```
class Stack(object):
    def __init__(self, maxsize):
        self.stack = []  # 申请的栈的数据
        self.maxsize = maxsize  # 栈的最大空间

    def push(self, data):  # 插入
        if len(self.stack) >= self.maxsize:
            print('The stack is full')
            return False
        self.stack.append(data)

    def pop(self):  # 删除
        if self.stack:
            return self.stack.pop()
        else:
            print('The stack is empty')

    def show(self):  # 显示最顶部的元素
        if self.stack:
            print("最顶上的元素是{}".format(self.stack[-1]))
        else:
            print('该栈没有元素')

    def size(self):  # 显示栈的当前大小
        print('该栈的长度为{}'.format(len(self.stack)))

    def showall(self):  # 显示栈的全部元素，从栈顶开始往下遍历
        for i in self.stack[::-1]:
            print("->{}".format(i))

    def is_empty(self):
        return not bool(self.stack)


def balanced_parentheses(string):
    '''
    :param string: 传经来的字符串
    :return:True括号成对 False括号不成对
    判断字符串的括号是否成对
    需要使用上面的class Stack
    '''
    stack = Stack(len(string))
    for j in string:
        if j == '(':
            stack.push(j)
```

```
        elif j == ')':
            if stack.is_empty():
                return False
            stack.pop()
    return stack.is_empty()


if __name__ == '__main__':
    print(balanced_parentheses('()()(())()'))
```