Data Structure

Tuple, List, Dictionary and Set

1. Tuple : odered collection of elements enclosed within() and Tuples are immutable.

In [1]:
```
tup1 = (1,2.4,"Boy",3+7j,True)
tup1
```
Out[1]:
```
(1, 2.4, 'Boy', (3+7j), True)
```

In [2]:
```
type(tup1)
```
Out[2]:
```
tuple
```

Extracting individual elements (Works same as strings)

In [3]:
```
tup1[1]
```
Out[3]:
```
2.4
```

In [4]:
```
tup1[0]
```
Out[4]:
```
1
```

In [5]:
```
tup1[-1]
```
Out[5]:
```
True
```

In [6]:
```
tup1[1:4]
```
Out[6]:
```
(2.4, 'Boy', (3+7j))
```

Tuple Basic Operations

In [7]:
```
#Finding Tuple length
```

In [8]:
```
len(tup1)
```
Out[8]:
```
5
```

In [9]:
```
#Concatinating Tuples
tup1 = (1,2,3)
tup2 = (4,5,6)
tup1 + tup2
```
Out[9]:
```
(1, 2, 3, 4, 5, 6)
```

In [10]:
```
tup1 = (1,'boy',3.1)
tup2 = (3,'girl', 3+4j)
tup1 + tup2
```
Out[10]:
```
(1, 'boy', 3.1, 3, 'girl', (3+4j))
```

In [11]:
```
#Repeating Tuple
tup1*3
```
Out[11]:
```
(1, 'boy', 3.1, 1, 'boy', 3.1, 1, 'boy', 3.1)
```

In [12]:
```
#Repeating + Concatenating
tup1*3+tup2
```
Out[12]:
```
(1, 'boy', 3.1, 1, 'boy', 3.1, 1, 'boy', 3.1, 3, 'girl', (3+4j))
```

Tuple Basic Functions

In [35]:
```
tup1=(1,3,5)
tup2=(2,4,6)
min(tup1)
```
Out[35]:
```
1
```

In [ ]:
```
#Finding Maximum Value
```

In [36]:
```
max(tup2)
```
Out[36]:
```
6
```

2. List: Odered collection of elements enclosed within []. Lists are mutable.

In [67]:
```
l1 = [1,"Boy",3.2, True]
l1
```
Out[67]:
```
[1, 'Boy', 3.2, True]
```

In [68]:
```
type(l1)
```
Out[68]:
```
list
```

In [69]:
```
l1[2] #calling index
```
Out[69]:
```
3.2
```

In [70]:
```
l1[1:3]
```
Out[70]:
```
['Boy', 3.2]
```

Modifying a List`

In [72]:
```
#Changing the element at jth index
```

In [73]:
```
l1[0]=100
l1
```
Out[73]:
```
[100, 'Boy', 3.2, True]
```

In [74]:
```
l1[1]="Girl"
l1
```
Out[74]:
```
[100, 'Girl', 3.2, True]
```

In [81]:
```
#Popping the jth element (Removing the jth element)
```

In [76]:
```
l1.pop()
```
Out[76]:
```
True
```

In [77]:
```
l1.pop(2)
```
Out[77]:
```
3.2
```

In [78]:
```
l1
```
Out[78]:
```
[100, 'Girl']
```

In [79]:
```
#Appending a new Element (Inserting in last position)
```

In [80]:
```
l1.append("Munni")
l1
```
Out[80]:
```
[100, 'Girl', 'Munni']
```

In [86]:
```
#But how to append at jth position? With 'Insert' function
l1.insert(1,'Boy')
l1
```
Out[86]:
```
[100, 'Boy', 'Girl', 'Munni']
```

In [ ]:
```
#Revercing element
```

In [84]:
```
l3 = [1,2,3]
l3.reverse()
l3
```
Out[84]:
```
[3, 2, 1]
```

In [85]:
```
l3_1 = [1,"apple",3.5,"gilr"]
l3_1.reverse()
l3_1
```
Out[85]:
```
['gilr', 3.5, 'apple', 1]
```

In [ ]:
```
#Sorting Element in alphabetical order
```

In [88]:
```
l4 = ['mango', 'jelly','butter','rolls']
l4.sort()
l4
```
Out[88]:
```
['butter', 'jelly', 'mango', 'rolls']
```

List Basic Operation

In [1]:
```
#1.Concatinating list
l1=[1,2,3]
l2=[4,5,6]
l1+l2
```
Out[1]:
```
[1, 2, 3, 4, 5, 6]
```

In [2]:
```
#Repeating Elements
l1 = [1,'Shila',3.6,True]
l1*3
```
Out[2]:
```
[1, 'Shila', 3.6, True, 1, 'Shila', 3.6, True, 1, 'Shila', 3.6, True]
```

3 . Dictionary: Dictionary is an unordered collection of key-value pairs enclosed within {}.It is mutable.

