<https://www.cs.usfca.edu/~galles/visualization/Algorithms.html>

Data structures deal with how the data is organized and held in the memory when a program processes it. It is important to note that the data that is stored in the disk as part of persistent storages (like relational tables) are not referred as data structure here.

The various data structures in computer science are divided broadly into two categories:

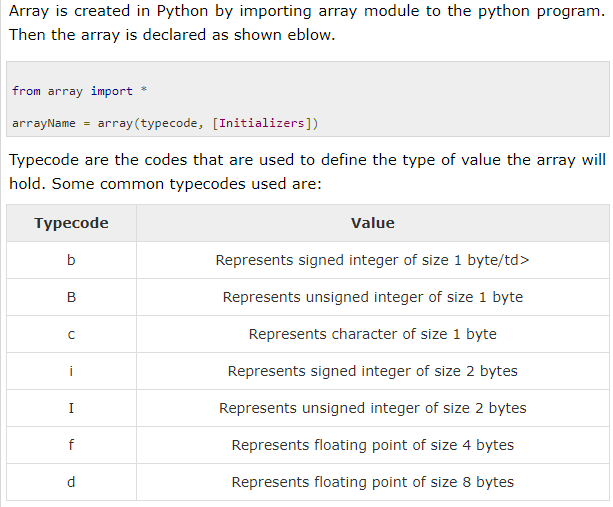
#### **LINEAR DATA STRUCTURES**

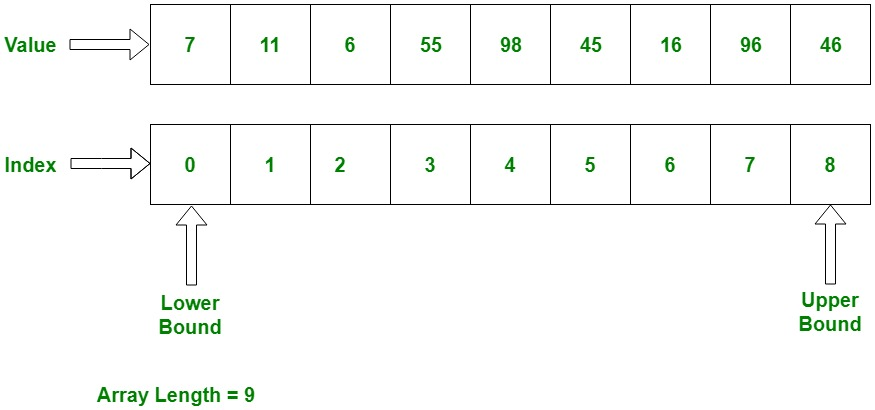
These are the data structures which store the data elements in a sequential manner.

* **Array:** It is a sequential arrangement of data elements paired with the index of the data element. Index starts with 0.

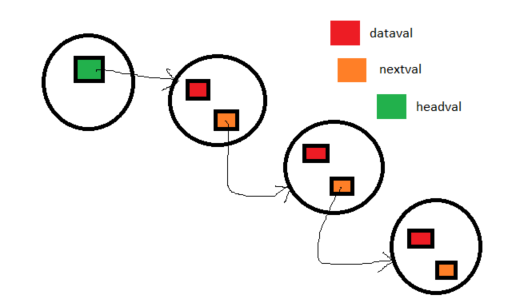
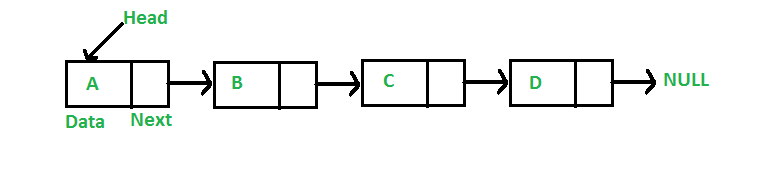
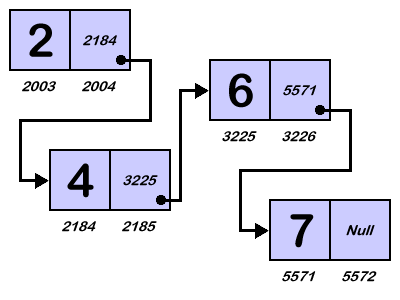
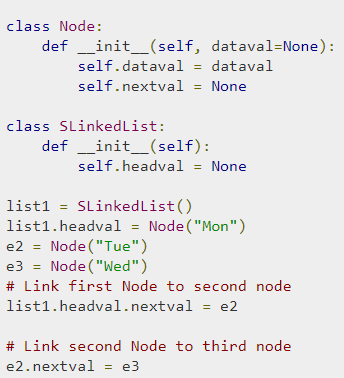
**Following are the basic operations supported by an array.**

