Python数据分析

（3）

殷传涛 教授

chuantao.yin@buaa.edu.cn

字符

- Python中默认编码为Unicode
- 求给定字符的编码: ord()
  `ord('a')`可得97
- 求给定编码的字符:chr()
  `chr(97)`可得’ a’
- 字符串:在Python中表示单个或多个字符

```
>>> ord("a")
97
>>> ord("A")
65
>>> ord("中")
20013
```

```
>>> chr(20013)
'中'
>>> chr(19990)
'世'
>>> chr(65)
'A'
```

```
>>> for char in "Hello Word":
...     print(ord(char))
...
72
101
108
108
111
32
87
111
114
100
```

字符串

- 由双（单）引号来创建
  - 例：’hello, world’ “good！@#” “汉字串”
  - 注：由三引号来创建的是跨行字符串
- 字符串中有引号的情况
  - 含有单引号，由双引号括住
  - 含有双引号，由单引号括住
  - 两者有有，使用转义字符 \’, \”

```
>>> print("I'am good")
I'am good
>>> print('"OK"')
"OK"
>>> print("\"OK\", I\'am good")
"OK", I'am good
>>>
```

```
>>> str='good'
>>> str1="morning"
>>> del str
>>> str1='good'
>>> str2="morining"
>>> str3="""i am very
hungry,
today"""
>>> print(str1)
good
>>> print(str2)
morining
>>> print(str3)
i am very
hungry,
today
>>>
```

字符串的索引

- 字符串本身是个字符序列，可以通过位置索引来访问其中的字符。**字符串名[索引]**
  - 对于长度为n的字符串
    - 索引为正数，则自左向右：0,1,2,……n-1
    - 索引为负数，则自左向右：-n,-(n-1),……-2,-1

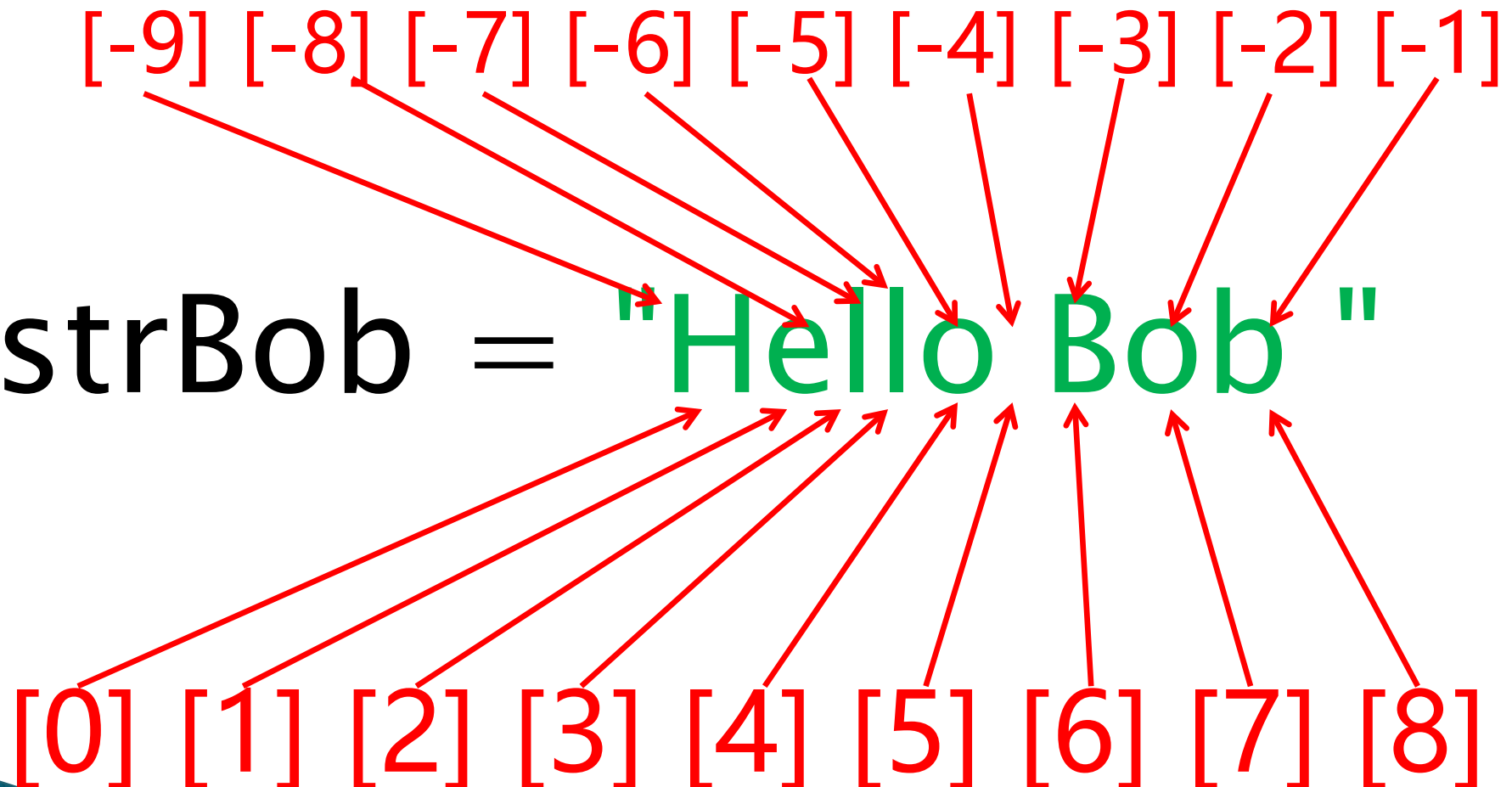
```
>>> str1="ABCDEFG"
>>> str1[0];str1[6];str1[-1];str1[-7]
'A'
'G'
'G'
'A'
>>> str1[1:5]
'BCDE'
>>> str1[:5]
'ABCDE'
```

```
>>> str1[1:]
'BCDEFG'

>>> str1[-6:]
'BCDEFG'
>>> str1[-6:-2]
'BCDE'
>>>
```

