字符串赋值的时候需要加上单引号***’’***

print(a,b,sep='\n') 输出ab的值，中间用\n隔开 sep即separation

三种编码方式： ASCII、Unicode、UTF-8

ASCII 一个字节 Unicode 两个字节 UTF-8 可变长度

在计算机内存中，统一使用Unicode编码，当需要保存到硬盘或者需要传输的时候，就转换为UTF-8编码。

用记事本编辑的时候，从文件读取的UTF-8字符被转换为Unicode字符到内存里，编辑完成后，保存的时候再把Unicode转换为UTF-8保存到文件。

ord(‘ ’) 读取该字符的整数表示

chr( ) 将整数表示为对应字符 只能在交互式上使用？

字符串类型 str 以字节为单位的为 bytes 后者以带b前缀的单或双引号表示。

例如：X=’ABC’ 这里X为字符串 X=b’ABC’ 此处A为bytes型

字符串类型占用空间较多，例如使用UTF-8编码，每个字符站占两个字节；但bytes型永远只占一个字节

encode(‘’)可以将字符串转化成相应编码方式的编码 单引号中为编码方式，一般为ascii或utf-8。（python中，编码以Unicode的形式储存在内存中）

例如： ’ABC’.encode(‘ascii’) 结果为b’ABC’

反之，将bytes型转化可以用 decode(‘’)函数

例如： b’ABC’.decode(‘ascii’) 结果为’ABC’

len() 可以计算相应str型数据的字符数或bytes型数据的字节数。

例如：len(‘相应’.encode(‘utf-8’)) 结果为6

*#!/usr/bin/env python3*

*# -\*- coding: utf-8 -\*-*

第一行注释是为了告诉Linux/OS X系统，这是一个Python可执行程序，Windows系统会忽略这个注释；

第二行注释是为了告诉Python解释器，按照UTF-8编码读取源代码，否则，你在源代码中写的中文输出可能会有乱码。

格式化输出 使用%进行站位 %s 输出字符串 %d 输出整数 %f 输出浮点数等。此部分与C语言相通。输出%时要用%%

print('一年来成绩进步%d'%r)

%f 可以定义输出小数的位数 例如 %.2f 为输出两位小数。

%d 可以定义是否补零 例如 %02d 输出两位数，不足两位的前方补零，超过的输出原始数据。 %2d 输出两位数，不足的在前方添加空格，超过的输出原始数据

%s 输出字符串

%x 十六进制整数

0x 开头为十六进制

0o开头为八进制

0b开头为二进制

Python只能输入字符串，然而字符串不能用于运算，需要用int()、float()等强制转换。

list型数据，类似矩阵或者集合。（学名列表）

定义方式A=[ , , , ]其中元素可以是字符串、数字或者另一个list，当然也可以混编。

其中第一个元素索引或者说序号为0。

可以用-1最为倒数第一个元素的序号，以此类推倒数第二第三也可以如此表示。

对一个list可以追加元素，用.append()或.insert()

其中，append只能添加元素至list末尾；insert可以插入在指定位置，

如A.insert(-1,’SB’) 表示在A倒数第一个位置插入数字字符串SB。实际上变成了倒数第二个，原来的倒数第一还是倒数第一。

若想删除指定位置元素，则用A.pop() 即弹出命令。 括号中数字为元素序号，若不写则为最后一个元素。这一指令可将被删除的值赋给另一变量

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

b=a.pop(5)  #b=5

list替换元素直接赋值即可。

与list类似，有tuple但是tuple一旦定义就被固定下来，无法删除、添加或者改变其中的元素。

tuple定义方式为A=( , , ) 为小括号（学名元组）

定义空tuple可以如此：A=()

