1-Indexing

In [2]:
```
# Make a string
a = 'somosa pakora'
a
```

Out[2]: 'somosa pakora'

In [3]:
```
a[0]
```

Out[3]: 's'

In [4]:
```
a[1]
```

Out[4]: 'o'

In [5]:
```
a[6]
```

Out[5]: ' '

In [6]:
```
a[3:4]
```

Out[6]: 'o'

In [7]:
```
a[0:5]
```

Out[7]: 'somos'

In [8]:
```
a[0:6]
```

Out[8]: 'somosa'

In [9]:
```
# lenth of Indexes
len(a)
```

Out[9]: 13

In [10]:
```
a[0:13]
```

Out[10]: 'somosa pakora'

In [11]:
```
a[12]
```

Out[11]: 'a'

In [12]:
```
a[-1]
```

Out[12]: 'a'

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

Out[13]: 'somosa pakora'

In [14]:
```
a[-6:-1]
```

Out[14]: 'pakor'

In [15]:
```
a[-6:0]
```

Out[15]: ''

In [16]:
```
a[-6:13]
```

Out[16]: 'pakora'

In [17]:
```
food="biryani"
food
```

Out[17]: 'biryani'

Strings Methods

In [19]:
```
food
```

Out[19]: 'biryani'

In [20]:
```
len(food)
```

Out[20]: 7

In [21]:
```
# For Capitilized
food.capitalize()
```

Out[21]: 'Biryani'

In [22]:
```
# upper Case
food.upper()
```

Out[22]: 'BIRYANI'

In [23]:
```
# Lower case
food.lower()
```

Out[23]: 'biryani'

In [24]:
```
food.replace("b", "sh")
```

Out[24]: 'shiryani'

In [25]:
```
food.replace("i"," ")
```

Out[25]:
```
'b ryan '
```

In [26]:
```
food.replace("a" ,"e" )
```

Out[26]:
```
'biryeni'
```

In [27]:
```
# Counting a specific alphabet in a strings
name="baba Ammar with Ammar tufail"
name
```

Out[27]:
```
'baba Ammar with Ammar tufail'
```

In [28]:
```
name.count("a")
```

Out[28]:
```
5
```

In [29]:
```
name.count("A")
```

Out[29]:
```
2
```

In [30]:
```
name.count("mm")
```

Out[30]:
```
2
```

In [31]:
```
name.count("m")
```

Out[31]:
```
4
```

finding an index in strings

In [33]:
```
name="baba Ammar with Ammar tufail"
name
```

Out[33]:
```
'baba Ammar with Ammar tufail'
```

In [34]:
```
name.find("b")
```

Out[34]:
```
0
```

In [35]:
```
name.find("aa")
```

Out[35]:
```
-1
```

In [36]:
```
name.find("f")
```

Out[36]:
```
24
```

In [37]:
```
name.find("w")
```

Out[37]: 11

In [38]:
```
name.find(" ")
```

Out[38]: 4

In [39]:
```
### how to split a strings
food= 'I love somosa,biryani,pakora and raita'
food
```

Out[39]: 'I love somosa,biryani,pakora and raita'

In [40]:
```
food.split(",")
```

Out[40]: ['I love somosa', 'biryani', 'pakora and raita']

Basic Data structures

1- Tuple

2- List

3- Dictioneries

4- Sets

1-Tuple

- orderd Collections of elements
- enclosese in ( ) brackets
- Different kinds of elements can be stored
- Unmutable (Don't be change)

In [61]:
```
tup=(12,'python',True,2.6)
tup
```

Out[61]: (12, 'python', True, 2.6)

In [44]:
```
print(type(tup))
```

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

Indexing in tuple

In [46]:
```
tup=[1]
tup
```

Out[46]: [1]

In [47]:
```
len("tup")
```

Out[47]: 3

In [48]:
```
tup=(12,'python',True,2.6)
tup
```

Out[48]: (12, 'python', True, 2.6)

In [49]:
```
print(len(tup))
```

4

In [50]:
```
tup[0:3]
```

Out[50]: (12, 'python', True)

In [51]:
```
tup1 = (2,'babaAmmar',3.4,False)
tup1
```

Out[51]: (2, 'babaAmmar', 3.4, False)

In [52]:
```
# Concatinate ( add two tuples or more than two)
tup + tup1
```

Out[52]: (12, 'python', True, 2.6, 2, 'babaAmmar', 3.4, False)

In [53]:
```
# Concatinate + Repeat
tup*2 + tup1
```

Out[53]: (12, 'python', True, 2.6, 12, 'python', True, 2.6, 2, 'babaAmmar', 3.4, False)