注意：str1[n：m]截取字符串的第n到m-1位

字符串操作：取字符



strBob[9]或strBob[–10]越界出错

字符串操作：取子串

- 切段：取一个索引范围内的字符.

  <字符串>[<start>:<end>]

  - 所取子串：位置索引从start ~ end–1
  - start或/和end可省略，缺省值为串的首/尾

```
>>> strBob="Hello Bob"
>>> strBob[0:3]
'Hel'
>>> strBob[5:9]
' Bob'
>>> strBob[:5]
'Hello'
>>> strBob[0:5]
'Hello'
>>> strBob[5:]
' Bob'
>>> strBob[5:9]
' Bob'
>>> strBob[:]
'Hello Bob'
>>> strBob[0:9]
'Hello Bob'
```

```
>>> strBob[9]
Traceback (most recent call last):
  File "<pyshell#24>", line 1, in <module>
    strBob[9]
IndexError: string index out of range
>>> strBob[-10]
Traceback (most recent call last):
  File "<pyshell#25>", line 1, in <module>
    strBob[-10]
IndexError: string index out of range
>>>
```

字符串的运算

- 字符串的运算符
  - +，x+y 将两个字符串连接
  - *，x*y字符串重复y次，y为int类型
  - [:]，截取字符串
  - in，x in y，判断y包含x
  - not in，判断y不包含x
  - r或R，不区分转义字符，

```
>>> "Hello"+"World"
'HelloWorld'
>>> "Hello"*3
'HelloHelloHello'
>>> "Hello"[2:4]
'll'
```

```
>>> "H" in "Hello"
True
>>> "H" not in "Hello"
False
>>> print("Hello\n")
Hello

>>> print(r"Hello\n")
Hello\n
>>>
```

字符串的长度

- len() 函数

```
>>> str1="good morning"
>>> len(str1)
12
>>>
```

```
>>> str1[0:len(str1)]
'good morning'
>>> for i in range(len(str1)):
        print(str1[i], end="")

good morning
```

