## 基础

**设置环境变量**

set PYTHONPATH=c:\python27\lib

**外部引入import**

**查看及设置包查找路径**

import sys

print(sys.path)

sys.path.append(new\_path)

**\_main\_与\_name\_**

if \_name\_==”\_main\_”:

#用于在调用脚本时，不运行其中的代码

原理：<https://www.zhihu.com/question/49136398>

**重新加载包**

from importlib import reload

reload(package)

**数据类型**

**整型：**和数学上的表示方法一样

**十六进制：**十六进制表示法加上前缀0x

**浮点数**：例 1.23e9或12.3e8; 1.2e-5

**字符串:**用””或’’括起来

**布尔值：**True或False, 可以使用逻辑运算and, or, not

**布尔值转0，1**：BOOL\*1

**空值：**None 该值和0不是一个概念

**查看变量内存位置：**id(x)

**格式化输出：**print()

Print()语句可以跟上多个字符串，用逗号“,”隔开，就可以连成一串输出, print会依次打印每个字符串，遇到逗号“,”会输出一个空格

Print()也可以直接接算式，并输出结果

**槽式输出** 例"{}{}{}".format("圆周率是",3.1415926,"...")

print('Vous etes la {} personne'.format(6)) → Vous etes la 6 personne

**槽式输出格式化** ‘text {:m.nf}’.format(…)

**%格式化** print(‘text %pattern’ %(content)) #pattern指定content的输出格式

print("Geeks : %2d, Portal : %5.2f" % (1, 05.333)) → Geeks : 1, Portal : 5.33

**f格式化** print(f‘the value is {variable^2}’) #加了f之后可以在括号内直接引用变量

%d表示输出数值，%s表示输出字符串，%f表示输出浮点数

被控制的输出量前要加%

浮点数控制符：%m.nf #m表示输出宽度，n表示小数位数，f表示浮点数

数值输出和字符串输出控制符中都可以加入m来控制输出宽度

注意：控制符是字符类型，所以使用时需要用引号括起来，且可以和其它字符共用引号

连续输出多个值时，在输出的变量后加 ,end=’ ’或者 逗号 可以使输出后不换行

**注释**

C中用的是//，Python里用的是**#**

**#**可以用来在调试时屏蔽某些语句

**变量定义**

变量名必须是大小写英文、数字和下划线（\_）的组合，且不能用数字开头

**动态语言：**变量的类型不用预先定义且可变

**静态语言：**变量的类型需要预先声明且不可变，如C、C++

**多行语句**

Python一般以新行作为语句的结束，但可以用 \ 来实现同一语句转行

a= 1+\

2+\

3

print(“a=”,6)

**查看函数说明**

help(函数名)

相当于R中的?

**字符串定义及转义字符**

字符串中有’，用“”框

字符串中有”，用‘’框

字符串中二者都有，在字符串内的 ’ 或 ” 前加转义字符/, 表示其是一个普通字符

Raw字符串：在字符串前外加r,则里面所有字符不进行转义

多行字符串：在第一行字符串前、最后一行字符串后各加’’’

**转义字符 \**  'What\'s your name?‘

**制表符 \t**

**回车符 \n**

**换行符 \r**

**键盘输入**

**输入字符串:** x=input(“提示字符”)

以这种方式输入数字，需要在后面使用数据类型转换

可以在输入时夹杂分隔符，在输入后使用str.split来实现输入多个内容

**数据类型转换**

**数字串转整数：**x=int(' 123 ')

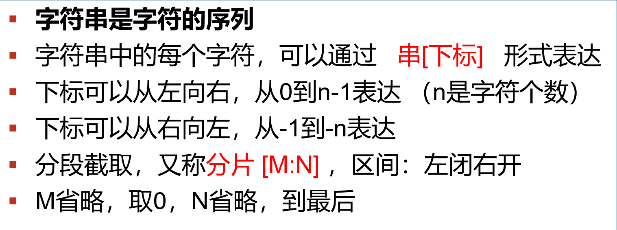
**数字串转浮点：**y=float(' 428.123 ')

**数值转字符串：**s=str(123+3.1)

**字符串表达式转换为数值并赋值为结果**：eval() **ASCII码转换为相应字符:** chr(97) **字符转换为相应ASCII码:** ord('a')

Python里不支持C语言那样ASCII码和字符类型直接加减

**字符串数组**

****

**算符**

**次幂 \*\***

**绝对值abs()**

**除号/**  #除下来是浮点数

**整除号//** #除下来是整数

**+=**

**-=**

**\*=**

**/=**

计算优先级：取余、次幂>乘除>加减>关系比较符

加号可用于连接两个字符串

乘号可用于重复复制一个字符串

**布尔值和逻辑运算的返回值**

**True**

**False**

Python里允许连续的逻辑运算，如1<x<10

Python把0、空字符串''和None看成 False，其他数值和非空字符串都看成 True

1. 在计算 a and b 时，如果 a 是 False，则根据与运算法则，整个结果必定为 False，因此返回 a；如果 a 是 True，则整个计算结果必定取决与 b，因此返回 b。

