

- 7.4. 生成器
- 7.5. 迭代器
- 8. 函数式编程
 - 8.1. 高阶函数
 - 8.1.1. map/reduce
 - 8.1.2. filter
 - 8.1.3. sorted
 - 8.2. 返回函数
 - 8.3. 匿名函数
 - 8.4. 装饰器
 - 8.5. 偏函数
- 9. 模块
 - 9.1. 使用模块
 - 9.2. 安装第三方模块
- 10. 面向对象编程
 - 10.1. 类和实例
 - 10.2. 访问限制
 - 10.3. 继承和多态
 - 10.4. 获取对象信息
 - 10.5. 实例属性和类属性
- 11. 面向对象高级编程
 - 11.1. 使用_slots_
 - 11.2. 使用@property
 - 11.3. 多重继承
 - 11.4. 定制类
 - 11.5. 使用枚举类
 - 11.6. 使用元类
- 12. 错误、调试和测试
 - 12.1. 错误处理
 - 12.2. 调试
 - 12.3. 单元测试
 - 12.4. 文档测试
- 13. IO编程
- 14. 进程和线程
- 15. 正则表达式
- 16. 常用内建模块
- 17. 常用第三方模块
- 18. 图形界面
- 19. 网络编程
- 20. 电子邮件
- 21. 访问数据库
- 22. Web开发
- 23. 异步IO
- 24. FAQ
- 25. 期末总结

map/reduce

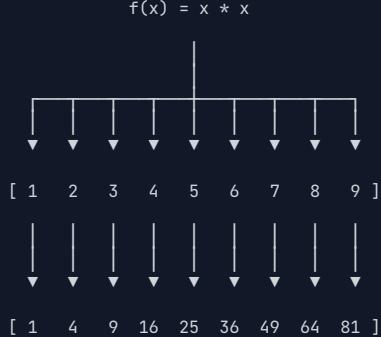
 廖雪峰 GitHub 知乎 Twitter
资深软件开发工程师，业余马拉松选手。

Python内建了 `map()` 和 `reduce()` 函数。

如果你读过Google的那篇大名鼎鼎的论文“MapReduce: Simplified Data Processing on Large Clusters”，你就能大概明白map/reduce的概念。

我们先看`map`。`map()` 函数接收两个参数，一个是函数，一个是 `Iterable`，`map` 将传入的函数依次作用到序列的每个元素，并把结果作为新的 `Iterator` 返回。

举例说明，比如我们有一个函数 $f(x)=x^2$ ，要把这个函数作用在一个list `[1, 2, 3, 4, 5, 6, 7, 8, 9]` 上，就可以用 `map()` 实现如下：

$$f(x) = x * x$$


[1 2 3 4 5 6 7 8 9]
[1 4 9 16 25 36 49 64 81]

现在，我们用Python代码实现：

```
>>> def f(x):  
...     return x * x  
...  
>>> r = map(f, [1, 2, 3, 4, 5, 6, 7, 8, 9])  
>>> list(r)  
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```

`map()` 传入的第一个参数是 `f`，即函数对象本身。由于结果 `r` 是一个 `Iterator`，`Iterator` 是惰性序列，因此通过 `list()` 函数让它把整个序列都计算出来并返回一个list。

你可能会想，不需要 `map()` 函数，写一个循环，也可以计算出结果：

```
L = []  
for n in [1, 2, 3, 4, 5, 6, 7, 8, 9]:  
    L.append(f(n))  
print(L)
```

的确可以，但是，从上面的循环代码，能一眼看明白“把 $f(x)$ 作用在list的每一个元素并把结果生成一个新的list”吗？

所以，`map()` 作为高阶函数，事实上它把运算规则抽象了，因此，我们不但可以计算简单的 $f(x)=x^2$ ，还可以计算任意复杂的函数，比如，把这个list所有数字转为字符串：

```
>>> list(map(str, [1, 2, 3, 4, 5, 6, 7, 8, 9]))  
['1', '2', '3', '4', '5', '6', '7', '8', '9']
```

只需要一行代码。

再看 `reduce` 的用法。`reduce` 把一个函数作用在一个序列 `[x1, x2, x3, ...]` 上，这个函数必须接收两个参数，`reduce` 把结果继续和序列的下一个元素做累积计算，其效果就是：

