Data Structures

Lecture 13

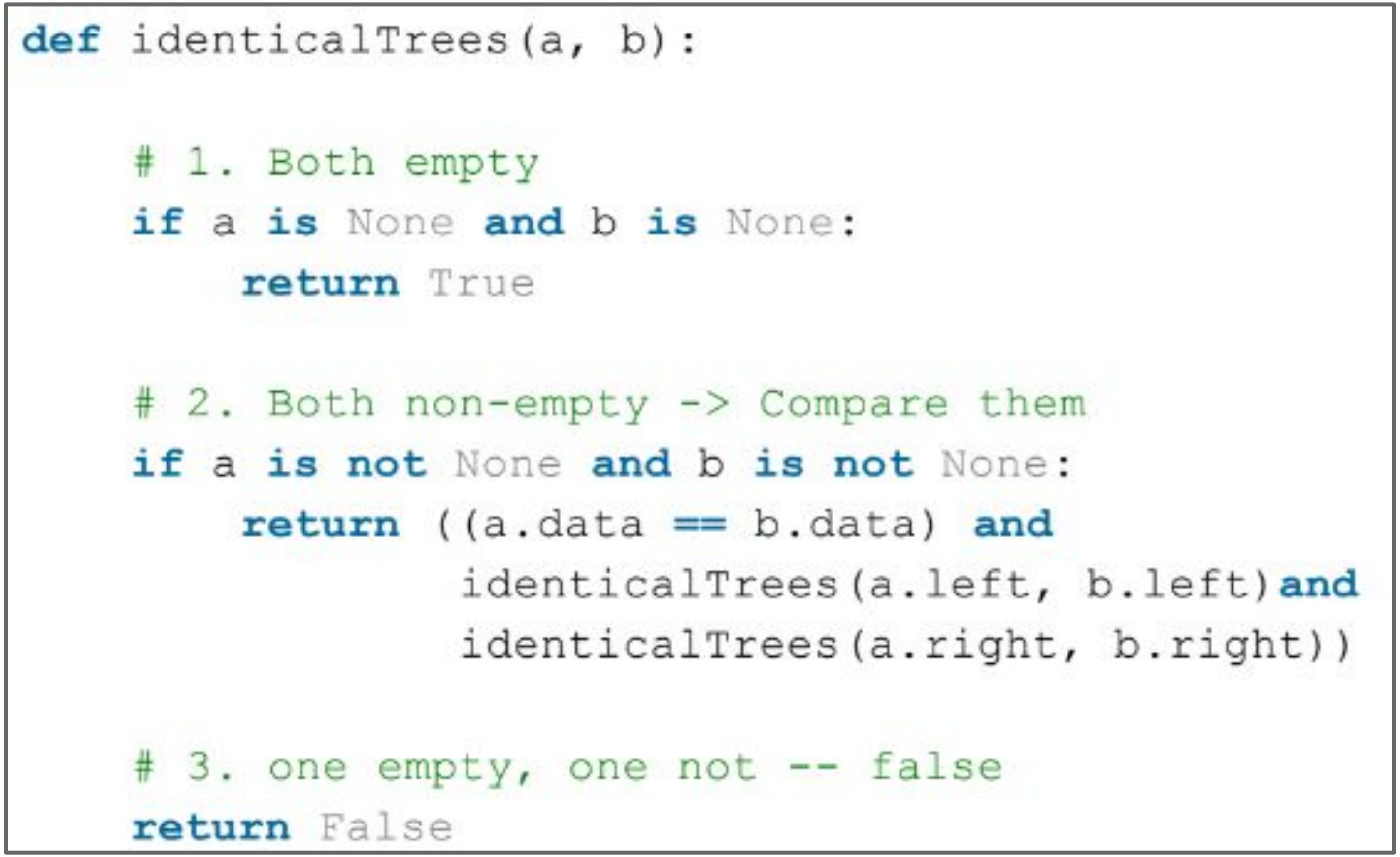
Tree Problems

Credit: Prantik Paul, Lecturer, BRAC University

Tree Problems

Check if Two Binary Trees are Identical

Tree Problems

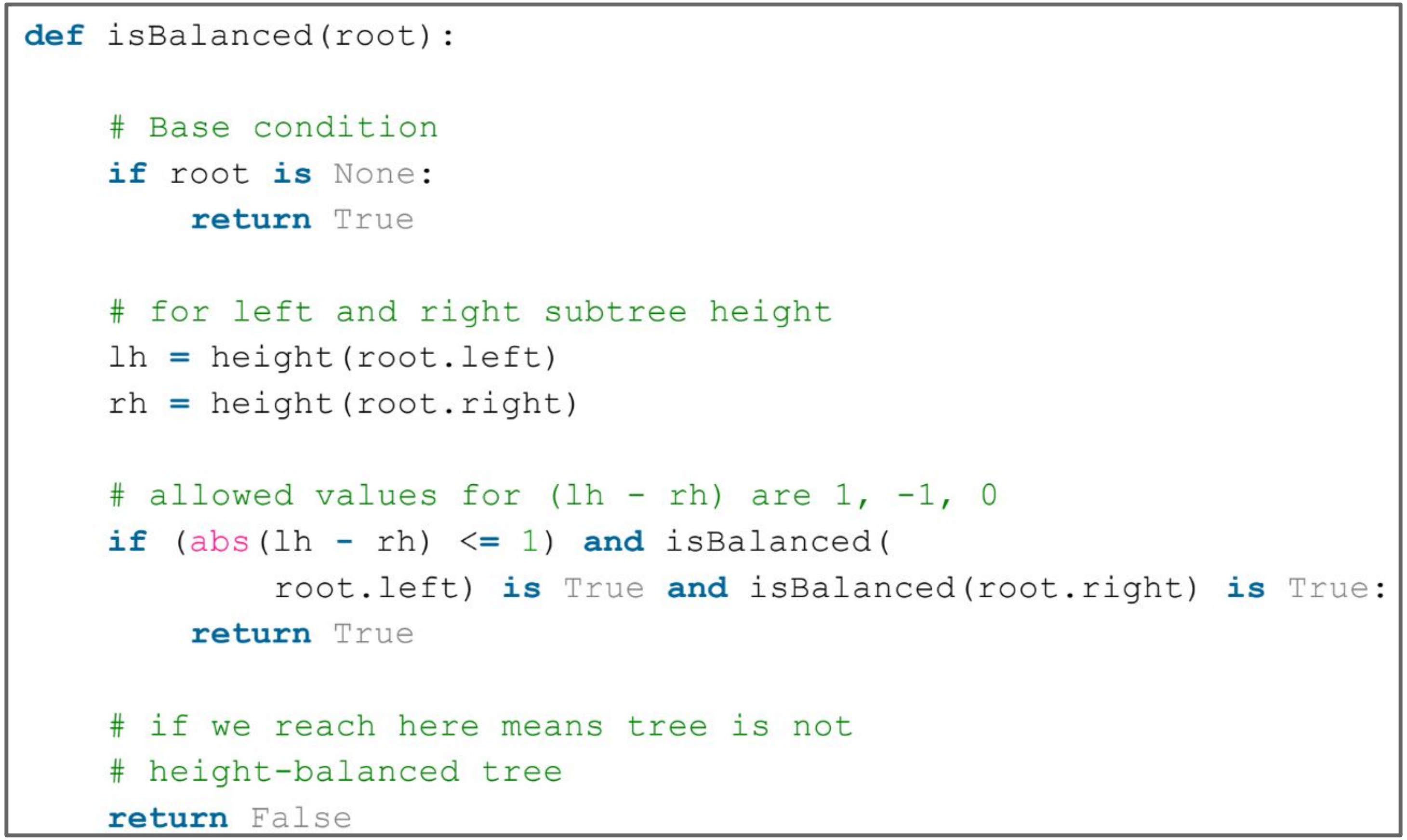


```python
def identicalTrees(a, b):

    # 1. Both empty
    if a is None and b is None:
        return True

    # 2. Both non-empty -> Compare them
    if a is not None and b is not None:
        return ((a.data == b.data) and
                identicalTrees(a.left, b.left)and
                identicalTrees(a.right, b.right))

    # 3. one empty, one not -- false
    return False
```