2. 在计算 a or b 时，如果 a 是 True，则根据或运算法则，整个计算结果必定为 True，因此返回 a；如果 a 是 False，则整个计算结果必定取决于 b，因此返回 b。

所以Python解释器在做布尔运算时，只要能提前确定计算结果，它就不会往后算了，直接返回结果。

**and 与逻辑**

**or 或逻辑**

**not 非逻辑**

**字符比较，以字母顺序，从小到大，大写小于小写，'9'< 'A'<’a’<’汉字’**

**赋值**

赋值允许连续赋值，如

x,y,z=1,2,3

因此，交换值可以直接x,y=y,x

还可以连续赋值：x=y=z=100

**引用库**

**import库名**

**if条件语句**

if 条件： if not 条件

指令1 指令1

指令2 指令2

If条件语句的指令前必须缩进4个空格，凡缩进了4个空格的语句皆视为该语句的指令

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

If 条件：

…

else：

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

If 条件1:

…

elif 条件2：

…

else:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

简写方式

a= value1 if condition else value2

**For循环**

For循环可以定义变量

comprehension syntax: [func(x) for iterable\_lower in iterable\_upper for x in iterable\_lower]

**range函数**

range(左区间，右区间，step)

表示一个区间，范围**左闭右开**

左区间缺省为0，step缺省为1

左区间、有区间可以是变量

**与for语句连用时，有i++的效果**

Range不支持浮点数

**随机数函数库**

**引用随机函数库：import random**

**初始化种子： seed(seed=None):**

**random.random(): 生成 [0,1) 之间的浮点数**

**b\*random.random()+a 生成[a,a+b)之间的浮点数**

**random.uniform(a,b)：生成[a,b]间的浮点数**

**random.randint(a,b)：生成一个[a,b]之间的整数**

**random.randrange(a,b,c)：生成[a,b)之间的值为a+cn的随机数**

**random.shuffle(L)：将list L中的元素打乱**

**random.choice(‘abcdefg’)：从字符’a’、’b’、’c’...中随机选择一个字符**

**random.choice(‘ab’,’cd’,’ef’)：从字符串’ab’、’cd’、’eg’中随机选择一个字符串**

**random.sample(' abcdefg ',5)：从字符’a’、’b’、’c’...中随机选择5个字符**

**程序暂停**

**import os**

**os.system("pause")**

**is语句**

用于判断两个对象是否是引出自同一个对象：

**if a is b** #如果a和b引自同一个对象，为True

**if a is not b** #如果a和b不是引自同一个对象，为True

**比较函数**

**cmp(a,b) #**如果a<b,输出-1；a==b，输出0；a>b,输出1

**定义函数**

**def 函数名(变量):**

函数可以return一个或多个值，遇到return时刻，函数终止

**设置缺省值：def 函数名(变量=缺省值)：**

可变参数的名字加 **\***号，就可以传入0个、1个或多个参数给可变参数

**def 函数名(\*变量)：**

此时，变量是一个tuple

**pass语句**

常用于空函数的填充，用于保证结构的完整性，不含任何操作

例如：

def void( ):

pass

**匿名函数**

**Y=lambda 变量名1,变量名2,…：函数式**

例：Y=lambda x:x+1

等价于

def s(x):

z=x+1

return z

Y=lambda a,b:a+b 等价于 def(a,b): return a+b

函数返回多个值时实际上是返回了一个tuple,但赋值给其它变量时可根据位置对应赋值；如果直接print的话，会print出一个tuple

函数内的局部变量如果和全局变量重名，会在函数体内使用局部变量而忽视全局变量

**在函数体内引用全局变量要在使用前声明：global 全局变量名**

**此时可以直接在函数体内修改全局变量**

引用函数时，赋值可以直接赋值而不考虑顺序

例def f(a,b,c)

引用时可以：F(a=1,c=2,b=3)、F(1,b=3,c=2) #1认为赋给了a

在这种引用方式下，第一个被指明的参数后的所有参数都必须指明

Python的函数可以使用外部函数做变量

**列表**

**定义列表：**

1. 用[ ]将列表内的元素框起来即可
2. L=[x^x for x in range(10) if x%3==0]

L=[9,26,81]

