python学习笔记

print函数

输出目的地为显示器		
输出数字	print(520) print(98.1)	数字直接输出
输出字符串	print('hello') print("hello") print("'hello'")	括号内加上单引号、双引号或三引号（用于告诉机器这个东西不用理解）、不可以不加引号
输出表达式	print(3+1) 输出4	3和1是操作数，+是运算符，表达式=操作数和运算符

ps:写注释 #

将数据输出到文件中

```
fp=open('D:/text.txt', 'a+')    #以读写的方式打开文件      如果D盘没有TEXT，则会新建    将其赋给一个变量fp
print('hello world',file=fp)    #将hello world输出到文件中      用file=
fp.close()                      #存储盘必须为C盘以外的其他盘
```

Ps:不进行换行输出 print（'hello'，'world'，'python'）

a+的含义：如果文件不存在就创建，存在就在文件的内容后面追加

转义字符

转义字符就是反斜杠\加上想要实现的转移功能首字母

换行	回的	水平制表（一组四个位置）	退格
\n	\n	\t	\b

水平制表：每四个字符就相当于一个\t hello已经占用了一个\t了，因此，O后面只有三个空格

回车：print('hello\nworld') 在显示器上只显示world，回车回到前面的位置把hello挤掉了——覆盖、

\b：退格，退一格，把O删掉了

原字符

不希望字符串中的转义字符起作用，就是在字符串之前加上r或R

print(r'hello\nworld')

注意事项，最后一个字符不能是反斜杠 （单引号前） 两个可以，三个也可以

二进制与字符编码

bit 8位（8个位置 一个字节）

Unicode把世界的字符都汇编在一起

要在二进制前面加上0b才可以被识别

中文英文统称字符，在计算机中可以用十进制 二进制 八进制 十六进制表示，但是在计算机中只转换成二进制

保留字和标识符

保留字：有一些单词被赋予了特定意义，这些单词在你给任何对象起名字的时候都不能用(如:true false)

import keyword

print(keyword.kwlist)

上述两行代码用于查找保留字

标识符:变量、函数、类、模块和其他对象起的名字

规则：同C语言 不能用保留字、区分大小写

变量的定义和使用

name='玛丽亚'

print(name)

变量由三部分组成

1.标识：表示对象所存储的内存地址，使用内置函数id(obj)来获取（如str)

2.类型：表示的是对象的数据类型，使用内置函数type（obj)来获取

3.值：表示对象所存储的具体数据，使用print(obj)可以将值进行打印输出

```
name='玛丽亚'
print（'标识', id(name))     #输出内存地址
print('类型', type（name)) #输出数据类型
print('值', name)       #打印输出值
#值   玛丽亚
```

常用数据类型

整数类型 int

	基本数	逢几进一	表示形式	
十进制（默认）	0，1，2，3，4，5，6，7，8，9	10	112	
二进制	0，1	2	0b1110011111	以0b开头
八进制	0，1，2，3，4，5，6，7	8	0O166	以0o开头
十六进制	0，1，2，3，4，5，6，7，8，9，A,B,C,D,E,F	16	0x11	以0x开头

```
print（‘二进制’，0b1110011）  #输出二进制对应十进制的数
```

浮点数类型 float(3.1415926)

布尔类型 bool(True=1 False=0) ——只可以取两个值，表示真或者假

```
#布尔类型
f1=True
f2=False
print(f1,type(f1))
print(f2,type(f2))
#布尔值可以用整数计算
print(f1+1)
print(f2+1)
```

字符串类型 str （人生苦短，我用python)——只要加上单引号，双引号，三引号都成为字符串类型（又称不可变的字符序列）

```
k1='人生苦短，我用PYTHON'
k2="人生苦短，我用PYTHON"
k3="""人生苦短
我用PYTHON"""
k4='''人生苦短
我用PYTHON'''
print(k1,type(k1))
print(k2,type(k2))
print(k3,type(k3))
```

Ps：单引号和双引号定义的字符串必须在一行

三引号定义的字符串 可以分布在连续的多行 三个单引号和三个双引号

```
#浮点类型
a=3.1415926
print(a,type(a))
n1=1.1
n2=2.2
print(n1+n2)        #运行结果是3.30000000000003 存储浮点数不精确
from decimal import Decimal         #解决方案 导入模块decimal
print(Decimal('1.1')+Decimal('2.2'))#3.3
```

