## python变量和数据类型

### 1.python的数据类型：

整数

浮点数

空值

布尔

字符串

### 2.python-print语句

print语句后可以跟多个字符串，用逗号隔开就可以一串输出。

### 3.python-注释

#注释

暂时不运行一行代码也可以#

### 4.python变量

变量是用 一个变量名表示，必须是大小写英语字母，数字和下划线组合，且不能是数字开头。

动态语言：变量本身类型不固定的语言，例如A = 124。相对应的就是静态语言。例如 int a =123。

### 5.python定义字符串

字符串中特殊字符进行“转义”\

常见转义字符有：

\n 换行

\t 一个制表符

\\ \本身

### 6.python-raw字符串与多行字符串

在一个字符串前加个r‘’前缀，就表示字符串不需要转义了。不能表示对行字符串，也不能表示包含‘和“。

多行字符串用...表示。

### 7.python-Unicode字符串

字符串中编码问题：最早计算机设计是8个bit作为一个字节（byte）。一个字节最大的整数是255,0-255来表示大小写字母、数字和一些符号，这个编码表被称为ascii编码，比如A的编码是65,z的编码为122.

表示中文是不够的，至少要两个字节，中国制定了GB2312编码。类似，其他国家语言也有这个问题为了统一所有文字的编码，Unicode就出现了。

Unicode通常用两个字节便是一个字符，只需把高位字节全部填写为0就可以。

python比Unicode标准发布还要早，所以python展示字符串用u‘...’表示。

ur’’’xxxxxx,

xxxx,

xxxx”’

如果在python环境中出现UnicodeDecodeError，.py保存的格式 有问题，可以在第一行添加注释：

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

告诉python解释器，用源代码读取源代码选择UTF-8格式保存。

### 8.python整数和浮点数

python支持对整数和浮点数进行四则欢和运算，和数学上的四则元算是一致的。

只能用小括号（）来提升优先级，并且可以嵌套很多层。

和数学四则元算不同的地方是，整数运算结果仍然是整数元算，浮点数运算结果仍然是浮点数，但整数和浮点数混合运算就是浮点数了。

11/4,结果为2,余数直接舍弃;

为了解决这个问题，可以把其中一个变成浮点数就可以了。

### 9.python-布尔类型

True与False

与运算 ：两个True为True

或运算：只要有一个True为True

非运算：not True，not False

and 和or运算的一条重要法则：短路计算.

1.a and b，a为False,根据运算法则，整个结果肯定为False，则返回a;如果a为True，则这个那个运算结果取决于b，则返回b。

2.a or b，a为False，运算结果取决于b，则返回b;若a为True，根据运算法则，结果必定是True，则返回a。

## List和Tuple类型

### 1.Python-创建list

python内置的一种数据类型是列表list。list是一种有序的集合，可以随时添加或删除其中的元素。

[‘Bob’，‘Michael’，‘Tracy’]元素顺序排列。

list对象可以赋值给变量。Classmates = [‘Bob’，‘Michael’，‘Tracy’]

list对象中元素并不要求必须是一种数据类型，L = [‘Michael’，100,True]

空list：empty\_list = []

### 2.按照序列访问list

C = [‘Bob’，‘Michael’，‘Tracy’]

>>> print C[0]

Bob

>>> print C[3]

IndexError:索引超出了范围。使用索引时，**千万注意不要越界。**

### 3.倒序访问list

C = [‘Bob’，‘Michael’，‘Tracy’]

>>> print C[-1]

Tracy

>>> print C[-2]

Michael

>>> print C[-4]

IndexError:索引超出了范围。使用倒序索引时，**千万注意不要越界。**

### 4.list添加元素

* list的append（）方法，追加尾部到list的尾部

C = [‘Bob’，‘Michael’，‘Tracy’]

>>> L.append（‘Mary’）

>>>print C

[‘Bob’，‘Michael’，‘Tracy’，‘Mary’]

* list的Insert（）方法，两个参数，第一个是索引号，第二个是待添加的新元素

C = [‘Bob’，‘Michael’，‘Tracy’，‘Mary’]

>>>L.Insert(0,‘Jack’)

>>>print C

[‘Jack’,‘Bob’,‘Michael’，‘Tracy’，‘Mary’]

