**Day01**

1、运算符的优先级，用()括起来可以对表达式中进行分组，()内的表达式先进行计算

\*\*

\* / // %

+ -

2、变量(用来绑定数据对象的标识符)

必须是字母或者下划线开头，后面跟字母、数字或者下划线

不能使用python保留的关键字

3、= 赋值运算符

变量 = 表达式/数据对象

变量1 = 变量2 = 表达式/数据对象

变量1,变量2 = 序列类型

说明：当变量不存在时，创建该变量，然后进行数据对象绑定

当变量存在时，改变此对象绑定的数据

一个变量只能绑定一个数据对象

多个变量可以绑定同一个数据对象

4、Python中的名词

关联 绑定 引用

5、序列赋值的语法

变量1,变量2 = 数据1,数据2

变量1,变量2 = 【】

变量1,变量2 =（）

变量1,变量2 = 变量2,变量1

6、is /is not

判断两个变量是否绑定的同一个对象，如果是同一个就返回True，否则返回False

7、id函数

id(变量),得到某一个变量的内存地址

8、del语句

del 变量名

9、内存关联以及引用计数

每个对象都会记录被几个变量绑定，如果引用计数为0，会被销毁

10、复合型赋值运算符

a += b …

**Day02**

1、比较运算符

< ，> ，<= ，>= ，== ，!=

2、数值对象的构造函数

float(obj)：将字符串或者数值,转换为浮点类型对象,如果不给参数则返回0

int(obj,base=10)：将字符串转换整型

complex(r=,i=)：复数类型

bool()：创建一个boolean类型的对象

3、内建的数值型函数

abs(x)：求绝对值

round(num1,num2)：对num1四舍五入，num2为要保留的小数位数，默认为0

pow(x,y,z=None)：相当于x\*\*y,或者 x\*\*y%z

4、input 基本输入函数

从控制台获得用户输入的数据，

input(“”)：返回字符串类型

1. print(,sep=,end=)
2. 逻辑运算符(and、or、not)

7、程序结构分为三种结构(顺序结构、分支结构、循环结构)

分支结构

if 条件1:

执行体

else if 条件2:

执行体

… …

else：

执行体

循环结构

while：

while 条件:

执行体

else：

执行体

注意：严格控制执行条件

for 变量1 in 变量2

pass

**Day03**

1、字符串，在非注释中

‘’ 两个单引号

“” 两个双引号

‘’’’’’ 6个单引号

“””””” 6个双引号

单引号中的双引号 不算字符串的结束符

双引号中的单引号 不算字符串的结束符

2、三引号字符串换行时会自动加上”\n”

三引号中可以包含单引号和双引号

3、字符串中转义字符

**\' \" \\ \n \r \f \t \v**

4、raw字符串（原始字符串）

如果一段字符串包含很多需要转义的字符，对每一个都要进行转义，可以在字符串前面及时 r，表示这是一个原始字符串，该显示什么就显示什么，就是让\无效

5、字符串运算

+ += \*(生产重复从字符串) \*=

“abc”+”abc”

“abc”\*2

6、字符串的比较运算

> < >= <= == !=

**'a'**-97

**"A"**-65

**'0'**-48

7、in /not in操作

用于在字符串、序列、字典、集合是否包含某一个字符串

str in 变量

8、索引操作和切片

序列都可以用索引来访问其元素，正向索引从0开始，反向索引从-1开始，

s = ‘abcde’

切片的语法：

str[(开始位置的索引)：(结束位置的索引)：(步长)]

作用：从字符串中去除一部分对应的元素组成一个新的字符串

说明：从开始的位置开始切，切到结束的位置(不包括结束的位置)，步长是每次切片获取一个元素后，向后移动的单位

反向切片：步长为负的

**Day04**

1、字符串切片：

[开始的索引:结束的索引:步长]

1.1、开始的位置切下包括，结束的位置不包括

1.2、步长是每一次切片之后向后移动的单位

1.3、没有步长，默认为1，:也省略了

1.4、步长为负，反向切片，反向切片时默认从最后一个元素往前切。

【0：10：-2】

2、Python常用的序列函数

len(str) max(str) min(str)

ord(str) 得到字符串的unicode

chr(i) 整数变成字符串

hex(i)整数变成16进制的字符串

oct(i)整数变成8进制的字符串

bin(i)整数变成2进制的字符串

str(obj) 对象转换成字符串

3、字符串的格式化表达式

格式字符串 % 参数

格式字符串 % (参数1，参数2…)

