# 浅谈面试阿里外包python开发工程师

<https://blog.csdn.net/qq_42781462/article/details/82919948>

2018年10月02日 00:09:21 [王紫又](https://me.csdn.net/qq_42781462" \t "https://blog.csdn.net/qq_42781462/article/details/_blank) 阅读数：148

本人也是抱着不成功则成仁的想法去的

先说一下肯定有干货

反正不管过不过 总的榨点他身上的东西 对吧！ 抓到了总要一顿乱问

晚上7点半接到面试邀请电话

请先简单的自我介绍一下自己

吧啦吧啦说了有两三分钟 主要干什么的 毕业哪里的 为什么选python 以及除了python还会不会别的语言等等

然后直接进入正题

说说你知道的python变量作用域

LEGB 总体来说就是先当前外部函数 然后全局变量 然后内建

python2 用过没 xrange和range的区别是什么？

就拿最简单的例子来说 for i in range(5)

xrange的类型是列表 range的类型是int

那你说说python的 装饰器迭代器和生成器吧

装饰器的本质又是什么？

装饰器的本质就是函数 外层函数的返回值是内部函数的引用

再详细一些。

我曾看过flask 路由的源码 就是运用装饰器原理实现的

第一层是带参数就是url

第二层和第三层是装饰器

简单来说就是定义一个字典 以url为键以视图函数为值

但是用了falsk装饰器路由的同时再加上另一个装饰器会发生循环导入问题

这就要刨地层了 因为装饰器并不是简单的装饰 是把被装饰的函数的内存也更改了。

用大白话来说 就是被装饰器的函数的内存 放到了装饰器的内存里简单来说

就是成为了装饰器的一小部分

那解决方案呢？

一在外层函数上加上@functoolwraps(被装饰函数)