* Traverse − print all the array elements one by one.
* Insertion − Adds an element at the given index.
* Deletion − Deletes an element at the given index.
* Search − Searches an element using the given index or by the value.
* Update − Updates an element at the given index.





* **Linked List:** Each data element contains a link to another element along with the data present in it. A linked list is created by using the node class. We create a Node object and create another class to use this node object. We pass the appropriate values through the node object to point the to the next data elements. The below program creates the linked list with three data elements. In the next section we will see how to traverse the linked list.

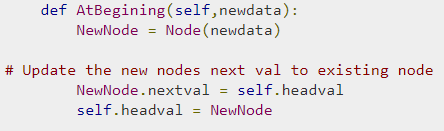


The above pic with circles explain the code above so the object with green square is SLinkedList one, other 3 are Node objects the dataval variable holds value while nextval point towards next object of node class

### 

### Inserting at the Beginning of the Linked List

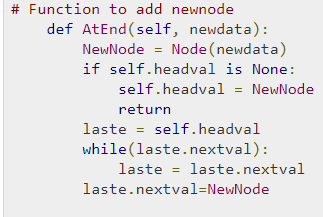
This involves pointing the next pointer of the new data node to the current head of the linked list. So the current head of the linked list becomes the second data element and the new node becomes the head of the linked list.



The above one works this way when we want to add value at the beginning we pass value to the above function as above. This will create new node object which is assigned to NewNode then the variable inside this object which is nextval will be moved to the headval value which will be first object and the value of headval will be moved to recently created new object which is first one in line.

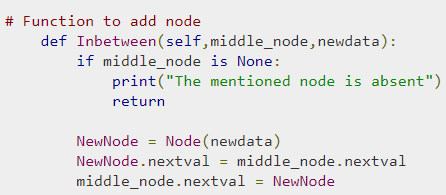
### Inserting at the End of the Linked List

This involves pointing the next pointer of the the current last node of the linked list to the new data node. So the current last node of the linked list becomes the second last data node and the new node becomes the last node of the linked list



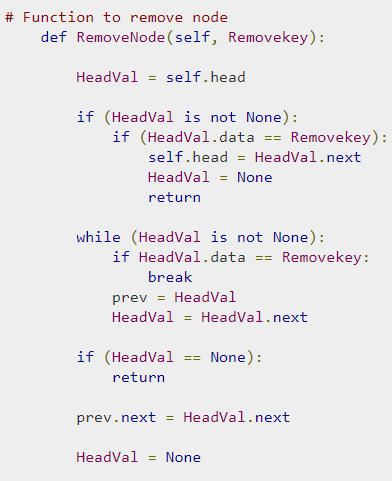
### Inserting in between two Data Nodes

This involves changing the pointer of a specific node to point to the new node. That is possible by passing in both the new node and the existing node after which the new node will be inserted. So we define an additional class which will change the next pointer of the new node to the next pointer of middle node. Then assign the new node to next pointer of the middle node.

