Python作者荷兰人Guido von Rossum，1991年第一个Python编译器诞生

Linux编译安装Python源码文件：

解压Python压缩包

yum install gcc 安装编译Python需要的GUN C Compiler

yum install gcc-c++ 安装编译Python可能会需要的C++编译环境

yum install zlib\* openssl\* 安装Python的pip需要依赖的包

Windows系统可以创建或修改C:\Users\kaku\pip\pip.ini文件：

[global]

index-url=https://pypi.tuna.tsinghua.edu.cn/simple设置Python包下载源为清华大学

./configure --prefix=/home/kaku/app/python --enable-optimizations 指定预编译目录，开启优化进行预编译Python

make C语言编译命令进行编译Python源码文件

make install 安装编译后的Python

pip3 install ipython 安装Python的工具ipython在Python的交互式窗口可以执行Linux命令

Linux下编译Python脚本文件的命令：python（python3） 文件名.py

#encoding/coding=utf-8 在Python文件第一行指定编码格式，Python2默认编码格式不支持中午包括注释内容

#-\*- encoding/coding:utf-8 -\*- 官方推荐的第一行指定文件编码格式

#注释单行或设置文件

'''多行内容'''或者"""多行内容"""注释多行内容

print("内容%d/s"%变量名) 输出内容变量名的值替换%数值/字符串位置，默认输出换行

print "" 输出，Python2支持

print(变量名,end="")直接输出变量名对应的值不换行，Python3方法

print("内容[%d/s]"%(变量名[,变量名]))变量名) 输出多个变量，前后位置个数对应

变量名=input("显示内容") 将赋值给变量名输入内容，Python3的input方法输入的内容默认为字符串处理，Python2会当做命令处理，Python2的raw\_input()与Python3的input()方法一致，Python3没有raw\_input()

Python用缩进来表示代码块，没有{}

标识符变量名规则：由字母下划线和数字组成且数字不能开头

Python的关键字['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

import keyword

keyword.kwlist 查看Python的关键字

使用反斜杠(\)承接多行语句

Python的变量类型：int (整数)、long（长整型，可以是八进制和十六进制）、float (浮点型)、complex (复数) 、bool (布尔)、String（字符串）、List（列表）、Tuple（元组）、Dictionary（字典）

Python的不可变类型：数值类型，字符串，元组，可变类型：列表，字典

type(变量名) 查看指定变量名的变量类型

id(变量名) 查看变量名对应值的内存空间位置，变量名之间赋值，引用的地址一致

int(变量名) 将指定变量名的类型转换为int型

int(x) 将x转换为一个整数

long(x) 将x转换为一个长整数

float(×) 将x转换到一个浮点数

complex(real [,imag]) 创建一个复数

str(x) 将对象x转换为字符串

repr(x) 将对象x转换为表达式字符串

eval(str) 用来计算在字符串中的有效Python表达式并返回一个对象

tuple(s) 将序列s转换为一个元组

list(s) 将序列s转换为一个列表

chr(×) 将一个整数转换为一个字符

unichr(x) 将一个整数转换为Unicode字符

ord(x) 将一个字符转换为它的整数值

hex(x) 将一个整数转换为一个十六进制字符串

oct(x) 将一个整数转换为一个八进制字符串

len(变量名) 查看变量名对应值的长度

Python运算符:

/ 除数

// 取商

% 取余

\*\* 次方幂

字符串\*数值 数值个字符串

if 条件: 条件""、None、0、[]、{}均为假false，is是引用比较即判定引用是否指向同一个对象，==是值比较即判定对象的值是否相等

语句

[elif 条件>0:

语句]

else:

语句

比较运算符：

== 等于

!= 不等于

<> 不等于（Python2可用，Python3不可用）

<= 小于

>= 大于

and 且

or 或

not 非

in 存在，遍历

not in 不存在

while 条件:

语句

for 变量名 in 集合（range(左边界默认从0开始,右边界,步长) Python2返回指定步长的区间列表不包含右边界但会由于右边界过大产生内存溢出的风险一般用xrange代替，Python3解决风险返回一个列表range标记用到列表指定的数时会生成返回指定的数）:

语句

break/continue 结束整个循环/结束本次循环

else:

语句（for语句的else在执行完成所有循环后执行，有break不执行else）

字符串变量名[下标] 获取字符串指定下标的字符的，下标从左向右从0开始从右向左从-1开始

变量名[开始:结束] 获取变量从开始下标到结束下标（不包括结束下标）位置的元素，下标不写默认为最后

变量名[开始:结束:步长] 获取变量按照步长从开始下标到结束下标位置的元素，步长默认1向右获取

变量名[-1::-1]/[::-1] 获取变量的逆序即倒排序

字符串.find("子字符串") 获取子字符串从左向右在字符串中第一个子字符串第一个字母对应的开始下标，返回-1表示未找到

字符串.rfind("子字符串") 获取子字符串从右向左在字符串中第一个子字符串第一个字母对应的开始下标

字符串.index(子字符串) 获取子字符串从左向右在字符串中第一个子字符串第一个字母对应的开始下标，没有产生异常

字符串.rindex(子字符串) 获取子字符串从右向左在字符串中第一个子字符串第一个字母对应的开始下标，没有产生异常

字符串.count(子字符串) 获取子字符串在字符串中的次数

字符串.replace(子字符串,对应子字符串,数值) 从左向右替换字符串中子字符串为对应子字符串指定数值个个数，数值默认所有

字符串.split(子字符串) 按照子字符串切分字符串，默认按照按照特殊字符进行切割

字符串.capitalize() 将字符串第一个字符转换成大写

字符串.title() 将字符串中所有单词的第一个字符转换成大写

字符串.endswith(子字符串) 查看字符串是否以子字符串为后缀即结尾

字符串.startswith(子字符串) 查看字符串是否以子字符串为前缀即开头

字符串.lower() 将字符串所有字符转换为小写

字符串.upper() 将字符串所有字符转换为大写

字符串.center(数值) 字符串放在指定数值长度的位置即居中

字符串.ljust(数值) 字符串放在指定数值长度的左位置即左对齐

字符串.rjust(数值) 字符串放在指定数值长度的右位置即右对齐

字符串.strip() 删除字符串中的空格

字符串.partition(子字符串) 从左向右将字符串按照第一个子字符串分片成元组

字符串.splitlines() 字符串按照换行符（\n）进行切分

字符串.isalpha() 字符串是否为纯字母

字符串.isdigit() 字符串是否为纯数字

字符串.isalnum() 字符串是否为数字或字母

字符串.isspace() 字符串是否为纯空格

字符串.join(元组) 元组中的元素用字符串进行连接

变量名=[元素,元素] 定义列表

列表.append(元素) 列表末尾追加元素

列表.insert(数值,元素) 列表指定数值下班位置插入元素

列表+列表 将两个列表合并

列表.extend(列表) 列表合并

列表.pop() 列表弹出最后一个元素即返回最后一个元素后删除

列表.remove(元素) 列表删除从左向右第一个元素

列表.sort([reverse=True]) 列表排序，默认从小到大[逆序]

列表.reverse() 列表逆序排序

del 列表[下标] 列表删除指定下标的元素

列表[下标]=元素 元素替换列表指定下标的元素

字典={键:值,键:元素} 定义字典，键只能是Python的不可变类型，字典的键为Hash存储

字典[键]/字典.get(键) 获取指定键的元素，没有返回异常/空（None）

字典[键]=元素 字典添加指定键值，键存在修改对应值

del 字典[键] 字典删除指定键的元素

字典.keys() 获取字典所有的键，Python2返回List列表，Python3返回列表的对象

字典.values() 获取字典所有的值

字典.items() 获取字典的所有键值对，返回元素为元组的列表

变量名[,变量名]=(元素[,元素]) 获取元组的元素赋值给对应变量

元组=(元素[,元素]) 定义元组，元组（包括元素）不支持修改

def 函数名(参数名[,参数名=默认值,\*参数名（不定长参数，没有参数的参数封装成元组）,\*\*参数名（不定长参数，有参数名的参数封装成字典）],匿名函数名): 定义函数