字符号的索引号

- 查询某字符的索引号
  - str.index()方法
  - str.find()方法

```
>>> str1="good"
>>> str1.index("d")
3
>>> str1.index("o")
1
>>> str1.index("x")
Traceback (most recent call last):
  File "<pyshell#48>", line 1, in <module>
    str1.index("x")
ValueError: substring not found
```

```
>>> str1.find("d")
3
>>> str1.find("o")
1
>>> str1.find("x")
-1
>>>
```

字符串的转化

- 字符串与数值和表达式的区别
- 通过函数str()和eval()实现表达式与字符串的互相转化
  - str()：将其它数据类型转化为字符串
  - eval()，将字符串转化为表达式/可执行的代码

```
>>> a=123
>>> b="123"
>>> a+10
133
>>> b+10
Traceback (most recent call last):
  File "<pyshell#29>", line 1, in <module>
    b+10
TypeError: must be str, not int
```

```
>>> eval(b)+10
133
>>> str(a)+b
'123123'
```

```
>>> a=5
>>> eval("a>4")
True
>>> str("a+3")
'a+3'
>>>
```

字符串的处理

- 函数（Function）
  - 独立的代码块，不依赖于类/对象
  - 实现某种运算过程，如：y=str(x)
  - 以参数接收输入
  - 以返回值作为输出
- 方法(Method)
  - 方法是对某类数据对象的操作，
  - 通过对象进行使用，实现其操作，如str1.upper()
  - 由数据对象.方法名称（参数）格式进行使用
  - 某类数据对象，通常拥有有一系列的方法
  - 以参数接收输入
  - 以返回值作为输出

字符串的处理

- 字符串中的空格删除方法
  - rstrip()：删除末尾空格
  - lstrip()：删除开头空格
  - strip()：删除开头和末尾的空格

```
>>> a="   python     "
>>> a.rstrip()
'   python'
>>> a.lstrip()
'python     '
>>> a.strip()
'python'
>>>
```

字符串的遍历

- 使用for语句

```
>>> str1="good"
>>> len(str1)
4
>>> for c in str1:
        print(c)

g
o
o
d
```

```
>>> for i in range(len(str1)):
        print(str1[i])

g
o
o
d
```

字符串的处理

- 更多python内建的字符串处理方法

```
>>> str1="Good Morning"
>>> str1.center(30)
'          Good Morning          '
>>> str1.ljust(30)
'Good Morning                  '
>>> str1.rjust(30)
'                  Good Morning'
>>> str1.lower()
'good morning'
>>> str1.upper()
'GOOD MORNING'
>>> str1.capitalize()
'Good morning'
>>> str1.swapcase()
'gOOD mORNING'
```

```
>>> str1="Good morning"
>>> str1.title()
'Good Morning'
```

```
>>> str1.isalnum()
False
>>> str1.isdigit()
False
>>> str1.islower()
False
>>> str1.istitle()
True
>>> str1.find("M")
5
>>> str1.find("z")
-1
>>> str1.count("o")
3
>>>
```

字符串的处理

- 更多处理方法

```
>>> dir(str)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__'
, '__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne_
_', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '
capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isdecimal
', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'r
eplace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```

- 利用help()函数，查看字符串各方法的用法

```
>>> help(str.count)
Help on method_descriptor:

count(...)
    S.count(sub[, start[, end]]) -> int

    Return the number of non-overlapping occurrences of substring sub in
    string S[start:end].  Optional arguments start and end are
    interpreted as in slice notation.