1. **list(range(a,b, step))** #产生[a，b)的列表, 间隔为step

向列表中加入新元素：

1. **列表名.append(新加入的元素)**

**L.append()**如果用来连接两个列表/字符串，会将第二个列表作为一个元素加入到第一个列表中

1. **列表名.insert(元素放置位置，新加入的元素)**
2. **加号**

删除列表中的元素：

**列表名.pop(删除元素的位置)**

**L.pop()**函数如果缺省，默认删除最后一个元素即pop(-1)

**列表名.remove(元素)**

返回元素索引：

**str.find(x,start\_n)**

替换列表中的元素：

**对list中的某一个索引赋值，就可以直接用新的元素替换掉原来的元素**

## 列表处理函数

**LA.extend(LB):** 列表A后附加列表B中的全部元素

extend函数不会申请新的储存空间，而会将原本的两个存储空间合并

列表的加法则会重新申请一个存储空间来存放a+b

**L.reverse()**逆转列表顺序

**L.sort()**缺省时从小到大排序列表元素 括号中若加入**reverse=True**则从大到小

**L2=sorted(L1)**#使用时列表中的元素必须互相具有可比性

PS:汉字字符的码总是比英文字母大

**L=sorted(L1, key=...)** #key是排序的关键词，可以是lambda表达式定义的匿名函数

e.g 按最后一个元素排序 L=sorted(L,key=lambda x:x[-1])

e.g 按两个元素排序 L=sorted(L,key=lambda x: (x[0],x[1]))

e.g 按两个元素排序,一个正序一个逆序 L=sorted(L,key=lambda x: (x[0],-x[1]))

**L.count(a)** 统计L字符串里元素a出现的次数

**L.index(a)** 返回L中第一个等于a的元素的下标

**len(L)**取列表L的长度

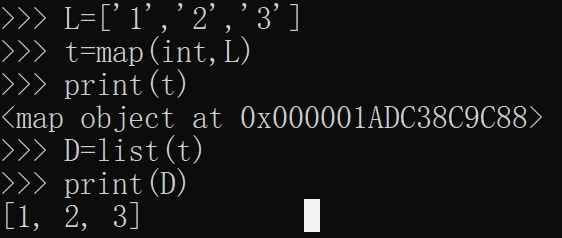
列表中的元素可用 列表名[元素序号] 来取

**sum(L):** 统计字符串L中的元素之和

**L.split(分隔符, 分割次数)**：将字符串中的字符分裂为多个多个字符串

**map(映射数据类型，原列表)：**批量处理列表中的数据，但只是返回一个映射指针，需要用该指针对一个新列表赋值才能以新列表形式输出

**支持自定义的列表处理函数**

例：

**L=zip(L,L’)** 将字符串L和L’中的元素按位置对应整合为元组，**返回**一个元组列表，常用于for循环中有两个循环量的情形

**zip(\*zip(L,L’))** 可进行逆运算，将元组列表还原为两个list

**L=enumerate(L)**将字符串中的元素和其对应的下标整合为元组，**返回**一个元组列表

**max(L)** 取最大元素

**min(L)** 取最小元素

**in**成员运算符

If not (a in L) #判断如果a不在L列表中

**L.[start:end:step]** 列表分片，左闭右开 #注意是方括号

**L=list(range(start,end,step))**生成列表 #注意中间是逗号

**A=L是引用，改A中的元素，L中的跟着变**

**A=L[:] 是拷贝，改A不改L 第一个：后缺省为列表中最大的元素序号**

## 字符串处理

s.isalpha() #判断是否是字母

s=s.replace(‘a’,’b’) #将s中所有的’a’都替换为’b’

s=s.split(‘delimiter’) #切分字符

s.upper() #转大写

s.lower() #转小写

s.strip() #去除头尾空格

s.lstrip() #去除头空格

s.rstrip() #去除尾空格

**L.upper()**大写转换

**L.lower()**小写转换

Aver=sum(数组)/len(数组)

join()函数：通过指定字符串连接列表中的元素，生成型字符串

**指定字符串.join(列表)**

**例：“-”.join((‘G’,’G’,’了’)) 结果：G-G-了**

## 对象和类

**对象**是**类**的个体，对象是确切的单位，类是一群相似单位的集合

isinstance(object, class)