数据类型转换

原因：将不同的数据类型拼接在一起

转换之前	转换之后	使用函数
str	int	int()
int	float	float()
float	str	str()
str	float	float()
int	str	str()
float	int	int()

```
#这段代码报错，不清楚原因
name='张三'
age=20
#print('我叫'+name+'今年‘+age+'岁')错误写法      #不可以将字符串和整形数据进行连接      +是连接符
#解决方案：类型转换
print('我叫name今年，str(age)岁')    #将int类型通过str()函数转成了str类型
```

```
print('-----------str()将其他类型转成str类型--------')
a=10
b=198.8
c=False
print(type(a),type(b),type(c))
print(str(a),str(b),str(c),type(str(a)),type(str(b)),type(str(c)))    #输出结果仍然是10，198.8，False
print('---------int()将其他的类型转int类型--------')

s1='128'
f2=98.7
s3='76.77'
ff=True
s4='hello'
print(type(s1),type(f2),type(s3),type(ff),type(s4))
print(int(s1),type(int(s1)))  #将str转成int，字符串为数字串
print(int(f2),type(int(f2))) #float转成int  只有整数部分
#print(int(s2),type(int(s3)))  #将str转成int不可以因为字符串为小数串
print(int(ff),type(int(ff)))
print(int(s4),type(int(s4)))  #将str转成int类型，字符串必须为整数数字串，非数字串不允许转换
```

函数	注意事项
str()	也可用引号转换 str(123)='123'
int()	文字类和小数类字符无法转化成整数 str('123') 将str转成int类型，字符串必须为整数数字串，非数字串不允许转换
	浮点数转化成整数，抹零取整
float()	文字类无法转成整数 float('9.9')输出结果9.9 如果字符串中发的数据是非数字串，则不允许转换
	整数转成浮点数，末尾为.0

注释

单行注释：以#开头，直到换行结束　#单行注释

多行注释：并没有单独的多行注释标记，将一对三引号之间的代码成为多行注释　'''多行注释'''

中文编码声明注释：在源文件开头加上中文声明注释，用以指定源码文件的编码格式

加上 #coding:gbk

input函数的使用

1.作用：接受来自用户的输入

2.返回值类型：输入值的类型为str

3.值的存储：使用=对输入的值进行存储

input('大圣想要什么礼物呢？')——里面这句话是一句提示语，在显示器上显示，你仍可以输入你想要输入的数据并且存储到变量中

```
#从键盘输入两个横竖，计算两个整数的和
a=input('请输入一个加数')
b=input('请输入另一个加数')
print(a+b)                                  #输出结果是1020，说明加号在这里的作用是连接
#解决方案--看a和b的数据类型--运用类型转换
#从键盘输入两个横竖，计算两个整数的和
a=input('请输入一个加数')
a=int(a)
b=input('请输入另一个加数')
b=int(b)
print(a+b)
#解决方法2
a=int(input('请输入一个加数'))
b=int(input('请输入另一个加数'))
print(a+b)
```

运算符

PS：除法运算有小数位

除法运算： /

整除运算： //（去掉小数部分，用.0代替）

幂运算：2**2　表示2的2次方

　　2***2　表示2的3次方

两个负数的整除：-9//-2=2

一正一负的整除：-9//4=-3　9//-4=-3　一正一负向下取整

　#9除以-4结果为-2.2，向下取整，则取比-2.2小的数，则为-3

余数运算：

一正一负：余数=被除数-除数*商

9%-4=-3　　=9-（-4）*（-3）=9-12=-3

-9%4=3　　=-9-4*（-3）=3

赋值运算符：运算顺序从右到左

赋值方式	
参数赋值	+=、-=、*=、/=、//=、%=
解包赋值	a,b,c=20,30,40 要求左右个数相同

用id()查看标识（内存地址）

a=b=c=20 只有一个整数对象，内存地址相同 a、b、c都指向这个内存地址

```
#交换两个变量的值
print(a,b)
a,b=b,a
print(a,b)
#比较运算符
a=10,b=20
print(a>b)   #结果 False
print(a<b)   #结果  True
print(a==b)  #结果  False
print(a! =b) #结果  True
print(a<=b)  #结果  True
print(a>=b) #结果 False
#比较变量的标识
a=10,b=10     #a已经存储了地址，而到b的时候会在内存中看一下有没有10这个对象，也指向这个id
print(a==b)   #True
print(a is b)  #True
print(a is not b)  #False  a 和b的id不相等
```

一个变量由三部分组成；标识、类型、值

运算符比较的是值

比较对象的标识（id地址）：is

布尔运算符 运算数	结果
and T T T F F T F F print（a==1 and b==2)	T F F F
or T T T F F T FF	T T T F
not T F	F T(取反)
in	
not in	
后面两个运算符可以用于查看某个东西在不在某个东西之中	

```
s='hello world'
print('w' not in s)     #False
print('s' not in s)     #True
print('w' in s)          #True
print('s' in s)         #False
```

位运算符

位运算符（将数据转成二进制进行计算）	
位与& ——对应位数都是1，结果数位才是1，否则为零	
位或| ——对应位数都是0，结果数位才是0，否则为1	
左移运算符<< ——高位溢出舍弃，低位补0	左移 位
右移运算符>> ——低位溢出舍弃，高位补0	右移 位

