**Data Structures:**

**Data Structures:**

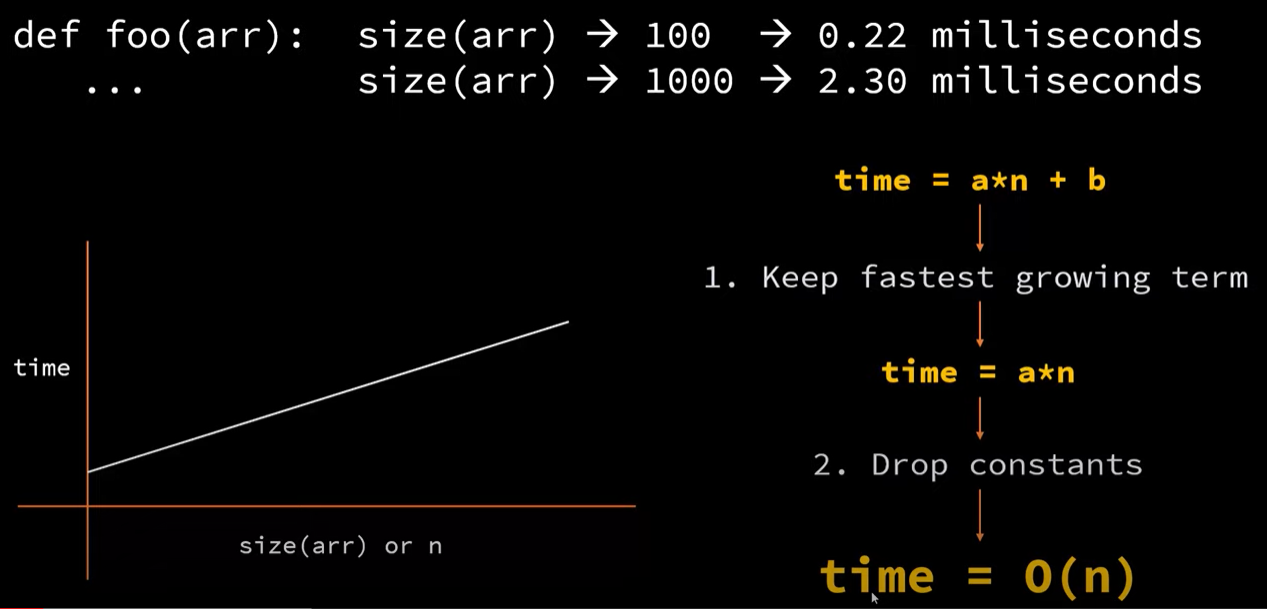
To arrange data in memory for efficient usage.

Ds are containers storing data in a specific memory layout.