In [54]:
```
tup3 = (12,33,44,55,66,11,34)
tup3
```

Out[54]: (12, 33, 44, 55, 66, 11, 34)

In [55]:
```
# Minimum Value
min(tup3)
```

Out[55]: 11

In [56]:
```
# Maximum Value
max(tup3)
```

Out[56]: 66

In [57]:
```
# Repeat
tup3*2
```

Out[57]: (12, 33, 44, 55, 66, 11, 34, 12, 33, 44, 55, 66, 11, 34)

In [67]:
```
tup.index("python")
```

Out[67]: 1

2- list

- Orderd Collections of elements
- encloses in [ ] square Brackets
- Mutable ,you can change any elements

In [60]:
```
list1 = [2,'babaAmmar',False]
list1
```

Out[60]: [2, 'babaAmmar', False]

In [61]:
```
type(list1)
```

Out[61]: list

In [62]:
```
len(list1)
```

Out[62]: 3

In [63]:
```
list1[2]
```

Out[63]: False

In [64]:
```
list2=[3,5,"Mujahid",False,450,55.5]
list2
```

Out[64]: [3, 5, 'Mujahid', False, 450, 55.5]

In [65]:
```
# Concatinate
list1 + list2
```

Out[65]: [2, 'babaAmmar', False, 3, 5, 'Mujahid', False, 450, 55.5]

In [66]:
```
list1 * 2
```

Out[66]: [2, 'babaAmmar', False, 2, 'babaAmmar', False]

In [67]:
```
list1.reverse()
list1
```

Out[67]: [False, 'babaAmmar', 2]

In [68]:
```
# For adding
list1.append("Learning python")
list1
```

Out[68]: [False, 'babaAmmar', 2, 'Learning python']

In [69]:
```
list1.clear()
list1
```

Out[69]: []

In [70]:
```
#
list2.copy()
```

Out[70]: [3, 5, 'Mujahid', False, 450, 55.5]

In [71]:
```
list2.append("1")
list2
```

Out[71]: [3, 5, 'Mujahid', False, 450, 55.5, '1']

In [72]:
```
count_1=list2.count(450)
```

In [ ]:
```

```

In [73]:
```
print(count_1)
```

```
1
```

In [74]:
```
count_1= list2.count(1)
```

In [75]:
```
count_1
```

Out[75]: 0

In [76]:
```
list2
```

Out[76]: [3, 5, 'Mujahid', False, 450, 55.5, '1']

In [77]:
```
extend_func=list2.extend("func")
print(extend_func)
```

```
None
```

In [78]:
```
# pop is used for delt and also write which is delt
list2.pop(1)
```

Out[78]: 5

In [79]:
```
list2
```

Out[79]: [3, 'Mujahid', False, 450, 55.5, '1', 'f', 'u', 'n', 'c']

In [80]:
```
list1
```

Out[80]:
```
[]
```

In [81]:
```
list2.extend(list1)
```

In [82]:
```
list2
```

Out[82]:
```
[3, 'Mujahid', False, 450, 55.5, '1', 'f', 'u', 'n', 'c']
```

In [83]:
```
list2.append(3)
```

In [84]:
```
list2
```

Out[84]:
```
[3, 'Mujahid', False, 450, 55.5, '1', 'f', 'u', 'n', 'c', 3]
```

In [85]:
```
list2.count(3)
```

Out[85]:
```
2
```

In [86]:
```
coun_s= list2.count(3)
```

In [87]:
```
coun_s
```

Out[87]:
```
2
```

In [88]:
```
#Count is used for  any number how many times repeat
coun_s= list2.count('Mujahid')
```

In [89]:
```
coun_s
```

Out[89]:
```
1
```

In [90]:
```
list2
```

Out[90]:
```
[3, 'Mujahid', False, 450, 55.5, '1', 'f', 'u', 'n', 'c', 3]
```

In [91]:
```
list1
```

Out[91]:
```
[]
```

In [92]:
```
list2[5]
```

Out[92]:
```
'1'
```

In [93]:
```
list2[3]
```

Out[93]:
```
450
```

In [94]:
```
# index is used for print provided element's indeces number
list2.index(450)
```

Out[94]: 3

In [95]:
```
#list2.index(2)
```

In [96]:
```
# Insert Method is used to insert an element in a particular position
list2.insert(6,'1')
```

In [97]:
```
list2
```

Out[97]: [3, 'Mujahid', False, 450, 55.5, '1', '1', 'f', 'u', 'n', 'c', 3]

In [98]:
```
list2.insert(6,1)
```

In [99]:
```
list2
```

Out[99]: [3, 'Mujahid', False, 450, 55.5, '1', 1, '1', 'f', 'u', 'n', 'c', 3]