#判断实例是否属于某一个类

类的定义：

**class 类名:**

**def \_\_init\_\_(self, 参数1,…, 参数n):**

# 构造函数，写法为 \_\_+init+\_\_下划线是两个,def后有空格

**确定类的属性，用self.属性名定义**

**def 行为1(self,形参表): #**Class内的函数全部都要加self,定义局部变

**行为逻辑** 量时也要加self.

**def 行为n(self, 形参表): #**在Class外部引用其内部函数时需要格式

**行为逻辑** 类名.函数名

**#**直接输入的参数必须赋给局部变量才能用

#构造函数\_\_init\_\_(self,…)作为类的构造器在创建类时被调用并接收输入参数

#self参数在类方法被调用时，自动指向该类方法所来自的类

#\_\_init\_\_不能返回东西

**继承**

一个子类可以重定义其父类的方法，也可定义新的方法

**class 子类(父类):**

**def \_\_init\_\_(self,****old\_inputs,new\_inputs):**

**super().\_\_init\_\_(old\_inputs)**

**self.new\_inputs=new\_inputs**

**def member\_methods(self):**

**…**

#super()自动指向父类，调用每个父类的属性和方法都要通过super().来进行

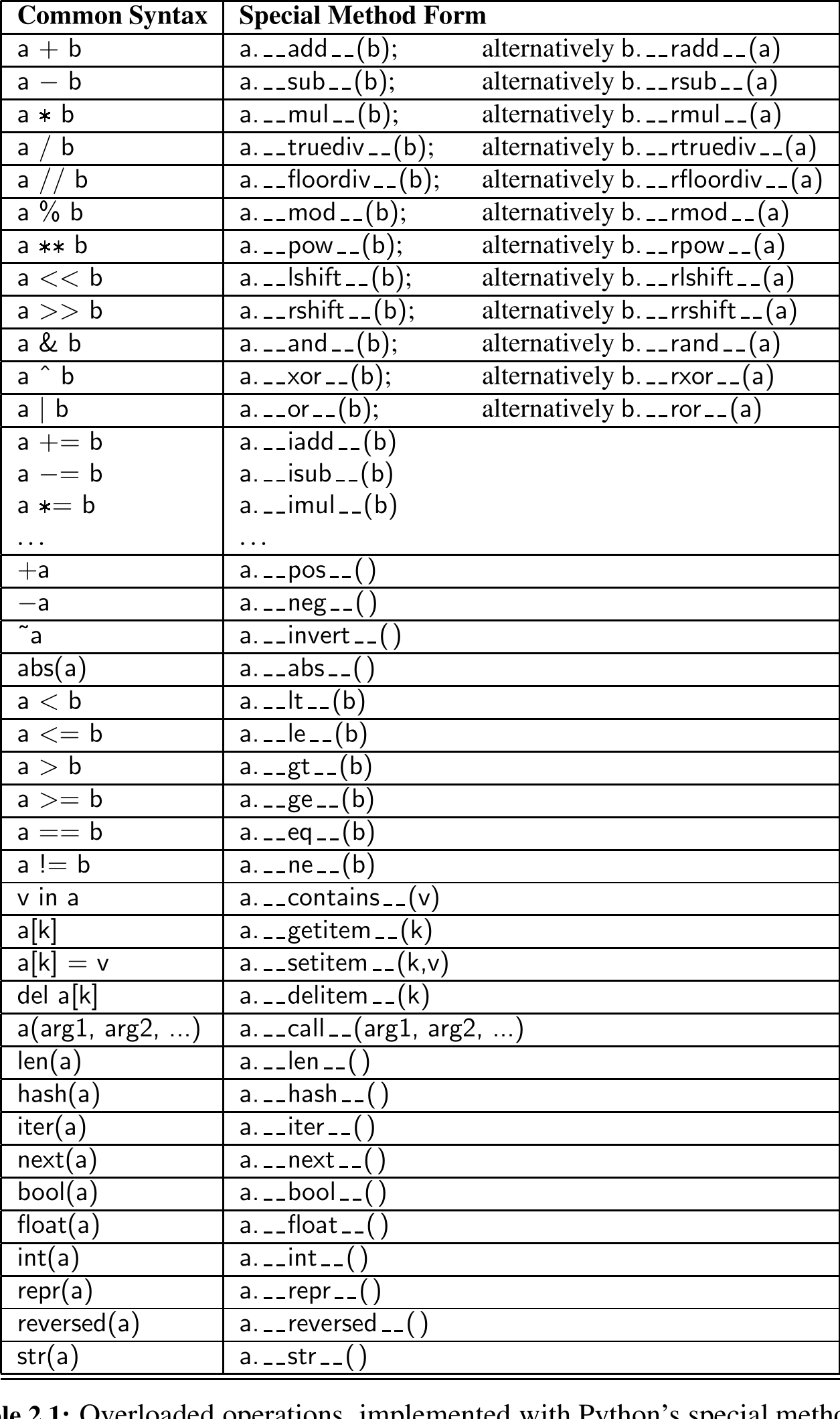
#self依旧指向当前类，即子类

**跨级调用父类私有方法或父类私有属性：**

instance.\_parentclass\_\_method()

instance.\_parentclass\_\_attr

**符号重载**



## 元组

Tuple列表：

**用()将列表内的元素框起来，tuple列表内的内容是不可以更改的**

不过如果tuple里的元素有list,该list内的元素是可修改的

**只包含一个数的元组，必须有一个“，”号 例：x=(45,)**

## Dict (字典)

**定义dict：**

**方法一**

字典名={

Key1:value1, #注意有逗号

Key2:value2,

… #最后一个元素不加逗号

}

**方法二**

dict(list(zip(keys,values)))