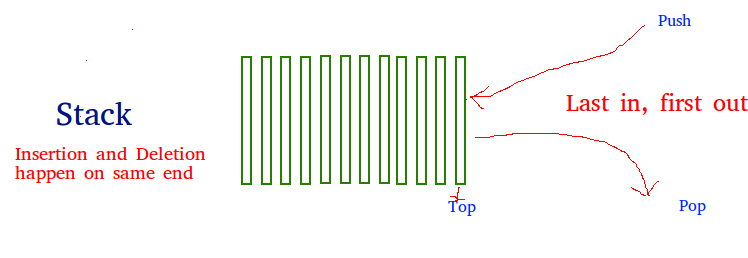
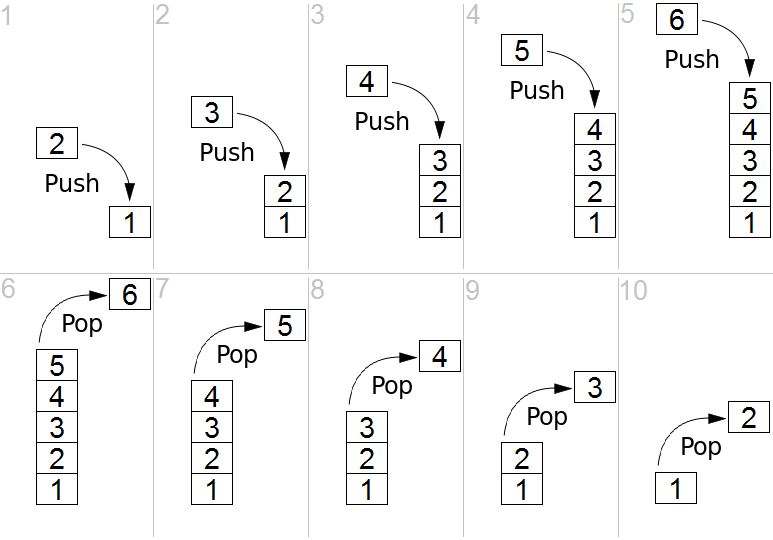


## Removing an Item form a Liked List

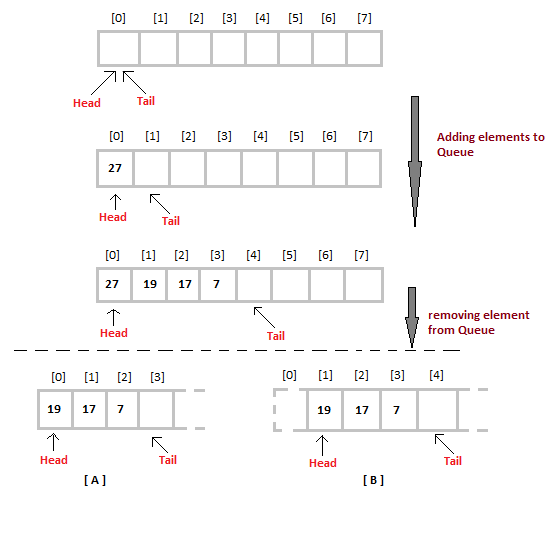
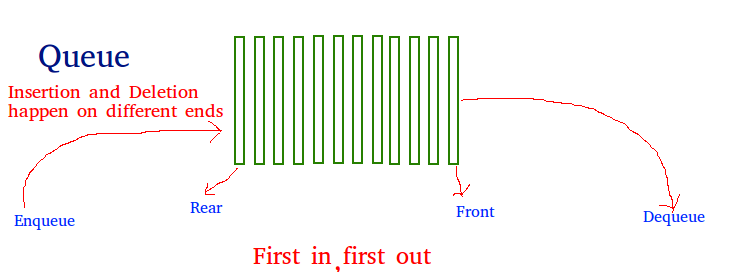
We can remove an existing node using the key for that node. In the below program we locate the previous node of the node which is to be deleted. Then point the next pointer of this node to the next node of the node to be deleted.