In [12]:
```
d1 = {"apple":50,"mango":100,"guava":200,"banana":300}
d1
```
Out[12]:
```
{'apple': 50, 'mango': 100, 'guava': 200, 'banana': 300}
```

In [13]:
```
type(d1)
```
Out[13]:
```
dict
```

In [14]:
```
d1.keys()
```
Out[14]:
```
dict_keys(['apple', 'mango', 'guava', 'banana'])
```

In [15]:
```
d1.values()
```
Out[15]:
```
dict_values([50, 100, 200, 300])
```

Modifying a Dictionary

In [17]:
```
#Adding a new Element
d1["grapes"]=150
d1
```
Out[17]:
```
{'apple': 50, 'mango': 100, 'guava': 200, 'banana': 300, 'grapes': 150}
```

In [18]:
```
#Changing the value of existing element
d1["apple"]=70
d1
```
Out[18]:
```
{'apple': 70, 'mango': 100, 'guava': 200, 'banana': 300, 'grapes': 150}
```

Dictionary Functions

In [ ]:
```
#Removing an element
```

In [19]:
```
d1.pop("apple")
d1
```
Out[19]:
```
{'mango': 100, 'guava': 200, 'banana': 300, 'grapes': 150}
```

In [21]:
```
#Upadte one dictionar's element with other
d2={"apple":10,"banana":20}
d3={"guava":30,"grapes":40}
d2.update(d3)
d2
```
Out[21]:
```
{'apple': 10, 'banana': 20, 'guava': 30, 'grapes': 40}
```

4. Set: set is an unordered, unindexed collection of elements enclosed within{}. unordered unindexed means that when you would print the elemnts, they would get printed in random fashion because in a 'Set' the elments has no index.

Duplicated are not allowed in sets

In [55]:
```
s1={1,"Boy",False,3.4}
s1
```
Out[55]:
```
{1, 3.4, 'Boy', False}
```

In [56]:
```
#See, the above ouput is random
```

In [57]:
```
type(s1)
```
Out[57]:
```
set
```

In [21]:
```
#Try to add duplicate element
s1 = {1,"Boy",False,"Boy"}
s1
```
Out[21]:
```
{1, 'Boy', False}
```

In [22]:
```
#"Boy" got printed only once, hence duplicates are not allowd in set.
```

Set Operation

In [23]:
```
#Adding new element
s1.add(3.4) #ads at random index.
s1
```
Out[23]:
```
{1, 3.4, 'Boy', False}
```

In [25]:
```
#Adding multiple elemnts
s1.update([2,4,6,"Girl", True,9.6])
s1
```
Out[25]:
```
{1, 2, 3.4, 4, 6, 9.6, 'Boy', False, 'Girl'}
```

In [63]:
```
#Removing an element
s1.remove("Girl")
s1
```
Out[63]:
```
{1, 2, 3.4, 4, 6, 9.6, 'Boy', False}
```

In [64]:
```
s2={900,100}
s2
```
Out[64]:
```
{100, 900}
```

In [65]:
```
s1.update(s2)
s1
```
Out[65]:
```
{1, 100, 2, 3.4, 4, 6, 9.6, 900, 'Boy', False}
```

In [66]:
```
s2
```
Out[66]:
```
{100, 900}
```

In [67]:
```
s1
```
Out[67]:
```
{1, 100, 2, 3.4, 4, 6, 9.6, 900, 'Boy', False}
```

Set Function

In [69]:
```
s1.union(s2)
```
Out[69]:
```
{1, 100, 2, 3.4, 4, 6, 9.6, 900, 'Boy', False}
```

In [70]:
```
s2
```
Out[70]:
```
{100, 900}
```

In [ ]:
```
#It seems that 'union' works similar to 'update'!
```

In [2]:
```
#Intersection of sets:To find the common elemnts between sets
set1 = {1,2,3,4,5}
set2 = {9,8,2,4,7}
set1.intersection(set2)
```
Out[2]:
```
{2, 4}
```