格式字符串中的%为占位符，占的位置由参数替换

格式字符串：

* 左对齐

+ 显示其正负符号

0 补0

数字 长度

数字.数字 总长度和小数点后位数

3.2、格式化字符串的完整表示形式

“%[(keynames)[flags][width][.precision]]typecode“ % (values)

说明：（）的内容可省略

keynames:使用key去values字典中查找value值

flags：- 或者 + 或者 不写或者0或者空格

width:长度

precision：小数点后的位数

width表示整个文本的长度，precision小数点后的长度如果precision<width,正常输出，如果不满足，优先precision

width和precision都支持\*，他们的值由value中的下一项来确定

**Day05**

1、字符串常见其他函数

str.find(“”)：查找是否包括子字符串，返回索引，否则返回-1

str.rfind(“”)：从后往前查找

str.index(“”)：也是查找，只不过找不到会报错

str.rindex(“”)：从后往前查找

str.count(“”)：查找子字符串出现的次数

str.replace(“”,””)：替换，后面的参数替换前面的参数

str.replace(“”,””,1)：指定了替换的次数为1

str.split()：默认按照空格进行分割，返回一个列表

str.split(“”)：按照指定的字符串进行分割

str.capitalize()：第一个字母大写

str.title()：每一个单词的首字母都大写，而其他字母都小写

str.endswith()：判断是否以某字符串结尾

str.startswith()：判断是否以某字符串开头

str.lower()：全部变成小写

str.upper()：全部变成大写

str.center(width)：把字符串居中，一共的长度是width，左右补空格

str.ljust(width)：左对齐，右补空格

str.rjust(width)：右对齐，左补空格

str.strip()：去掉所有空格

str.rtrip()：去掉右边空格

str.ltrip()：去掉左边空格

2、range(start,end,step)函数

从start(默认为0)开始，到end(不包括)结束，每一次步长是step(默认为1)，生成一系列整数组成的可迭代对象（整数序列生成器）。

range(4) 0,1,2,3

range(3,6) 3,4,5

range(1,10,2) 1,3,5,7,9

range(5,0,-2) 5,3,1

range(4,0) 空

3、列表(列表是由一系列特定的元素组成，元素和元素之间没有任何关系，但是他们有先后的顺序关系。列表是一种容器，是序列的一种，列表是可以改变的)

4、列表的创建

创建一个空列表：L = []

创建非空列表：L = [1,2,3,4]，L = [1,2,”#4”,False]

列表的创建函数：list()生成空列表 list(可迭代对象)

5、列表的运算

+：列表拼接 \*：生成重复列表内容 += \*=

> ，< ，>= ，<= ，== ，!= ，列表的元素依次比较

3.4、切片(和字符串类似)

列表[:]，列表[::]

列表的切片返回一个新的列表，原列表内容不变

3.5、切片赋值

针对原列表进行的操作，改变切片区域的序列，切片的同时进行赋值，赋值的内容必须是可迭代对象。

注意：如果切片的步长大于1的时候，等号右边的可迭代对象的元素的个数要相同。

**Day06**

列表：创建、运算、切片、切片赋值

1、针对于列表的一些常用函数

len(L)：序列的长度

max(L)：序列最大值，所有元素必须是数值类型

min(L)：序列最小值，所有元素必须是数值类型

sum(L)：序列和值，所有元素必须是数值类型

1. count(obj)：某一元素在列表中的个数

2、增

切片赋值、列表的拼接

L.append(obj) 添加一个元素到列表尾部，入栈

L.insert(index,obj) 指定插入元素的位置

L.extend (L1) 将一个列表添加到另一个列表中

3、删

L.pop() 删除最后一个元素并返回这个元素出栈

L.remove(obj) 删除元素，只删一次

del L[index] 指定删除元素的索引

4、改

L[index] = obj 通过下标索引修改

切片赋值

5、查

in /not in 返回boolean类型

obj in L /obj not in L

6、列表的非真操作（非真：0，False，空null）

any(L) 真值测试，只要有个是真值，就返回true，否则返回false

all(L) 真值测试，所有元素都是真值，返回true，否则返回false

7、列表的拷贝

复制的时候，如果元素是列表，这个时候这个元素单独占一个空间，复制的时候就是复制的刻个空间，这种现象只复制了一层，是浅拷贝。

浅拷贝 L.copy() copy.copy(L)