或者 在外层函数返回时把人家的内存地址给返回

为什么django没事呢？

因为django的路由是循环。 如果需要优化路由的话

最简单的方式就是把常访问的路由放在第一位

那你知道装饰器的args和kwars的解包解出来是什么吗

我......十脸懵逼

然后虚心请教 他说比如元祖是1字典是q:1

解包出来是 元祖是1，q:1字典是空的

还是十脸懵逼（我内心在想谁没事解包玩啊！！！能用就完事了）

说说迭代器吧？

迭代器 就要说可迭代对象了。

他说了很多包括int

我说Int是 他说int不是。

然后我说用range 然后他说你说的有道理。

我感觉他不太满意就赶紧说

其实for循环的本质就是iter取值next遍历

还有呢？

for循环自带捕获异常 while不带

说说生成器吧

其实最简单的方式就是列表推导式的中括号改成小括号

但同时又多了一个yield 那你能说说执行原理吗？

就是当你第一次next的时候到yield停止然后返回信息但是当你下次next的时候他是从yield继续执行的

还有拓展吗？

yield又跟协成有关。

但是封装了很多层比如greenlet 和gevent 他们两个都是对yield的封装

当然gevent不支持time如果你想用打个补丁 monkey

那你知道不知道 greenlet和gevent是扩展包不是内置包

我.....十脸懵逼

进程线程和协请你简单描述一下

我说举个简单的例子吧

一个公司是就好比一个进程 每个部门又是一个线程 然后比如技术部你我都可以成为一个协成

那你说说区别吧

进程比线程安全 但是同时进程所需的资源也是庞大的。因为如果你是单进程一个进程蹦了还有很多进程

线程相对来说所耗资源比较少但是不够安全 上面也说道了如果进程崩了所有的线程都不可执行

所以现在开发都是多进程和协程 因为协程比线程所需要的资源更少

但你在选择时候也有说法 进程不共享全局变量 所以你需要注意数据的安全

线程共享全局变量但是在处理大量数据时候会出现资源竞争所以你需要加锁 互斥锁有时候会出现死锁

说说死锁吧

最简单的例子就是下标越界 在return的同时一定把锁释放不然就会发生死锁

gil全局性解释性锁有听说过吗

有！  这个是python设计之初遗留下的问题 为了保证数据安全

线程也只是并发并不是并行

并发就是当你切换线程时候gil临时解锁加到目前所执行的线程上

所以并不存在真正的并行 只是计算机切换的速度比较快而已

关系型数据库和菲关系型数据库你熟悉吗

（心里暗暗自喜终于问到强项了）

mysql的访问量每秒并发1100次

redis的访问量每秒并发可到11W次

我个人感觉redis的提出就是为了减少访问关系型数据库

索引知道吗

所以其实就是约束 举个生活中的例子

你去看电影时候 你肯定选择一些 分类也就是目录 动作 喜剧爱情 等等

索引虽然可以优化你查询速度但是同时也会增加你增删改的速度

但是利大于弊总而来说 合适的

mysql集群听说过吗

您说的是主从搭建吧

对

简单来说就是 主服务器只做增删改 因为没有人会天天改密码和重复注册

从服务器做查 这样分工就很明确因为sql的访问最多就是查

优化sql语句懂吗？

举个简单例子 查询userid

select \* from user where = userid

select id \* from user where =userid

选下面哪种

为什么？

如果你在小公司随便 如果数据量过大的时候 整个sql服务器 经不起你几次 select \* from user

所以一定避免整表查询

is null  <> != 免了吧 这是整表查询的

多表查询你懂吗？

懂。 分组一定配合聚合函数使用不然没意义

内连接左连接右连接

都会出现笛卡尔积 虽然得到的数据是没有笛卡尔积 但是这些数据都是从笛卡尔积过滤出来的

一般取相同用的是内连接  左右用的少

即使用也是limit 避免整表查询

用的最多是子查询

子查询是能简单概况一下吗

可以分为独立子查询和不独立子查询

一般用独立子查询

为什么

因为非独立查询会出现一个返回值

独立子查询不会

给你一个需求很简单 要你查出来一条数据 如果你用三条查询语句很轻松 一条很难

但同时又没有要求你用三条或者一条 你会选择哪种

（送分题啊）一条

为什么？

sql的并发太低

说说建表吧 三大范式知道吗

举个例子 我们每天都在上班下班坐公交车

公交车为一张表  记录 座位数 几路公交车 是否有空调

人为一张表 老年人中年人和小孩

账单为一张表 现金 支付宝 老年卡 学生卡 公交卡

然后看是否满足三大范式

1 是否够原子性

2 只描述一件事

3 表中是否存在冗余字段

谈谈表格分割吧

千万级数据表如何分割

（又是送分题前几天我刚问了我叔 我叔在京东做后端）

就拿淘宝来说 订单表

比如淘宝是2000年成立

以18分割 2018年前查询18前这张表

2018后查询18年之后这张表 很多电商和银行的信息表都是这么存的

这只是分表

还有分库分表一说

说说看

把经常查询的放到一个库里 不经常查询的放到另一个库中

你说你写过小型服务器说说看

网络是用socket实现 处理简单并发是用的协成

截取http 请求体用的 切片os 和正则

