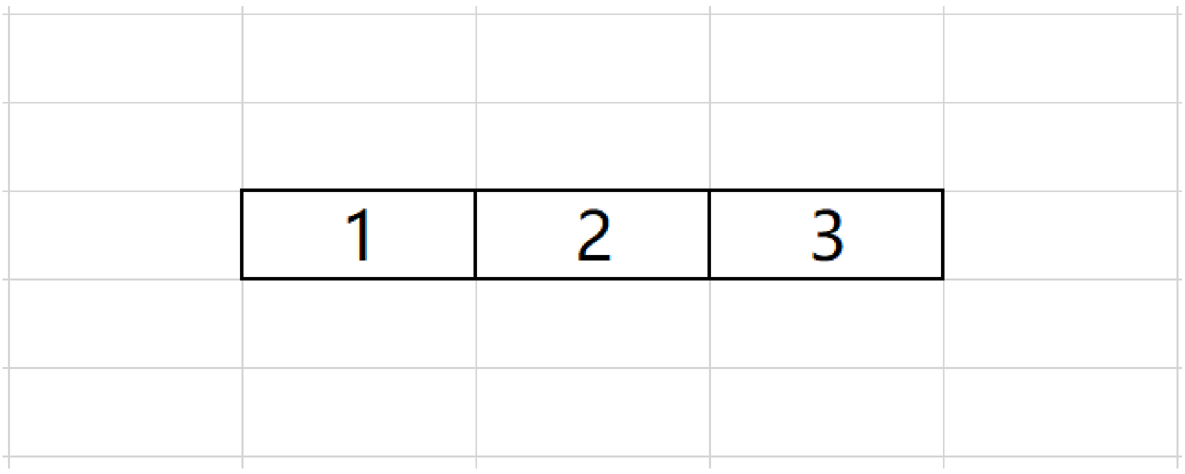
列表内存自动管理

案例：

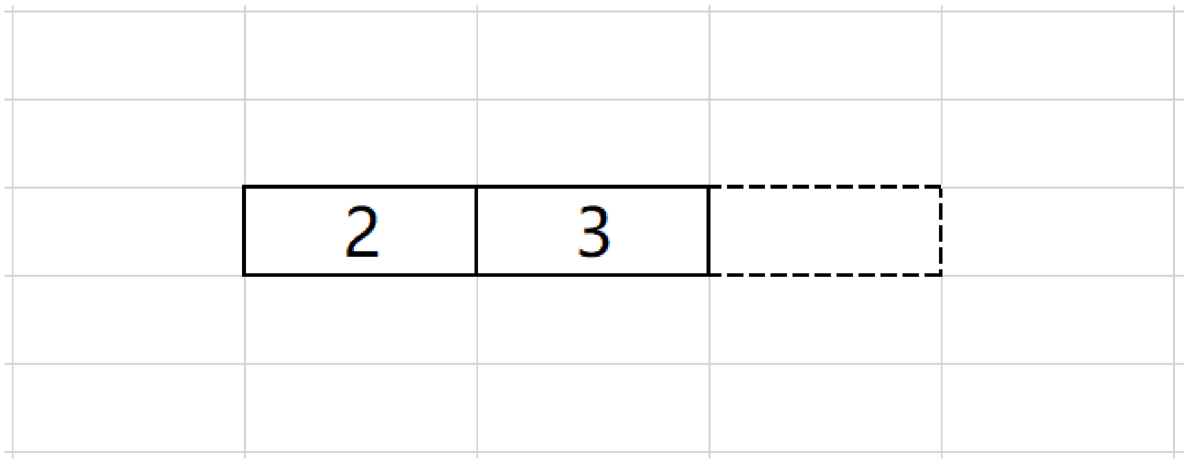
```
lis = [1, 2, 3]

for item in lis:
    lis.remove(item)

print(lis)
```