深拷贝 copy.deepcopy(L)

8、列表推导式：利用可迭代对象生成列表

L = [变量/表达式 for 变量 in 可迭代对象]

L = [变量/表达式 for 变量 in 可迭代对象 if boolean]

L = [变量/表达式 for 变量1 in 可迭代对象 for 变量2 in 可迭代对象]

**Day07**

1. 元组tuple

和列表类似，不同之处为：

a. 元组的元素不能修改，列表可以

b. 元组使用小括号，列表使用中括号

1. 元组的基本操作

和字符串类型，索引下标从0开始，可以进行切片、组合等操作

算术运算 + += \* \*=

比较运算 > < >= <= != ==

in、not in 也支持

切片 也支持

切片赋值 不支持，因为元组的不可变对象，

可以通过索引去访问元组的元素，但是不可以改变

算术运算时，必须是元组和元组之间的操作

不能删除元组的元素内容，但是可以使用del语句来删除整个元组

3、列表的排序

3.1 sorted(L) 临时排序，对原列表的内容补改变，返回排序之后的元素组成的新列表，默认情况是升序排列，使用关键字传参的方式reverse=true可以指定降序排列

3.2 L.sort() 永久排序，对源列表的元素内容做修改，也可以使用关键字传参的方式reverse=true指定降序排列

3.3 L.reverse() 反转

对于元组是无法直接进行排序，可以将元组转换程列表，再进行排序

4、字典

通过下标来处理列表的元素时，容易出错，工作量比较大，当存储相同的数据时，使用列表或者元组比较合适,但是对于类型不一致的元素时，使用列表或元组就不适合了。

A.字典是一种可变的容器，可以存储任意类型的数据

B.字典中的每一个数据都用键(key)进行索引，而不是像序列那样使用下标

C.字典中的元素没有先后顺序，字典的存储是无序的

D.字典的数据都是以key-value键值对的方式进行的存储映射

E.字典的key值不能重复，且只能是不可变类型作为字典的键

5、创建字典的方式

dict = {"name":"郭德纲","age":55,"gender":True}

dict()

dict(可迭代对象) 可迭代对象的每一个元素，会被创建程一个键值对

dict(\*\*kwargs) 关键字传参

6、字典的增删改查（info是自定义的字典名字）

info = {}

info[“name”] = “张三” 增

info[“name”] = “李四” 改

del info[“name”] 删

info.get(“name”) 查

7、字典的其他操作

len(d)：键值的个数

info.keys()：返回所有的键值组成的列表

info.values()：返回所有的value值组成的列表

info.items()：key和value封装成一个元组，若干个元组封装成一个列表返回

**Day08**

1、Python的每一个对象都可以分为可变/不可变类型，主要的核心类型：数组、字符串、元组都是不可变类型，字典和列表是可变类型。对于不可变类型，如果进行重新赋值，实际上的让变量指向新的对象，原对象并没有改变。

字符串：双引号/单引号/三引号，不可变类型

列表：[]，可以存储不同的类型，可重复，可变类型

元组：()，可以存储不同的类型，可重复，不可变类型

字典：{:,:,:},value可以存储不同的类型，vlaue可重复，可变类型

数据类型相同的时候使用列表，数据类型不同时使用字典，数据要求不可变时使用元组。

2、集合

无序、不可重复，可以不同类型，其内部元素必须是不可变类型。

作用：去重，把一个有重复元素的列表或者字典等数据类型转换位集合，其中的重复元素就会被去重。使用集合的创建函数。进行关系测试，测试两个集合的交集、差集，并集等。

3、常用函数

s.add(obj) 添加元素

s.copy() 复制集合

s.clear() 清空集合

s.pop() 出栈，删除一个元素并返回这个元素，随机的出栈。

s.remove(obj) 删除一个元素，如果没有这个元素就报错

s.discard(obj) 删除一个元素，如果没有这个元素不做任何提示

4、集合的操作

4.1 子集：某一个集合的元素全部包括另一个结合的元素

A<B 或者 A.issubset(B)，返回布尔类型

4.2 差集：A集合和B集合的差集，是只属于A集合，不属于B集合的元素组成的集合，A-B 或者 A.difference(B)