>>> 'good morining'.count('o')
3
>>> a="good morning"
>>> a.count('o')
3
>>> a[0:5].count('o')
2
```

序列（Sequence）型数据

- 有序的序列：字符串、列表、元组
  - 有序序列的通用操作
    - 索引、分片、加、乘
    - 检查是否属于序列
    - 长度、最大、最小…
- 无序的序列
  - 集合、字典

列表

- 列表（list）是Python中最具有灵活性的有序集合对象类型
  - 列表创建：方括号，逗号
  - 可以包含任何种类的对象元素：数字、字符串、自定义对象、列表本身……
  - 有序：从左到右的位置顺序
  - 长度可变，元素可以修改
  - 列表可以嵌套：列表作为另一个列表中的元素

```
a_list = ['a', 'b', 'mpilgrim', 'z', 'example']
```

```
a_list = ['a', 'b', 'mpilgrim', 'z', 'example',[1,2,3]]
```

列表

- 列表的含义

```
a_list = ['a', 'b', 'mpilgrim', 'z', 'example']
```

- 说明：
  - 建立a_list列表，则计算机建立了包含5个元素的可以按照序号访问的数据对象，因此，列表是一个有序的元素序列
  - 既然是有序序列，那么，各个元素就有序号
  - 第一个元素序号为0，从左到右：0，1，2……
  - 也可以负值索引，最后一个元素为-1，从右到左：-1，-2，-3……

列表的创建

- 利用方括号[ ]创建
  - 可以创建任意多的元素
  - 中间用逗号“，”隔开

```
>>> list1 = ["good" , "student" , 3 , 4 , 5 , "!" ]
>>> list2 = []
```

- 利用list( )函数
  - 将其它序列类型转化为列表

```
>>> list1 = list("good")
>>> list1
['g', 'o', 'o', 'd']
>>> list2 = list(range(3,9,2))
>>> list2
[3, 5, 7]
>>> list3=list()
>>> list3
[]
```

列表的相关操作

- 访问列表中的元素
- 增加元素
- 删除元素
- 修改元素
- 查找元素
- 和其他数据类型的转换
- 其他内置的功能函数

列表元素的访问

- 列表的索引
  - n个元素的列表，从左到右，索引分别为0,1,2,3,n-1
  - 可被负值索引，从左到右，索引分为为-n,-(n-1),-2,-1
- 访问列表中的某个元素
  - 列表名[索引]
  - 若超出规定的索引号，则程序出错

```
>>> list1 = ["good" , "student" , 3 , 4 , 5 , "!" ]
>>> list1
['good', 'student', 3, 4, 5, '!']
>>> list1[1]
'student'
>>> list1[5]
'!'
>>> list1[-1]
'!'
>>> list1[-5]
'student'
```

```
>>> list1[20]
Traceback (most recent call last):
  File "<pyshell#44>", line 1, in <module>
    list1[20]
IndexError: list index out of range
```

列表元素的访问

- 切片访问列表中连续的多个元素
  - 列表名[i:j]
    - 若切片索引从0开始，则可以省去开始的索引号i。列表名[:j]
    - 若切片索引到列表最后，则可以省去结束的索引号j。列表名[i:]
  - 返回一个新列表

```
>>> list1 = ["good" , "student" , 3 , 4 , 5 , "!" ]
>>> list1[1:3]
['student', 3]
>>> list1[-3:-1]
[4, 5]
>>> list1[:3]
['good', 'student', 3]
>>> list1[4:]
[5, '!']
```

```
>>> list2=list1[1:4]
>>> print(list1)
['good', 'student', 3, 4, 5, '!']
>>> print(list2)
['student', 3, 4]
```

列表元素的查询

- 判断某元素是否在列表中
  - 使用in和not in运算符
  - 使用count()方法

```
>>> list1 = ["good" , "student" ,  3 , 4 , 5 , "!" ]
>>> 3 in list1
True
>>> if "good" in list1:
            print("good is in list1")
else:
            print("good is not in list1")

good is in list1
>>> "bad" not in list1
True
```

```
>>> list2 = [ 1 , 2 , 3 , 1 , 3, 4]
>>> list2.count(3)
2
```

```
>>> list1.count(3)
1
>>> if list1.count("good")>0:
            print("good is in list1")
else:
            print("good is not in list1")

good is in list1
```

向列表中添加元素

- list.append(x)
  - 向list列表最末端添加1个元素
  - 相当于list[len(list):]=[x]
- list.extend(seq)
  - 将1个序列seq追加在list列表的后面
  - 相当于list[len(list):]=seq
- list.insert(i,x)
  - 向list列表的i位置添加对象x
  - 插入位置后的元素，索引相应后移
- 使用+运算符
  - 合并两个列表成新列表

向列表中添加元素

- list.append(x)
- 将1个对象元素添加到列表末尾
- 注意
  - 仅能一次添加一个
  - 若添加一个列表，则列表作为一个元素进行添加

