Lists

Main functions of a list

- `len(list)` -> no of elements in the **list**
- `list.count(elt)` -> find the no of occurrences of an element **elt** in the list
- `list.append(elt)` -> append the **elt** at the end of the **list**
- `list.insert(index,elt)` -> insert the **elt** at the specified **index**
- `list.remove(elt)` -> remove the first occuring **elt** in the **list**
- `list.pop(index)` -> remove the element present in the **index**
- `del list[index]` -> element present in the **index** will be removed from the **list**
- `list.sort()` -> sort the elements in the **list**
- `list.reverse()` -> reverse elements in the **list**
- `list1.extend(list2)` or `list1 + list2` -> elements in **list2** will be appended at the end of **list1**
- `sum(list)` , `max(list)` , `min(list)` -> sum/max/min operations performed in the **list**

```
my_list = []
my_list = [1,2,3,4]
my_list = [[1,2],[3,4]]
my_list = [1,'a',[8,9]]

print(my_list)
```

```
[1, 'a', [8, 9]]
```

```
print(len(my_list))
```

```
3
```

```
my_list.append("1")
print(my_list)
```

```
[1, 'a', [8, 9], '1']
```

```
my_list.insert(1,'b')
print(my_list)
```

```
[1, 'b', 'a', [8, 9], '1']
```

```
my_list.remove(1) #remove the first occuring element
print(my_list)
```

```
['b', 'a', [8, 9], '1']
```

```
new_list = my_list + [0,10,20,30]
```

```
print(new_list)
```

```
['b', 'a', [8, 9], '1', 0, 10, 20, 30]
```

```
print(new_list.pop(2))
print(new_list)
```

```
[8, 9]
['b', 'a', '1', 0, 10, 20, 30]
```

```
print('1' in new_list)
print('1' not in new_list)
```

```
True
False
```

```
new_list.reverse()
print(new_list)
```

```
[30, 20, 10, 0, '1', 'a', 'b']
```

```
num = [0,6,4,3,8,1,2,9,10]
num.sort()
print(num)
```

```
[0, 1, 2, 3, 4, 6, 8, 9, 10]
```

```
num1 = [1,2,3,4,5,6,7,8,9,10]
num2 = num1 #pass by reference
num2[0] = 11
print(num1)
```

```
[11, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
print(num[0])
print(num[-1])
print(num[0:5])
print(num[0:5:2])
print(num[::2])
```

```
0
10
[0, 1, 2, 3, 4]
[0, 2, 4]
[0, 2, 4, 8, 10]
```

```
print(num.count(10)) #count the number of occurrences
```

```
1
```

```
print(max(num))
print(min(num))
```

```
print(sum(num))
```

```
10
0
50
```

```
for n in num:
  print(n*2)
```

```
0
2
4
6
8
12
16
18
20
```

List Comprehension

```
num = [0, 1, 2, 3, 4, 6, 7, 8, 9, 10]
```

```
num_squared = [i**2 for i in num]
print(num_squared)
```

```
[0, 1, 4, 9, 16, 36, 49, 64, 81, 100]
```

```
num_odd_squared = [i**2 for i in num if i%2==1]
print(num_odd_squared)
```

```
[1, 9, 49, 81]
```

```
matrix = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]
#transpose
transpose = []
for i in range(len(matrix[0])):
  lst = []
  for row in matrix:
    lst.append(row[i])
  transpose.append(lst)
print(transpose)
```

```
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

```
transpose = [[row[i] for row in matrix] for i in range(len(matrix[0]))]
print(transpose)
```

```
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
```

Tuples

Main functions of a tuple

- `len(tuple)` -> length of a **tuple**
- `del tuple` -> delete the **tuple**
- `tuple.count(elt)` -> count the no. of occurrences of **elt** in the **tuple**
- `tuple.index(elt)` -> find the index of the first occurrence of **elt**
- `sum(tuple)` , `max(tuple)` , `min(tuple)` -> find the max, sum & min of the tuple
- `tuple1 + tuple2` -> appending 2 tuples. `.append()` function won't work
- `sorted(tuple)` -> sorts a **tuple** and returns a **list**. Then change it to tuple by `tuple(list)`

```
mtp = ()
print(mtp)
mtp = (1,2,3)
print(mtp)
mtp = (4,5,6,[1,2,3])
print(mtp)
```

