1. Python是一种跨平台、开源、免费的高级动态编程语言。

对

1. Python 3.x完全兼容Python 2.x。

错

Python 3.x和Python 2.x有较大差异,(依赖包名称变换,print在3中必须要有括号)

1. 已知 x = 3，那么赋值语句 x = 'abcedfg' 是无法正常执行的。

错

可以执行,执行完之后x的地址改变

1. Python变量使用前必须先声明，并且一旦声明就不能再当前作用域内改变其类型。

错

Python是弱类型语言,使用前不需要声明

1. Python采用的是基于值得自动内存管理方式。

对

Python中的变量并不直接存储值，而是存储了值的内存地址或者引用，假如为不同变量赋值为相同值，这个值在内存中只有一份，多个变量指向同一块内存地址。

例如：

x=100

y=x

print(id(x),id(y))

out:1366822448 1366822448

1. Python不允许使用关键字作为变量名，允许使用内置函数名作为变量名，但这会改变函数名的含义。

对

这也是为什么不建议使用内置函数名作为变量名的原因

1. 在Python中可以使用if作为变量名。

错

Python的关键字不能作为变量名,一共有: **['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. 在Python 3.x中可以使用中文作为变量名。

对

别问,问就是中国no.1

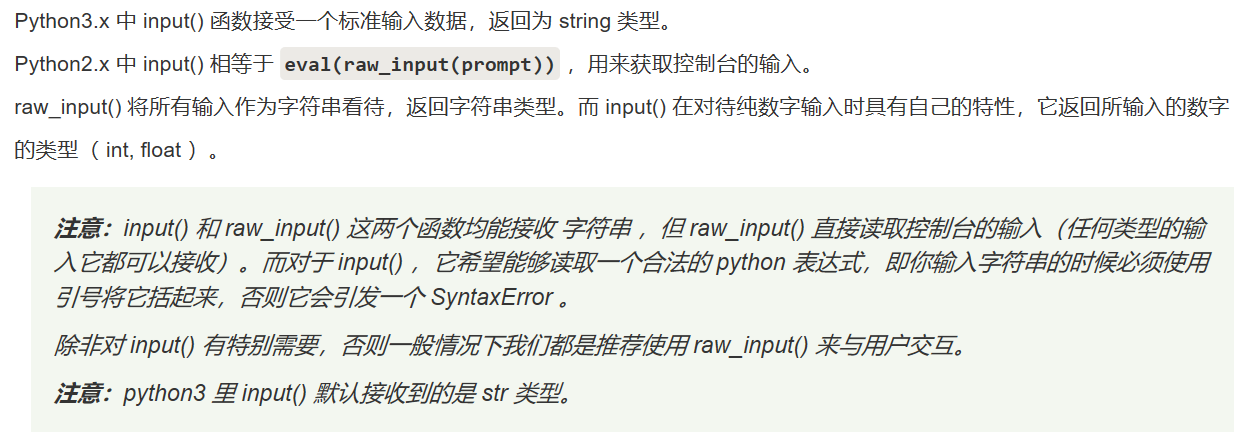
1. Python变量名必须以字母或下划线开头，并且区分字母大小写。

对

感觉有点矛盾,但是有可能代指的所有版本而不是单一3.0

1. 不管输入什么，Python 3.x中input()函数的返回值总是字符串。

对

如图就是原因,因注意不同版本的差异,比如这里若是python2.0的返回值就可以不是string

1. Python使用缩进来体现代码之间的逻辑关系。

对

Python中，是通过代码的缩进，来决定代码的逻辑的。

通俗的说，Python中的代码的缩进，不是为了好看，而是觉得代码的含义，上下行代码之间的关系。

缩进弄错了，就会导致程序出错，执行结果变成不是你想要的了。

1. 在Python中可以使用 id 作为变量名，尽管不建议这样做。

对

**id()** 函数用于获取对象的内存地址。

1. Python关键字不可以作为变量名。

对

1. 执行语句 from math import sin 之后，可以直接使用 sin() 函数，例如 sin(3)。

对

别问,问就是憨憨

1. Python变量名区分大小写，所以student和Student不是同一个变量。

对

1. 如果只需要math模块中的sin()函数，建议使用from math import sin来导入，而不要使用import math导入整个模块。

对

买菜开兰博基尼?