4.3 并集：两个集合的所有元素组成的集合，A|B或者 A.union(B)

4.4 交集：同时属于两个集合的元素组成的集合，A&B 或者A.intersection(B)

4.5 对称差：两个集合的对称差是只属于一个集合而不属于另一个集合的元素组成的集合，A^B 或者 A.symmetric\_difference(B)

5、其他常用函数

all(s),any(s),len(s),max(s),min(s),sum(s)，in/not in

sorted(s),返回排完序的元素组成的列表

6、集合推导式

{变量/表达式 for 变量 in 可迭代对象 if 真值表达式}

numbers = [1,3,5,7,9,3,4,6,5,7,9]

s = {x\*\*2 for x in numbers if x%2==1}

7、固定集合(不可变且无序)

创建固定集合的方式：s = frozenset()

s = frozenset(iter)

可以作为字典的key值来使用，集合的基本操作和普通集合一样。相当于普通集合去掉了修改集合的方法。

**Day09**

字符串、列表、字典、元组、集合

1、什么是函数：可以重复执行的语句块，独立的代码相当于一个整体。

作用：封装语句块，提高重用性；定义用户级别的函数

2、语法：def 函数名(参数列表):

语句块

说明：函数的名字的命名规则和变量相同(小写，单词之间用“\_”)；函数有自己的命名空间，在函数的外部不可以访问函数内部的变量，在函数外部可以访问函数外部的全局变量；函数的参数列表可以为空；语句部分也可以为空，但是要补充pass语句

3、函数的调用

函数的调用是一个表达式；如果函数中没有return语句，此函数调用完毕后返回None

4、return语句

函数中遇到return语句的时候，就不继续往下执行，当需要返回多个数据的时候，可以用列表等数据类型进行返回

5、函数的嵌套：一个函数中调用了另一个函数

6、全局变量和局部变量

当函数内部的局部变量和外部的全局变量重名的时候，在函数内部使用的实际上是局部变量，在函数内部如果像修改那个全局变量，实际上是修改了重名的局部变量。

解决办法：在修改全局变量的位置加一个关键字，global，

对于字典、列表等数据类型，在函数中修改其元素内容，不加global，也是修改的全局变量，因为仅仅修改了其元素内容，如果对于整个字典、列表进行修改，就需要加上global。

7、Python函数调用时，参数的传递规则

A.位置传参

实参(实际参数)和形参的对应关系按照位置依次进行匹配

B.序列传参

在函数的调用过程中，使用\*将序列拆散后再按照位置传参的方式进行参数传递

C.关键字传参

传参的时候，使用形参的关键字进行传递；形参进行匹配时，其顺序可以不受约束，只要保证，每一个形参都被匹配到就可以。

D.字典关键字传参

实参为字典，使用\*\*拆解后，进行关键字传参；字典的key要和形参保持一致；字典的key要是字符串

综合使用：再能确定每一个形参都能被唯一匹配到的情况下，可以任意结合使用

**Day10**

1、函数的缺省参数

语法 def fun(形参1=缺省值, 形参2=缺省值，…):

pass

说明：形参的缺省值必须从右向左依次存在，也就是说如果某一个形参有缺省值，其后面所有的形参都要有缺省值

2、函数的形参的定义规则

A.位置形参

def fun(形参1，形参2…)

Pass

B.星号元组形参

def fun(\*args)

pass

以元组的形式接收参数

C.命名关键字形参

def fun(\*,命名关键字形参):

pass

或者

def fun(\*args, 命名关键字形参):

pass

说明，\*前面的形参是0个或多个，\*号后面的形参是1个或多个

作用：约束了\*后所有的参数都必须用关键字传参或者字典关键字传参进行传值

D.双星号字典形参

def fun(\*\*字典形参)

pass

以字典的形式接收参数，通常字典形参命名为kwargs

**Day11**

1、函数变量

函数可以当作变量，变量名就是函数名。

**def** f1():  
 print(**"f1函数被调用"**)

f1就是绑定了函数的一个变量；绑定了函数的这种变量，可以是相关交换的，也可以是作为调用函数时的参数

2、匿名函数

列表的排序

第一种排序方式，sorter(列表)，返回一个排序后的新列表，原列表不变

第二种方式，列表.sort(),改变了列表的内容，reverse=True是降序排列

当列表中的每一个元素都是字典的时候，是无法进行排序的，因为字典和字典无法进行比较，可以通过sort的形参key，去指定每一个字典之间的比较规则，key接收一个函数。

