

5.7. 使用dict和set

6. 函数

6.1. 调用函数

6.2. 定义函数

6.3. 函数的参数

6.4. 递归函数

7. 高级特性

7.1. 切片

7.2. 迭代

7.3. 列表生成式

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

sorted



廖雪峰



资深软件开发工程师，业余马拉松选手。

排序也是在程序中经常用到的算法。无论使用冒泡排序还是快速排序，排序的核心是比较两个元素的大小。如果是数字，我们可以直接比较，但如果是字符串或者两个dict呢？直接比较数学上的大小是没有意义的，因此，比较的过程必须通过函数抽象出来。

Python内置的 `sorted()` 函数就可以对list进行排序：

```
>>> sorted([36, 5, -12, 9, -21])
[-21, -12, 5, 9, 36]
```

此外，`sorted()` 函数也是一个高阶函数，它还可以接收一个 `key` 函数来实现自定义的排序，例如按绝对值大小排序：

```
>>> sorted([36, 5, -12, 9, -21], key=abs)
[5, 9, -12, -21, 36]
```

`key` 指定的函数将作用于list的每一个元素上，并根据`key`函数返回的结果进行排序。对比原始的list和经过 `key=abs` 处理过的list：

```
list = [36, 5, -12, 9, -21]
keys = [36, 5, 12, 9, 21]
```

然后 `sorted()` 函数按照`keys`进行排序，并按照对应关系返回list相应的元素：

```
keys sort  => [5, 9, 12, 21, 36]
              | | | | |
result sort => [5, 9, -12, -21, 36]
```

我们再看一个字符串排序的例子：

```
>>> sorted(['bob', 'about', 'Zoo', 'Credit'])
['Credit', 'Zoo', 'about', 'bob']
```

默认情况下，对字符串排序，是按照ASCII的大小比较的，由于 `'z' < 'a'`，结果，大写字母 `z` 会排在小写字母 `a` 的前面。

现在，我们提出排序应该忽略大小写，按照字母序排序。要实现这个算法，不必对现有代码大加改动，只要我们能用一个 `key` 函数把字符串映射为忽略大小写排序即可。忽略大小写来比较两个字符串，实际上就是先把字符串都变成大写（或者都变成小写），再比较。

这样，我们给 `sorted` 传入`key`函数，即可实现忽略大小写的排序：

```
>>> sorted(['bob', 'about', 'Zoo', 'Credit'], key=str.lower)
['about', 'bob', 'Credit', 'Zoo']
```

要进行反向排序，不必改动`key`函数，可以传入第三个参数 `reverse=True`：

```
>>> sorted(['bob', 'about', 'Zoo', 'Credit'], key=str.lower, reverse=True)
['Zoo', 'Credit', 'bob', 'about']
```

从上述例子可以看出，高阶函数的抽象能力是非常强大的，而且，核心代码可以保持得非常简洁。

小结

`sorted()` 也是一个高阶函数。用 `sorted()` 排序的关键在于实现一个映射函数。

练习

假设我们用一组tuple表示学生名字和成绩：

```
L = [('Bob', 75), ('Adam', 92), ('Bart', 66), ('Lisa', 88)]
```

请用 `sorted()` 对上述列表分别按名字排序：

```
L = [('Bob', 75), ('Adam', 92), ('Bart', 66), ('Lisa', 88)]

def by_name(t):
    pass

L2 = sorted(L, key=by_name)
print(L2)
```

再按成绩从高到低排序：

```
L = [('Bob', 75), ('Adam', 92), ('Bart', 66), ('Lisa', 88)]

def by_score(t):
    pass

L2 = sorted(L, key=by_score)
print(L2)
```

参考源码

[do_sorted.py](#)

« filter

[返回函数](#) »



Comments

Loading comments...

©liaoxuefeng.com - 微博 - GitHub - License

5.7. 使用dict和set

6. 函数

6.1. 调用函数

6.2. 定义函数

6.3. 函数的参数

6.4. 递归函数

7. 高级特性

7.1. 切片

7.2. 迭代

7.3. 列表生成式

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