In [103…
```
list2.remove(False)
```

In [105…
```
list2
```

Out[105… [3, 'Mujahid', 450, 55.5, '1', 1, '1', 'f', 'u', 'n', 'c', 3]

In [107…
```
# pop is used for remove provided indeces value and also print that value(element)
list2.pop(1)
```

Out[107… 'Mujahid'

In [109…
```
 list2
```

Out[109… [3, 450, 55.5, '1', 1, '1', 'f', 'u', 'n', 'c', 3]

In [111…
```
list3= [9,3,5,3,6,1,2]
```

In [113…
```
list3.sort()
list3
```

Out[113… [1, 2, 3, 3, 5, 6, 9]

3-Dictioneries

- Unordered collection of elements
- encloses in Curly Brackets { }
- Key and Value Pairs
- Mutatable / change the elements

In [148…
```
food1 = {"somosa":30,"pakora":100,"Raita":50,"chicken role":90}
food1
```

Out[148… {'somosa': 30, 'pakora': 100, 'Raita': 50, 'chicken role': 90}

In [127…
```
# keys
keys=food1.keys()
```

In [129…
```
keys
```

Out[129… dict_keys(['somosa', 'pakora', 'Raita', 'chicken role'])

In [131…
```
values1=food1.values()
```

In [133…
```
values1
```

Out[133… dict_values([30, 100, 50, 90])

In [135…
```
# For adding new elements
food1["Tikki"]=50
```

In [137…
```
food1
```

Out[137… {'somosa': 30, 'pakora': 100, 'Raita': 50, 'chicken role': 90, 'Tikki': 50}

In [141…
```
# updating values
food1["Tikki"]=55
```

In [143…
```
food1
```

Out[143… {'somosa': 30, 'pakora': 100, 'Raita': 50, 'chicken role': 90, 'Tikki': 55}

In [145…
```
food1
```

Out[145… {'somosa': 30, 'pakora': 100, 'Raita': 50, 'chicken role': 90, 'Tikki': 55}

In [147…
```
food2 = {"Bubblies":200,"Juice":900,"Sawiyan":1000}
food2
```

Out[147… {'Bubblies': 200, 'Juice': 900, 'Sawiyan': 1000}

In [149…
```
# Concatinate
food1.update(food2)
food1
```

Out[149…
```
{'somosa': 30,
 'pakora': 100,
 'Raita': 50,
 'chicken role': 90,
 'Tikki': 55,
 'Bubblies': 200,
 'Juice': 900,
 'Sawiyan': 1000}
```

In [78]:
```
food1
```

Out[78]: {}

In [76]:
```
food1.clear()
food1
```

Out[76]: {}

In [82]:
```
food1.copy()
```

Out[82]: {'somosa': 30, 'pakora': 100, 'Raita': 50, 'chicken role': 90}

In [84]:
```
lists=[1,2,3,4,5]
lists
```

Out[84]: [1, 2, 3, 4, 5]

In [86]:
```
## fromkeys is to change from lists to DICTIONERY
dict.fromkeys(lists)
```

Out[86]: {1: None, 2: None, 3: None, 4: None, 5: None}

In [90]:
```
{}.fromkeys(range(1,7),10)
```

Out[90]: {1: 10, 2: 10, 3: 10, 4: 10, 5: 10, 6: 10}

In [96]:
```
Employee = {"Sallary":50000,"Name":"raj","age":20}
Employee
```

Out[96]: {'Sallary': 50000, 'Name': 'raj', 'age': 20}

In [104…
```
# get is used for key value without error
Employee.get("Sallary")
```

Out[104… 50000

In [114…
```
food1
```

Out[114… {'somosa': 30, 'pakora': 100, 'Raita': 50, 'chicken role': 90}

In [116…
```
## items is used to convert dictionery to list
get_food=food1.items()
```

In [118…
```
get_food
```

Out[118… dict_items([('somosa', 30), ('pakora', 100), ('Raita', 50), ('chicken role', 90)])

In [120…
```
list(get_food)
```

Out[120… [('somosa', 30), ('pakora', 100), ('Raita', 50), ('chicken role', 90)]

In [126…
```
as_list=list(get_food)
```

In [128…
```
as_list
```
Out[128…
```
[('somosa', 30), ('pakora', 100), ('Raita', 50), ('chicken role', 90)]
```

In [130…
```
as_list[0]
```
Out[130…
```
('somosa', 30)
```

In [134…
```
food1
```
Out[134…
```
{'somosa': 30, 'pakora': 100, 'Raita': 50, 'chicken role': 90}
```

In [140…
```
food1.pop('somosa')
```
Out[140…
```
30
```

In [152…
```
food1
```
Out[152…
```
{'somosa': 30, 'pakora': 100, 'Raita': 50, 'chicken role': 90}
```