"说明文档即help的内容"

匿名函数名()

return 变量名[,变量名] 默认多个返回值封装成元组返回

(元素,) 一个元素的元组表示

() 空元组

函数名(变量名[,参数名=值],\*参数名,\*\*参数名,lambda 匿名函数参数[,匿名函数参数]:匿名函数体) 函数调用（将\*参数名参数传递给元组不定长参数，\*\*参数名参数传递给字典不定长参数，传递匿名函数）

定义局部变量的位置使用global 变量名 声明全局变量

eval(字符串) 将字符串转换为语句表达式

Python是动态语言，函数功能可以不事先确定即可以传递匿名函数改变函数功能

变量,变量名=变量名,变量 交换两个变量值

Python中的等号赋值与函数参数传递都是引用，但由于不可变类型的变量不能被修改所有在函数参数传递时不可变类型变量的值不会真正修改，可变类型的参数可以被修改

文件流=open("文件路径名","r/w/a[b][+]") 打开文件只读（文件不存在异常）/只写（文件不存在创建存在覆盖）/追加（文件不存在创建）[二进制格式][读写方式打开]

文件流.close() 关闭文件

文件流.read(数值) 文件读取指定数值个字符即字节，读取完成后文件流指针向文件末尾移动指定字符，默认全部读取

文件流.write(字符串) 文件写入字符串

文件流.readline() 读取一行

文件流.readlines() 读取文件所有，返回元素为每一行的列表

文件流.seek(数值,0/1/2) 文件指针移动到开始/当前文件指针/结束位置后指定数值个字符（字节），正数向文件后负数向文件左，Python2支持负数Python不支持

文件流.tell() 获取文件指针位置（字节）

import os 引入操作系统模块

os.rename(文件路径名,新文件路径名) 文件重命名

os.remove(文件路径名) 删除文件

os.mkdir(文件路径名) 创建文件夹

os.rmdir(文件路径名) 删除文件夹

os.getcwd() 获取当前路径

os.chdir(路径名) 改变当前路径

os.listdir(文件路径名) 获取路径下所有文件名

os.\_\_file\_\_ 操作系统模块文件的路径

class 类名(父类名[,父类名]（类默认继承object类）, metaclass=创建类的方法（Python3元类方式）): 定义类，实例属性和方法一个下划线开头为保护类型，两个下划线为私有类型，无下划线开头默认公有类型

\_\_metaclass\_\_=创建类的方法即元类方法 创建类的元类，默认type，Python2元类方式

\_\_slots\_\_=("属性名",["属性名"]) 限制类可以添加的属性

变量名=变量值 类属性，类的所有对象公有属性

@classmethod 装饰器，指定类方法

def 类方法名(cls): 定义类方法，cls表示类自己，类方法的至少有一个参数，第一个参数表示类自己

类名.属性名 调用类属性

cls.属性名 调用类属性

def \_\_new\_\_(cls): 创建类对象方法，C语言的构造方法相当于\_\_new\_\_和\_\_init\_\_两个方法，创建对象时，类先调用new方法后调用init方法

return 父类名.\_\_new\_\_(cls)

def \_\_init\_\_(self): 类对象初始化方法，初始化对象时自动调用

def \_\_str\_\_(self): 输出类的实例化对象时自动调用

def \_\_del\_\_(self): 类的对象无引用时自动调用即GC的垃圾回收调用方法

def \_\_call\_\_(self): 使用类的实例化对象名()时被自动调用

def \_\_getattribute\_\_(self, item): 定义属性拦截器即访问类属性或方法时自动将访问的属性名或方法名作为item参数传递并调用方法，一般用于输出日志即Log，类方法调用时相当于getattribute的返回结果调用即返回结果()执行，属性访问拦截器getattribute方法内部禁止调用类的属性即self.属性名（会导致属性名再次自动调用getattribute方法进入死循环），一般使用object. \_\_getattribute\_\_(self,属性名)代替self.属性名