### 5.list删除元素

* list的pop（）方法，总是删除list的最后一个元素。

>>>C = [‘Jack’,‘Bob’,‘Michael’，‘Tracy’，‘Mary’]

>>>C.pop（）

‘Mary’

>>>print C

C = [‘Jack’,‘Bob’,‘Michael’,‘Tracy’]

### 6.替换元素

>>>C = [‘Jack’,‘Bob’,‘Michael’,‘Tracy’]

>>>L[2] = ‘chanel’或者L[-1] = ‘danny’

>>>print C

[‘Jack’,‘Bob’,‘chanel’,‘Tracy’]或者[‘Jack’,‘Bob’,‘Michael’,‘danny’]

### 7.创建Tuple

tuple是另外一种有序的列表，为元组。tuple一旦建立是不能修改的。

tuple是（）来表示。

tuple中元素的索引方式和list一样，使用t[0],t[1],t[2]等访问元素，不能赋值元素。

### 8.创建單元素Tuple

包含0个元素的tuple，也就是空tuple直接用（）表示:

>>>t = ()

>>>print t

()

包含1一个元素的tuple，需要在元素后面加上逗号，来区分，否则python会把（）当成运算中优先级的表示，解释器会把该元素当成输出结果输出，产生歧义。

### 9.‘可变’的Tuple

>>>t = (‘a’,’b’,[‘A’,’B’])

>>>L = t[2]

>>>L[0] = ‘X’

>>>L[1] = ‘Y’

>>>print L  
 (‘a’,’b’,[‘X’,’Y’])

tuple的“不变”指的是每个元素指向永远不发生变化，指向一个list，就不能再指向其他的对象，但是指向的list本身的元素是可以改变的。

## 条件判断和循环

### Python-if语句

python代码的缩进原则：四个空格，不能使用tab健，更不能混合tab和空格。

if后接代码块，用冒号：表示代码块开始。

交互环境下代码留意缩进，退出缩进需要多敲一次回车。

### if-else语句

if age >=18**:**

print ‘adult’

else**:**

print ‘teenager’

### if-elif-else语句

if age >=18:

print ‘adult’

elif age>=6:

print ‘teenage’

elif age>=3:

print ‘kids’

### for循环

C = [‘Bob’，‘Michael’，‘Tracy’，‘Mary’]

for name in C:

print name

name变量是在for循环中定义的。依次取出list中的每一个元素，并把元素赋值给name，然后执行for循环体。

### while循环

while循环不会迭代list或tuple的元素，而是根据表达式判断循环是否结束。

N = 10

n = 0

while n < N:

print n

n=n+1

### break退出循环

sum = 0

x =1

while true：

sum=sum+x

x=x+1

if x>100:

break

print sum

### continue继续循环

L = [58,67,78,89,90,65]

sum = 0.0

n = 0

for x in L:

if x<60:

continue 不继续执行循环体的后续代码

sum=sum+x

n=n+1

print sum/n

### 多重循环

for x in [‘A’,’B’,’C’]:

for y in [‘1’,’2’,’3’]:

print x+y

x每循环一次，y就会循环3次即：

A1 A2 A3 B1 B2 B3 C1 C2 C3

## Dict和Set类型

### 什么是dict

dict表示“名字”-“成绩”，key-value，通过key来查找value。

用{}来表示一个dict，按照key：value格式书写，最后一个key：value的逗号省略。

dict也是一个集合len（）函数计算集合的大小：

>>>

d = {

‘amy’： 95,

‘may’ ： 90,

‘jack’ ：89

}

### 访问dict

dict使用key查找对应的value。只要key存在，dict就返回对应的value，如果可以不存在，会直接报错：keyError。

d = {

‘amy’： 95,

‘may’ ： 90,

‘Jack’ ：89

}

避免KeyError发生有两个方法:

1. 先判断key是否存在，用in操作符：

if ‘Jack’ in C：

print C[‘Jack’]

如果Jack不存在，if语句判断为false，不会执行print C[‘Jack’]，避免错误。

1. get方法，在key不存在时返回None：

>>>print d.get(‘amy’)

>>>95

>>>print d.get(‘Mary’)