```
reduce(f, [x1, x2, x3, x4]) = f(f(f(x1, x2), x3), x4)
```

比方说对一个序列求和，就可以用 `reduce` 实现：

```
>>> from functools import reduce  
>>> def add(x, y):
```

7.4. 生成器
7.5. 迭代器
8. 函数式编程
8.1. 高阶函数
8.1.1. map/reduce
8.1.2. filter
8.1.3. sorted
8.2. 返回函数
8.3. 匿名函数
8.4. 装饰器
8.5. 偏函数
9. 模块
9.1. 使用模块
9.2. 安装第三方模块
10. 面向对象编程
10.1. 类和实例
10.2. 访问限制
10.3. 继承和多态
10.4. 获取对象信息
10.5. 实例属性和类属性
11. 面向对象高级编程
11.1. 使用_slots_
11.2. 使用@property
11.3. 多重继承
11.4. 定制类
11.5. 使用枚举类
11.6. 使用元类
12. 错误、调试和测试
12.1. 错误处理
12.2. 调试
12.3. 单元测试
12.4. 文档测试
13. IO编程
14. 进程和线程
15. 正则表达式
16. 常用内建模块
17. 常用第三方模块
18. 图形界面
19. 网络编程
20. 电子邮件
21. 访问数据库
22. Web开发
23. 异步IO
24. FAQ
25. 期末总结

```
...     return x + y
...
>>> reduce(add, [1, 3, 5, 7, 9])
25
```

当然求和运算可以直接用Python内建函数 `sum()`，没必要动用 `reduce`。

但是如果要把序列 `[1, 3, 5, 7, 9]` 变换成整数 `13579`，`reduce` 就可以派上用场：

```
>>> from functools import reduce
>>> def fn(x, y):
...     return x * 10 + y
...
>>> reduce(fn, [1, 3, 5, 7, 9])
13579
```

这个例子本身没多大用处，但是，如果考虑到字符串 `str` 也是一个序列，对上面的例子稍加改动，配合 `map()`，我们就可以写出把 `str` 转换为 `int` 的函数：

```
>>> from functools import reduce
>>> def fn(x, y):
...     return x * 10 + y
...
>>> def char2num(s):
...     digits = {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9}
...     return digits[s]
...
>>> reduce(fn, map(char2num, '13579'))
13579
```

整理成一个 `str2int` 的函数就是：

```
from functools import reduce

DIGITS = {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9}

def str2int(s):
    def fn(x, y):
        return x * 10 + y
    def char2num(s):
        return DIGITS[s]
    return reduce(fn, map(char2num, s))
```

还可以用lambda函数进一步简化成：

```
from functools import reduce

DIGITS = {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9}

def char2num(s):
    return DIGITS[s]

def str2int(s):
    return reduce(lambda x, y: x * 10 + y, map(char2num, s))
```

也就是说，假设Python没有提供 `int()` 函数，你完全可以自己写一个把字符串转化为整数的函数，而且只需要几行代码！

`lambda`函数的用法在后面介绍。

练习

利用 `map()` 函数，把用户输入的不规范的英文名字，变为首字母大写，其他小写的规范名字。输入： `['adam', 'LISA', 'barT']`，输出： `['Adam', 'Lisa', 'Bart']`：

```
def normalize(name):
    pass

# 测试：
```

L1 = ['adam', 'LISA', 'barT']
L2 = list(map(normalize, L1))
print(L2)

Python提供的 `sum()` 函数可以接受一个list并求和, 请编写一个 `prod()` 函数, 可以接受一个list并利用 `reduce()` 求积:

```
from functools import reduce  
  
def prod(L):  
    pass  
  
print('3 * 5 * 7 * 9 =', prod([3, 5, 7, 9]))  
if prod([3, 5, 7, 9]) == 945:  
    print('测试成功!')  
else:  
    print('测试失败!')
```

利用 `map` 和 `reduce` 编写一个 `str2float` 函数, 把字符串 '123.456' 转换成浮点数 123.456 :