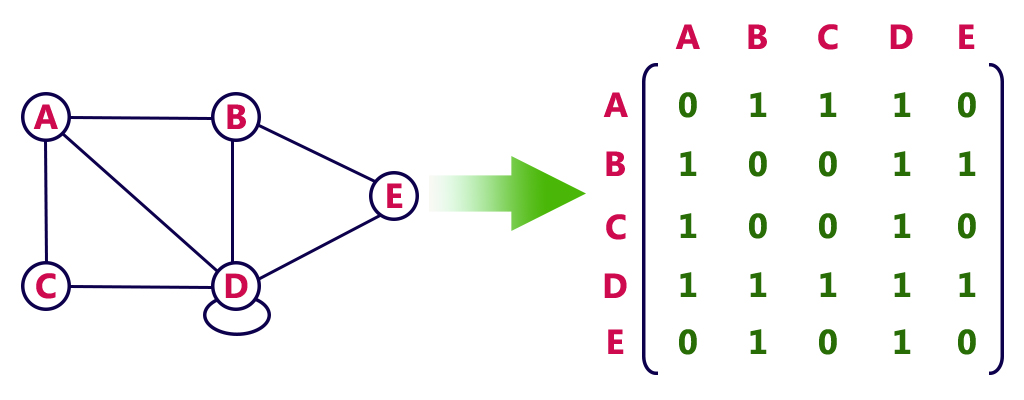
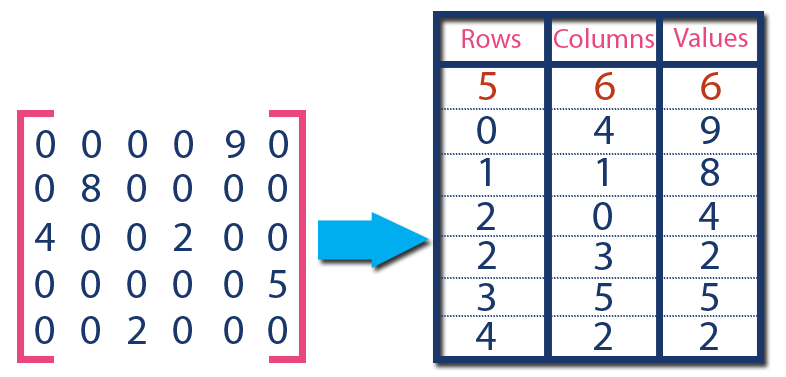
* **Stack:** It is a data structure which follows only to specific order of operation. LIFO(last in First Out) or FILO(First in Last Out).



* **Queue:** It is similar to Stack but the order of operation is only FIFO(First In First Out).



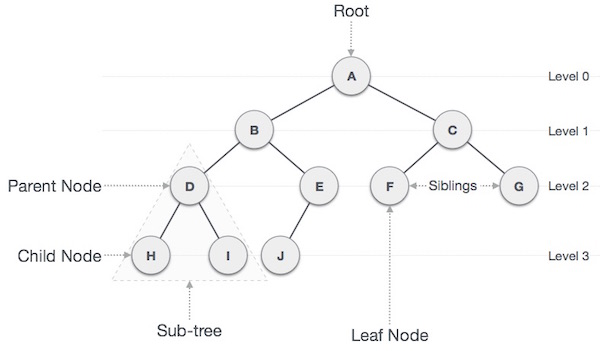
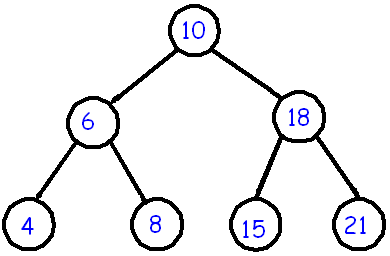
* **Matrix:** It is two dimensional data structure in which the data element is referred by a pair of indices.