>>>None

### dict的特点

1. 查找速度快。无论dict有10个元素，还是10万个元素，查找速度都是一样的。list的查找速度会随着元素增加而逐渐减少。

dict缺点是占用内存大，浪费空间。list的缺点相反，内存小，查找速度慢。

在一个dict中，key不能重复。

1. 存储的key-value序列时没有顺序，dict内部是无序的。
2. key的元素必须不可变。字符串，整数，浮点数，tuple都是不可变的，都可以作为key。list是可变的，不能作为key。

不可变仅作用于key，value无所谓。

### 更新dict

>>>d [‘july’] = 88

>>>print d

{‘amy’： 95,‘may’ ： 90,‘Jack’ ：89,’july’:88}

若key存在，则替换原来的value。

### 遍历dict

>>>for key in d:

...print key

amy

may

Jack

july

### 什么是set

set为一系列元素（），不能重复，无序（和dict的key很像）。

创建set方法是调用set（）并传入一个list，list的元素将作为set的元素：

>>>s = set（[‘A’，‘B’，‘C’，‘C’]）

>>>print s

set（[‘A’，‘C’，‘B’]）

set的存储方式是无序的，自动去掉重复的元素 。

### 访问set

in（）操作符：

>>> ‘A’ in s

true

>>>’d’ in s

False

### set的特点

唯一，无序，不变对象。

Weekdays = set（[‘MON’,’TUE’,’WED’,’THU’,’FRI’,’SAT’,’SUN’]）

X = ‘’ #用户输入的字符串

if X in weekdays：

print ‘input OK’

else

print ‘input ’

### 遍历set

>>>for week in weekdays:

...print name

MON

TUE

…

SUN

### 更新set

新元素的添加add（）方法：

>>>weekdays.add(‘NO’)

>>>Print weekdays

set（[‘MON’,’TUE’,’WED’,’NO’,’THU’,’FRI’,’SAT’,’SUN’]）

添加已存在的元素不会报错，但不加进去。

元素的删除remove（）方法：

>>>weekdays.remove（‘NO’）

>>>print weekdays

set（[‘MON’,’TUE’,’WED’,’THU’,’FRI’,’SAT’,’SUN’]）

删除不存在的元素报错KeyError。

可以直接添加元素add（），删除元素remove（）需要判断。

## 函数

### 什么是函数

函数本身只需要写一次，可以无数次的调用；

抽象

函数是最基本的一种代码抽象的方法。python不仅可以灵活定义函数，本身也内置了许多函数，可以直接调动。

### 调用函数

要调用函数，必须要知道函数的名称和参数，可官网查找文档，也可以在交互命令行help（abs）查看abs的帮助信息；

调用函数时参数数量不对或参数类型不被接受都会报TypeError；

内置常用

### 编写函数

定义一个函数要使用def语句+函数名（参数）：

缩进块编写函数体

返回值用return语句返回（无return会返回None，return none可简称none），一旦执行到return，函数就执行完毕并返回结果。

### 函数返回多值

用import引用函数，返回多值其实是一个假象，python返回的其实是一个tuple(),返回一个tuple可以省略括号，多个变量可以同时接受同一个tuple，按位置赋给对应的值。

Import math

def move（x，y，step，angle）：

nx = x+step\*math.cos(angle)

ny = y+step\*math.sin(angle)

return nx,ny

### 递归函数

函数内部调用函数本身，就是递归函数。定义简单，逻辑清晰。所有的递归都能通过for循环实现，但是不如递归清晰。

fact（n）=n！=1\*2\*3×...\*n=n\*fact(n-1),n==1除外。

注意防止栈溢出。函数调用是通过栈这种数据结构实现的，调用一个函数，栈就增加一层栈帧，每当函数返回，栈就减少一个栈帧。大小不是无限的，递归调用过多就会溢出。

1. 定义默认参数

简化调用；函数参数按照从左到右的顺序匹配，默认函数只能在必须函数的后面。

Def power（x，n=2）：

s = 1

while n>0:

n = n -1

s = s\*x

return s

>>>power(5)