Tree Problems

Check if a Binary Tree is Balanced

Tree Problems

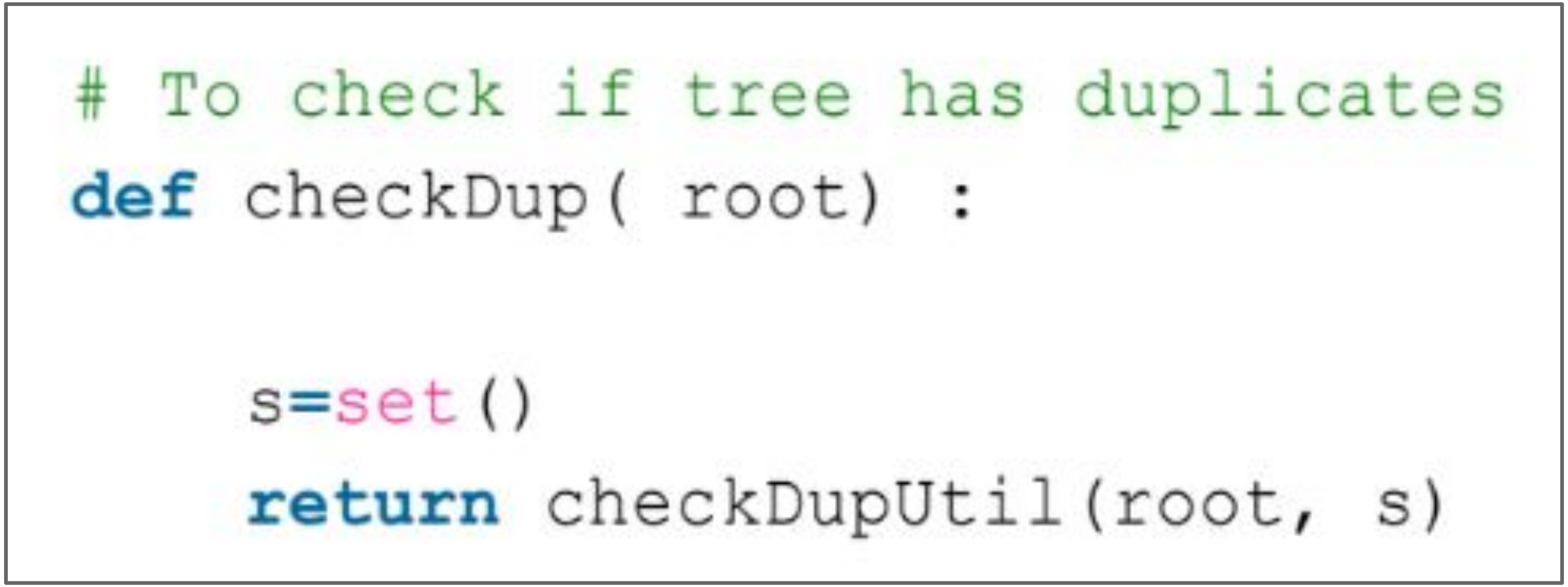


```python
def isBalanced(root):

    # Base condition
    if root is None:
        return True

    # for left and right subtree height
    lh = height(root.left)
    rh = height(root.right)

    # allowed values for (lh - rh) are 1, -1, 0
    if (abs(lh - rh) <= 1) and isBalanced(
            root.left) is True and isBalanced(root.right) is True:
        return True

    # if we reach here means tree is not
    # height-balanced tree
    return False
```

