



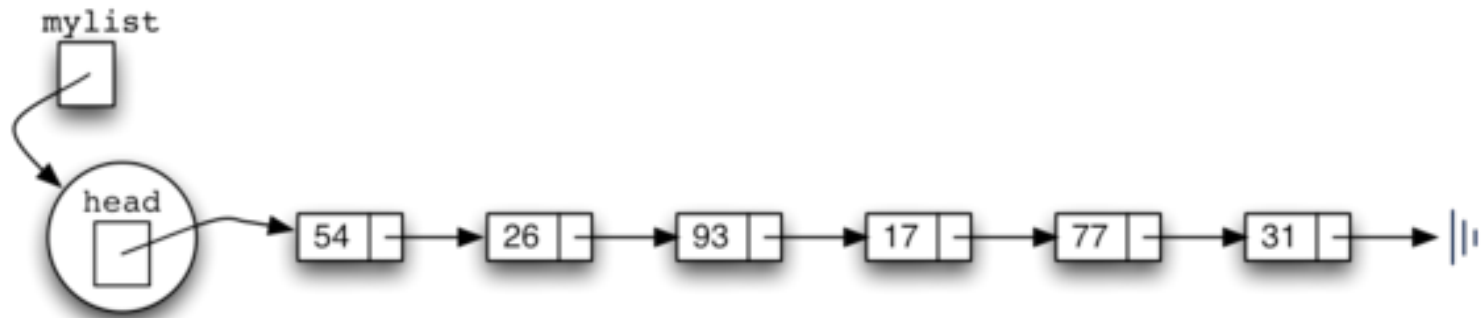
# 数据结构与算法 (Python版)

## 无序表的链表实现

陈斌 北京大学 [gischen@pku.edu.cn](mailto:gischen@pku.edu.cn)

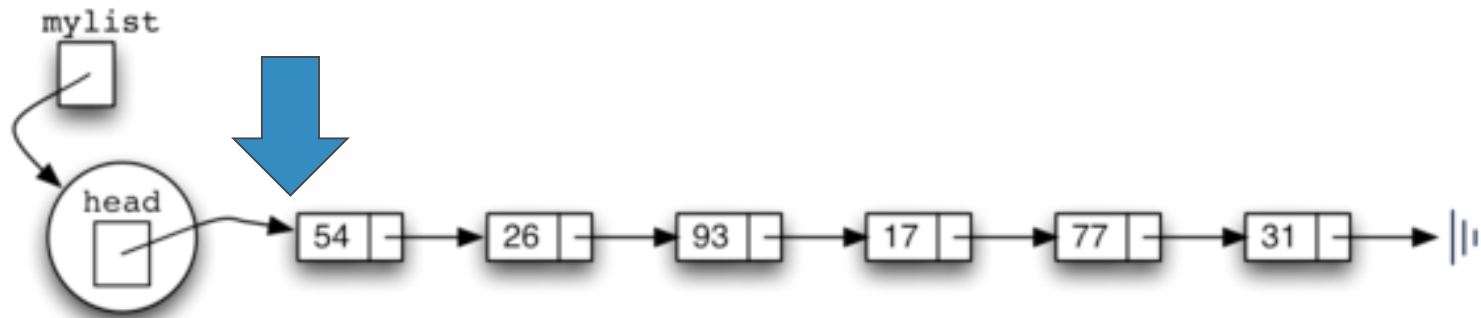
# 链表实现：无序表UnorderedList

- ❖ 接下来，考虑如何实现向无序表中添加数据项，实现**add**方法。
- ❖ 由于无序表并没有限定数据项之间的顺序
- ❖ 新数据项可以加入到原表的**任何**位置
- ❖ 按照实现的性能考虑，应添加到**最容易加入**的位置上。



# 链表实现：无序表UnorderedList

- ❖ 由链表结构我们知道
- ❖ 要访问到整条链上的所有数据项
- ❖ 都必须从表头head开始沿着next链接逐个向后查找
- ❖ 所以添加新数据项最快捷的位置是表头，整个链表的首位置。