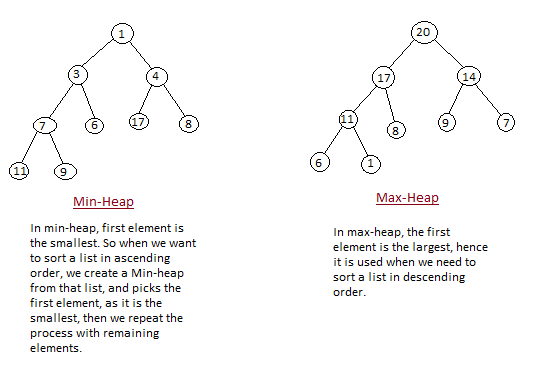
#### **NON-LINEAR DATA STRUCTURES**

These are the data structures in which there is no sequential linking of data elements. Any pair or group of data elements can be linked to each other and can be accessed without a strict sequence.

* **Binary Tree:** It is a data structure where each data element can be connected to maximum two other data elements and it starts with a root node. We create a tree data structure in python by using the concept of node.

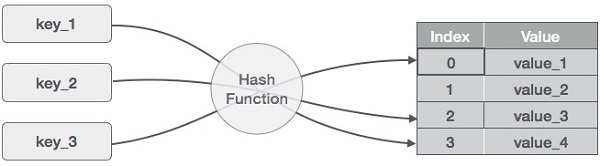
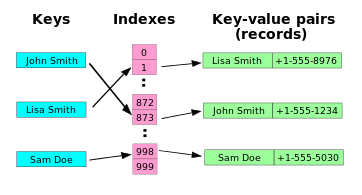


* **Heap:** It is a special case of Tree data structure where the data in the parent node is either strictly greater than/ equal to the child nodes or strictly less than it’s child nodes.



A heap is created by using python’s inbuilt library named heapq. This library has the relevant functions to carry out various operations on heap data structure. Below is a list of these functions.

* heapify - This function converts a regular list to a heap. In the resulting heap the smallest element gets pushed to the index position 0. But rest of the data elements are not necessarily sorted.
* heappush – This function adds an element to the heap without altering the current heap.
* heappop - This function returns the smallest data element from the heap.
* heapreplace – This function replaces the smallest data element with a new value supplied in the function
* **Hash Table:** It is a data structure which is made of arrays associated with each other using a hash function. It retrieves values using keys rather than index from a data element.



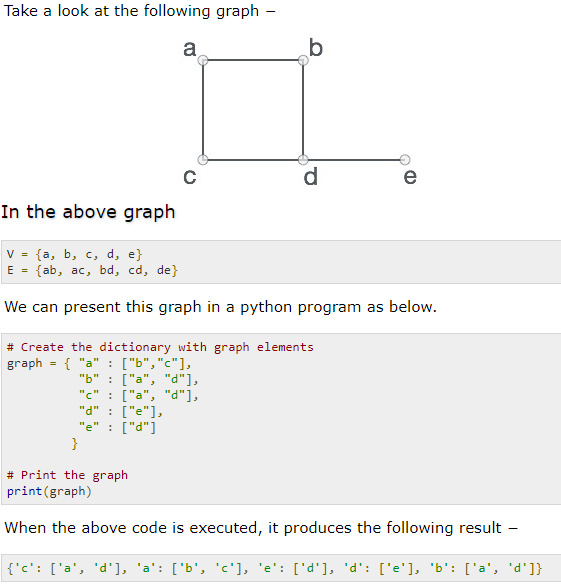
Hash tables are a type of data structure in which the address or the index value of the data element is generated from a hash function. That makes accessing the data faster as the index value behaves as a key for the data value. In other words Hash table stores key-value pairs but the key is generated through a hashing function. So the search and insertion function of a data element becomes much faster as the key values themselves become the index of the array which stores the data.