但定义只有一个元素的tuple必须添加逗号，如：A=(1,) 否则会与数学运算小括号混淆。

元组内的元素不可变化，但元组本身可以被重新赋值，且元素内部可以被重新赋值

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

a[0]=1                  #错误，无法对元组元素赋值

a[9][0]=1               #正确，对元组元素的元素可以赋值

a[9].append(1)          #正确，元组的元素可以进行其他操作

a=(5,10,15,20)          #正确，可以重新赋值元组

判断语句，*if: else: elif:*

if语句执行有个特点，它是从上往下判断，如果在某个判断上是True，把该判断对应的语句执行后，就忽略掉剩下的elif和else

elif与if应该在同一缩进位置。因为if语句是运行缩进块代码。

循环语句：

*for in ：* 循环 第一个空位一个变量，第二个空为一个list或者tuple

把每个元素代入变量x，然后执行缩进块的语句。

range(x)函数可以生成一个小于x的，从零开始的整数数列，然后再使用：

list( ) 可以转换成list类型 左闭右开，

例如：list(range(5)) 结果为 0 1 2 3 4

*while 条件 ：* 循环， 只要条件满足，就一直执行缩进块的语句

dict语句 使用花括号{ }进行定义、赋值等操作。括号里边各个元素（key）与其对应值（box、value）之间应用冒号：隔开；元素之间用逗号，隔开。

元素可以是字符串、数字等；对应值可以是数字、字符串等

可以对dict内部元素进行赋值，但会覆盖之前的值。

可以将dict元素理解为指针。

使用in语句检测相应的key是否存在于dict中

使用.get()语句查询某个元素是否在dict中。并且可以在括号内设定一个返回值。

使用.pop()语句来删除dict中的一个元素及其对应值

dict中可以定义重复元素，但以最后定义的值为准。

dict的key必须是**不可变对象**

set与dict类似，但只有元素，无对应值。使用set([ ]) 内方外圆定义

可以使用.add()语句来添加set元素，可以添加重复元素，但无用。

可以使用.remove()语句来删除set元素。

.replace( ) 可以用来替换字符串或者list中的元素

List与tuple中可以放重复元素，不会被冲掉

abs( ) 返回绝对值，括号内仅一个数字元素

max() 返回最大值，括号内可以任意多的数字元素、或者tuple、list

hex() 转换为16进制字符串

oct() 转换为8进制字符串

bin() 转换为2进制字符串

sum() 返回求和，对象为列表或元组

函数名也可更改，如a=abs， a(10)与abs(10)等价

函数之间可以嵌套

函数参数类型：未知参数、可变参数、关键字参数、命名关键字参数

输出多个变量时要用括号括起来

关键字参数：即在调用函数时将实参名称写入

def xMinusy(x,y):

    return  x-y

xMinusy(y=7,x=5) #这两个表达等价

xMinusy(5,7)

默认参数：在没有提供实参的情况下为形参赋值：但只能放在末尾

def xMinusy(x,y=10):

    return  x-y

xMinusy(5)         #这两个表达等价

xMinusy(5,10)

对于函数来说，可以修改列表类型的实参，除非在函数内部复制一个。但不能修改其他类型实参。

对于任意数量的实参，用一个星号表示：

def make\_pizza(size,\*materials):

    size=int(size)

    print("You just ordered a %d inches pizza with these:"%size)

    for i in materials:

        print("-"+i)

make\_pizza(15,'water','ash','pepper')\

此处实际上生成了一个元组

对于任意数量的关键字函数，用两个星号表示：

def make\_pizza(size,\*\*materials):

    size=int(size)

    print("You just ordered a %d inches pizza with these:"%size)

    for i,j in materials.items():

        print("%d unit of %s"%(j,i))

make\_pizza(15,water=4,ash=1,pepper=8,gold=8)

这里实际上生成了一个键值对，但键只能用字符串型数据？

使用 def 来定义新函数 例如： def myabs(x):

x为定义函数变量，与定义函数语句外的x不冲突

定义函数时一定要有return 否则无法返回值

pass语句 用来站位，表示暂时将此位置空出，可以防止程序报错

定义函数时可以添加默认值，一般来说默认值要设置在靠后的位置。若输入时有多个默认值，则需要对不按顺序的输入添加默认值变量名=….的内容

默认参数必须指向不变对象。

可变参数：指传入的参数个数是可以变化的。

可以在list或者tuple型变量前加\*将其变成可变参数

关键字参数：扩展函数功能。并且可以将接收到的数据组装形成dict。因此对其进行输入时也应该按照dict的形式进行输入。在变量前加上\*\*即可定义一个关键字参数。

命名关键字参数：输入时必须附带变量名；定义时需要有\*进行分隔。

切片操作：L[5:10] 从序号5开始开始，取到序号10（不包括三），一共取出5个。

如果从零开始，可以将零省略：L[:3]

倒数也可以：L[-10:-5] 表示从倒数第十开始取到倒数第五（不含倒数第五），一共五个

同样如果取到倒数第一，可以省略-1 L[-5:]

可以设置每n个取一个：l[10:20:5] 从第11个开始到第20个，每五个去一个

若不写前两个参数，则代表全都取出：l[::5]

若三个都不写，则代表所有数据都取出，故l等价于l[::]也等价于l[:]

l[::-2]代表从后往前，每两个数据去一个

总结来说：三个参数意义为：[起始:结束:步长] 左闭右开区间，步长正负代表顺序或逆序

对tuple类数据也可以进行类似操作

对字符串数据也可以进行类似操作

对list切片，结果仍未list，对tuple切片结果仍为tuple

生成要素+for循环+判断语句=列表生成器 注意生成物需转化成list才能作为list使用

可以用两层或多层循环，生成全排列：

l=list([m+n+l for m in 'AB' for n in 'XY' for l in '12'])

for n in l:

print('%s'%n)