def 实例方法名(self): 定义实例方法，self表示类的实例自己，实例方法的至少有一个参数，第一个参数表示类自己

self.属性名 类中获取实例属性名的值，Python的实例属性可以不事先确定

实例方法语句

父类名/super().父实例方法(self) 调用子类重写的父实例方法

pass 空语句

@property

def get私有属性名(self):

@私有属性名.setter

def set私有属性名(self):

私有属性名=property(get私有属性名(),set私有属性名()) 设置私有属性的复制调用函数，即类名.私有属性名=值语句会被Python解释执行对象的set私有属性名()方法，property也可使用装饰器即@方式

类名() 创建类对象

类名/对象名.属性名=属性值 类/对象动态添加或绑定属性，Python为动态语言（可以在运行过程中修改已经定义完成的类、对象、方法等），静态语言（编译时已经定义确定的程序不能修改）

@staticmethod/classmethod

def 方法名(/cls):

语句

类名.方法名=方法名 动态添加或绑定类的静态方法/类方法

def 方法名(self):

语句

import types 引入类型模块

实例对象名.方法名=types.MethodType(方法名,实例对象名) 动态添加或绑定实例对象的方法，types.MethodType(方法名,实例对象名)会将实例对象作为self参数传递给方法名

import sys 引入sys模块

sys.argv 接收Python程序文件的参数，参数以列表形式存放（下标从0开始），Python程序文件名为第一参数

sys.path 获取引入模块的默认查找路径（列表），第一个空表示当前路径，可以修改path列表改变引入模块的查找路径

sys.getrefcount(类对象名) 获取系统中类的对象名的引用个数，但由于引用传递，方法的参数类对象也算作一个引用，所以方法返回个数会比正常引用个数多一个

类的内置属性：

\_\_dict\_\_ 类的属性（包含一个字典，由类的数据属性组成）

\_\_doc\_\_ 类的文档字符串

\_\_name\_\_ 类名

\_\_module\_\_ 类定义所在的模块，类的全名是\_\_main\_\_.类名

\_\_bases\_\_ 类的所有父类构成元素（包含了一个由所有父类组成的元组）

类名.\_\_mro\_\_ 查看类的继承关系（元组）实例方法调用从左向右查找实例方法

类名.方法名() 调用类方法名

@staticmethod 装饰器，指定静态方法

def 静态方法名(): 定义静态方法

类名.静态方法名() 调用类的静态方法名

class 单例对象名: 创建单例对象

\_\_instance=None

def \_\_new\_\_(cls):

if cls.\_\_instance==None:

cls.\_\_instance=object.\_\_new\_\_(cls)

else:

return cls.\_\_instance

try: 异常处理

可能出现异常的语句

raise 自定义异常对象（抛出自定义异常，自定义异常类需要继承Exception），无自定义异常对象默认抛出原产生的异常

[except (异常类型[,异常类型]):

异常处理语句]

except Exception（所有异常） as 变量名: 异常信息赋值给指定变量

else:

没有异常才会执行的语句

finally:

