python自带的功能库

1. math

向上取整：

```python
import math

# 示例数值
number = 3.7

# 向上取整
rounded_up = math.ceil(number)

print(rounded_up)  # 输出将会是 4
```

2. collections

collections 中有一个defaultdict，可以指定字典的 value 的默认数据类型，很方便，特别是当 dict 的 value 是 list 时。

```python
dict = defaultdict(list)
# {key:list}
```

3. bisect

在生成 list 时，如果直接在有序列表中插入一个元素并保持列表的有序性，可以使用 `bisect` 模块提供的 `insort` 函数，，实现有序数组的生成：

```python
import bisect

# 创建一个有序列表
my_list = [1, 2, 4, 5]

# 使用insort_left在列表中插入元素3
bisect.insort_left(my_list, 3)

# 使用insort_right在列表中插入元素2
bisect.insort_right(my_list, 2)

print(my_list)  # 输出: [1, 2, 2, 3, 4, 5]
```

而且相当强大的是，只要这个 list 中的元素是根据某种规律排序，bisec 就可以实现插入：

```python
# 假设你的列表是根据第一个元素（key）排序的
sorted_list = [[1, 'a'], [2, 'b'], [4, 'd'], [5, 'e']]

key_to_find = 3
bisect.insort(sorted_list, [3,'f'])
print(sorted_list)
```

3.1. 二分查找（lower_bound）功能的实现

lower_bound 是 cpp 中一个可以在有序容器中返回不大于 x 的值的元素的位置。

类似的功能可以使用bisect 的bisect_left 函数，找到需要插入的位置：

```python
import bisect

# 创建一个有序列表
my_list = [1, 2, 4, 5]

# 使用bisect找到插入位置
index = bisect.bisect_left(my_list, 3)

# 插入元素
my_list.insert(index, 3)

print(my_list)  # 输出: [1, 2, 3, 4, 5]
```

而且相当强大的是，只要这个 list 中的元素是根据某种规律排序，bisec 就可以实现查找：

```python
# 假设你的列表是根据第一个元素（key）排序的
sorted_list = [[1, 'a'], [2, 'b'], [4, 'd'], [5, 'e']]

key_to_find = 3
index = bisect.bisect_left(sorted_list, [key_to_find], key=lambda x: x[0])

# 输出插入位置
print(index)  # 返回 2

```

4. itertools