE │         │ MUTABLE │
              └───────────┘         └─────────┘
               ╱        ╲            ╱         ╲
       ┌────────┐   ┌───────┐   ┌──────┐   ┌────────────┐
       │ STRING │   │ TUPLE │   │ LIST │   │ DICTIONARY │
       └────────┘   └───────┘   └──────┘   └────────────┘
```

* SEQUENCE UNICODE CHARACTER * VALUE CAN NOT BE MODIFIED * SEQUENCE OF CHARACTER * TWO WAY INDEXING	* ORDERED SEQUENCE * ADDING AND REMOVING VALUES NOT POSSIBLE * VALUES STORED ALPHA-NUMERIC *VALUE CAN NOT MODIFIED	* ORDERED SEQUENCE * ADDING AND REMOVING VALUE IS POSSIBLE * VALUES STORED ALPHA-NUMERIC * VALUES CAN BE CHANGED	* UNORDERED SET OF KEY:VALUE PAIRS * EACH KEY MUST BE UNIQUE * INDEXED BY IMMUTABL KEY
CODE = "POWERFUL"	T1= (1,'A',3.5,"ZERO",2)	L=['A',3,6,"THREE",4.5,"TWO"]	D1={ "A":STRING, 1:"NUMBER"}