无论是否有异常都会执行的语句

import time 引入时间模块

time.sleep(数值) 休眠数值秒

import random 引入随机模块

random.randint(数值,数值) 获取指定数值之间的随机数

sudo pip/pip3 install 模块名（模块对应的文件为模块名.py） Python/Python3安装模块

\_\_all\_\_=["模块内对象名/模块名","模块内对象名/模块名"] 被其他模块引入所有即\*时包含的模块内对象/模块，\_\_all\_\_在模块中为模块内对象在包（包括\_\_init\_\_.py和多个模块的文件夹，Python引入包时会自动执行\_\_init\_\_.py文件）的\_\_init\_\_.py中为包目录下的模块

import 模块名/包名 as 模块别名/包别名 引入模块（先在当前目录查找相应模块或包未找到去Python的默认目录查找），引入的模块整体会被执行

模块名/模块别名.模块内对象名 调用模块内对象

包名/包别名.模块名.模块内对象 调用包内模块的内对象

from 模块名/包名/.（当前目录下） import 模块内对象名/包内模块名/模块名[,模块内对象名,\*（所有，import后引入的模块会覆盖先引入模块的重名对象，一般不用\*）] 调用模块内对象/模块，可以直接使用模块内对象名调用

自定义模块引入后第一次执行Python解释器会对自定义模块进行编译形成字节码文件模块名.Python的解释器语言python-版本数.pyc，默认保存在\_\_pycache\_\_目录下

from imp import \* 引入import模块

reload(模块名/包名) 已经引入的模块或包修改需要重新加载

if \_\_name\_\_=="\_\_main\_\_":（\_\_name\_\_模块名，模块自己执行自己\_\_name\_\_值为\_\_main\_\_，模块被其他模块引入值为模块名）

测试语句，避免其他模块引入可能会将测试程序一起执行

在与包相同同目录下创建setup.py：

from distutils.core import setup 引入安装模块

setup(name="包构建后的名",version="包构建后的版本",description="包描述",author="包作者",py\_modules=['需要构建的包名.需要构建的模块名','包名.模块名']) 设置安装属性

在setup.py同目录下使用命令python/python3 setup.py build进行构建包会产生build目录，build目录下有lib目录，lib目录下有构建的包的目录，构建的包的目录下有相应模块Python文件

python/python3 setup.py sdist 生成构建包的压缩包，产生dist目录和MANIFEST文件，dist目录下有相应构建包的压缩文件

解压构建的压缩文件（包含包信息文件PKG-INFO和相应包模块的文件）

python/python3 setup.py install 安装setup.py对应的构建包到Python模块默认的系统目录下

变量名=[变量 for 变量 in range(左边界,右边界,步长)] 使用列表生成式生成列表

变量名=[(变量,[变量]) for 变量 in range(左边界,右边界,步长) for 变量 in range(左边界,右边界,步长)/if 条件（表达式的总次数为列表元素个数）] 使用列表生成式生成列表

变量名={元素[,元素}} 定义Set集合

set(列表) 列表转换为Set集合

list(集合) 集合转换为列表

import pygame 引入模块，可以开发2D显示

窗口对象名=pygame.display.set\_mode((窗口宽单位像素,窗口高),0,32) 创建窗口

背景对象名=pygame.image.load("文件路径名") 创建加载图片文件背景对象

窗口对象名.blit(背景对象名,(窗口坐标,窗口坐标))

pygame.display.update() 更新窗口显示

from pygame.locals import \* 引入显示模块的事件机制

事件对象名=pygame.event.get() 获取事件

事件对象名.type 获取事件类型（QUIT/KEYDOWN 退出/按键）

事件对象名.key 获取KEYDOWN类型事件的按键（K\_小写字母/K\_LEFT/K\_SPACE 字母键/方向键/空格键）

import copy 引入复制模块

copy.deepcopy(对象) 深复制对象，深复制即完整复制（递归完整复制即对象内部的引用也会完整复制），浅复制即引用复制

copy.copy(对象) 复制对象，完整复制可变类型对象但不递归完整复制对象内部的引用，不可变类型对象复制为浅复制即引用复制

bin(数值) 转换为二进制数

oct(数值) 转换为八进制数

hex(数值) 转换为十六进制数

int(0b/0o/0x) 转换二/八/十六进制数为十进制数

\_\_对象名\_\_ 对象名前后各有两个下划线一般为系统对象

对象名\_ 对象名后一个下划线一般为避免与Python关键字冲突，一般不使用

dir(对象名) 返回对象的属性列表

Python的私有属性其实是名称重整name mangling为\_类名\_\_私有属性名来实现\_\_私有属性名拒绝访问，但使用\_类名\_\_私有属性名可以访问