对dict类型数据进行迭代，默认迭代key，可在变量后边添加.values迭代value或者添加.items同时迭代key与value

使用collections 模块中 Iterable类型判断可以判断某一数据是否可迭代

from collections import Iterable

s=isinstance(l,Iterable)

print('%s'%s)

min,max = L[0],L[0] 可使用此种方式进行赋值

可以直接用max、min语句在一个tuple或list中找最大最小值

.lower() 可以将字符串中大写字母改成小写字母

>>> L=['ASD','BS','TRe']

>>> L[0].lower()

'asd'

>>> [s.lower() for s in L]

['asd', 'bs', 'tre']

但对于非字符串类型，使用此函数会报错

isinstance可以用来判断函数类型

isinstance(x,int) 函数会输出true或者false

将列表生成式的[ ]改变为( )即可变成generator

若遇到类似斐波那契数列需要同时利用多个变量并且要对这几个变量同时赋值时，可以使用：

a,b=b,a+b

此做法等价于：

t=a

a=a+b

b=t

可以省略临时变量t，少写两行代码

yield 与print功能类似，但用在generator语句中

杨慧三角形可以在一个数列前边加零，另一个数列后边加零，然后两者相加即可

一个函数能够把其它函数作为参数使用，这个函数就是高阶函数。

python3中取消了全局命名空间中的reduce()函数，将reduced()放到了functools模块中，要使用reduce()的话，要先加载functools.

from functools import reduce

map(function,list/tuple) 表示把某一函数作用在整个list或者tuple上

reduce(function,list/tuple) 表示

reduce(f, [x1, x2, x3, x4]) = f(f(f(x1, x2), x3), x4)

filter() 用于过滤数列，若函数作用结果为true，则返回，否则不返回。

map、filter返回后的结果需list转化为列表

reduce返回结果为一个单独元素，故不需要转为list

Python内置的sorted()函数就可以对list进行排序：

默认按照由小到大排序。

可以在后方加入key函数，表示将key函数运用在每个list元素之后再排列，但不改变原有list本身的元素。例如：

sorted([], key=abs)

结果为：5，9，-12，-21，36

若希望反向排列，则加入第三个函数：reverse=True，例如：

sorted([36, 5, -12, 9, -21], key=abs, reverse=True)

其中True可以替换，若为0以为的数字，则与True相同，若为0或者False则相当于仍正向排列。但不可以是字符串或者其他类型数据

利用函数返回函数时要注意，其中包含的变量若有其他运算会被改变。

进行复制操作：a=f() 后； a只是函数， a() 才是函数结果。即，再次调用才会进行计算。

f=lambda x:x\*\*2

与

def f(x):

return x\*\*2

等价

int 可以将字符串转化为十进制整型，默认传入的字符串为十进制

可以用int(‘1111’，n) 使默认字符以n进制

nt2 = functools.partial(int, base=2)

函数意义为使int函数base（进制）的默认值为2

即该函数可以固定某些函数的默认参数

输出多个参数时，要在%后边用括号括起来：

print('%d×%d=%d'%(j,i,i\*j),end=' ')

在某一函数内定义的函数，在该函数外不可用

\_\_name\_\_可以用来获取函数名称，例如f.\_\_name\_\_ 此处为双下划线

装饰器，可以理解为在不改变现有函数代码的情况下给函数添加其他功能，如在函数运行前后添加时间戳、计数等

以函数为参数（一般情况下还要以函数作为return对象）的函数即为装饰器。

被装饰器装饰过的函数其名称会改变,可以在装饰器定义时通过

def decorator1(a):

@wraps(a)

def sb():

print('before!')

a()

来防止此现象发生。但要提前

from functools import wraps

或

import functools

但此时需使用functools.wraps

@functools.wraps(a)

可以在def上一行@某个装饰器进行装饰例如：

@decorator1

def f():

print('fffffff')

return 0

相当于执行了f = decorator1 ( f )

import time 之后可以利用：

print(time.strftime("%Y-%m-%d %H:%M:%S", time.localtime()))

打印出2016-03-20 11:45:39格式的时间

time.sleep() 可以使线程暂停一段时间，中间为秒数，无返回值.

if \_\_name\_\_=='\_\_main\_\_':

可以用来判断当前程序是否为主函数，一般用于测试

利用class 创建群类，且后紧接 object（实际上写不写无所谓，但写是好习惯）；例如：

class Student(object):