**访问：**

1. **print(字典名[‘key’])** #key若不存在，则报错
2. **print(字典名.get(‘key’))** # key若不存在，则输出none
3. **print(字典名.keys())** #输出所有键
4. **print(字典名.values())** #输出所有值
5. **print(字典名)** #输出完整字典
6. **list(dict.items())** #输出键值对构成的元组list
7. **dict.clear()** #清楚字典内内容

**dict查询速度不随字典大小变慢，但内部元素无序排列，因此不能用来存储有序数据**

**key必须是不可变动的数据类型，list因此不能被用作key**

**len(d)** #返回d中的 键/值 对的数量

**d[k]** #返回关联到键k上的值

**d[k] = v #**将值v 关联到键k上

**del d[k] #**删除键k的项

**k in d #**判断k是否是d中的一个键

**通过value反向筛选key**

list(filter(lambda x:d[x]==value,d))

## Set集合

集合中无冗余元素，集合中元素是乱序的

set(可迭代对象)#创建集合

s.add(a) #添加一个元素

s.update([a,b,c]) #添加一些元素

s.remove(a) #删除一个元素（不存在则报错）

s.discard(a) #删除一个元素（不存在不报错）

s.pop() #随机弹出一个元素

s.clear() #清空集合

s=s1.intersection(s2) #求交集

s=s1.union(s2) #求并集

s=s1.difference(s2) #求s1中有而s2中没有的元素

s1.issubset(s2) #判断s1是否是s2子集

s1.issuperset(s2) #判断s1是否是s2子集

s1.isdisjoint(s2) #判断s1是否与s2有交集

s.copy() #生成集合的副本

**在遍历集合元素时，如果需要同时进行增删元素操作，则遍历的只能是集合副本，以使得集合长度在遍历过程中保持不变**

for item in s.copy():

s.discard(a)

## 读写文件

打开txt文件：

**f=open(文件名,mode,encoding=’gbk’)**

#文件名：包含路径的文件字符串，例'C:\\newdata.txt'

技巧：在文件名外前加r（集体转义），文件路径便可以只写一个 \

# mode：打开模式，如'r'读，'w'写

# f则是被操作文件的标识，称为文件句柄，是一个对象

**文件打开方式：**

* **'r' ：读**
* **'w'：文件不存在创建，存在覆盖**
* **'x'：写，文件不存在，创建；存在，报错**
* **'a'：追加写模式，文件不存在，创建**
* **'b'：二进制模式**
* **'t'：文本模式，默认值**
* **'+'：模式组合。配合r/w/a/x模式，增加同时读写**
* **模式间合并写，如'rb'，二进制读**