```
()
(1, 2, 3)
(4, 5, 6, [1, 2, 3])
```

```
mtp = ('stringtest')
print(type(mtp))
mtp = ('stringtest', )
print(type(mtp))
```

```
<class 'str'>
<class 'tuple'>
```

```
mtp = (0, 1, 2, 3, 4, 5, 6)
print(len(mtp))
```

```
7
```

```
print(mtp[0])
```

```
0
```

```
mtp[1] = 2
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-28-3ee687856bf3> in <module>()
```

```
del mtp[1]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-29-4eb265e0e971> in <module>()
----> 1 del mtp[1]

TypeError: 'tuple' object doesn't support item deletion
```

SEARCH STACK OVERFLOW

```
del mtp
```

```
t = (1, 3, 5) + (2, 4, 6)
print(t)
t = (1, 2, 3).append((4,5,6))
print(t)
```

```
(1, 3, 5, 2, 4, 6)
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-34-023d2dcc2a7f> in <module>()
      1 t = (1, 3, 5) + (2, 4, 6)
      2 print(t)
----> 3 t = (1, 2, 3).append((4,5,6))
      4 print(t)

AttributeError: 'tuple' object has no attribute 'append'
```

SEARCH STACK OVERFLOW

```
sorted_t = sorted(t)
print(sorted_t)
print(type(sorted_t))
new_t = tuple(sorted_t)
print(new_t)
```

```
[1, 2, 3, 4, 5, 6]
<class 'list'>
(1, 2, 3, 4, 5, 6)
```

```
print(min(t))
print(max(t))
print(sum(t))
```

```
1
6
21
```

```
print(t.count(1))
print(t.index(1))
```

```
1
0
```

```
for elt in t:
  print(elt)
```

```
1
3
5
2
4
6
```

```
t = (-3, -2, -1, 0, 1, 2, 3, 4, 5, 6)
[elt**2 if elt%2==0 else elt**3 for elt in t if elt>0] #square in case of even numbers, cube i
```

```
[1, 4, 27, 16, 125, 36]
```

Sets

Main functionalities of sets

- `set.add(elt)` -> add the **elt** to the **set**
- `set.update([elts])` -> add many elements in the input **list of elts** to the **set**
- `set.discard(elt)` -> removes an **elt** if that element **presents or not** in the set
- `set.rermove(elt)` -> removes the **elt** only if the element presents in the set. If not, **throw error**
- `set.pop()` -> removes a random element from set
- `set.clear()` -> remove all the elements

Set operations

- `s1.union(s2)` or `s1 | s2` -> union of two sets (i.e.) all the elements will be put together
- `s1.intersection(s2)` or `s1 & s2` -> Intersection of two sets (i.e) common elements
- `s1.difference(s2)` or `s1 - s2` -> elements in s1 but not in s2
- `s1.symmetric_difference(s2)` or `s1 ^ s2` -> all elements in s1&s2 but not the common elements
- `s1.issubset(s2)` -> Boolean : true if all elements in s1 present in s2
- `s1.isdisjoint(s2)` -> Boolean : true if no elements in s1 present in s2

`frozenset()` -> no addition/removal of the keys are allowed. All other operations are allowed

```
s = {1,2,3}
print(s)
```

```
{1, 2, 3}
```

```
s.add(4)
print(s)
```

```
{1, 2, 3, 4}
```

```
s.update([1,4,5,6]) #add multiple elements
print(s)
s.update([11],{7,8,9})
print(s)
```

```
{1, 2, 3, 4, 5, 6}
{1, 2, 3, 4, 5, 6, 7, 8, 9, 11}
```

```
s.discard(11)
print(s)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

```
s.remove(100)
print(s)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-60-c66074f56fd7> in <module>()
----> 1 s.remove(100)
      2 print(s)

KeyError: 100
```

SEARCH STACK OVERFLOW

```
s.discard(100)
print(s)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

```
s.pop() #removes a random element
print(s)
```

```
{2, 3, 4, 5, 6, 7, 8, 9}
```

```
s.clear()
print(s)
```

```
set()
```

Set Operations

```
s1 = {1,2,3,4,5}
s2 = {3,4,5,6,7}
```

```
print(s1.union(s2))
```

```
print(s1 | s2)
```

```
{1, 2, 3, 4, 5, 6, 7}
{1, 2, 3, 4, 5, 6, 7}
```

```
print(s1.intersection(s2))
print(s1 & s2)
```

```
{3, 4, 5}
{3, 4, 5}
```

```
print(s1.difference(s2)) #Elements in s1 but not in s2
print(s1 - s2)
```

```
{1, 2}
{1, 2}
```

```
print(s1.symmetric_difference(s2)) #set of elements in both s1 & s2, but not the common elemen
print(s1^s2)
```