说明：lambda关键字，仅仅是告诉编译器此处是一个匿名函数

冒号之前的匿名函数的参数，冒号之后是匿名函数的返回值，不需要写return

3、匿名函数的特点：

A.不用起函数名，在业务比较复杂的情况下是比较头疼的问题

B.可以在使用的地方直接定义，如果需要修改，直接修改就可以了

C.语法简单清晰，不需要写def关键字，

匿名函数的意义：

在Python的开发过程中，经常使用某一类函数，仅仅需要使用一次，没有必要单独的定义出来，使用匿名函数就可以了，def函数重点去处理负责的业务。

**Day12**

1、eval()

格式 eval(source,global=None,locals=None)

作用：把一个字符串当作表达式来执行，返回表达式执行的结果。

参数：source：必选参数，可以是字符串，也可以是对象，如果是一个字符串，执行字符串的字面量，使用global和locals提供的命名空间

globals：提供命名空间，存储全局变量，如果被提供，必须是字典类型

locals：提供命名空间，存储局部变量，可以是任意的映射对象，如果被忽略了，eval函数就会使用globals提供的命名空间。

如果globals和locals都被忽略了，eval就会使用当前环境的命名空间

2、exec()

格式：exac(source,globals=None,locals=None)

作用：把一个字符串当作Python程序来执行

参数：

source：必选参数，表示要执行的python程序字符串，或者是一个对象。

globals：同上

locals：同上

3、函数式编程

使用函数解决一系列问题，函数是一等公民，函数本身可以赋值给变量，也可以作为函数的参数，也可以作为函数的返回值。

函数的可重入性：一个函数的参数一定，其返回值也是一定的。

要求：函数中不要访问全局变量

作用：保证了线程安全，多个线程访问同一个函数的时候，其结果是一样的

4、高阶函数

满足以下条件：

A.函数接收一个或多个函数作为参数

B.函数返回一个函数

5、Python中内建的一些高阶函数map(),filter(),sorted()

map(func,\*iterables),将可迭代对象的每一个元素放入函数(第一个参数)对去执行，计算成新的可迭代对象返回，当最短的可迭代对象不再提供元素式，计算结束。

星号元组，可以接收多个参数，以元组的形式获取。

6、filter(func,iterable)

将可迭代对象的元素依次去func执行，这个func会计算，然后返回boolean，如果是false就舍弃，反之保留，最终得到保留的那些元素周组成的可迭代对象返回。

7、sorted(iterable,key=None,reverse=Fasle)

将可迭代对象的数据进行排序，生成排序后的列表，不改变原可迭代对象，iterable表示需要排序的对象，key排序的依据，reverse默认升序

**Day13**

1、递归函数

函数直接或者间接调用本身

注意事项：

A.递归一定要控制层数，当符合某一个条件的时候要终止递归

B.集合所有的递归问题都可以用while来代替

优点：复杂的问题简单化，思路更清晰，代码更简洁

缺点：受环境影响比较大，当递归深度比较大时，可能会得到不可预测的结果

2、闭包

将内嵌函数(函数中定义的函数)的语句和环境打包在一起，得到一个对象称为闭包。

闭包必须满足三个条件：

A.必须有一个内嵌函数

B.内嵌函数必须引用外部函数的局部变量

C.外部函数返回的内容必须是内嵌的引用

3、闭包详解：一般情况下，在我们的认知中，函数调用结束后，函数内部的所有的东西都会被释放，但是闭包是一种特殊情况，如果函数在调用结束后，发现自己有局部变量将来可能在内部函数中使用到，就会把这个局部变量绑定给内部函数，然后自己再结束。

4、闭包中修改外部函数的局部变量：

内嵌函数可以随意访问外部函数的局部变量，但是不能修改(重新赋值也仅仅是定义了一个名字相同的变量而已)，在一般的函数中如果修改全局变量，可以使用global，告诉编译器此处不是重新定义的变量，而是全局变量，或者这个变量是可变类型(list,set,dict),在闭包的内嵌函数中，可以使用nonlocal表示这个变量不是内嵌函数的变量，而是需要往上一层环境去查找。