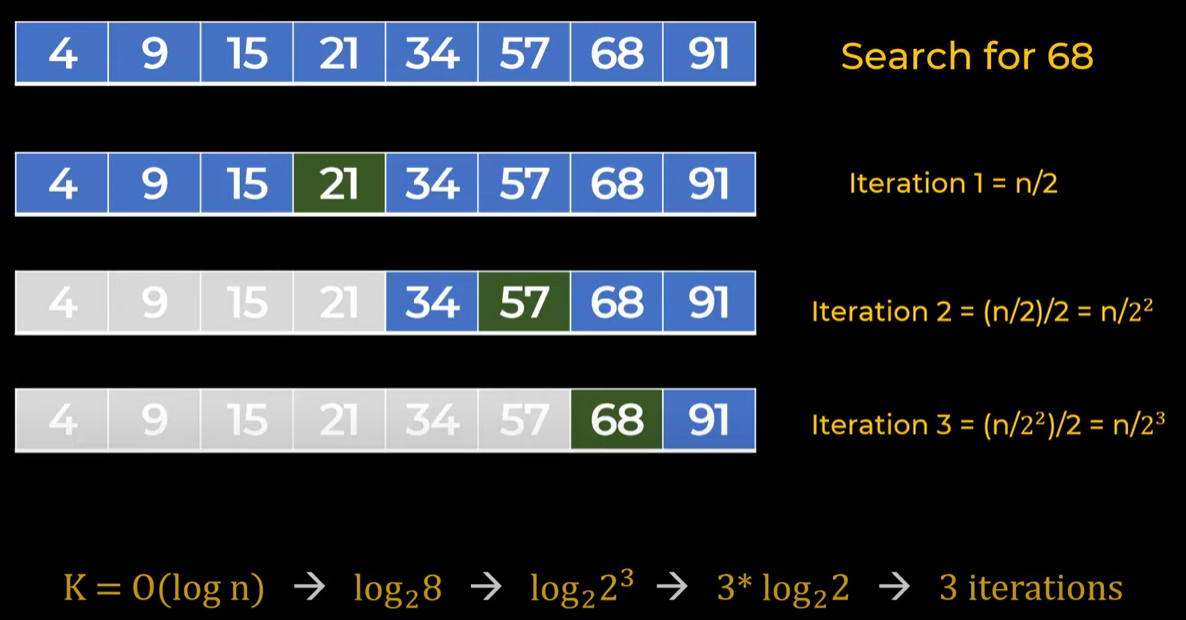


**Big O notations:**

Used to measure how **Running time** or **Space requirement** for your program grows as input size grows.



Binary Search Big O notation.



**Array:**

Most used Data Structures in programming language.

1. Get particular index value, a[2] 🡪 O(1)
2. Search particular value 🡪 O(n)
3. Print all the values 🡪 O(n)
4. Insert a new value in List 🡪 Shift all the values by 1 🡪 O(n)
5. Delete an index 🡪 Shift all values by 1 🡪 O(n)

**Imp:**

1. Python uses **Dynamic Arrays.** (No size limit to array)
2. Arrays are stored in continuous memory location.
3. Dynamic Arrays Uses more memory than static array (**ex:** Initial size is 5, then to add a sixth element it will reserve new [ 5 + 5\*2 memory space])
4. Python list are Heterogenous, can store multiple data types.

**Linked List:**

Link to the **next** address.

Link list consists of **Node,** each node consists of value & address of next node.

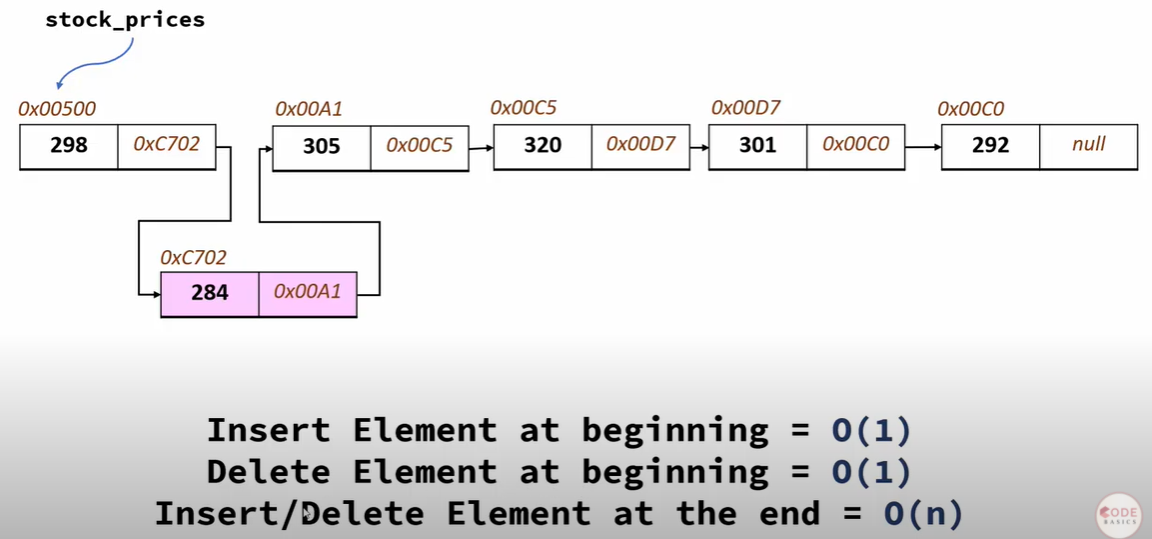
**Issues with Array:**

Insertion in Array 🡪 O(n)

**Advantages of Linked list over Arrays:**

No need to pre-allocate space.