25

### 定义可变参数

def 函数名（\*参数）：

把传入的一组参数组装成一个tuple传递给可变参数，直接把参数看成一个tuple即可。

## 切片

### 对list进行切片

L[0:3] 提取从索引0开始到索引3(不包括索引3)，3个元素。

L[:3] 从索引0开始可省略；

L[1:3] 从索引1开始，2个元素；

L[:] 复制同一个list

L[：：3] 每3个元素取一个，每隔两个取一个。

list与tuple执行切片操作完全相同，只是切片的结果变成了tuple。

### 倒序切片

倒序切片包含起索引，不包括结束索引。[起索引，结束索引)

L = [‘A’，‘B’，‘C’，‘D’]

L[-1] [‘D’]

L[-4:-1]

L[-2:]

L[:-2]

L[-3:-1]

L[-4：-1：2]

list与tuple执行切片操作完全相同，只是切片的结果变成了tuple。

### 对字符串切片

‘ABCDEFG’[:3]

‘ABC’

‘ABCDEFG’[-3:]

‘EFG’

‘ABCDEFG’[::2]

‘ACEG’

## 迭代

### 什么是迭代

在python中，迭代就是指的for循环。

for…in

python的for循环不仅可以用在list或tuple上，还可以用在任何可迭代对象（有序集合：list/tuple/str/Unicode、无序集合：set、无序集合并具有key-value：dict）上。

迭代与按下标访问数组最大的不同：

后者是一种具体的迭代实现方式，前者只关心结果，根本不关心迭代内部如何实现的。

### 索引迭代

迭代是取元素本身，非元素索引。

enumerate（）函数

L = [‘A’，‘B’，‘C’，‘D’]

for index，name in enumerate（L）：

...print index，‘-’，name

0 -A

1 -B

2 -C

3 -D

enumerate（）

函数 把[‘A’，‘B’，‘C’，‘D’]

变成了类似[（0，‘A’），（1，‘B’），,（2，‘C’），（3，‘D’）]

迭代的每一个元素都是一个tuple：

for t in enumerate（L）：

index = t[0]

name = t[1]  
print index,’-’,name

每个tuple元素都包含两个元素for循环可以为：

for index，name in enumerate（L）：

print index，‘-’，name

### 迭代dict的value

dict对对象可用for循环迭代，可以得到dict的key。

迭代dict对象的value，可用dict对象的value（）方法，转换dict成一个包含所有value的list。

d = {

‘amy’： 95,

‘may’ ： 90,

‘Jack’ ：89

}

print d.value()  
 #[95,90,89]

for v in d.value()

print v

#95

#90

#89

迭代效果完全一样的还有itervalues（），两种方法的区别在于：

itervalues（）不是转换，而是直接在dict对象中取出value，节省了生成list所需的内存； values（）需要把dict转换成了包含value的list；

一个对象可以迭代，就可以用for循环进行迭代（不只是list/tuple/dict/set/str/Unicode等）。迭代是一个抽象的数据操作，对迭代对象内部的数据没有任何要求。

### 迭代dict的key和value

item（）方法

d = {

‘amy’： 95,

‘may’ ： 90,

‘Jack’ ：89

}

>>>print d.item()

[(‘amy’,95),(‘may’ , 90),( ‘Jack’ ,89)] #转换成了一个包含多个tuple的list，对list进行迭代：

for key，value in d.item():

print key,’:’,value

’ amy’： 95,

‘may’ ： 90,

‘Jack’ ：89

对应的有iteritem（）方法，不转换成list，而是不断给出tuple，不占额外的内存。

## 列表生成式

### 生成列表

>>>[x\*x for x in range（1,11）] #python特有的列表生成式

### 复杂表达式

此处涉及到html格式，需要输入的太多，暂时空白。

### 条件过滤

>>>[x\*x for x in range(1,11) if x%2 == 0]

### 多层表达式

>>>[m+n for m in ‘ABC’ for n in ‘123’]

翻译成循环代码为：

L = []

for m in ‘ABC’

for n in ‘123’

L.append(m + n)

