**Python学习**

笔记书写者：付佳

编写时间：2018-8-15

联系电话：15641689260

[1、python基础 6](#_Toc531870280)

[1.1变量 6](#_Toc531870281)

[1.1.1 变量命名规则 6](#_Toc531870282)

[1.1.1.2 查看关键字的方法： 6](#_Toc531870283)

[1.1.2 变量声明 7](#_Toc531870284)

[1.1.3 变量类型 7](#_Toc531870285)

[1.1.3.1 Number数字类型 7](#_Toc531870286)

[1、整数 7](#_Toc531870287)

[进制转化： 8](#_Toc531870288)

[2、浮点数 9](#_Toc531870289)

[2.1 小数 9](#_Toc531870290)

[2.2 复数complex 9](#_Toc531870291)

[2.3 布尔值 9](#_Toc531870292)

[1.1.3.2 字符串Str 9](#_Toc531870293)

[1、转义字符：（回车换行等） 10](#_Toc531870294)

[2、字符串的格式化 10](#_Toc531870295)

[3、None 11](#_Toc531870296)

[1.1.3.3 表达式 11](#_Toc531870297)

[1.1.3.4 运算符 11](#_Toc531870298)

[1、算数运算符 11](#_Toc531870299)

[2、比较或者关系运算符 12](#_Toc531870300)

[3、赋值运算符 12](#_Toc531870301)

[4、逻辑运算符 12](#_Toc531870302)

[5、成员运算符 13](#_Toc531870303)

[6、身份运算符 13](#_Toc531870304)

[8、运算符优先级 14](#_Toc531870305)

[2、程序结构 14](#_Toc531870306)

[2.1 分支 15](#_Toc531870307)

[2.1.1 分支基本语法（if） 15](#_Toc531870308)

[（1） 单项分支： 15](#_Toc531870309)

[（2） 双向分支： 15](#_Toc531870310)

[（3） 多路分支： 16](#_Toc531870311)

[2.1.2 if语句其他 17](#_Toc531870312)

[2.2 循环（for/while） 17](#_Toc531870313)

[2.2.1 for循环 18](#_Toc531870314)

[2.2.1.1 for循环结构 18](#_Toc531870315)

[2.2.1.2 range介绍 18](#_Toc531870316)

[2.2.1.3 for-else:语句 18](#_Toc531870317)

[2.2.1.4 for循环之break, continue, pass 19](#_Toc531870318)

[（1）break 19](#_Toc531870319)

[（2）continue 19](#_Toc531870320)

[（3）pass 20](#_Toc531870321)

[2.2.2 while循环 20](#_Toc531870322)

[while循环语法1： 20](#_Toc531870323)

[while循环语法2： 20](#_Toc531870324)

[3、函数 21](#_Toc531870325)

[3.1定义函数以及函数的调用 22](#_Toc531870326)

[3.1.1 定义函数 22](#_Toc531870327)

[3.1.2 函数调用 22](#_Toc531870328)

[3.2 函数的参数和返回值 22](#_Toc531870329)

[3.2.1参数的定义和使用 23](#_Toc531870330)

[（1） 例子： 23](#_Toc531870331)

[（2） return例子 23](#_Toc531870332)

[（3）使用函数打印99乘法表 23](#_Toc531870333)

[3.2.2 参数详解 24](#_Toc531870334)

[3.2.2.1 普通参数 24](#_Toc531870335)

[3.2.2.2 默认参数 24](#_Toc531870336)

[3.2.2.3 关键字参数 25](#_Toc531870337)

[3.2.2.4 收集参数 25](#_Toc531870338)

[3.2.3返回值 29](#_Toc531870339)

[3.3函数文档 29](#_Toc531870340)

[3.4函数的作用域 30](#_Toc531870341)

[3.5 常见函数 32](#_Toc531870342)

[3.5.1 globals、locals函数 32](#_Toc531870343)

[3.5.2 eval()函数 32](#_Toc531870344)

[3.5.3 exec()函数 32](#_Toc531870345)

[3.5.4 递归函数 33](#_Toc531870346)

[4、内置数据结构（变量类型） 35](#_Toc531870347)

[4.1 list（列表） 35](#_Toc531870348)

[4.1.1 列表常用操作 36](#_Toc531870349)

[1、访问： 36](#_Toc531870350)

[2、分片操作： 36](#_Toc531870351)

[例1：增长幅度为1 37](#_Toc531870352)

[例2：增长幅度为2，隔一个打印一个 37](#_Toc531870353)

[例3：下标可以超出范围，超出后不在考虑多余下表内容 37](#_Toc531870354)

[例4：下标值，增长幅度可以为负数 37](#_Toc531870355)

[3、列表方法： 39](#_Toc531870356)

[3.1 del方法 39](#_Toc531870357)

[3.2 列表相加 40](#_Toc531870358)

[3.3使用乘号操作列表 40](#_Toc531870359)

[3.4 成员资格运算 40](#_Toc531870360)

[3.5 列表的遍历 40](#_Toc531870361)

[3.5.1 for循环遍历 40](#_Toc531870362)

[3.5.2 while循环遍历 41](#_Toc531870363)

[3.5.3 双层列表循环 41](#_Toc531870364)

[3.6列表内涵：list content 42](#_Toc531870365)

[3.6.1 for创建 42](#_Toc531870366)

[4、列表的常见函数： 43](#_Toc531870367)

[4.1 len()：列表的长度 43](#_Toc531870368)

[4.2 max():求列表最大值 43](#_Toc531870369)

[min():求列表最小值 43](#_Toc531870370)

[4.3 list() :将其他格式的数据转换为list 44](#_Toc531870371)

[4.4 append()插入一个内容 44](#_Toc531870372)

[4.5 insert():指定位置插入 44](#_Toc531870373)

[4.6 删除 44](#_Toc531870374)

[（1） del: 删除指定下标元素 44](#_Toc531870375)

[（2） pop: 从队尾拿出一个元素，即把最后一个元素取出来，有返回值 44](#_Toc531870376)

[（3） remove: 在列表中删除指定的值的元素 45](#_Toc531870377)

[4.7 clear():清空 45](#_Toc531870378)

[4.8 reverse: 翻转列表内容，原地翻转 46](#_Toc531870379)

[4.9 extend：扩展列表 46](#_Toc531870380)

[4.10 count：查找列表中指定值或元素的个数 46](#_Toc531870381)

[4.11 copy：浅拷贝 47](#_Toc531870382)

[5、传值和传地址的区别 48](#_Toc531870383)

[4.2 set 48](#_Toc531870384)

[4.2.1 集合的定义 48](#_Toc531870385)

[4.2.2 集合的特征 49](#_Toc531870386)

[4.2.3 集合序列操作 49](#_Toc531870387)

[4.2.4 集合遍历操作 49](#_Toc531870388)

[1、单个输出 49](#_Toc531870389)

[2、整体输出 50](#_Toc531870390)

[4.2.5 集合的内涵 50](#_Toc531870391)

[4.2.6 集合函数、关于集合的函数 51](#_Toc531870392)

[1、 len, max, min：跟其他基本函数一致 51](#_Toc531870393)

[2、# set：生成一个集合 51](#_Toc531870394)

[3、# add: 向集合内添加元素 51](#_Toc531870395)

[4、# clear: 清空 51](#_Toc531870396)

[5、# copy拷贝 52](#_Toc531870397)

[# remove: 52](#_Toc531870398)

[# discard： 52](#_Toc531870399)

[6、# pop：随机移除一个元素 52](#_Toc531870400)

[4.2.7 集合函数 52](#_Toc531870401)

[4.2.8 集合的数学操作 53](#_Toc531870402)

[4.2.9 frozen set :冰冻集合 53](#_Toc531870403)

[4.3 dict 54](#_Toc531870404)

[1、字典的创建 54](#_Toc531870405)

[2、字典的特征 54](#_Toc531870406)

[3、字典常见操作 55](#_Toc531870407)

[3.1 访问数据 55](#_Toc531870408)

[3.2 成员检测 55](#_Toc531870409)

[3.3 遍历 55](#_Toc531870410)

[4、字典生成式 56](#_Toc531870411)

[5、字典相关函数 57](#_Toc531870412)

[5.1 通用函数 len, max, min, dict 57](#_Toc531870413)

[5.2 clear:清空字典 57](#_Toc531870414)

[5.3 keys:返回字典的键组成的一个结构 57](#_Toc531870415)

[5.4 values:同理，一个可以迭代的结构 57](#_Toc531870416)

[5.5 get: 根据指定键返回相应的值，好处是，可以设置默认值 58](#_Toc531870417)

[5.5 fromkeys：使用指定的序列作为键，使用一个值作为字典的所有的键的值 58](#_Toc531870418)

[4.4 tuple 58](#_Toc531870419)

[4.4.1 元祖创建 58](#_Toc531870420)

[（1） 创建空元祖 58](#_Toc531870421)

[（2） 创建只有一个元素的元祖 58](#_Toc531870422)

[（3） 创建多个值的元祖 59](#_Toc531870423)

[4.4.2 元祖的特性 59](#_Toc531870424)

[4.4.2.1 索引操作 59](#_Toc531870425)

[4.4.2.2 超标操作 60](#_Toc531870426)

[4.4.2.3 切片操作 60](#_Toc531870427)