```
{1, 2, 6, 7}
{1, 2, 6, 7}
```

```
print({1,2,4}.issubset(s1))
```

```
True
```

```
print(s1.isdisjoint(s2))
```

```
False
```

```
fset = frozenset({1,2,3,4,5,6,7,8})
print(fset)
```

```
frozenset({1, 2, 3, 4, 5, 6, 7, 8})
```

```
fset.add(10)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-79-efe063c795db> in <module>()
----> 1 fset.add(10)

AttributeError: 'frozenset' object has no attribute 'add'
```

SEARCH STACK OVERFLOW

Dictionary

Main functionalities of Dictionary

- `d.get(key)` or `d[key]` -> get the **value** in the **key** . If not present, `[]` notation will throw error but `.get()` notation will return `None`
- `d[newkey]=value` or `d[key]=newvalue` -> to update or add new key in the dictionary
- `del d[key]` or `d.pop(key)` -> remove a key
- `d.popitem()` -> randomnly removes a key value pair
- `d.clear()` -> removes all elements in the dictionary
- `d.items()` -> return all pairs of items
- `d.keys()` -> return all keys
- `d.values()` -> return all values

```
d = {}
print(d)
d = {"a":1, "b":2}
print(d)
d = {1:"asdf", 2:["a", "b", "c"]}
print(d)
d = dict([(1, "asdf"), (2, ["a", "b"])])
print(d)
```

```
{}
{'a': 1, 'b': 2}
{1: 'asdf', 2: ['a', 'b', 'c']}
{1: 'asdf', 2: ['a', 'b']}
```

```
d = {"name":"raghu ram", "age":28, "address":"43 Pnthouse apts", "city":"Chandigarh"}
print(d)

print(d.get("name"))
print(d["name"])
```

```
{'name': 'raghu ram', 'age': 28, 'address': '43 Pnthouse apts', 'city': 'Chandigarh'}
raghu ram
raghu ram
```

```
print(d.get("state"))
print(d["state"])
```

```
None
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-83-b89ab0b7c457> in <module>()
      1 print(d.get("state"))
----> 2 print(d["state"])

KeyError: 'state'
```

SEARCH STACK OVERFLOW

```
d["name"] = "Raghu Ram T"
```

```
print(d)
```

```
{'name': 'Raghu Ram T', 'age': 28, 'address': '43 Pnthouse apts', 'city': 'Chandigarh'}
```

```
d["city"] = "Zirakpur"
d["state"] = "Chandigarh"
d["zipcode"] = 140604
print(d)
```

```
{'name': 'Raghu Ram T', 'age': 28, 'address': '43 Pnthouse apts', 'city': 'Zirakpur', 'st
```

```
d.pop("address")
print(d)
del d['state']
print(d)
```

```
{'name': 'Raghu Ram T', 'age': 28, 'city': 'Zirakpur', 'state': 'Chandigarh', 'zipcode':
{'name': 'Raghu Ram T', 'age': 28, 'city': 'Zirakpur', 'zipcode': 140604}
```

```
d.popitem() #random key removal
print(d)
```

```
{'name': 'Raghu Ram T', 'age': 28, 'city': 'Zirakpur'}
```

```
d.clear()
print(d)
```

```
{}
```

```
d = {'name': 'Raghu Ram T', 'age': 28, 'address': '43 Pnthouse apts', 'city': 'Zirakpur', 'sta
newd = d.copy()
newd['name'] = 'Vivek Raja T'
newd['age'] = 25
print(d)
print(newd)
```

```
{'name': 'Raghu Ram T', 'age': 28, 'address': '43 Pnthouse apts', 'city': 'Zirakpur', 'st
{'name': 'Vivek Raja T', 'age': 25, 'address': '43 Pnthouse apts', 'city': 'Zirakpur', 's
```

```
items = d.items()
keys = d.keys()
values = d.values()
print(items)
print(keys)
print(values)
```