```
>>> a = [ 1 , 2 ]
>>> a.append(3)
>>> a
[1, 2, 3]
>>> a.append(4,5)
Traceback (most recent call last):
  File "<pyshell#79>", line 1, in <module>
    a.append(4,5)
TypeError: append() takes exactly one argument (2 given)
```

```
>>> b = [ 4 , 5 ]
>>> a.append(b)
>>> a
[1, 2, 3, [4, 5]]
>>> a.append("good")
>>> a
[1, 2, 3, [4, 5], 'good']
```

向列表中添加元素

- list.extend(seq)
- extend()方法，将1个序列追加到列表末尾,形成一个新列表

```
>>> a = [ 1 , 2 ]
>>> b = [ 4 , 5 ]
>>> a.extend(b)
>>> a
[1, 2, 4, 5]
>>> a.extend(3)
Traceback (most recent call last):
  File "<pyshell#90>", line 1, in <module>
    a.extend(3)
TypeError: 'int' object is not iterable
```

```
>>> c = "good"
>>> a.extend(c)
>>> a
[1, 2, 4, 5, 'g', 'o', 'o', 'd']
>>> a.extend(range(3))
>>> a
[1, 2, 4, 5, 'g', 'o', 'o', 'd', 0, 1, 2]
```

向列表中添加元素

- list.insert(i,x)
- 将单个元素x插入到列表中i的位置。

```
>>> a = [ 1 , 2 , 4 ]
>>> a.insert(2,3)
>>> a
[1, 2, 3, 4]
```

```
>>> b = [ "boy" , "girl" ]
>>> b.insert(1,"and")
>>> b
['boy', 'and', 'girl']
```

```
>>> b.insert(5, "is")
>>> b
['boy', 'and', 'girl', 'is']
>>> b.insert(0,"a")
>>> b
['a', 'boy', 'and', 'girl', 'is']
>>> b.insert(-7, "a")
>>> b
['a', 'a', 'boy', 'and', 'girl', 'is']
```

向列表中添加元素

- +连接运算符：将别的列表元素增加到本列表尾部
- 注：不推荐使用(消耗较多内存)

```
>>> a = "good"
>>> b = "morning"
>>> c = a+b
>>> a
'good'
>>> b
'morning'
>>> c
'goodmorning'
```

```
>>> a = [ 1 , 2 ]
>>> b = [ 4 , 5 ]
>>> c = a + b
>>> a
[1, 2]
>>> b
[4, 5]
>>> c
[1, 2, 4, 5]
```

从列表中删除元素

- del list[i], del list[i:j]
  - 删除list中索引为i的元素
  - 删除list中索引i到j-1的元素
- list.remove(x)
  - 删除list中值为x的第一个元素
- list.pop(i)
  - 返回list中索引i为的元素，并将其删除
  - 如果i空缺，则返回并删除最后一个元素

从列表中删除元素

- del list[i], del list[i:j]
  - 删除list中索引为i的元素
  - 删除list中索引i到j-1的元素

```
>>> a = [ 1 , 2 , 3 , 4 ]

>>> del a[1]
>>> a
[1, 3, 4]
>>> del a[1:]
>>> a
[1]
>>> del a[0]
>>> a
[]
```

```
>>> a = [ 1 , 2 , 3 , 4 ]
>>> del a
>>> a
Traceback (most recent call last):
  File "<pyshell#140>", line 1, in <module>
    a
NameError: name 'a' is not defined
```

从列表中删除元素

- list.remove(x)
  - 删除list中值为x的第一个元素