```
from functools import reduce  
  
def str2float(s):  
    pass  
  
print('str2float(\'123.456\') =', str2float('123.456'))  
if abs(str2float('123.456') - 123.456) < 0.00001:  
    print('测试成功!')  
else:  
    print('测试失败!')
```

参考代码

[do_map.py](#)
[do_reduce.py](#)

小结

`map` 用于将一个函数作用于一个序列, 以此得到另一个序列;
`reduce` 用于将一个函数依次作用于上次计算的结果和序列的下一个元素, 以此得到最终结果。

« 高阶函数 filter »



一站式 DevOps 研发效能平台
灵活选择部署方式 | 支持 SaaS 在线使用 | 私有化部署
进入 Gitee 官网

Comments

Comments loaded. To post a comment, please Sign In

 造轮子 @ 2025/12/26 01:02:33

```
#练习一。  
UP={};LOW={};  
for x in [ (chr(ord('a')+i),chr(ord('A')+i)) for i in range(0,26)]:  
    UP[x[0]]=x[1];LOW[x[1]]=x[0];
```

- 7.4. 生成器
- 7.5. 迭代器
- 8. 函数式编程
- 8.1. 高阶函数
- 8.1.1. map/reduce
- 8.1.2. filter
- 8.1.3. sorted
- 8.2. 返回函数
- 8.3. 匿名函数
- 8.4. 装饰器
- 8.5. 偏函数
- 9. 模块
- 9.1. 使用模块
- 9.2. 安装第三方模块
- 10. 面向对象编程
- 10.1. 类和实例
- 10.2. 访问限制
- 10.3. 继承和多态
- 10.4. 获取对象信息
- 10.5. 实例属性和类属性
- 11. 面向对象高级编程
- 11.1. 使用_slots_
- 11.2. 使用@property
- 11.3. 多重继承
- 11.4. 定制类
- 11.5. 使用枚举类
- 11.6. 使用元类
- 12. 错误、调试和测试
- 12.1. 错误处理
- 12.2. 调试
- 12.3. 单元测试
- 12.4. 文档测试
- 13. IO编程
- 14. 进程和线程
- 15. 正则表达式
- 16. 常用内建模块
- 17. 常用第三方模块
- 18. 图形界面
- 19. 网络编程
- 20. 电子邮件
- 21. 访问数据库
- 22. Web开发
- 23. 异步IO
- 24. FAQ
- 25. 期末总结

```
def cvt(t):
    if t[0]==0:
        return UP.get(t[1],t[1])
    return LOW.get(t[1],t[1])
def normalize(name):
    s=""
    for c in map(cvt,enumerate(name)):
        s="%s%s"%(s,c)
    return s

#练习二、
```

[Read More ▾](#)



爱夏天的大西瓜 @ 2025/12/24 03:49:42

```
from functools import reduce

DIGITS = {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9}

def str2float(s):
    def char2num(s):
        return DIGITS[s]
    def fn(x, y):
        return x * 10 + y
    int_part = reduce(fn, map(char2num, s.split('.')[0]))
    float_part = reduce(fn, map(char2num, s.split('.')[1])) / (10 ** len(s.split('.')[1]))
    return int_part + float_part

print('str2float(\'123.456\') =', str2float('123.456'))
```

[Read More ▾](#)



杰 @ 2025/12/21 00:57:20

很多的作业用“parts = s.split(.)”分割成整数/小数两部分。也可以直接对s是否有小数点 “.” 来判断整数/小数，然后用切片方法操作。

```
def char_to_num(c):
    """
    查字典，将单个字符转换为数字 ('0'-'9' -> 0-9)
    """

    if not c.isdigit():
        raise ValueError(f"字符 '{c}' 不是数字。")

    digits = {
        '0': 0, '1': 1, '2': 2, '3': 3, '4': 4,
        '5': 5, '6': 6, '7': 7, '8': 8, '9': 9
    }
    return digits[c]
```

[Read More ▾](#)



已读乱回 @ 2025/12/19 03:55:08

```
def normalize(name):
    return name.capitalize()

