Indexing

In [12]:
```
# make a string
a= 'samosa pakora'
a
```

Out[12]:
```
'samosa pakora'
```

In [13]:
```
a
```

Out[13]:
```
'samosa pakora'
```

In [18]:
```
#Length of index
len(a)
```

Out[18]:
```
13
```

In [14]:
```
a[0]
```

Out[14]:
```
's'
```

In [15]:
```
a[1]
```

Out[15]:
```
'a'
```

In [16]:
```
a[3]
```

Out[16]:
```
'o'
```

In [17]:
```
a[7]
```

Out[17]:
```
'p'
```

In [21]:
```
#last index is exclusive
a[0:6]
```

Out[21]:
```
'samosa'
```

In [22]:
```
a[-2]
```

Out[22]:
```
'r'
```

In [23]:
```
a[6:9]
```

Out[23]:
```
' pa'
```

In [24]:
```
food= "biryani"
food
```

Out[24]:
```
'biryani'
```

In [26]:
```
len(food)
```

Out[26]:
```
7
```

In [27]:
```
food[1:5]
```

Out[27]:
```
'irya'
```

string method

In [30]:
```
food.capitalize()
#to capitalize the first letter
```

Out[30]:
```
'Biryani'
```

In [32]:
```
food.upper()
#capitalize every letter
```

Out[32]:
```
'BIRYANI'
```

In [33]:
```
food.lower()
#Lowercase every letter
```

Out[33]:
```
'biryani'
```

In [34]:
```
food.replace('b','Sh')
#replace first letter with second
```

Out[34]:
```
'Shiryani'
```

In [35]:
```
#counting a specific alphabet in a string
name='faryal'
name
```

Out[35]:
```
'faryal'
```

In [36]:
```
name.count('a')
```

Out[36]:
```
2
```

finding an index number in string

In [38]:
```
line="you must be tired, aren't you"
line
```

Out[38]:
```
"you must be tired, aren't you"
```

In [40]:
```
line.find('y')
```

Out[40]:
```
0
```

In [43]:
```
### how to split a string
line="you must be tired, aren't you"
line.split(',')
```

Out[43]:
```
['you must be tired', " aren't you"]
```

BASIC DATA STRUCTURES IN PYTHON

1. TUPLE
2. LIST
3. DICTIONARIES
4. SET

1. TUPLE

- Ordered collection of elements
- Enclosed in ()
- Different kind of elements can be stored
- unchangeable elements

In [45]:
```
tup1 = (2,'hi', False, 8.9)
tup1
```

Out[45]:
```
(2, 'hi', False, 8.9)
```

In [46]:
```
# type of a tuple
type(tup1)
```

Out[46]:
```
tuple
```

-Indexing a Tuple

In [48]:
```
tup1[2]
```

Out[48]:
```
False
```

In [49]:
```
tup1[3]
```

Out[49]:
```
8.9
```

In [51]:
```
tup1[0:4]
```

Out[51]:
```
(2, 'hi', False, 8.9)
```

In [52]:
```
tup2=(3,"hello", True, 7.4)
tup2
```

Out[52]:
```
(3, 'hello', True, 7.4)
```

In [53]:
```
# concatinate(to add two tuples)
tup1+ tup2
```

Out[53]:
```
(2, 'hi', False, 8.9, 3, 'hello', True, 7.4)
```

In [55]:
```
tup1*3+tup2
```

Out[55]:
```
(2,
 'hi',
 False,
 8.9,
 2,
 'hi',
 False,
 8.9,
 2,
 'hi',
 False,
 8.9,
 3,
 'hello',
 True,
 7.4)
```

In [57]:
```
tup3=(2,3,4,5,8)
tup3
```

Out[57]:
```
(2, 3, 4, 5, 8)
```

In [58]:
```
tup4=(1,2,3,4,5,8,9,6)
tup4
```

Out[58]:
```
(1, 2, 3, 4, 5)
```

In [62]:
```
min(tup3)
```

Out[62]:
```
2
```

In [63]:
```
max(tup4)
```

Out[63]:
```
5
```

In [85]:
```
tup4.count(9)
```

Out[85]:
```
0
```

In [84]:
```
tup4.index(5)
```

Out[84]:
```
4
```

---

2. List

- Ordered collection of elements
- Enclosed in [ ]
- unchangeable elements

In [67]:
```
list1= [1,'hellooooooooo', True, 5.8]
list1
```

Out[67]:
```
[1, 'hellooooooooo', True, 5.8]
```

In [69]:
```
type(list1)
```

Out[69]:
```
list
```

In [70]:
```
len (list1)
```

Out[70]:
```
4
```

In [71]:
```
list1[3]
```

Out[71]:
```
5.8
```

In [73]:
```
list2=[3,5,6,'fary', 4.6]
list2
```

Out[73]:
```
[3, 5, 6, 'fary', 4.6]
```

In [74]:
```
list1 + list2
```

Out[74]:
```
[1, 'hellooooooooo', True, 5.8, 3, 5, 6, 'fary', 4.6]
```

In [77]:
```
3*list1
```

Out[77]:
```
[1,
 'hellooooooooo',
 True,
 5.8,
 1,
 'hellooooooooo',
 True,
 5.8,
 1,
 'hellooooooooo',
 True,
 5.8]
```

