序列的创建：

字符串：' " '''

元组：小括号 () 或者直接用 " , "隔开各个元素创建

列表：中括号 [ ]

类型转换：str(),tuple(),list(),set(),dict() 字符串/元组/列表/集合/字典

可变：列表、集合、字典 不可变：字符串、元组

>>> ls = [1,2,3,4,5]

>>> ls.append(6)

>>> #返回的是None代表是对本身的操作

>>> ls

[1, 2, 3, 4, 5, 6]

>>> ss = 'hello world'

>>> ss.capitalize()

'Hello world' #使用方法后直接返回答案

>>> ss

'hello world' #原始的ss并没有改变本身，所以srt是不可变对象

>>> ss = ss.capitalize() #若要改变需重新赋值

>>> ss

'Hello world'

序列的通用操作：

**索引**：索引从0（从左向右）开始，所有序列可通过这种方式进行索引

**切片**：用来访问一定范围内的元素，通过冒号相隔的两个索引来实现，不同的步长，有不同的输出[::]

**序列相加**：同一类型才能相加 +

**重复**：序列\*重复的次数

**检查成员**： in 、not in判断元素是否在列表里

**长度,最大值,最小值**  len ()/max ()/min()

一.字符串类型 str()

‘ 单引号

‘’ 双引号

‘’’ 三引号 多行用’’’

字符串的运算

len( ) 求字符串长度

a = 'hello python'

len(a) = 12

取值：

a[0] = 'h'

a[1] = 'e'

a[11] = 'n'

a[-1] = 'n'

切片：

a = [0,1,2,3,4,5,6,7,8,9,10]

a[:2]=[0, 1]

a[2:4] = [2, 3]

a[:] = [0,1,2,3,4,5,6,7,8,9,10] 取所有

a[::2] = [0, 2, 4, 6, 8, 10] 间隔一个取一个

a[:-5]=[0, 1, 2, 3, 4, 5]

a[-5:]=[6, 7, 8, 9, 10]

部分字符串的操作方法：

s.[capitalize()](http://www.runoob.com/python3/python3-string-capitalize.html)**:**将字符串的第一个字符转换为大写

s.count(x)：返回字符串x在s中出现的次数，带可选参数

s.endswith(x)：如果字符串s以x结尾，返回True

s.startswith(x)：如果字符串s以x开头，返回True

s.find(x) ：返回字符串中出现x的最左端字符的索引值，如果不在则返回-1

s.rfind(x)：和find()类似，但是从右往左搜索

s.index(x)：返回字符串中出现x的最左端的索引值，如果不在则抛出valueError异常

s.rindex(x)：和index()类似，但是从右往左搜索

s.isalpha ()：测试是否全是字母，都是字母则返回 True,否则返回 False.

s.isdigit () ：测试是否全是数字，都是数字则返回 True 否则返回 False.

s.islower ()：测试是否全是小写

s.isupper ()：测试是否全是大写

s.lower () ：将字符串转为小写

s.upper () ：将字符串转为大写

s.replace (‘x’,’y’[, count]) ：替换,在字符串s中出现字符串x的任意位置都用y进行替换 （相当于替换）[count]是替换个数

s.split()：返回一系列用空格分割的字符串列表

s.split(‘a’,b)：a,b为可选参数，a是将要分割的字符串，b是说明最多要分割几个

str --通过encode----变成bytes

bytes --通过decode解码 -----变成str

s.encode() 方法以指定的编码格式编码字符串。errors参数可以指定不同的错误处理方案。

s.decode() 方法以指定的编码格式解码 bytes 对象。默认编码为 'utf-8'。

>>> y = '什么'

>>> y.encode('utf-8')

b'\xe4\xbb\x80\xe4\xb9\x88'

>>> z = b'\xe4\xbb\x80\xe4\xb9\x88'

>>> z.decode()

'什么'

字符串拼接

第一种方法：

a = 'hello'

b = 'python'

c = '!'

a+b+c

'hellopython!'

第二种方法：

'my name is %s'%'taka'

'my name is taka'

'%s %s %s'%(a,b,c) 把蓝色%的值填到红色%处

'hello python !'

'my name is %s,hhhhhdf' % 'taka'

'my name is taka,hhhhhdf'

注：s前面可以加对象名，后面以字典的方式填入

a = 'hello' b = 'python' c = '!'

'%(x)s %(y)s %(z)s'%({'x':a,'y':b,'z':c})

'hello python !'

第三种方法：''.join()方式，注意括号里是要连接的可以是列表，元祖

a = 'hello' b = 'python' c = '!'

'\*'.join([a,b,c])

'hello\*python\*!'

>>> a = 'python'

>>> b = list(a)

>>> b

['p', 'y', 't', 'h', 'o', 'n']

>>> ''.join(b)

'python'

第四种方法：.format方式 (注：{}里面可以填入与后面相对应的符号)

a = 'hello' b = 'python' c = '!'

'{} {} {}'.format(a,b,c)

'hello python !'

当{}里面有索引值时，按前面的索引值将后面的每项依次填入

'{2} {0} {1}'.format(a,b,c)

'! hello python'

后面通过赋值的方式给前面的相应的值，后面是无序的

'{n1} {n2} {n3}'.format(n1=a,n3=c,n2=b)

'hello python !'