1. 在Python 3.x中，使用内置函数input()接收用户输入时，不论用户输入的什么格式，一律按字符串进行返回。

对

1. Python支持使用字典的“键”作为下标来访问字典中的值。

对

同样,列表和元组也可以通过下标的方式访问字典中的元素

1. 列表可以作为字典的“键”。

错

不行,无法使用一个列表作为一个字典的键

1. 元组可以作为字典的“键”。

对

**对象能不能作为字典的key，就取决于其有没有\_\_hash\_\_方法。所以所有python自带类型中，除了list、dict、set和内部至少带有上述三种类型之一的tuple之外，其余的对象都能当key。**字典的键可以是任意不可变类型，需要注意的是tuple元组作为键时，其中不能以任何方式包含可变对象。

1. 字典的“键”必须是不可变的。

对

1. 已知x为非空列表，那么表达式 sorted(x, reverse=True) == list(reversed(x)) 的值一定是True。

错





Sorted默认升序排列



Reversed是反转

1. 已知x为非空列表，那么x.sort(reverse=True)和x.reverse()的作用是等价的。

错

1. Python集合中的元素不允许重复。

对

1. Python字典中的“键”不允许重复。

对

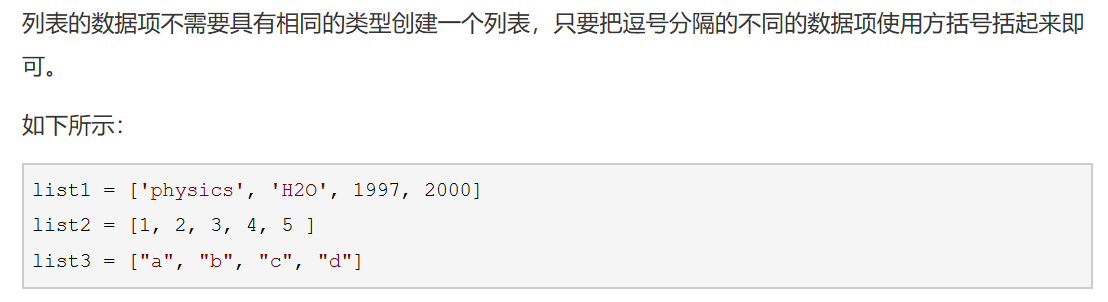
1. Python字典中的“值”不允许重复。

错

1. Python字典中的“键”可以是元组。

对

1. Python列表中所有元素必须为相同类型的数据。

错

1. Python列表、元组、字符串都属于有序序列。

对

且都支持双向索引

1. 列表对象的append()方法属于原地操作，用于在列表尾部追加一个元素。

对



1. 使用Python列表的方法insert()为列表插入元素时会改变列表中插入位置之后元素的索引。

对

1. 假设x为列表对象，那么x.pop()和x.pop(-1)的作用是一样的。

对



1. 使用del命令或者列表对象的remove()方法删除列表中元素时会影响列表中部分元素的索引。

对

1. 已知列表 x = [1, 2, 3]，那么执行语句 x = 3 之后，变量x的地址不变。

错

Python根据值变地址

1. 使用列表对象的remove()方法可以删除列表中首次出现的指定元素，如果列中不存在要删除的指定元素则抛出异常。

对

1. 元组是不可变的，不支持列表对象的inset()、remove()等方法，也不支持del命令删除其中的元素，但可以使用del命令删除整个元组对象。

对

1. Python字典和集合属于无序序列。

对

对照29 Python列表、元组、字符串都属于有序序列。则:简单知道字典和集合无序,而列表元组,字符串有序

1. 无法删除集合中指定位置的元素，只能删除特定值的元素。

