Basic Data Structures- Python

# 1. Arrays - Lists

Arrays store data in continuous memory address. Python uses lists as array.

* List is dynamic array
* hetrogeneous
* allows nesting to create multidimensional array
* Supports Indexing
* Supports Slicing
* Supports Iteration [in cases like ‘for’ loop]
* Mutable

#### Complexity

* Lookup time by index: O(1)
* Lookup time by value: O(n)
* Traversal: O(n)
* Insertion by index: O(n) # can use arr.insert()
* Deletion by index: O(n)

#### Important Methods

* Insert(), append()
* sort(), reverse(), sorted() -> all have time complexity O(nlogn)
* remove(), pop()
* len()
* index(element), count(element)
* copy()
* "*separator* ".join(list)

#### List Comprehension

L = [2\*i+1 for i in range(10)] creates list [1,3,5,7,9,11,13,15,17,18]

# 2. Linked List

A linked list consists of a series of nodes, each containing two main components: the data or value that the node holds and a reference (or pointer) to the next node in the sequence.The first node of the linked list is called the "head."

* Insertion is easier
* Dynamic memory allocation is very efficient



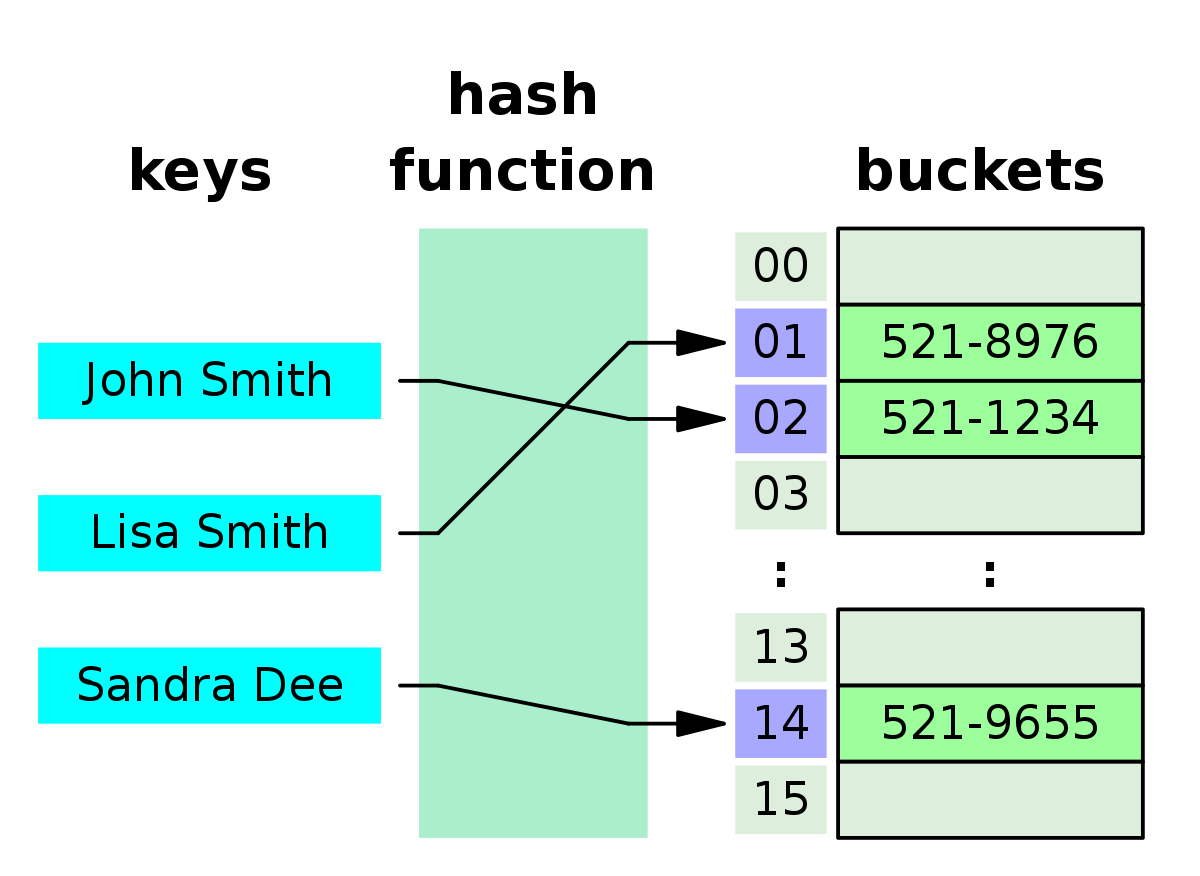
#### Complexity

* Insertion/ Deletion at head: O(1)
* Insertion/ Deletion: O(n)
* Traversal: O(n)
* Access element by value/index [search]: O(n)

Doubly linked list: Links to next and previous elements

# 3. Hash Maps

Hash maps/ Hash tables are used to store key-value pair. First we create an empty array. Then we define a hash function that converts the key to an index value. This index is used to store the corresponding value.