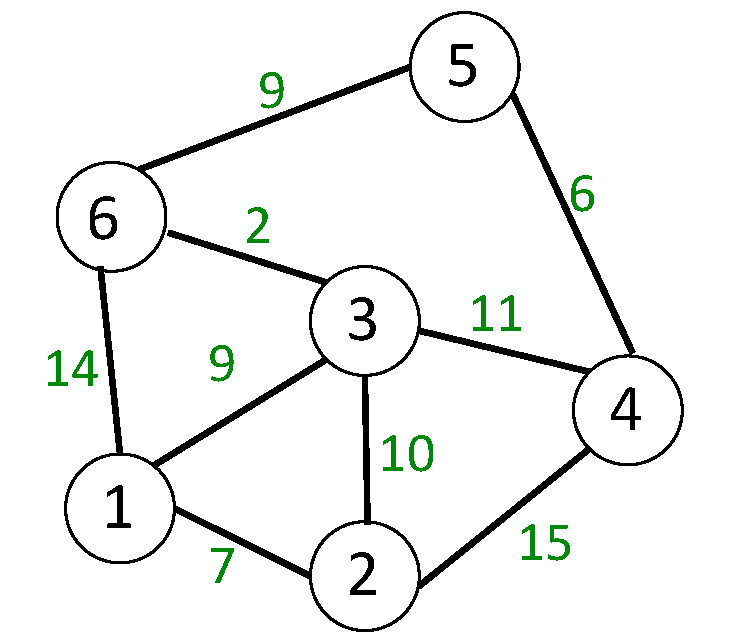
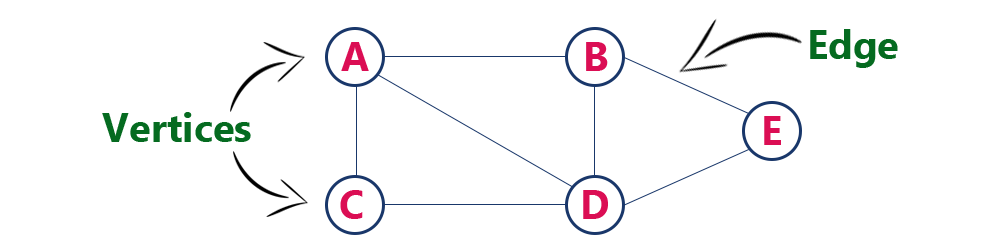
In Python, the Dictionary data types represent the implementation of hash tables. The Keys in the dictionary satisfy the following requirements.

* The keys of the dictionary are hashable i.e. the are generated by hashing function which generates unique result for each unique value supplied to the hash function.
* The order of data elements in a dictionary is not fixed.

**Graph: .**It is an arrangement of vertices and nodes where some of the nodes are connected to each other through links.

A graph can be easily presented using the python dictionary data types. We represent the vertices as the keys of the dictionary and the connection between the vertices also called edges as the values in the dictionary.