一般来说类名的第一个字母需大写。

可以在类中定义函数，进而可以在类内部访问数据，调用时用(实例名.函数名)即可

此处的函数参数必须为self，但外部调用时括号内部无参数；例如：

def print\_score(self):

print('%s:%s'%(self.name,self.score))#类中定义时,实例名为sam

sam.print\_score()#外部调用时

若需要强制绑定类的属性，可以在其后添加\_\_init\_\_函数；该函数第一个参数必须为self，其他随意。但绑定之后，所有的实例都必须有相应的属性。例如：

class Student(object):

def \_\_init\_\_(self,name,score): #定义时a=Student(‘Sam’,99)

self.name=name #可以添加参数默认值

self.score=score #可在此添加默认值，如self.hieht=160

self.name 是外部调用时使用的变量名，

等号后边的name相当于函数的参数在此处赋值给了self.name

可以给某一实例添加其他属性，但只对该实例有效。如：a.weight=120

可以在\_\_init\_\_定义时在变量名前加上双下划线，将该变量设置为私有变量，使外界无法访问，也无法进行修改。

若必要对私有变量进行访问与修改，则需在类定义时添加相应的访问、修改函数。

class Student(object):

def \_\_init\_\_(self, name, gender):

self.\_\_name = name

self.\_\_gender = gender

但实际上可以通过 实例名.\_类名\_\_属性名 进行强制访问。例如：

bart.\_Student\_\_gender

定义新类时的object可以被替换成已有的类名，此时新类为子类，原类为父类。好处是子类可以继承父类所有属性。同时可以在子类下定义与父类相同的函数，从而覆盖父类原函数。

定义一个以类为接受参数的函数时，定义不需加小括号，但调用时需要小括号；例如：

def runX2(Animal):

Animal.run()

runX2(Cat) #不可运行

runX2(Cat()) #可运行

并且该函数接受的类中只需要有run函数即可运行，其他均无限制。

可以在程序中设置复数，例如a=1+2j 一般虚部用j表示

a.real 提出实部1 a.imag 提出虚部2

complex（,） 用于创建一个虚数，前后分别为实部虚部

// 表示除法取商

% 除法取余

\*\* 乘方

and 、or 、 not 为逻辑运算符号

in、not in 用于判断某元素是否在列表、元组内

& 按二进制位进行 与 操作

| 按二进制位进行 或 操作

^ 按二进制位进行 异或 操作

~ 按二进制位进行 取反 操作 或者 ~x == -x-1

<< 、>> 用于快速将某个数放大或缩小2的某次方被 缩小的话向下取整

本质上是对二进制数的左移或右移，左移放大、右移缩小。多出的位数会舍去。因此只能对整数使用。注意，若要赋值，还要x=x<<1才可以

print 函数后添加end=‘’ 可以阻止自动换行。例如：

print('%d×%d=%d'%(j,i,i\*j),end=' ')

print函数输出时，用单引号时，输出内容不可有单引号,可以由双引号；用双引号时，输出内容不可有双引号。可以由单引号；若想输出双引号，可以用三引号进行输出，即三个单引号或三个双引号。同时三引号还可以多行输出。

print('''let's go

shame on you

he said: "i’m imortal"''')

另一种方法，利用转义字符\

但若想输出转义字符\本身，则需要对其本身进行转义

print('i\'m imortal') #i'm imortal

print("\"alarm!\" ") #"alarm!"

print('"alarm!"') #"alarm!"

print("\"alarm!\"") #"alarm!"

print("'alarm!'") #'alarm!'

print("\\\*-hahah-\*\\") #\\*-hahah-\*\

输出转义字符本身的另一种方法：在输出前添加r

print(r'\ is the key') # \ is the key

capitalize() 可将字符串第一位转为大写；首位非字母也无妨，不会报错。

upper() 可将字符串所有转大写

lower() 可将字符串所有转小写

注意用法

x='1abcd'

y=x.capitalize() #1abcd

z=x.upper() #1ABCD

w=x.lower() #1abcd

find函数可以查询目标字符串是否存在于对象字符串中。有则返回其索引值，无则返回-1

可以设置寻找范围；若有多个目标字符串，只返回第一个

x='abcdefghijklmnopqrstuvwxyzefg'

y=x.find('efg',4,10) #4

z=x.find('efg') #4

in函数与之类似，但返回的是布尔值。

x='abcdefghijk'

y='efg' in x #True

z='cds' in x #False

index与find类似，但若不含目标字符串会抛出错误。其他与find完全相同

x='abcdefghijklmnopqrstuvwxyzefg'