[4.4.2.4 序列相加 60](#_Toc531870428)

[4.4.2.5 元祖相乘 61](#_Toc531870429)

[4.4.2.6 成员检测 61](#_Toc531870430)

[4.4.3 成员遍历（一般使用for） 61](#_Toc531870431)

[（1） 单层元祖遍历 61](#_Toc531870432)

[（2） 双层元祖的遍历 61](#_Toc531870433)

[4.4.3 关于元祖的函数 62](#_Toc531870434)

[4.4.4 元祖的函数 63](#_Toc531870435)

[4.4.5 元祖变量交换法 63](#_Toc531870436)

Ctrl+Enter 进入编辑模式，编辑之后 alt+enter 命令执行

安装文件：http://www.cnblogs.com/zhouzhishuai/p/8401103.html

# 1、python基础

打印字符串helloworld：print("helloworld")

打印数字类型：print(1234)

虚拟机下打开控制台：ctrl +alt +t

运行Python：ctrl+enter

python注释：# 给人看的东西 或者 ''' 注释 '''

## 1.1变量

程序=数据结构+算法

### 1.1.1 变量命名规则

（1） 变量命名不可以以数字开头，如4four,3man;

（2） 不推荐使用以下划线开头，下划线开头的内容在python中有特殊意义，如\_age,\_name

（3） 推荐使用固定单词及其缩写，如skt=soket

（4） 以posix命名规则为主，posix命名规则单词全部小写，且所有单词之间以下划线连接，如my\_first\_love；

（5） 驼峰命名法：所有单词自动连接，且每个单词首字母均大写，分为大驼峰命名法和小驼峰命名法：

大驼峰命名法：所有单词首字母均大写，如MyFirstLove 用于给类命名

小驼峰命名法：第一个单词首字母小写，其余均大写，如myFirstLove；

（6） 避开关键字和保留字，用于给普通变量或函数命名

##### 1.1.1.2 查看关键字的方法：

import keyword

print(keyword.kwlist)

['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']

### 1.1.2 变量声明

var\_name(变量名) = var\_value(变量值) 一个等号相当于赋值

var1 = var2 = var3 =var\_value(从右往左读)

var1 ,var2 ,var3 = v1,v2,v3

例如：age = 18

print(age)

print(18)

结果 18 18

### 1.1.3 变量类型

严格意义来讲，Python只有一个类型

标准数据一共有六种：

（1） 数字类型 Number

（2） 字符串类型Str

（3） 列表List

（4） 元组tuple

（5） 字典Dictionary

（6） 集合Set

#### 1.1.3.1 Number数字类型

数字类型没有大小限制（没有最大188888888888888888888888这种类似java的bigInteger类型，多大都可以）

###### 1、整数

没有小数部分，表示个数的数字

自然数 0 ﹣自然数

整数是分进制

二进制

计算机常用

是计算机唯一能理解的数字

表示为以0b开头的0，1代码

例如：0b1101,0b1001

八进制

不常用

以0o开头的包含0-7的数字（逢8进一）

十六进制

逢16进一

其实是每四位二进制表示一位十六进制

包含0-9 a-f

数字上以0x开头表示

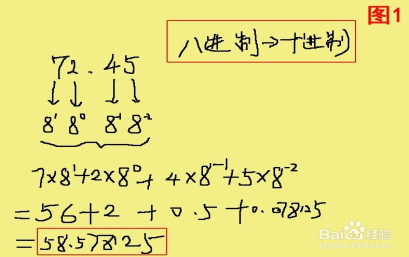
十进制

18， 89 ，178

进制转化：

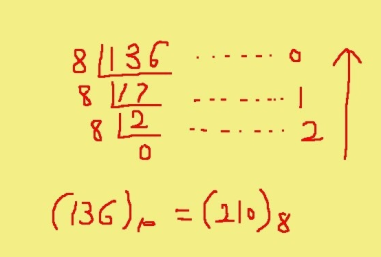
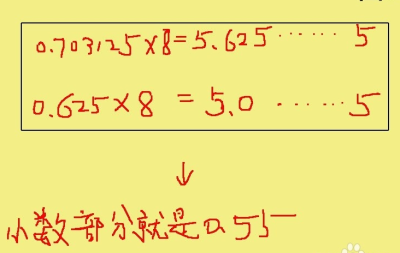
1、八进制转10进制

72.45 =》 7\*81+2\*80+4\*8-1+5\*8-2



2、10进制转八进制

整数部分，除8取余法，每次将整数部分除以8，余数为该位权上的数，商继续除以8

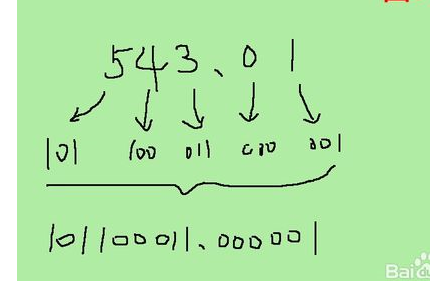
3、二进制转八进制

二进制转换成八进制的方法是，取三合一法，即从二进制的小数点为分界点，向左（或向右）每三位取成一位，如图2所示。



4、八进制转二进制

下面看看将八进制转为二进制，反过来啦，方法就是一分三，即一个八进制数分成三个二进制数，用三位二进制按权相加，最后得到二进制，小数点依旧就可以啦。如图5所示。



5、十进制转16进制：除以16

###### 2、浮点数

2.1 小数

计算机里面表示浮点数可以有条件省略

3.1415

3. (3.0)

.5 (0.5)

科学计数法

定义跟数学定义相同

写法是用e/E后面跟整数表示10的指数

176.34 == 1.7634e2

0.876 == 8.76e-1

2.2 复数complex

定义同数字一致

一个由实部和虚部构成的数字

虚部用J/j标书

5+3j

3j

(3j)

2.3 布尔值

表示真假的值(True/False)大写

布尔值可以直接跟数字直接操作

age = 18 + True => 19

age = 18 + false => 17

#### 1.1.3.2 字符串Str

用来表示一串文字信息

如何表示

单引号

双引号

三引号（三个单引号或者三个双引号）

能够扩起多行信息

例如：

love = " i love wangxiaojing "

love = ' 我爱你 '

单双引号可以交错使用：

love = ' i love "wangxiaojing" and "xiaoming" '

love = """ hahhha1

hahahah2

hahahah3

"""

##### 1、转义字符：（回车换行等）

用一个特殊字符将表示出不方便的内容表达出来

借助反斜杠字符，一旦字符串中出现反斜杠字符，就说明已经转义

例如：如果想表示出let's go,两种方法

（1）引号嵌套 " let's go "

（2）转义字符 s=' let\'s go '

a、" ' " 单引号 用 " \' " 表示

b、" \ " 反斜杠 用 " \\" 表示

c、回车换行符 ：不同系统对回车换行符有不同的定义

Windows：\n

Linus: \r\n

##### 2、字符串的格式化

把字符串按照一定的格式进行打印或者填充

格式化有两种方法：

（1）利用百分号%

在字符串中，利用%表示一个特殊的含义，表示对字符进行格式化

%d：此处应该放入一个整数

%s：此处应该放入一个字符串

eg1: s=" I love %s"%"wangxiaojing" 或者 print(" i love %s"%"wangxiaojing" ) 或者 print( s%"wangxiaojing")

eg2: s="i am %d yesrs old" print( s%18 )

eg3: s="i am %s , i am %d yesrs old"

print( s%("tuling",18) )

有几个占位符，则由几个实际内容代替,或者一个也没有，多个内容替换，用括号括起来

（2）利用format函数完成

直接用format函数进行格式化

在使用上，以{}和：代替%号，后面用format带参数完成

eg1：单参数：s=" i love {} ".format( "wangxiaojing" )

eg2:多参数：s=" i am {1} years old , i love {0} and i am {1} years old. ".format("fujia",18)

例如： s="xxx 您好，我是图灵学院的，邀请您学习。"

##### 3、None

表示什么也没有

如果函数没有返回值，可以返回None

用来占位

用来解除变量的绑定

#### 1.1.3.3 表达式

由一个或者几个数字或者变量和运算符组合成的一行代码

通常会返回一个结果

#### 1.1.3.4 运算符

由一个以上的值经过变化得到新值的过程就叫运算

用于运算的符号叫运算符

运算符分类：

（1） 算数运算符

（2） 比较或者关系运算符

（3） 赋值运算符

（4） 位运算符

（5） 成员运算符

（6） 身份运算符

（7） 逻辑运算符

##### 1、算数运算符

进行算数运算的符号：

（1） + - 和正常的运算一样

（2） 乘号用\*号代替

（3） 除号用/代替 ：除号在python2（取整数部分9/4=2）和python3（9/4=2.25）是不一样的，此处以3系列为准

（4） %取余运算

（5） 取整也叫地板除//

（6） 幂运算 \*\* 9\*\*4=6561

注意：Python没有自增和自减运算符

##### 2、比较或者关系运算符

对两个变量或者值进行比较的运算符

比较的结果是布尔值，即True/False

（1） == ， 等于号

a=199

b = a==80

print(b)

结果：False

执行顺序：1、a==80

2、把结果放到b中

（2） ！= 不等于：print(9!=8)

