Python语言基础与应用

高级特性 / 自定义对象的排序

陈斌 北京大学 gischen@pku.edu.cn

自定义对象的排序

- 列表排序
- 内置排序函数
- 特殊方法
- 可扩展的“大小”比较及排序

列表排序

列表方法sort()

- 对原列表进行排序，改变原列表内容

  如果列表中的元素都是数字，默认按升序排序

  通过添加参数reverse = True可改为降序排列

```
>>> num = [4, 2, 7, 0, 1]
>>> num.sort()
>>> num
[0, 1, 2, 4, 7]
>>> num.sort(reverse = True)
>>> num
[7, 4, 2, 1, 0]
```

  如果元素都是字符串，则会按照字母表顺序排列

```
>>> name = ['John','Connor','Bruce','Arthur','Edward']
>>> name.sort()
>>> name
['Arthur', 'Bruce', 'Connor', 'Edward', 'John']
```

内置排序函数

通用函数sorted()

类似sort()，但返回的是排好序的列表副本，原列表内容不变

```
>>> name = ['John','Connor','Bruce','Arthur','Edward']
>>> sorted_name = sorted(name)
>>> sorted_name
['Arthur', 'Bruce', 'Connor', 'Edward', 'John']
>>> name
['John', 'Connor', 'Bruce', 'Arthur', 'Edward']
```

只有当列表中的所有元素都是同一种类型时，sort()和sorted()才会正常工作

特殊方法

特殊方法__lt__

- 由于Python的可扩展性，每种数据类型可以定义特殊方法

```
def __lt__(self, y)
```

返回True视为比y“小”，排在前

返回False视为比y“大”，排在后

- 只要类定义中定义了特殊方法__lt__，任何自定义类都可以使用x<y这样的比较

可扩展的“大小”比较及排序

- 例子：Student

  姓名name，成绩grade

- 按照成绩由高到低排序

```
class Student:
    def __init__(self, name, grade):
        self.name, self.grade = name, grade

    # 内置sort函数只引用 < 比较符来判断前后
    def __lt__(self, other):
        # 成绩比other高的，排在他前面
        return self.grade > other.grade

    # Student的易读字符串表示
    def __str__(self):
        return "(%s,%d)" % (self.name, self.grade)

    # Student的正式字符串表示，我们让它跟易读表示相同
    __repr__ = __str__
```

可扩展的“大小”比较及排序

- 构造一个列表，加入Student对象

```
# 构造一个Python List对象
s = list()

# 添加Student对象到List中
s.append(Student("Jack", 80))
s.append(Student("Jane", 75))
s.append(Student("Smith", 82))
s.append(Student("Cook", 90))
s.append(Student("Tom", 70))
print("Original:", s)

# 对List进行排序，注意这是内置sort方法
s.sort()

# 查看结果，已经按照成绩排好序
print("Sorted:", s)
```

可扩展的“大小”比较及排序

- **直接调用列表sort方法**

  可以根据__lt__定义排序

- **直接检验Student对象的大小**

  S[i]<s[j]

- **另外可以定义其它比较符**

  __gt__等

```
============= RESTART: /Users/chenbin/Documents/homework/stu.py =
Original: [(Jack,80), (Jane,75), (Smith,82), (Cook,90), (Tom,70)]
Sorted: [(Cook,90), (Smith,82), (Jack,80), (Jane,75), (Tom,70)]
>>> s[0]<s[1]
True
>>>
```

可扩展的“大小”比较及排序

重新定义__lt__方法，改为比较姓名

这样sort方法就能按照姓名来排序

```
class Student:
    def __init__(self, name, grade):
        self.name, self.grade = name, grade

    # 内置sort函数只引用 < 比较符来判断前后
    def __lt__(self, other):
        # 姓名字母顺序在前，就排在他前面
        return self.name < other.name

    # Student的易读字符串表示
    def __str__(self):
        return "(%s,%d)" % (self.name, self.grade)

    # Student的正式字符串表示，我们让它跟易读表示相同
    __repr__ = __str__
```

```
============= RESTART: /Users/chenbin/Documents/homework/stu2.py =
Original: [(Jack,80), (Jane,75), (Smith,82), (Cook,90), (Tom,70)]
Sorted: [(Cook,90), (Jack,80), (Jane,75), (Smith,82), (Tom,70)]
>>> s[0]<s[1]
True
>>>
```