y=x.index('efg',4,10) #4

z=x.index('eg') #Error

count函数 统计目标字符串在对象字符串中出现的次数。格式与以上函数相同。

x='abcdefghijklmnopqrstuvwxyzefg'

y=x.count('efg',4,10) #1

z=x.count('efg') #2

replace函数 替换字符串，可以约束替换次数。若无可替换对象则误操作，不报错。

x='abcdefghijklmnopqrstuvwxyzefgz'

y=x.replace('efg','2333',5) # abcd2333hijklmnopqrstuvwxyz2333z

split函数 用于分隔字符串，可设定分隔符与分隔次数，默认以空字符完全分隔。结果为list

x='xxxxeyyyyezzzzewwww'

y=x.split('e',2) # ['xxxx', 'yyyy', 'zzzzewwww']

z=x.split('e',-1) # ['xxxx', 'yyyy', 'zzzz', 'wwww']

w=x.split('e') # ['xxxx', 'yyyy', 'zzzz', 'wwww']

.title() 可将字符串所有隔开的英文首字母变为大写，其他小写 不改变原始数据

x='LISA is a pig'

y=x.title() # Lisa Is A Pig

strip、lstrip、rstrip函数用于切除两边、左边、右边制定字符，默认空格。不改变原始数据

x='\*\*\*LISA is a pig\*\*\*'

y=x.strip('\*') # LISA is a pig

z=x.lstrip('\*') # LISA is a pig\*\*\*

w=x.rstrip('\*') # \*\*\*LISA is a pig

字符本身可以进行加法操作，将两者连接在一起。其他操作不可行。或者成为连接操作。

字符串可以用f字符串插入其他类型变量，如数字、字符串、数组、类等

a=Tiger()

b='\*\*\*LISA is a pig\*\*\*'

c={1,2,3,4,5,6,7,8}

d=7.10

e=f"{a}{b}{c}{d}"

print(e)

#输出结果为：<\_\_main\_\_.Tiger object at 0x000002829F387520>\*\*\*LISA is a pig\*\*\*{1, 2, 3, 4, 5, 6, 7, 8}7.1

starswith、endswith用于判断字符串是否以\*\*\*开头或结尾。可设定查询范围，返回布尔值

x='\*\*\*LISA is a pig\*\*\*'

y=x.startswith('\*') #True

z=x.startswith('L',3,100) #True

w=x.endswith('\*',5,100) #True

ljust、rjust、center函数可以将字符串左对齐、右对齐、居中操作，并在不足的位上补所设定的字符。若超过则返回原字符串。

x='LISA is a pig'

y=x.ljust(15,'\*') #LISA is a pig\*\*

z=x.rjust(15,'\*') #\*\*LISA is a pig

w=x.center(15,'\*') #\*LISA is a pig\*

使用extend对两个列表进行合并

a=['sam','alex']

b=['sasm','alucy']

a.extend(b) #['sam', 'alex', 'sasm', 'alucy']

del 根据下标删除；.pop() 删除最后一个或指定的某个并返回被删除的值 ；.remove() 根据元素值删除。

a=['sam','alex','lucy','pegy']

del a[2] #['sam', 'alex', 'pegy']

a.remove('alex') #['sam', 'pegy']

a.pop() #['sam']

列表排序可用sort、reverse语句。也可在其中添加key函数用于高级排序

a=[5,7,-1,9,-6]

a.sort() #[-6, -1, 5, 7, 9]

a.reverse() #[9, 7, 5, -1, -6]

a.sort(key=abs,reverse=1) #[9, 7, -6, 5, -1]

a=sorted(a,key=abs,reverse=1) #[9, 7, -6, 5, -1]

注意，sorted函数会额外生成一个列表，并不改变原列表的排列方式。

但sort、reverse语句会直接改变原列表的排列

注意这三个命令的使用方法

字典的创建方式为：{ key : value, key : value, key : value,}

a={(5,):[100,100,50],"x,y":(5,5),"point":10,99:20}

键可以是任意类型的数据，值也可以是任意类型数据

可以直接对字典的键赋值来修改或新增一个键值对。如对上边的a进行赋值：

a[5]="five"

即可对a增加新的键值对

删除键值对则使用del命令，同时删除键和值两者：

del a[(5,)]

对字典使用 .get ( ) 函数可以返回其值，若不存在则返回None或者指定值

这一命令有两个参数，一是键值，而是如果不存在需要返回的值。第二个值可不写

a={'a1':'123','a2':'456','a3':'678'}

b=a.get('2') #None

c=a.get('111','s') #'s'

d=a.get('a1') #'123'