## 函数式编程

### 1.函数式编程简介

functional，编程范式

函数 =/函数范式

python -->函数式

c语言 --->函数

汇编语言

计算机硬件 --->指令

函数是编程特点：

函数是编程，把计算视为函数而非指令；

纯函数式编程不需要变量，没有副作用，测试简单；

支持高阶函数，代码简洁。

python支持的函数式编程：

不是纯函数是编程：允许有变量

支持高阶函数：函数也可以作为变量介入

支持闭包：有了闭包就能返回函数

有限度的支持匿名函数

### 2.高阶函数

变量可以指向函数

函数名其实就是指向函数的变量

高阶函数：能接收函数做参数的函数

变量可以指向函数

函数的参数可以接收变量

一个函数可以接收另一个函数作为参数

能接收函数作为参数的函数就是高阶函数

### 3.把函数作为参数

def add（x，y，f）：

return f（x）+f（y）

如果传入abs作为参数f的值：

add（-5,9，abs）#f传入其他的函数就可以得到不同的返回值。

函数执行的代码相当于：abs（-5）+abs（9）

### 4.map（）函数

map（）是python内置的一个高阶函数，可以接收一个函数f和一个list，通过把函数f依次作用在list的每一个元素上，得到一个新的list返回;

不改变原来的元素，只是新生成了一个新元素；

map（）不仅可以处理只包含数值的list，事实上可以处理包含

Def f（x）：

return x\*x

print map（f，[1,2,3,4,5,6,7,8]）

[1,4,9,16,25,36,49,64]

### 5.reduce（）函数

也是python内置的一个高阶函数，也是接收一个函数f，一个list，但是传入的函数f必须接收两个参数,对list的每个元素反复调用函数f，并返回最终结果值。

Def f（x，y）：

return f（x）+f（y）

reduce（f，[1,3,5,7,9]）

reduce()函数还可以接收第三个可选参数，作为计算的初始值。

reduce（f，[1,3,5,7,9],100)

### 6.filter（）函数

自动过滤掉不符合条件的元素，返回符合条件的元素组成的list；

判断奇数的函数：

def is\_odd(x):

return x%2==1

filter(is\_odd,[1,4,6,7,9,12,17])

[1,7,9,17]

删除None或者空字符串：

def is\_not\_empty(s):

return s and len(s.strip()) >0

filter(is\_not\_empty,[‘test’,’’,’none’,‘ ’，‘end’])

[‘test’,’none’,’end’]

s.strip(rm):删除s字符串中开头、结尾处的rm序列的字符。

当rm为空时，默认删除空白字符（\n，\r,\t, 空格）

a = ‘ 123 ’

a.strip()

‘123’

### 7.自定义排序函数

sorted（）对list进行排序。

Def reversed\_cmp(x，y):

if x>y:

return -1

if x<y

return 1

return 0

sorted([34,56,67,2,65,],reversed\_cmp)

[67,65,56,34,2]

sorted()函数也可以比较字符串排序，字符串默认按照ASCII大小比较。

### 8.返回函数

def f():

print ‘call f()’

def g():

print ‘call g()’

return g

>>>x = f() #调用f（）

call f()

>>>x #变量x是f（）返回的函数

<function g at …..>

>>>x() #x指向函数，因此可以调用  
call g() #调用x（）就是执行g（）函数定义的代码

返回函数可以把一些计算延后执行。

### 9.闭包

内层函数使用了外层函数的变量（参数也属于变量），然后返回内层函数的情况，成为闭包；

确保引用的局部变量在函数返回后不能变，返回函数不要引用任何循环变量，或者后续会发生变化的变量；

### 10.匿名函数

lambda关键字表示匿名函数，冒号前面的x表示函数参数；

匿名函数只能有一个表达式，不写return，返回值就是该表达式的结果。

>>>map(lambda x:x\*x,[1,2,3,4,5,6,7,8])

>>>sorted([1,3,5,7,9],lambda x,y:-cmp(x,y))

>>>myabs = lambda x:-x if x<0 else x

>>>myabs(-1)

>>>myabs(1)

### 11.decorator装饰器

@decorator

需要在函数运行的时候动态增加功能，又不想更改代码本身，就需要装饰器。

@语法是简化装饰器的调用

@new\_fn

def f1（x):

return x\*2

等于

def f1（x):

return x\*2

f1 = new\_fn(f1)

装饰器作用：

极大简化代码，避免函数编写重复

打印日志：@log

检测性能：@performance

数据库事务：@transaction

URL路由：@post（`/register）

### 12.编写无参数decorator

def log(f):

def fn(x):

print ‘call ’+ f.\_name\_ +‘()...’