还有一点需要注意，在闭包的使用过程中，一旦外部函数被调用且返回了内部函数的引用，虽然每次调用内部函数，是开启函数执行，然后消亡，但是闭包只有一份，每次开启的都是同一份闭包。

好处：可以访问函数的局部变量；让这些变量始终保存在内存中，不随外部函数结束而结束

5、进一步解释：

全局变量：优点：可重用， 缺点：易被污染

局部变量：优点：局部访问，不会被污染

缺点：不可重用

闭包：既在全局使用局部变量，又保证不被污染

6、装饰器：函数装饰器是指装饰一个函数，传入一个函数，返回的一个函数，函数装饰器最终作用到一个函数上

装饰器语法：def 装饰器名(参数)

语句块

return 函数对象

函数使用装饰器的语法：@装饰器名

7、函数的文档字符串

在函数中第一个没有被赋值给任何变量的字符串称为文档字符串，可以理解为帮助文档

8、函数的\_\_doc\_\_属性

绑定了函数的文档字符串，函数的\_\_name\_\_属性，返回一个字符串，绑定了函数的名称。

9、一个标准的函数

[函数装饰器1]

[函数装饰器2]

… …

def 函数名(位置形参，\*元组形参，命名关键字形参，\*\*字典形参)

“文档字符串“

语句块

**Day14**

1、什么是模块(Module)

模块是一个包含函数、数据、语句、类等组成的程序组

模块是一个文件，通常以.py结尾

2、模块的作用

让一些相关的数据、函数、类等有逻辑关系的东西组织在一起，让逻辑关系更清晰；模块中的数据、函数、类等可以提供给其他模块使用。

1. 模块的分类

A.内置模块，在解析器内部直接使用

B.标准库模块，安装解释器时自带

C.第三方模块，需要手动安装

D.用户自己的模块

4、模块的使用、导入

语法 import 模块名 [as 别名1][as别名2]

以上语句的作用：将某个模块整体导入到当前模块

dir(模块名) 返回一个字符串列表，模块的所有属性

help(模块名) 查看模块的文档字符串

5、from 模块名 import 模块的属性 [as 别名1][as 别名]

from 模块名 import \*

6、time

time. mktime(tuple),参数一个时间元组，将这个时间元组转换成格里高利，秒

time.localtime(seconds),将一个秒转换成一个时间元组

时间元组：9个数值组成的一个元组

四位数的年份

月(1-12)，日(1-31)，时(0-23)，分(0-59)，秒(0-59)，星期(0-6)，元旦开始日(1-366)，夏令修正时 -1、1 or 0

random,sys,

os

\_\_name\_\_ \_\_file\_\_ \_\_doc\_\_

生成器、异常的处理、文件操作

Python期末大纲

1、什么是匿名函数，以及其意义，使用场景

在Python的开发过程中，经常使用某一类函数，仅仅需要使用一次，没有必要单独定义出来，使用匿名函数就可以了，def函数重点去处理负责的业务。

2、raw字符串什么意义

告诉解释器忽略字符串中的转义字符，即在处理包含大量反斜杠的字符串时，使用原始字符串可以避免手动添加额外的转义字符，使得代码更加清晰和易读。

3、Python核心数据类型

数组、字符串、元组、字典、列表

4、字符串的常见操作

参考Day03、Day04、Day05

5、列表的切片和列表的切片赋值

a.列表的切片：

列表[:]，列表[::]

列表的切片返回一个新的列表，原列表内容不变

b.列表的切片赋值：

针对原列表进行的操作，改变切片区域的序列，切片的同时进行赋值，赋值的内容必须是可迭代对象。

注意：如果切片的步长大于1的时候，等号右边的可迭代对象的元素的个数要相同。

6、列表和元组的区别

a.元组的元素不可以修改，而列表可以

b.元组使用小括号，列表使用中括号

7、字典的存储特性

a.字典是一种可变的容器，可以存储任意类型的数据

b.字典中的每一个数据都用键(key)进行索引，而不是像序列那样使用下标

c.字典中的元素没有先后顺序，字典的存储是无序的

d.字典的数据都是以key-value键值对的方式进行的存储映射

e.字典的key值不能重复，且只能是不可变类型作为字典的键

8、函数的意义

a.结构化和模块化 函数允许将代码分割为小块，每个函数负责特定的任务。这种结构化编程方式使得代码更易于理解、维护和扩展，同时也促进了代码的模块化，可以将函数组合成更大的功能模块。