用len( ) 可以计算字典中键-值对的个数

a={'a1':'123','a2':'456','a3':'678'}

b=len(a) # b==3

删除字典元素使用del 若无指定元素会出错或 .clear( )直接清空整个字典

a={'a1':'123','a2':'456','a3':'678'}

del a['a1'] #{'a2': '456', 'a3': '678'}

del a['a1'] #出错

a.clear() #将这个字典清空

print(a) #{}

获取字典的键： .keys() 获取字典的值： .values() 获取字典的元素： .items()

a={'a1':'123','a2':'456','a3':'678'}

a.keys() #dict\_keys(['a1', 'a2', 'a3'])

a.values() #dict\_values(['123', '456', '678'])

a.items() #dict\_items([('a1', '123'), ('a2', '456'), ('a3', '678')])

遍历字典的元素方法：

for i  in c:             #遍历字典的键

for i  in c.values():    #遍历字典的值

for i  in c.keys():      #遍历字典的键

for i,j in c.items():     #遍历字典的键值对

集合(set)：用花括号定义，特点是其元素不可重复

a={1,2,5,3,4,6,1}       #类型为set，a= {1, 2, 3, 4, 5, 6}

可以在print函数后添加输出的数量：

print('\*'\*5) #\*\*\*\*\*

全局变量：所有函数都能调用的变量。使用 global 在函数内声明，仅对所声明的函数有用。

若非全局变量，函数只能调用，但无法对其进行改变。

局部变量：只能在其声明的函数内调动。

nonlocal 与全局变量类似，但其声明的对象在某一函数中，用于嵌套函数中。

匿名函数也可以有多个对象：

f=lambda x,y:x\*\*y

a=f(3,3) #27

时间模块 引入：

import time

time.time ( ) 为按秒计算的时间偏移量，以1970.1.1 00:00:00 为起始时间

print(time.time()) #1566482843.074206

print(time.strftime('%Y-%m-%d %H:%M:%S',time.localtime()))

#按特定格式输出 2019-08-22 22:16:46

print(time.strftime('%a %b %d %H:%M:%S %Y',time.localtime()))

#按特定格式输出 Thu Aug 22 2019 22:16:46

a='2019-08-22 22:14:11'

b='Thu Aug 22 22:14:11 2019'

print(time.mktime(time.strptime(a,'%Y-%m-%d %H:%M:%S')))

#由某一时间反推时间偏移量 1566483251.0

print(time.mktime(time.strptime(b,'%a %b %d %H:%M:%S %Y')))

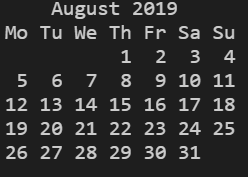
#由某一时间反推时间偏移量 1566483251.0

日历函数：

先载入 calendar模块 利用 calendar.month (year，month) 进行输出

import calendar

print(calendar.month(2019,8))



随机函数：

载入随机数模块 import random

a=random.random() #返回[0,1) 的随机出小数

b=random.randint(a,b) #返回整数，闭区间，必须左小右大

c=random.uniform(a,b) #返回小数，闭区间，左右大小无要求

d=random.randrange(a,b,c) #返回整数，[a,b),以c为步长，只能传入整数

e=random.choice(list) #列表、元组、字符串等中返回随机元素

random.shuffle(list) #对列表进行打乱顺序,其本身并不能作为输出

random.sample(a,b) #在a中取b个元素组合成新的列表，a可为列表或字符串

多个装饰器时，系统自下而上读取，即以下两者等价

@dec1

@dec2

def f():

f=dec1(dec2(f))

对于有参函数的修饰器，需要在修饰器内部函数定义时添加相应函数，但必须保证装饰器中的函数与被装饰函数保持参数对应关系。

或者，可以直接添加两个不定长参数，不必考虑对应关系。

def dec1(f):

print('decorating 1')

def inner(\*a,\*\*b): #inner(a, b)

print('confirming aurthority 1')

f(\*a,\*\*b) #f(a, b)

return inner

@dec1

def f(a,b):

print(max(a,b))

对带有参数的装饰器，在原有装饰器上外包一层即可

打开、读取、写入文件

利用.open(‘文件名.后缀’,’打开模式’) 来打开文件例如：

f=open('text.txt','r') #以只读(read)模式打开 text.txt文件

打开模式有很多种，以只读r(read)，写入(write)，添加a(add)为基本，打开模式后添加b为以二进制打开，添加+是读写模式。其中w会覆盖原始文件，所以慎重使用。添加则会将指针放置在文件末端，写入的内容会在最后。