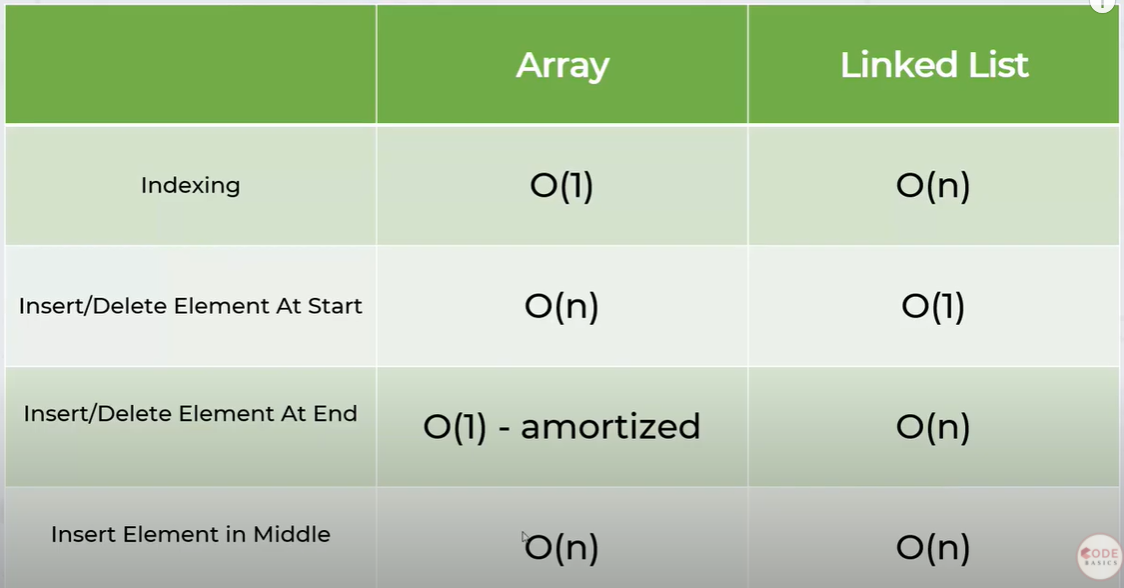
Insertion is easier.



**Double Linked List:**

Link to the **next as well previous** address.

Reverse traversal is also very easy.



**Hash Maps/ Hash Table:**

* hashmap = key-value pairs.
* No duplicate keys
* O(1) for add, get, delete fuctions.
* Also called dictionary, map, hash table, associative array

Components of Hash Map

* Array
* Hash function 🡪 to give index to key. E.g (‘rice’ 🡪 index(len(‘rice’)-1)
* Collision handling 🡪 When multiple keys of dict is assigned to same index.

**Stack (LIFO)**

Last in first out

Different ways to implement Stack:

1. List:
   1. Pros:
      1. By default, list behaves as Stack.
      2. List has indexing advantage
   2. Cons: List are Dynamic and memory issues are there.
   3. Implementation: Webpages visited.
2. Collections.deque
   1. Pros:
      1. Its faster than list.
      2. Can pop element from left also
      3. In backend it implements as Doubled linked list
   2. Cons:
      1. Doesn’t have indexing.

