filter

Python内建的`filter()`函数用于过滤序列。

和`map()`类似，`filter()`也接收一个函数和一个序列。和`map()`不同的是，`filter()`把传入的函数依次作用于每个元素，然后根据返回值是`True`还是`False`决定保留还是丢弃该元素。

例如，在一个list中，删掉偶数，只保留奇数，可以这么写：

```
def is_odd(n):
    return n % 2 == 1

list(filter(is_odd, [1, 2, 4, 5, 6, 9, 10, 15]))
# 结果: [1, 5, 9, 15]
```

把一个序列中的空字符串删掉，可以这么写：

```
def not_empty(s):
    return s and s.strip()

list(filter(not_empty, ['A', '', 'B', None, 'C', '  ']))
# 结果: ['A', 'B', 'C']
```

可见用`filter()`这个高阶函数，关键在于正确实现一个“筛选”函数。

注意到`filter()`函数返回的是一个`Iterator`，也就是一个惰性序列，所以要强迫`filter()`完成计算结果，需要用`list()`函数获得所有结果并返回list。

用filter求素数

计算素数的一个方法是埃氏筛法，它的算法理解起来非常简单：

首先，列出从`2`开始的所有自然数，构造一个序列：

2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...

取序列的第一个数`2`，它一定是素数，然后用`2`把序列的`2`的倍数筛掉：

3, ~~4~~, 5, ~~6~~, 7, ~~8~~, 9, ~~10~~, 11, ~~12~~, 13, ~~14~~, 15, ~~16~~, 17, ~~18~~, 19, ~~20~~, ...

取新序列的第一个数`3`，它一定是素数，然后用`3`把序列的`3`的倍数筛掉：

5, ~~6~~, 7, ~~8~~, ~~9~~, ~~10~~, 11, ~~12~~, 13, ~~14~~, ~~15~~, ~~16~~, 17, ~~18~~, 19, ~~20~~, ...

取新序列的第一个数`5`，然后用`5`把序列的`5`的倍数筛掉：

7, ~~8~~, ~~9~~, ~~10~~, 11, ~~12~~, 13, ~~14~~, ~~15~~, ~~16~~, 17, ~~18~~, 19, ~~20~~, ...

不断筛下去，就可以得到所有的素数。

用Python来实现这个算法，可以先构造一个从`3`开始的奇数序列：

```
def _odd_iter():
    n = 1
    while True:
        n = n + 2
        yield n
```

注意这是一个生成器，并且是一个无限序列。

然后定义一个筛选函数：

```
def _not_divisible(n):
    return lambda x: x % n > 0
```

最后，定义一个生成器，不断返回下一个素数：

```
def primes():
    yield 2
    it = _odd_iter() # 初始序列
    while True:
        n = next(it) # 返回序列的第一个数
        yield n
        it = filter(_not_divisible(n), it) # 构造新序列
```

这个生成器先返回第一个素数`2`，然后，利用`filter()`不断产生筛选后的新的序列。

由于`primes()`也是一个无限序列，所以调用时需要设置一个退出循环的条件：

```
# 打印1000以内的素数:
for n in primes():
    if n < 1000:
        print(n)
    else:
        break
```

注意到`Iterator`是惰性计算的序列，所以我们可以用Python表示“全体自然数”，“全体素数”这样的序列，而代码非常简洁。

练习

回数是指从左向右读和从右向左读都是一样的数，例如`12321`，`909`。请利用`filter()`筛选出回数：

```
# -*- coding: utf-8 -*-
def is_palindrome(n):
    pass



# 测试:
output = filter(is_palindrome, range(1, 1000))
print('1~1000:', list(output))
if list(filter(is_palindrome, range(1, 200))) == [1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, 44, 55, 66, 77, 88, 99, 101, 111, 121, 131, 141, 151, 161, 171, 181, 191]:
    print('测试成功!')
else:
    print('测试失败!')
```

▶ Run

小结

`filter()`的作用是从一个序列中筛出符合条件的元素。由于`filter()`使用了惰性计算，所以只有在取`filter()`结果的时候，才会真正筛选并每次返回下一个筛出的元素。

参考源码

do_filter.py

prime_numbers.py

读后有收获可以支付宝请作者喝咖啡，读后有疑问请加微信群讨论：

还可以分享给朋友：

分享到微博

Previous Page Next Page

Comments

Make a comment

Sign in to make a comment