数据容器

1. 定义：

可以容纳多份数据的数据类型，每一份数据称为一个元素。

2. 类型：

(1) 列表(list)

(2) 元组（tuple）

(3) 字符串(str)

(4) 集合(set)

(5) 字典(dict)

数据容器根据特点不同分为：

1.是否可以修改

2.是否有序

3.是否可以有重复的元素

(一) 列表(list):

1. 定义：

一个可以有序，可重复，可修改,容纳多个数据的数据容器,其中的数据可以是任意的数据类型。列表的元素在一个[]中。

list = [元素1,元素2,元素3,...]

2. 下标索引：

同数组一样：

从前向后的元素的索引为 0,1,2,3,....

从后向前的元素的索引为 -1,-2,-3,...

其中的数据在调用时则使用 列表名[该数据的下标索引]

eg:

```
list = [1,2,3,4,5]
print(list[0]) #结果为1
print(list[-1]) #结果为5
```

3. 常用方法:

方法：在类中的封装的函数称之为方法

1.增加

append():将数据添加到列表的末尾

```
#list.append(数据)
l = [1,2,3]
l.append(4)
print(l) # 结果为[1,2,3,4]
```

extend():将列表中的第一层数据追加到到列表的末尾

```
#list.extend(列表)
l = [1,2,3]
l.extend([4,5])
print(l) #结果为[1,2,3,4,5]


l = [1,2,3]
l.extend([4,5,[6,7]])
print(l) #结果为[1,2,3,4,5,[6,7]]
```

insert(下标,数据)：将数据插入到该下标的数据前

```
#list.insert(下标,数据)
l = [1,2,3]
l.insert(0,0)
print(l) #结果为[0,1,2,3]
```

2.删除

del list[下标索引]

```
l = [1,2,3,4]
del l[0]
print(l) #结果为[2,3,4]
```

pop(下标)：将该下标索引的值从列表中删除,如果不指定下标则默认为最后一个数字

```
#list.pop(下标)
l = [1,2,3]
l.pop(0)
print(l) #结果为[2,3]
```

pop()不指定时：

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

remove(数据):将列表中与该数据的第一个匹配值删除

```
#list.remove(数据)
l = [1,2,3,4,3,5]
l.remove(3)
print(l) #结果为[1,2,4,3,5]
```

3.修改
直接改：
list[下标索引] = 值

```
l = [1,2,3,4,5]
l[0] = 0
print(l) #结果为[0,2,3,4,5]
```

4.查找
index(数据)：查找列表中与该数据第一个匹配的数据，并返回其下标索引

```
l = [1,2,3,4,3,5]
print(l.index(3)) #结果为2
```

5.统计
count(数据)：统计改数据在列表中的个数

```
#list.count(数据)
l = [1,2,3,4,3,5]
print(l.count(3)) #结果为2
```

len(列表)：返回列表的长度

```
#len(list)
l = [1,2,3,4,5]
print(len(l)) #结果为5
```

6.清空

clear()：清空列表

```
l = [1,2,3,4,5]
l.clear()
print(l) #结果为 []
```

(二) 元组(tuple):

1. 定义：

一个可以有序，可重复，不可修改,容纳多个数据的数据容器,其中的数据可以是任意的数据类型。列表的元素在一个()中。

tuple = (元素1,元素2,元素3,...)

当定义元组时，不可以只定义一个(1)来作为一个包含1的元组，应定义为(1,)

```
t1 = (1)
print(t1) #结果为1
print(type(t1)) #<class:"int">
t2 = (1,)
print(t2) #(1,)
print(type(t2)) #<class:"tuple">
```

2. 下标索引：

同列表一样：

从前向后的元素的索引为 0,1,2,3,....

从后向前的元素的索引为 -1,-2,-3,...

其中的数据在调用时则使用 列表名[该数据的下标索引]

eg:

```
tuple = (1,2,3,4,5)
print(tuple[0]) #结果为1
print(tuple[-1]) #结果为5
```

3. 常用方法:

1.查找

index(数据)：查找元组中与该数据第一个匹配的数据，并返回其下标索引

```
t = (1,2,3,4,3,5)
print(t.index(3)) #结果为2
```

2.统计
count(数据)：统计改数据在元组中的个数

```
#tuple.count(数据)
t = (1,2,3,4,3,5)
print(t.count(3)) #结果为2
```

len(元组)：返回元组的长度

```
#len(tuple)
t = [1,2,3,4,5]
print(len(t)) #结果为5
```

(三) 字符串(str)：

1. 定义：
一个可以有序，可重复，不可修改,容纳多个数据的数据容器,其中的数据***只能是字符串***。列表的元素在一个""中。
str = "str"

2. 下标索引：
同列表一样：
从前向后的元素的索引为 0,1,2,3,....
从后向前的元素的索引为 -1,-2,-3,...
其中的数据在调用时则使用 列表名[该数据的下标索引]
eg:

```
str = "abc"
print(tuple[0]) #结果为a
print(tuple[-1]) #结果为c
```

3. 常用方法:
1.查找
index(字符串)：查找中与该字符串第一个匹配的字符串，并返回其下标索引

```
str = "abca"
print(str.index(a)) #结果为0
```

当需要匹配的是字符串时,这匹配第一个字母的下标

```
str = "abcdabcd"
print(str.index(abc)) #结果为0
```

2.统计

count(字符串)：统计改字符串在字符串中的个数

```
#str.count(数据)
str = "abca"
print(str.count("a")) #结果为2
```

当遇到有重叠时，字符串匹配则会将前一次匹配上的字符串作为一个整体

```
str = "abcabca"
print(str.count("abca")) #结果为1，因为第一次匹配到的abca作为一个整体字符串,第二次时，程序在判
```

len(字符串)：返回字符串的长度

```
#len(str)
str = "abcde"
print(len(t)) #结果为5
```

3.规整

str.strip(字符串):删除字符串前后的字符串中的每个字符

被删除的字符为方法中传入的字符串中的每一个字符,从前向后匹配，如果遇到匹配的字符，则删除匹配的字符，直到遇到没有匹配字符串中任意一个字符的字符。之后再从后向前匹配，删除匹配的字符，遇到没有匹配的则停止。

```
str = "abca"
str = str.strip("a")
print(str.strip("a"))#bc
print(str)#bc
```

4.分隔

str.split(字符)：将字符串中的字符按照函数中的字符进行分隔，并返回一个被分割后的字符的列表，其中分隔符的位置为列表中的逗号

```
str = "a|b|c"
list = str.split("|")
print(str) #结果为['a','b','c']
```

5.替换：

str.replace(字符串1，字符串2)：将字符串中的字符串中的字符串1替换为字符串2，并返回替换后的字符串。但是不改变原本的字符串，因为字符串不可被改变。

```
str1 = "abca"
str2 = str1.replace("a","b")
print(str2)#结果为"bbca"
print(str1)#结果为"abca" ，因为字符串不可改变，所以还是原本的字符串
```