字符串的格式化：

%s 格式化字符串

%c 格式化ASCII字符

%d 格式化整数

%f 格式化小数

%(+/-)m.nf 格式化定点数，当m大于格式化位数时才起作用显示m位，n是显示的小数的位数(+/-)表示左右对齐方式

%o 格式化无符号八进制

%x 格式化无符号十六进制

%e 用科学计数法格式化定点数

字符串的转义：（注：'\'不能放在字符的末尾）

转义标识符 ' \ '，当“ ’ ”要保留是用\转义

a = 'I\'m a student'

"I'm a student"

\\ 反斜杠 \‘ 单引号 \“双引号 \n 换行 \续行符 \e 转义

\a 提示音 \b 退格键 \r 回车键 \t Tab键 \f 换页 \000 空

\t 横向制表符 \v 纵向制表符

当要正常输出特殊的转义字符时，使用 r

例：>>> print (r'abc\ndef')

abc\ndef

>>> print ('abc\ndef')

abc

def

二.元组 tuple()

创建空元组

tup1 = ();

元组中只包含一个元素时，需要在元素后面添加逗号

tup1 = (50,);

元组不能被修改，可以加法，不能减法

>>> x, y = 1, 2

>>> tup1 = (50,)

>>> tup2 = ('abc', 'xyz')

>>> tup3 = tup1 + tup2;

>>> print tup3;

(50, 'abc', 'xyz')

元组的属性和方法：

.count（obj）统计某个元素在元组中出现的次数

.index（obj）从列表中找某个值第一个匹配项的索引位置

三.列表的属性和方法 list()

.append（obj） 在列表末尾添加新的对象

.count（obj） 统计某个元素在列表中出现的次数

.extend（seq） 用新列表扩展原来的列表

.index（obj,index） 从列表中找某个值第一个匹配项的索引位置

.insert（index,obj）插入，可以指定位置

.pop(index) 出栈，可以指定位置

.remove(obj) 移除指定元素从左边开始的第一个

.reverse() 反向列表中元素

.sort() 对原列表进行排序

.clear() 清空列表

.copy() 复制列表

四.集合 set()

集合的创建： { } 、set([ ])

集合的特点：无序、元素是唯一的，不能有列和字典

>>> b = set() #用set()创建空集合

>>> type(b)

<class 'set'>

>>> b = set([3,4,5]) #有元素加[]

>>> a = {} #直接用大括号创建是字典

>>> type(a)

<class 'dict'>

集合的运算

交集 &

并集 |

差集 -

集合的属性和方法：

in、not in 判断元素在不在set 中，在则返回True

.add(x) 添加单个元素

.update([x,x,x]) 添加多个元素

.remove() 移除元素

.clear() 清空集合

五.字典创建 dict()

{ key ：value } dict(key=value)

字典里的键和值用“ ：”隔开，一对键和值组成一个项，项和项之间用‘ ，’隔开；

字典的键可以是数字、字符串或者是元组，键必须唯一，无序的；

键可以为任何不可变类型，列表和集合不能作为字典的键；

a = {1:'a',2:'b'} #键为数字

{1: 'a', 2: 'b'}

d = {'a':1,'b':2} #键为字符串

{'a': 1, 'b': 2}

a = {(1,2,3):'xxx'} #键为元组

{(1, 2, 3): 'xxx'}

dict1 = dict(a=1,b=2) #dict都是以字符串类型

{'a': 1, 'b': 2}

访问字典中的值（因为是无序，所以要用键来查询）

dict2 = {'name': 'jack', 'age': 20, 'sex': 'man'}

dict2[‘name’] = ’jack’

>>> dict1 = dict(name='taka',age=21,gender='female')

>>> dict2 = dict(name='jack',age=20,gender='male')

>>> li = [dict1,dict2]

>>> li

[{'name': 'taka', 'age': 21, 'gender': 'female'}, {'name': 'jack', 'age': 20, 'gender': 'male'}]

>>> li[0]['name']

'taka'

字典添加和修改：

新值所要对应的键名如果存在，就是修改操作，如果不存在就相当于添加操作

dict2 = {'name': 'jack', 'age': 20, 'sex': 'man'}

dict2['country'] = 'China' #添加

dict2 = {'name': 'jack', 'age': 20, 'sex': 'man', 'country': 'China'}

dict2['country'] = 'England' #修改

dict2 = {'name': 'jack', 'age': 20, 'sex': 'man', 'country': 'England'}

字典删除的方法：

del 方法：可删除字典的单个项，也可以删除整个字典

.pop(‘key’) 方法：返回键对应的值，输出后再删除

.clear() 方法：清空整个字典

>>> d = {'a':1,'b':2}

>>> del d['b']

>>> d

{'a': 1}

字典的属性和方法：

in 、not in 判断键在不在字典中，在则返回True

.[copy()](http://www.runoob.com/python3/python3-att-dictionary-copy.html) 返回一个字典的浅复制

.update({ }) 在字典中添加多个项

.items() 以列表返回可遍历的(键, 值) 元组数组

.keys() 返回字典的键

.values() 返回字典的值

.pop(k) 返回并移除键k所对应的元素，不存在则抛出异常

.pop(k,x) 返回并移除键k所对应的元素，不存在则返回x

.get(k) 如果键k在，返回键k的值，不存在则返回None

.get(k,x) 如果键k在，返回键k的值，不存在则返回x

[.setdefault(key, default=None)](http://www.runoob.com/python3/python3-att-dictionary-setdefault.html) 和get()类似, 但如果键不存在于字典中，将会添加键并将值设为default

[.fromkeys()](http://www.runoob.com/python3/python3-att-dictionary-fromkeys.html) 创建一个新字典，以序列seq中元素做字典的键，val为字典所有键对应的初始值

>>> seq = ('name', 'age', 'sex')

>>> dict1 = dict.fromkeys(seq)

>>> dict1

{'name': None, 'age': None, 'sex': None}

>>> dict1 = dict.fromkeys(seq,10)

>>> dict1

{'name': 10, 'age': 10, 'sex': 10}