Data Structure

List,

Tuple,

Set,

Dictionary

List

Lists are used tostoremultiple items in single variable.

Properties of List

List store element in the sequential order.

List stores hetrogeneous element.

List allow duplicate value.

List are mutable or changeable

__Example__

In [1]:
```
num=[10,20,30,40,50]  #homogeneous kind of data
```

In [2]:
```
type(num)
```

Out[2]:
```
list
```

In [3]:
```
num1=[10,20.2,'happy']    #hetrogeneous kind of data
```

In [4]:
```
type(num1)
```

Out[4]:
```
list
```

In [5]:
```
num2=[[1,2,3],[4,5,6]]     #nested list
```

In [6]:
```
type(num2)
```

Out[6]:
```
list
```

Memory id of list

In [1]:
```
num3=[1,2,3,4,5]
id(num3)
```

Out[1]:
```
1385772774144
```

Replace the element

In [2]:
```
num3[1]=6
```

In [3]:
```
num3
```

Out[3]:
```
[1, 6, 3, 4, 5]
```

In [6]:
```
id(num3)    #after replacing the element also its having same memory id
```

Out[6]:
```
1385772774144
```

Reverse / Negative Indexing

In [11]:
```
num4=[10,20,30,40,50,60,70,80,90,100]
```

In [12]:
```
num4[-5]
```

Out[12]:
```
60
```

In [13]:
```
num4[-8]
```

Out[13]:
```
30
```

In [14]:
```
num4[-8:-1]
```

Out[14]:
```
[30, 40, 50, 60, 70, 80, 90]
```

In [84]:
```
pets=['cat','dog','bird','fish','rabbit']
```

In [85]:
```
pets[-4]
```

Out[85]:
```
'dog'
```

In [15]:
```
num5=[[1,2,3],[4,5,6]]
```

In [16]:
```
num5[0][0]
```

Out[16]:
```
1
```

In [17]:
```
num5[0][2]
```

Out[17]:
```
3
```

In [18]:
```
num5[1][2]
```

Out[18]:
```
6
```

*_Tab Functions_*

In [19]:
```
num6=[1,2,3,4,5,6,7,8,9,10]
```

append

In [21]:
```
num6.append(11)      #it will add 11 as a last element
```

In [22]:
```
num6
```

Out[22]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 11]
```

pop

In [23]:
```
num6.pop()           #it will remove the last element,if need specifically,should mention index number
```

Out[23]:
```
11
```

In [24]:
```
num6
```

Out[24]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

Reverse it will change last to first

In [25]:
```
num6.reverse()
```

In [26]:
```
num6
```

Out[26]:
```
[11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Sort

In [31]:
```
num6.sort()        #sort the list ascending by default
```

In [32]:
```
num6
```

Out[32]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]
```

Remove

In [38]:
```
num6.remove(10)         #remove the specific element in list
```

In [46]:
```
num6
```

Out[46]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 10, 10, 9]
```

In [47]:
```
num6.remove(10)
```

In [48]:
```
num6
```

Out[48]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 10, 9]
```

In [49]:
```
num6.remove(10)
```

In [50]:
```
num6
```

Out[50]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [51]:
```
num6.insert(9,10)         #will insert the element in the list, 9-position, 10-element
```

In [52]:
```
num6
```

Out[52]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Index

In [61]:
```
num6.index(8)          #index position of particular element
```

Out[61]:
```
7
```

In [67]:
```
num7=[11,12,13,14,15]
```

Extend

In [70]:
```
num6.extend(num7)     #adds the specific list to the end of the current list.
```

In [71]:
```
num6
```

Out[71]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 11, 12, 13, 14, 15]
```

Remove

In [73]:
```
num6.remove(11)
```

In [74]:
```
num6
```

Out[74]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 11, 12, 13, 14, 15]
```

Count

In [77]:
```
num6.count(12)     #count the element
```

Out[77]:
```
2
```

Copy

In [78]:
```
num6.copy()
```

Out[78]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 11, 12, 13, 14, 15]
```

