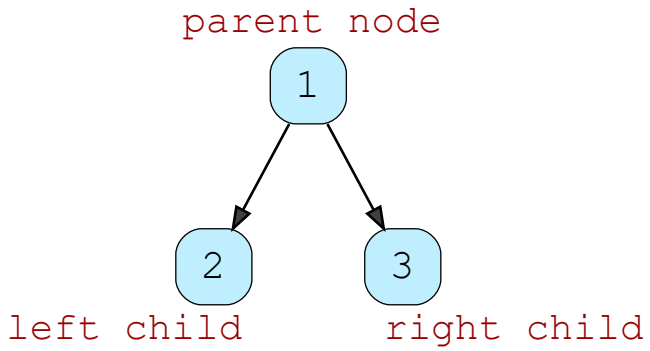
# Introduction

In this lesson, you will be introduced to Binary Trees and their implementation in Python.

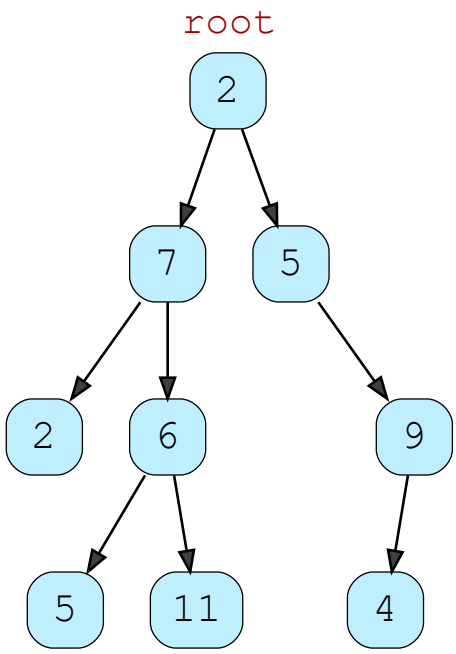
WE'LL COVER THE FOLLOWING



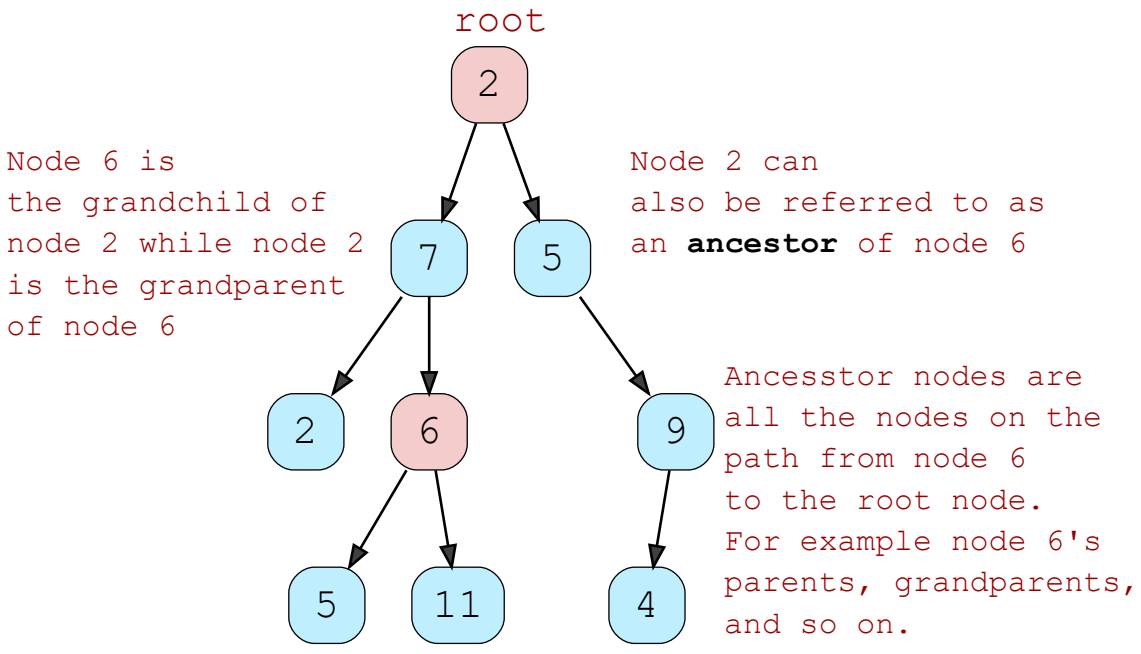
A **binary tree** is a tree data structure in which each node has at most two children, which are referred to as the *left child* and the *right child*. Have a look at an elementary example of a binary tree:



