Data Structures in python

Data structures stores a set of values.

Arrays in python

Array consists of set of elements of the same type. Array is mutable that means we can change the elements inside the array or add or remove elements. Each item stored in an array is called an element and each location of an element in an array has a numerical index, which is used to identify the element. Index in array start from 0. In order to create array we use python array module. Arrays are generally defined in [ ].

In [3]:
```
import array
from array import *

#to create an array we use :
#array_name = array('typecode','initialisation') for eg:
arr_ = array('b',[10,20,30,40,50,60])
# the type code specifies the type of the array that we have created, 'b' re
arr_
```

Out[3]:
```
array('b', [10, 20, 30, 40, 50, 60])
```

Operations on Array

In [15]:
```
arr_[2] #to access an element of the array we use array name and square brac

```

Out[15]:
```
30
```

In [16]:
```
arr_.insert(2,100) #to insert an element in array we pass the index and the
arr_
```

Out[16]:
```
array('b', [10, 20, 100, 30, 40, 50, 60])
```

In [17]:
```
arr_.index(50) #In order to search for an element inside an array we use ind
```

Out[17]:
```
5
```

In [19]:
```
arr_[5]=120 #In order to update any existing element we pass the index the e
arr_
```

Out[19]:
```
array('b', [10, 20, 100, 30, 40, 120, 60])
```

In [20]:
```
arr_.append(125) # append() inserts the element towards the last of the arra
arr_
```

Out[20]:
```
array('b', [10, 20, 100, 30, 40, 120, 60, 125])
```

In [22]:
```
arr_.count(100) #return the count i.e, number of times the element has appea
```

Out[22]:
```
1
```

In [4]:
```
#Concatenation and extending array
#CONCATENATION
#lets create one more array
Arr_ = array('b',[70,80,90])
concat_ = arr_ + Arr_
print (arr_)
print (Arr_)
concat_ #it includes elements of both the arrays and stores it in concat_, b

```

```
array('b', [10, 20, 30, 40, 50, 60])
array('b', [70, 80, 90])
```

Out[4]:
```
array('b', [10, 20, 30, 40, 50, 60, 70, 80, 90])
```

In [32]:
```
#EXTEND:
#with extend command it changes one of the arrays as,
arr_.extend(Arr_)
print (arr_)
print (Arr_)
```

```
array('b', [10, 20, 30, 40, 50, 60, 70, 80, 90])
array('b', [70, 80, 90])
```

In [33]:
```
#pop: removes the element whose index is passed as a parameter, if no index
arr_.pop(5)
```

Out[33]:
```
60
```

In [37]:
```
arr_.remove(10)#removes the element whose value is passed as a parameter
arr_
```

Out[37]:
```
array('b', [20, 30, 40, 50, 60])
```

In [ ]:
```
del(arr_)#to delete the array
```

Lists:

Lists consists of set of elements of the multiple type. Lists is mutable that means we can change the elements inside the lists or add or remove elements. Each item stored in list is called an element and each location of an element in list has a numerical index, which is used to identify the element. Index in list start from 0. List are generaly defined in [ ].

In [3]:
```
lis_1 = ['abc','xyz','pqr',1,2,3,4,5]
lis_2= [30,12,70,50]
type(lis_1)
```

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

Operation on lists

Lists has the same operations as an array

In [71]:
```
lis_1.append(10) #append adds the element at the last of the list
lis_1.count(2) #returns the number of appearances of the given element
lis_1.extend(lis_2) #adds elements of the second list in the first list
lis_1.insert(5,50) #inserts the element 50 at index 5
lis_1.pop(2) #pops out the element at index 2
lis_1.remove(3) #remove element = 3
lis_2.sort() #sort the list in an increasing order
print(lis_2)
print (len(lis_2)) #len() function returns the length of the list that is pa
```

```
[12, 30, 50, 70]
4
```

In [52]:
```
lis_1.clear() #clears all the elements of the list
del(lis_1) #deletes the whole list (no instance in the memory left)
```

Tuples

In tuples alse we can store elements of any datatype. Only difference is that we cannot manipulate the tuple or in other words tuples are immutable. Tuples are generally defined in ( ).

In [53]:
```
tup_1 = (1,2,3,4,'abc','pqr')
tup_2 = (60,70,80)
#we can concatenate two tuples as it wont affect the actual tuples
concat_ = tup_1 + tup_2
print (tup_1)
print (tup_2)
print (concat_)
```

```
(1, 2, 3, 4, 'abc', 'pqr')
(60, 70, 80)
(1, 2, 3, 4, 'abc', 'pqr', 60, 70, 80)
```

Dictionary:

Dictionary store values in key and value format. In order to access values in dictionary we use key's instead of indices. A dictionary is represented by { }

Operation on dictionary:

In [67]:
```
dict_ = {'john':'12345','michael':67890,'Abrahm':45678,'David':89234,'Smith'
print (dict_['john']) #accessing a value in dictionary we need to call by ke
```

```
12345
```

In [56]:
```
dict_['Abrahm'] = 12345678 #to change value in dictionary, we directly give
dict_
```

Out[56]:
```
{'john': '12345',
 'michael': 67890,
 'Abrahm': 12345678,
 'David': 89234,
 'Smith': 23789}
```

In [57]:
```
dict_.keys() #to get all keys
```

Out[57]:
```
dict_keys(['john', 'michael', 'Abrahm', 'David', 'Smith'])
```

In [58]:
```
dict_.values() #to get all values
```

Out[58]:
```
dict_values(['12345', 67890, 12345678, 89234, 23789])
```

In [60]:
```
dict_['Warner'] = '00000' #to add new value to dictionary we specify the new
dict_
```

Out[60]:
```
{'john': '12345',
 'michael': 67890,
 'Abrahm': 12345678,
 'David': 89234,
 'Smith': 23789,
 'Warner': '00000'}
```

In [62]:
```
dict_.pop('john') #pops out the value of the key that has been passed
```

Out[62]:
```
'12345'
```

In [64]:
```
dict_.clear() # clears all the element of the dictionary but the instance of
dict_
```

Out[64]:
```
{}
```

In [68]:
```
del(dict_) #deletes the dictionary, i.e no values remain inside the dictiona
```

Converting list to tuple and vice-versa

In [73]:
```
x = (1,2,3,4) #tuple
# 'x' is a tuple ,in order to convert tuple to list we use
y = list(x)
type(y)
#Now 'y' is a list, similarly we use tuple() to convert a list to tuple
```

Out[73]:
```
list
```

Practise Questions:

In [11]:
```
#perform the following operations on the given list.
x = [1,2,3,4,5,6]
# update the element at index 0 by 150
# add an element 50 between 2 and 3
# add an element 100 at the end
# remove element 3 by passing index as parameter
# copy the elements of z = [10,20] in x
```

In [ ]:
```
# Perform the following operations on the dictionary:
## create a dictionary containing 5 names and their ages
## update the age of one of the person
## insert one more name with age 30
## clear all the elements of this dict
```

In [ ]:
```
# 1) Print in proper format  '  print      letters    in     correct   format
# 2) create a list name desserts holding two values 'ice-cream','cookies'
# 3) sort desserts in alphabetical order
# 4) display the index of the ice-cream
# 5) copy the contents of the desserts in other list called object
# 6) remove cookies from desserts
```