In [86]:
```
list1
```

Out[86]:
```
[1, 'hellooooooooo', True, 5.8]
```

In [94]:
```
list1.reverse()
list1
```

Out[94]:
```
[5.8, True, 'hellooooooooo', 1]
```

In [108…
```
list1.append('hi biro')  #it will add the term in bracket in the list
list1
```

Out[108…
```
[5.8,
 True,
 'hellooooooooo',
 1,
 'hi biro',
 'hi biro',
 'hi biro',
 'hi biro',
 'hi biro',
 'hi biro',
 'hi biro',
 'hi biro',
 'hi biro',
 'hi biro',
 'hi biro']
```

In [115…
```
list1.clear() #it will clear the list
list1
```

Out[115…
```
[]
```

In [114…
```
list3=[1,2,3,4,5,6]
```

```
list3
```

Out[114…
```
[1, 2, 3, 4, 5, 6]
```

In [126…
```
list3.insert(5,7) #it will tell where to insert an item in the list. first one will sho
list3
```

Out[126…
```
[1, 2, 3, 4, 5, 7, 7, 7, 7, 6]
```

In [128…
```
list3.copy()
list3
```

Out[128…
```
[1, 2, 3, 4, 5, 7, 7, 7, 7, 6]
```

In [130…
```
list4=[0,9,8,7,6]
list4
```

Out[130…
```
[0, 9, 8, 7, 6]
```

In [133…
```
list4.extend('hi kia haal hai') #it willl split the STRING in to each character
list4
```

Out[133…
```
[0,
 9,
 8,
 7,
 6,
 't',
 'w',
 'o',
 'h',
 'i',
 ' ',
 'k',
 'i',
 'a',
 ' ',
 'h',
 'a',
 'a',
 'l',
 ' ',
 'h',
 'a',
 'i']
```

In [138…
```
list4.index(7)
list4
```

Out[138…
```
[0,
 9,
 8,
 7,
```

```
6,
't',
'w',
'o',
'h',
'i',
' ',
'k',
'i',
'a',
' ',
'h',
'a',
'a',
'l',
' ',
'h',
'a',
'i']
```

In [282…
```
list4.pop() #will show the term 1 by 1
```

Out[282…
```
' '
```

In [147…
```
list5=[1,0,3,4,5,5,6,7,8]
list5
```

Out[147…
```
[1, 0, 3, 4, 5, 5, 6, 7, 8]
```

In [148…
```
list5.sort()
list5
```

Out[148…
```
[0, 1, 3, 4, 5, 5, 6, 7, 8]
```

In [149…
```
list6=['r', 'g', 'p', 'a']
list6
```

Out[149…
```
['r', 'g', 'p', 'a']
```

In [150…
```
list6.sort()
list6
```

Out[150…
```
['a', 'g', 'p', 'r']
```

In [273…
```
list7=[2,2,2,2,2,4,5,6,7,8,9]
list7
```

Out[273…
```
[2, 2, 2, 2, 2, 4, 5, 6, 7, 8, 9]
```

In [274…
```
list7.count(2) #it will tell how many times 2 appered in the list
```

Out[274… 5

In [ ]:

3-Dictionaries

- disordered collection of elements
- key and value
- curly brackets { }
- mutable/changeable values

In [155…
```
#food and their prices
d1 = {"pakora": 30, "salad" :50, "samosa":15}
#samosa is KEY and 30 is VALUE
d1
```

Out[155… {'pakora': 30, 'salad': 50, 'samosa': 15}

In [157…
```
type(d1)
```

Out[157… dict

In [164…
```
#extract data
keys= d1.keys()
keys
```

Out[164… dict_keys(['pakora', 'salad', 'samosa'])

In [165…
```
values= d1.values()
values
```

Out[165… dict_values([30, 50, 15])