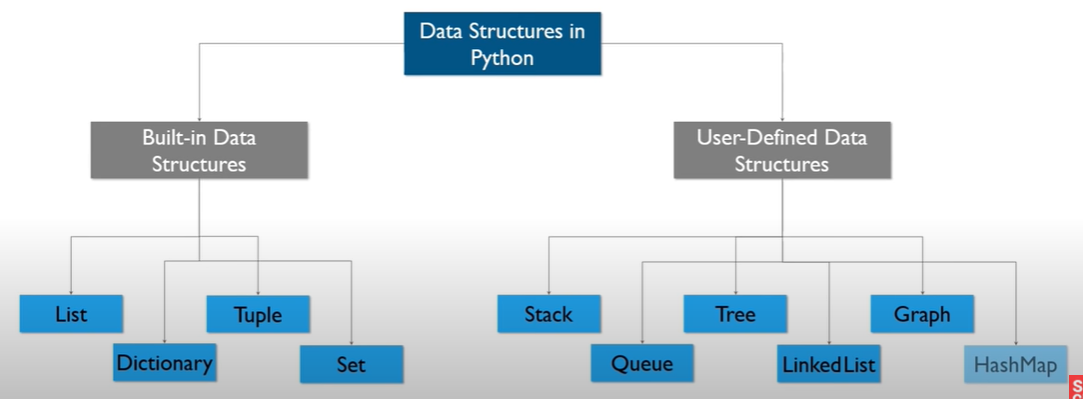
**Queue (Loose Couple) (FIFO)**

Also called producer consumer architecture

Implementation

List/Collections.deque

**Data Structures in Python**



**Built in Data Structures:**

1. List
2. Tuple
3. Dictionary
4. Set

**User Defines Data Strcuture:**

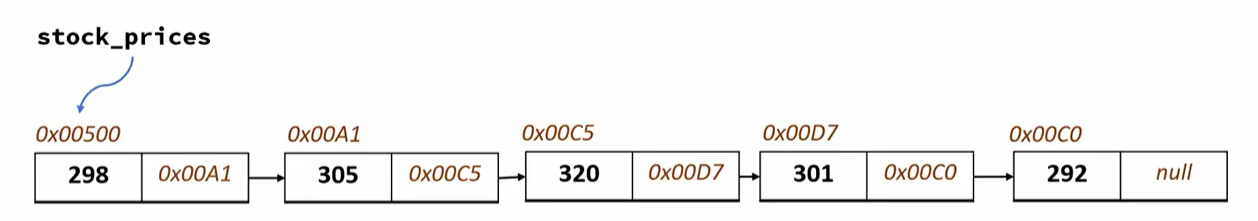
1. **Stack**
   1. **LIFO 🡪 Last In First Out**
   2. **Linear Data Structure**
   3. **Array structure (Push & Pop)**
   4. **Uses only Top location**
   5. **Used in Recursive programming, reversing words.**
   6. **Collections.deque**
2. **Queue**
   1. **FIFO 🡪 First In First Out**
   2. **Linear Data Structure**
   3. **Uses ‘Head’ and ‘Tail’ location.**
   4. Used in Job Scheduling and Traffic Conjestion
3. **Tree**
   1. **Non Linear data Structure**
   2. **Root and Node locations.**
   3. **Used for searching purpose**
4. **Linked List**
   1. **Linear Data Structure**
   2. **Used in Maps**
5. **Graphs**
   1. **To represent maps**
6. **HashMaps**
   1. **Same as Dictionary in python**