八进制，八个位置，前面位置没占满，需要补0

```
#  4的二进制  100--00000100
#  8的二进制  1000--00001000
#比较，对应位数，可知比较后为 00000000即0
print(4&8)     # 0
print(4|8)     #00001100对应12
print(4<<1) #向左移动一个位置
print(4<<2) #向左移动两个位置
print(4>>1) #向右移动一个位置
print(4>>2) #向右移动两个位置
```

左/右移运算符							
0	0	0	0	0	1	0	0
0（前一位零已经溢出，舍弃）	0	0	0	1	0	0	0（补0）
0	0	0	0	0	1	0	0
0（高位补零，后面低位超出的就舍弃了）	0	0	0	0	0	1	0

向左移动一位相当于乘以2 4变成8

向右移动一位相当于除以2 4变成2

运算符的优先级（有括号先计算括号）

算数 幂运算

运算 先乘除后加减 * / // % + -

符 左右移位 << >>

位运算符 & |

比较运算符 < > >= <= == !=

布尔 and

运算 or

赋值运算

程序的调试

在行号旁边打一个断点（点击），找小虫子，点击下一句的图标

程序的组织结构

```
    ### 顺序结构
```

PS：对象的布尔值：python一切皆对象，所有对象都有一个布尔值（即True或False），获取对象的布尔值，使用内置函数bool()

以下对象的布尔值为False：

False、数值（）、None、空字符串、空列表、空元组、空字典、空集合

其他的都为True

```
#测试对象的布尔值
print(bool(False))
print(bool(0))
print(bool(0.0))
print(bool(None))
print(bool(''))        #空字符串
print(bool(""))        #空字符串（只有单引号或者双引号）
print(bool([]))        #空列表
print(bool(list()))    #空列表
print(bool(tuple()))   #空元组
print(bool({}))
print(bool(dict()))    #空字典
print(bool(set()))     #空集合
```

单分支结构

语法结构

if 表达式:

```
money=1000    #余额
s=int(intput('请输入取款金额'))       #取款金额
if money>=s:
    money=money-s
    print('取款成功，余额为：',money)
```

双分支结构(二选一执行)

语法结构

```
if条件表达式:
  条件执行体1
else:
   条件执行体2
```

```
num=int(input('请输入一段整数'))
if num%2==0:
    print('是偶数')
else:
    print('是奇数')
```

多分支结构

```
if 条件表达式1:                  #由表达式1一直判断到最后
  条件执行体1
elif 条件表达式2:
  条件执行体2
elif 条件表达式N:
  条件执行体N
else:
  条件执行体N+1
```

```
score=int(input('请输入一个成绩'))
if score>=90 and score<=100:
    print('A')
elif score>=80 and score<=89:
    print('B')
elif score>=70 and score<=79:
    print('C')
elif score>=60 and score<=69:
    print('D')
elif 0<=score<=59:    #只有在python才可以这样表达 其他语言不可
    print('E')
else:
    print('请重新输入')
```

If的嵌套

```
if 条件表达式1:
  if 内层条件表达式:
    内存条件执行体1
  else:
    内存条件执行体2
else:
  if 内层条件表达式:
    内存条件执行体1
  else:
    内存条件执行体2
```

条件表达式（结合对象的布尔值判断，尤其是0——False)

条件表达式是if...else 简写

语法结构：　　X　if　判断条件 else y

运算规则：

如果判断条件的布尔值为True，执行X；如果为False，则执行y

```
'''num_a=int(input('请输入第一个整数'))
num_b=int(input('请输入第二个整数'))
if num_a>=num_b:
    print(num_a,'大于等于',num_b)
else:
    print(num_a,'小于',num_b)'''
print('用表达式进行比较')
num_a=int(input('请输入第一个整数'))
num_b=int(input('请输入第二个整数'))
print((num_a,'大于等于',num_b) if num_a>=num_b else (num_a,'小于',num_b) )     #
(11, '小于', 22)
print(str(num_a)+'大于等于'+str(num_b) if num_a>=num_b else str(num_a)+'小
于'+str(num_b))
#用加号进行字符串连接必须转换成str类型，用str()的意思不是数字变成文字，字符串也包括数字在内
```

Pass语句

Pass语句什么都不做，只是一个占位符，用在语法上需要语句的地方

什么时候用：先搭建语法结构，还没想好代码怎么写的时候

那些语句一起使用：

if语句的条件执行体

for-in语句的循环体

定义函数时的函数体

```
answer=input('您是会员吗？y/n')
if answer=='y':
    pass
else:
    pass
```

range函数的使用

内置函数range()——前面不加任何前缀，可以直接使用的一个函数

用于生成一个整数序列

创建range对象的三种方式（记住存储到一个对象中）

1. range(stop)——创建一个[0,stop)之间的整数序列，步长为1
2. range(start,stop)——创建一个[start,stop)之间的整数序列，步长为1
3. range(start,stop,step)——创建一个[start，stop)之间的整数序列，步长为step

