2017-06-07 Wed 16:11, by 田宇伟

排序和搜索

1 list.sort 方法和 sorted 函数

list.sort 方法会就地排序列表，所以这个方法的返回值是 None 。

如果一个函数或者方法对对象进行的是就地改动，那它就应该返回 None ，让调用者知道传入的参数发生了变动，并且未产生新的对象。

与 list.sort 不同的是内置函数 sorted ，其会新建一个列表作为返回值。这个方法可以接收任何形式的可迭代对象作为参数，包括不可变序列或者生成器，不管其接收的是怎样的参数，它最后都返回一个列表。

list.sort 和 sorted 都有2个可选的关键字参数：

reverse
: 如果设定为 True ，被排序的序列里的元素会以降序输出（把最大值当作最小值来排序）。此参数默认值是 False 。

key
: 一个只有一个参数的函数，这个函数会被用在序列里的每一个元素上，所产生的结果将是排序算法依赖的对比关键字。这个参数的默认值是恒等函数，即默认用元素自己的值来排序。

```
fruits = ['apple', 'banana', 'pear', 'raspberry', 'strawberry']
print(sorted(fruits))
print(sorted(fruits, reverse=True))
print(sorted(fruits, key=len))
print(sorted(fruits, key=len, reverse=True))
print(fruits)
fruits.sort()
print(fruits)
```

```
['apple', 'banana', 'pear', 'raspberry', 'strawberry']
['strawberry', 'raspberry', 'pear', 'banana', 'apple']
['pear', 'apple', 'banana', 'raspberry', 'strawberry']
['strawberry', 'raspberry', 'banana', 'apple', 'pear']
['apple', 'banana', 'pear', 'raspberry', 'strawberry']
['apple', 'banana', 'pear', 'raspberry', 'strawberry']
```

2 使用 bisect 模块来管理已排序序列

已排序的序列可以用来进行快速搜索，标准库的 bisect 模块提供了二分查找算法。

bisect 模块包含两个主要函数， bisect 和 insort ，两个函数都利用二分查找算法来在有序序列中查找或插入元素。

2.1 用 bisect 来搜索

bisect(haystack, needle) 在 haystack （干草垛）里搜索 needle （针）的位置，该位置满足的条件是，把 needle 插入到这个位置后， haystack 还能保持升序，即此函数返回的位置前面的值，都小于或等于 needle 的值。其中 haystack 必须是一个有序的序列。

可以先用 bisect(haystack, needle) 查找位置 index ，再用 haystack.insert(index, needle) 来插入新值。或者用 insort 来一步到位，速度会更快一些。

```
# bisect_demo.py

import bisect
import sys

HAYSTACK = [1, 4, 5, 6, 8, 12, 15, 20, 21, 23, 23, 26, 29, 30]
NEEDLES = [0, 1, 2, 5, 8, 10, 22, 23, 29, 30, 31]

ROW_FMT = '{0:2d} @ {1:2d}    {2}{0:<2d}'


def demo(bisect_fn):
    for needle in reversed(NEEDLES):
        position = bisect_fn(HAYSTACK, needle)
```

```
        offset = position * '  |'
        print(ROW_FMT.format(needle, position, offset))


if __name__ == '__main__':
    if sys.argv[-1] == 'left':
        bisect_fn = bisect.bisect_left
    else:
        bisect_fn = bisect.bisect

    print('DEMO:', bisect_fn.__name__)
    print('haystack ->', ' '.join('%2d' % n for n in HAYSTACK))
    demo(bisect_fn)
```

(demo)

```
$ python3 bisect_demo.py
DEMO: bisect
haystack ->  1  4  5  6  8 12 15 20 21 23 23 26 29 30
31 @ 14      |  |  |  |  |  |  |  |  |  |  |  |  |  |31
30 @ 14      |  |  |  |  |  |  |  |  |  |  |  |  |  |30
29 @ 13      |  |  |  |  |  |  |  |  |  |  |  |  |29
23 @ 11      |  |  |  |  |  |  |  |  |  |  |23
22 @  9      |  |  |  |  |  |  |  |  |22
10 @  5      |  |  |  |  |10
 8 @  5      |  |  |  |  |8
 5 @  3      |  |  |5
 2 @  1      |2
 1 @  1      |1
 0 @  0    0
```

```
$ python3 bisect_demo.py left
DEMO: bisect_left
haystack ->  1  4  5  6  8 12 15 20 21 23 23 26 29 30
31 @ 14      |  |  |  |  |  |  |  |  |  |  |  |  |  |31
30 @ 13      |  |  |  |  |  |  |  |  |  |  |  |  |30
29 @ 12      |  |  |  |  |  |  |  |  |  |  |  |29
23 @  9      |  |  |  |  |  |  |  |  |23
22 @  9      |  |  |  |  |  |  |  |  |22
10 @  5      |  |  |  |  |10
 8 @  4      |  |  |  |8
 5 @  2      |  |5
 2 @  1      |2
 1 @  0    1
 0 @  0    0
```

`bisect` 的表现可以从两个方面来调整。

1. 用它的两个可选参数——`lo` 和 `hi` ——来缩小搜寻范围。`lo` 的默认值是0，`hi` 的默认值是序列的长度。
2. `bisect` 起始是 `bisect_right` 的别名，对应的函数是 `bisect_left` 。

`bisect` 可用来建立一个用数字作为索引的查询表格，比如把分数和成绩对应起来。

```
import bisect


def grade(score, breakpoints=[60, 70, 80, 90], grades='FDCBA'):
    i = bisect.bisect(breakpoints, score)
    return grades[i]

print([grade(score) for score in [33, 99, 77, 70, 89, 90, 100]])
```

```
['F', 'A', 'C', 'C', 'B', 'A', 'A']
```

2.2 用 `bisect.insort` 插入新元素

`insort(seq, item)` 把变量 `item` 插入到序列 `seq` 中，并能保持 `seq` 的升序顺序。

```
import bisect
import random

SIZE = 7

random.seed(1730)

my_list = []
```

```
for i in range(SIZE):
    new_item = random.randrange(SIZE*2)
    bisect.insort(my_list, new_item)
    print('%2d ->' % new_item, my_list)
```

```
 3 -> [3]
13 -> [3, 13]
 5 -> [3, 5, 13]
 2 -> [2, 3, 5, 13]
 8 -> [2, 3, 5, 8, 13]
12 -> [2, 3, 5, 8, 12, 13]
 6 -> [2, 3, 5, 6, 8, 12, 13]
```

insort 跟 bisect 一样，有 lo 和 hi 两个可选参数用来控制查找的范围。它也有个变体叫 insort_left ，这个变体在背后用的是 bisect_left 。

Date: 2017-06-07 Wed 16:11

Author: 田宇伟