对

只有用remove直接删除指定元素

1. 元组的访问速度比列表要快一些，如果定义了一系列常量值，并且主要用途仅仅是对其进行遍而不需要进行任何修改，建议使用元组而不使用列表。

对

1. 当以指定“键”为下标给字典对象赋值时，若该“键”存在则表示修改该“键”对应的“值”，若不存在则表示为字典对象添加一个新的“键-值对”。

对

1. 只能通过切片访问列表中的元素，不能使用切片修改列表中的元素。

错

列表元素的增删改查以及元素替换都可以通过切片实现,并且不影响列表对象内存地址

切片是python序列的重要操作之一,可用于列表,元组,字符串,range对象

1. 只能通过切片访问元组中的元素，不能使用切片修改元组中的元素。

对

元组不可修改

1. Python集合不支持使用下标访问其中的元素。

对

集合是无序序列

1. 删除列表中重复元素最简单的方法是将其转换为集合后再重新转换为列表。

对

1. 已知列表x中包含超过5个以上的元素，那么语句 x = x[:5]+x[5:] 的作用是将列表x中的元素循环左移5位。

错

1. 对于生成器对象x = (3 for i in range(5))，连续两次执行list(x)的结果是一样的。

错

生成器对象只能访问一次，若要再次访问，必须重新创建生成器对象。

1. 对于大量列表的连接，extend()方法比运算符+具有更高的效率。

对

1. 列表对象的extend()方法属于原地操作，调用前后列表对象的地址不变。

对

1. 对于数字n，如果表达式 0 not in [n%d for d in range(2, n)] 的值为True则说明n是素数。

对

1. 创建只包含一个元素的元组时，必须在元素后面加一个逗号，例如(3,)。

对

1. 表达式 list('[1, 2, 3]') 的值是[1, 2, 3]。

错

拆分开来了

结果是['[', '1', ',', '2', ',', '3', ']']

1. 同一个列表对象中的元素类型可以各不相同。

对

1. 已知x为非空列表，那么执行语句x[0] = 3之后，列表对象x的内存地址不变。

对

1. 列表可以作为集合的元素。

错

1. 集合可以作为列表的元素。

对

1. 元组可以作为集合的元素。

对

1. 集合可以作为元组的元素。

对

1. 字典可以作为集合的元素。

错

1. 列表对象的pop()方法默认删除并返回最后一个元素，如果列表已空则抛出异常。

对

1. 表达式 {1, 2} \* 2 的值为 {1, 2, 1, 2}。

错

Run了是error，集合不能重复

1. 假设re模块已成功导入，并且有 pattern = re.compile('^'+'\.'.join([r'\d{1,3}' for i in range(4)])+'$')，那么表达式 pattern.match('192.168.1.103') 的值为None。