返回值是一个迭代器对象——创建之后看不到数据，如果想要看到数据，需要使用list函数

```
r=range(10)
print(r)
print(list(r))   #用于查看range对象中的整数序列，list是i列表
#[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]默认从0开始，默认相差1（步长）
r=range(1,10)
print(list(r))
r=range(1,10,2)
print(list(r))
'''判断指定的整数在序列中是否存在  in,not in'''
print(10 in r)     #结果为False，10不在r序列中
print(9 in r)
print(10 not in r)
print(9 not in r)
```

range类型的优点：不管range对象表示的整数序列有多长，所有range对象占用的内存空间都是相同的，因为仅仅需要存储start,stop,step，只有用到range对象时，才会去计算序列中的相关元素

in和not in判断整数序列中是否存在（不存在）指定的整数

循环结构

分类：while　for-in

语法结构：

while 条件表达式：

　条件执行体（循环体）

选择结构的if和循环结构while的区别

if是判断一次，条件为True执行一行

while是判断n+1次，条件为True执行N次（执行完语句之后，又回去判断条件是否成立）

while循环的执行流程（四部循环法）——初始化的变量与条件判断的变量与改变的变量为同一个

初始化变量、条件判断、条件执行体（循环体）、改变变量

```
sum=0
a=1
while a<=100:
    if a%2==0:      #偶数和
        sum+=a
    a+=1
print(sum)

sum=0
a=1
while a<=100:
    if a%2:      #a为偶数布尔值为0 所以计算不进行if语句内的操作
        sum+=a
    a+=1
print(sum)

sum=0
a=1
while a<=100:
    if not bool(a%2):     #取反，当a为偶数，布尔值为真，进行if语句的计算
        sum+=a
    a+=1
print(sum)
```

for-in循环

in表达从（字符串、序列等）中依次取值，又称为遍历

for-in遍历的对象必须是可迭代对象

语法结构

for 自定义的变量 in 可迭代对象：

　循环体

循环体内不需要访问自定义变量，可以将自定义变量替代为下划线

目前可迭代对象有序列、字符串

```
for item in 'python':
    print(item)        #从字符串中取出每一个字母进行输出，并赋给item，并输出item
    #P y t h o n
for r in range(10):      #打印 0 1 2 3 4 5 6 7 8 9
    print(r)
    #循环体内不需要访问自定义变量，可以将自定义变量替代为下划线
for _ in range(5):       #循环执行5次
    print('人生苦短，我用python')
```

```
#使用for循环，计算1--100之间的偶数和
sum=0
for item in range(1,101):
    if item%2==0:
        sum+=item
print(sum)
#输出100--999之间的水仙花数
'''153=1*1*1+5*5*5+3*3*3'''
for item in range(100,1000):
    ge=item%10
    shi=item//10%10
    bai=item//100
    if(ge**3+shi**3+bai**3==item):
        print(item)
```

流程控制语句break

用于结束循环结构，通常与分支结构if一起使用

for ... in...:

  ...

 if ...:

    break

while 条件:

    ...

if ...:

    break

```
for item in range(3):       #循环三次 0 1  2
    pwd=input('请输入密码：')
    if pwd=='8888':           #判断语句一定要在常量那加上单引号
        print('密码正确')
        break
    else:
        print('密码不正确')

item=0
while item<3:
    pwd=input('请输入密码')
    if pwd=='8888':
```

```
        print('密码正确')
        break
    else:
        print('密码不正确')
    item+=1
```

流程控制语句continue

用于结束当前循环，进入下一次循环，通常与分支结构中的if一起使用

for ... in...:

...

if ...:

continue

while 条件:

...

if ...:

continue

```
for item in range(1,51):
    if item%5!=0:      #记得加上冒号
        continue
    else:
        print(item)
```

else语句

if...:　　#if表达式不成立时执行else

...

else:

...

while...:　　#没有碰到break时执行else 循环的正常次数执行完就会执行else

...

else:

...

for...:

...

else:

...

```
for item in range(3):
    pwd=input('请输入密码：')    #input前面不能加上int()函数，否则会报错
    if pwd=='8888':
        print('密码正确')
        break
    else:
        print('请重新输入密码')
else:        #与for-in搭配
    print('三次密码均使用错误')

a=0
while a<3:
    pwd=input('请输入密码：')
    if pwd=='8888':
        print('密码正确')
        break
    else:
        print('请重新输入密码')
    a+=1
else:
    print('三次密码军输入错误')
```

```
print()      #换行
print(3)
print()       #换行
print(1)
#输出一个三行四列的矩形
for i in range(1,4):
    for j in range(1,5):        #不换行输出
        print('*',end='\t')     #end='\t'表示空格 注意，此处为反斜杠
    print( )           #每一行输出完成之后换行
#打印一个直角三角形
for i in range(1,10):
    for j in range(1,i+1):
        print('*',end='\t')
    print()

for i in range(1,10):
    for j in range(1,i+1):
        print(i,'*',j,end='\t')
    print()
```

