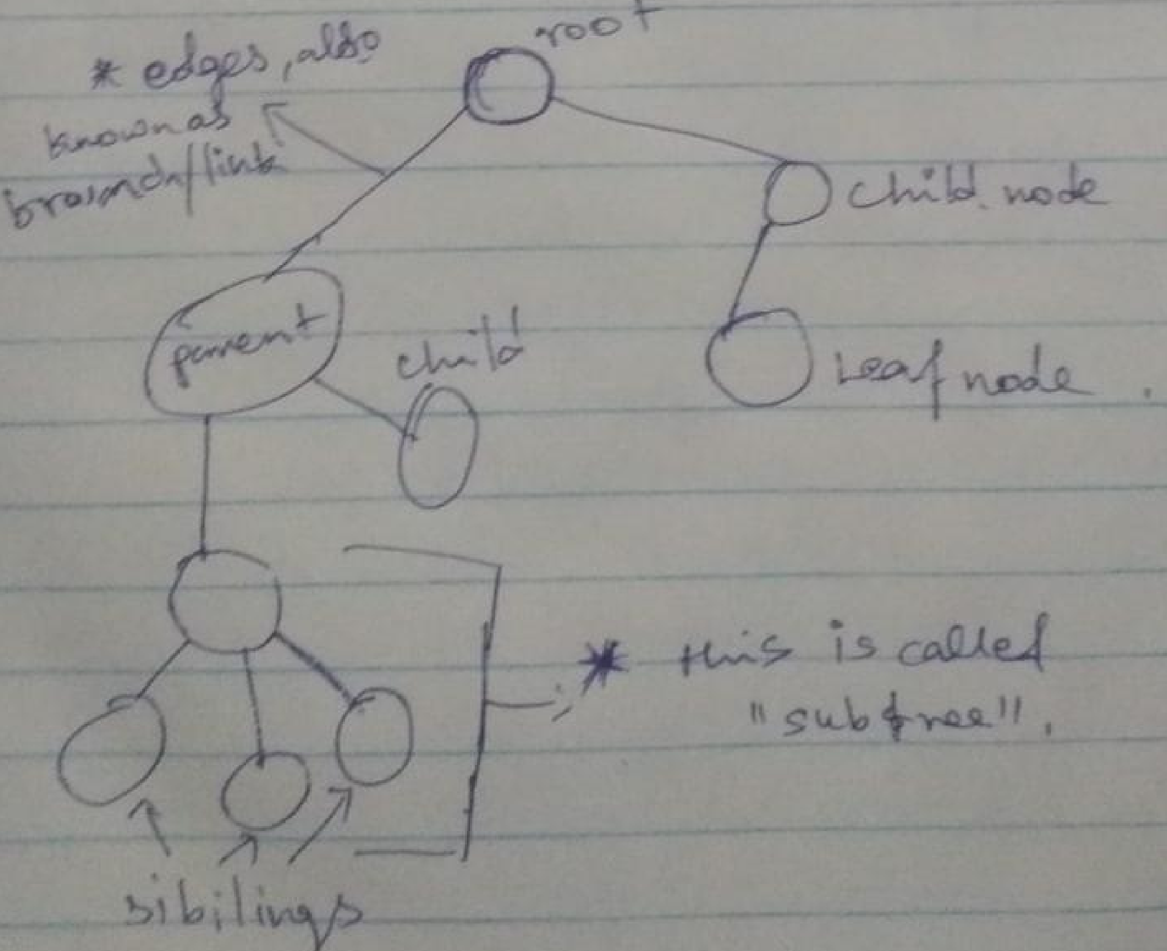
# DAY -9

## Binary Tree

Tree :- ⇒ A non-linear datastructure that ⇒ represent nodes connected by edges.

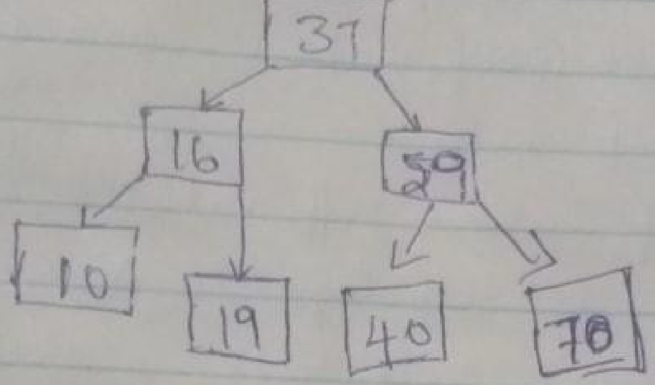
Binary Tree ⇒ Every node has at most two children (left and right). the "root" of the tree is on top.
→ Every node has parentnode above.



root
* edges, also known as branch/link
child node
parent
child
Leaf node.
* this is called "subtree".
sibilings

## Implementation :-

An Example:-



* A node's left child must have a value less than its parent's value, and the node's right child must have a value greater than its parent value.

## Insertion:

compares ~~to~~ with parent node's value to choose where to add (right or left.)

*as told above if less -> left or else -> right side.

## Searching:-

=> need to traverse the node lef to right and with the parent node.

```
class Node:
    def __init__(self, data):
        self.left = None
        self.right = None
        self.data = data
    def insert (self, data):  # used to
                              # create nodes
        if self.data:
            if data < self.data:
                if self.left is None:
                    self.left = Node(data)
                else:
                    self.left.insert(data)
            elif data > self.data:
                if self.right is None:
                    self.right = Node(data)
                else:
                    self.right.insert(data)
        else:
            self.data = data
    def searchVal(self, val)  # compare
        if val < self.data:   # and find.
            if self.left is None:
                return "Not found"
            return self.left.searchval(val)
        elif val > self.data:
            if self.right is None:
                return "Not found"
            return self.right.searchval(val)
```

```
else :
    return  str(self.data) +" is there "

def PrintTree (self):
    if self.left :
        self. left .PrintTree( )
    print (self, data)
    if self. right :
        self .right .PrintTree( ).
root = Node(37 )
root .insert (16)
root. insert(59)
        ...   [* I have  mentioned the
diagram  above*]
```