In [79]:
```
num6
```

Out[79]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 13, 14, 15, 11, 12, 13, 14, 15]
```

Clear

In [82]:
```
num6.clear()       #delete entire element in the list
```

In [83]:
```
num6
```

Out[83]:
```
[]
```

Tuple

Properties of tuple:

Tuple store element in sequential order.
Tuple allow duplicate values.
Tuple is not mutable.
Tuple run under()

In [86]:
```
tup1=(1,2,3,4,5,6,7,8)
```

In [87]:
```
type(tup1)
```

Out[87]:
```
tuple
```

In [89]:
```
tup2=2,4,6,8
```

In [90]:
```
type(tup2)
```

Out[90]:
```
tuple
```

In [91]:
```
tup3=((1,2),(2,3))
```

In [92]:
```
type(tup3)
```

Out[92]:
```
tuple
```

Tab Function

In [96]:
```
tup1
```

Out[96]:
```
(1, 2, 3, 4, 5, 6, 7, 8)
```

Count

In [98]:
```
tup1.count(2)       #count of particular element
```

Out[98]:
```
1
```

Index

In [100]:
```
tup1.index(6)      #index num of particular element
```

Out[100]:
```
5
```

set

properties of set

Set stores element in random order
set don't aloow duplicate values
set is not mutable
Indexing and slicing is not possible in set
it runs undeer {}

In [101]:
```
set1={2,4,6,8,10}
```

In [104]:
```
len(set1)
```

Out[104]:
```
5
```

In [105]:
```
type(set1)
```

Out[105]:
```
set
```

In [106]:
```
set2={(1,2),4.5,"hi"}
```

In [107]:
```
type(set2)
```

Out[107]:
```
set
```

Tab Function

In [108]:
```
set1
```

Out[108]:
```
{2, 4, 6, 8, 10}
```

Update

In [113]:
```
set1.update({12})    #it will add the element in the end
```

In [114]:
```
set1
```

Out[114]:
```
{2, 4, 6, 8, 10, 12}
```

In [115]:
```
set1.update({'set'})
```

In [116]:
```
set1
```

Out[116]:
```
{10, 12, 2, 4, 6, 8, 'set'}
```

In [119]:
```
set1.update({"sett"})
```

In [120]:
```
set1
```

Out[120]:
```
{10, 12, 2, 4, 6, 8, 'set', 'sett'}
```

In [121]:
```
set1.pop()       #it will delete random element
```

Out[121]:
```
2
```

In [122]:
```
set1
```

Out[122]:
```
{10, 12, 4, 6, 8, 'set', 'sett'}
```

Pop

In [124]:
```
set1.pop()
```

Out[124]:
```
4
```

In [125]:
```
set1
```

Out[125]:
```
{10, 12, 6, 8, 'set', 'sett'}
```

In [126]:
```
set1.remove('sett')       #remove specific element
```

In [127]:
```
set1
```

Out[127]:
```
{10, 12, 6, 8, 'set'}
```

Discard

In [128]:
```
set1.discard('set')      #remove the specific element
```

In [129]:
```
set1
```

Out[129]:
```
{6, 8, 10, 12}
```

In [130]:
```
set1.discard('sett')      #will not raise an error if specified element doen't exist
```

In [131]:
```
set1
```

Out[131]:
```
{6, 8, 10, 12}
```

Add

In [132]:
```
set1.add(1)            #add element in the set
```

In [133]:
```
set1
```

Out[133]: {1, 6, 8, 10, 12}

Copy

In [134]:
```
set1.copy()
```

Out[134]: {1, 6, 8, 10, 12}

Clear

In [135]:
```
set1.clear()
```

In [136]:
```
set1
```

Out[136]: set()

In [148]:
```
set2={1,2,3,4,5,6,7,8}
set3={2,4,6,8}
set4={1,3,5,7}
```

Difference

In [155]:
```
set5=set2.difference(set3)      #set2 element which are unique from set3
```

In [156]:
```
set5
```

Out[156]: set()

In [150]:
```
set2.difference(set4)
```

Out[150]: {2, 4, 6, 8}

In [151]:
```
set3.difference(set4)
```

Out[151]: {2, 4, 6, 8}

In [152]:
```
set2
```

Out[152]: {1, 2, 3, 4, 5, 6, 7, 8}

Difference_update

In [153]:
```
set2.difference_update(set4)        #keep only those element which are present in first set alone
```

In [154]:
```
set2
```

Out[154]: {2, 4, 6, 8}

In [157]:
```
print(set2)
print(set3)
print(set4)
```

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

Intersection

In [158]:
```
set2.intersection(set3)         #gives set2 elemennt which exist in set3
```

Out[158]:
```
{2, 4, 6, 8}
```

In [159]:
```
set2.intersection(set4)        #gives set2 elemennt which exist in set4
```

Out[159]:
```
set()
```

Intersection_update

In [162]:
```