In [156…
```
food1.values()
```
Out[156…
```
dict_values([30, 100, 50, 90])
```

4-Sets

- unmutable and unindexes
- enclose in { } curly Brackets
- Unordered and don't accept Dupplicates
-

In [19]:
```
s1 = {1,23,44,"Ammar","codanics","Faisalabad"}
s1
```
Out[19]:
```
{1, 23, 44, 'Ammar', 'Faisalabad', 'codanics'}
```

In [156…
```
s1.add('Ammar')
s1
```
Out[156…
```
{1, 23, 44, 'Ammar', 'Faisalabad', 'codanics'}
```

In [158…
```
s1.add("Ammar1")
s1
```
Out[158…
```
{1, 23, 44, 'Ammar', 'Ammar1', 'Faisalabad', 'codanics'}
```

In [160…
```
s1.clear()
```

In [162…
```
s1
```

Out[162…
```
set()
```

In [188…
```
s1 ={1, 23, 44, 'Ammar', 'Ammar1', 'Faisalabad', 'codanics'}
s1
```

Out[188…
```
{1, 23, 44, 'Ammar', 'Ammar1', 'Faisalabad', 'codanics'}
```

In [166…
```
s1.copy()
```

Out[166…
```
{1, 23, 44, 'Ammar', 'Ammar1', 'Faisalabad', 'codanics'}
```

In [21]:
```
s2= {"lahore",4,5,44,1,"codanics","Faisalabad"}
s2
```

Out[21]:
```
{1, 4, 44, 5, 'Faisalabad', 'codanics', 'lahore'}
```

In [172…
```
#s1.difference(s2)
```

Out[172…
```
{23, 'Ammar', 'Ammar1'}
```

In [180…
```
s2.difference(s1)
```

Out[180…
```
{4, 5, 'lahore'}
```

In [190…
```
s1
```

Out[190…
```
{1, 23, 44, 'Ammar', 'Ammar1', 'Faisalabad', 'codanics'}
```

In [184…
```
s1.difference_update(s2)
```

In [186…
```
s1
```

Out[186…
```
{23, 'Ammar', 'Ammar1'}
```

In [194…
```
# used for remove elements and  show error in case of non existing value
s1.remove("Ammar")
s1
```

Out[194…
```
{1, 23, 44, 'Ammar1', 'Faisalabad', 'codanics'}
```

In [7]:
```
s1.remove("Ammar")
s1
```

Out[7]:
```
{1, 23, 44, 'Faisalabad', 'codanics'}
```

In [9]:
```
# used for remove elements and dont show error in case of non existing value

s1.discard("Ammar2")
s1
```

Out[9]:
```
{1, 23, 44, 'Faisalabad', 'codanics'}
```

In [15]:
```
s2
```

Out[15]:
```
{1, 4, 44, 5, 'Faisalabad', 'codanics', 'lahore'}
```

In [17]:
```
s1.intersection(s2)
```

Out[17]:
```
{1, 44, 'Faisalabad', 'codanics'}
```

In [23]:
```
s1.union(s2)
```

Out[23]:
```
{1, 23, 4, 44, 5, 'Ammar', 'Faisalabad', 'codanics', 'lahore'}
```

In [25]:
```
# disjoint method is used to see if there is no relationship between two sets
# allways return Boolian (True or False)
s1.isdisjoint(s2)
```

Out[25]:
```
False
```

In [27]:
```
# use of subsets or supersets
## allways return Boolian (True or False)
s1={1,2,3,4,5}
s2={2,3}
```

In [29]:
```
s1.issubset(s2)
```

Out[29]:
```
False
```

In [31]:
```
s2.issubset(s1)
```

Out[31]:
```
True
```

In [33]:
```
s1.issuperset(s2)
```

Out[33]:
```
True
```

In [35]:
```
s1
```

Out[35]:
```
{1, 2, 3, 4, 5}
```

In [45]:
```
# REMOVE FIEST ELENENT
pop()
s1
```

Out[45]:
```
{3, 4, 5}
```

In [51]:
```
s1= {1,2,3,4}
s2= {5,6,7,8}
print(s1)
print(s2)
```

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

In [53]:
```
## summetric difference is used to keeps the elements that are no present in both s
s1.symmetric_difference(s2)
```

Out[53]:
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [55]:
```
s1
```

Out[55]:
```
{1, 2, 3, 4}
```

In [57]:
```
# Update is used for adding multiple values
s1.update([9,10,12])
```

In [59]:
```
s1
```

Out[59]:
```
{1, 2, 3, 4, 9, 10, 12}
```

In [144…
```
s1
```

Out[144…
```
{1, 2, 3, 4, 9, 10, 12}
```