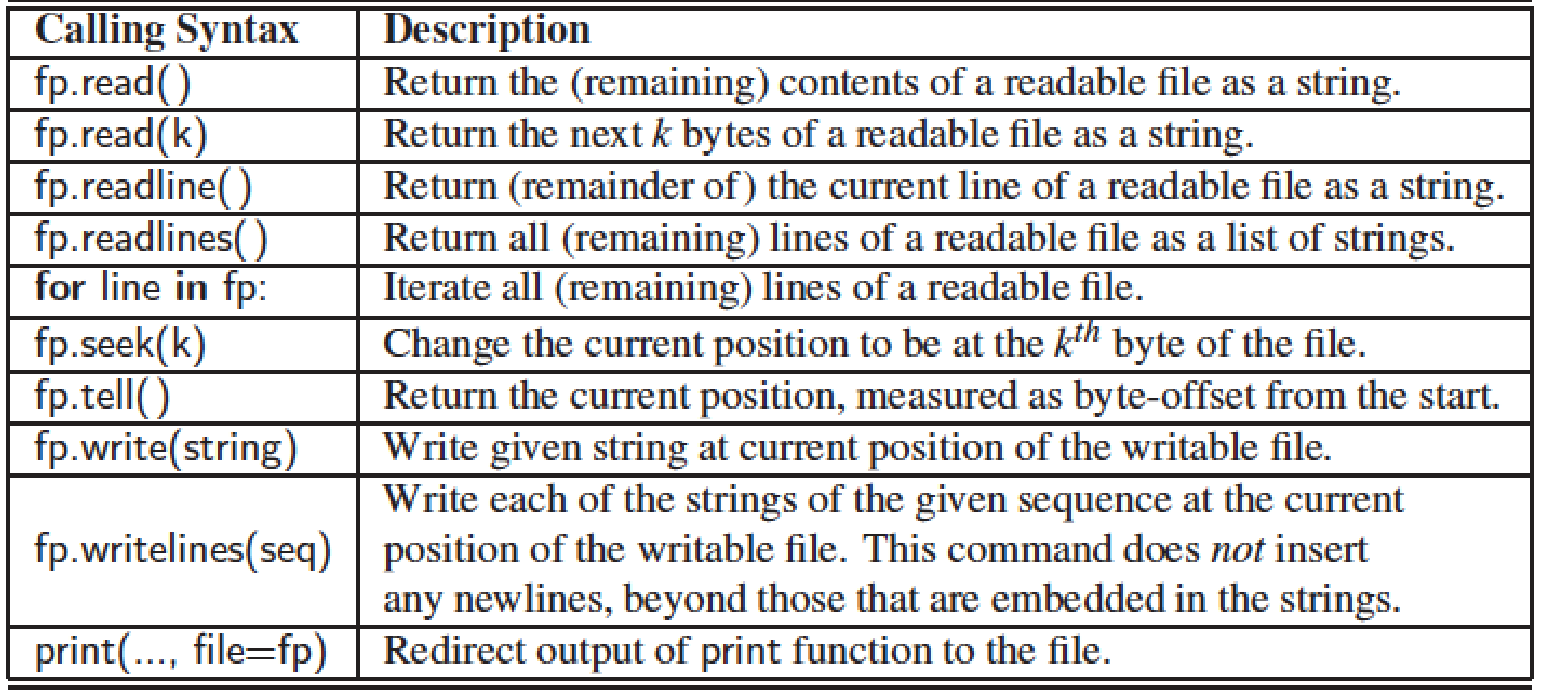
读文件：

**一次读一行：f.readline ()**

**一次性全部读入：**

**lines=f.readlines()** #每行做一个str

**line=f.read()** #全部内容做一个str



**读JSON：**

import json

with open('file.json','r') as f:

data = f.read()

data = json.loads(data)

**写JSON：**

with open(r"file.json",'w') as f:

json.dump(dict,f)

**写文件：**

**f.write(x): 变量x写入文件当前读取位置，x必须是str类型**

**f.writelines(x)：x可以是列表类型，列表的元素是str类型；x也可以是str类型**

关闭文件：

**f.close()**

**每个文件末尾都有一个EOF字符，用以表名文件的末尾**

**读取的数据如果是字符串，需要转换为数值类型：**

1. **split分割**
2. **map转换类型**

**可以将文件地址赋值给一个字符（串）来代指该文件**

## Exec函数

**orders=”””**

**指令**

**”””**

**exec(orders, 字典形式输入参数)**

****

## 迭代器和生成器

[**https://www.runoob.com/python3/python3-iterator-generator.html**](https://www.runoob.com/python3/python3-iterator-generator.html)

**创建迭代器**

iter(iterable)

**创建生成器**

def Generator(x):