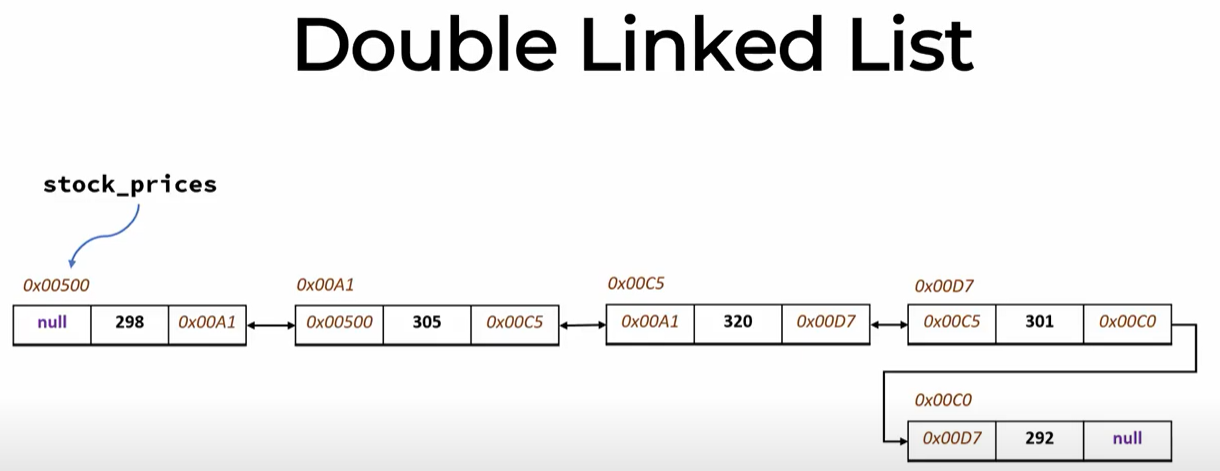
**Big O, notifications 🡪 To get the complexity of program.**

Ex: two for loops means O(n)2

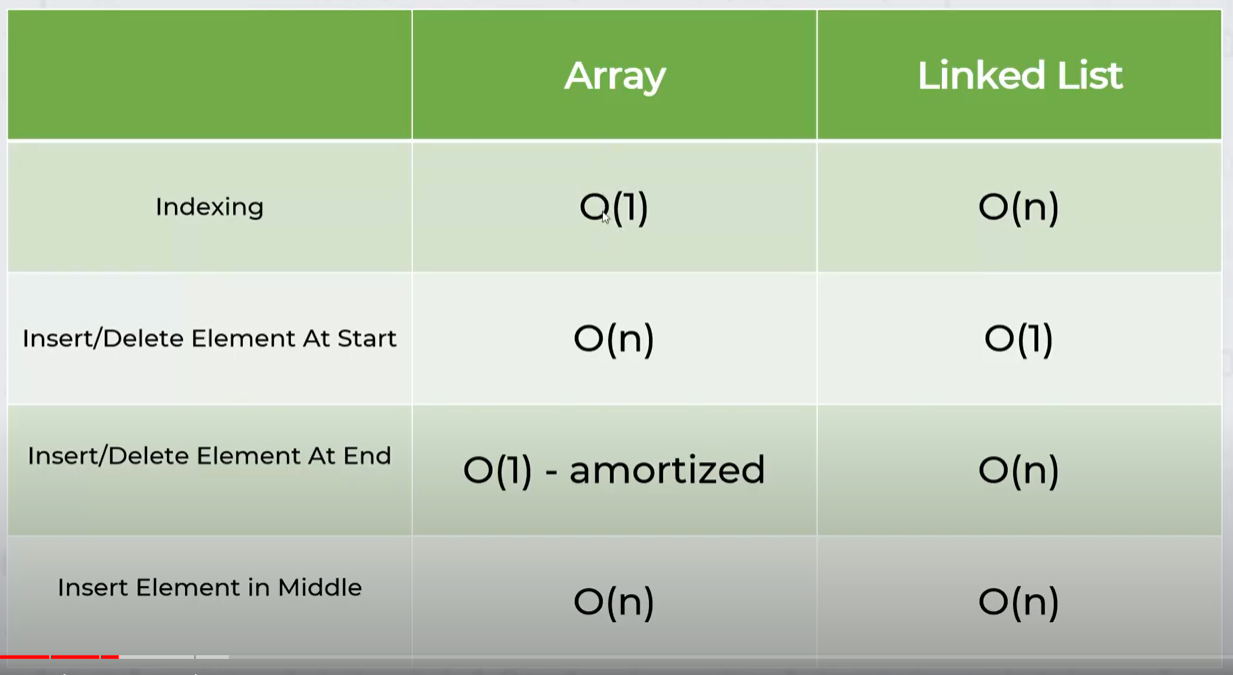
**Linked List**

1. Like List but have few advantages over list.
2. Each element not only store value but also the address of next element.





