heapq - Heap queue algorithm

heapq 模块提供了堆算法。heapq是一种子节点和父节点排序的树形数据结构。这个模块提供**heap[k] <= heap[2\*k+1] and heap[k] <= heap[2\*k+2]**。为了比较不存在的元素被人为是无限大的。heap最小的元素总是[0]。

**heapify**(...)

#Transform list into a heap, in-place, in O(len(heap)) time.

heappop(...)

#删除并返回堆中最小的元素

heappush(...)

#heappush(heap, item) -> None. Push item onto heap, maintaining the heap invariant.

heappushpop(...)

#heappushpop(heap, item) -> value. Push item on the heap, then pop and return the smallest item

from the heap. The combined action runs more efficiently than heappush() followed by a separate call to heappop().

heapreplace(heap, item)

#Pop and return the current smallest value, and add the new item.

**merge**(\*iterables, key=None, reverse=False)

#Merge multiple sorted inputs into a single sorted output.

Similar to sorted(**itertools.chain(\*iterables)**) but returns a generator,

>>> list(merge([1,3,5,7], [0,2,4,8], [5,10,15,20], [], [25]))

[0, 1, 2, 3, 4, 5, 5, 7, 8, 10, 15, 20, 25]

>>> list(merge(['dog', 'horse'], ['cat', 'fish', 'kangaroo'], **key=len**))

['dog', 'cat', 'fish', 'horse', 'kangaroo']

**nlargest(n, iterable, key=None)**

#返回列表中最大的n个值

Equivalent to: sorted(iterable, key=key, reverse=True)[:n]

**nsmallest(n, iterable, key=None)**

#返回列表中最小的n个值

Equivalent to: sorted(iterable, key=key)[:n]

>>>portfolio **=** [

{'name': 'IBM', 'shares': 100, 'price': 91.1},

{'name': 'AAPL', 'shares': 50, 'price': 543.22},

{'name': 'FB', 'shares': 200, 'price': 21.09},

{'name': 'HPQ', 'shares': 35, 'price': 31.75},

{'name': 'YHOO', 'shares': 45, 'price': 16.35},

{'name': 'ACME', 'shares': 75, 'price': 115.65}

]

>>>cheap **=** heapq**.**nsmallest(3, portfolio, key**=lambda** s: s['price'])

>>>expensive **=** heapq**.**nlargest(3, portfolio, key**=lambda** s: s['price'])