set2.intersection_update(set4)
```

In [163]:
```
set2
```

Out[163]:
```
set()
```

In [164]:
```
set2.intersection_update(set3)
```

In [165]:
```
set2
```

Out[165]:
```
set()
```

In [166]:
```
print(set2)
print(set3)
print(set4)
```

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

Clear

In [167]:
```
set2.clear()           #delete all the elements in set
```

In [168]:
```
set2
```

Out[168]:
```
set()
```

Union

In [169]:
```
set3.union(set4)     #set elements along with all elements in specified set
```

Out[169]:
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

isdisjoint

In [173]:
```
set3.isdisjoint(set4)    #none of the elements are not present in both sets,output will be true,
```

Out[173]:
```
True
```

In [176]:
```
even={2,4,6,8}
odd={3,5,7,9}
```

In [177]:
```
even.isdisjoint(odd)
```

Out[177]:
```
True
```

In [180]:
```
a={4,5,6}
b={6,7,8}
```

In [181]:
```
a.isdisjoint(b)          #elements are present in both sets den its false
```

Out[181]:
```
False
```

In [192]:
```
c={1,2,3,4,5,6}
d={4,5,6}
e={1,2,3,4,5,6,8}
```

issuperset

In [193]:
```
c.issuperset(d)       #if all elements in the specified set exists in the original set,its true
```

Out[193]:
```
True
```

In [194]:
```
c.issuperset(e)      #if not one element matches its false
```

Out[194]:
```
False
```

issubset

In [195]:
```
d.issubset(e)        #true, if all set element are exists in specified set
```

Out[195]:
```
True
```

In [191]:
```
c.issubset(d)         #false if all set element are not in specified set
```

Out[191]:
```
False
```

Symmetric_difference

In [196]:
```
 c.symmetric_difference(d)    #elements that are not present in both sets
```

Out[196]:
```
{1, 2, 3}
```

In [197]:
```
c.symmetric_difference(e)
```

Out[197]:
```
{8}
```

symmetric_difference_update

In [205]:
```
c.symmetric_difference_update(d) #remove the same element and remains will come as output
```

In [206]:
```
c
```

Out[206]:
```
{1, 2, 3}
```

In [207]:
```
d
```

Out[207]:
```
{4, 5, 6}
```

Empty List,Tuple,Set,Dictionary

List

In [208]:
```
empty_list=[]
type(empty_list)
```

Out[208]:
```
list
```

Tuple

In [209]:
```
empty_tuple=()
type(empty_tuple)
```

Out[209]:
```
tuple
```

Set

In [211]:
```
empty_set=set()
type(empty_set)
```

Out[211]:
```
set
```

Dictionary

In [212]:
```
empty_dict={}
type(empty_dict)
```

Out[212]:
```
dict
```

Dictionary

Properties of dictionary

stores data as key and value,"key":"value".
it won't allow duplicate value.
it is mutable and chaneable.

In [213]:
```
roll_no1={'name':'Gomathi Ravi','Qualification':'M.Tech'}   #key-name,value-gomathi
```

In [214]:
```
roll_no1
```

Out[214]:
```
{'name': 'Gomathi Ravi', 'Qualification': 'M.Tech'}
```

In [215]:
```
roll_no1['Age']='25'     #adding this pair to existing dict.
```

In [216]:
```
roll_no1
```

Out[216]:
```
{'name': 'Gomathi Ravi', 'Qualification': 'M.Tech', 'Age': '25'}
```

In [297]:
```
roll_no1['name']='Ammu'      #it will overwrite the keyvalue in the dict,wont repeat same.
```

In [298]:
```
roll_no1
```

Out[298]:
```
{'name': 'Ammu', 'Qualification': 'M.Tech', 'Age': '25'}
```

Indexing and Slicing

In [283]:
```
food={'fruit':{'apple':'vitamin c','pomegranate':'vitamin e'},'vegetable':['potato','carrot'],'juice':{'f
```

In [284]:
```
food
```

Out[284]:
```
{'fruit': {'apple': 'vitamin c', 'pomegranate': 'vitamin e'},
 'vegetable': ['potato', 'carrot'],
 'juice': {'fruit juice': 'healthy', 'milkshake': 'tasty'}}
```

In [285]:
```
food['fruit']['apple']
```

Out[285]:
```
'vitamin c'
```

In [286]:
```
food['fruit']['pomegranate']
```
Out[286]: 'vitamin e'

In [287]:
```
food['vegetable'][0]
```
Out[287]: 'potato'

In [288]:
```
food['vegetable'][0]
```
Out[288]: 'potato'

In [289]:
```
food['vegetable'][1]
```