appliction的路由是抄袭flask的装饰器实现

连接数据库是用的pymysql

有没有考虑到数据安全

您说的是ssl认证吧。 我自己写着玩的 公司的肯定有买

sql事物用过吗 事物等级知道吗

简单来说就是一起成功一起失败 等级是四个

经典类和新试类的区别知道吗

十脸懵逼 虚心请教

经典类和新试类的继承是有区别的

一个是深度遍历一个是广度遍历

深度遍历是先深入左侧然后返回然后深入右侧返回

广度遍历是 从底层开始一个超节点一个超节点遍历

听同事说继承都不太完美 这方面我也不是很懂

单利模式知道吗

知道 没用过

shell有接触吗

没有如果公司有用到我相信我在入职之前就能掌握

（十脸懵逼 之后有自己学了shell）

运维呢？

（十脸懵逼）

木有

java有接触吗

最近正在看c++和java 入职前肯定能掌握java基础

如果公司需要入职前能掌握

我这边的项目主要是开发跟运维有关的

我这边差不多问完了

你是否愿意来杭州工作？

（那个程序员不愿意去阿里云即使外派我也乐意）

愿意

你还有想问我的嘛

我问了业务和工作流程

然后说了自己有把握胜任

嗯确实工作难度不大 而且你有一门语言的基础学另一门不难

他说我这边要协商一下

好的麻烦了。

最后十分感谢您来面试我

不得不得说阿里的面试官说话真的超级有含量而且巨温柔绝不会怼你

而且非常有礼貌

最后我过了五天还没得到通知应该是被别人刷下来的 凉了

也看了shell 和一些运维

所以说java 不管你是前端后端软件开发 都要看

c++也一定要看

还有这次我面试很经常 希望大家面试还是保证乐观的心态前去

最后祝愿大家从BAT面试出来那一瞬间有着战士收刀入鞘的骄傲

# python笔试题（1）

  为了充实自己，小编决定上传自己见到的笔试题和面试题。可能要写好长时间，一时半会写不了多少，只能说遇到多少写多少吧，但是只要小编有时间，会持续上传（但是答案却不能保证，所以有看到错误的及时联系小编，以免误导其他人）。

### ****1.单引号，双引号，三引号的区别****

1. 分别阐述3种引号用的场景和区别
2. 1),单引号和双引号主要用来表示字符串
3. 比如:
4. 单引号:'python'
5. 双引号:"python"
6. 2).三引号
7. 三单引号:'''python ''',也可以表示字符串一般用来输入多行文本,或者用于大段的注释
8. 三双引号："""python"""，一般用在类里面,用来注释类,这样省的写文档,直接用类的对象\_\_doc\_\_访问获得文档
9. 区别:
10. 若你的字符串里面本身包含单引号,必须用双引号
11. 比如:"can't find the log\n"

### ****2.Python的参数传递是值传递还是引用传递****

1. 举例说明Python函数参数传递的几种形式，并说明函数传参是值传递还是引用传递
2. 1).Python的参数传递有：
3. 位置参数
4. 默认参数，
5. 可变参数,
6. 关键字参数
7. 2).函数的传值到底是值传递还是引用传递，要分情况
8. a.不可变参数用值传递：
9. 像整数和字符串这样的不可变对象，是通过拷贝进行传递的，因为你无论如何都不可能在原处改变不可变对象
10. b.可变参数是用引用传递的
11. 比如像列表，字典这样的对象是通过引用传递，和C语言里面的用指针传递数组很相似，可变对象能在函数内部改变.

## ****3.什么是lambda函数？它有什么好处?****

1. 举例说明lambda的用法，并说明用lambda的优点
2. 1).lambda的用法：
3. lambda是匿名函数，用法如下:lambda arg1,arg2..argN:expression using args
4. 2).优点
5. lambda能和def做同样种类的工作，特别是对于那些逻辑简单的函数，直接用lambda会更简洁，
6. 而且省去取函数名的麻烦(给函数取名是个技术活)

## ****4.字符串格式化:%和.format的区别****

1. 字符串的format函数非常灵活，很强大,可以接受的参数不限个数，并且位置可以不按顺序，
2. 而且有较为强大的格式限定符(比如:填充，对齐,精度等)

## ****5.Python是如何进行内存管理的****

1. 1).对象的引用计数机制
2. Python内部使用引用计数，来保持追踪内存中的对象，所有对象都有引用计数。
3. 引用计数增加的情况：
4. 一个对象分配一个新名称
5. 将其放入一个容器中（如列表、元组或字典）
6. 引用计数减少的情况：
7. 使用del语句对对象别名显示的销毁
8. 引用超出作用域或被重新赋值
9. 2).垃圾回收
10. 当一个对象的引用计数归零时，它将被垃圾收集机制处理掉。
11. 3).内存池机制
12. Python提供了对内存的垃圾收集机制，但是它将不用的内存放到内存池而不是返回给操作系统:
13. Pymalloc机制：为了加速Python的执行效率，Python引入了一个内存池机制，用于管理对小块内存的申请和释放。
14. 对于Python对象，如整数，浮点数和List，都有其独立的私有内存池，对象间不共享他们的内存池。
15. 也就是说如果你分配又释放了大量的整数，用于缓存这些整数的内存就不能再分配给浮点数。

