原文标题：一名python web后端开发工程师的面试总结

先介绍下我的情况

通信背景，工作一年多不到两年。之前一直在做C++的MFC软件界面开发工作。公司为某不景气的国企研究所。（喏，我的工作经验很水：1是方向不对；2是行业有偏差）。然后目前是在寻找python后端开发这一块的工作，使用的框架为django；之前一直通过CSDN以及其他几家技术博客/论坛吸收大家的经验，在感激之余，也想输出点什么，造福大家，因此就有了这篇水文，希望大家能够多多吸取我的经验教训，早日找到一份自己满意的工作！

**面试的几家：**北京的两家创业公司，规模均在40-50人之间；果壳（止步于电话面，拉钩投的）；知乎（止步于电话面，论坛发帖后的内推）；爱奇艺（拉钩上投的）；杭州的网易（同学内推）。

面试这一轮下来，最大的感受就一个：平时**一定要坚持每天都码点代码**。就算再烂的项目，也要坚持上传github……真的，只要你能坚持一周有5天以上都能持续上传Git，半年下来，面试官绝对对你刮目相看。

其他感受：不同的公司着重点都不一样，不过真的就如这篇博文（http://blog.csdn.net/yueguanghaidao/article/details/49638261）的博主所说，木桶原理，哪块都不能少。我就是因为平时只关注做自己的小博客系统，而没做其他的部分，导致爱奇艺倒在了手写代码上面……哎，忧伤……平时多刷刷leetcode，看看剑指offer/面试金典，都是套路啊~~

面试这几家公司所遇到的面试/笔试题，目前还能记住的如下。虽然可能绝大部分都是基础，但希望大家**不要只是看看就过去了**，最好还是假装你被问到这个问题，你来把答案说出来或写出来：（不按公司分了）

**python语法以及其他基础部分**

1可变与不可变类型

Python的每个对象都分为可变和不可变，主要的核心类型中，数字、字符串、元组是不可变的，列表、字典是可变的. 对不可变类型的变量重新赋值，实际上是重新创建一个不可变类型的对象，并将原来的变量重新指向新创建的对象（如果没有其他变量引用原有对象的话（即引用计数为0），原有对象就会被回收）。比如说创建一个字符串m为”abc”,如果让m改为”abcd”,实际则是在内存空间又创建一个区域保存”abcd”,并让变量m指向其区域.

可参考: https://www.cnblogs.com/blackmatrix/p/5614086.html

2 赋值,浅拷贝与深拷贝的实现方式、区别

先理解两个变量/对象a与b的内存空间,还有变量/对象a与b的值占用的内存空间,其次,要知道在python中，对象的赋值”=”，拷贝（深/浅拷贝）之间是有差异的:

= 赋值,a=b, a与b变量本身占用同一个内存空间,其引用的值理所当然的占用占一个内存空间

浅,b=copy.copy(a),之后a与b变量本身占用不同的内存空间,但其引用的值却是同一个内存空间,所以,对a的值进行修改也会影响b的值, 注意,这里的值指的是可变类型

深,b=copy.deepcopy(a), 之后a与b变量本身占用不同的内存空间,其引用的值也是不同的内存空间,同上,这里的值指的是可变类型,所以,对a的值进行修改不会影响b的值

对于不可变类型的对象，对于深浅拷贝毫无影响，最终的地址值和值都是相等的,即如果a的值是不可变类型,那么b不管是深还是浅拷贝,其值和a的值都指向同一个内存空间,如果a的值改变了,那么a的值的内存空间一定改变,那么b的值不会变,因为b引用的仍是原来的值的内存空间.

参考1: <http://python.jobbole.com/82294/>

参考1中的总结:

本文介绍了对象的赋值和拷贝，以及它们之间的差异：

Python中对象的赋值都是进行对象引用（内存地址）传递

使用copy.copy()，可以进行对象的浅拷贝，它复制了对象，但对于对象中的元素，依然使用原始的引用.

如果需要复制一个容器对象，以及它里面的所有元素（包含元素的子元素），可以使用copy.deepcopy()进行深拷贝

对于非容器类型（如数字、字符串、和其他’原子’类型的对象）没有被拷贝一说

如果元祖变量只包含原子类型对象，则不能深拷贝

参考2: https://blog.csdn.net/u014745194/article/details/70271868

参考2中的总结;

0, 赋值=是对源对象的引用,不管是对象还是对象中的元素都占用相同的内存空间.

1，深浅拷贝都是对源对象的复制,占用不同的内存空间.