# 链表实现：无序表UnorderedList

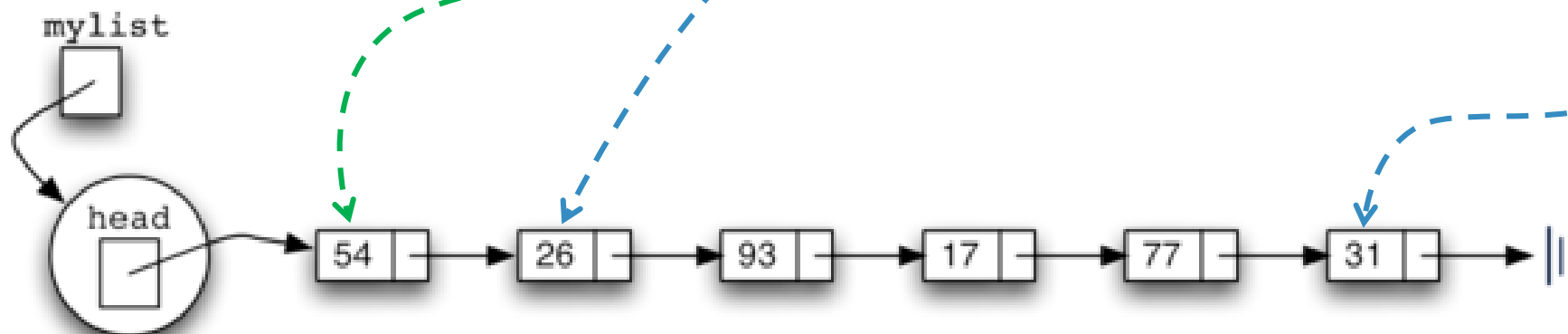
## ❖ add方法

## ❖ 按照右图的代码调用，形成的链表如下图

31是最先被加入的数据项，所以成为链表中最后一个项

而54是最后被加入的，是链表第一个数据项

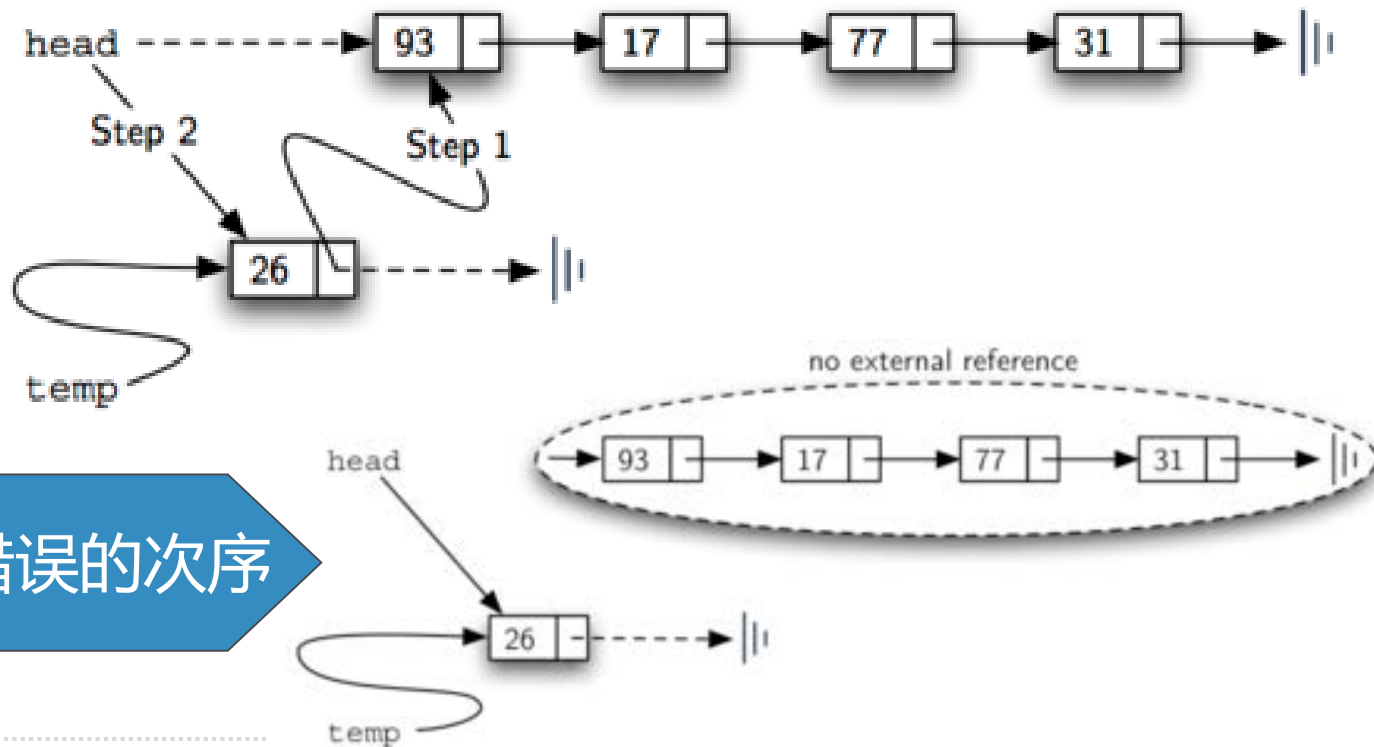
```
mylist.add(31)
mylist.add(77)
mylist.add(17)
mylist.add(93)
mylist.add(26)
mylist.add(54)
```



