（一）集合

世界上随机没有纯随机，只有伪随机。如果集合的随机整数结果一直不变，这是因为底层算法实现的随机是伪随机。请不要抬杠，因为伪随机也是随机。

1.定义

无序，元素不重复，可修改，可容纳不同数据类型的数据容器
形式：变量 = {元素1，元素2，元素3，....}

2. 空集合

变量名 = set( )

3. 常用方法

1.增加：
add():

集合.add(元素)

将元素加入集合中，如果集合中有该元素，则只保留一个该元素。没有则加入

```
se = {1,2,3}
se.add(4)
print(se)#包含1,2,3,4的集合 位置随机（集合的特点：无序）
```

2.删除

pop()

集合.pop()

将集合的元素随机删除，并返回一个结果

```
se = {1,2,3}
s = dic.pop()
print(s)
ptint(se)
```

remove()

集合.remove(元素)

将该元素在集合中删除

```
se = {1,2,3}
se.remove(1)
print(se)#包含2,3的集合，位置随机
```

3.清空

clear()

集合.clear()

清空集合

```
se = {1,2,3}
se.clear()
print(se)#结果为 set()，表示空集合
```

4.差集

difference()

集合1.difference(集合2)

求出集合1、集合2的差集(将集合1中含有的集合2中的元素进行删除)，并返回一个新集合

```
se1 = {1,2,3}
se2 = {3,4,5}
se3 = se1.difference(se2)
print(se1)#含有1,2,3的集合，位置随机
print(se2)#含有3,4,5的集合，位置随机
print(se3)#含有1,2的集合，位置随机
```

difference_update()
集合1.difference_update(集合2)
在集合1中，将集合2的元素进行删除

```
se1 = {1,2,3}
se2 = {3,4,5}
se1.difference(se2)
print(se1) #返回含有1,2的集合，位置随机
```

5.并集

union()

集合1.union(集合2)

将集合1和集合2合并，并返回一个新集合

```
se1 = {1,2,3}
se2 = {2,3,4}
se3 = se1.union(se2)
print(se1)#含有1,2,3的集合
print(se2)#含有2,3,4的集合
print(se3)#含有1,2,3,4,5的集合
```

6.统计长度

len()

len(集合)

统计集合的长度

```
se = {1,2,3}
length = len(se)
print(length)#结果为3
```

7.循环

只支持for循环，不支持while循环

```
for i in set:
print(i)
```

(二) 字典

1. 定义

无序，不可重复，可修改（其中，不可直接对key值进行修改，需要先删除原key,再添加新key）具有key-value键值对的数据容器。其中key必须是不可以变的数据容器，值可以是任意数据类型。

2. 空字典

dic = {}
dic = dict()

3. 常用方法

1.添加/更新

字典[key] = 值

将新的key-value键值对加入字典中，如果字典中没有该键值对，则增加键值对。否则，覆盖该键值对.

添加:

```
dic = {"one":1,"two":2}
dic["three"] = 3
print(dic)#结果为{"one":1,"two":2,"three":3}
```

更新

```
dic = {"one":1,"two":2,"three":3}
dic["one"] = 4
print(dic)#结果为{'one': 4, 'two': 2, 'three': 3}
```

2.删除

pop()

字典.pop(key)

删除key对应的key-value键值对,并返回被删除的键值对的值,类型为值的数据类型

```
dic = {"one":1,"two":2,"three":[1,2,3]}
dele = dic.pop("three")
print(dic)#结果为{'one':1,'two':2}
print(dele)#[1,2,3]
print(type(dele))#<class 'list'>
```

3.清空

clear()

字典.clear()

清空字典

```
dic = {"one":1,"two":2}
dic.clear()
print(dic) #结果为{}
```

4.获取所有的键(key)

keys()

字典.keys()

获取字典的所有的键,返回的类型为dict_keys

```
dic = {"one":1,"two":2}
kes = dic.keys()
print(kes)#dict_keys(['one','two'])
```

5.统计长度

len()

len(字典)

获取字典的长度

```
dic = {"one":1,"two":2}
length = len(dic)
print(length) #结果为2
```

4.循环

不支持while循环，只支持for循环

```
for key in dic.keys():
print(dic[key])
```

(三) 序列的切片

1. 序列的定义

内容连续、有序、可以使用下标索引的数据容器为序列。列表、字符串、元组都是序列

2. 切片

1.语法

序列[起始位置:终止位置:步长]

在序列中取出从起始位置开始，增加步长的数量的下标索引，到终止位置(不包括终止位置)的数据

其中起始位置可以省略，省略时默认从头开始；终止位置可以省略，省略时，默认为到结尾(包括结尾)；步长可以省略，省略时，默认为1.

```
l = [1,2,3]
print(l[::])#结果为[1,2,3]
print(l[1:2:1])#结果为[2]
```

当步长为负数时，表示从后向前取，以起始位置开始，到终止位置结束(不包含终止位置)，下标索引增加步长的数量。此时起始位置要比终止位置大。

```
l = [1,2,3,4,5]
print(l[::-1])#[5, 4, 3, 2, 1]
print(l[4:1:-1])#[5, 4, 3]
```

序列不会影响序列本身，是返回一个新的序列

(四) 总结

1.数据容器通用函数

1，sorted(数据容器,reverse = False(默认为False))

对有序序列进行排序，默认升序.

sorted 方法返回的是一个新的 list，而不是在原来的基础上进行的操作。

2.len(数据容器)：统计容器的长度

3.max(数据容器)/min(数据容器) 取最大/最小值

4.set(数据容器) 将数据容器转换为集合来去重

5.tuple(数据容器) 将数据容器转换为元组

6.list(数据容器) 将数据容器转换为列表类型

7.str(数据容器) 将数据容器转换为字符串类型

2.字符串比较

从头依次比较两个字符串的ACII码值，直到两个ACII码值等于/大于/小于为止,返回

```
print('abd'>'Aaceda')#True
```