

# 【Python】列表list、元组tuple

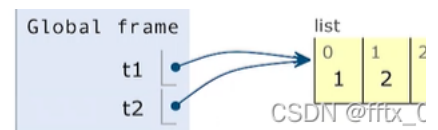
## 列表是可变类型

列表的字面量不是常量，而是表达式

```
>>> a = 6
>>> b = 8
>>> a
6
>>> b
8
>>> t = [6,8]
>>> t = [a, b]
>>> t
[6, 8]
>>> t = [a+2, b-a]
>>> t
[8, 2]
```

列表变量是列表的管理者（指针），而非所有者

```
1 t1 = [1,2,3]
2 t2 = t1
3 |
```



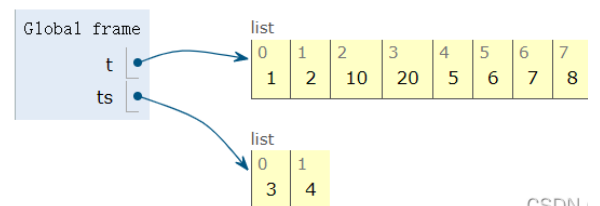
```
>>> a = "123"
>>> b = a
>>> id(a) == id(b)
True
>>>
>>> a = [1, 2, 3]
>>> b = a
>>> id(a) == id(b)
True
```

如何分管两个值一样的列表呢？——切片作右值

```
t2 = t1[:]
```

切片作左值时可以修改原列表

```
1 t = [1,2,3,4,5,6,7,8,9,10]
2 ts = t[2:4]
3 t[2:4] = [10,20]
```



## 浅拷贝、深拷贝

什么是可变对象，什么是不可变对象：

可变对象是指，一个对象在不改变其所指向的地址的前提下，可以修改其所指向的地址中的值；

不可变对象是指，一个对象所指向的地址上值是不能修改的，如果你修改了这个对象的值，那么它指向的地址就改变了，相当于你把这个对象复制出来一份，然后做了修改后存到另一个地址上了，但是可变对象就不会做这样的动作，而是直接在对象所指的地址上把值给改变了，而这个对象指向这个地址。

深拷贝和浅拷贝需要注意的地方就是可变元素的拷贝：

在浅拷贝时，拷贝出来的新对象的地址和原对象是不一样的，但是新对象里面的可变元素（如列表）的地址和原对象里的可变元素的地址是相同的。浅拷贝只拷贝的是浅层次的数据结构（不可变元素），对象里的可变元素作为深层次的数据结构并没有被拷贝到新地址里面去，而是和原对象指向同一个地址，所以在新对象或原对象里对这个可变元素做修改时，两个对象是同时改变的，但是深拷贝不会这样，这个是浅拷贝最根本的区别。

## 一、列表操作

### 定义、索引 []

```
1 # 列表定义
2 bickcles = ['trek', 'cannondale', 'redline', 'specialized']
3 # 打印列表
4 print(bickcles)
5 # 访问列表
6 print(bickcles[0])
7 print(bickcles[1])
8 print(bickcles[-1])
9 print(bickcles[-2])
10
11 # 修改列表
12 bickcles[-1] = 'newLast'
```

### 长度 len()

```
1 # 确定列表长度
2 cars = ['bmw', 'audi', 'toyota', 'subaru']
3 len(cars)
```

### 增加 list.append(),list.extend(),list.insert()

```
1 # 添加元素(末尾)
2 motorcycles = []
3 motorcycles.append('honda') #末尾增加一个元素
4 motorcycles.extend([7,8,9]) #末尾增加多个元素
5
6 # 添加元素(任意位置)
7 motorcycles.insert(0, 'ducati') # 插入在0位置
8
9 """
10 当list.insert()超过了最大下标，会做list.append()的动作
11 """
12 motorcycles.insert(100,-1) # 最后一个元素变成-1
```

### 删除 del,list.pop(),list.remove(),切片

```
1 del motorcycles[1]
2
3 popped_motorcycle = motorcycles.pop() # 弹出末尾元素
4 popped_motorcycle = motorcycles.pop(1) # 弹出1位置的元素
5
6
7 motorcycles.remove(2) # 删除第一次出现为2的元素
8 motorcycles.remove('ducati') # 删除第一次出现的值为'ducati'的元素
9
10 motorcycles[2] = []
```