可以迭代Iterable对象有集合数据类型和generator（包含生成器和带yield的generator）

from collections import Iterable/ Iterator 引入Iterable/ Iterator模块

isinstance(对象名,Iterable/Iterator) 判断对象是否为Iterable/Iterator类的实例

iter(Iterable对象) 可迭代对象转换为迭代器对象

变量名=方法名，函数引用即变量名()可以调用对象的方法名方法，Python中所有变量为对象包括类和方法，名称类似是指向对象的指针

闭包：函数内部定义内部函数，内部函数使用外部函数的变量，内部函数与内部函数用到的外部函数变量总称为闭包

函数内部任何一部分被外部引用，函数会被认为用到函数整体不会被销毁

def 外方法名(方法变量名):

import functools 引入函数工具模块

@functools.wraps(方法变量名) 避免装饰器进行装饰后help信息为内方法名的help信息

def 内方法名(\*args,\*\*kwargs): 定义通用装饰器

return 方法变量名(\*args,\*\*kwargs)

return 内方法名

@外方法名/类名 装饰器，Python解释器解释到@时不用调用自动执行，等价于在下方方法之后执行方法名=外方法名(方法名)方法名重新引用被外方法的返回方法/方法名=类名(方法名)方法名重新引用将方法作为参数创建的类对象

def 方法名():

语句

def 参数方法名(参数):

def 外方法名(方法变量名):

def 内方法名(\*args,\*\*kwargs): 定义通用装饰器

return 方法变量名(\*args,\*\*kwargs)

return 内方法名

return 外方法名

@参数方法名(参数) 参数装饰器，Python解释器会先执行参数方法名(参数) 函数后将结果引用作为@结果进行装饰下方方法

globals() 获取当前所有全部变量

locals() 获取当前所有局部变量

Python使用LEGB规则查找名称对应的的对象，LEGB的规则先后顺序为locals（当前命名空间）、enclosing（外部嵌套函数的命名空间）、globals（模块的命名空间）、builtins（系统默认内嵌命名空间）

dir(\_\_builtin\_\_) 返回系统默认内嵌的对象

生成器名=(生成器表达式) 定义生成器

next(生成器名)/生成器名.\_\_next\_\_() 获取生成器的下一个值，没有值返回异常

def 生成器名(): 定义生成器对象方法，next(生成器名())会生成一个变量名的数值（生成器会停在yield不进行执行，下一次next()方法时生成器才从停止处继续执行，yield表达式整体结果为None）

for 变量名 in 表达式:

变量=yield 变量名

生成器名().send(None/对象) 生成一个变量名的数值，生成器会停在yield不进行执行并将None（第一次send()方法会由于生成器停止位置不在yield处可以通过传递None或使用next()方法避免异常）/对象作为yield表达式整体结果，下一次next()方法时生成器才从停止处继续执行

Python中一切皆对象和Linux中一切皆文件类似

类名=type(类名,父类名的元组,属性的字典)动态创建类，等价于使用class进行定义类，元类可以创建类，元类即type

对象.\_\_class\_\_ 获取对象的类即type(对象)

hasattr(对象,属性) 判断对象是否包含对应的属性

GIL全局解释器锁，Python比C语言性能低的原因在于GIL

小整数对象池：Python为了优化速度将区间[-5,257)的整数对象和单个字符对象已经提前创建好常驻内存不会被GC垃圾回收，即这个区间内的整数的所有对象引用指向同一内存空间单个字符对象的所有引用指向同一内存空间，区间外的大整数在使用时创建