循环体

yield output

**输出下一个元素**

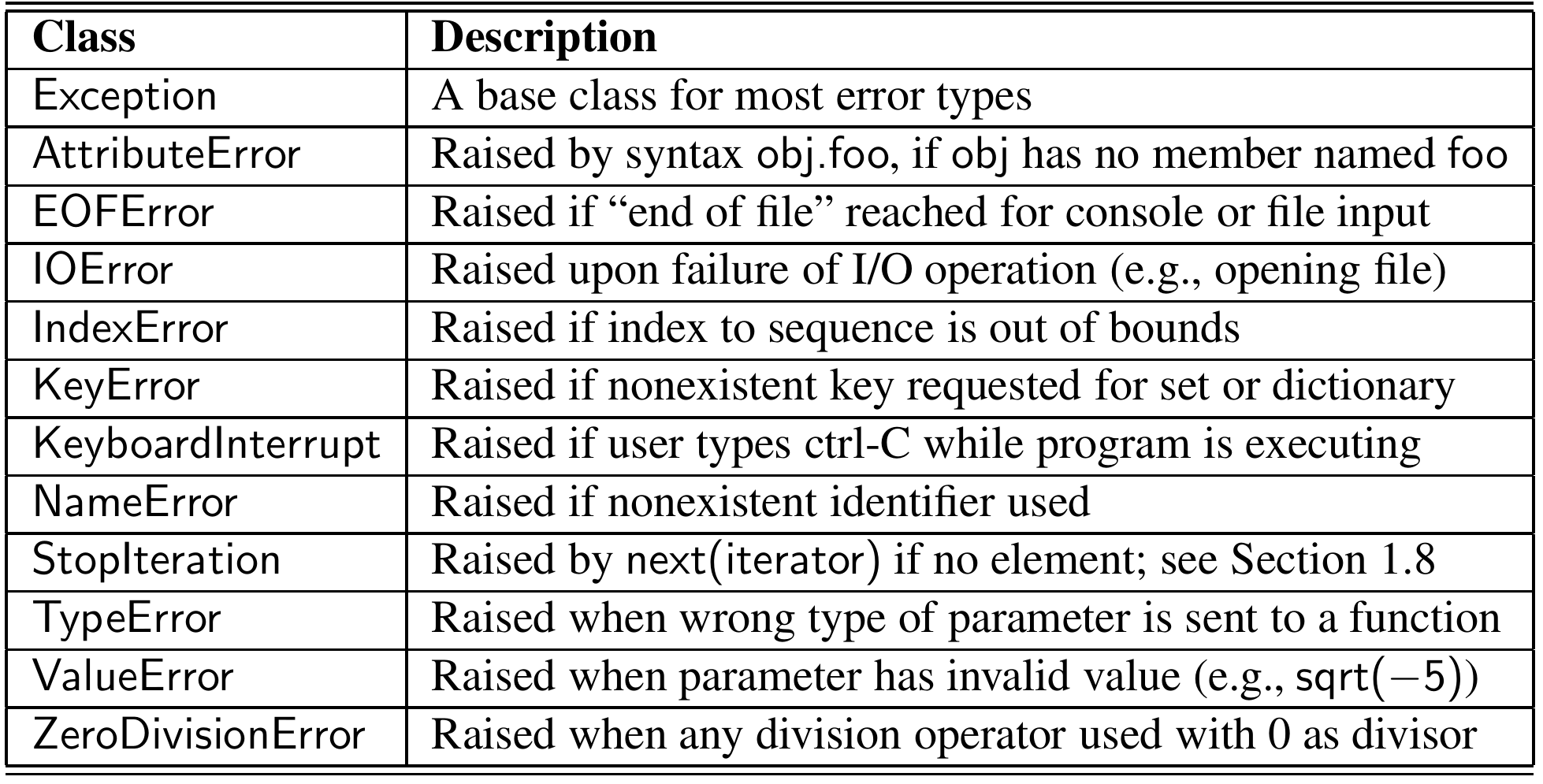
next(Iterator/Generator)

**显示所有元素**

list(Iterator/Generator)