```
>>> a = [ 1 , 2 , 3 , 4 , 2 , 2 , 3 ]
>>> a.remove(3)
>>> a
[1, 2, 4, 2, 2, 3]
```

  - 如何删除一个列表中所有值为x的元素？

In [3]:
```
a =[1,2,3,4,2,2,3]
for i in a:
    if i==2:
        a.remove(i)
print(a)
```

```
[1, 3, 4, 2, 3]
```

In [4]:
```
a =[1,2,3,4,2,2,3]
for i in range(a.count(2)):
    a.remove(2)
print(a)
```

```
[1, 3, 4, 3]
```

从列表中删除元素

- list.pop(i)
  - 返回list中索引i为的元素，并将其删除
  - 如果i空缺，则返回并删除最后一个元素

```
>>> a = [ 1 , 2 , 3 , 4 , 2 , 2 , 3 ]
>>> a.pop()
3
>>> a
[1, 2, 3, 4, 2, 2]
>>> a.pop(0)
1
>>> a
[2, 3, 4, 2, 2]
```

```
>>> a.pop(2)
4
>>> a
[2, 3, 2, 2]
>>> a.pop(6)
Traceback (most recent call last):
  File "<pyshell#167>", line 1, in <module>
    a.pop(6)
IndexError: pop index out of range
```

列表元素的访问

- 修改列表中的元素
  - 对索引的元素进行赋值

```
>>> a = [ 1 , 2 , 3 , 4 , 2 , 2 , 3 ]
>>> a [0] =0
>>> a
[0, 2, 3, 4, 2, 2, 3]
>>> b = ["good", "morning"]
>>> b [1] = "afternoon"
>>> b
['good', 'afternoon']
```

```
>>> a [1:2]=[0,0]
>>> a
[0, 0, 0, 3, 4, 2, 2, 3]
>>> a [4:]=[0, 0, 0, 0]
>>> a
[0, 0, 0, 3, 0, 0, 0, 0]
```

列表的其它操作

- 对于列表可以操作的内建函数
  - len(list),max(list),min(list),sum(list)

```
>>> a = [ 1 , 2 , 3 , 4 , 2 , 7 , 3 ]
>>> len(a)
7
>>> max(a)
7
>>> min(a)
1
```

```
>>> a = [ 1 , 2 , 3 , 4 , 2 , 7 , 3 ]
>>> sum(a)
22
```

```
>>> b = ["good", "morning"]
>>> len(b)
2
>>> max(b)
'morning'
>>> min(b)
'good'
```

```
>>> c = ["a", "b", 3]
>>> max(c)
Traceback (most recent call last):
  File "<pyshell#187>", line 1, in <module>
    max(c)
TypeError: '>' not supported between instances of 'int' and 'str'
```

```
>>> b = ["good", "morning"]
>>> sum(b)
Traceback (most recent call last):
  File "<pyshell#195>", line 1, in <module>
    sum(b)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

列表的其它操作

- 列表操作的其它方法
  - list.count(x): 返回列表中出现x对象的次数
  - list.index(x)：返回第一次出现x对象的索引
  - list.reverse( )：倒排列表中的元素
  - list.sort():按大小排序列表中的元素

```
>>> a = [ 1 , 2 , 3 , 4 , 2 , 7 , 3 ]
>>> a.count(2)
2
>>> a.index(7)
5
>>> a.reverse()
>>> a
[3, 7, 2, 4, 3, 2, 1]
```

```
>>> a.sort()
>>> a
[1, 2, 2, 3, 3, 4, 7]
>>> a.sort(reverse=1)
>>> a
[7, 4, 3, 3, 2, 2, 1]
```

列表的其它操作

- 判断元素是否在列表中
  - in, not in
- 列表中元素的遍历
  - for variable in list
- 列表嵌套
  - list[i][j]

```
>>> a=["hello", 1, 3, "kityy"]
>>> "hello" in a
True
>>> "kityy" not in a
False
>>>
```

```
>>> a=["hello", 1, 3, "kityy"]
>>> for x in a:
        print(x)


hello
1
3
kityy
>>>
```

```
>>> a=[[0, 1, 2], [3, 4]]
>>> a[0]
[0, 1, 2]
>>> a[0][1]
1
>>>
```