（3） >大于

（4） <小于

（5） >= 、<=

##### 3、赋值运算符

（1） = 等于号 ：a=0 c=a=4

（2） += 是缩写 a=0; a+=7; a=a+7的缩写

（3） -=

所有数学运算符都有缩写格式

-= ， x= , /= ,//= , \*\*= ,%=

##### 4、逻辑运算符

对布尔值进行计算的符号

（1） and 逻辑与

（2） or 逻辑或

（3） not 逻辑非

Python中逻辑运算没有异或运算

运算规则：

and看做乘法 or看做加法

True看做1 False看做0

则逻辑运算就能转化成数学运算

最后结果如果是0则为False,否则为True

eg：a=True

b=False

c=True

d = a and b or c 1\*0+1 True

逻辑运算短路问题

逻辑运算式，按照运算顺序计算，一旦能够确定整个式子未来的值，则不再进行计算，直接返回

eg1：a= True or @@@@@@@@@

运行到or则不再继续往下运算

下面运算表达式，如果xxx中包含赋值表达式，则结果很难预测

eg2：b=0

a= a or (b=9) and 6

假定不报语法错误，则b最终的值是0而不是9

##### 5、成员运算符

用来检测某一个变量是否是另外一个变量的成员

（1） in

（2） not in

eg：l = {1,2,3,4}

a = 7

b = a in l

##### 6、身份运算符

（1） is：用来检测两个变量是不是同一个变量

语法就是 var1 is var2

（2） is not：两个变量不是同一个变量

eg1： a = 9; a="i love wang"

b = 9; b="i love wnag"

print(a is b) print(a is b)

True False

**7、位运算符**

按位运算符是把数字看做二进制来计算的

a=60 b=13二进制格式如下：

a = 0011 1100

b = 0000 1101

a&b =00001100

&：



##### 8、运算符优先级

永远记住，括号具有最高优先级

优先级表格：

|  |  |
| --- | --- |
| \*\* | 指数 （最高优先级） |
| ~ + - | 按位翻转，一元加号和减号（最后两个的方法名为 +@ 和 -@ ） |
| \* / % // | 乘除取模和取整数 |
| + - | 加减法 |
| >> << | 右移，左移运算符 |
| & | 位‘AND’ |
| ^ | | 位运算符 |
| <= < > >= | 比较运算符 |
| <> == != | 等于运算符 |
| = %= /= //= -= += \*= \*\*= | 赋值运算符 |
| is is not | 身份运算符 |
| in not in | 成员运算符 |
| not and or | 逻辑运算符 |

# 2、程序结构

三大结构：顺序、分支、循环

## 2.1 分支

### 2.1.1 分支基本语法（if）

#### （1） 单项分支：

if 条件表达式：（冒号必须有）

语句1

语句2

语句3

......

条件表达式就是计算结果必须为布尔值的表达式

表达式后面的冒号不能少

注意if后面的出现的语句，如果属于if语句块，则必须同一个锁紧等级

条件表达式结果为True执行if后面的缩进语句块

eg1：age = 17

if age <18 :

print("去叫家长")

print("我们不带你玩")

print("开课了")

输出：age=17去叫家长我们不带你玩

age=19 开课了，与if同一级别的为else

#### （2） 双向分支：

if...else...语句

if 条件表达式:

语句1

语句2

...

else :

语句1

语句2

...

双向分支有两个分支，当程序执行if...else ...语句的时候，一定会执行if或者else中的一个，也仅执行一个

缩进问题，if和else一个层级，其余的语句块一个层级

eg: input的作用：

1、在屏幕上输出括号里面的字符串

2、接收用户输入的内容并返回到程序

3、input返回的类型一定是字符串类型

gender=input("请输入性别：")

print("你输入的姓名是:{0}".format(gender))

if gender=="nan":

print("写作业") 语句块里面的保持一致就可以了，不必所有的都保持一致

else: 但是最好一致，美观

print("发糖喽")

print("11111")

print("开课喽")

#### （3） 多路分支：

很多分支的情况，简称多路分支

if 条件表达式：

语句1

...

elif 条件表达式:

语句1

elif 条件表达式:

语句1

......（elif可以有无限多个）

else:

语句1

elif可以有无限多个

else可选

多路分支只会选择一个执行

eg：

#考试成绩

# 90以上 A

# 80-90 B

# 70-80 C

# 60-70 D

# 60以下 不及格

score = input("请输入学生成绩：")

# 需要把Str转为int(必须为形如int的一个字符串)

score = int(score)

if score > 90 :

print("A")

elif score>=80:

print("B")

elif score>=70:

print("C")

elif score>=60:

print("D")

else:

print("不及格")

--------------------------------------------------------------------------------------------------------------

上面这段代码是下面这段代码的替换

-----------------------------------------------------------------------------------------

**eg：**

#考试成绩

# 90以上 A

# 80-90 B

# 70-80 C

# 60-70 D

# 60以下 不及格

score = input("请输入学生成绩：")

# 需要把Str转为int(必须为形如int的一个字符串)

score = int(score)

if score >= 90 :

print("A")

if score>=80 and score<90:

print("B")

if score>=70 and score<80:

print("C")

if score>=60 and score<70:

print("D")

if score<60:

print("不及格")

但是建议用下面这段代码，上面代码用不好容易出错

### 2.1.2 if语句其他

if语句可以嵌套使用，但不推荐

python语句里面没有switch-case语句

## 2.2 循环（for/while）

重复执行某些规定动作或者处理基本固定的事物

分类

for循环

while循环

### 2.2.1 for循环

#### 2.2.1.1 for循环结构

for 变量 in 序列:

语句1

语句2

......

列表：列表就是一列数字或者其他值，一般用中括号表示

例如：{'张三','李四','王五','静静'}

打印学生列表姓名

for name in {'张三','李四','王五','静静'}: (不要忘记冒号)

print(name)

eg2：

for name in {'张三','李四','王五','静静'}:

print(name)

if name=="静静":

print("我的最爱{0}出现了".format(name))

else:

print("同学我们不约{0}，同学请自重".format(name))

#### 2.2.1.2 range介绍

生成一个数字序列

具体的范围可以设定

注意：一般在Python，如果有表示数字范围的两个数，一般是包含左边数字不包含右边数字。

range函数在Python2和Python3有严重区别，现在说的是Python3

例如：打印从1到10的数字

for i in range(1,11):

print(i)

#### 2.2.1.3 for-else:语句

当for循环结束的时候，会执行else语句

else语句可选

例如：

# 打印列表中的同学，如果没有胡泽红列表结束了，我们需要打出提示语句，表示不再爱了

for name in {'张三', '静静','李四','王五'}:

print(name)

if name=="静静":

print("我的最爱{0}出现了".format(name))

else:

print("同学我们不约{0}，同学请自重".format(name))

else:

print("别的都不是我的学僧，我不会再爱了")

#### 2.2.1.4 for循环之break, continue, pass

break： 无条件结束整个循环,简称循环猝死

continue： 无条件结束本次循环，重新进入下一轮循环

pass： 表示略过

##### （1）break

# 在1-10的数字中，寻找数字7，一旦找到，打印出来，其余则什么都不做

# for循环中的标量表示，一般用 i,k,m,n 或者index，idx,item之类

# 在Python中，故循环名称不重要，可以用下划线（\_）代替

# for \_ in range(1,11):

# print("i love hanfengxuan ")

for i in range(1,11):

if i == 3 :

print("我找到了")

break

else:

print(i)

结果：

1

2

我找到了

##### （2）continue

例题：在1-10的数字中，寻找所有偶数，一旦找到，打印偶数，奇数什么也不做

【1】

for i in range(1,11):

if i % 2 == 0 :

print("我找到了偶数{0}".format(i))

continue

else:

print("{0}不是偶数哦".format(i))

【2】

for i in range(1,11):

if i % 2 == 0 :

print("我找到了偶数{0}".format(i))

【3】

# 本案例完整展示continue的用法和作用

for i in range(1,11):

if i % 2 == 1 :

continue

print("我找到了偶数{0}".format(i))

##### （3）pass

pass一般用于占位

pass没有跳过功能

for i in range(1,11):

pass

print("我在占位")

### 2.2.2 while循环

一个循环语句

表示当某条件成立的时候，就循环

不知道具体循环次数，但能确定循环条件成立的时候用while循环

while循环语法1：

while 条件表达式：

语句块

eg:

# 如果说年利率是6.7%，本钱是每年翻滚，则多少年后本钱会翻倍

benqian=100000

year=0;

while benqian < 200000:

benqian= benqian \* (1+0.067)

year+=1; # year = year+1

print("第{0}年拿了{1}块钱".format(year,benqian))

while循环语法2：

while 条件表达式：

语句块1

else:

语句块2

例题1：

# 99乘法表

for m in range(1,10):

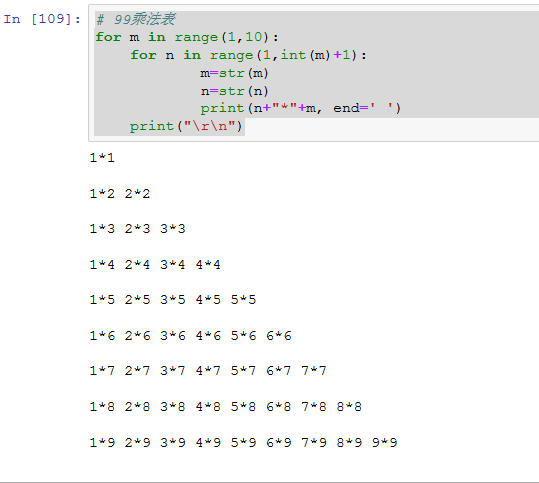
for n in range(1,int(m)+1):

m=str(m)

n=str(n)

print(n+"\*"+m, end=' ') （限制不换行）

print("\r\n")