## 异常处理

**内置Exceptions**



**raise errortype(‘text’) #弹出exception并显示说明文字**

**try:**

**…**

**except (error\_1,error\_2,…): #在try执行过程中对一些exception做统一处理**

**…**

**raise #raise在此将涉及的exception弹出到console端**

**except error\_k as e: #对一些exception做特殊处理，并赋给变量**

**print(‘error happens:’,e) #输出exception内容**

**except: #对未提及的exception做统一处理**

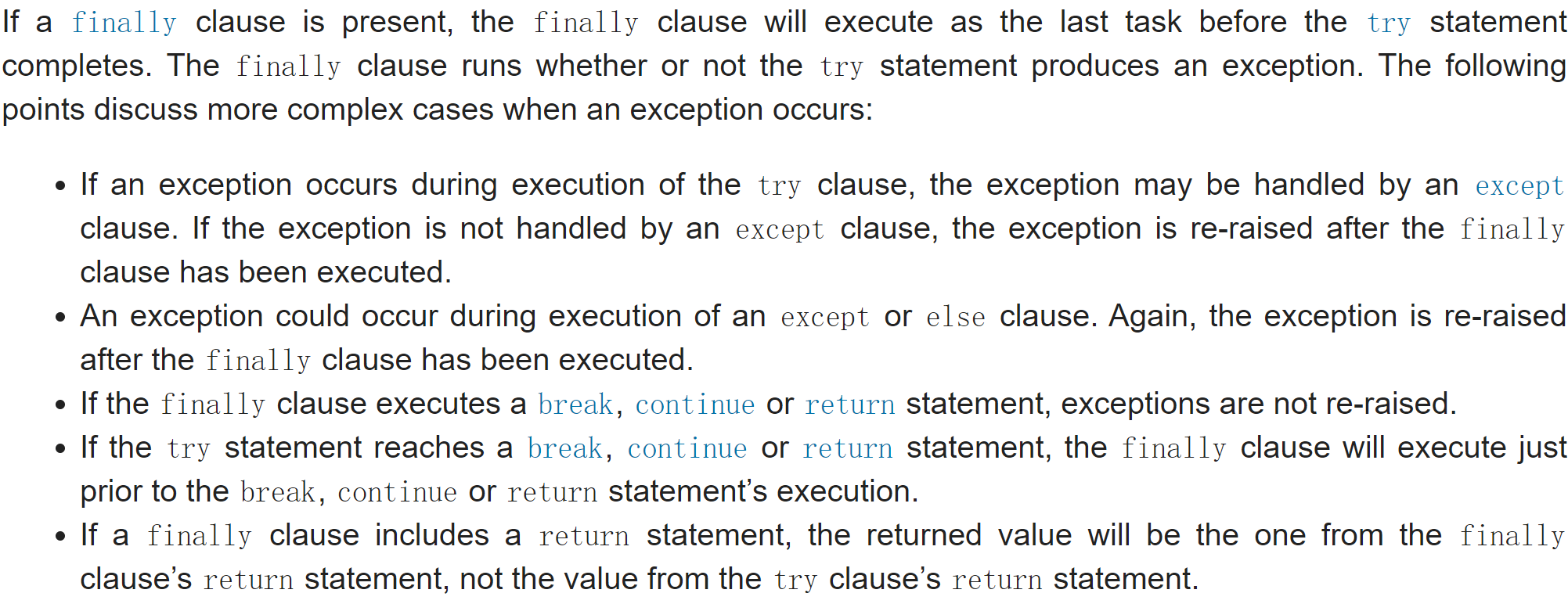
**…**

**else: #使用了except时当未发生任何exception时执行，**

**… 未使用except时，执行所有未提及exception及正常情况**

**finally: #finally中的操作会在try执行完成前最后被执行**

**…**



**连锁Exception**

raise exception from parent\_exception

**自定义Exception**

class Myexception(Exception):

## 函数

**定义**

def function(param1,param2,…):

函数体

return

**设置默认值**

def function(param1=default1,param2=default2,…):

**获取函数信息**

函数名.\_\_code\_\_.co\_argcount #函数输入参数个数

函数名.\_\_code\_\_.co\_varnames #函数输入参数名

函数名.\_\_defaults\_\_ #函数默认值

## 修饰符

**@classmethod**

[**https://blog.csdn.net/weixin\_48580001/article/details/115220956**](https://blog.csdn.net/weixin_48580001/article/details/115220956)

#用于类方法的修饰，被修饰的类方法可以不通过实例来调用

#e.g 若定义了class A及其类方法action(self)，普通情况下需要先创建一个A的实例B，然后B.action()来调用该类方法。但如果类方法action(self)被@classmethod修饰，则可以直接通过A.action()来调用

#被修饰的类方法第一个参数必须是表示自身类的cls，即action(cls)，此cls作用相当于self

**@lru\_cache**

from functools import lru\_cache

@lru\_cache(maxsize=None)

def function(…)

#被修饰的函数在执行时，其输入-输出会被创建为哈希对象存放于内存中，当再次调用此函数并输入了相同的参数时，会直接从内存中取得输出，而不用再次执行函数

## 其他

### Map & Reduce函数

map(func,iterable)

#将iterable中每个元素作为输入，输入到func中得到结果。

#map函数返回的是生成器，即map函数是**惰性函数**

#内置的map的底层实现是单线程运行，而一些库如multiprocess中的map函数底层是多核并行计算

from multiprocessing import Pool

pool = Pool()

pool.map(func,iterable)

pool.close()

from functools import reduce

reduce(func,iterable)

# reduce(f, [x1, x2, x3, x4]) = f(f(f(x1, x2), x3), x4)，可用于实现如阶乘、梯度下降等迭代计算

### 排列组合

**无重复组合**

from itertools import combinations

combinations(list,k)

**有重复组合**

from itertools import combinations\_with\_replacement

combinations\_with\_replacement(list,k)

**排列**

from itertools import permutations

permutations(list,k)

**笛卡尔积**

from itertools import product

product(list1,list2)

#生成一个list，元素为list1和list2中元素两两匹配构成的元组

### 微积分

**from sympy import \***

**积分对象字母 = symbols('积分对象字母')** #消除积分对象字母的字符含义

**integrate(函数式，(积分对象,积分下限,积分上限)** #返回值为积分结果

### 加载图片、GIF

from IPython.display import Image

Image("01.png")

import matplotlib.pyplot as plt

plt.imshow(ndarray,cmap=plt.cm.binary) #显示灰度图

plt.show()

### 对象列表高级排序

**Lambda语句：**

#指用该属性进行排序

**L2=sorted(L1, , reverse=True, key=lambda 对象名:对象名.属性)**

#reverse=True表示从大到小排序

**attrgetter函数：**

**L2=sorted(L1, reverse=True, key=attrgetter('属性1','属性2'))** #多属性排序

### 批量命名变量

globals()[‘变量名’]=赋值 #全局应用

locals()[‘变量名’]=赋值 #适用于函数体内

exec(“self.变量名=赋值”) #适用于类体内

### 执行shell命令

[How to Run Bash Script/Command Using Python? (geekflare.com)](https://geekflare.com/python-run-bash/)

import subprocess

subprocess.run(["ls"])