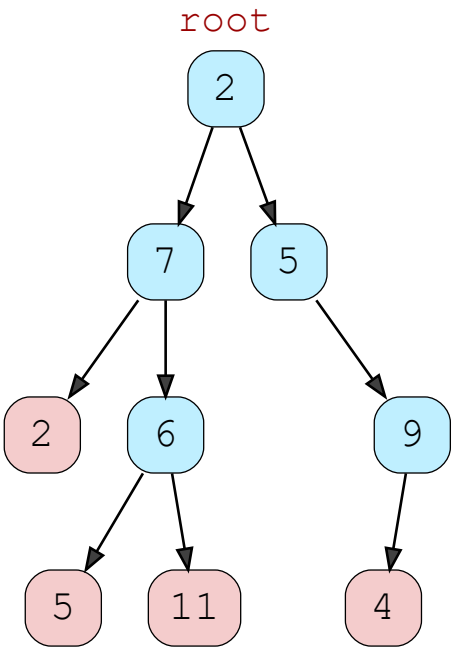
Here is another example of a binary tree that introduces us to other related terminologies:




**1** of 3




**2** of 3

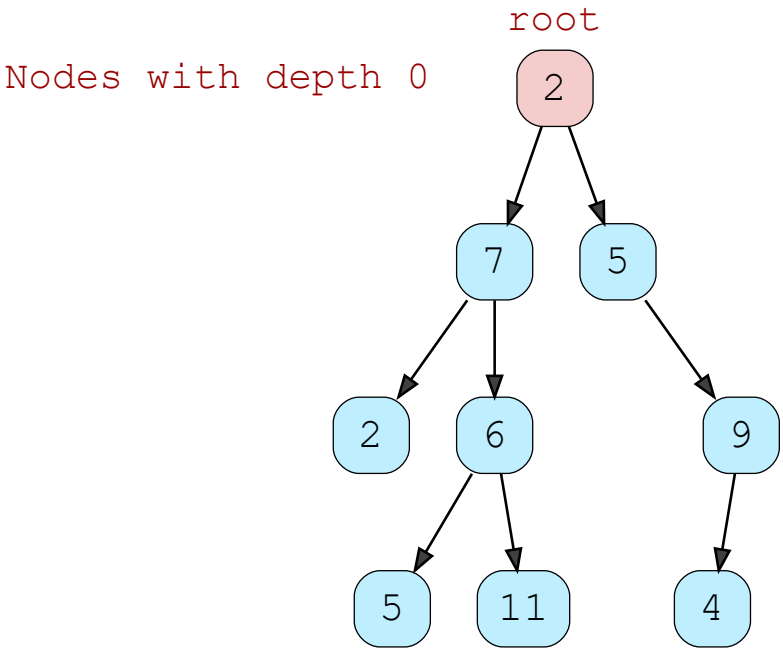



Leaf Nodes : Nodes without any children

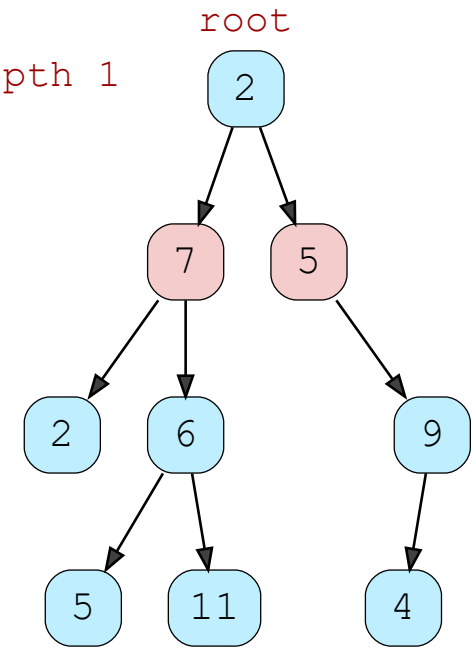


## Depth of a Node #

The length of the path from a node, $n$, to the root node. The depth of the root node is 0.