return f(x)

return fn

对于参数不是一个的函数，就会报错；

可以利用python的\*args和\*\*kw,保证任何个数的参数都能正常调用。

def log(f):

def fn(\*args，\*\*kw):

print ‘call ’+ f.\_name\_ +‘()...’

return f(\*args，\*\*kw)

return fn

### 13.编写带参数decorator

### 14.完善decorator

### 15.偏函数

## 模块

### 1.模块和包的概念

在文件系统中，包就是文件夹，包可以有多级，模块就是xxx.py；

与普通目录的区别：包下面必须有\_init\_.py,每层都得有，即使是空文件，都得有\_init\_.py

### 2.导入模块

import math

math.pi

math.power(2,0.5)

from math imort sin,log

import math,logging

math.log(10)

logging.log(10,’something’)

from math import log

from logging import log as logger

print log(10)

logger(10,’something from logging’)

### 3.动态导入模块

导入的模块不存在，报错ImportError

两个模块提供了相同的功能，例如stringIO和cstringIO都提供了stringIO功能。

python是动态语言，代码执行比较慢，StringIO是python代码编写，cstringIO部分函数是c语言编写，大大提高执行速度。

利用importError错误在python中动态导入模块：

try：

from cstringIO import StringIO

except ImportError：

from StringIO import stringIO

try的作用是捕捉错误，并在捕捉到指定错误时执行except 语句。

### 4.使用\_future\_

要试用某一新的特征时，可以试用\_future\_模块的某些功能实现。

在python2.7中引用python3.X的除法原则，打入\_future\_的division：

from\_future\_ import division

### 5.安装第三方模块

--easy install

--pip(推荐，已经内置到python2.7)

安装python过程中勾选上pip和add...to path即可。

例如安装web.py的第三方模块：  
cmd-pip install web.py-python互动环境-import web

## 面向对象编程基础

### 1.面向对象编程

一种程序设计范式

把程序看做不同对象的相互调用

对现实世界建立对象模块  
基本思想：

类和实例

类（class）用于抽象类型

实例根据类的定义被创建出来

例如，人是一种类，那么小军，小明等就是一个个实例

class Person：

pass

xiaoming = person()

xiaojun = person()

数据封装：

class Person():

def \_init\_（self，name）：

self.name = name

p1 = person(‘xiaojun’)

p2 = person(‘xiaoming’)

### 2.定义类并创建实例

类的一个大写字母开头；

class Person(object：该类是从哪个类继承)；

创建实例以类名+（），例如：

xiaojun = Person（）

xiaoming = Person（）

### 3.创建实例属性

通过Person类建立xiaoming、xiaojun的实例，但是实例之间还会有很多不同的属性：

xiaoming = Person（）

xiaoming.name = ‘xiaoming’  
xiaoming.gender = ‘Male’

xiaojun = Person()

xiaojun.grade = xiaoming.grade +1

### 4.初始化实例属性

定义Person类中添加一个\_init\_()方法，在创建实例时，\_init\_()方法被自动调用，可以为每个实例统一加上属性：

class Person():

def \_init\_（self，name，gender，birth）：

self.name = name

self.gender = gender

self.birth = birth

\_init\_（）方法第一个参数必须是self；

创建实例时，需要提供除self以外的参数：

xiaoming = Person(‘xiaoming’,‘Male’,’1990-1-1’)

xiaojun = Person（‘xiaojun’，‘Female’‘1993-3-3’）

注意：第一个参数不能把self忘记，会找到创建失败或者运行不正常，因为第一个参数name被python解释器传入了实例的引用，从未导致整个方法的调用参数位置错位。

### 5.访问限制

一个实例可以绑定多个属性，但是属性不希望被外部访问。如果一个属性由双下划线开头，就无法被外部访问。

Class Person（object）：

def \_init\_(self,name):

self.name = name

self.title = ‘Mr’

self.\_\_job = ‘Student’

p = Person(‘Bob’)

print p.name

# =>Bob

print p.title

**# =>Mr**

print p.\_\_job

# =>Error

以\_xxx\_形式定义的属性，都为特殊属性，很多预定义的属性都使用此形式定义，通常情况不会把普通属性用\_xxx\_定义；  
以单下划线定义的属性，虽然可以被外部访问，但是按照习惯，不应该被外部调用。