例题2：

# 杨辉三角

for m in range(1,8):

for n1 in range(1,8-m):

print(" ",end='')

for n2 in range(1,m+1):

if (n2==1) or ((n2==m) and (m!=1)):

print("1"+" ",end='')

# elif (m>=6) and ((n2==3) or (n2==m-2)):

# print("&"+" ",end='')

# elif (m>=5) and (m%2!=0) and(n2==(m//2+1)):

# print(str(2\*(m-2))+" ",end='')

else:

print(str(m-1)+" ",end='')

print("\r\n")

# 3、函数

# 查找函数帮助文档

# 1、用help函数

help(print)



函数：

代码的一种组织形式

一个函数一般完成一项特定的功能

函数定义：

函数使用：

1、函数需要先定义

2、使用函数，俗称调用

## 3.1定义函数以及函数的调用

###### 3.1.1 定义函数

# 只是定义不会执行

# 1、def关键字，后跟一个空格

# 2、函数名，自己定义，起名需要遵循命名规则，约定俗成，大驼峰命名只给类用

# 3、后面括号和冒号不能省略，括号内可以有参数

# 4、函数内所有代码缩进

###### 3.1.2 函数调用

# 直接函数名后面跟括号

func()

## 3.2 函数的参数和返回值

1、参数：负责给函数传递一些必要的数据或信息

（1）形参：（形式参数）在函数定义的时候用到的参数没有具体值，只是一个占位的符号，称为形参。

（2）实参：（实际参数）在调用的时候输入的值。

2、返回值：函数的执行结果

（1）使用return关键字

（2）没有return 默认返回None

（3）函数一旦执行return，则无条件返回，即结束函数的执行

###### 3.2.1参数的定义和使用

（1） 例子：

# 参数person只是一个符号，代表的是调用的时候的某一个数据

# 调用的时候，会用p代替所有的person

def hello(person):

print("{0},你肿么了".format(person))

print("Sir,你不理额饿就走咧")

p="fujia"

hello(p)

输出：



（2） return例子

def hello(person):

print("{0},你肿么了".format(person))

return "haha ,我提前结束了"

print("Sir,你不理额饿就走咧")

return "我已经跟{0}打招呼了，{1}不理我".format(person,person)

p="fujia"

rst=hello(p)

print(rst)

输出：

fujia,你肿么了

haha ,我提前结束了

（3）使用函数打印99乘法表

def printLine(row):

for col in range(1,row+1):

# print函数默认任务打印完成之后换行

print(row\*col ,end=' ')

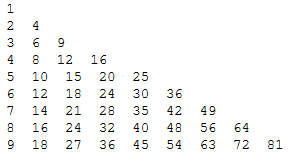
print()

for row in range(1,10):

# 打印一行

printLine(row)

效果图：



###### 3.2.2 参数详解

参考资料：https://www.cnblogs.com/bingabcd/p/6671368.html

python参考资料：headfirst python -> 零基础入门学习Python（小甲鱼）

---->后期可以考虑腾讯公开免费课

参数分类：

（1）普通参数

（2）默认参数

（3）关键字

（4）收集参数

3.2.2.1 普通参数

参见上例子

定义的时候直接定义变量名

测试的时候直接把变量或者值放入指定位置

--------------------------------------------------------------------------------------

定义：

def func(参数1，参数2，...):

函数体

调用：

func(value1，value2,....)

调用的时候，具体值参考的是位置，按位置赋值

--------------------------------------------------------------------------------------

3.2.2.2 默认参数

形参带有默认值

调用的时候，如果没有对应形参赋值，则使用默认值

--------------------------------------------------------------------------------------

def func\_name(p1=v1,p2=v2):

func\_block

调用1：

func\_name()

调用2：

value1=100

value2=200

func\_name(value1,value2)

# 默认参数实例

# 报名函数，需要知道学生性别

# 学习Python的基本都是男生，默认是男生

def reg(name,age,gender="male"):

if gender=="male":

print("{0} is {1} ,and he is a good student".format(name,age))

else:

print("{0} is {1} ,and she is a good student".format(name,age))

# 调用默认1

reg("xiaoming",21)

# 调用默认2

reg("xiaojing",22,"famale")

输出：

xiaoming is 21 ,and he is a good student

xiaojing is 22 ,and she is a good student

3.2.2.3 关键字参数

规范：Python语句中 = 左右要加空格，参数的=左右没有空格，‘，’后面有一个空格

（1） 语法：

def func(p1=v1, p2=v2....):

func\_body

调用：

func(p1=value1, p2=value2....)

（2） 比较麻烦，但是也有好处：不容易混淆，一般实参和形参按照位置一一对应即可，容易出错，使用关键字参数，可以不考虑参数的位置

# 关键字参数实例

# 普通参数只按照位置传递，容易错

def stu\_key(name="no name", age=0 , addr="no addr"):

print("i am s student")

print("我叫{0}，我今年{1}岁了，我住{2}".format(name, age, addr))

stu\_key(name="fujai", age=18, addr="北票")

3.2.2.4 收集参数

1、基本概念及语法

（1） 把没有位置，不能和定义时的参数位置对应的参数，放入一个特定的数据结构中。

（2） 语法：

def func( \*args):

func\_body

按照list使用方式访问args得到传入的参数

调用：

func(p1,p2,p3,...)

（3） 参数名args不是必须这么写，（推荐这么写），约定俗成

（4） 参数名args前需要有星号\*\*\*\*\*

（5） 收集参数可以和其他参数并存

# 收集参数实例1

# 函数模拟一个学生进行自我介绍，但是具体内容不清楚

# args把它看做一个list

def stu( \*args):

print("hello,大家好，我自我介绍一下，简单说两句：")

# type函数左右是检测变量的类型

print(type(args))

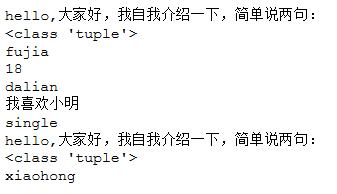
for item in args:

print(item)

stu("fujia",18,"dalian","我喜欢小明","single")

stu("xiaohong")

输出：



# 收集参数案例2

# 说明收集参数可以不带任何实参调用，此时收集参数为空tuple

stu()

2、收集参数之关键字收集参数

（1） 把关键字参数按字典格式存入收集参数

（2） 语法：

def func( \*\*kwargs):

func\_body

调用

func(p1=v1, p2=v2, p3=v3...)

（3）kwargs一般约定俗成

（4） 调用的时候，把多余的关键字参数放入kwargs

（5） 访问kwargs的时候需要按照字典格式访问

# 收集参数实例3

# 自我介绍

# 调用的时候使用关键字参数调用

def stu( \*\*kwargs):

# 在函数体内对于kwargs的使用时候不需要加\*号

print("hello,大家好，我自我介绍一下，简单说两句：")

print(type(kwargs))

# 对于字典的访问，Python2和Python3有区别

for k,v in kwargs.items():

print(k,"-----",v)

stu(name="fujia", age=18, addr="通州", lover="小明", work="teacher")

# 字符串的乘法，把该字符串打印几次

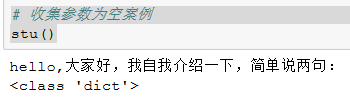
print("\*"\*50)

stu(name="周大神")



# 收集参数为空案例

stu()



**3、**收集参数混合调用的顺序问题

（1） 收集参数，关键字参数，普通参数可以混合使用

（2） 使用规则是，普通参数和关键字参数优先

（3） 定义的时候一般找普通参数，关键字参数，收集参数tuple，收集参数dict

# 收集参数混合调用例子

# stu模拟一个学生的自我介绍

def stu(name, age, \*args, hobby="没有", \*\*kwargs): 注意顺序，什么都有的时候收集参数在前面

print("hello大家好")

print("我叫{0}，我今年{1}岁。".format(name,age))

if hobby=="没有":

print("我没有爱好,很抱歉")

else:

print("我的爱好是{0}".format(hobby))

print("\*"\*20)

for item in args:

print(item)

print("#"\*30)

for k,v in kwargs.items():

print(k +"----------"+v)

# 开始调用

name="fujia"

age=19 在前面，带=号的在后面

stu(name, age)

stu(name,age,"王晓静","陈石头",hobby="swim",hobby2="烹饪",hobby3="做饭")

4、收集参数的解包问题

（1） 把参数注入list中，直接把list/dict中的值放入收集参数中

（2） 语法：

# 收集参数的解包问题案例

def stu( \*args):

print("哈哈哈")

# 用来表示调试

n = 0

for i in args:

print(type(i))

print(n)

n += 1

print(i)

# stu("付佳","222","333","呵呵呵")

l = list()

# l.append("付佳")

# l.append(20)

# l.append(230)

l1=["付佳",200,101]