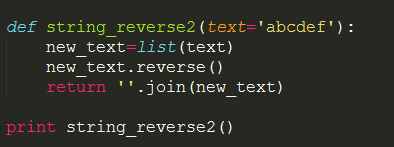
## ****6.写一个函数, 输入一个字符串, 返回倒序排列的结果****

输入: string\_reverse(‘abcdef’), 返回: ‘fedcba’,写出你能想到的多种方法

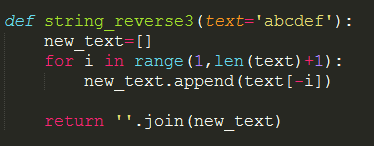
****1).利用字符串本身的翻转****

1. def string\_reverse1(text='abcdef'):
2. return text[::-1]

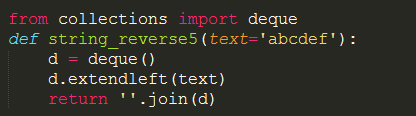
****2).把字符串变成列表，用列表的reverse函数****



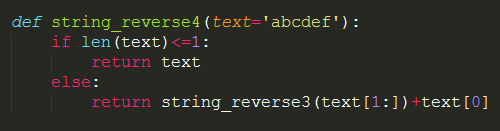
****3).新建一个列表，从后往前取****



****4).利用双向列表deque中的extendleft函数****



****5).递归****



## ****7.按升序合并如下两个list, 并去除重复的元素****

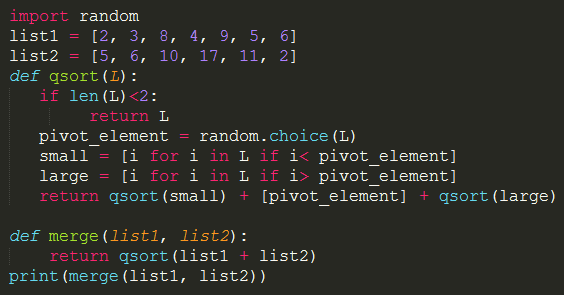
1. list1 = [2, 3, 8, 4, 9, 5, 6]
2. list2 = [5, 6, 10, 17, 11, 2]

****1).最简单的方法用set****

1. list3=list1+list2
2. print sorted(list(set(list3)))

****2).递归****

先选一个中间数，然后一边是小的数字，一边是大的数字，然后再循环递归，排完序(是不是想起了c里面的冒泡)



## ****8.以下的代码的输出将是什么? 说出你的答案并解释****

1. class Parent(object):
2. x = 1
3. class Child1(Parent):
4. pass
5. class Child2(Parent):
6. pass
7. print Parent.x, Child1.x, Child2.x
8. Child1.x = 2
9. print Parent.x, Child1.x, Child2.x
10. Parent.x = 3
11. print Parent.x, Child1.x, Child2.x
12. >>
13. 1 1 1
14. 1 2 1
15. 3 2 3