2， 对于不可变类型的值/对象元素，对于深浅拷贝毫无影响，最终的地址值和值都是相等的。

3，对于可变类型的值/对象元素：

copy浅拷贝：值相等，地址相等

deepcopy深拷贝：值相等，地址不相等

Deepcopy如果你来设计，如何实现； 

3.\_\_new\_\_() 与 \_\_init\_\_()的区别

def \_\_init\_\_(self), \_\_init\_\_ 方法在初始化一个类的实例的时候调用

def \_\_new\_\_(cls), \_\_new\_\_方法在创建这个类实例的时候调用的方法

所以，\_\_init\_\_ 和 \_\_new\_\_ 最主要的区别在于：

1.\_\_init\_\_ 通常用于初始化一个新实例，控制这个初始化的过程，比如添加一些属性， 做一些额外的操作，发生在类实例被创建完以后。它是实例级别的方法。

2.\_\_new\_\_ 通常用于控制生成一个新实例的过程。它是类级别的方法。

参考: <http://python.jobbole.com/86506/>

3.1 \_\_new\_\_()实现单例模式

事实上，当我们理解了\_\_new\_\_方法后，我们还可以利用它来做一些其他有趣的事情，比如实现 设计模式中的 单例模式(singleton) 。

因为类每一次实例化后产生的过程都是通过\_\_new\_\_来控制的，所以通过重载\_\_new\_\_方法，我们 可以很简单的实现单例模式。

\_\_new\_\_()实现单例模式的代码如下

|  |  |
| --- | --- |
|  | class Singleton(object):      def \_\_new\_\_(cls):          # 关键在于这，每一次实例化的时候，我们都只会返回这同一个instance对象          if not hasattr(cls, 'instance'):              cls.instance = super(Singleton, cls).\_\_new\_\_(cls)          return cls.instance    obj1 = Singleton()  obj2 = Singleton()    obj1.attr1 = 'value1'  print obj1.attr1, obj2.attr1  print obj1 is obj2 |

输出结果:

value1 value1

True

你知道几种设计模式；   
编码和解码你了解过么；

序列,可迭代对象,迭代器之间的区别和联系

列表生成式/列表推导式list comprehension和生成器的区别,优劣

列表生成式

即List Comprehensions，是Python内置的非常简单却强大的可以用来创建list的生成式,语法为[].

列表生成式代替一层循环:

>>> [x \* x for x in range(1, 11)]

[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]

列表生成式代替二层循环:

>>> [m + n for m in 'ABC' for n in 'XYZ']

['AX', 'AY', 'AZ', 'BX', 'BY', 'BZ', 'CX', 'CY', 'CZ']

列表生成式迭代字典里的key和value

>>> d = {'x': 'A', 'y': 'B', 'z': 'C' }

>>> [k + '=' + v for k, v in d.items()]

['y=B', 'x=A', 'z=C']

综上:列表生成式用在对一个序列(字符串,列表,元组,字典)中的各个元素作处理,且处理的过程不是很复杂时

生成器 generator:

列表生成式，可以创建一个列表,但也会占用同等大小的内存,浪费内存空间. 而生成器中的元素可以按照某种算法推算出来，可以在循环的过程中不断推算出后续的元素, 且一边循环一边计算, 从而节省大量的空间,这就是生成器的优势.

生成器实现的两种方式:

第一种方法，只要把一个列表生成式的[]改成()

第二种方式, 用函数去实现生成器, 在函数中添加yield 变量名,变成生成器函数

生成器的使用方式:

1.在for 循环中

2.用next()去调用

什么是装饰器

在了解什么是装饰器之前,要先了解什么是闭包: 函数里面定义了一个函数,并且这个内层函数用到了外边函数的变量,那么这个函数以及用到的一些变量称之为闭包

接下来说明装饰器,装饰器分为函数装饰器与类装饰器,其作用就是在不改变testFun函数(即被装饰的函数)名字的情况下给其函数增加功能,使得代码遵循开发封闭原则,使得已经实现的功能代码不允许被修改，但可以被扩展.( 封闭：已实现的功能代码块. 开放：对扩展开放)

函数装饰器就是用闭包来写的装饰方法,类装饰器是用类来写的装饰类,主要是\_\_call\_\_方法.