# 此时，args的表示形式是字典内一个list类型的元素，即args=(["fujia",120])

# 很显然跟我们最初的想法违背

# 此时的调用，我么就需要解包符号，即调用的时候前面加\*

# 同理，dict类型收集参数也可以一样解包，但是需要在调用之前加两个星号\*\*

stu(\*l1)

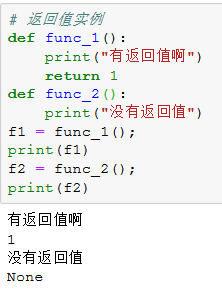
###### 3.2.3返回值

（1） 函数和过程的区别：有无返回值

（2） 需要用return显示返回内容

（3） 如果没有返回，则默认返回None

（4） 推荐写法：无论有无返回值，最后都要以return结束



## 3.3函数文档

函数文档的作用是对当前函数提供使用相关的参考信息

文档的写法：

（1） 在函数内部开始的第一行使用三引号定义符

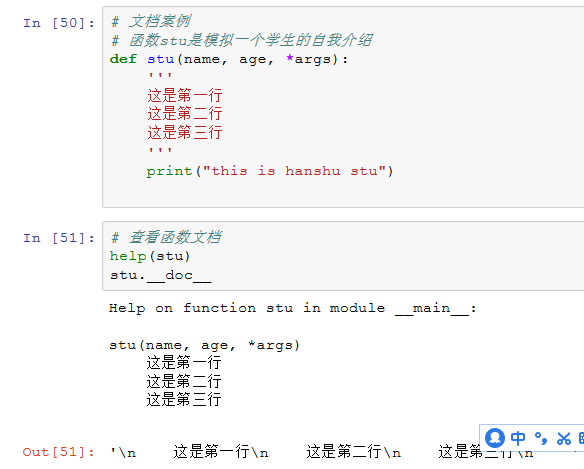
（2） 一般具有特定格式

（3） 参看案例

查看方法：

（1） 使用help函数，形如help(func)

（2） 使用doc {stu,\_\_doc\_\_} 左右分别两个下划线



第一行：文档的文字内容

第二行：参数的详细说明 param name :表示姓名

最后：参数的返回值

## 3.4函数的作用域

1、变量由作用范围限制

2、分类：按照作用域分类

（1） 全局（global）:在函数外部定义

（2） 局部（loacal）:在函数内部定义

3、变量作用范围：

（1） 全局变量：在整个全局范围内有效

（2） 全局变量在局部可以使用，（即函数内部可以使用函数外部定义的变量）

（3） 局部变量寻思局部可以使用

（4）局部变量在全局范围无法使用

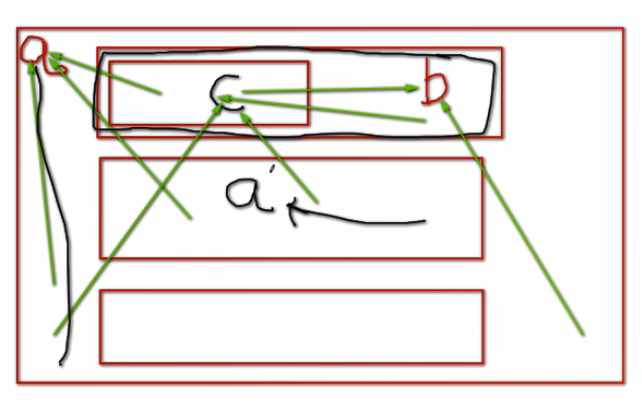
4、LEGB原则

L (local）:局部作用域

E (Enclosing function locale)：外部嵌套函数作用域

G (Global module）:函数定义所在模块作用域

B (Buildin) :Python内置魔抗的作用域



里面可以访问外面，外面不可以访问里面，晓得可以访问大的，大的不可以访问小的。

5、提升局部变量为全局变量

（1） 使用global： 如果真的想要在函数体内修改全局变量的值，就要使用global关键字

# 提升案例

def fun():

global b1

b1 = 100

print("变化后面="+str(b1))

print("i am a fun")

# # a2的作用域是fun

# b2 = 99

# print(b2)

b1=1

print("变化前"+str(b1)) fun()

# print(b2)

输出：

变化前1

变化后面=100

i am a fun

（2） 使用nonlocal

nonlocal，字面意思就是指当前的这个变量不是局部变量。nonlocal是Python3.0中新增的关键字，python2.x不支持

def fun():

num2=3

def fun2():

nonlocal num2

num2\*=2

print("num2=",num2)

return fun2()

fun()

输出：num2= 6

## 3.5 常见函数

### 3.5.1 globals、locals函数

（1） 可以使用glabals和locals显示出局部变量和全局变量

（2） 参见以下案例

（3） globals 和 locals函数叫做内建函数

a = 1

b = 2

def fun(c,d):

e = 111

print("Locals={0}".format(locals()))

print("Globals={0}".format(globals()))

fun(100,200)

### 3.5.2 eval()函数

# 把一个字符串当成一个表达式来执行，返回表达式执行侯的结果

# 语法：

eval(string\_code,global=None,locals=None)

例子：

x=100

y=100

z1 = x+y

z2 = eval("x+y")

print(z1)

print(z2)

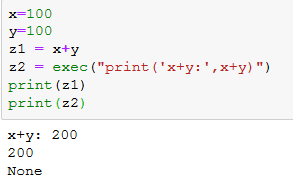
### 3.5.3 exec()函数

跟eval功能类似，但是，不返回结果

语法：

exec(string\_code,global=None,locals=None)

例子：



### 3.5.4 递归函数

函数直接或者间接调用自身

优点：简洁，理解容易

缺点：对递归深度有限度，消耗资源大

Python对递归深度有限制，超过限制报错

在写递归程序的时候，一定要注意结束条件

# 递归调用深度限制代码

x = 0

def fun():

global x

x += 1

print(x)

# 函数自己调自己

fun()

# 调用函数

fun()

程序报错：递归错误，超过了最大的递归次数，所以要注意结束条件。

RecursionError: maximum recursion depth exceeded in comparison

经典案例：

# 斐波那契数列

# 一列数组，第一个值是1，第二个也是1，从第三个开始，每一个数字的值等于钱两个数字出现的值的和

# 数学公式为： f(1)=1, f(2)=1 ,f(n)=f(n-1)+f(n-2)

# 例如： 1，1，2，3，5，8，13...

# n表示第n个数字的斐波那契数列值

def fib(n):

if n == 1:

return 1

if n == 2:

return 1

return fib(n-1) + fib(n-2)

fib(10)

输出：55

经典案例：

# 汉诺塔问题

- 规则：

1. 每次移动一个盘子

2. 任何时候大盘子在下面，小盘子在上面

- 方法：

1. n=1: 直接把A上的盘子移动到C上面 A->C

2. n=2:

1. 把小盘子从A移动到B上， A->B

2. 把大盘子从A移动到C上， A->C

3. 把小盘子从B移动到C上， B->C

3. n=3:

1. 把A上的两个盘子通过C移动到B上去 调用递归实现

2. 把A上的最大盘子，也是唯一一个，移动到C上去 A->C

3. 把B上的n-1个盘子，借助于A，移动的C上 调用递归

方法：

def hano(n, a, b, c):

'''

n: 代表几个盘子

a: 代表第一个盘子

b: 代表第二个盘子

c: 代表第三个盘子,

'''

if n == 1:

print(a, "--->", c)

return None 不要忘记返回值，否则会报堆栈错误

if n == 2:

print(a, "--->", b)

print(a, "--->", c) 这部分没有也没事，但是最好让人清晰明白

print(b, "--->", c)

return None

hano(n-1, a, c, b)

print(a, "--->", c) 不要忘记这个

# 把n-1个盘子，从b塔借助a塔,挪到c塔上去

hano(n-1, b, a, c)

a = "A"

b = "B"

c = "C"

n = 1

hano(n, a, b, c)

# 4、内置数据结构（变量类型）

## 4.1 list（列表）

（1）一组有顺序的数据的组合

（2）创建列表

# 1、创建空列表

l1 = []

# type是内置函数，辅助打印变量的类型

print(type(l1))

print(l1)

# 2、创建带值的列表

l2 = [100]

print(type(l2))

print(l2)

# 3、创建带值的列表

l3 = [2,4,5,1,8]

print(type(l3))

print(l3)

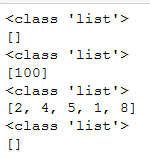
# 4、创建list（）

l4 = list()

print(type(l4))

print(l4)

输出：



#### 4.1.1 列表常用操作

##### 1、访问：

（1） 使用下标操作（索引）

（2） 列表的位置是从0开始

l = [1,12,3,14,5]

print(l[2])

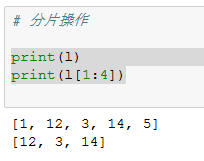
##### 2、分片操作：

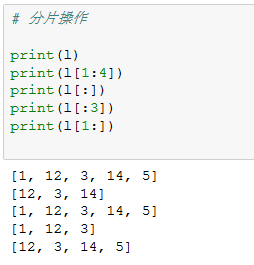
（1） 对列表进行任意一段的截取，注意截取的范围，包含左边的下标值，不包含右边的下标值。

（2） 下标值可以为空，如果不写，左边下标值默认为0，右边下标值为最大数加1，即表示截取到最后一个数据。

（3）分片可以控制增长幅度，默认增长幅度为1

list[：]