intern机制：不可变类型的字符串中没有特殊字符时默认开启intern机制，所有引用对象指向同一内存空间，引用计数为0时销毁

C语言的内存需要程序自己维护，malloc(字节数)申请字节数的内存，free释放内存

GC（Garbage Collection）垃圾回收：

Ruby使用标记清楚方式回收内存，Python使用以引用计数为主标记清除和隔代回收为辅机制

import gc 引入GC模块

gc.disable() 关闭GC垃圾回收机制，GC垃圾回收机制默认自动开启

gc.get\_count() 获取当前自动执行垃圾回收的计数器

gc.get\_threshold() 获取GC模块中自动执行垃圾回收的频率

gc.set\_threshold(剩余对象个数阈值,零代链表清理次数阈值,一代链表清理次数阈值) 设置自动执行垃圾回收的频率，新创建对象个数减去已经释放对象个数即剩余对象个数超过指定阈值会进行零代链表清理，零代链表清理次数达到阈值后进行一代和零代链表清理，一代链表清理次数达到阈值后进行二代一代和零代链表清理

gc.collect() 手动执行GC垃圾回收机制

gc.garbage 获取上次GC垃圾回收清理的对象列表

map(lambda 变量名[,变量名]:表达式,取值范围[,取值范围]) 根据提供的函数（可以为匿名函数）对指定序列做映射

filter(lambda 变量名[,变量名]:表达式,取值范围[,取值范围]) 根据提供的函数（可以为匿名函数）对指定序列做过滤即表达式过滤出表达式为true的值

reduce(lambda 变量,变量名:表达式,取值范围,值]) 根据提供的函数（可以为匿名函数）对指定序列做聚合即将值作为变量取值范围的第一个值作为变量名传递给表达式执行结果传递给变量，取值范围第二个值作为变量名继续表达式操作依次类推直到结束（没有值默认取值范围第一个值作为变量，第二个值作为变量名进行第一次表达式）

sorted(集合,reverse=1) 排序集合，reverse是否倒序

集合的操作符

& 交集

| 并集

* 差集

^ 对称差集即并集后减去交集

偏函数：将一个函数的部分参数设置默认传递后获得新的函数

import functools 引入函数工具模块

新函数名=functools.partial(函数名,参数) 偏函数将函数名的函数传递参数后获得新函数

Python标准库即模块：builtins（内建函数默认加载）、os（操作系统接口）、sys（Python自身的运行环境）、functools（常用的工具）、json（编码和解码JSON对象）logging（记录日志调试）、multiprocessing（多进程）、threading（多线程）、copy（拷贝）、time（时间）、datetime（日期和时间）、calendar（日历）、hashlib（加密算法）、random（生成随机数）、re（字符串正则匹配）、socket（标准的BSD Sockets API）、shutil（文件和目录管理）、glob（基于文件通配符搜索）

import hashlib 引入Hash模块

Hash对象=hashlib.md5() 创建Hash对象

Hash对象.update("需要加密的字符串") 更新哈希对象以字符串参数

Hash对象.hexdigest() 获取Hash加密后的字符串

python3 -m http.server 端口号 开启当前IP简单的服务器

Python常用扩展库：requests（使用的是urlib3继承urllib2的所有特性）、urllib（基于http的高层库）、scrapy（爬虫）、beautifulsoup4（HTML/XML的解析器）、celery（分布式任务调度模块）、redis（缓存）、Pillow(PIL)（图像处理）、xIsxwriter（仅写exce功能，支持xlsx）、xlwt（仅写exce功能，支持xls，2013或更早版 office）、xlrd（仅读 excle功能）、elasticsearch（全文搜索引擎）、pymysql（数据库连接库）、mongoengine/pymongo（mongodbpython接口）、matplotlib（画图）、numpy/scipy（科学计算）、django/tornado/flask（web框架）、xmltodict（xml转dict）、SimpleHTTPServer（简单地Http Server不使用Web框架）、gevent（基于协程的Python网络库）、fabric（系统管理）、pandas（数据处理库）、scikit-learn（机器学习库）