bisect — Maintain Lists in Sorted Order

Purpose: Maintains a list in sorted order without having to call sort each time an item is added to the list.

The `bisect` module implements an algorithm for inserting elements into a list while maintaining the list in sorted order.

Inserting in Sorted Order

Here is a simple example in which `insort()` is used to insert items into a list in sorted order.

```
# bisect_example.py

import bisect

# A series of random numbers
values = [14, 85, 77, 26, 50, 45, 66, 79, 10, 3, 84, 77, 1]

print('New  Pos  Contents')
print('---  ---  --------')

l = []
for i in values:
    position = bisect.bisect(l, i)
    bisect.insort(l, i)
    print('{:3}  {:3}'.format(i, position), l)
```

The first column of the output shows the new random number. The second column shows the position where the number will be inserted into the list. The remainder of each line is the current sorted list.

```
$ python3 bisect_example.py

New  Pos  Contents
---  ---  --------
 14    0 [14]
 85    1 [14, 85]
 77    1 [14, 77, 85]
 26    1 [14, 26, 77, 85]
 50    2 [14, 26, 50, 77, 85]
 45    2 [14, 26, 45, 50, 77, 85]
 66    4 [14, 26, 45, 50, 66, 77, 85]
 79    6 [14, 26, 45, 50, 66, 77, 79, 85]
 10    0 [10, 14, 26, 45, 50, 66, 77, 79, 85]
  3    0 [3, 10, 14, 26, 45, 50, 66, 77, 79, 85]
 84    9 [3, 10, 14, 26, 45, 50, 66, 77, 79, 84, 85]
 77    8 [3, 10, 14, 26, 45, 50, 66, 77, 77, 79, 84, 85]
  1    0 [1, 3, 10, 14, 26, 45, 50, 66, 77, 77, 79, 84, 85]
```

This is a simple example,. In fact, given the amount of data being manipulated, it might be faster to simply build the list and then sort it once. By contrast, for long lists, significant time and memory savings can be achieved using an insertion sort algorithm such as this, especially when the operation to compare two members of the list requires expensive computation.

Handling Duplicates

The result set shown previously includes a repeated value, 77. The `bisect` module provides two ways to handle repeats: New values can be inserted either to the left of existing values, or to the right. The `insort()` function is actually an alias for `insort_right()`, which inserts an item after the existing value. The corresponding function `insort_left()` inserts an item before the existing value.

```
# bisect_example2.py

import bisect
```

```
# A series of random numbers
values = [14, 85, 77, 26, 50, 45, 66, 79, 10, 3, 84, 77, 1]

print('New  Pos  Contents')
print('---  ---  --------')

# Use bisect_left and insort_left.
l = []
for i in values:
    position = bisect.bisect_left(l, i)
    bisect.insort_left(l, i)
    print('{:3}  {:3}'.format(i, position), l)
```

When the same data is manipulated using `bisect_left()` and `insort_left()`, the results are the same sorted list but the insert positions are different for the duplicate values.

```
$ python3 bisect_example2.py

New  Pos  Contents
---  ---  --------
 14    0 [14]
 85    1 [14, 85]
 77    1 [14, 77, 85]
 26    1 [14, 26, 77, 85]
 50    2 [14, 26, 50, 77, 85]
 45    2 [14, 26, 45, 50, 77, 85]
 66    4 [14, 26, 45, 50, 66, 77, 85]
 79    6 [14, 26, 45, 50, 66, 77, 79, 85]
 10    0 [10, 14, 26, 45, 50, 66, 77, 79, 85]
  3    0 [3, 10, 14, 26, 45, 50, 66, 77, 79, 85]
 84    9 [3, 10, 14, 26, 45, 50, 66, 77, 79, 84, 85]
 77    7 [3, 10, 14, 26, 45, 50, 66, 77, 77, 79, 84, 85]
  1    0 [1, 3, 10, 14, 26, 45, 50, 66, 77, 77, 79, 84, 85]
```

See also

- Standard library documentation for bisect
- Wikipedia: Insertion Sort – A description of the insertion sort algorithm.

heapq – Heap Sort Algorithm

queue — Thread-Safe FIFO Implementation

Quick Links
Inserting in Sorted Order
Handling Duplicates

This page was last updated 2017-01-28.

Navigation
heapq – Heap Sort Algorithm
queue — Thread-Safe FIFO Implementation

The output from all the example programs from PyMOTW-3 has been generated with Python 3.7.1, unless otherwise noted. Some of the features described here may not be available in earlier versions of Python.

Looking for examples for Python 2?

This Site

Module Index
Index

Other Writing

Blog
The Python Standard Library By Example