列表与字符串

- 字符串和形表都是序列型数据对象
  - 可通过索引访问其中的元素
  - 可以通过切片访问其中连续的元素
- 列表与字符串的区别：
  - 字符串是一个特殊的列表；
  - 列表的元素可以是任何数据类型，而字符串中只能是字符；
  - 列表的元素可修改，而字符串的元素不能修改。

列表与字符串

- 字符串中的元素不能修改

```
>>> list1 = [10, 12, 3, 25]
>>> list1[2]
3
>>> list1[3]=23
>>> print(list1)
[10, 12, 3, 23]
```

```
>>> str1="Hello World"
>>> str1[2]
'l'
>>> str1[2]="x"
Traceback (most recent call last):
  File "<pyshell#37>", line 1, in <module>
    str1[2]="x"
TypeError: 'str' object does not support item assignment
```

字符串和列表的相互转换

- 字符串转为列表
  - 使用list()函数，直接把字符串转化为列表

```
>>> str1="good morning"
>>> list1=list(str1)
>>> print(list1)
['g', 'o', 'o', 'd', ' ', 'm', 'o', 'r', 'n', 'i', 'n', 'g']
>>>
```

  - 使用字符串的方法split()，把字符串转化为列表

```
>>> list1=str1.split()
>>> print(list1)
['good', 'morning']
>>>
```

```
>>> list2=str1.split('m')
>>> print(list2)
['good ', 'orning']
```

```
>>> str2="this-is-a-good-story"
>>> print(str2.split("-"))
['this', 'is', 'a', 'good', 'story']
>>>
```

字符串与列表的相互转换

- 列表转化为字符串
  - 使用str()函数直接转化列表为字符串，注意方括号

```
>>> list1=['a','b','c']
>>> str1=str(list1)
>>> print(str1)
['a', 'b', 'c']
```

  - 使用字符串的方法join()，将列表转化为字符串

```
>>> str1="--"
>>> str1.join(list1)
'a--b--c'
>>> str2="Good"
>>> str2.join(list1)
'aGoodbGoodc'
>>> "".join(list1)
'abc'
```

更多的列表操作

- 使用dir()与help()函数，查看更多方法使用

```
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__',
'__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '
__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__', '__i
ter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce_
_', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__se
titem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear', 'copy', 'c
ount', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

```
>>> help(list.sort)
Help on method_descriptor:

sort(...)
    L.sort(key=None, reverse=False) -> None -- stable sort *IN PLACE*

>>> list1=[3, 5, 1, 0, 6]
>>> list1.sort(reverse=True)
>>> list1
[6, 5, 3, 1, 0]
```

元组

- 元组（tuple）
  - 圆括号，逗号
  - 与列表的使用方法类似
  - 元组的内容创建后，其元素只允许访问，不允许修改
  - 字符串是一种特殊的元组

```
#!/usr/bin/python3

tuple = ( 'abcd', 786 , 2.23, 'runoob', 70.2  )
tinytuple = (123, 'runoob')

print (tuple)             # 输出完整元组
print (tuple[0])          # 输出元组的第一个元素
print (tuple[1:3])        # 输出从第二个元素开始到第三个元素
print (tuple[2:])         # 输出从第三个元素开始的所有元素
print (tinytuple * 2)     # 输出两次元组
print (tuple + tinytuple) # 连接元组
```

集合

- 集合（set）是无序不重复元素的序列
  - 大括号，逗号，或者set( )函数来创建或转换
  - 空集合用set( )来创建
  - 主要功能为判断是否为集合的成员和删除重复元素
  - 集合运算：并集、交集、差集等

