可迭代对象

内建函数	函数签名	说明
iter	iter(iterable)	把一个可迭代对象包装成迭代器
next	next(iterable[, default])	取迭代器下一个元素 如果已经取完，继续取抛StopIteration异常
reversed	reversed(seq)	返回一个翻转元素的迭代器
enumerate	enumerate(seq, start=0)	迭代一个可迭代对象，返回一个迭代器 每一个元素都是数字和元素构成的二元组

迭代器

- 特殊的对象，一定是可迭代对象，具备可迭代对象的特征
- 通过iter方法把一个可迭代对象封装成迭代器
- 通过next方法，获取 迭代器对象的一个元素
- 生成器对象，就是迭代器对象。但是迭代器对象未必是生成器对象

可迭代对象

- 能够通过迭代一次次返回不同的元素的对象
  - 所谓相同，不是指值是否相同，而是元素在容器中是否是同一个，例如列表中值可以重复的，['a', 'a']，虽然这个列表有2个元素，值一样，但是两个'a'是不同的元素
- 可以迭代，但是未必有序，未必可索引
- 可迭代对象有：list、tuple、string、bytes、bytearray、range、set、dict、生成器、迭代器等
- 可以使用成员操作符in、not in
  - 对于线性数据结构，in本质上是在遍历对象，时间复杂度为O(n)

```python
lst = [1, 3, 5, 7, 9]

it = iter(lst) # 返回一个迭代器对象
print(next(it))
print(next(it))

for i, x in enumerate(it, 2):
    print(i, x)
#print(next(it)) # StopIteration
print()

for x in reversed(lst):
    print(x)

# 比较下面的区别，说明原因？
it = iter(lst)
print(1 in it)
print(1 in it)
```

```
print(1 in lst)
print(1 in lst)
```

内建函数

排序sorted

定义`sorted(iterable, *, key=None, reverse=False) ->list`

```
sorted(lst, key=lambda x:6-x) # 返回新列表
list.sort(key=lambda x: 6-x) # 就地修改

sorted([1, '2', 3], key=lambda x: str(x))
sorted([1, '2', 3], key=str)
```

过滤filter

- 定义`filter(function, iterable)`
- 对可迭代对象进行遍历，返回一个迭代器
- function参数是一个参数的函数，且返回值应当是bool类型，或其返回值等效布尔值。
- function参数如果是None，可迭代对象的每一个元素自身等效布尔值

```
list(filter(lambda x: x%3==0, [1,9,55,150,-3,78,28,123]))
list(filter(None, range(5)))
list(filter(None, range(-5, 5)))
```

映射map

- 定义`map(function, *iterables) -> map object`
- 对多个可迭代对象的元素，按照指定的函数进行映射
- 返回一个迭代器

```
list(map(lambda x: 2*x+1, range(10)))
dict(map(lambda x: (x%5, x), range(500)))
dict(map(lambda x,y: (x,y), 'abcde', range(10)))
```

拉链函数zip

- zip(*iterables)
- 像拉链一样，把多个可迭代对象合并在一起，返回一个迭代器
- 将每次从不同对象中取到的元素合并成一个元组

```
list(zip(range(10),range(10)))
list(zip(range(10),range(10),range(5),range(10)))

dict(zip(range(10),range(10)))
{str(x):y for x,y in zip(range(10),range(10))}
```

马哥教育
IT人的高薪职业学院