二重循环中的break和continue用于控制本层循环

break：推出内层循环

continue：继续执行本层循环，只是推出了该次循环，对内层循环没有影响

```
for i in range(5):        #代表外层要执行5次
    for j in range(1,11):
        if j%2==0:
            break      #退出当前循环结构 循环，执行完本次循环，不进行下一次循环
        print(j)      #再缩进两格是因为在if层下会被认为是当前语句

for i in range(5):
    for j in range(1, 11):
        if j%2==0:
            continue       #退出该次循环
        print(j,end='\t')
    print()
```

列表

为什么需要列表

1.变量可以存储一个元素，而列表是一个“大容器”可以存储N多个元素，程序可以方便地对这些数据进行整体基本操作

2.列表相当于其它语言中的数组

可以存储不同类型的数据

变量存储的是对象的ID值（地址）——存储对象的引用地址

变量：ID+类型+值

```
a=10
lst=['hello','world',11]
print(id(lst))        #查看列表的地址
print(type(lst))      #查看列表的类型(list)
print(lst)            #输出列表
```

列表的创建（使用中括号、调用内置函数list()以及列表生成式）

列表需要使用中括号[],元素之间使用英文的逗号进行分隔,且需要赋值给对象（变量）还有赋值运算符

lst=list(['hello','world',11])

变量实际上存储的是列表的ID

```
lst=list(['hello','world',11])
print(lst)
```

列表的特点

1.列表元素按顺序有序排序

2.索引映射唯一一个数据　　lst[0]——从前往后索引　　lst[-1]——从后往前索引 依次为-1、-2

3.列表可以存储重复数据

4.任意数据类型混储

5.根据需要动态分配内存和回收内存

```
lst=list(['hello','world',11])
print(lst)
print(lst[0],lst[-3])
```

列表的查询操作

获取列表中指定元素的索引——index

1.如果列表中有 相同元素值返回列表中相同元素的第一个元素的索引

2.如果查询的元素在列表中不存在，则会抛出ValueError

3.还可以在指定的start和stop之间进行查找

```
lst=list(['hello','world','hello',11])
print(lst.index('hello'))       #中间使用的是实心点
#print(lst.index('python'))
#print(lst,index('hello',1,3))          #不包括3
print(lst.index('hello',1,4))
```

获取列表中的单个元素

1.正向索引从0到N-1　　举例：lst[0]　lst[N-1]

2.逆向索引从-N到-1　　举例：lst[-N]　lst[-N]

3.指定索引不存，抛出IndexError

获取列表中的多个元素

语法格式：　列表名[start:stop:step]

切片操作

切片的结果：原列表片段的拷贝

切片的范围：[start，stop）——切出来之后变成一个新的列表对象，另存，只拷贝对象，不拷贝ID

step默认为1：简写为[start,stop]

step为正数：[:stop:step]:切片的第一个元素默认是列表的第一个元素

[start::step]——切片的最后一个元素默认是列表的最后一个元素

从start开始往后计算切片

step为负数：[:stop:step]——切片的第一个个元素默认是列表的最后一个元素

[start::step]——切片的最后一个元素默认是列表的第一个元素

从start开始往前计算切片

```
lst=[10,20,30,40,50,60,70,80]
print('原列表',lst)
print('原列表',id(lst))
lst2=lst[1:6:1]
print('切的片段',id(lst2))
print(lst[1:6])  #默认步长为1
print(lst[1:6:])
```

```
print(lst[1:6:2]) #步长2
print(lst[:6:2]) #默认从0开始
print(lst[1::2])
print('步长为负数的情况')
print(lst[::-1])    #逆序输出
print(lst[7::-1])
print(lst[6:0:-2])  #不包括lst[0]
```

列表元素的查询操作

判断指定元素在列表中是否存在

元素 in 列表名

元素 not in 列表名

列表元素的遍历

for 迭代变量 in 列表名:

操作

```
lst=[10, 20,30,40,50,60,70,80]
print(10 in lst)
print(100 in lst)
print(10 not in lst)
print(100 not in lst)
#遍历列表中的元素
for item in lst:
    print(item)
```

列表元素的增加操作

增加操作	切片	在列表的任意位置添加至少一个元素
	append()	在列表的末尾添加一个元素
	extend()	在列表的末尾至少添加一个元素
	insert()	在列表的任意位置添加一个元素

```
lst=[10, 20,30]
print(id(lst))
lst.append(100)
print(lst)
print(id(lst))      #标识都相同，说明没有创建新的列表对象
#lst指向的id首都一样，只不过尾巴会加上int相对应的字节
lst2=['hello','world']
lst.append(lst2)      #把lst2整体作为一个元素放到了lst中
print(lst)
lst.extend(lst2)    #把lst2中的每一个元素都单独放到lst的末尾，一次性添加多个元素
print(lst)
#在任意位置上添加一个元素
lst.insert(1,90)
print(lst)
#切片,在任意位置上添加N多个元素
lst3=[True,False,'hello']
```

```
lst[1:]=lst3  #没有结束，说明一直到最后一个元素都会被删掉，没有步长，用等于号
print(lst)    #删除的部分用新的列表去替换
```