Python dictionary implements hash maps.

Counter in python is another data structure that implements hash maps. It counts the number of occurences of every element and stores them in a hash map.

A hash value, also known as a hash code or hash digest, is a fixed-size string of characters generated by a hash function from input data of any size.

#### Complexity

* Insertion/ Deletion: O(1)
* Lookup by key[search]: O(1) # This is average case. Collisions can reduce this

#### Methods

* get(): used to return a default value(like 0) if the key is not present in a hashmap.

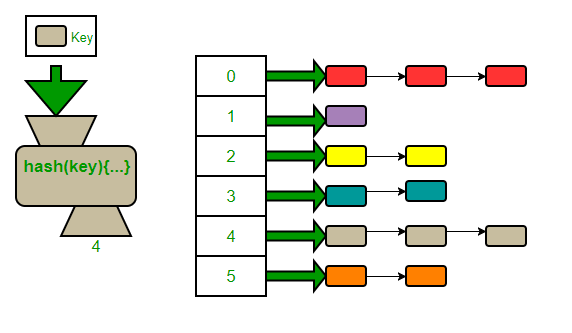
Eg: map = {‘a’:[1,2,3]}

print(map.get(‘b’,0)) # prints 0

#### Collisions

When multiple keys point to the same index, the situation is called collision. Some common ways to handle collision are:

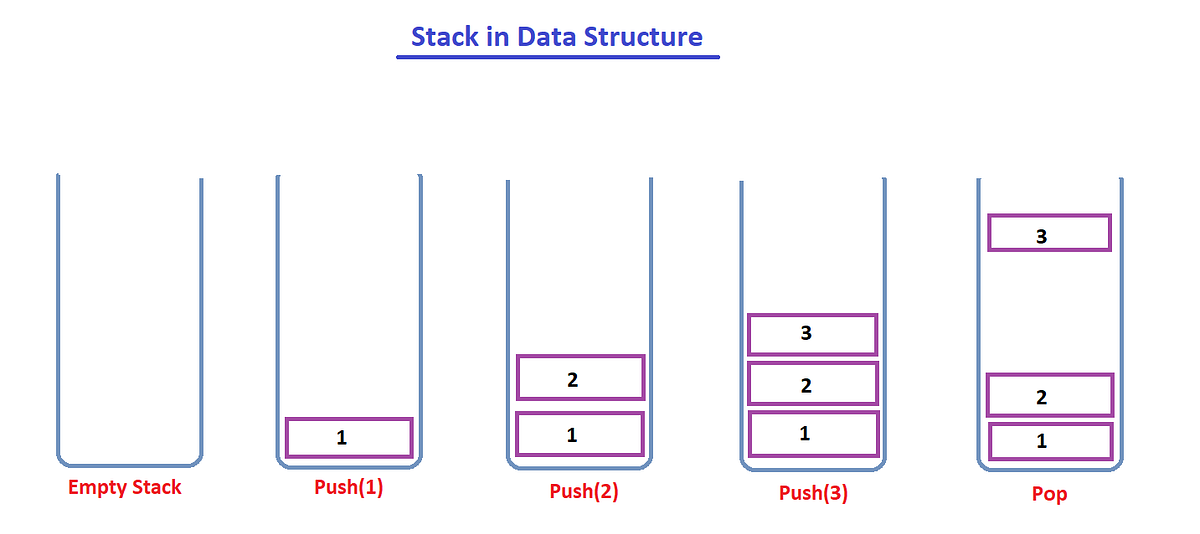
1. **Chaining**: Creating an linked list at the collision part and chaining the (key,value) there.



1. **Linear Probing**: Stores the value in the next available slot in the array. We are ‘linearly probing’ the array for an empty slot.

# Stack

Stack is a linear data structure that follows a particular order in which the operations are performed. The order is that the last inserted element is popped out first [lifo].



**Top:** The last inserted element in stack. It is imagined to exist at the topmost position of the stack.

**Seek:** The process of displaying the top element

**Push:** insert the element to the top of stack

**Pop:** Remove element from top of stack

We can use list as stack in python. We can use stack[-1] to seek the top element, append() to push and pop() to pop. But a problem with list is that it doesnot help us create static stack, with a specified number of elements. We can use **collections.deque** to implement static stack.