Difference between Array and List



Advantages of Linked List over Arrays:

1. Don’t need to pre allocate space.
2. Insertion is easier.

**Generators:**

**Algorithms:**

**It uses Data Structures, step wise execution.**

**Errors in Python:**

1. Compiles Errors
   1. Syntactical Errors.
2. Logical Error:
   1. Not correct Output
3. Run time Error
   1. Mistake is done by user (File is missing, internet is not there)

**Multi threading & multi Tasking:**

When we break down a big task into smaller tasks and execute tasks parallel.

* Now days we have multicore CPU’s
* With multiple core we can run multiple tasks,

**Note:** Python is Interpreted and compiled language both. (Compilation is done at backend by PVM).

**Algorithms:**

Linear Search:

* Works for both sorted and unsorted Array.

Binary Search:

* Works only for unsorted Array.

Bubble Sort (Sinking Sort):

* Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.
* In every iteration the last element will be the biggest one.
* We check the adjacent element e.g. (n with n-1)
* Total iterations: (n-1)!

Selection Sort:

* Searches for smallest element in the array and then places it to the first location in array. Similarly will start from 2nd position and searches for next smallest element.
* Check one value with all others and place the min value at first location.

Insertion Sort:

* Builds the final sorted list one item at a time.

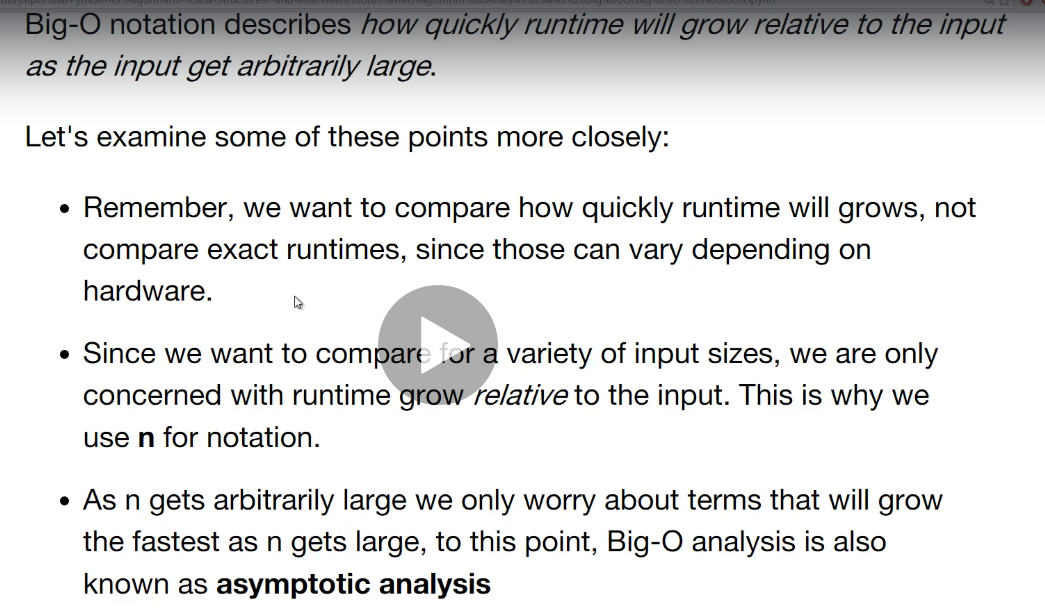
**Note:** Insertion Sort is similar to Selection sort. The only difference is insertion just insert and selection compares all the elements before inserting.

Shell Sort :

* Uses GAP and selection sort.
* Reduces the Gap in each iteration.

Quick Sort (Partition Exchange Sort)

Divide, conquer and combine



Diff between Binary Tree and Binary Search Tree.

**Polymorphism in Python:**

**Poly** 🡪 Many, **Mor** 🡪 Forms (Single object many forms.)

Four ways of implementing Polymorphism in Python:

1. Duck Typing
2. Operator Overloading
3. Method Overloading
4. Method Overriding