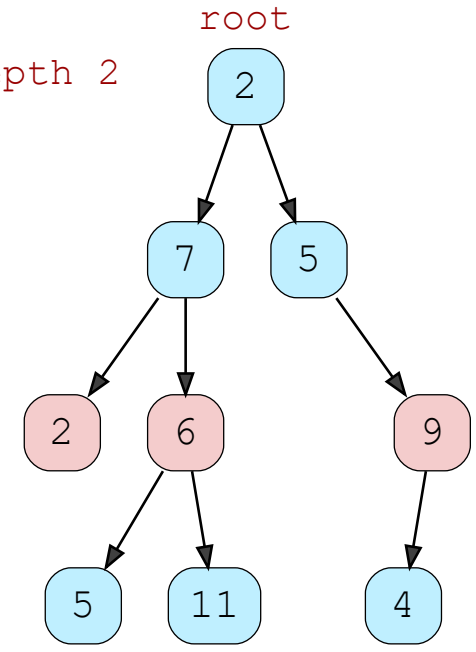


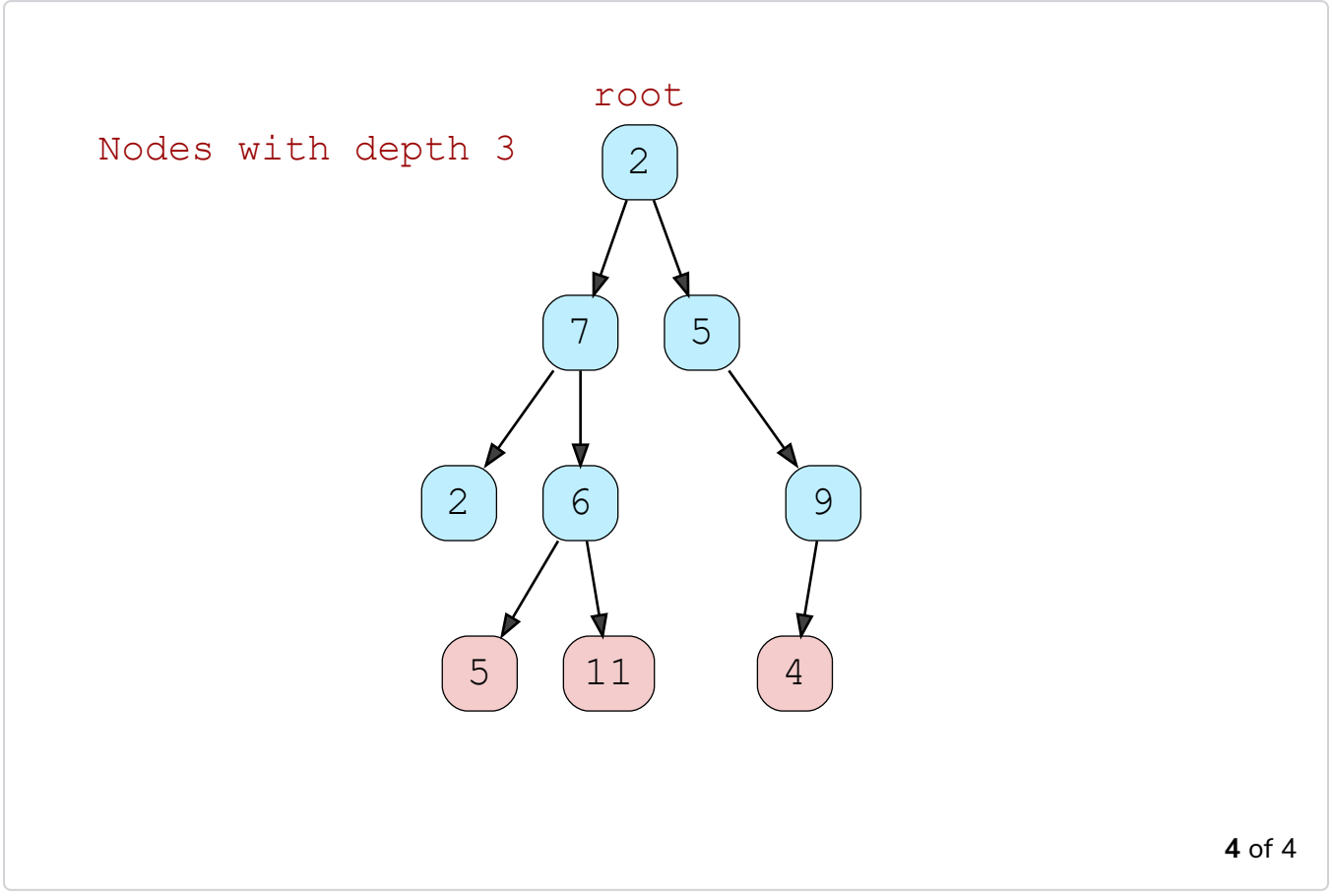
Nodes with depth 1






Nodes with depth 2

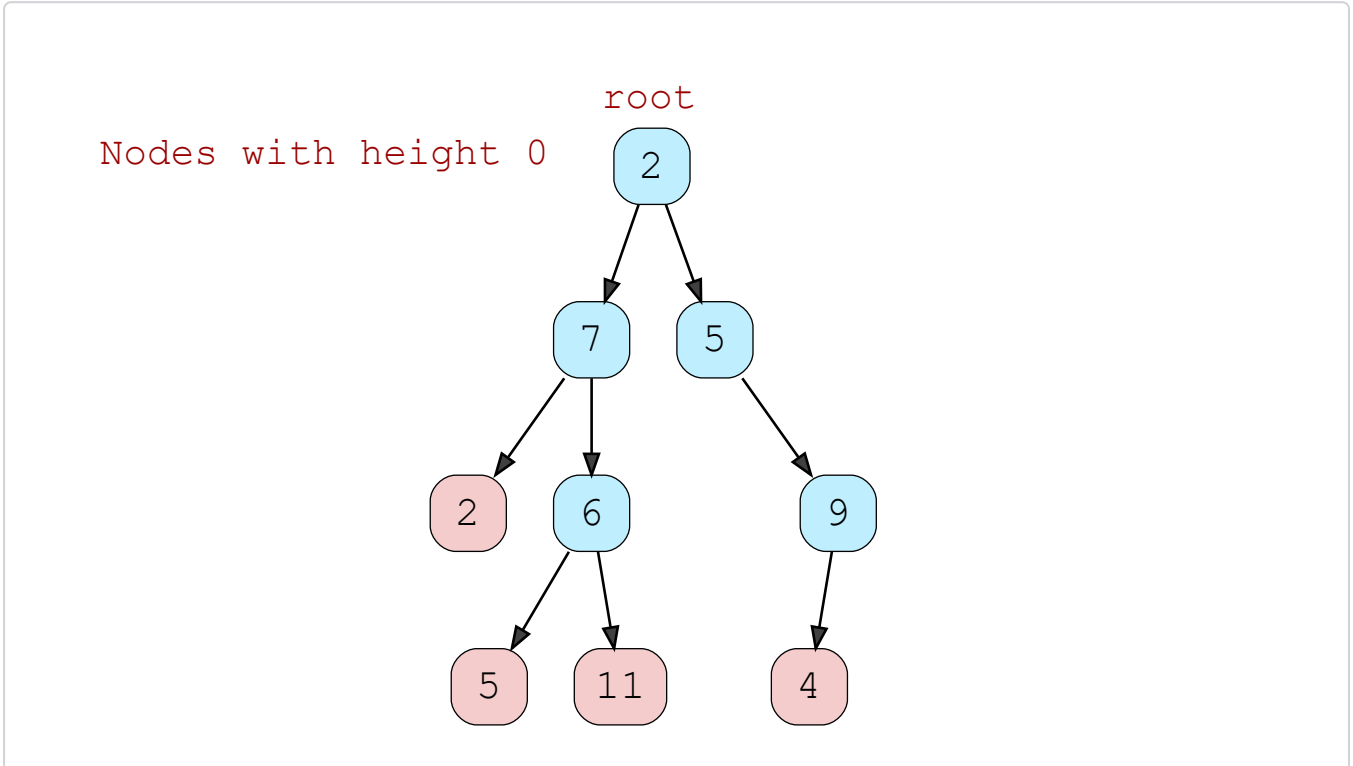




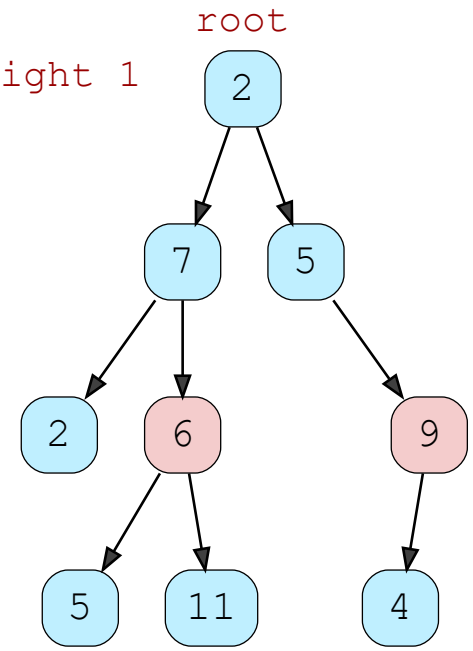



## Height of a Tree #

The length of the path from *n* to its deepest descendant. The height of the tree itself is the height of the root node, and the height of leaf nodes is always 0.