（3） 分片可以控制增长幅度

###### 例1：增长幅度为1

print(l[1:5:1])

###### 例2：增长幅度为2，隔一个打印一个

print(l[1:5:2])

###### 例3：下标可以超出范围，超出后不在考虑多余下表内容

print(l[2:10])

输出：[12, 3, 14, 5]

[12, 14]

[3, 14, 5]

###### 例4：下标值，增长幅度可以为负数

# 为负数，表明顺序是从右往左

# 规定：数组最后一个数字的下表是-1

# 分片之负数下标

print(l)

# 下面显示是空，因为默认分片总是从左向右截取

# 即正常情况，分片左边的值一定要小于右边的值

print(l[-2:-4])

print(l[-4:-2])

# 如果分片一定左边的值比右边大，则步长参数需要使用负数

# 此案例为一个list直接正反颠倒提供了一种思路

print(l[-2:-4:-1])

输出：

[1, 12, 3, 14, 5]

[]

[12, 3]

[14, 3]

（4） 分片操作是生成一个新的list

内置函数id，负责显示一个变量或者数据的唯一确定编号

# 通过id可以直接判断出分片是重新生成了一份数据还是使用的同一份数据

l = [3,2,54,54,645,1]

l1 = l[:]

lll = l1

# 如果两个id值一样，则表明分片产生的列表是使用的同一个地址同一个数据

# 否则，则表明分片是从新生成了一份数据，即一个新的列表，然后把数据拷贝到列表中

print(id(l))

print(id(l1))

print(id(lll))

# 通过id知道，l1和lll是同一份数据，验证代码如下

l[1] = 100

print(l)

print(l1)

l1[1] = 100

print(l1)

print(lll)

输出：

84579400

84578760

84578760

[3, 100, 54, 54, 645, 1]

[3, 2, 54, 54, 645, 1]

[3, 100, 54, 54, 645, 1]

[3, 100, 54, 54, 645, 1]

# 汉诺塔问题

- 规则：

   1. 每次移动一个盘子

   2. 任何时候大盘子在下面，小盘子在上面

- 方法：

   1. n=1: 直接把A上的盘子移动到C上面 A->C

   2. n=2:

       1. 把小盘子从A移动到B上， A->B

       2. 把大盘子从A移动到C上， A->C

       3. 把小盘子从B移动到C上， B->C

   3. n=3:

       1. 把A上的两个盘子通过C移动到B上去 调用递归实现

       2. 把A上的最大盘子，也是唯一一个，移动到C上去 A->C

       3. 把B上的n-1个盘子，借助于A，移动的C上   调用递归

def hano(n, a, b, c):

**'''**

**n: 代表几个盘子**

**a: 代表第一个盘子**

**b: 代表第二个盘子**

**c: 代表第三个盘子,**

**'''**

if n == 1:

print(a, "--->", c)

return None

if n == 2:

print(a, "--->", b)

print(a, "--->", c)

print(b, "--->", c)

return None

hano(n-1, a, c, b)

print(a, "--->", c)

**# 把n-1个盘子，从b塔借助a塔,挪到c塔上去**

hano(n-1, b, a, c)

a = "A"

b = "B"

c = "C"

n = 1

hano(n, a, b, c)

##### 3、列表方法：

###### 3.1 del方法

1、del 删除命令:在原有的基础的删除，删除之后没有生成新的list

a = [13, 12, 32, 4]

print(id(a))

del a[2]

print(id(a))

print(a)

结果：

78908680

78908680

[13, 12, 4]

2、del一个变量后不能再继续使用此变量

del a

print(a)

###### 3.2 列表相加

使用加号连接两个列表

a = [1, 2, 3, 4, 5]

b = [11, 22, 33, 44, 55]

d = ['a', 'b', 'c']

c = a + b + d

print(c)

输出结果：

[1, 2, 3, 4, 5, 11, 22, 33, 44, 55, 'a', 'b', 'c']

###### 3.3使用乘号操作列表

列表跟一个整数相乘

a = [1, 2, 3, 4, 5]

b = a \* 2

print(b)

输出：

[1, 2, 3, 4, 5, 1, 2, 3, 4, 5]

###### 3.4 成员资格运算

判断一个元素是否在list里面

a = [1, 2, 3, 4, 5]

b = 1

c的值一定是一个布尔值

c = b in a

print(c)

b = 4

c = b not in a

print(c)

结果：

True

False

###### 3.5 列表的遍历

3.5.1 for循环遍历

**例一：**

**# for in list**

a = [1, 2, 4, 56, 6]

for i in a:

print(i)

**例二：**

**# range**

**# in 后面的变量要求是可以迭代的内容**

for i in range(1, 6):

print(i)

print(type(range(1,2)))

**例三：**

a = [1, 2, 3, 4]

for i in range(0, len(a)):

print(a[i])

i += 1

3.5.2 while循环遍历

**# while循环访问list**

**# 一般不用while遍历list**

a = [1, 2, 3, 4]

length = len(a)

**# index表示list的下标**

index = 0

while index < length:

print(a[index])

index += 1

3.5.3 双层列表循环

**例1：**

**# 双层列表循环**

**# 双层列表或者a为嵌套列表**

a = [["one", 1], ["two", 2], ["three", 3]]

for k, v in a:

print(k, "--", v)

结果：

one -- 1

two -- 2

three -- 3

**例2：**

**# 这个例子说明k, v, w的个数应该跟解包出来的个数一致**

a = [["one", 1, 1], ["two", 2, 1], ["three", 3, 1]]

for k, v, w in a:

print(k, "--", v, "--", w)

结果：

one -- 1 -- 1

two -- 2 -- 1

three -- 3 -- 1

###### 3.6列表内涵：list content

通过简单方法创建列表

3.6.1 for创建

**（1）用list a创建一个 list b**

a = ['a', 'b', 'c']

**# 用list a创建一个 list b**

**# 下面的代码含义是，对于所有a中的元素，逐个放入新列表b中**

b = [i for i in a]

print(b)

结果：

['a', 'b', 'c']

**示例二：**

**# 对a中的所有元素×10，生成一个新的list**

a = ['a', 'b', 'c', 'd']

c = [1, 2, 3, 4]

**# 用list a创建一个 list b**

**# 下面的代码含义是，对于所有a中的元素，逐个放入新列表b中**

b = [i\*5 for i in a]

d = [i\*10 for i in c]

print(b)

print(d)

结果：

['aaaaa', 'bbbbb', 'ccccc', 'ddddd']

[10, 20, 30, 40]

**示例三：**

**# 还可以过滤把原来list中的内容放入新列表**

**# 比如原有列表a ,需要把a中的偶数生成新列表b**

a = [i for i in range(1, 35)] #生成1~34的整数

print(a)

b = [m for m in a if m % 2 == 0]

print(b)

结果：

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34]

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34]

（4） 列表生成式也可以嵌套

# 有两个列表a, b

a = [i for i in range(1,4)] #生成list a

print(a)

b = [i for i in range(100, 400) if i % 100 ==0]

print(b)

# l列表生成可以嵌套, 此时等于两个for循环嵌套

c = [ m + n for m in a for n in b]

print(c)

# 上面代码跟下面代码等价

for m in a:

for n in b:

print(m+n ,end=" ")

print()

# 嵌套的列表生成式也可以用条件表达式

c = [ m + n for m in a for n in b if m+n <250]

print(c)

结果：

[1, 2, 3]

[100, 200, 300]

[101, 201, 301, 102, 202, 302, 103, 203, 303]

101 201 301 102 202 302 103 203 303

[101, 201, 102, 202, 103, 203]

##### 4、列表的常见函数：

###### 4.1 len()：列表的长度

a = [i for i in range(1,100)]

print(len(a))

输出：99

###### 4.2 max():求列表最大值

###### min():求列表最小值

a = [i for i in range(1,100)]

print(max(a))

b = ["man", "yython", "python"]

print(max(b))

输出：

99

yython

###### 4.3 list() :将其他格式的数据转换为list

例一：

s = "l love wang xiao jing"

print(list(s))

输出：

['l', ' ', 'l', 'o', 'v', 'e', ' ', 'w', 'a', 'n', 'g', ' ', 'x', 'i', 'a', 'o', ' ', 'j', 'i', 'n', 'g']

例二：

# 把range产生的内容转换为list

print(list(range(1,19)))

输出：

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]

###### 4.4 append()插入一个内容

a = [i for i in range(1, 5)]

print(a)

a.append(100)

print(a)

输出：

[1, 2, 3, 4]

[1, 2, 3, 4, 100]

###### 4.5 insert():指定位置插入

语法： insert(index, data), 插入位置是index的前面

print(a)

a.insert(3, 666)

print(a)

输出：

[1, 2, 3, 4, 100]

[1, 2, 3, 666, 4, 100]

###### 4.6 删除

（1） del: 删除指定下标元素

（2） pop: 从队尾拿出一个元素，即把最后一个元素取出来，有返回值

print(a)

last\_ele = a.pop()

print(last\_ele)

print(a)

输出：

[1, 2, 3, 666, 4, 100]

100

[1, 2, 3, 666, 4]

