Maximum Depth of Binary Tree

Question: Given a binary tree, find its maximum depth.

The maximum depth is the number of nodes along the longest path from the root node down to the farthest leaf node.

Solutions:

```
class TreeNode:
    def __init__(self, x):
        self.val = x
        self.left = None
        self.right = None


class Solution:
    # @param root, a tree node
    # @return an integer
    def maxDepth_recursive(self, root):
        if root == None:
            return 0
        return max(self.maxDepth(root.left),self.maxDepth(root.right))+1


    def maxDepth_interative(self, root):
        if root == None:
            return 0
        nodeStack = [root];
        depthStack = [1];
        maxDepth = 0;
        while len(nodeStack)>0:
            node = nodeStack.pop();
```

```
            depth = depthStack.pop();
            maxDepth = maxDepth if maxDepth > depth else depth
            if node.left != None:
                nodeStack.append(node.left)
                depthStack.append(depth+1)
            if node.right != None:
                nodeStack.append(node.right)
                depthStack.append(depth+1)
        return maxDepth

if __name__ == '__main__':
    BT, BT.right, BT.right.left = TreeNode(1), TreeNode(2), TreeNode(3)
    print ( Solution().maxDepth_interative(BT) )
```