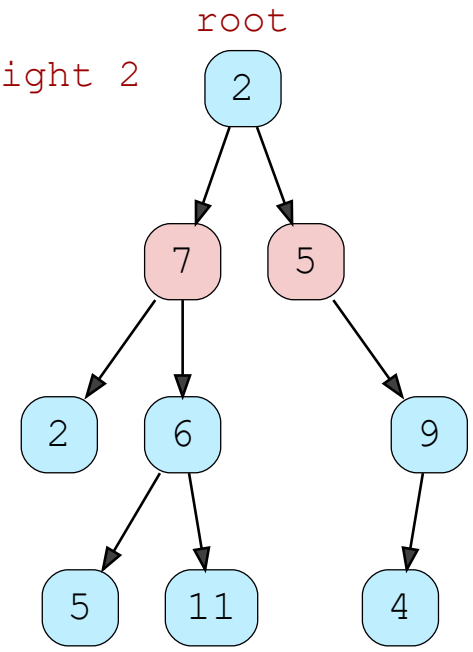


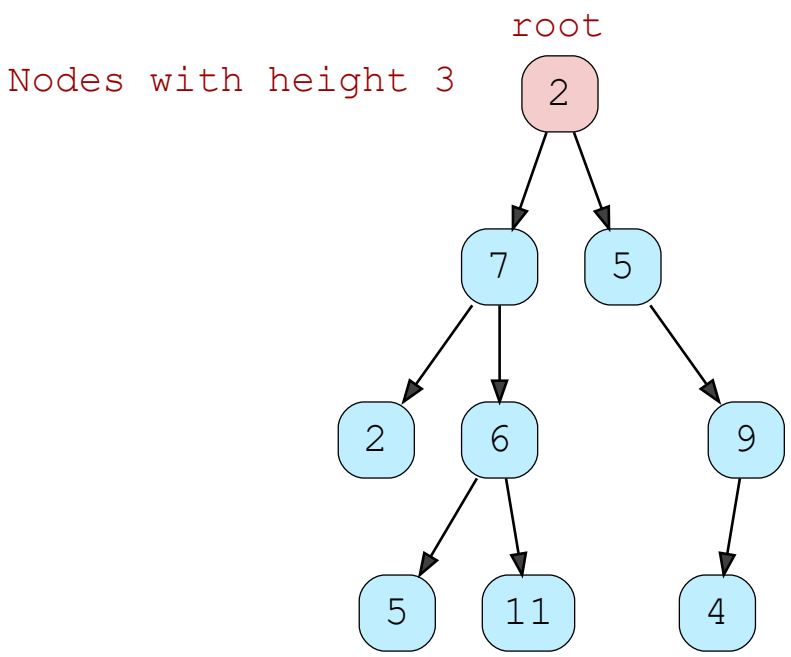
Nodes with height 1



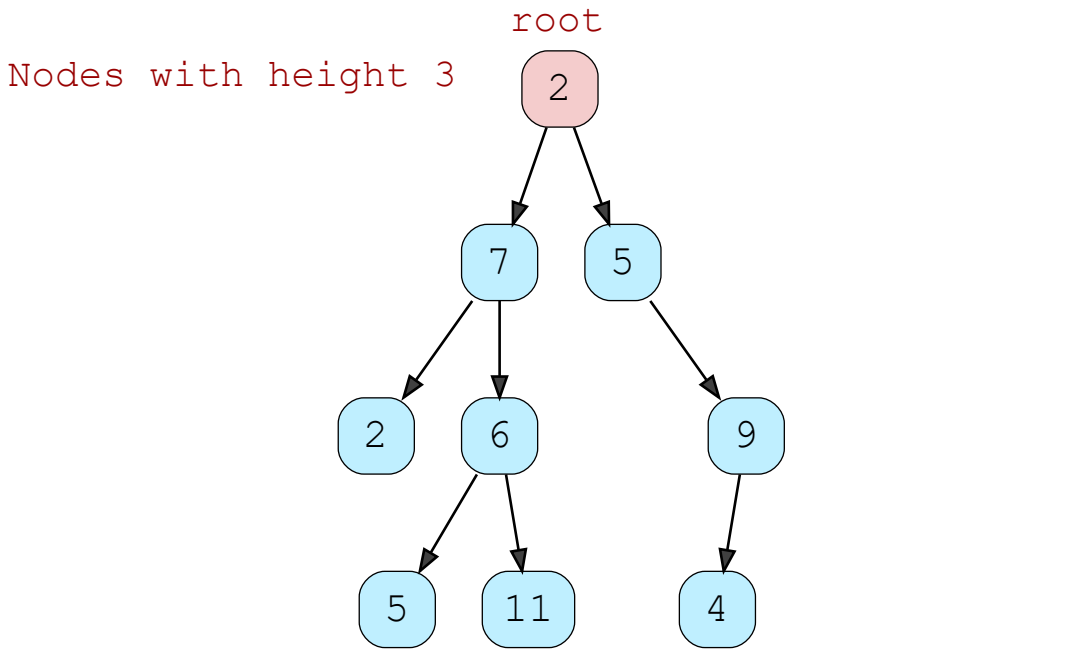



Nodes with height 2






**4** of 5

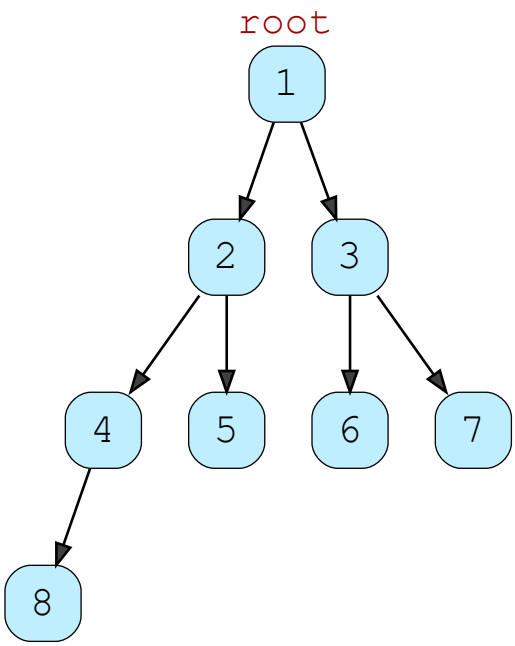



**5** of 5

# Types of Binary Trees #

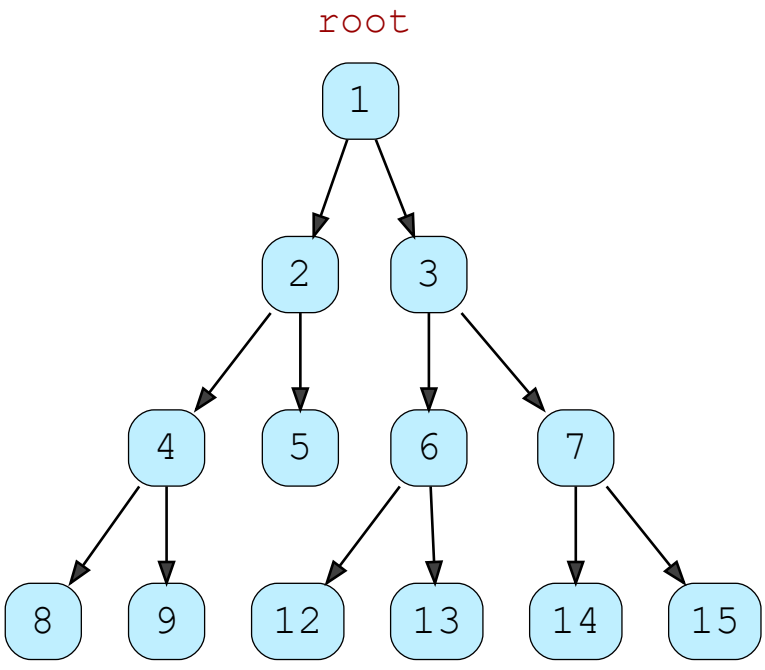
## Complete Binary Tree #

In a **complete** binary tree, every level *except possibly the last*, is completely filled and all nodes in the last level are as far left as possible.




## Full Binary Tree #

A **full** binary tree (sometimes referred to as a **proper** or **plane** binary tree) is a tree in which every node has either 0 or 2 children.



## Implementation #

To implement a binary tree in Python, we will first implement the `Node` class.

```
class Node(object):