```
dict_items([('name', 'Raghu Ram T'), ('age', 28), ('address', '43 Pnthouse apts'), ('city
dict_keys(['name', 'age', 'address', 'city', 'state', 'zipcode'])
dict_values(['Raghu Ram T', 28, '43 Pnthouse apts', 'Zirakpur', 'Chandigarh', 140604])
```

```
for item in items:
  print("Item is {} with type {}".format(item, type(item)))
```

```
Item is ('name', 'Raghu Ram T') with type <class 'tuple'>
Item is ('age', 28) with type <class 'tuple'>
Item is ('address', '43 Pnthouse apts') with type <class 'tuple'>
Item is ('city', 'Zirakpur') with type <class 'tuple'>
Item is ('state', 'Chandigarh') with type <class 'tuple'>
Item is ('zipcode', 140604) with type <class 'tuple'>
```

```
#Uppercasing all using comprehension
con = [(k.upper(), v.upper() if type(v) == str else v) for (k,v) in d.items()]
print(con)
print(type(con))
print(con[1])
print(type(con[1]))
```

```
[('NAME', 'RAGHU RAM T'), ('AGE', 28), ('ADDRESS', '43 PNTHOUSE APTS'), ('CITY', 'ZIRAKPU
<class 'list'>
('AGE', 28)
<class 'tuple'>
```

```
con = {(k.upper(), v.upper() if type(v) == str else v) for (k,v) in d.items()}
print(con)
print(type(con))
print(('CITY', 'ZIRAKPUR') in con)
```

```
{('CITY', 'ZIRAKPUR'), ('STATE', 'CHANDIGARH'), ('NAME', 'RAGHU RAM T'), ('ADDRESS', '43
<class 'set'>
True
```

```
con = {k.upper(): v.upper() if type(v) == str else v for (k,v) in d.items()}
print(con)
print(type(con))
print(con['ADDRESS'])
print(type(con['ADDRESS']))
```

```
{'NAME': 'RAGHU RAM T', 'AGE': 28, 'ADDRESS': '43 PNTHOUSE APTS', 'CITY': 'ZIRAKPUR', 'ST
<class 'dict'>
43 PNTHOUSE APTS
<class 'str'>
```

String

Some functionalities for string

- `s1 + s2` -> concatenates two strings
- `s.split(char)` -> splits a word into list of words based on the input **char** . If the argument is empty, it'll be split based on **(space)**
- `"char".join[lst]` -> join the list of strings available in **lst** using the **char**
- `s.upper()` -> transform the word into uppercase
- `s.lower()` -> transform the word into lowercase

- `s.replace(a,b)` -> find the string **a** in the input string **s** and replace it with **b**
- `s.find(str)` -> find the index of the first occurrence of **str** in the word **s**
- `"str" in word` -> Boolean : true if **str** is a substring of **word**
- `for w in word:` -> iterating a word using it's characters

```
word = "Testt"
print(word)
word = 'Testt'
print(word)
word = '''multi
            line
            word'''
print(word)
```

```
Testt
Testt
multi
            line
            word
```

```
word = 'Hello-World'
print(word[1])
print(word[2:8])
print(word[5:-3])
```

```
e
llo-Wo
-Wo
```

```
word[1] = 'a'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-129-d7ab97e4ac21> in <module>()
----> 1 word[1] = 'a'

TypeError: 'str' object does not support item assignment
```

SEARCH STACK OVERFLOW

```
del word
```

```
s1 = 'Hello'
s2 = 'World'
print(s1+'-'+s2)
```

```
Hello-World
```

```
count = 0
for c in s1+s2:
  if c == 'o':
    count+=1
```

```
print(count)
```

```
2
```

```
print('o' in s1)
print('or' in s2)
```

```
True
True
```

```
lst = [s1, s2]
print(lst)
```

```
['Hello', 'World']
```

```
newword = " ".join(lst)
print(newword)
print(newword.split())
```

```
Hello World
['Hello', 'World']
```

```
print(s1.upper())
print(s2.lower())
```

```
HELLO
world
```

```
print(newword.find('Wo'))
```

```
6
```

```
print(newword.replace(" ", " beautiful "))
```

```
Hello beautiful World
```

```
print(list(s1) == list(s2))
print(list('word') == list('word'))
```

```
False
True
```

✓ 0s completed at 7:06 PM