Flatten Binary Tree

Question: Given a binary tree, flatten it to a linked list in-place.

For example:

Given

```
      1
     / \
    2   5
   / \   \
  3   4    6
```

The flattened tree should look like:

```
 1 \ 2 \ 3 \ 4 \ 5 \ 6
```

Solutions:

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

class Solution:
    # @param root, a tree node
    # @return nothing, do it in place
    def flatten(self, root):
        if root == None:        return
        stack = [root.right, root.left]
        current = root
        while len(stack) != 0:
            nextNode = stack.pop()
            if nextNode == None:
                continue
            else:
                current.left = None
                current.right = nextNode
                current = current.right
                stack.append(current.right)
                stack.append(current.left)

        return root

    def printtree(self, tree_node):
        if tree_node.left is not None:
            self.printtree(tree_node.left)
        print(tree_node.val)
        if tree_node.right is not None:
            self.printtree(tree_node.right)

if __name__ == '__main__':
    BT, BT.right, BT.right.right, BT.left, BT.left.right, BT.left.left = TreeNode(1),
TreeNode(5), TreeNode(6), TreeNode(2), TreeNode(4), TreeNode(3)
    LL = Solution().flatten(BT)
    Solution().printtree(LL)
```