错

1. 已知x = (1, 2, 3, 4)，那么执行x[0] = 5之后，x的值为(5, 2, 3, 4)。

错，元组不能改变

元组是(),列表,字典是[],集合是{}

1. 已知x = 3，那么执行x += 6语句前后x的内存地址是不变的。

错

1. 内置函数len()返回指定序列的元素个数，适用于列表、元组、字符串、字典、集合以及range、zip等迭代对象。

对

1. 已知x和y是两个等长的整数列表，那么表达式sum((i\*j for i, j in zip(x, y)))的作用是计算这两个列表所表示的向量的内积。

对



1. 已知x和y是两个等长的整数列表，那么表达式[i+j for i,j in zip(x,y)]的作用时计算这两个列表所表示的向量的和。

对

1. 表达式int('1'\*64, 2)与sum(2\*\*i for i in range(64))的计算结果是一样的，但是前者更快一些。

对

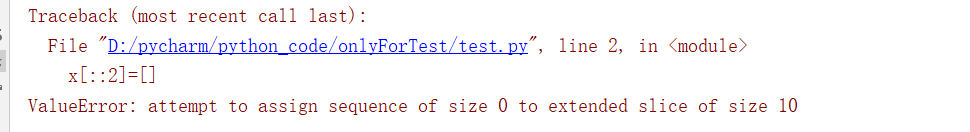


1. 已知x = list(range(20))，那么语句del x[::2]可以正常执行。

对

1. 已知x = list(range(20))，那么语句x[::2] = []可以正常执行。

错



1. 已知x = list(range(20))，那么语句print(x[100:200])无法正常执行。

错

返回空列表

1. 已知x是个列表对象，那么执行语句y = x之后，对y所做的任何操作都会同样作用到x上。

对

1. 已知x是个列表对象，那么执行语句y = x[:]之后，对y所做的任何操作都会同样作用到x上。

错

71.72中要明白,71是获取到x的地址,72是获取x的值

1. 表达式(i\*\*2 for i in range(100))的结果是个元组。

错



生成器推导式的结果是一个生成器对象，既不是列表，也不是元组。

1. 在Python中元组的值是不可变的，因此，已知x = ([1], [2])，那么语句x[0].append(3)是无法正常执行的。

错

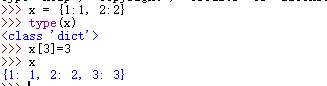
1. Python内置的字典dict中元素是按添加的顺序依次进行存储的。

错

字典的储存是无序的与列表不同

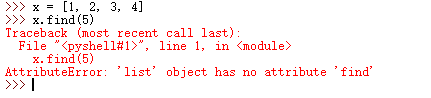
1. 已知x = {1:1, 2:2}，那么语句x[3] =3无法正常执行。

错



1. 已知列表x = [1, 2, 3, 4]，那么表达式x.find(5)的值应为-1。

错，find是字符串方法，不是列表方法





Find是字符串的

1. 列表对象的排序方法sort()只能按元素从小到大排列，不支持别的排序方式。

可以从大到小

1. 已知x是一个列表，那么x = x[3:] + x[:3]可以实现把列表x中的所有元素循环左移3位。

对

1. 如果仅仅是用于控制循环次数，那么使用for i in range(20)和for i in range(20, 40)的作用是等价的。

对

1. 在循环中continue语句的作用是跳出当前循环。

错

1. 在编写多层循环时，为了提高运行效率，应尽量减少内循环中不必要的计算。

对

1. 带有else子句的循环如果因为执行了break语句而退出的话，则会执行else子句中的代码。

错

1. 对于带有else子句的循环语句，如果是因为循环条件表达式不成立而自然结束循环，则执行else子句中的代码。

对

1. 在条件表达式中不允许使用赋值运算符“=”，会提示语法错误。

对

1. Python运算符%不仅可以用来求余数，还可以用来格式化字符串。

对

print("I'm %s. I'm %d year old" % ('Vamei', 99))

上面的例子中，

"I'm %s. I'm %d year old" 为我们的模板。%s为第一个格式符，表示一个字符串。%d为第二个格式符，表示一个整数。('Vamei', 99)的两个元素'Vamei'和99为替换%s和%d的真实值。   
在模板和tuple之间，有一个%号分隔，它代表了格式化操作。

1. 如果需要连接大量字符串成为一个字符串，那么使用字符串对象的join()方法比运算符+具有更高的效率。

对

Python join() 方法用于将序列中的元素以指定的字符连接生成一个新的字符串。

1. 正则表达式模块re的match()方法是从字符串的开始匹配特定模式，而search()方法是在整个字符串中寻找模式，这两个方法如果匹配成功则返回match对象，匹配失败则返回空值None。

对

1. 已知x为非空字符串，那么表达式 ''.join(x.split()) == x 的值一定为True。

错

join()是string模块提供的一个方法，用于将序列中的元素以指定的字符连接生成一个新的字符串。因此它需要两个参数，一个是words，另一个是分隔符seq ,string.join(words,seq)

split()：拆分字符串。通过指定分隔符对字符串进行切片，并返回分割后的字符串列表list,语法str.split(seg="",num=string.count(seg))[n]

分隔符seg默认为空格.

num表示分割的次数,默认为seg的个数,可以形成num+1个切片.

