# 反转一个单链表

作者：selfjt
原文链接：https://ld246.com/article/1573043047676
来源网站：链滴

反转一个单链表。

# 场景

## 中文描述

反转一个单链表。

## English

Reverse a singly linked list.

输入: 1->2->3->4->5->NULL
输出: 5->4->3->2->1->NULL

# 代码示例

单链表节点

```
type ListNode struct {
    next  *ListNode
    value interface{}
}
```

单链表

```
type LinkedList struct {
    head *ListNode
}
```

单链表反转

时间复杂度O(n)

```
func (this *LinkedList) Reverse()  {
    //判断链表的长度是否有两个节点 没有两个节点就不用反转了
    if nil == this.head || nil == this.head.next || nil == this.head.next.next {
        return
    }
    var pre *ListNode = nil
    cur := this.head.next
    for cur != nil {
        temp := cur.next
        cur.next = pre
        pre = cur
        cur = temp
    }
    this.head.next = pre
}
```

打印链表

```
func (this *LinkedList) Print() {
    cur := this.head.next
    format := ""
    for nil != cur {
        format += fmt.Sprintf("%v", cur.value)
        cur = cur.next
        if nil != cur {
            format += "<-"
        }
    }
    fmt.Println(format)
}
```

测试代码

```
var l *LinkedList
//测试 初始化数据
func init() {
    n5 := &ListNode{value: 5}
    n4 := &ListNode{value: 4, next: n5}
    n3 := &ListNode{value: 3, next: n4}
    n2 := &ListNode{value: 2, next: n3}
    n1 := &ListNode{value: 1, next: n2}
    l = &LinkedList{head: &ListNode{next: n1}}
}

//go test -v -run TestLinkedList_Reverse -o linkedListAlgo_test.go
func TestLinkedList_Reverse(t *testing.T) {
    l.Print()
    l.Reverse()
    l.Print()
}
```

输出结果

```
=== RUN   TestLinkedList_Reverse
1<-2<-3<-4<-5
5<-4<-3<-2<-1
--- PASS: TestLinkedList_Reverse (0.00s)
PASS
```

## 源码

- 点击查看源码(singlelinkedlist.go)
- 点击查看源码测试方法(singlelinkedlist_test.go)