

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

匿名函数



廖雪峰

资深软件开发工程师，业余马拉松选手。

当我们在传入函数时，有些时候，不需要显式地定义函数，直接传入匿名函数更方便。

在Python中，对匿名函数提供了有限支持。还是以 `map()` 函数为例，计算 $f(x)=x^2$ 时，除了定义一个 `f(x)` 的函数外，还可以直接传入匿名函数：

```
>>> list(map(lambda x: x * x, [1, 2, 3, 4, 5, 6, 7, 8, 9]))
[1, 4, 9, 16, 25, 36, 49, 64, 81]
```

通过对比可以看出，匿名函数 `lambda x: x * x` 实际上就是：

```
def f(x):
    return x * x
```

关键字 `lambda` 表示匿名函数，冒号前面的 `x` 表示函数参数。

匿名函数有个限制，就是只能有一个表达式，不用写 `return`，返回值就是该表达式的结果。

用匿名函数有个好处，因为函数没有名字，不必担心函数名冲突。此外，匿名函数也是一个函数对象，也可以把匿名函数赋值给一个变量，再利用变量来调用该函数：

```
>>> f = lambda x: x * x
>>> f
<function <lambda> at 0x101c6ef28>
>>> f(5)
25
```

同样，也可以把匿名函数作为返回值返回，比如：

```
def build(x, y):
    return lambda: x * x + y * y
```

练习

请用匿名函数改造下面的代码：

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

L = list(filter(is_odd, range(1, 20)))

print(L)
```

小结

Python对匿名函数的支持有限，只有一些简单的情况下可以使用匿名函数。

« 返回函数

装饰器 »

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

gitee | 企业版

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](#)



海海海 @ 2025/12/21 20:36:37

```
L=list(filter(lambda n:n % 2 ==1,range(1,20)))
```



杨彦 @ 2025/12/18 07:54:35

```
L = list(filter(lambda x:x%2==1 , range(1, 20))) print(L)
```



啊，你以为我傻 @ 2025/12/11 05:08:09

```
L = list(filter(lambda n: n % 2 == 1,range(1,20)))
```



Hypersomnia @ 2025/12/1 01:43:27

```
L = list(filter(lambda x: x % 2 == 1, range(1, 20)))
```



多得多 @ 2025/11/13 07:37:02

```
L = list(filter(lambda n:n%2 == 1,range(1,20)))  
print(L)
```



大陆 @ 2025/11/2 00:59:09

```
L = list(filter(lambda n: (n % 2) == 1, range(1, 20)))
```



Star Platinum @ 2025/10/24 04:00:07

```
L = list(filter(lambda n: n % 2 == 1, range(1, 20))) print(L)
```



bot吃宵夜 @ 2025/10/14 05:10:07

```
L = list(filter(lambda x: x % 2 == 1, range(1, 20)))
```

```
print(L)
```



地球村 @ 2025/9/27 07:06:57

```
L = list(filter(lambda x : x % 2 == 1, range(1,20)))  
print(L)
```



DF11-0045 @ 2025/9/25 04:13:08

```
L = list(filter(lambda n: n % 2 == 1 , range(1, 20)))
```

print(L)

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



.. @ 2025/9/24 09:54:26

```
L=list(filter(lambda x:x%2==1,range(1,20))) print(L)
```



greensea @ 2025/9/18 22:49:29

```
L = list(filter(lambda n : n % 2 == 1, range(1, 20)))  
  
print(L)
```



子不语 @ 2025/9/15 22:28:49

```
L = list(filter(lambda n:n % 2 > 0,range(1,20))) print(L)
```



k桦 @ 2025/9/15 03:55:40

```
L=list(filter(lambda x:x%2==1,range(1,20)))  
print(L)
```



Coisini @ 2025/9/13 22:22:38

```
L = list(filter(lambda x:x % 2 == 1,range(1,20)))  
print(L)
```



Fredmars @ 2025/9/9 07:26:36

```
L = list(filter(lambda x: x % 2 == 1,range(1,20)))
```



makabaka @ 2025/9/3 04:47:15

```
L = list(filter(lambda n : n % 2 == 1, range(1,20) ))
```



.. @ 2025/8/31 19:11:12

```
print(list(filter(lambda n: n % 2 == 1,range(1,20))))
```



王大可 @ 2025/8/29 03:04:08

```
L = list(filter(lambda n:n % 2 ==1, range(1, 20)))  
print(L)
```



钟旭 @ 2025/8/16 05:13:27

```
>>> list(filter(lambda x:x%2 == 1, range(1, 20)))  
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```