（3） remove: 在列表中删除指定的值的元素

如果被删除的值没存在与list中，则报错

即：删除list指定值的操作应该使用try...expect语句，或者现行判断

if x in list:

list.remove(x)

a.insert(4,4)

print(a)

print(id(a))

a.remove(4)

print(a)

print(id(a))

输出：输出两个id一样，说明remove操作是对list直接操作

[2, 3, 100, 4, 4]

88167304

[2, 3, 100, 4]

88167304

###### 4.7 clear():清空

但是列表结果还在，只是清空数据

print(a)

print(id(a))

a.clear()

print(a)

print(id(a))

a = []

print(a)

print(id(a))

输出：

[1, 2, 3, 4]

84150216

[]

84150216

如果不需要列表地址保持不变，则清空列表可以使用一下方式：

a = []

print(a)

print(id(a))

输出：

[]

84151880

###### 4.8 reverse: 翻转列表内容，原地翻转

a = [1, 2, 3, 4, 5]

print(a)

print(id(a))

a.reverse()

print(a)

print(id(a))

输出：[1, 2, 3, 4, 5]

88046216

[5, 4, 3, 2, 1]

88046216

###### 4.9 extend：扩展列表

两个列表，把一个直接拼接到后一个上

a = [1, 2, 3, 4, 5]

b = [1, 2, 6, 7, 8, 9, 10]

print(a)

print(id(a))

a.extend(b)

print(a)

print(id(a))

输出：

[1, 2, 3, 4, 5]

83894152

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

83894152

###### 4.10 count：查找列表中指定值或元素的个数

a.append(1)

a.insert(1, 1)

print(a)

a\_len = a.count(1)

print(a\_len)

输出：

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

18

###### 4.11 copy：浅拷贝

# 列表类型变量赋值示例

a = [1, 2, 3, 4, 5, 666]

print(a)

# list类型，简单赋值操作，是传地址

b = a

b[3] = 777

print(a)

print(id(a))

print(b)

print(id(b))

print("-------------------------")

# 为了解决以上问题，通常我们使用copy()

a = [1, 2, 3, 4, 5, 666]

print(a)

b = a.copy()

b[3] = 777

print(a)

print(b)

print(id(a))

print(id(b))

输出：

[1, 2, 3, 4, 5, 666]

[1, 2, 3, 777, 5, 666]

88167496

[1, 2, 3, 777, 5, 666]

88167496

-------------------------

[1, 2, 3, 4, 5, 666]

[1, 2, 3, 4, 5, 666]

[1, 2, 3, 777, 5, 666]

84151880

88046920

（2） 深拷贝跟浅拷贝的区别

出现下列问题的原因是，copy函数是个浅拷贝函数，即只是拷贝一层内容

深拷贝需要使用特定工具

a0 = [10, 20, 30]

a = [1, 2, 3, a0]

b = a.copy()

print(id(a))

print(id(b))

print(id(a[3]))

print(id(b[3]))

a[3][2] = 666

print(a)

print(b)

输出：  
83892616

83849736

84148872

84148872

[1, 2, 3, [10, 20, 666]]

[1, 2, 3, [10, 20, 666]]

##### 5、传值和传地址的区别

对于简单的数据，参与传值操作，即在函数内对参数的操作不影响外面的变量

对于复杂变量，采用传地址操作。此时函数内的参数和外部变量是同一份内容，任何地方对此内容的更改都影响另外的变量活参数的使用。

## 4.2 set

- 集合是高中数学的一个概念

- 一堆确定的无序的唯一的数据，集合中每一个数据称为一个元素

### 4.2.1 集合的定义

s = set()

print(s)

print(type(s))

# 此时，大括号里面一定要有值，否则定义出来的是一个dict

s = {1, 2, 4, 5}

print(s)

输出：

set()

<class 'set'>

{1, 2, 4, 5}

# 如果只是用大括号定义，则定义的是一个dict类型

d = {}

print(type(d))

print(d)

输出：

<class 'dict'>

{}

### 4.2.2 集合的特征

- 集合内数据无序，即无法使用索引和分片

- 集合内部数据元素具有唯一性，可以用来排除重复数据

- 集合内的数据，str, int, float, tuple, 冰冻集合等，即内部只能防止可哈希数据

- 集合本身不可哈希

### 4.2.3 集合序列操作

# 成员检测

# in, not in

s = {4, 5, "i", "love", "wangxiaojing"}

print(s)

if "love" in s:

print("哎")

if "hahh" not in s:

print("哎个锤子")

输出：

{4, 5, 'wangxiaojing', 'i', 'love'}

哎

哎个锤子

### 4.2.4 集合遍历操作

# for循环

s = {4, 5, "i", "love", "wangxiaojing"}

for i in s:

print(i, end=" ")

输出：4 5 wangxiaojing i love

# 带有元祖的集合遍历

s = {(1, 2, 3), ("i", "love", "wangxiaojing"), (4, 5, 6)}

#### 1、单个输出

for k, m ,n in s:

print(k, "--", m, "--", n)

输出：

4 -- 5 -- 6

i -- love -- wangxiaojing

1 -- 2 -- 3

#### 2、整体输出

for k in s:

print(k, end=" ,")

输出：(4, 5, 6) ,('i', 'love', 'wangxiaojing') ,(1, 2, 3) ,

### 4.2.5 集合的内涵

# 普通集合内涵

# 以下集合在初始化后自动过滤重复元素

s = {1, 2, 3, 14, 15, 6, 7, 11, 1, 2, 3, 4}

print(s)

输出：{1, 2, 3, 4, 6, 7, 11, 14, 15}

# 普通集合内涵

ss = {i for i in s}

print(ss)

输出：{1, 2, 3, 4, 6, 7, 11, 14, 15}

# 带条件的集合内涵（集合生成式）

sss = {i for i in s if i % 2 == 0}

print(sss)

输出：{2, 4, 6, 14}

# 多循环的集合内涵(无规则)

s1 = {1, 2, 3, 4}

s2 = {"i", 'love', 'wangxiaojing'}

s = {m\*n for m in s2 for n in s1 if n == 2}

print(s)

输出：{'lovelove', 'ii', 'wangxiaojingwangxiaojing'}

### 4.2.6 集合函数、关于集合的函数

#### 1、 len, max, min：跟其他基本函数一致

s = {83, 12, 1, 3543, 45, 1}

print(len(s))

print(max(s))

print(min(s))

输出：5

3543

1

#### 2、# set：生成一个集合

l = [1, 2, 43, 4, 6, 1, 2, 43, 5]

s = set(l)

print(s)

输出：{1, 2, 4, 5, 6, 43}

#### 3、# add: 向集合内添加元素

s = {1, 2, 4, 5, 7, 3}

s.add(11)

print(s)

输出：{1, 2, 3, 4, 5, 7, 11}

#### 4、# clear: 清空

s = {1, 2}

print(id(s))

s.clear()

print(id(s))

print(s)

# 结果表明clear只是原地把数据清空

输出：84652776

84652776

set()

#### 5、# copy拷贝

# remove: 移除指定的值直接改变原有值，如果要删除的值不存在，报错

# discard：移除集合中指定的值，跟remove一样，但是删除不存在不报错

s = {1, 2, 4, 5, 6}

s.remove(4)

print(s)

s.discard(1)

print(s)

print("\*"\*20)

s.discard(100)

print(s)

s.remove(100)

print(s)

输出：{1, 2, 5, 6}

{2, 5, 6}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

{2, 5, 6}

# 为啥remove不存在的值会报keyerror

#### 6、# pop：随机移除一个元素

s = {11, 2, 3, 4, 5, 6, '11', 'kk'}

d = s.pop()

print(d)

print(s)

输出：2

{3, 4, 5, 6, 11, '11', 'kk'}

### 4.2.7 集合函数

# intersection: 交集

# difference: 差集

# union：并集

# issubset: 检查一个集合是否为另一个子集

# isSuperSet: 检查一个集合是否为另一个超集

举例：

s1 = {1, 2, 3, 4, 5, 6}

s2 = {5, 6, 7, 8, 9}

s\_1 = s1.intersection(s2)

print(s\_1) ----------{5, 6}

s\_2 = s1.difference(s2) # 谁差谁

s\_3 = s2.difference(s1)

print(s\_2) ----------{1, 2, 3, 4}

print(s\_3) ----------{8, 9, 7}

s\_4 = s1.issubset(s2)

print(s\_4) ----------False

s\_5 = s1.union(s2)

print(s\_5) ----------{1, 2, 3, 4, 5, 6, 7, 8, 9}

### 4.2.8 集合的数学操作

s1 = {1, 2, 3, 4, 5, 6}

s2 = {5, 6, 7, 8, 9}

s\_1 = s1 - s2

print(s\_1)

输出：{1, 2, 3, 4}

### 4.2.9 frozen set :冰冻集合

* 冰冻就是不可以进行任何修改的集合
* frozenset就是一种特殊集合

# 创建

s = frozenset()

print(type(s))

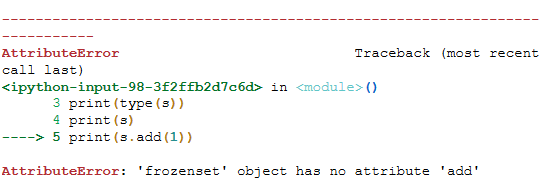
print(s)

print(s.add(1))

输出：

<class 'frozenset'>

frozenset()