列表元素的删除操作

remove（）	一次删除一个元素、重复元素值删除第一个、元素不存在抛出ValueError
pop()	删除一个指定索引位置上的元素、指定索引不存在抛出IndexError（超出索引范围）、不指定索引，删除列表中的最后 一个元素
切片	一次至少删除一个元素
clear()	清空列表（会产生新的列表对象）
del	删除列表

```
lst=[10, 20,30,40,50,60,30]
lst.remove(30)   #以列表中移除一个元素，如果有重复元素只移除第一个元素
print(lst)
lst.pop(1)
print(lst)
lst.pop()  #如果不指定参数（索引），将删除列表中的最后一个元素
print(lst)
#切片--会产生一个新的列表对象
newlst=lst[1:3]
print('原列表',lst)        #原列表不发生任何改变
print('切片后的列表',newlst)
#如何用切片使其不产生新的列表对象，但是删除原列表的内容
lst[1:3]=[] #用空列表替代，[1,3）的位置
print(lst)
#清楚列表中的所有元素
lst.clear()
print(lst)
#del将列表对象删除
del lst
print(lst)
```

列表元素的修改操作（改变指向的ID）

为指定索引的元素赋一个新值

为指定的切片赋予一个新值

```
lst=[10,20,30,40]
#一次修改一个值
lst[2]=100
print(lst)
lst=[10,20,30,40]
#一次修改一个值
lst[2]=100
print(lst)
lst[1:3]=[300,400,500]    #把20、30去掉，用300、400、500替代
print(lst)
```

列表元素的排序操作

常见的两种方式

1.调用sort（）方法，列表中的所有元素默认按照从小到大的顺序进行排序，可以指定reverse=True，进行降序排序

2.调用内置函数sorted()，可以指定reverse=True，进行降序排序，原列表不发生改变——会产生新的列表对象，原列表不发生任何改变

PS：内置函数，啥都不需要，直接拿过来用

```
lst=[20,40,10,98,54]
print('排序前',lst)
lst.sort()
print(lst)
#用id()可以知道没有产生新的列表，因为一样
#排序是在原来的基础上进行的
#通过指定关键字参数，将列表中的元素进行降序排序
lst.sort(reverse=True)     #降序排序
print(lst)
lst.sort(reverse=False)    #升序排序
print(lst)
print('使用内置函数进行排序')
lst=[20,40,10,98,54]
newlist=sorted(lst)
print(newlist)
#指定关键字参数，实现列表参数的降序排序
desc_list=sorted(lst,reverse=True)
print(desc_list)
up_list=sorted(lst,reverse=False)
print(up_list)
```

列表生成式（生成列表的公式）------列表中的元素要有一定的规则

语法格式： [i*i for i in range(1,10)]

range(1,10) 会产生一个1-9的整数序列

```
lst=[i for i in range(1,10)]  #方括号中存的是产生的整数序列，整数序列是i，列表中存储的就是i的值
print(lst)
lst2=[i*i for i in range(1,10)]  #i*i的意思就是表面意思    i*i被称为表示列表元素的表达式(列表中真正包含的那个元素的值)
print(lst2)
lst3=[i*2 for i in range(1,6)]
print(lst3)
```

字典

python内置的数据结构之一，与列表一样是一个**可变序列**（可以执行增删改操作）

以键值对的方式存储数据，字典是一个无序的序列，列表是一个有序序列其中序指的是排列，第一个放进列表的就在第一个位置上

scores={'张三'：100，'李四'：98}冒号之前称为键，冒号之后称为值，即键值对

字典需要经过哈希函数计算key再决定存储位置，因此放在字典当中的键必须是一个不可变序列，如int,str，不可执行增删改操作

eg：s='hello' 如果想要给s再加上一个python，那就重新开辟一段存储空间'hello python'

字典的实现原理：字典的实现原理与查字典类似，python中字典是根据使用hash函数计算key查找value所在的位置

字典的创建

最常用的方式：

使用花括号　　scores={'张三'：100，'李四'：98} #记得存储到变量当中

使用内置函数　dict(name='jack',age=20)

```
scores={'张三':100,'李四':98}
print(scores)
print(type(scores))

student=dict(name='jack',age=20)     #等号左侧是键，等号右侧是值，字符串记得加上单引号、赋值给变量
#{'name': 'jack', 'age': 20}     #只要是字符串都会有单引号
print(student)

#空字典
d={} #无键值对
print(d)
```

字典中元素的获取

一：[] 举例：scores['张三']

二：get() 举例：scores.get('张三')