# 链表实现：add方法实现

❖ **链接次序很重要！**

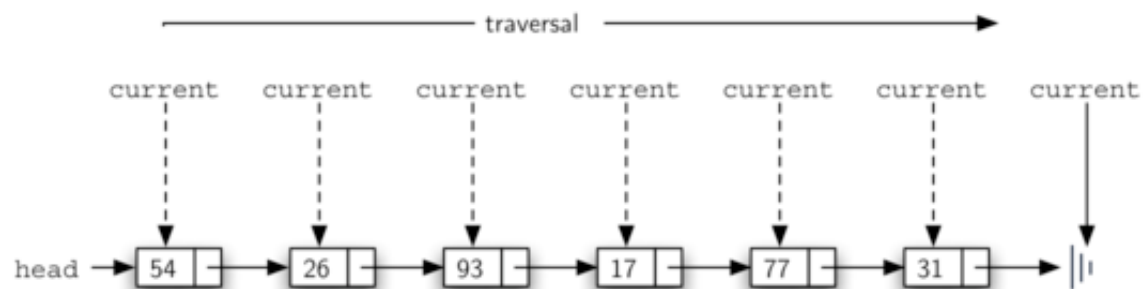
```
def add(self, item):  
    temp = Node(item)  
    temp.setNext(self.head)  
    self.head = temp
```



错误的次序

# 链表实现: size

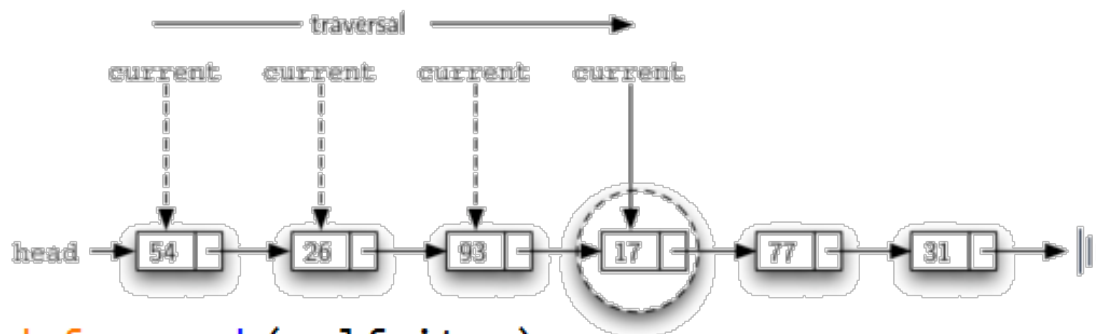
❖ size: 从链条头head开始遍历到表尾同时用变量累加经过的节点个数。



```
def size(self):  
    current = self.head  
    count = 0  
    while current != None:  
        ➡ count = count + 1  
        current = current.getNext()  
  
    return count
```

# 链表实现: search

- ❖ 从链表头head开始遍历到表尾，同时判断当前节点的数据项是否目标



```
def search(self,item):  
    current = self.head  
    found = False  
    while current != None and not found:  
        ➡ if current.getData() == item:  
            found = True  
        else:  
            current = current.getNext()  
  
    return found
```