# 测试:
def test_normalize():
    l1 = ['adam', 'LISA', 'barT']
    l2 = list(map(normalize, l1))
    print(l2)

def prod(elm):
    return reduce(lambda x, y: x * y, elm)
```

7.4. 生成器
7.5. 迭代器
8. 函数式编程
8.1. 高阶函数
8.1.1. map/reduce
8.1.2. filter
8.1.3. sorted
8.2. 返回函数
8.3. 匿名函数
8.4. 装饰器
8.5. 偏函数
9. 模块
9.1. 使用模块
9.2. 安装第三方模块
10. 面向对象编程
10.1. 类和实例
10.2. 访问限制
10.3. 继承和多态
10.4. 获取对象信息
10.5. 实例属性和类属性
11. 面向对象高级编程
11.1. 使用_slots_
11.2. 使用@property
11.3. 多重继承
11.4. 定制类
11.5. 使用枚举类
11.6. 使用元类
12. 错误、调试和测试
12.1. 错误处理
12.2. 调试
12.3. 单元测试
12.4. 文档测试
13. IO编程
14. 进程和线程
15. 正则表达式
16. 常用内建模块
17. 常用第三方模块
18. 图形界面
19. 网络编程
20. 电子邮件
21. 访问数据库
22. Web开发
23. 异步IO
24. FAQ
25. 期末总结

```
def test_prod():
```

[Read More ▾](#)

Rory @ 2025/12/17 03:04:12

根据需求自定义函数，到了高阶函数部分的学习了

QQ用户 @ 2025/12/16 20:15:03

1

哆啦A没有梦 @ 2025/12/16 04:59:35

```
#str转换为float from functools import reduce def a(l): for i in range(len(l)+1): if l[i]=='.': return i def b(l): digit={'1':1,'2':2,'3':3,'4':4,'5':5,'6':6,'7':7,'8':8,'9':9,'0':0} return digit[l] def c(x,y): return x*10+y def s2f(x): m=reduce(c,map(b,x[a(x)])) n=reduce(c,map(b,x[a(x)+1:])) n=n/(10**len(str(n))) return m+n print(s2f('123.124'))
```

啊，你以为我傻 @ 2025/12/11 04:42:19

```
def normalize(name):  
    return name[0].upper() + name[1:].lower()
```

```
from functools import reduce  
def prod(L):  
    def multiply(x,y):  
        return x * y  
    return reduce(multiply,L)
```

```
DIGITS = {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4,  
         '5': 5, '6': 6, '7': 7, '8': 8, '9': 9}  
def str2float(s):  
    if '.' in s:  
        int_part,dec_part = s.split('.').  
    else:
```

[Read More ▾](#)

易 @ 2025/12/3 03:41:19

```
def normalize(name):  
    return name[0].upper()+name[1:].lower()  
  
# 测试:  
L1 = ['adam', 'LISA', 'barT']  
L2 = list(map(normalize, L1))  
print(L2)
```

```
from functools import reduce  
  
def prod(L):  
    return reduce(lambda x,y:x*y, L)  
  
print('3 * 5 * 7 * 9 =', prod([3, 5, 7, 9]))  
if prod([3, 5, 7, 9]) == 945:
```

[Read More ▾](#)

大芒果 @ 2025/12/2 05:01:44

```
def str2float(s):  
    def char2num(s):  
        digits = {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8}
```

7.4. 生成器

7.5. 迭代器

8. 函数式编程

8.1. 高阶函数

8.1.1. map/reduce

8.1.2. filter

8.1.3. sorted

8.2. 返回函数

8.3. 匿名函数

8.4. 装饰器

8.5. 偏函数

9. 模块

9.1. 使用模块

9.2. 安装第三方模块

10. 面向对象编程

10.1. 类和实例

10.2. 访问限制

10.3. 继承和多态

10.4. 获取对象信息

10.5. 实例属性和类属性

11. 面向对象高级编程

11.1. 使用_slots_

11.2. 使用@property

11.3. 多重继承

11.4. 定制类

11.5. 使用枚举类

11.6. 使用元类

12. 错误、调试和测试

12.1. 错误处理

12.2. 调试

12.3. 单元测试

12.4. 文档测试

13. IO编程

14. 进程和线程

15. 正则表达式

16. 常用内建模块

17. 常用第三方模块

18. 图形界面

19. 网络编程

20. 电子邮件

21. 访问数据库

22. Web开发

23. 异步IO

24. FAQ

25. 期末总结