[n]表示选取第n个切片

1. 已知x为非空字符串，那么表达式 ','.join(x.split(',')) == x 的值一定为True。

对

1. 当作为条件表达式时，[]与None等价。

对

1. 当作为条件表达式时，{}与None等价。

对

1. 当作为条件表达式时，空值、空字符串、空列表、空元组、空字典、空集合、空迭代对象以及任意形式的数字0都等价于False。

对

1. 正则表达式对象的match()方法可以在字符串的指定位置开始进行指定模式的匹配。

对

re.match 尝试从字符串的起始位置匹配一个模式，如果不是起始位置匹配成功的话，match()就返回none。

1. 使用正则表达式对字符串进行分割时，可以指定多个分隔符，而字符串对象的split()方法无法做到这一点。

对

re.split(pattern, string, [maxsplit], [flags])

1

pattern：表示模式字符串，由要匹配的正则表达式转换而来。

string：表示要匹配的字符串。

maxsplit：可选参数，表示最大的拆分次数。

flags：可选参数表示标志位，用于控制匹配方式，如是否区分子母大小写

1. 正则表达式元字符“^”一般用来表示从字符串开始处进行匹配，用在一对方括号中的时候则表示反向匹配，不匹配方括号中的字符。

对

匹配输入字符串的开始位置。如果设置了 RegExp 对象的 Multiline 属性，^ 也匹配 '\n' 或 '\r' 之后的位置。

1. 正则表达式元字符“\s”用来匹配任意空白字符。

对

匹配任何空白字符，包括空格、制表符、换页符等等。等价于 [ \f\n\r\t\v]。

1. 正则表达式 元字符“\d”用来匹配任意数字字符。

对

匹配一个数字字符。等价于 [0-9]。

1. 已知x和y是两个字符串，那么表达式sum((1 for i,j in zip(x,y) if i==j))可以用来计算两个字符串中对应位置字符相等的个数。

对

**zip()** 函数用于将可迭代的对象作为参数，将对象中对应的元素打包成一个个元组，然后返回由这些元组组成的列表。

如果各个迭代器的元素个数不一致，则返回列表长度与最短的对象相同，利用 \* 号操作符，可以将元组解压为列表。

1. 已知x = 'hellow world.'.encode()，那么表达式x.decode('gbk')的值为'hellow world.'。

## 对

## 描述

decode() 方法以指定的编码格式解码 bytes 对象。默认编码为 'utf-8'。

**语法**

decode()方法语法：

bytes.decode(encoding="utf-8", errors="strict")