  def __init__(self, value):
    self.value = value
    self.left = None
    self.right = None
```

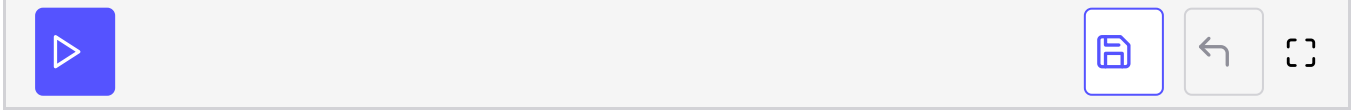
In the code above, we have defined the `Node` class with a “new” style of defining classes in Python. The `Node` class has three attributes:

1. `self.value`
2. `self.left`
3. `self.right`

`self.value` will be equal to the `value` passed to the constructor while `self.left` and `self.right` will contain the nodes which will be the left and the right child of this node.

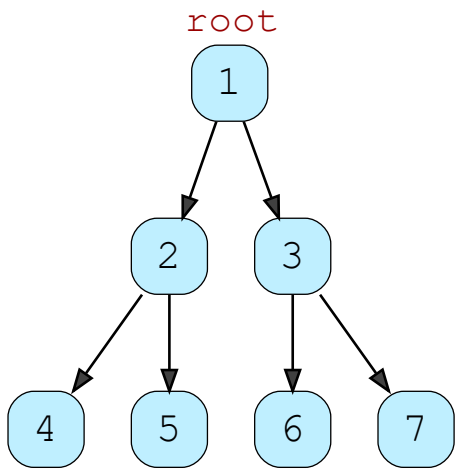
Let’s go ahead and implement `BinaryTree` class:

```
class Node(object):
  def __init__(self, value):
    self.value = value
    self.left = None
    self.right = None

class BinaryTree(object):
  def __init__(self, root):
    self.root = Node(root)


tree = BinaryTree(1)
tree.root.left = Node(2)
tree.root.right = Node(3)
tree.root.left.left = Node(4)
tree.root.left.right = Node(5)
tree.root.right.left = Node(6)
tree.root.right.right = Node(7)
```



The `BinaryTree` contains `root`, which is an object of the `Node` class and contains the value passed as `root` in the constructor of `BinaryTree` class.

From **lines 12-18**, we construct an object from the `BinaryTree` class and populate the tree by creating nodes. The visual representation of that tree will be as follows:




The implementation of the Binary Tree was pretty straightforward. At this point, we need some way to traverse the tree and visualize through code in Python. In the next lesson, we will go over some of the tree traversal methods.