为什么没有得到空列表？

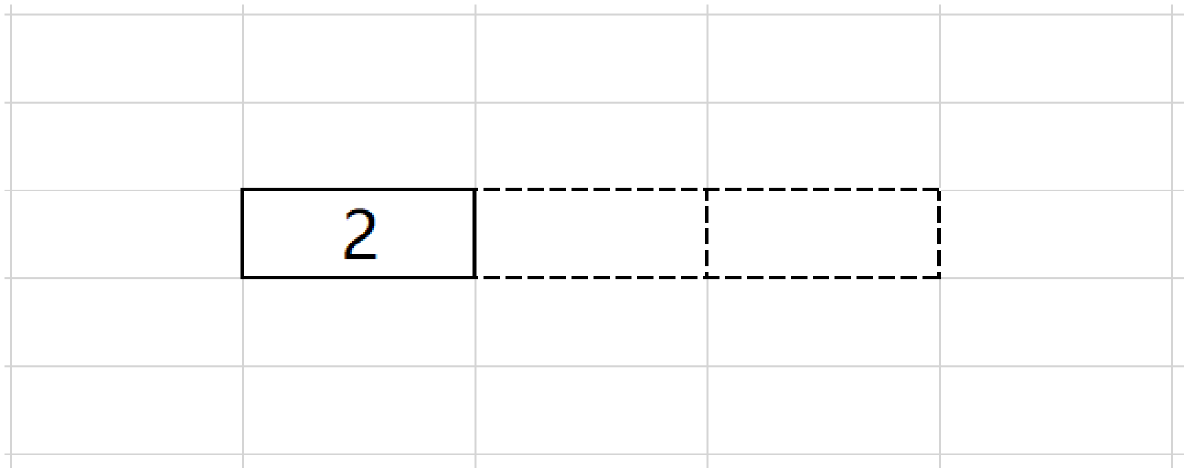
如果原列表在内存中为：



那么第一次遍历执行 lis.remove(item) 时，将索引0对应的元素删除（即删除元素1），然后对列表内存进行收缩，使得后面的元素2、3都往前移动，则列表在内存中变为：



第二次遍历执行 lis.remove(item) 时，将索引1对应的元素删除（即删除元素3）；第三次遍历时会因为超出索引范围而停止遍历，所以最终的列表为 [2]



列表内存自动管理功能： 在删除列表中的元素时，Python会自动对列表内存进行收缩，并移动列表中的元素以保证元素之间没有间隙，所以遍历删除列表中的元素时，被删元素后面的值会向前顶，导致漏删。

如何解决这个问题呢？

思路：遍历一个新的列表，不受原数据修改的影响即可

```python
import copy


lis = [1, 2, 3]
lis2 = [1, 2, 3]  # 思考：换成 lis2 = lis 是否可
以？
for item in lis2:
    lis.remove(item)
print(lis)


lis = [1, 2, 3]
for item in lis[:]:  # 浅拷贝
    lis.remove(item)
print(lis)


lis = [1, 2, 3]
lis2 = lis.copy()  # 浅拷贝
for item in lis2:
    lis.remove(item)
print(lis)


lis = [1, 2, 3]
lis2 = copy.copy(lis)  # 浅拷贝
for item in lis2:
    lis.remove(item)
print(lis)


lis = [1, 2, 3]
lis2 = copy.deepcopy(lis)  # 深拷贝
for item in lis2:
    lis.remove(item)
print(lis)


lis = [1, 2, 3]
lis2 = list(lis)  # 返回一个新的对象，同理：
tuple()/set()也可以
for item in lis2:
    lis.remove(item)
print(lis)
```

字典、集合遍历问题

字典、集合在遍历时，如果改变原数据的size，则会造成迭代时报错

```
dic = {"name": "Tom", "age": 18, "height": 188}
for key in dic:
    dic.pop(key)  # 改变了原数据大小，所以报错
print(dic)


dic = {"name": "Tom", "age": 18, "height": 188}
for key in dic:
    dic.update({"weight": 88})  # 改变了原数据大
小，所以报错
print(dic)
```

```python
dic = {"name": "Tom", "age": 18, "height": 188}
for key in dic:
    dic.update({"age": 22})  # 没有改变原数据大小，所以不报错
print(dic)
```

```python
set1 = {"name", "age", "height"}
for key in set1:
    set1.pop()  # 改变了原数据大小，所以报错
print(set1)


set1 = {"name", "age", "height"}
for key in set1:
    set1.add("weight")  # 改变了原数据大小，所以报错
print(set1)


set1 = {"name", "age", "height"}
for key in set1:
    set1.add("age")  # 没有改变原数据大小，所以不报错
print(set1)
```

解决思路：遍历一个新的字典/集合，不受原数据修改的影响即可

```python
import copy


dic = {"name": "Tom", "age": 18, "height": 188}
dic2 = {"name": "Tom", "age": 18, "height": 188}
 # 思考：换成 dic2 = dic 是否可以？
```

```python
for key in dic2:
    dic.pop(key)
print(dic)


dic = {"name": "Tom", "age": 18, "height": 188}
dic2 = dic.copy()  # 浅拷贝
for key in dic2:
    dic.pop(key)
print(dic)


dic = {"name": "Tom", "age": 18, "height": 188}
dic2 = copy.copy(dic)  # 浅拷贝
for key in dic2:
    dic.pop(key)
print(dic)


dic = {"name": "Tom", "age": 18, "height": 188}
dic2 = copy.deepcopy(dic)  # 深拷贝
for key in dic2:
    dic.pop(key)
print(dic)


dic = {"name": "Tom", "age": 18, "height": 188}
key_list = list(dic)  # 返回一个新的对象，同理：
tuple()/set()也可以
for key in key_list:
    dic.pop(key)
print(dic)
```

```python
dic = {"name": "Tom", "age": 18, "height": 188}
dict_keys = dic.keys()  # 思考：这样可以吗？
for key in dict_keys:
    dic.pop(key)
print(dic)
```