1. 正则表达式'^http'只能匹配所有以'http'开头的字符串。

对

|  |  |
| --- | --- |
| ^ | 匹配字符串的开头 |

1. 正则表达式'^\d{18}|\d{15}$'只能检查给定字符串是否为18位或15位数字字符，并不能保证一定是合法的身份证号。

对

1. 正则表达式'[^abc]'可以一个匹配任意除'a'、'b'、'c'之外的字符。

对

1. 正则表达式'python|perl'或'p(ython|erl)'都可以匹配'python'或'perl'。

对

1. 调用函数时，在实参前面加一个型号\*表示序列解包。

对

1. 函数是代码复用的一种方式。

对

1. 定义函数时，即使该函数不需要接收任何参数，也必须保留一对空的圆括号来表示这是一个函数。

对

1. 编写函数时，一般建议先对参数进行合法性检查，然后再编写正常的功能代码。

对

1. 一个函数如果带有默认值参数，那么必须所有参数都设置默认值。

错

1. 定义Python函数时必须指定函数返回值类型。

错

1. 定义Python函数时，如果函数中没有return语句，则默认返回空值None。

对

1. 函数中必须包含return语句。

错

1. 不同作用域中的同名变量之间互相不影响，也就是说，在不同的作用域内可以定义同名的变量。

对

1. 函数内部定义的局部变量当函数调用结束后被自动删除。

对

1. 在函数内部，既可以使用global来声明使用外部全局变量，也可以使用global直接定义全局变量。

对

根据一个变量起作用的范围不同，可以将变量分为全局变量与局部变量

全局变量在全局范围内起作用，局部变量在一个函数内部起作用

一般全局变量是不可以更改的，但是在python中，可以使用global进行更改

更改后的全局变量将以新的值在全局范围内继续起作用

1. 在函数内部没有办法定义全局变量。

错

1. 在函数内部直接修改形参的值并不影响外部实参的值。

对

1. 调用带有默认值参数的函数时，不能为默认值参数传递任何值，必须使用函数定义时设置的默认值。

错

一般说来定义多少个形参，就需要传入多少个实参。在一些特殊情况下，函数虽然定义了形参，但在调用函数时可以不用传入实参，这就是默认参数的作用。默认参数的意思就是给函数的形参设置一个默认值，如果在调用函数时没有传入实参，那么这个默认值将会作为实参传递给函数。默认参数可以简化函数的调用，调用者不需要传入过多的实参。

给函数设置默认参数时要遵循该参数具有共性和不变属性的规则，在特殊情况下可以用传入的实参代替默认值。例如在一个计算银行利息的函数中，声明的参数有利率、本金和存款存期，在这三个参数中利率一般是不变的，它具有共性和不变属性，可以设置为默认参数，当因特殊情况利率发生变化时，可以传入实参来代替默认值。

1. 在同一个作用域内，局部变量会隐藏同名的全局变量。

对

1. 形参可以看做是函数内部的局部变量，函数运行结束之后形参就不可访问了。

对

1. 在函数内部没有任何声明的情况下直接为某个变量赋值，这个变量一定是函数内部的局部变量。

对

1. 在Python中定义函数时不需要声明函数参数的类型。

对

1. 在Python中定义函数时不需要声明函数的返回值类型。

对

1. 在函数中没有任何办法可以通过形参来影响实参的值。

错

在函数内定义

1. 已知 x = 3，那么执行语句 x+=6 之后，x的内存地址不变。

错

会改变

1. 在定义函数时，某个参数名字前面带有一个\*符号表示可变长度参数，可以接收任意多个普通实参并存放于一个元组之中。

对

### 用来将任意个数的参数导入到 Python 函数中，单星号（\*）：\*agrs

将所以参数以元组(tuple)的形式导入。

1. 在定义函数时，某个参数名字前面带有两个\*符号表示可变长度参数，可以接收任意多个关键参数并将其存放于一个字典之中。

对

双星号（\*\*）将参数以字典的形式导入:

1. 定义函数时，带有默认值的参数必须出现在参数列表的最右端，任何一个带有默认值的参数右边不允许出现没有默认值的参数。

对

1. 在调用函数时，可以通过关键参数的形式进行传值，从而避免必须记住函数形参顺序的麻烦。

对

>>> hello\_2(greeting='Hello', name='world')   
world, Hello!   
像这样使用名称指定的参数称为关键字参数，主要优点是有助于澄清各个参数的作用。

1. lambda表达式中可以使用任意复杂的表达式，但是必须只编写一个表达式。

对

1. g = lambda x: 3不是一个合法的赋值表达式。

错

1. 语句pass仅起到占位符的作用，并不会做任何操作。

对

1. 在Python中定义类时，如果某个成员名称前有2个下划线则表示是私有成员。

对

1. 在类定义的外部没有任何办法可以访问对象的私有成员。

错

# 直接访问对象的私有的\_\_age属性会出错 # print('直接访问对象的私有的\_\_age属性'+str(zhangsan.\_\_age)) # 访问私有属性 正确姿势 print(zhangsan.\_Person\_\_age)

1. Python中一切内容都可以称为对象。

对

1. 定义类时所有实例方法的第一个参数用来表示对象本身，在类的外部通过对象名来调用实例方法时不需要为该参数传值。

对

1. 面向对象程序设计中，函数和方法是完全一样的，都必须为所有参数进行传值。

错

1. Python中没有严格意义上的私有成员。

对

1. 在IDLE交互模式下，一个下划线“\_”表示解释器中最后一次显示的内容或最后一次语句正确执行的输出结果。

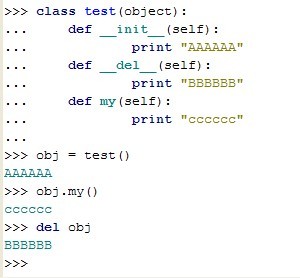
对

1. 对于Python类中的私有成员，可以通过“对象名.\_类名\_\_私有成员名”的方式来访问。

对

1. 如果定义类时没有编写析构函数，Python将提供一个默认的析构函数进行必要的资源清理工作。

对



上图中的“\_\_del\_\_”就是一个析构函数了，当使用del 删除对象时，会调用他本身的析构函数，另外当对象在某个作用域中调用完毕，在跳出其作用域的同时析构函数也会被调用一次，这样可以用来释放内存空间。