用.write() 进行写入，与print函数基本相同，但似乎不能用end=’ ’的语句

f.write('surprise,motherfucker!\n') #可以参照print写入%s，%d等

利用 .read( ) 来读取文件， 括号内部为字节数，即可以控制读取文件的大小。

返回变量为一个字符串。read函数从光标开始的位置往后读取。例如：

x=f.read(25) #将文件f的光标所在位置之后的25个字节信息保存在x里

对于内容少的文件，可以用 .readlines( ) 一次性将内容提取到某变量中，该变量会以列表形式保存，每个元素即为原文件的一行数据

.readline（） 函数一次读取一行，以str格式保存数据。括号内可以填入数字来设置读取数量，但这样就无法按行读取，会变成顺序读取。

利用 .tell() 语句来获取当前光标位置，括号内不可方参数。该函数会返回一个int

f=open('text.txt','r')

y=f.tell() #初始位置为0

.seek(偏移量 , 相对位置) 相对位置有三个：0,1,2。0代表从文件头，1代表从当前位置，2代表从文章尾。默认值为0。

但如果相对位置为1或2，必须在文件读取模式处选择二进制读取。否则偏移量只能设置为零，否则会报错。

文件重命名与删除，均需要引入os模块。

注意，python删除的文件直接消失，不会放入回收站。另外，python中的路径是两个\

改名操作对文件夹同样有效。或者前边加上r

import os

os.rename('a.txt','sbb3b.txt') #重命名

os.rename('C:\\Users\\acer\\Desktop\\new\\sbb3b.txt','aaaa.txt')

#不同路径重命名

os.remove('C:\\Users\\acer\\Desktop\\new\\sbbb.txt') #不同路径删除

文件夹创建、删除、与改变。若不注明路径，则视为按当前路径

os.mkdir("C:\\Users\\acer\\Desktop\\%s"%name) #创建文件夹

os.rmdir("C:\\Users\\acer\\Desktop\\%s"%name) #删除文件夹

os.listdir('C:\\Users\\acer\\Desktop') #获取目录列表，以列表形式保存

os.listdir('./') #获取当前目录列表

a=os.getcwd() #获取当前目录，

os.chdir(a) #将当前目录转移为a

os.chdir('../') #将当前目录转移上一层目录

from ast import literal\_eval #载入相关模组

a="{'name': 'mike', 'sex': '男', 'phone': '9998-8882'}"

print(type(a)) # a为字符串类型

a=literal\_eval(a)

print(type(a)) # a为字典类型

读取文件时，若想在整个文件读取完毕后退出，需在readline循环中设置if break语句退出

while 1:

i=f.readline()

if i=='':

break

学生管理系统v2.00与v2.10的教训在原程序的基础上额外添加保存文件操作即可。

‘异常分类’：

NameError #访问未声明变量

ZeroDivisionError #除数为零

SyntaxError #语法错误

IndexError #使用序列中不存在的索引

KeyError #字典中不存在的键

FileNotFoundError #打开或访问不存在的文件

AttributeError #访问未知对象属性

ValueError #函数对象错误

try: #以下为测试代码

a=int(input('被除数: '))

b=int(input('除数: '))

print('商：%d'%(a/b))

except (ZeroDivisionError,ValueError): #可以用tuple写入多个异常类型

print('除数不能为零,或未输入数字')

except ZeroDivisionError:

print('除数不能为零') #Exception本质是一个类

except Exception as r: #不确定的异常类型可以使用Exception，

print(type(r),r) #可以将异常类型保存在某变量中

except: #必须放在有对象的except之后

print('出现错误')

else: #若无异常则执行的语句。可选非必须

pass

finally: #有无异常均会执行的语句，必须放最后。可选非必须

pass

raise语句：

a=IndentationError() #可以不经过a赋值直接引发小括号非必须

raise a('shabi!') #①人为制造异常或②在输出描述信息

try:

a=1/0

except Exception as a:

raise IndexError from a #③利用raise传递原来的异常或④用异常引发异常

assert语句：用于人为设置某些限制：

a=1

assert a==0 #不满足要求，抛出AssertionError

a=1 #assert 只针对其之前的程序有效，之后不起作用

自定义异常，需要定义新的异常类（异常本质为一个类）

with 语句：用于预定义清理操作：

with open("sb.txt",'w') as a: #在执行完缩进块的语句后自动关闭文件

a.write('test，test\n')