****解答:****

1. 使你困惑或是惊奇的是关于最后一行的输出是 3 2 3 而不是 3 2 1。
2. 为什么改变了 Parent.x 的值还会改变 Child2.x 的值，但是同时 Child1.x 值却没有改变？
4. 这个答案的关键是，在 Python中，类变量在内部是作为字典处理的。
5. 如果一个变量的名字没有在当前类的字典中发现，将搜索祖先类（比如父类）直到被引用的变量名被找到.
7. 首先，在父类中设置 x = 1 会使得类变量 x 在引用该类和其任何子类中的值为 1。
8. 这就是因为第一个 print 语句的输出是 1 1 1
10. 然后，如果任何它的子类重写了该值（例如，我们执行语句 Child1.x = 2）该值仅仅在子类中被改变
11. 。这就是为什么第二个 print 语句的输出是 1 2 1
13. 最后，如果该值在父类中被改变（例如，我们执行语句 Parent.x = 3），这个改变会影响
14. 到任何未重写该值的子类当中的值（在这个示例中被影响的子类是 Child2）。
15. 这就是为什么第三个 print 输出是 3 2 3

## ****9.下面的代码会不会报错****

1. list = ['a', 'b', 'c', 'd', 'e']
2. print list[10:]

****不会报错，而且会输出一个 []，并且不会导致一个 IndexError****

****解答:****

1. 当试图访问一个超过列表索引值的成员将导致 IndexError（比如访问以上列表的 list[10]）。
2. 尽管如此，试图访问一个列表的以超出列表长度数作为开始索引的切片将不会导致 IndexError，
3. 并且将仅仅返回一个空列表
4. 一个讨厌的小问题是它会导致出现 bug ，并且这个问题是难以追踪的，
5. 因为它在运行时不会引发错误，吐血啊~~

## ****10.说出下面list1,list2,list3的输出值****

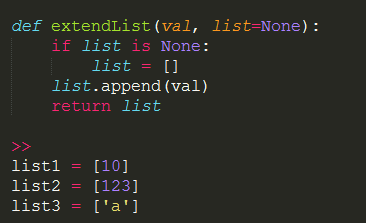
1. def extendList(val, list=[]):
2. list.append(val)
3. return list
4. list1 = extendList(10)
5. list2 = extendList(123,[])
6. list3 = extendList('a')
7. print "list1 = %s" % list1
8. print "list2 = %s" % list2
9. print "list3 = %s" % list3
10. >>
11. list1 = [10, 'a']
12. list2 = [123]
13. list3 = [10, 'a']

　　许多人会错误的认为 list1 应该等于 [10] 以及 list3 应该等于 ['a']。认为 list 的参数会在 extendList 每次被调用的时候会被设置成它的默认值 []。

　　尽管如此，实际发生的事情是，新的默认列表仅仅只在函数被定义时创建一次。随后当 extendList 没有被指定的列表参数调用的时候，其使用的是同一个列表。这就是为什么当函数被定义的时候，表达式是用默认参数被计算，而不是它被调用的时候。

　　因此，list1 和 list3 是操作的相同的列表。而list2是操作的它创建的独立的列表（通过传递它自己的空列表作为list参数的值）

****所以这一点一定要切记切记.下面我们把list置为None就可以避免一些麻烦了****



## ****11.写出你认为最Pythonic的代码****

****Pythonic编程风格是Python的一种追求的风格，精髓就是追求直观，简洁而容易读****.

下面是一些比较好的例子

****1).交互变量****

****非Pythonic****

1. temp = a
2. a = b
3. b = temp

****pythonic:****

1. a,b=b,a

****2).判断其值真假****

1. name = 'Tim'
2. langs = ['AS3', 'Lua', 'C']
3. info = {'name': 'Tim', 'sex': 'Male', 'age':23 }

****非Pythonic****

1. if name != '' and len(langs) > 0 and info != {}:
2. print('All True!')

****pythonic:****

1. if name and langs and info:
2. print('All True!')