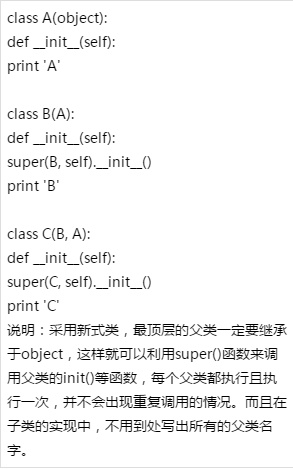
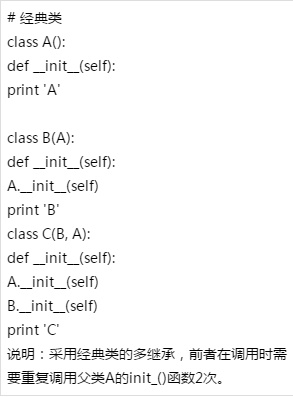
\_\_del\_\_()也是可选的，如果不提供，则Python 会在后台提供默认析构函数

如果要显式的调用析构函数，可以使用del关键字，方式如下：

del对象名

1. Python支持多继承，如果父类中有相同的方法名，而在子类中调用时没有指定父类名，则Python解释器将从左向右按顺序进行搜索。

对



1. 在Python中定义类时实例方法的第一个参数名称不管是什么，都表示对象自身。

对

1. Python类的构造函数是\_\_init\_\_()。

对

1. 定义类时，在一个方法前面使用@classmethod进行修饰，则该方法属于类方法。

对

**classmethod** 修饰符对应的函数不需要实例化，不需要 self 参数，但第一个参数需要是表示自身类的 cls 参数，可以来调用类的属性，类的方法，实例化对象等。

1. 定义类时，在一个方法前面使用@staticmethod进行休息，则该方法属于静态方法。

对

python staticmethod 返回函数的静态方法

1. Python类不支持多继承。

错

1. 函数和对象方法是一样的，内部实现和外部调用都没有任何区别。

错

1. 扩展库os中的方法remove()可以删除带有只读属性的文件。

错

1. 使用内置函数open()且以”w”模式打开的文件，文件指针默认指向文件尾。

错，只向开头，因为会清空文件

'w'：以只写模式打开。若文件存在，则会自动清空文件，然后重新创建；若文件不存在，则新建文件。使用这个模式必须要保证文件所在目录存在，文件可以不存在。该模式下不能使用read\*()方法

1. 使用内置函数open()打开文件时，只要文件路径正确就总是可以正确打开的。

错

使用这个模式必须要保证文件所在目录存在

1. 对文件进行读写操作之后必须显式关闭文件以确保所有内容都得到保存。

对

1. 二进制文件不能使用记事本程序打开。

错

1. 使用普通文本编辑器软件也可以正常查看二进制文件的内容。

错

需用二进制编辑器

1. 二进制文件也可以使用记事本或其他文本编辑器打开，但是一般来说无法正常查看其中的内容。

对

1. Python标准库os中的方法exists()可以用来测试给定路径的文件是否存在。

对

1. 文件对象的tell()方法用来返回文件指针的当前位置。

对

**tell()** 方法返回文件的当前位置，即文件指针当前位置。

1. 以读模式打开文件时，文件指针指向文件开始处。

对

1. 以追加模式打开文件时，文件指针指向文件尾。

对

1. 二进制文件也可以使用记事本程序打开，只是无法正确阅读和理解其中的内容。

对

1. 文本文件是可以迭代的，可以使用for line in fp类似的语句遍历文件对象fp中的每一行。

对