### 查找 in,list.index(),list.count()

```
1 # 判定元素是否在列表中(in关键字)
2 'mushrooms' in requested_toppings
3 'mushrooms' not in requested_toppings
4
5 # 查找元素在列表中首次出现的位置
6 requested_toppings.index('onions')
7
8 # 计算元素出现次数
9 requested_toppings.count('mushrooms')
```

## 判空

```
1 # 判定列表是否为空
2 requested_toppings = []
3 if requested_toppings:
4     print("do something.")
5 else:
6     print("do others.")
```

## 二、list.sort(),sorted(),list.reverse(),reversed()

```
1 # 对列表永久排序并按字母逆序输出
2 cars.sort()
3 cars.sort(reverse=True)
4
5 # 对列表临时排序并按字母逆序输出
6 sorted(cars)
7 sorted(cars, reverse=True)
8
9 # 逆转列表存储
10 cars.reverse()
```

## 三、遍历列表 for in

```
1 # 遍历列表
2 magicians = ['alice', 'david', 'carolina']
3 for magician in magicians:
4     print(magician)
```

## 四、数值列表【range()或列表推导(解析)式】

```
1 # range()函数
2 for value in range(1, 5): # [1,5)
3     print(value)
4
5 # 使用range()生成数值列表
6 numbers = list(range(1, 6)) # [1,6),默认步长1
7
8 even_numbers = list(range(2, 11, 2)) # [2,11),步长2
9
10 # 使用range()充当for计数循环
```



## 五、切片（处理列表部分元素）

```
1 # 列表切片
2 players = ['charles', 'martina', 'michael', 'florence', 'eli']
3 print(players[0:3]) # [0,3)
4
5 print(players[:4]) # [0,4)
6 print(players[2:]) # [2,4]
7 print(players[-3:]) # {-3,-2,-1}即最后3名
8
9 # 复制列表
10 my_foods = ['pizza', 'falafel', 'carrot cake']
```



```
print(data[-3:])      # 最后3个元素
print(data[:-5])      # 除最后5个元素之外的所有元素
```

```
>>> [1, 2, 3] + ['c', 'java', 'python']
[1, 2, 3, 'c', 'java', 'python']
>>> [1]*10
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
>>> [1, 2, 3] < [2, 3, 4]
True
>>> weekdays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
>>> weekdays[2]
'Wednesday'
>>>
```

加减  
比较  
索引

CSDN @fftx\_00

## 六、列表推导式

### 列表推导式

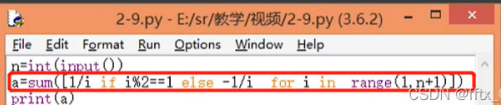
- 列表推导式（又称列表解析式）提供了一种简明扼要的方法来创建列表。
- 它可以将循环和条件判断结合，从而避免语法冗长的代码，同时提高程序性能。
- 基本格式：

```
[ expression for item in iterable ]
```

```
>>> n1 = [number for number in range(1, 8)]
>>> n1
[1, 2, 3, 4, 5, 6, 7]
>>> n1 = [number for number in range(1, 8) if number%2 == 1]
>>> n1
[1, 3, 5, 7]
>>>
```

CSDN @fftx\_00

求  $1-1/2+1/3-1/4+\dots$  之前n项和( $n \geq 10$ )



```
2-9.py - E:/sr/教学/视频/2-9.py (3.6.2)
File Edit Format Run Options Window Help
n=int(input())
a=sum(1/i if i%2==1 else -1/i for i in range(1,n+1))
print(a)
```

```
>>> '6'*10
'6666666666'
>>> ['6'*i for i in range(1, 10)]
['6', '66', '666', '6666', '66666', '666666', '6666666', '66666666', '666666666', '6666666666']
>>> [int('6'*i) for i in range(1, 10)]
[6, 66, 666, 6666, 66666, 666666, 6666666, 66666666, 666666666, 6666666666]
>>> sum([int('6'*i) for i in range(1, 10)])
740740734
```

CSDN @fftx\_00

## 六、元组

Python将不能修改的值称为不可变的  
不可变的列表称为元组

```
1 # 元组定义
2 dimensions = (200, 50)
3
```

```
4 | my_t = (3,) # 严格地说,元组是由逗号标识的
5 | my_t = 3,4 ## 即my_t = (3,4)
6 |
7 | # 虽然不能修改元组的元素,但是可以重新给整个元组赋值
8 | dimensions = (200, 50)
9 | dimensions = (400, 100)
```