****3).列表推导式****

1. [x for x in range(1,100) if x%2==0]

****4).zip创建键值对****

1. keys = ['Name', 'Sex', 'Age']
2. values = ['Jack', 'Male', 23]
3. dict(zip(keys,values))

## 12.写出一段python 代码实现一个删除一个list 里面重复元素。

****答案：****

1. list\_element = ['a','c,','z','x','a'] #此列表元素有重复
2. delete\_element = list( set(list\_element)) #利用集合的唯一性删除重复元素
3. print("原始列表为：",list\_element)
4. print("修改后的列表为：",delete\_element)

结果：

1. 原始列表为： ['a', 'c,', 'z', 'x', 'a']
2. 修改后的列表为： ['c,', 'x', 'z', 'a']
4. Process finished with exit code 0

但是这样做有缺点，就是去重后，元素的排序改变了，想保持原来的排序，我们需要用下面的方法：

1. list\_element = ['a','c,','z','x','a'] #此列表元素有重复
2. delete\_element = list( set(list\_element)) #利用集合的唯一性删除重复元素
3. delete\_element.sort(key = list\_element.index) #对修改后的列表进行排序
4. print("原始列表为：",list\_element)
5. print("修改后的列表为：",delete\_element)

结果：

1. 原始列表为： ['a', 'c,', 'z', 'x', 'a']
2. 修改后的列表为： ['a', 'c,', 'z', 'x']
4. Process finished with exit code 0

## ****13.什么是切片？****

1. 答案：
2. 切片的第一个索引为起始点，第二个索引则是比我们需要的最后一个元素的索引值大1的数字。
3. 严格的讲，list[i,j]  就是原始列表从索引i（包含）处开始，一直到索引 j 处（不包含）结束的一个切片。
5. 注释：Python使用这种约定的原因是为了与列表索引的合法规则保持一致（从0开始，最高比例列表长度小1的数字）

## 14：模块是什么？

1. 模块(module)是 Python 中非常重要的东西，你可以把它理解为 Python 的扩展工具。
2. 换言之，Python 默认情况下提供了一些可用的东西，但是这些默认情况下提供的还远远不
3. 能满足编程实践的需要，于是就有人专门制作了另外一些工具。这些工具被称之为“模块”
5. 任何一个 Pythoner 都可以编写模块，并且把这些模块放到网上供他人来使用。
6. 当安装好 Python 之后，就有一些模块默认安装了，这个称之为“标准库”，“标准库”中
7. 的模块不需要安装，就可以直接使用。
8. 如果没有纳入标准库的模块，需要安装之后才能使用。模块的安装方法，我特别推荐使用 pip 来安装。

## ****15：dir（）是什么指令？****

1. dir(module)是一个非常有用的指令，可以通过它查看任何模块中所包含的工具。

## ****16：如何把一个文件内的字符串形式通过json转化为相应的字典格式？****

****常见错误：****

1. #account\_file是文件的绝对路径
2. with open(account\_file, "r", encoding="utf-8") as f: #打开文件
3. file\_data = json.load(account\_file)
4. print(file\_data)
6. 这样竟然出错了！！
7. 错误信息：AttributeError: 'str' object has no attribute 'read'

****改正后为：****

1. #改正：
2. if os.path.isfile(account\_file): #如果用户文件存在(即用户存在)
3. with open(account\_file, "r", encoding="utf-8") as f: #打开文件
4. file\_data = json.load(f)
5. print(file\_data)

## 17：反斜杠的困扰

1. 与大多数编程语言相同，正则表达式里面使用“ \ ”作为转义字符，这就可能造成反斜杠困扰
2. ，假如你需要匹配文本中的字符'' \ '' 那么使用编程语言表示的正则表达式里面将需要4个反斜
3. 杠"\\\\",前两个和后两个分别用于在编程语言里转义成反斜杠，转换成两个反斜杠后再在正则表
4. 达式里面转义成一个反斜杠，python里的原生字符串很好的解决了这个问题，这个例子中的正则
5. 表达式可以使用  r"\\  表示。同样，匹配一个数字的"\\d"可以写成r"\d.有了原生字符串，你再
6. 也不用担心是不是漏写了反斜杠，写出来的表达式也更直观。

仅仅需要知道及格匹配模式

1. re.I(re.IGNORECASE): 忽略大小写（括号内是完整写法，下同）
2. M(MULTILINE): 多行模式，改变'^'和'$'的行为（参见上图）
3. S(DOTALL): 点任意匹配模式，改变'.'的行为