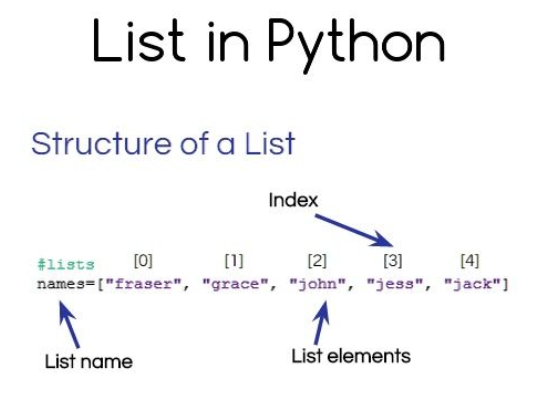




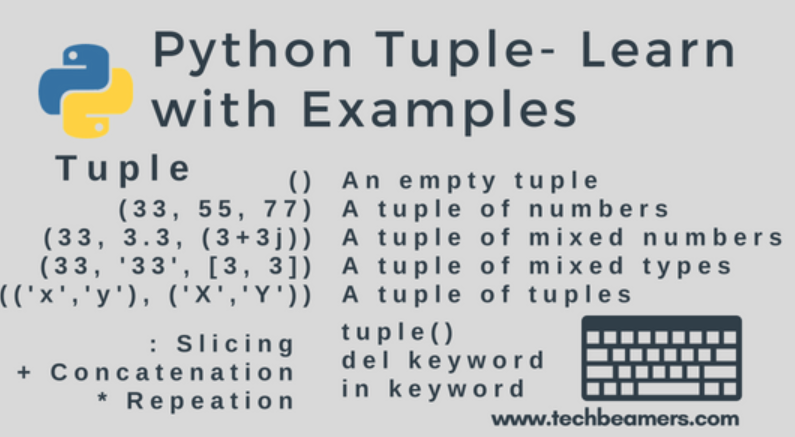
#### **PYTHON SPECIFIC DATA STRUCTURES**

These data structures are specific to python language and they give greater flexibility in storing different types of data and faster processing in python environment.

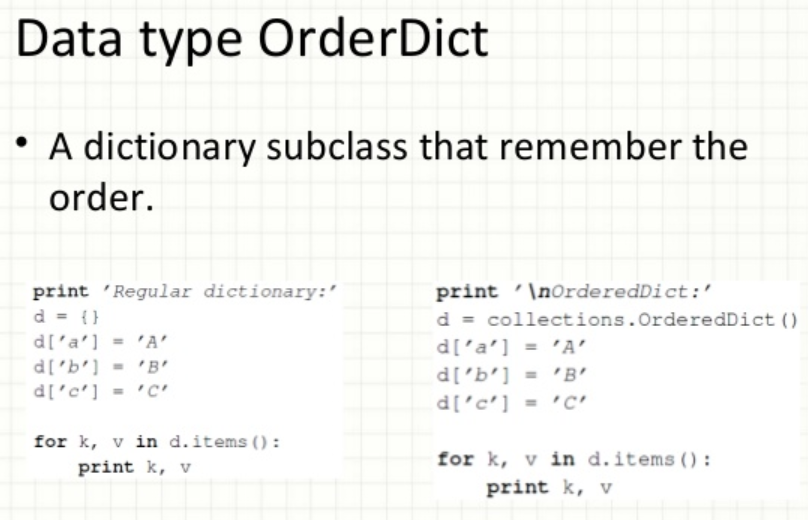
* **List:** It is similar to array with the exception that the data elements can be of different data types. You can have both numeric and string data in a python list.



* **Tuple:** Tuples are similar to lists but they are immutable which means the values in a tuple cannot be modified they can only be read.

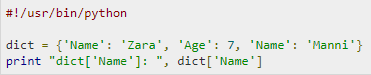


* **Dictionary:** The dictionary contains Key-value pairs as its data elements.

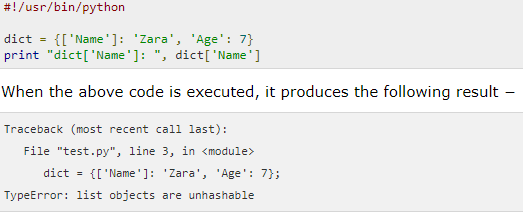


There are two important points to remember about dictionary keys −

* More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins.



* Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed.



**Set:**

A set is a collection of items not in any particular order.

* The elements in the set cannot be duplicates.
* The elements in the set are immutable(cannot be modified) but the set as a whole is mutable.
* There is no index attached to any element in a python set. So they do not support any indexing or slicing operation.
* The sets in python are typically used for mathematical operations like union, intersection, difference and complement etc.

**ChainMaps**: Python Maps also called ChainMap is a type of data structure to manage multiple dictionaries together as one unit. The combined dictionary contains the key and value pairs in a specific sequence eliminating any duplicate keys. The best use of ChainMap is to search through multiple dictionaries at a time and get the proper key-value pair mapping.