b.代码重用 函数可以被重复调用，避免了重复编写相同功能的代码。这样做不仅提高了代码的复用性，也减少了代码量，使得程序更加简洁和可维护。

c.提高可读性 通过函数命名和注释，可以更清晰地表达代码的意图和功能。函数提供了一种高层次的抽象，使得代码更易于阅读和理解。

d.参数传递 函数可以接受参数，这样可以将数据传递给函数进行处理。参数允许函数在不同的上下文中执行相同的操作，增加了函数的灵活性和通用性。

e.减少错误和调试 将代码分解为函数可以减少代码的复杂性，每个函数专注于特定的任务，减少了出错的可能性。当出现问题时，函数的模块化结构也使得调试更加容易。

f.支持高级特性 函数也是许多Python高级特性的基础，例如递归、闭包、装饰器等。这些特性增强了函数的灵活性和功能。

9、函数的调用时，参数的传递规则

a.位置传参

实参(实际参数)和形参的对应关系按照位置依次进行匹配

b.序列传参

在函数的调用过程中，使用\*将序列拆散后再按照位置传参的方式进行参数传递

c.关键字传参

传参的时候，使用形参的关键字进行传递；形参进行匹配时，其顺序可以不受约束，只要保证，每一个形参都被匹配到就可以。

d.字典关键字传参

实参为字典，使用\*\*拆解后，进行关键字传参；字典的key要和形参保持一致；字典的key要是字符串

10、函数定义时，形参的定义规则

A.位置形参

def fun(形参1，形参2…)

Pass

B.星号元组形参

def fun(\*args)

pass

以元组的形式接收参数

C.命名关键字形参

def fun(\*,命名关键字形参):

pass

或者

def fun(\*args, 命名关键字形参):

pass

说明，\*前面的形参是0个或多个，\*号后面的形参是1个或多个

作用：约束了\*后所有的参数都必须用关键字传参或者字典关键字传参进行传值

D.双星号字典形参

def fun(\*\*字典形参)

pass

以字典的形式接收参数，通常字典形参命名为kwargs

11、eval()和exec()函数的意义，参数的意义

A.eval(source,global=None,locals=None)

函数的意义：把一个字符串当作表达式来执行，返回表达式执行的结果。

参数的意义：(1)source：必选参数，可以是字符串，也可以是对象，如果是一个字符串，执行字符串的字面量，使用global和locals提供的命名空间，(2)global：提供命名空间，存储全局变量，(3)locals：提供命名空间，存储局部变量。

B.exec(source,global=None,locals=None)

函数的意义：把一个字符串当作Python程序来执行

参数的意义：(1)source：必选参数，表示要执行的Python程序字符串，或者是一个对象，(2)global：同上，(3)locals：同上。

12、什么是闭包，使用的意义

a.将内嵌函数(函数中定义的函数)的语句和环境打包在一起，得到一个对象称为闭包。

b.意义：可以访问函数的局部变量，让这些变量始终保持在内存中，不随外部函数结束而结束。

13、什么是模块

模块是一个包含函数、数据、语句、类等组成的程序组。模块是一个文件，通常以.py结尾。

14、函数的递归计算前N项的和值和阶乘

a.递归计算和值

def sum\_of\_n(n):

if n == 0: # 基本情况：当n为0时，返回0作为和的初始值

return 0

else:

return n + sum\_of\_n(n - 1) # 递归调用自身，计算n + (n-1) + ... + 1

b.递归计算阶乘

def factorial(n):

if n == 0 or n == 1: # 基本情况：0的阶乘和1的阶乘都是1

return 1

else:

return n \* factorial(n - 1) # 递归调用自身，计算n \* (n-1) \* ... \* 1

15、集合的相关操作

集合提供了便捷的方法进行元素的添加、删除和检查，同时也支持对多个集合进行并集、交集和差集等操作，这些功能在数据处理和去除重复元素时非常有用。

16、使用列表推导式定义出符合要求的列表