```
return digits[s]

def str2Int(x, y):
    return x * 10 + y;

def str2Digital(x, length):
    return x / (10 ** length)

[num1, num2] = s.split('.')
return reduce(str2Int, map(char2num, num1)) + str2Digital(reduce(str2Int, map(char2num, num2)), len(num2))
```

Hypersomnia @ 2025/11/27 08:41:56

```
def normalize(name):
    return name[0].upper() + name[1:].lower()
```

```
from functools import reduce
def prod(L):
    return reduce(lambda x, y: x * y, L)
```

```
from functools import reduce
def char2num(s):
    return {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9}[s]
def str2float(s):
    dot = s.find('.')
    int_part = s[:dot]
    float_part = s[dot + 1:]
    ans1 = reduce(lambda x, y: x * 10 + y, map(char2num, int_part))
    ans2 = float('0.' + float_part)
    return ans1 + ans2
```

Read More ▾

Floris @ 2025/11/27 02:32:41

```
# coding=utf-8
from functools import reduce

def normalize(name):
    return name[0].upper() + name[1:].lower()

#####
def prod(L):
    return reduce(lambda x, y: x*y, L)

#####

def str2float(s):
    a = str.split(s, ".")[1]
    return int(str.split(s, ".")[0]) + float(int(a) / 10**len(a))
```

别管实现，你就说能不能用嘛

wanna @ 2025/11/25 05:02:36

```
#字符串转换为数值
def str2float(s):

    # 输入验证
    if not isinstance(s, str):
        raise TypeError("输入必须是字符串")

    #拆分整数和小数部分
    parts=s.split('.')

    # 验证数字格式
    if not s.replace('.','').isdigit():
        raise ValueError("输入包含非数字字符")
```

验证小数点数量

[Read More ▾](#)

7.4. 生成器
7.5. 迭代器
8. 函数式编程
8.1. 高阶函数
8.1.1. map/reduce
8.1.2. filter
8.1.3. sorted
8.2. 返回函数
8.3. 匿名函数
8.4. 装饰器
8.5. 偏函数
9. 模块
9.1. 使用模块
9.2. 安装第三方模块
10. 面向对象编程
10.1. 类和实例
10.2. 访问限制
10.3. 继承和多态
10.4. 获取对象信息
10.5. 实例属性和类属性
11. 面向对象高级编程
11.1. 使用_slots_
11.2. 使用@property
11.3. 多重继承
11.4. 定制类
11.5. 使用枚举类
11.6. 使用元类
12. 错误、调试和测试
12.1. 错误处理
12.2. 调试
12.3. 单元测试
12.4. 文档测试
13. IO编程
14. 进程和线程
15. 正则表达式
16. 常用内建模块
17. 常用第三方模块
18. 图形界面
19. 网络编程
20. 电子邮件
21. 访问数据库
22. Web开发
23. 异步IO
24. FAQ
25. 期末总结

⌚ @ 2025/11/24 10:32:38

第三题为了把这两弄进去想了很久

```
# 利用map()函数，把用户输入的不规范的英文名字，变为首字母大写，其他小写的规范名字。  
# 输入: ['adam', 'LISA', 'barT']，输出: ['Adam', 'Lisa', 'Bart']:  
from dataclasses import replace  
from warnings import deprecated  
  
def normalize(name):  
    return name.capitalize()  
  
# 测试:  
L1 = ['adam', 'LISA', 'barT']  
L2 = list(map(normalize, L1))  
print(L2)
```

[Read More ▾](#)

⌚ @ 2025/11/24 10:33:38

```
['Adam', 'Lisa', 'Bart']  
测试成功!  
3 * 5 * 7 * 9 = 945  
测试成功!  
str2float('123.456') = 123.456  
测试成功!
```

进程已结束，退出代码为 0

杨志强 @ 2025/11/22 22:04:50