Tree Problems

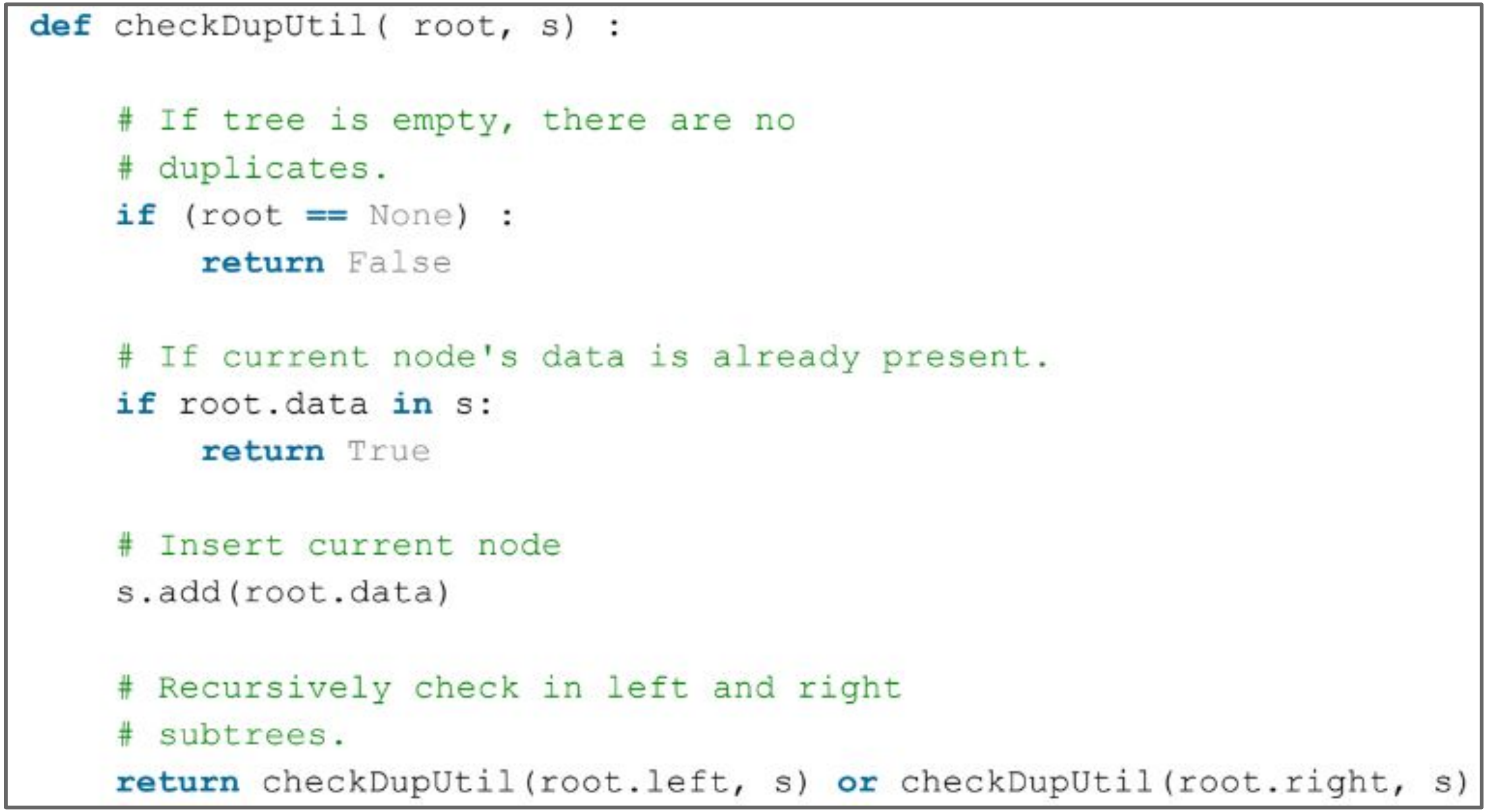
Check if a Binary Tree has Duplicate Elements

Tree Problems



```python
# To check if tree has duplicates
def checkDup( root) :


    s=set()
    return checkDupUtil(root, s)
```

Tree Problems



```python
def checkDupUtil( root, s) :

    # If tree is empty, there are no
    # duplicates.
    if (root == None) :
        return False

    # If current node's data is already present.
    if root.data in s:
        return True

    # Insert current node
    s.add(root.data)

    # Recursively check in left and right
    # subtrees.
    return checkDupUtil(root.left, s) or checkDupUtil(root.right, s)
```

