# Python Data Structures

## Structure:

- It stores the collection of data with same data type

### Data Structure: ¶

- It stores the collection of data with different data types

### Python Data Structures:

- Lists
- Tuples
- Dictionaries
- Sets

### What is meant by list?

- It stores the collection of data
- Represented by []
- Lists sre mutable(we can change or modify)

## List Methods:

- append() - adding the element at the end of the list.
- extend() - entire list is adding to other list.
- index() - returns the position of the element.
- count() - returns the number of elements of the list.
- pop() - deletes the element at the last.
- insert() - insert an element into a list.
- clear() - clears the entire list.
- sort() - by default arranges in the ascending order.
- remove() - remove the element.

In [17]:

```
s=[30,30,10,20,30,30,40,50]
a=s.append(100)             #append
print(a)
b=s.count(30)               #count
print(b)
c=s.insert(2,25)            #insert
print(c)
print(s.pop())              #pop
print(s.pop(3))             #pop at particular given position
print(s.index(40))          #index
d=s.sort()                  #sorting in ascending order
print(d)
e=s.sort(reverse=True)      #sorting in descending order
print(e)
print(s.clear())            #clear
```

```
None
4
None
100
10
6
None
None
None
```

## Mathematical Functions

- sum
- min
- max
- len

In [21]:

```
m=[1,2,3,4,5]
print(sum(m))
print(min(m))
print(max(m))
print(len(m))
print(sum(m)//len(m))
```

```
15
1
5
5
3
```

In [33]:

```
#Task-1:
#Average of alternate elements in given list
l=[12,34,67,89,78]
l=l[::2]
a=sum(l)
b=len(l)
c=a/b
c
```

Out[33]:

```
52.333333333333336
```

In [39]:

```
#Task-2:
#Add list elements
a=[1,2,3]
b=[3,4,6]
print([a[0]+b[0],a[1]+b[1],a[2]+b[2]])
```

```
[4, 6, 9]
```

In [40]:

```
a=[1,2,3]
b=[3,4,6]
l=[]
for i in range(len(a)):
    l.append(a[i]+b[i])
print(l)
```

```
[4, 6, 9]
```

In [49]:

```
#Task-3:
l=[2,3,4]
print([l[1],l[2],l[0]])
```

```
[3, 4, 2]
```

In [50]:

```
def rot(l):
    return[l[1],l[2],l[0]]
rot([2,3,4])
```

Out[50]:

```
[3, 4, 2]
```

In [1]:

```
#Task-4:
l=[1,2,3,3,2,1]
li=[]
for i in l:
    if i not in li:
        li.append(i)
print(li)
```

```
[1, 2, 3]
```

In [ ]:

```
#Task-5:
l=[12,45,78,90,87,90,45,90]
li=[]
for i in l:
    if i not in li:
        li.append(i)
print(li)
li.sort()
print(li[-2])
```

In [5]:

```
#Task-6:
l=[['a',1,2,3,4],['b',4,5,6,7],['c',9,8,7,6]]
for i in range(len(l)):
    l[i].append(sum(l[i][1:]))
l
```

Out[5]:

```
[['a', 1, 2, 3, 4, 10], ['b', 4, 5, 6, 7, 22], ['c', 9, 8, 7, 6, 30]]
```

## Tuples

- It is similar to lists which is used to store the collection of data.
- These are immutable.
- Represented by using parenthesis "( )".

In [1]:

```
t=('sri',1,2,3,4)
t
```

Out[1]:

```
('sri', 1, 2, 3, 4)
```

In [2]:

```
t='hi','how','are','you'
print(t)
```

```
('hi', 'how', 'are', 'you')
```

In [6]:

```
t=('hi','how','are','you')
t[0]
```

Out[6]:

```
'hi'
```

In [7]:

```
t[::-1]
```

Out[7]:

```
('you', 'are', 'how', 'hi')
```

In [8]:

```
t[::2]
```

Out[8]:

```
('hi', 'are')
```

In [10]:

```
t1=('1','2','3')
t2=('hi','good','afternoon')
tu=t1+t2
print(tu)
```

```
('1', '2', '3', 'hi', 'good', 'afternoon')
```

In [11]:

```
# Nested tuples
tu=(t1,t2)
tu
```

Out[11]:

```
(('1', '2', '3'), ('hi', 'good', 'afternoon'))
```

In [12]:

```
t=('1','2','3')
t=t*3
t
```

Out[12]:

```
('1', '2', '3', '1', '2', '3', '1', '2', '3')
```

In [13]:

```
tu*3
```

Out[13]:

```
(('1', '2', '3'),
 ('hi', 'good', 'afternoon'),
 ('1', '2', '3'),
 ('hi', 'good', 'afternoon'),
 ('1', '2', '3'),
 ('hi', 'good', 'afternoon'))
```

In [18]:

```
#Task-1:
t=('s','t','r','i','n','g')
print(''.join(t))
```

```
string
```

In [20]:

```
#Task-2:
#Creating atupleby using for loop
t=(1,2,3,4)
n=4
for i in range(1,n+1):
    t=(t,)
    print(t)
```

```
((1, 2, 3, 4),)
(((1, 2, 3, 4),),)
((((1, 2, 3, 4),),),)
(((((1, 2, 3, 4),),),),)
```

## Dictionaries

- It contains values as pairs that are defined as key and value.
- Represented by using curly braces "{ }".
- These are mutable.

