Count BST nodes that lie in a given range

Given a Binary Search Tree (BST) and a range **l-h(inclusive)**, count the number of nodes in the BST that lie in the given range.

- The values smaller than root go to the left side
- The values greater and equal to the root go to the right side

**Input:
** 10
/
5 50
/ /
1 40 100
l = 5, h = 45
Output: 3 **Explanation:** 5 10 40 are the node in the range

```
def getCount(root,low,high):
    ##Your code here
    count = [0]
    seen = set()
    helper(root,count,low,high,seen)
    return count[0]

def helper(root,count,low,high,seen):
    if root:
        if low<=root.data<=high:
            if root.data not in seen:
                seen.add(root.data)
                count[0] = count[0]+1
        helper(root.left,count,low,high,seen)
        helper(root.right,count,low,high,seen)
```

Approach2:

```
class newNode:

    # Constructor to create a new node
    def __init__(self, data):
        self.data = data
```

```
        self.left = None
        self.right = None

# Returns count of nodes in BST in
# range [low, high]
def getCount(root, low, high):

    # Base case
    if root == None:
        return 0

    # Special Optional case for improving
    # efficiency
    if root.data == high and root.data == low:
        return 1

    # If current node is in range, then
    # include it in count and recur for
    # left and right children of it
    if root.data <= high and root.data >= low:
        return (1 + getCount(root.left, low, high) +
                    getCount(root.right, low, high))

    # If current node is smaller than low,
    # then recur for right child
    elif root.data < low:
        return getCount(root.right, low, high)

    # Else recur for left child
    else:
        return getCount(root.left, low, high)

# Driver Code
if __name__ == '__main__':

    # Let us construct the BST shown in
    # the above figure
    root = newNode(10)
    root.left = newNode(5)
    root.right = newNode(50)
    root.left.left = newNode(1)
    root.right.left = newNode(40)
    root.right.right = newNode(100)
```

```
# Let us constructed BST shown in above example
#     10
#     / \
# 5     50
# /     / \
# 1     40 100
l = 5
h = 45
print("Count of nodes between [", l, ", ", h,"] is ",
                                    getCount(root, l, h))
```