Tree Problems

Find the Unique Elements in a Binary Tree

Tree Problems

Find the Duplicate Elements in a Binary Tree

Tree Problems

Find Occurance of an Element in a Binary Tree

Tree Problems

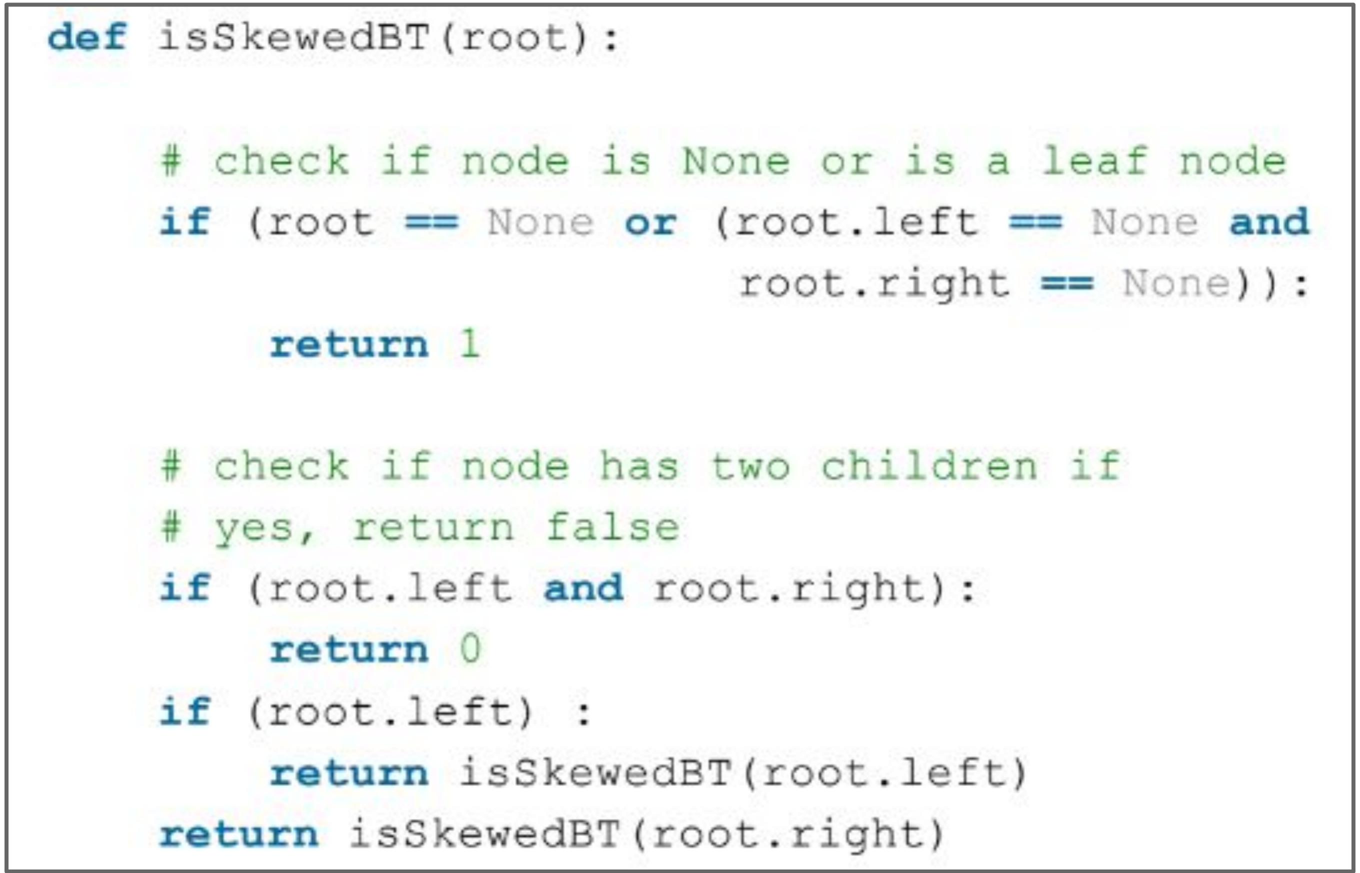
Find Occurance of All Elements in a Binary Tree

Tree Problems

Check if a Binary Tree is Skewed

each node has only one child node or none.

Tree Problems



```python
def isSkewedBT(root):

    # check if node is None or is a leaf node
    if (root == None or (root.left == None and
                         root.right == None)):
        return 1

    # check if node has two children if
    # yes, return false
    if (root.left and root.right):
        return 0
    if (root.left) :
        return isSkewedBT(root.left)
    return isSkewedBT(root.right)
```

Tree Problems

Print all the Full Nodes of a Binary Tree

both left and right children are non-empty.