In [166…
```
#we can add new element
d1['tikki']=10
d1
```

Out[166… {'pakora': 30, 'salad': 50, 'samosa': 15, 'tikki': 10}

In [167…
```
#update the values
d1['tikki']=15
d1
```

Out[167… {'pakora': 30, 'salad': 50, 'samosa': 15, 'tikki': 15}

In [168…
```
d2={'dates':50 , 'chocolates':200 , "macroni" :400}
```

```
d2
```

Out[168…
```
{'dates': 50, 'chocolates': 200, 'macroni': 400}
```

In [172…
```
#concatinate
d1.update(d2)
d1
```

Out[172…
```
{'pakora': 30,
 'salad': 50,
 'samosa': 15,
 'tikki': 15,
 'dates': 50,
 'chocolates': 200,
 'macroni': 400}
```

4- Sets

- Disordered and Unindexed
- {}
- no duplicates allowed

In [186…
```
s1 = {3,6.7, 8, "hi", "faryal", 'hello', False}
s1
```

Out[186…
```
{3, 6.7, 8, False, 'faryal', 'hello', 'hi'}
```

In [187…
```
s1.add('jjjj')
s1
```

Out[187…
```
{3, 6.7, 8, False, 'faryal', 'hello', 'hi', 'jjjj'}
```

In [188…
```
s1.remove('jjjj')
s1
```

Out[188…
```
{3, 6.7, 8, False, 'faryal', 'hello', 'hi'}
```

In [190…
```
s2={3,6.7,9}
s2
```

Out[190…
```
{3, 6.7, 9}
```

In [200…
```
s2.difference(s1)
#items of the first set not present in the second
```

Out[200…
```
{9}
```

In [197…
```
s3={1,2,3,4,5}
```

```
s3
```

Out[197…
```
{1, 2, 3, 4, 5}
```

In [198…
```
s4={4,5,6,7,8}
s4
```

Out[198…
```
{4, 5, 6, 7, 8}
```

In [204…
```
s4.difference(s3)
```

Out[204…
```
{4, 5, 6, 7, 8}
```

In [206…
```
s4.difference_update(s3)
s4
```

Out[206…
```
{4, 5, 6, 7, 8}
```

In [209…
```
s5={5,8,6,1,2}
s5
```

Out[209…
```
{1, 2, 5, 6, 8}
```

In [210…
```
s6={1,2,9,6,4,7,3}
s6
```

Out[210…
```
{1, 2, 3, 4, 6, 7, 9}
```

In [211…
```
s5.intersection(s6)
```

Out[211…
```
{1, 2, 6}
```

In [216…
```
s5.isdisjoint(s2)
```

Out[216…
```
True
```

In [217…
```
s5.issubset(s6)
```

Out[217…
```
False
```

In [218…
```
s7 = {1,2,3}
s7
```

Out[218…
```
{1, 2, 3}
```

In [219…
```
s8 = {1,2,3,4,5}
```

```
s8
```

Out[219…
```
{1, 2, 3, 4, 5}
```

In [220…
```
s7.issubset(s8)
```

Out[220…
```
True
```

In [221…
```
s8.issuperset(s7)
```

Out[221…
```
True
```

In [239…
```
s8.pop() #1 by 1 the items will get pop up
```

Out[239…
```
5
```

In [245…
```
s9={1,2,3}
s9
```

Out[245…
```
{1, 2, 3}
```

In [246…
```
s10={4,5,6}
s10
```

Out[246…
```
{4, 5, 6}
```

In [248…
```
s9.update(s10) #merge two sets
s9
```

Out[248…
```
{1, 2, 3, 4, 5, 6}
```

In [262…
```
sa={1,2,3,4,5}
sa
```

Out[262…
```
{1, 2, 3, 4, 5}
```

In [263…
```
sb={4,5,6,7,8}
sb
```

Out[263…
```
{4, 5, 6, 7, 8}
```

In [264…
```
sa.symmetric_difference(sb)   #all items except common items
```

Out[264…
```
{1, 2, 3, 6, 7, 8}
```

In [266…
```
sc={1,2,6,0,8}
```

```
sc
```

Out[266…
```
{0, 1, 2, 6, 8}
```

In [267…
```
sd={14,6,8,5,3}
sd
```

Out[267…
```
{3, 5, 6, 8, 14}
```

In [268…
```
sc.symmetric_difference_update(sd)   #sc will get updated with the difference but sd wi
sc
```

Out[268…
```
{0, 1, 2, 3, 5, 14}
```

In [269…
```
sd
```

Out[269…
```
{3, 5, 6, 8, 14}
```

In [270…
```
sd.union(sc)
```

Out[270…
```
{0, 1, 2, 3, 5, 6, 8, 14}
```

In [ ]:
```

```