```
#!/usr/bin/python3

student = {'Tom', 'Jim', 'Mary', 'Tom', 'Jack', 'Rose'}

print(student)   # 输出集合，重复的元素被自动去掉

# 成员测试
if('Rose' in student) :
    print('Rose 在集合中')
else :
    print('Rose 不在集合中')
```

```
{'Mary', 'Jim', 'Rose', 'Jack', 'Tom'}
Rose 在集合中
```

```
a = set('abracadabra')
b = set('alacazam')

print(a)

print(a - b)     # a和b的差集

print(a | b)     # a和b的并集

print(a & b)     # a和b的交集

print(a ^ b)     # a和b中不同时存在的元素
```

```
{'b', 'a', 'c', 'r', 'd'}
{'b', 'd', 'r'}
{'l', 'r', 'a', 'c', 'z', 'm', 'b', 'd'}
{'a', 'c'}
{'l', 'r', 'z', 'm', 'b', 'd'}
```

集合

- 集合的元素添加
  - set.add(x)，将元素x加入到集合set中
  - set.update(set1)，将set1中的元素加入到集合set中

```
>>> a=set("boy")
>>> a.add("python")
>>> a
{'python', 'o', 'b', 'y'}
>>>
```

```
>>> a=set("boy")
>>> a.update("python")
>>> a
{'h', 't', 'p', 'b', 'y', 'n', 'o'}
>>>
```

- 集合的元素删除
  - set.remove(x)，将集合set中的x元素删除

```
>>> a={"o","b","y","python"}
>>> a
{'python', 'o', 'b', 'y'}
>>> a.remove("python")
>>> a
{'o', 'b', 'y'}
>>>
```

- 用{ }创建空字典，而不是空集合

字典

- 字典（dictionary）是成对的数据项的集合
  - 数据项的形式：键(key):值(value)
  - 大括号，逗号，冒号，或者用dict( )创建
  - 键值不可改变，并且在字典中唯一
    - 通常使用字符串或者数字作为键值
  - 用{ }创建的是空字典，而不是空集合

```
>>> a={}
>>> type(a)
<class 'dict'>
>>>
```

```
>>> student={"001":"jenny","002":"tom","005":"mary"}
>>> student["002"]
'tom'
```

字典的变量名　键　值　数据项

字典

- 通过键值（而不是索引）来访问数据项的值

```
>>> student={"001":"jenny","002":"tom","005":"mary"}
>>> student["002"]
'tom'
>>> student["005"]
'mary'
```

- 也可使用get( )来访问数据项的值

```
>>> student.get("005")
'mary'
```

- 用keys( )和values() 分别返回字典中所有的键和所有的值

```
>>> student.keys()
dict_keys(['001', '002', '005'])
>>> student.values()
dict_values(['jenny', 'tom', 'mary'])
```

字典

- 向字典中添加数据项

```
>>> student={"001":"jenny","002":"tom","005":"mary"}
>>> student["004"]="andy"
>>> student
{'001': 'jenny', '002': 'tom', '005': 'mary', '004': 'andy'}
```

- 改变某一个数据项的值

```
>>> student["004"]="mandy"
>>> student
{'001': 'jenny', '002': 'tom', '005': 'mary', '004': 'mandy'}
```

- 将一个字典合并到另外一个字典

```
>>> student1={"1001":"liu","1002":"zhang"}
>>> student.update(student1)
>>> student
{'001': 'jenny', '002': 'tom', '005': 'mary', '004': 'mandy', '
1001': 'liu', '1002': 'zhang'}
```

字典

- 删除某个数据项的值：del或者dict.pop( )

```
>>> del student["002"]
>>> student
{'001': 'jenny', '005': 'mary', '004': 'mandy', '1001': 'liu',
'1002': 'zhang'}
>>> student.pop("001")
'jenny'
>>> student
{'005': 'mary', '004': 'mandy', '1001': 'liu', '1002': 'zhang'}
```

字典的使用

python

```python
movie_ratings = {
    'The Godfather': 9.2,
    'The Shawshank Redemption': 9.3,
    'The Dark Knight': 9.0,
    # ... 其他电影评分
}

# 打印每部电影及其评分
for movie, rating in movie_ratings.items():
    print(f"{movie}: {rating}")
```

谢谢！