参考: <https://www.cnblogs.com/lianyingteng/p/7743876.html>

参考: <http://python.jobbole.com/86632/>

如果想在函数之后进行装饰，应该怎么做；

手写个使用装饰器实现的单例模式；

1. **def** singleton(cls, \*args, \*\*kw):
2. instance={}
3. **def** \_singleton():
4. **if** cls **not** **in** instance:
5. instance[cls]=cls(\*args, \*\*kw)
6. **return** instance[cls]
7. **return** \_singleton
9. @singleton
10. **class** test\_singleton(object):
11. **def** \_\_init\_\_(self):
12. self.num\_sum=0
13. **def** add(self):
14. self.num\_sum=100

参考: <https://blog.csdn.net/thn_sweety/article/details/52565688>

使用装饰器的单例和使用其他方法的单例，在后续使用中，有何区别；

手写：正则邮箱地址；

介绍下垃圾回收：引用计数/分代回收/孤立引用环；

多进程与多线程的区别；CPU密集型适合用什么；   
进程通信的方式有几种；   
介绍下协程，为何比线程还快；

range和xrange的区别

python2中range()方法返回一个列表,而xrange()用法一样,只是返回的是<type 'xrange'>类型的生成器对象,节省内存空间

python3中,range()这种实现被移除了,保留了xrange()的实现,且将xrange重新命名为range(),如果想让其变成列表,则用list(range(..))方法

xrange做循环的性能比range好，尤其是返回很大的时候。尽量用xrange吧，除非你是要返回一个列表。

参考1: <http://www.360doc.com/content/14/0322/01/9482_362598142.shtml>

参考2: <http://www.nowamagic.net/academy/detail/1302446>

由于我有C/C++背景，因此要求用C来手写：将IP地址字符串（比如“172.0.0.1”）转为32位二进制数的函数。

1. 先将ip以’.’为分割符生成一个列表iplist,用的方法是str.split(‘.’)
2. 再将iplist中每部分转为32位二进制字符串,用bin(int(iplist[i]))

ip = “172.0.0.1”

ip\_int\_list = [int(x) for x in ip.split(‘.’)] #[172,0,0,1]

ip\_str\_list = [int2str(x) for x in ip\_int\_list] #[]

ipStr = “.”.join(ip\_str\_list) # “”

# python整型转换为32位二进制字符串

INTLEN = 32

def int2str(i):

str = ""

flag = -1

while i>0 or flag==-1:

if(i%2 == 1):

flag = -1

str="1"+str

else:

str="0"+str

flag = 0

i = i/2

if(flag==-1):

str = "1"+str

return str.zfill(INTLEN)

列,可迭代对象,迭代器,生成器之间的区别与联系

序列:

常见序列类型包括字符串(普通字符串和unicode字符串),列表和元组.所谓序列,即成员有序排列,可通过下标访问. Python包含 6 种内建的序列，包括列表、元组、字符串、Unicode字符串、buffer对象和xrange对象。

序列通用操作有:

1. 索引
2. 切片
   1. obj[start\_index:end\_index], 切片中冒号左边索引对应的元素晚于右边索引对应元素出现时,返回结果是[]
   2. [start\_index:end\_index:step] (step>0)
   3. [start\_index:end\_index:step] (step<0), 此时start\_index对应的元素要晚于end\_index对应的元素出现,否则返回[]
3. 相加,乘法
   1. 两种相同的序列才可以进行加法操作,即list1+list2,str1+str2
   2. str1\*3 ,list1\*3
4. 判断成员资格

If x in str1, if x in list1:

1. 最大值max(),最小值min(),求长度len()

参考1: <https://www.cnblogs.com/diaosir/p/6575891.html>

参考2: <https://blog.csdn.net/hedan2013/article/details/55000018>

可迭代对象:

1. 精确定义: 只要一个对象定义了可以返回一个迭代器的\_\_iter\_\_()方法，或者定义了可以支持下标索引的\_\_getitem\_\_()方法 ，那么它就是一个可迭代对象。
2. 模糊定义: 可以用for..in 循环遍历的对象就是可迭代对象,比如, list,dict,set,str，generator（生成器）,iterator（迭代器）全部都是可迭代对象（英语叫Iterable）

迭代器:

精确定义:在python2中必须实现了next()方法的类对象,在python3中实现了\_\_iter\_\_和\_\_next\_\_()方法的对象

那么什么是迭代器呢？它是一个带状态的对象，他能在你调用next()方法的时候返回容器中的下一个值，任何实现了\_\_iter\_\_和\_\_next\_\_()（python2中实现next()）方法的对象都是迭代器，\_\_iter\_\_返回迭代器自身，\_\_next\_\_返回容器中的下一个值，如果容器中没有更多元素了，则抛出StopIteration异常，至于它们到底是如何实现的这并不重要。