## 链表实现：remove(item)方法

❖ 首先要找到item，这个过程跟search一样，但在删除节点时，需要特别的技巧

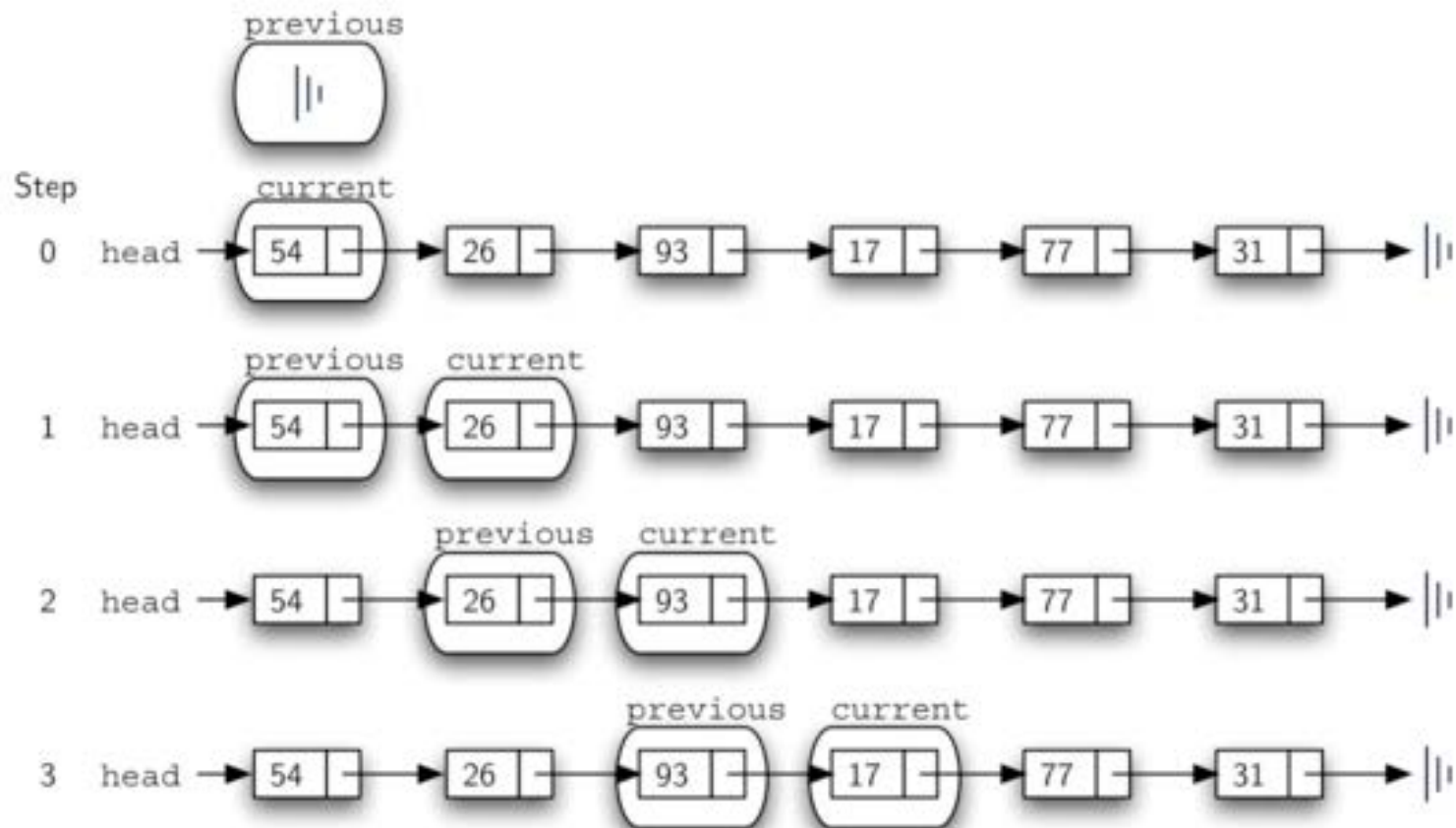
current指向的是当前匹配数据项的节点

而删除需要把前一个节点的next指向current的下一个节点

所以我们在search current的同时，还要维护前一个(previous)节点的引用

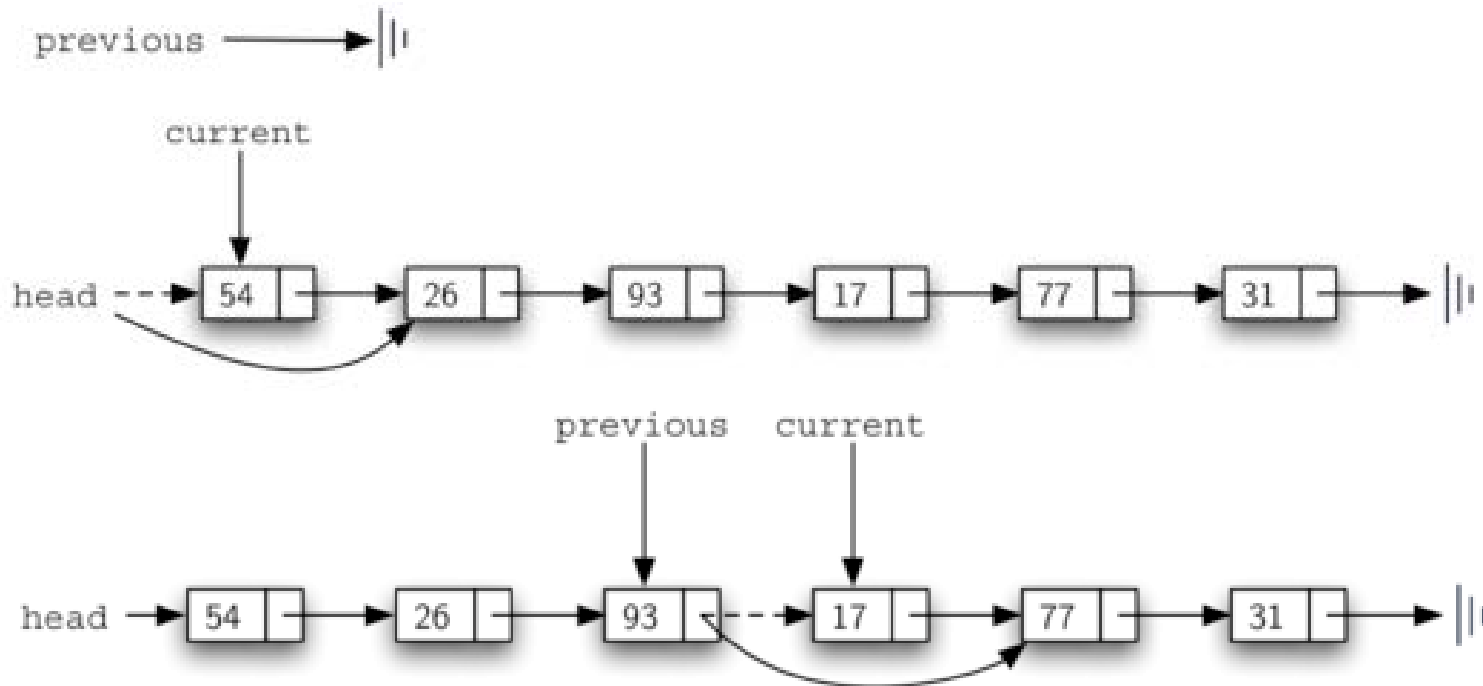


# 链表实现：remove(item)方法



# 链表实现：remove(item)方法

- ❖ 找到item之后，current指向item节点，previous指向前一个节点，开始执行删除，需要区分两种情况：  
current是首个节点；或者是位于链条中间的节



# 链表实现：remove(item)代码

```
def remove(self, item):  
    current = self.head  
    previous = None  
    found = False  
    while not found:
```

```
        if current.getData() == item:  
            found = True
```

```
        else:
```

```
            previous = current  
            current = current.getNext()
```



```
    if previous == None:  
        self.head = current.getNext()  
    else:
```

```
        previous.setNext(current.getNext())
```