In [22]:

```
d={'k1':10,'k2':20,'k3':30}
d
```

Out[22]:

```
{'k1': 10, 'k2': 20, 'k3': 30}
```

In [23]:

```
d.keys()
```

Out[23]:

```
dict_keys(['k1', 'k2', 'k3'])
```

In [24]:

```
d.values()
```

Out[24]:

```
dict_values([10, 20, 30])
```

In [25]:

```
d.items()
```

Out[25]:

```
dict_items([('k1', 10), ('k2', 20), ('k3', 30)])
```

In [29]:

```
d.get('k2')
```

Out[29]:

```
20
```

In [33]:

```
d.pop('k2')
```

Out[33]:

```
20
```

In [34]:

```
d.popitem()
```

Out[34]:

```
('k3', 30)
```

In [37]:

```
d.update({'colour':'green','fruit':'banana'})
d
```

Out[37]:

```
{'k1': 10, 'colour': 'green', 'fruit': 'banana'}
```

In [12]:

```
#Task-1:
d1={'k1':10,'k2':20,'k3':30}
d2={'v1':66,'v2':79,'v3':84}
d3={'place':"zoo",'animal':"monkey"}
d4=(d1,d2,d3)
d4
```

Out[12]:

```
({'k1': 10, 'k2': 20, 'k3': 30},
 {'v1': 66, 'v2': 79, 'v3': 84},
 {'place': 'zoo', 'animal': 'monkey'})
```

In [13]:

```
d1={'k1':10,'k2':20,'k3':30}
d2={'v1':66,'v2':79,'v3':84}
d3={'place':"zoo",'animal':"monkey"}
d={}
for i in (d1,d2,d3):
    d.update(i)
print(d)
```

```
{'k1': 10, 'k2': 20, 'k3': 30, 'v1': 66, 'v2': 79, 'v3': 84, 'place': 'zoo',
'animal': 'monkey'}
```

## Build a contact application by using dictionaries

- Add contacts
- Search contacts
- Update contacts
- Delete contacts

In [15]:

```
contacts={}
def addcontact(name,phone):
    if name not in contacts:
        contacts[name]=phone
        print("Contact is added",name)
    else:
        print("Contact already exists",name)
addcontact('namex',9876543219)
addcontact('namey',8976543218)
addcontact('namez',7659843217)
addcontact('namex',6308765436)
```

```
Contact is added namex
Contact is added namey
Contact is added namez
Contact already exists namex
```

In [16]:

```
contacts
```

Out[16]:

```
{'namex': 9876543219, 'namey': 8976543218, 'namez': 7659843217}
```

In [27]:

```
def searchcontact(name):
    if name in contacts:
        print("Contact exists and name is",name)
    else:
        print("Contact does not exist with",name)
searchcontact('namex')
searchcontact('name1')
contacts
```

```
Contact does not exist with namex
Contact does not exist with name1
```

Out[27]:

```
{'namey': 8976543218, 'namez': 7659843217}
```

In [28]:

```
def update(name,phone):
    if name in contacts:
        contacts[name]=phone
        print(name,phone)
    else:
        print("Contact does not exist with",name)
update('namex',9876543218)
update('name1',9876435548)
```

```
Contact does not exist with namex
Contact does not exist with name1
```

In [26]:

```
def delete(name):
    if name in contacts:
        contacts.pop(name)
        print("Contact is deleted")
    else:
        print("Cannot delete as it does not exist")
delete('namex')
delete('name1')
```

```
Cannot delete as it does not exist
Cannot delete as it does not exist
```

## Sets

- Collection of different datatype elements.
- Doesn't allow duplicates.
- It doesn't follow any order.
- It is mutable.
- Represented by "{ }".
- Using set() function we can define a set.

In [29]:

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

Out[29]:

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

In [30]:

```
print(dir(set),end=" ")
```

```
['__and__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc_
_', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash_
_', '__iand__', '__init__', '__init_subclass__', '__ior__', '__isub__', '__i
ter__', '__ixor__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__o
r__', '__rand__', '__reduce__', '__reduce_ex__', '__repr__', '__ror__', '__r
sub__', '__rxor__', '__setattr__', '__sizeof__', '__str__', '__sub__', '__su
bclasshook__', '__xor__', 'add', 'clear', 'copy', 'difference', 'difference_
update', 'discard', 'intersection', 'intersection_update', 'isdisjoint', 'is
subset', 'issuperset', 'pop', 'remove', 'symmetric_difference', 'symmetric_d
ifference_update', 'union', 'update']
```

In [31]:

```
s1={99,77,55,33}
s1.add(44)
s1
```

Out[31]:

```
{33, 44, 55, 77, 99}
```

In [39]:

```
s.update(s1)
s
```

Out[39]:

```
{3, 4, 33, 44, 55, 77, 99}
```

In [40]:

```
s.pop()
s
```

Out[40]:

```
{4, 33, 44, 55, 77, 99}
```

In [41]:

```
s.remove(99)
s
```

Out[41]:

```
{4, 33, 44, 55, 77}
```

In [43]:

```
s.discard(33)
s
```

Out[43]:

```
{4, 44, 55, 77}
```

In [44]:

```
s.intersection_update(s1)
s
```

Out[44]:

```
{44, 55, 77}
```

In [45]:

```
s2={1,2,3,4,5,6}
s3={4,5,6,7,8,9}
s2.difference_update(s3)
s2
```

Out[45]:

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

In [48]:

```
s4={1,2,3,4,5,6}
s5={4,5,6,7,8,9}
s4.symmetric_difference_update(s5)
s4
```

Out[48]:

```
{1, 2, 3, 7, 8, 9}
```

In [54]:

```
s1={1,2,3}
s2={1,2,3,4,5,6}
s1.issubset(s2)
```

Out[54]:

```
True
```

In [55]:

```
s2.issuperset(s1)
```

Out[55]:

```
True
```

In [57]:

```
s1.clear()
s1
```

Out[57]:

```
set()
```

In [ ]:

```
```