Algorithms and Data structures: -

[a ] 1. Linked list :   
[ b] Node. Nodes are put in chains, it makes linked lists.   
[ c] Values. Nodes contain Values. (.value) =3, property.   
[d ] Pointer, (links to the next node). Acess pointers with (.next) property.   
[e ] First node in the list is head, then head.next then head.next.next. Last node with no next pointer is called tail.

[f] Interface: This is how we interact with a data structure. Done with **Properties** and **Methods**.

,next or .value are properties.

Functions storerd on an object are Methods. Like adding value to the list like

head.addToTail (8) : add to tail a node with a value 8.

head.getValue (3) ; get value of Node index no 3 the 4th : .

[g] Implementation : How we code Data structures and methods.

Nodes are implemented as class in OOP in Python. Classes produces objects. One Node class can create many Nodes.

[ ] Pors - Adding (new) items or dealiting items.  
[ ] Cons - Not good at retrieving even with index known or not good for searching.

[ ] 2. Array-[  ]  
[ ] Continued block of sell in computer memory.  
[ ] Pros - Good at retrieving items. Like adding.  
[ ] Cons - For high level languages, Adding may increase the size of memory blocks into other computer memory blocks.  For lower level languages have to declear the size of the Array in advance.

[ ] 3. Hash Table- {  }  
[ ] Dictionary in python. Key value pair. Like array. Array key goes through hashing function, to give the computer noncontinuous  memory location anywhere. Not to worry about increasing size.  
[ ] Pro - Good at retrieving and adding.  
[ ] Depending upon the hashing algorithom,  two keys can hash to same memory location, collision. Can be resolved.  
[ ] Con - key memory Collision.

[ ] 4. Stack and Que -  
[ ] Call stack (((.  ))).. DFS Def for search algorithm.  
[ ] Stack - Last in./.push(1)  in the top; first out. /. pop() off the top.  
[ ] Que - like a line, first in first out.  
[ ] .enqueing(1)  
[ ] .dequeuing()  
[ ] BFS breath for search algorithm.   
[ ] Pros - Efficient add and remove  
[ ] Cons - limited use cases.

[ ] 5. Graphs and Trees -  
[ ] Graph theory. Like linked lists.  Pointers are called "edeges" that can have "weights" with numbers assigned to them.   
[ ] Two cities connection rode is the edge, and the length of the rode is the weight. Example.  
[ ] Trees are graphs which grows in one direction. Like family tree. Parent children tree, html tree with nested elements.  
[ ] Binary search trees - Can search efficiently Rules - Eaxh node can have maximum 2 children, left or right. Left must be less than the node and right must be more than the node.  
[ ] But BST can get unbalanced.