凡是可以返回一个迭代器的对象都可称之为可迭代对象

迭代器有一种具体的迭代器类型，比如list\_iterator，set\_iterator。可迭代对象实现了\_\_iter\_\_方法，该方法返回一个迭代器对象。

生成器:

生成器算得上是Python语言中最吸引人的特性之一，生成器其实是一种特殊的迭代器，不过这种迭代器更加优雅。它不需要再像上面的类一样写\_\_iter\_\_()和\_\_next\_\_()方法了，只需要一个yiled关键字。 生成器一定是迭代器（反之不成立），但迭代器不一定是生成器

生成器的实现方式:

1. 生成器表达式,返回一个生成器对象
2. 带yield的函数为生成器函数

参考1: <http://python.jobbole.com/87805/>

参考2: <https://www.cnblogs.com/wswang/p/6047994.html>

参考3: <https://blog.csdn.net/u014745194/article/details/70176117>

参考4: <https://www.zhihu.com/question/20829330>

参考5: <http://python.jobbole.com/85240/>

Python 2 和 Python 3 有哪些主要区别？

参考: <https://www.zhihu.com/question/19698598>

1.Python是如何进行内存管理的？

答:从三个方面来说,一对象的引用计数机制,二垃圾回收机制,三内存池机制

一、对象的引用计数机制

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。

引用计数增加的情况：

1，一个对象分配一个新名称

2，将其放入一个容器中（如列表、元组或字典）

引用计数减少的情况：

1，使用del语句对对象别名显示的销毁

2，引用超出作用域或被重新赋值

sys.getrefcount( )函数可以获得对象的当前引用计数

多数情况下，引用计数比你猜测得要大得多。对于不可变数据（如数字和字符串），解释器会在程序的不同部分共享内存，以便节约内存。

二、垃圾回收

1，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。

2，当两个对象a和b相互引用时，del语句可以减少a和b的引用计数，并销毁用于引用底层对象的名称。然而由于每个对象都包含一个对其他对象的应用，因此引用计数不会归零，对象也不会销毁。（从而导致内存泄露）。为解决这一问题，解释器会定期执行一个循环检测器，搜索不可访问对象的循环并删除它们。

三、内存池机制

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。

1，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。

2，Python中所有小于256个字节的对象都使用pymalloc实现的分配器，而大的对象则使用系统的malloc。

3，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。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。

4.请写出一段Python代码实现删除一个list里面的重复元素

1,使用set函数，set(list)

2，使用字典函数，

>>>a=[1,2,4,2,4,5,6,5,7,8,9,0]

>>> b={}

>>>b=b.fromkeys(a)

>>>c=list(b.keys())

>>> c

3.编程用sort进行排序，然后从最后一个元素开始判断

if \_\_name\_\_ == '\_\_main\_\_':

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

a.sort()

last = a[-1]

for i in range(len(a)-2,-1,-1):

if last == a[i]:

del a[i]

else:

last = a[i]

print(a)

'''

输出结果

[1, 2, 3, 4, 5]

[Finished in 0.2s]

'''

6.Python里面如何拷贝一个对象？（赋值，浅拷贝，深拷贝的区别）

赋值（=），就是创建了对象的一个新的引用，修改其中任意一个变量都会影响到另一个。

浅拷贝：创建一个新的对象，但它包含的是对原始对象中包含项的引用（如果用引用的方式修改其中一个对象，另外一个也会修改改变）{1,完全切片方法；2，工厂函数，如list()；3，copy模块的copy()函数}

深拷贝：创建一个新的对象，并且递归的复制它所包含的对象（修改其中一个，另外一个不会改变）{copy模块的deep.deepcopy()函数}

9.介绍一下Python下range()函数的用法？

10.如何用Python来进行查询和替换一个文本字符串？

答：可以使用re模块中的sub()函数或者subn()函数来进行查询和替换，

格式：sub(replacement, string[,count=0])（replacement是被替换成的文本，string是需要被替换的文本，count是一个可选参数，指最大被替换的数量）

>>> import re

>>>p=re.compile(‘blue|white|red’)

>>>print(p.sub(‘colour’,'blue socks and red shoes’))

colour socks and colourshoes

>>>print(p.sub(‘colour’,'blue socks and red shoes’,count=1))

colour socks and redshoes

subn()方法执行的效果跟sub()一样，不过它会返回一个二维数组，包括替换后的新的字符串和总共替换的数量

11.Python里面match()和search()的区别？

答：re模块中match(pattern,string[,flags]),检查string的开头是否与pattern匹配。

re模块中research(pattern,string[,flags]),在string搜索pattern的第一个匹配值。

>>>print(re.match(‘super’, ‘superstition’).span())

