数据类型的区别

- 常用内置数据类型
  - 定义:
    - list: 链表,有序的项目, 通过索引进行查找,使用方括号”[]”；
    - tuple: 元组,元组将多样的对象集合到一起,不能修改,通过索引进行查找, 使用括号”()”；
    - dict: 字典,字典是一组键(key)和值(value)的组合,通过键(key)进行查找,没有顺序,使用大括号”{}”；
    - set: 集合,无序,元素只出现一次, 自动去重,使用”set([])”
  - 应用场景:
    - list, 简单的数据集合,可以使用索引;
    - tuple , 把一些数据当做一个整体去使用,不能修改;
    - dict, 使用键值和值进行关联的数据;
    - set, 数据只出现一次,只关心数据是否出现, 不关心其位置;
  - 代码:
    - mylist = [1, 2, 3, 4, 'Oh']
    - mytuple = (1, 2, 'Hello', (4, 5))
    - mydict = {'Wang' : 1, 'Hu' : 2, 'Liu' : 4}
    - myset = set(['Wang', 'Hu', 'Liu', 4, 'Wang'])
    - print(mylist)
    - print(mytuple)
    - print(mydict)
    - print(myset)
  - 输出:
    - [1, 2, 3, 4, 'Oh']
    - (1, 2, 'Hello', (4, 5))
    - {'Liu': 4, 'Wang': 1, 'Hu': 2}
    - set(['Liu', 4, 'Wang', 'Hu'])
- collections模块提供其他的数据类型
  - namedtuple:
    - 生成可以使用名字来访问元素内容的tuple
    - namedtuple('名称', [属性list])

      point=namedtuple("point",['x','y'])
  - deque:
    - 双端队列，可以快速的从另外一侧追加和推出对象,适合用于队列和栈
    - xx=deque(iter,maxlen)

      qu=deque(['a','b','c'],maxlen=4)
    - 常用方法
      - append(),appendleft();添加元素
      - clear()；清除

        - count();计数
        - extend(),extendleft();扩展
        - index();索引
        - pop(),popleft();出栈
        - remove();移除
        - reverse();翻转
        - rotate();循环位移
    - Counter: 计数器，主要用来计数

      c=Counter("obj")
        - 常用方法
            - most_common(n);列出频率最高的n个元素
            - sorted(c);分类出元素
            - sum(c.values());求总数
    - OrderedDict: 有序字典

      oD = OrderedDict(items)
    - defaultdict: 带有默认值的字典

      使用defaultdict，只要你传入一个默认的工厂方法，那么请求一个不存在的key时， 便会调用这个工厂方法使用其结果来作为这个key的默认值。
- heapq模块提供了堆的数据结构

  heap是一个小顶堆
    - 常用方法
        - heapq.heappush(heap, item) 把item添加到heap中（heap是一个列表）
        - heapq.heappop(heap) 把堆顶元素弹出，返回的就是堆顶
        - heapq.heappushpop(heap, item) 先把item加入到堆中，然后再pop，比heappush()再heappop()要快得多
        - heapq.heapreplace(heap, item) 先pop，然后再把item加入到堆中，比heappop()再heappush()要快得多
        - heapq.heapify(x) 将列表x进行堆调整，默认的是小顶堆
        - heapq.merge(*iterables) 将多个列表合并，并进行堆调整，返回的是合并后的列表的迭代器
        - heapq.nlargest(n, iterable, key=None) 返回最大的n个元素（Top-K问题）
        - heapq.nsmallest(n, iterable, key=None) 返回最小的n个元素（Top-K问题）