Out[289]: 'carrot'

In [290]:
```
food['juice']['fruit juice']
```
Out[290]: 'healthy'

In [291]:
```
food['juice']['milkshake']
```
Out[291]: 'tasty'

In [292]:
```
food['vegetable'][1][2]
```
Out[292]: 'r'

In [293]:
```
food['juice']['milkshake'][2]
```
Out[293]: 's'

In [294]:
```
food['juice']['fruit juice'][0]
```
Out[294]: 'h'

In [295]:
```
food['fruit']['apple'][3]
```
Out[295]: 'a'

Tab Function

In [300]:
```
roll_no1
```
Out[300]: {'name': 'Ammu', 'Qualification': 'M.Tech', 'Age': '25'}

Keys

In [301]:
```
roll_no1.keys()     #keys in the dict
```
Out[301]: dict_keys(['name', 'Qualification', 'Age'])

Values

In [302]:
```
roll_no1.values()     #values in the dict
```
Out[302]: dict_values(['Ammu', 'M.Tech', '25'])

items

In [303]:
```
roll_no1.items()     #item=(key,value), gives the items in the dict
```

Out[303]:
```
dict_items([('name', 'Ammu'), ('Qualification', 'M.Tech'), ('Age', '25')])
```

Pop

In [305]:
```
roll_no1.pop('Age')     #removes the specific item from dict
```

Out[305]:
```
'25'
```

In [307]:
```
roll_no1
```

Out[307]:
```
{'name': 'Ammu', 'Qualification': 'M.Tech'}
```

popitem

In [308]:
```
roll_no1.popitem()      #removes the last key-value pair from the dict
```

Out[308]:
```
('Qualification', 'M.Tech')
```

Get

In [309]:
```
roll_no1.get('name')       #returns the value of the specific key
```

Out[309]:
```
'Ammu'
```

In [310]:
```
roll_no1.get('Ammu')   #nouse
```

In [311]:
```
roll_no1
```

Out[311]:
```
{'name': 'Ammu'}
```

fromkeys

In [312]:
```
roll_no2={'name','qualification','age'}        #create new dict from given sequence of elements with value
res=dict.fromkeys(roll_no2)
res
```

Out[312]:
```
{'age': None, 'name': None, 'qualification': None}
```

In [318]:
```
x=("a","b","c")
y=(1,2,3)
res=dict.fromkeys(x,y)
res
```

Out[318]:
```
{'a': (1, 2, 3), 'b': (1, 2, 3), 'c': (1, 2, 3)}
```

In [327]:
```
roll_no3=('name','qualification','age')
a="students"
res=dict.fromkeys(roll_no3,a)
res
```

Out[327]:
```
{'name': 'students', 'qualification': 'students', 'age': 'students'}
```

In [328]:
```
x=("a","b","c")
y=(1)
res=dict.fromkeys(x,y)
res
```

Out[328]:
```
{'a': 1, 'b': 1, 'c': 1}
```

In [329]:
```
key={'a','e','i','o','u'}
value='vowel'
res=dict.fromkeys(key,value)
res
```

Out[329]:
```
{'e': 'vowel', 'a': 'vowel', 'u': 'vowel', 'i': 'vowel', 'o': 'vowel'}
```

In [331]:
```
roll_n04={'name':'Gomathi Ravi','qualification':'M.Tech'}
```

setdefault

In [332]:
```
roll_n04.setdefault('name','age')     #returns the value of the key
```

Out[332]:
```
'Gomathi Ravi'
```

In [333]:
```
roll_n04.setdefault('age','25')     # if key not exist, mentioned pair will added as item in dict
```

Out[333]:
```
'25'
```

In [334]:
```
roll_n04
```

Out[334]:
```
{'name': 'Gomathi Ravi', 'qualification': 'M.Tech', 'age': '25'}
```

In [336]:
```
roll_n04.setdefault('mobile no','call')    #can assign the value as we want
```

Out[336]:
```
'call'
```

In [338]:
```
roll_n04
```

Out[338]:
```
{'name': 'Gomathi Ravi',
 'qualification': 'M.Tech',
 'age': '25',
 'mobile no': 'call'}
```

In [342]:
```
roll_n04.update({'location':'chennai'})     #inserts the specified items(dict or iterable) in dict
```

In [343]:
```
roll_n04
```

Out[343]:
```
{'name': 'Gomathi Ravi',
 'qualification': 'M.Tech',
 'age': '25',
 'mobile no': 'call',
 'location': 'chennai'}
```

update

In [349]:
```
roll_n04.update({1:['Chennai','bangalore']})   #iterable
```

In [350]:
```
roll_n04
```

Out[350]:
```
{'name': 'Gomathi Ravi',
 'qualification': 'M.Tech',
 'age': '25',
 'mobile no': 'call',
 'location': 'chennai',
 1: ['Chennai', 'bangalore']}
```

Conditional Statement

#if
#if---else
#if---elif---elif....

#nested if

In [352]:
```
a=2
if a==2:
    print("hi gomathi")
print("how are you")
```