区别：

[]如果字典中不存在指定的key，抛出keyError异常（键不存在）

get()如果字典中不存在指定的key，并不会抛出keyError而是给出None，可以通过参数设置默认的value，以便指定的key不存在时返回value的值

```
scores={'张三':100,'李四':98}
print(scores['张三'])   #给出丈夫输出妻子
print(scores.get('张三'))
print(scores.get('麻七',99))  #99是在查找'麻七'所对的value不存在时，提供的一个默认值
```

key的判断

in ——指定的Key在字典中存在返回True——'张三' in scores

not in ——指定的key在字典中不存在返回True——'Marry' not in scores

字典元素的删除

del scores['张三']

字典元素的新增、修改

scores['Jack']=90

```
scores={'张三':100,'李四':98}
print('张三' in scores)
print('张三' not in scores)
print('Marry' in scores)
del scores['张三']
print(scores)
scores.clear()      #清空所有元素
print(scores)
scores={'张三':100,'李四':98}
scores['陈六']=98
print(scores)
scores['陈六']=100    #也可以相当于修改
print(scores)
```

获取字典视图的三个方法

keys()——获取字典中所有的key

values()——获取字典中所有value

items()——获取字典中所有的key，value对

```
scores={'张三':100,'李四':98}
keys=scores.keys()
print(keys)  #视图
print(type(keys))    #keys类型
print(list(keys))    #转换成由键组成的列表
values=scores.values()
print(list(values))
print(type(values))   #values类型
items=scores.items()
print(items)    #dict_items([('张三', 100), ('李四', 98)])--被一组方括号括起来了方括号是原组的意思
print(list(items))  #[('张三', 100), ('李四', 98)]两个元素，每个元素是一组元组（）
```

字典元素的遍历

for item in scores:

  print(item)   #输出的是键

```
scores={'张三':100,'李四':98}
for item in scores:
    print(item) #输出键

for item in scores:
    print(item,scores[item],scores.get(item))
```

字典的特点

1.都是键值对

2.key不允许重复，value可以重复

3.元素是无序的

4.key重复会出现值覆盖的情况

5.key必须是不可变对象——int str （列表是可变的）

6.字典可以根据需要动态地伸缩

7.字典会浪费较大的内存——存放位置之间可能会有很多个空没存放，是一种使用空间换时间的数据结构

字典会根据键去计算存储位置

```
scores={'张三':98,'张三':100}
print(scores)
scores={'张三':98,'Marry':98}
print(scores)
lst=[10,20,30]
lst.insert(1,100)
print(lst)
```

字典生成式

内置函数zip()

用于将可迭代的对象作为参数，将对象中对应的元素打包成一个元组，然后返回由这些元组组成的列表

现在有两个列表 items=['fruit','books','others']

  prices=[96,78,85] 把items作为key，prices作为values

如果key和value个数不相等，那么zip()在打包的时候就会元素少的为基准

```
items=['Fruit','Books','Others']
prices=[10,20,30]
d={item:price  for item ,price in zip(items,prices)}  #item是单数，代表单个的意思
    #items:prices 表示遍历的items是键，prices是做值的
print(d)
d={item.upper():price for item ,price in zip(items,prices)}    #大写item.upper()
print(d)
```

元组（数据结构）

定义：python的数据结构之一，是一个不可变序列

可变序列：列表、字典 可以对序列执行增、删、改操作，对象地址不发生更改

不可变序列：字符串、元组 没有增、删、改操作，**操作之后对象地址发生改变**

语法结构：

t=('python','hello',90) 括号内的元素就是元组的内容——与列表的区别仅在于一个是方括号，另一个是（）

元组的创建方式——输出括号全部内容，原封不动

t=（‘python','hello',90）

使用内置函数tuple   t=tuple(('python','hello',90))

只包含一个元组的元素需要使用逗号和小括号 t=(10,) ,如果不加上逗号，计算机会以为这是他本身的数据类型，哪怕你加上括号了如10 int

```
t=('python','hello',90)
print(type(t))
print(t)
t=tuple(('python','world',98))
print(t)
print(type(t))
t2='python','world',98    #省略了小括号也可以创建元组
print(type(t2))
print(t2)
t3=('python',)
print(t3)
print(type(t3))
lst=[]
lst1=list()
d={}
d2=dict()
t4=()
t5=tuple()
print('空列表',lst,lst1)
print('空字典',d,d2)
print('空元组',t4,t5)
s='python'
s=s+'world'    #python world
```