## 4.3 dict

字典是一种组合数据，没有顺序的组合数据，数值以键值对形式出现

### 1、字典的创建

# 创建空字典1

d = {}

print(d) --------------{}

print(type(d)) --------------<class 'dict'>

# 创建空字典2

d = dict()

print(d) -------------{}

# 创建有值的字典, 每一组数据用冒号隔开，每一对数据用逗号隔开

d = {"one":1, "two":2, "three":3}

print(d) ------------{'one': 1, 'two': 2, 'three': 3}

# 用dict创建有内容字典1

d = dict({"one":1, "two":2, "three":3})

print(d) ------------{'one': 1, 'two': 2, 'three': 3}

# 用dict创建有内容字典2

# 利用关键字参数创建

d = dict(one = 1, two = 2, three = 3)

print(d) ------------{'one': 1, 'two': 2, 'three': 3}

# 用tuple创建

d = dict([("one", 1),("two", 2), ("three", 3)])

print(d) ------------{'one': 1, 'two': 2, 'three': 3}

### 2、字典的特征

* 字典是序列类型，但是是无序序列，索引没有分片和索引
* 字典中的数据每个都由键值对组成，即k v对
  + key：必须是可哈希的值，比如int, string, float, tuple, 但是，list, set, dict不行
  + value：任何值

### 3、字典常见操作

#### 3.1 访问数据

d = {'one':1, 'two':2, 'three':3}

# 注意访问格式

# 中括号内部是键值

print(d['one']) ---------------1

d["one"] = "eins"

print(d) --------------{'one': 'eins', 'two': 2, 'three': 3}

# 删除某个操作

# 使用del操作

del d["one"]

print(d) --------------{'two': 2, 'three': 3}

#### 3.2 成员检测

# 成员检测 in ,not in

# 成员检测检测的是key的内容

d = {'one':1, 'two':2, 'three':3}

if 2 in d:

print("value")

if "two" in d:

print("key")

if ('two',2) in d:

print("kv")

输出：key

#### 3.3 遍历

# 遍历在Python2 和 Python3 中区别比较大，代码不通用

# 按key来使用for 循环

d = {'one':1, 'two':2, 'three':3}

# 使用for循环，直接按key值访问

for k in d:

print(k, d[k])

print("\*"\*20)

输出：

one 1

two 2

three 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 上述代码可以改写成如下：

for k in d.keys():

print(k, d[k])

print("\*"\*20)

输出：

one 1

two 2

three 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 只访问字典的值

for v in d.values():

print(v)

print("\*"\*20)

输出：

1

2

3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 注意以下特殊语法

for k, v in d.items():

print(k, "--", v)

输出：

one -- 1

two -- 2

three -- 3

### 4、字典生成式

d = {'one':1, 'two':2, 'three':3}

# 常规字典生成式

dd = {k:v for k, v in d.items()}

print(dd) ----------- {'one': 1, 'two': 2, 'three': 3}

# 加限制条件的字典生成式

dd = {k:v for k, v in d.items() if v % 2 == 0}

print(dd) ------------ {'two': 2}

### 5、字典相关函数

#### 5.1 通用函数 len, max, min, dict

# str(字典)：返回字典的字符串格式

d = {'one':1, 'two':2, 'three':3}

print(str(d))

输出：

{'one': 1, 'two': 2, 'three': 3}

#### 5.2 clear:清空字典

# items():返回字典的键值对的元祖格式

d = {'one':1, 'two':2, 'three':3}

i = d.items()

print(type(i))

print(i)

输出：

<class 'dict\_items'>

dict\_items([('one', 1), ('two', 2), ('three', 3)])

#### 5.3 keys:返回字典的键组成的一个结构

k = d.keys()

print(type(k))

print(k)

输出：

<class 'dict\_keys'>

dict\_keys(['one', 'two', 'three'])

#### 5.4 values:同理，一个可以迭代的结构

v = d.values()

print(type(v))

print(v)

输出：

<class 'dict\_values'>

dict\_values([1, 2, 3])

#### 5.5 get: 根据指定键返回相应的值，好处是，可以设置默认值

d = {'one':1, 'two':2, 'three':3}

print(d.get('one')) ----------1

print(d.get("one333")) -----------None

# get默认值是None，可以设置

print(d.get("one333",100)) ----------100

# 体会区别

print(d["one3333"]) -----------报错KeyError

#### 5.5 fromkeys：使用指定的序列作为键，使用一个值作为字典的所有的键的值

l = ["eins", "zwei", "dree"]

# 使用fromkeys两个参数的类型

# 注意fromkeys的调用主体

d = dict.fromkeys(l, "德语数字")

print(d)

输出：

{'eins': '德语数字', 'zwei': '德语数字', 'dree': '德语数字'}

## 4.4 tuple

元祖：元祖可以看成是一个不可更改的list

### 4.4.1 元祖创建

#### （1） 创建空元祖

t = ()

print(type(t))

#### （2） 创建只有一个元素的元祖

t = (1,) # 记得一定要有逗号，否则类型是int型了

print(type(t))

print(t)

t = 1,

print(type(t))

print(t)

#### （3） 创建多个值的元祖

t = (1, 3, 5, 6)

print(type(t))

print(t)

t = 1, 3, 5, 6

print(type(t))

print(t)

# 使用其他结构创建

l = [1, 3, 5, 7]

t = tuple(l)

print(type(t))

print(t)

### 4.4.2 元祖的特性

（1） 是序列表，有序

（2）元祖数据值可以访问，不能修改，不能修改，不能修改

（3） 元祖数据可以是任意类型

（4） 总之，list所有特性，除了不可修改之外，元祖都具有

（5） list具有一些操作，比如索引，分片，序列相加，相乘，成员资格操作等一模一样

#### 4.4.2.1 索引操作

t = (1, 2, 4, 5, 9)

print(t[4])

输出：9

#### 4.4.2.2 超标操作

print(t[10])

输出： IndexError

#### 4.4.2.3 切片操作

t = (1, 2, 3, 4, 5, 6)

t1 =t[1::2] # 下标从1开始，步长是2

print(id(t))

print(id(t1))

print(t1)

# 切片可以超标

t2 = t[1:100]

print(t2)

输出：

80660424

84448024

(2, 4, 6)

(2, 3, 4, 5, 6)

#### 4.4.2.4 序列相加

t1 = (1, 2, 3)

t2 = (5, 6, 7)

# 传地址操作

print(t1)

print(id(t1))

t1 += t2

# 相当于 t1 = t1 + t2

print(t1)

print(id(t1))

# 以上操作类似于

t1 = (1, 2, 4)

t1 = (1, 4, 6)

输出：

(1, 2, 3)

83795832

(1, 2, 3, 5, 6, 7)

54711496

# tuple的不可修改，指得是内容不可修改

t1[1] = 100 #报错TypeError

#### 4.4.2.5 元祖相乘

t = (1, 2, 3)

t = t\*3

print(t)

输出：

(1, 2, 3, 1, 2, 3, 1, 2, 3)

#### 4.4.2.6 成员检测

t = (1, 2, 3)

if 2 in t:

print("YES")

else:

print("NO")

输出：YES

### 4.4.3 成员遍历（一般使用for）

#### （1） 单层元祖遍历

t = (1, 2, 3, "wangxiaojing", "i", "love")

for i in t:

print(i, end=" ")

输出：

1 2 3 wangxiaojing i love

#### （2） 双层元祖的遍历

t = ((1, 2, 3), (2, 3, 5), ("i", "love", "wangxiaojing"))

# 对于以下元祖的遍历，可以如下

# 1.

for i in t:

print(i)

输出：

(1, 2, 3)

(2, 3, 5)

('i', 'love', 'wangxiaojing')

# 2.

for k, m, n in t:

print(k,"--", m, "--", n)

输出：

1 -- 2 -- 3

2 -- 3 -- 5

i -- love -- wangxiaojing

### 4.4.3 关于元祖的函数

* 以下看代码
* 以下函数，对list基本试用

1、# len :获取元祖的长度

# juypter特性：将最后一个元素打印出来，不用print

t = (1, 5, 2, 3, 4, 5)

len(t)

输出：6

2、# max, min最大、最小值

# 如果列表或元祖中有多个最大最小值，则实际打印出哪个:答案：第一个

print(t.index(max(t)))

print(min(t))

3、# tuple: 转化或创建元祖

l = [1, 2, 3, 4, 5]

t = tuple(l)

print(t)

print(type(t))

t = tuple()

print(t)

输出：

(1, 2, 3, 4, 5)

<class 'tuple'>

()

### 4.4.4 元祖的函数

* 基本跟list通用

1、# count: 计算指定数据出现的次数

t = (2, 2, 4, 5, 6, 6, 2, 2)

print(t.count(2))

# index：求指定元素在元祖中的索引位置

print(t.index(5))

# 如果需要查找的数字有很多个，则返回第一个

print(t.index(2))

输出： 4 3 0

### 4.4.5 元祖变量交换法

就是两个变量交换值

# 两个变量交换值

a = 1

b = 2

print(a)

print(b)

print("\*"\*20)

# java程序员

c = a

a = b

b = c

print(a)

print(b)

print("\*"\*20)

# python的写法

a,b = a,b

print(a)

print(b)

输出：

1

2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2

1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2

1（已经交换完成）