## 18：python中 if \_\_name\_\_ == '\_\_main\_\_': 的作用是什么呢

　　这段代码的功能理解如下：

一个python的文件有两种使用方法：

1. 作用一：直接作为脚本执行
2. 作用二：import 到其他的python脚本中被调用执行

****简而言之：\_\_name\_\_就是当前模块名，当模块被直接运行时候模块名称为\_\_main\_\_。当模块被直接运行的时候，代码被运行，当模块被导入的时候，代码不被运行。****

　　详细解释：

　　模块是对象，并且所有的模块都有一个内置属性\_\_name\_\_。一个模块的\_\_name\_\_的值取决于我们如何应用模块，如果import 一个模块，那么模块\_\_name\_\_的值通常是模块文件名，不带路径或者文件扩展名，但是我们也可以像一个标准的程序直接运行模块，在这种情况下，\_\_name\_\_的值将是一个特别缺省"\_\_main\_\_"

　　首先，可以让大家看一下在cmd中运行.py文件则\_\_name\_\_的值为“\_\_main\_\_”

1. Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 18:11:49) [MSC v.1900 64 bit (AMD64)] on win32
2. Type "help", "copyright", "credits" or "license" for more information.
3. >>> \_\_name\_\_
4. '\_\_main\_\_'

　　而在import 一个.py文件后,\_\_name\_\_的值就不是'\_\_main\_\_'了;

　　从而用if \_\_name\_\_ == '\_\_main\_\_'来判断是否是在直接运行该.py文件

　　运行原理

每个python模块（python文件）都包含内置的变量\_\_name\_\_，当运行模块被执行的时候，\_\_name\_\_等于文件名（包含了后缀.py）。如果import到其他模块中，则\_\_name\_\_等于模块名称（不包含后缀.py）。而“\_\_main\_\_”等于当前执行文件的名称（包含了后缀.py）。所以当模块被直接执行时，\_\_name\_\_ == '\_\_main\_\_'结果为真；而当模块被import到其他模块中时，\_\_name\_\_ == '\_\_main\_\_'结果为假，就是不调用对应的方法。

## 19：深浅copy

具体内容见博客：<http://www.cnblogs.com/wj-1314/p/7436299.html>

## ****20：递归函数（假设面试官让你把number =[2, -5, 9, -7, 2, 5, 4, -1, 0, -3, 8]中的正数的平均值求出来，你怎么算）****

　　首先循环列表中的值，累计次数，并对大于0的数进行累加，最后求取平均值。

　　这就是命令式编程——你要做什么事情，你得把达到目的的步骤详细的描述出来，然后交给机器去运行。

　　这也正是命令式编程的理论模型——图灵机的特点。一条写满数据的纸带，一条根据纸带内容运动的机器，机器每动一步都需要纸带上写着如何达到。

1. #计算数组中正整数的平均值
3. number =[2, -5, 9, -7, 2, 5, 4, -1, 0, -3, 8]
4. count = 0
5. sum = 0
7. for i in range(len(number)):
8. if number[i]>0:
9. count += 1
10. sum += number[i]
12. print sum,count
14. if count>0:
15. average = sum/count
17. print average
19. #========输出===========
20. 30 6
21. 5

　　还有一种方法：

1. number =[2, -5, 9, -7, 2, 5, 4, -1, 0, -3, 8]
3. positive = filter(lambda x: x>0, number)
5. average = reduce(lambda x,y: x+y, positive)/len(positive)
7. print average
9. #========输出===========
10. 5

　　这段代码最终达到的目的同样是求取正数平均值，但是它得到结果的方式和 之前有着本质的差别：通过描述一个列表->正数平均值 的映射，而不是描述“从列表得到正数平均值应该怎样做”来达到目的。

# 阿里大数据架构师总结16道Python面试题！不可不看啊！