为什么要将元组设计成不可变序列

1.在多任务环境下，同时操作对象时不需要加锁——可查看，但是不可以使用增删改操作，不会对内容产生改变

2.因此，在程序中尽量使用不可变序列

一旦创建不可变类型的对象，对象内部的所有数据就不能再被修改了，这样就避免了由于修改数据而导致的一个错误

注意事项：元组中存储的是对象的引用

1.如果元组中对象本身存储的是不可变对象，则不能再引用其他对象（增添元素）

2.如果元组中的对象是可变对象，则可变对象的引用不允许改变，但数据可以增删改

元组不允许修改元素

```
t=(10,[20,30],9)  #[20,30]是可变的，其他两个元素不可变
#不可以将t[1]=100因为中间元素存的不是数值，而是引用（指向），可以向列表中添加元素
t=(10,[20,30],9)
print(t)
print(type(t))
print(t[0],type(t[0]),id(t[0]))
print(t[1],type(t[1]),id(t[1]))
print(t[2],type(t[2]),id(t[2]))
t[1].append(100) #由于[20,30]列表，而列表是可变序列，所以可以向列中添加元素，而列表的内存地址不变
print(t,id(t[1]))   #列表地址不变
```

元组的遍历

用索引，如t[1]、t[2]——要知道索引的范围

元组是可迭代对象，所以可以用for... in...进行遍历

t=tuple(('python','hello',90))

for item in t:

  print(item)

```
t=('python','hello',90)
print(t[0])
print(t[1])
print(t[2])
for item in t:
    print(item)
```

集合（set）

定义：

1.python语言提供的内置数据结构

2.与列表、字典一样都属于可变类型的序列

3.集合是没有value的字典，**只有**键，也运用hash函数计算

集合的创建方式

s={'python','hello',98}——使用花括号，类似于列表

使用内置函数set()

注意事项：集合当中的元素不允许重复

集合是无序的，如：最后一个放的元素，可能在输出时是第一个输出的

```
s={2,2,3,4,5,6,7,7}
```

```
print(s)
s1=set(range(6)) #range(6)产生一个0--5的整数，将其变成集合
print(s1,type(s1))
print(set([3,4,4,5,6])) #将列表当中的元素转成集合中的元素同时去掉了重复元素
s2=set((3,4,5,5,6,7))
print(s2) #将原组类型的元素转成集合类型
s3=set('python')
print(s3,type(s3)) #{'o', 'p', 'n', 't', 'h', 'y'} <class 'set'>
s4=set({12,323,2123,12,12,123,12121323})
print(s4,type(s4))
s6={}  #空集合？空字典？
print(type(s6))   #字典类型
s7=set()  #定义一个空集合
print(type(s7))
```

集合的相关操作

集合元素的判断操作： in或not in

集合元素的新增操作： 1. add()，一次添加一个元素

2.update()至少添加一个元素

集合元素的删除：1.remove(),一次删除一个指定元素，如果指定元素不存在则会抛出异常KeyError

2.discard()，一次删除一个指定元素，如果指定元素不存在也不会抛出异常

3.pop()，一次只删除一个任意元素，括号内不能指定参数

4.clear()，清空集合

```
s={10,20,30,40,50,60,70,80,90,100}
print(10 in s)
print(23123 in s)
s.add(101)
print(s)
s.update({200,400})  #添加多个元素记得用上花括号
print(s)
s.update([123,222]) #可以添加列表
print(s)
s.update((12313,11111111)) #可以添加元组
print(s)
s.remove(100)
print(s)
s.discard(500)
print(s)
s.discard(11111111)
print(s)
s.pop()
print(s)
s.pop()
print(s)
s.clear()
print(s)
```

集合间的关系

关系	操作	
两个集合是否相等	可以使用运算符==或！=	
一个集合是否是另一个集合的子集	调用issubset进行判断	B是A的子集
一个集合是否是另一个集合的超集	调用issuperset进行判断	A是B的超集（意思就是说B是A的子集）
两个集合是否没有交集	调用isdisjoint进行判断	

```
s={10,20,30,40}
s2={30,40,20,10}
print(s==s2)
print(s!=s2)
s1={10,20,30,40,50,60}
s2={10,20,30,40}
s3={10,20,90}
print(s2.issubset(s1))
print(s3.issubset(s1))
print(s1.issuperset(s2))
print(s1.issuperset(s3))
print(s1.isdisjoint(s2))
print(s2.isdisjoint(s3))
s4={100,200,300}    #有交集False
print(s4.isdisjoint(s1))   #没有交集 True
```

集合的数学操作

差集：A集合减去A与B集合有交集的数据所剩的元素

对称差集：B集合减去A与B集合有交集的数据所剩的元素

操作之后原集合不变

```
s1={10,20,30,40}
s2={20,30,40,50,60}
print(s1.intersection(s2))  #求交集
print(s1 & s2) #求交集
print(s1.union(s2)) #求并集
print( s1| s2)#求并集
print(s1.difference(s2)) #求差集
print(s1-s2) #求并集
print(s1.symmetric_difference(s2)) #求对称差集
print(s1^ s2) #求对称差集(数字键6)
```

集合生成式

元组无生成式，因为元组不可变序列

将列表生成式改为花括号即可

{ i*i for i in range(1,10)}

```
lst=[ i*i  for i in range(10)]
print(lst)
s={ i*i for i in range(10)}
print(s)
```

字典当中的键实际上就是set