```
from functools import reduce  
  
def str2float(s):  
    DIGITS = {'0': 0, '1': 1, '2': 2, '3': 3, '4': 4, '5': 5, '6': 6, '7': 7, '8': 8, '9': 9}  
  
    n = 0  
    for i in s:  
        if i == '.':  
            break  
        n = n + 1  
  
    def char2num(x):  
        if x in DIGITS:  
            return DIGITS[x]  
  
    def f(x, v):
```

[Read More ▾](#)

待绝笔墨痕干 @ 2025/11/16 20:55:14



```
from functools import reduce  
  
def normalize(name):  
    lower_name=name.lower()  
    return lower_name.capitalize()  
  
# 测试:  
L1 = ['adam', 'LISA', 'barT']  
L2 = list(map(normalize, L1))
```

- 7.4. 生成器
- 7.5. 迭代器
- 8. 函数式编程
- 8.1. 高阶函数
- 8.1.1. map/reduce**
- 8.1.2. filter
- 8.1.3. sorted
- 8.2. 返回函数
- 8.3. 匿名函数
- 8.4. 装饰器
- 8.5. 偏函数
- 9. 模块
- 9.1. 使用模块
- 9.2. 安装第三方模块
- 10. 面向对象编程
- 10.1. 类和实例
- 10.2. 访问限制
- 10.3. 继承和多态
- 10.4. 获取对象信息
- 10.5. 实例属性和类属性
- 11. 面向对象高级编程
- 11.1. 使用_slots_
- 11.2. 使用@property
- 11.3. 多重继承
- 11.4. 定制类
- 11.5. 使用枚举类
- 11.6. 使用元类
- 12. 错误、调试和测试
- 12.1. 错误处理
- 12.2. 调试
- 12.3. 单元测试
- 12.4. 文档测试
- 13. IO编程
- 14. 进程和线程
- 15. 正则表达式
- 16. 常用内建模块
- 17. 常用第三方模块
- 18. 图形界面
- 19. 网络编程
- 20. 电子邮件
- 21. 访问数据库
- 22. Web开发
- 23. 异步IO
- 24. FAQ
- 25. 期末总结

```

print(L2)

def prod(L):
    def fn(x,y):
        return x*y
    return reduce(fn, L)

```

[Read More ▾](#)

无泷 @ 2025/11/15 13:23:29

```

def normalize(name): return name.capitalize()

L1 = ['adam', 'LISA', 'barT'] L2 = list(map(normalize, L1)) print(L2)

```

Meow @ 2025/11/14 04:31:20

```

from functools import reduce

def str2float(s):
    # 划分成'123'和'456'
    left_str, right_str = s.split('.')

    # 将字符串转为整数
    def char2num(c):
        return ord(c) - ord('0')

    # 计算整数部分的数值
    left_value = reduce(lambda x, y: x * 10 + y, map(char2num, left_str))

    # 计算小数部分的数值
    right_value = reduce(lambda x, y: x * 10 + y, map(char2num, right_str)) / (10 ** len(right_str))

```

[Read More ▾](#)

残梦往事 @ 2025/11/12 10:01:10

```

def str2float(s):
    if '.' not in s:
        return reduce(lambda x, y: x * 10 + y, map(lambda x: DIGITS[x], s))
    else:
        integer_str = s.split('.')[0]
        decimal_str = s.split('.')[1]
        print(decimal_str)
        return (reduce(lambda x, y: x * 10 + y, map(lambda x: DIGITS[x], integer_str))
                + 0.1 * reduce(lambda x, y: x * 10 + y, map(lambda x: DIGITS[x], decimal_str)))

```

◀
▶

keep going @ 2025/11/12 07:24:39

```

L1 = ['adam', 'LISA', 'barT'] def normolize(m): return m[0].upper() + m[1:].lower()

L2 = list(map(normolize,L1))

print(L2)

```

keep going @ 2025/11/12 08:26:13

要强迫map()完成计算结果！！！！！ L2 = map(normolize , L1) list(L2) 终端显示: <map object at 0x73dbd238c280>

©liaoxuefeng.com - 微博 - GitHub - License