Tree Problems

Print all the Nodes with 1 Child of a Binary Tree

Tree Problems

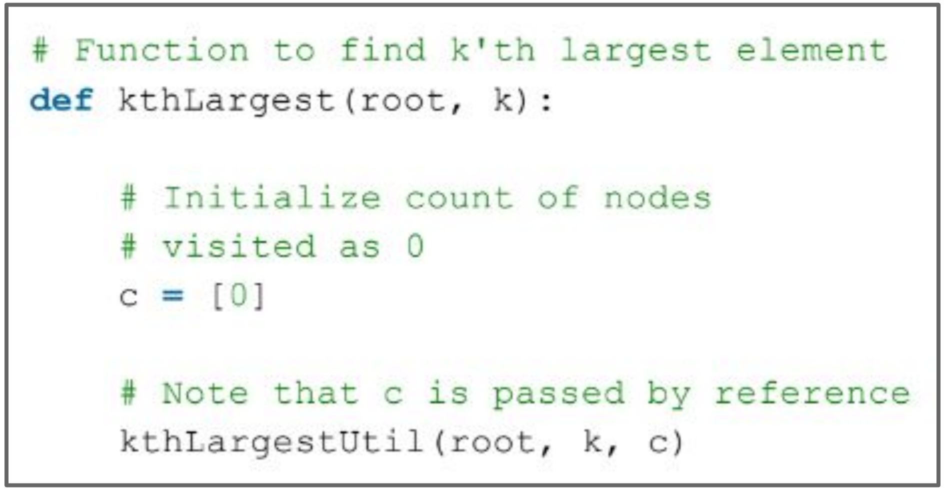
Given a BST Print Elements in Descending Order

Do yourself

Tree Problems

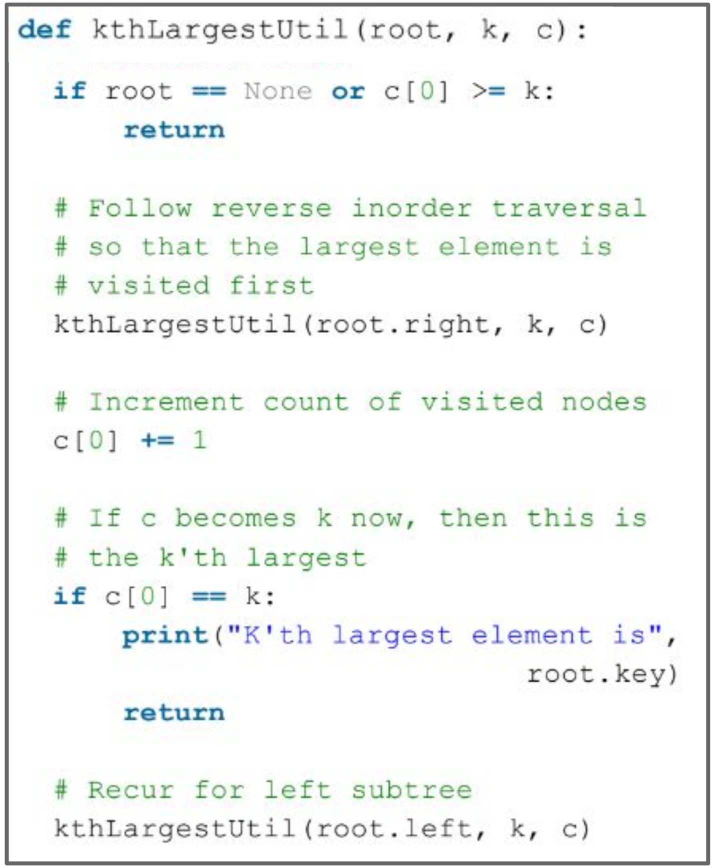
Given a BST find Kth Largest Element

Tree Problems



```
# Function to find k'th largest element
def kthLargest(root, k):

    # Initialize count of nodes
    # visited as 0
    c = [0]

    # Note that c is passed by reference
    kthLargestUtil(root, k, c)
```

Tree Problems



```python
def kthLargestUtil(root, k, c):

    if root == None or c[0] >= k:
        return

    # Follow reverse inorder traversal
    # so that the largest element is
    # visited first
    kthLargestUtil(root.right, k, c)

    # Increment count of visited nodes
    c[0] += 1

    # If c becomes k now, then this is
    # the k'th largest
    if c[0] == k:
        print("K'th largest element is",
                                  root.key)
        return

    # Recur for left subtree
    kthLargestUtil(root.left, k, c)
```