```
hi gomathi
how are you
```

In [353]:
```
a=4
if a!=5:
    print('hi gomathi')
print('how are you')
```

```
hi gomathi
how are you
```

In [354]:
```
a=4
if a>5:
    print('hi gomathi')
print('how are you')
```

```
how are you
```

In [355]:
```
a=4
if a<5:
    print('hi gomathi')
print('how are you')
```

```
hi gomathi
how are you
```

In [356]:
```
a=4
if a<=5:
    print('hi gomathi')
print('how are you')
```

```
hi gomathi
how are you
```

In [357]:
```
a=4
if a>=5:
    print('hi gomathi')
print('how are you')
```

```
how are you
```

In [359]:
```
a=2
b=4
print('hi')
if(a+b==6 and a<b):
    print('gomathi')
print('how are you')
```

```
hi
gomathi
how are you
```

In [360]:
```
a=2
b=4
print('hi')             #true
if(a+b==6 and a<b):
    print('gomathi')
else:
    print('hello')
print('how are you')
```

```
hi
gomathi
how are you
```

In [361]:
```
a=2
b=4
print('hi')                #false statement
if(a+b==5 and a<b):
    print('gomathi')
else:
    print('hello')
print('how are you')
```

```
hi
hello
how are you
```

In [363]:
```
a=21
if a%7==0:
    print('learn python')
else:
    print('wlcome')
```

```
learn python
```

In [366]:
```
a=22
if a==10:
    print('welcome')
elif a!=22:
        print('python')
elif a%2==0:
        print('hello')
else:
    print('dot')
print("say hi to all")
```

```
hello
say hi to all
```

In [ ]:
```

```