### 6.创建类属性

实例属性每个实例各自拥有，个自独立，而类属性有且只有一份；

类属性是直接绑定在类上的，访问类属性不需要创建实例，就可以直接访问，如Person.address;

实例不需要单独创建类上的属性，直接就可以访问所属类的属性,如xiaoming.address;

python是动态语言，可以之间进行修改和添加，如Person.address = ‘China’,Person类address改变了，那实例访问类中的属性也发生了变化。

### 7.类属性和实例属性名字冲突问题

当实例属性和类属性重名时，实例属性的优先级高，直接屏蔽对类属性的访问；

不要通过改变实例属性改变类属性的内容，也改变不了。

### 8.定义实例方法

实例方法是在类中定义的函数，第一个参数一定是self，指向该方法的函数本身，其他参数和其他函数是一样的；

Class Person（object）：

def \_init\_(self,name):

self.name = name

def get\_name（self）：

return self.\_name

p1 = Person(‘Bob’)

print p1.get\_name() #self不需要显性传入

外部需要访问私有属性就可以通过方法调用获得，这种数据封装的形式除了保护内部数据一致性外，还可以简化外部调用的难度。

### 9.方法也是属性

在class类定义的实例方法也是属性，实际上也是一个函数对象；

types.MethodType()把一个函数变成一个方法；

types.MethodType(函数名，实例名，类名)

### 10.定义类方法

和属性类似，方法分类方法和实例方法；

Class中定义的全都是实力方法，实例方法第一个参数self是实例本身；

Class中定义类方法需要@classmethod，该方法绑定到Person类上，而不是类的实例:

Class Person(object):

@classmethod

Def how many(cls)

Return cls.count

类方法的第一个参数将传入类本身，通常参数名命名为cls，cls.count相当于Person.count;

## 类的继承

### 什么是继承

新类不需要重新编写；

从现有的类继承，自动拥有现有类的所有功能；

只需要编写现有类缺少的新功能。

优点：复用已有的代码，自动拥有现有类所有功能，只需要编写缺少的新功能。

Is关系

Has关系

### 继承一个类

Class Person(object):

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

Self.name = name

Self.gender = gender

Class Student(Person):  
 def \_\_init\_(self,name,gender,score):  
 super(Student,self).\_init\_(name,gender)

Self.score = score

super(Student,self).\_init\_(name,gender)去初始化父类，否则继承Person的Student将没有name和gender。

super(Student,self)返回当前类继承的父类，即Person，然后调用.\_init\_（）方法，self参数已经在super（）中传入，.\_init\_（）隐式传递，不需要再写入。

### 判断类型

Isinstance（变量名（实例也是变量），类名）可以判断变量的类型，TURE或False返回。

### 多态

方法调用将作用在x的实际类型上。

动态语言调用实例方法，不检查类型，只要方法存在，参数正确，就可以调用。

### 多重继承

允许从多个父类继承，成为多重继承；

目的是从多个继承树中分别继承出子类，以便组合功能使用。

### 获取对象信息

Type（）函数获取变量的类型，返回一个type对象；

Dir（）获取变量的所有属性；

Getattr（）获取变量的属性getattr（s，‘name’）； getattr（s，‘name’，20）变量不存在就返回20

Settattr（）设置变量的属性；settattr（s，name，‘Bob’）

## 定制类

### 1.什么是特殊方法

任何数据类型都有一个特殊方法：

\_\_str\_\_()

用于print的\_\_str\_\_

用于len的\_\_len\_\_

用于cmp的\_\_cmp\_\_

特殊方法定义在class中

不需要直接调用

某些函数或者操作符会调用对应的特殊方法

只需要编写需要的特殊方法

有关联性的特殊方法都必须实现

\_\_gettattr\_\_()必须也要实现下面两个方法

\_\_settattr\_\_()

\_\_deltattr\_\_()

### 2.\_str\_和\_repr\_

Def \_\_str\_\_():

Return ‘(Person:%s,%s)’ % (self.name,self.gender)

\_\_str\_\_()用于显示给用户；

\_\_repr\_\_()用于显示给开发人员。

也可以：\_\_repr\_\_() = \_\_str\_\_()

### 3.\_cmp\_

### 4.\_len\_

### 5.数学运算

### 6.类型转换

### 7.@property

### 8.\_slots\_

### 9.\_call\_