模块使用：

import time #引入time模块，其中的函数须time.xxx调用

from time import sleep #从time模块引入sleep函数，后引入的同名函数会覆盖

from time import \* #从time模块引入所有函数 不推荐如此使用

import time as t #引入time模块，其中的函数须t.xxx调用

from random import randint as ran #可直接以ran调用该函数

模块制作：平常制作的python文件均可作为模块使用，文件名即为模块名。

但引入模块时会将所有的代码都运行一遍，因此在模块测试时可用

if \_\_name\_\_=='\_\_main\_\_':

pass

来确保引入该模块时不出现多余的问题

class Rex(Cat,Dog): #可以继承多个对象，冲突的属性与方法会以第一个为准

pass

属性有类属性与实例属性两种：

class Animal(object):

tail=True #类属性，可用类名或实例名访问

def \_\_init\_\_(self):

self.legs=4 #实例属性，只能用实例名访问，会覆盖前者

pip download pygame #先下载

pip install pygame #然后安装

在cmd下直接输入以上命令即可添加下载特定模块

../ 表示当前文件所在的目录的上一级目录   
./ 表示当前文件所在的目录(可以省略)   
/ 表示当前站点的根目录(域名映射的硬盘目录)

class HeroPlane(Plane):

def \_\_init\_\_(self,screen,name): #重新定义init函数

super().\_\_init\_\_(screen,name) #若使用父类init属性则使用super()

self.x=230 #重新定义或添加其他属性

制定一些规则：

1. 程序完成之后，应保证：第一个缩进块之间一个空行；import与主干部分一个空行；def之间一个空行；用于最后计算总行数。
2. 在程序最后添加备注，注明版本更新内容以及存在的问题。
3. 应保留每次更新的文件以及一个用于更改升级的原文件
4. 程序开头应设置：

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

空一行，接下来为导入的模块，先写import，然后在from…import

空一行，程序调试所用到的一些变量，要写注释

空一行，程序所需要用到的常量，

空一行，对pygame等库进行初始化

空一行，程序需要提前设置的一些量，如空列表或需要对调试变量进行操作的量。

空两行，class定义，每个class之间空两行，内部函数之间空一行。

空两行，函数定义，函数内部函数之间空一行。

空两行，主函数

空一行，更新日志。

注释→模块→变量→常量→初始化→设置量

→→类定义→→函数定义→→主函数→日志

类命名：大驼峰法。

函数命名：小驼峰。

常量：全大写。

print 函数默认以换行结尾，若需要改变，可以添加：end=‘\*\*’ \*\*即为结尾的字符

print('sssss',end='!')

在元组、列表、字典前加上\*可以将其拆分开来：

但拆分字典只能得到key

print(\*(3,5))

>>>3,5

print((3,5))

>>>(3,5)

p=list.index(value)

#list为列表的名字

#value为查找的值

#p为value在list的位置

Python可以将较长的数字用下划线隔开，但实际运行时会自动忽略这些下划线

print(100000000) #100000000

print(100\_000\_000) #100000000

print(3.1415926) #3.1415926

print(3.14\_159\_26) #3.1415926

通过这一代码可以得到一首不错的诗

import this

列表解析：可以用一行代码生成指定列表

a=[i\*\*0.5 for i in range(1,101)]

列表赋值不能通过直接赋值的方式，而需要用切片的方式复制

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

b=a                         #错误

b=a[:]                      #正确

空列表、空元组也可以作为判断语句，布尔值为false

类的继承

在\_\_init\_\_方法中添加super.\_\_init\_\_(self, \*\*\* , \*\*\*)

class ElectricCar(Car):

    def \_\_init\_\_(self,make,model,year):

        super().\_\_init\_\_(make,model,year)

可以在子类中覆盖父类的方法，在子类中重新定义即可

with open('A.txt') as file1:

contents1=file1.readlines()

#从头读取到位，放入列表中，一行为一个元素

contents2=file1.read()

#光标所在处读取到末尾，放到列表中，只有一个元素，元素为全体文本内容，末尾会读取#一个换行号

对文件的打开可以使用with open函数，在运行代码块内容后自动关闭文件。

with open('D:\Code\Python\学生信息储存\学生信息储存.txt') as file1:

#可在打开文件夹时直接输入路径打开文件，可以绝对目录，也可以相对目录

with open('D:\Code\Python\学生信息储存.txt',encoding='utf-8') as file1:

    for i in file1:

        print(i)

#可以对open之后的对象进行逐行输出，但注意会自动输出每行末尾的换行号

#可对代开的文件设置编码格式

fileobject.write("窝窝头，一块钱四个，嘿嘿！\n")

#对文件进行写入，与但程序不会自动换行，需要手动添加

import os

os.chdir("D:\Code\Python\chapter\_10") #更换目录

retval = os.getcwd() #得到程序的绝对目录，数据类型为str

os.listdir(rootDir)  #获取rootDir下的子目录

os.path.isdir(x) #判断x是否为目录

a = b\ #反斜杠用于续行

+c\

+d