(0, 5)

>>>print(re.match(‘super’, ‘insuperable’))

None

>>>print(re.search(‘super’, ‘superstition’).span())

(0, 5)

>>>print(re.search(‘super’, ‘insuperable’).span())

(2, 7)

12.用Python匹配HTML tag的时候，<.\*>和<.\*?>有什么区别？

答：术语叫贪婪匹配( <.\*> )和非贪婪匹配(<.\*?> )

例如:

test

<.\*> :

test

<.\*?> :

13.Python里面如何生成随机数？

答：random模块

随机整数：random.randint(a,b)：返回随机整数x,a<=x<=b

random.randrange(start,stop,[,step])：返回一个范围在(start,stop,step)之间的随机整数，不包括结束值。

随机实数：random.random( ):返回0到1之间的浮点数

random.uniform(a,b):返回指定范围内的浮点数。

14.有没有一个工具可以帮助查找python的bug和进行静态的代码分析？

答：PyChecker是一个python代码的静态分析工具，它可以帮助查找python代码的bug, 会对代码的复杂度和格式提出警告

Pylint是另外一个工具可以进行codingstandard检查

# 简单基础

2.什么是lambda函数？它有什么好处?

答：lambda 表达式，通常是在需要一个函数，但是又不想费神去命名一个函数的场合下使用，也就是指匿名函数

lambda函数：首要用途是指短小的回调函数

用法: lambda [arguments] : expression

>>> a=lambdax,y:x+y

>>> a(3,11)

3.Python里面如何实现tuple和list的转换？

答：直接使用tuple和list函数就行了，type()可以判断对象的类型

7.介绍一下except的用法和作用？

答：try…except…except…[else…][finally…]

执行try下的语句，如果引发异常，则执行过程会跳到except语句。对每个except分支顺序尝试执行，如果引发的异常与except中的异常组匹配，执行相应的语句。如果所有的except都不匹配，则异常会传递到下一个调用本代码的最高层try代码中。

try下的语句正常执行，则执行else块代码。如果发生异常，就不会执行

如果存在finally语句，最后总是会执行。

8.Python中pass语句的作用是什么？

答：pass语句不会执行任何操作，一般作为占位符或者创建占位程序

15.如何在一个function里面设置一个全局的变量？

答：解决方法是在function的开始插入一个global声明：

**其它python面试题**

<https://www.cnblogs.com/klchang/p/4752441.html>

<https://www.cnblogs.com/acvc/p/4714848.html>

<https://www.cnblogs.com/tom-gao/p/6645859.html>

<https://www.cnblogs.com/Allen-rg/p/7693394.html>

**算法排序部分**

手写快排；堆排；几种常用排序的算法复杂度是多少；快排平均复杂度多少，最坏情况如何优化；   
手写：已知一个长度n的无序列表，元素均是数字，要求把所有间隔为d的组合找出来，你写的解法算法复杂度多少；   
手写：一个列表A=[A1，A2，…,An]，要求把列表中所有的组合情况打印出来；   
手写：用一行python写出1+2+3+…+10\*\*8 ；   
手写python：用递归的方式判断字符串是否为回文；   
单向链表长度未知，如何判断其中是否有环；   
单向链表如何使用快速排序算法进行排序；   
手写：一个长度n的无序数字元素列表，如何求中位数，如何尽快的估算中位数，你的算法复杂度是多少；   
如何遍历一个内部未知的文件夹（两种树的优先遍历方式）

**网络基础部分**

TCP/IP分别在模型的哪一层；   
socket长连接是什么意思；   
select和epoll你了解么，区别在哪；   
TCP UDP区别；三次握手四次挥手讲一下；   
TIME\_WAIT过多是因为什么；   
http一次连接的全过程：你来说下从用户发起request——到用户接收到response；   
http连接方式。get和post的区别，你还了解其他的方式么；   
restful你知道么；   
状态码你知道多少，比如200/403/404/504等等；

**数据库部分**

MySQL锁有几种；死锁是怎么产生的；   
为何，以及如何分区、分表；   
MySQL的char varchar text的区别；   
了解join么，有几种，有何区别，A LEFT JOIN B，查询的结果中，B没有的那部分是如何显示的（NULL）；   
索引类型有几种，BTree索引和hash索引的区别（我没答上来这俩在磁盘结构上的区别）；   
手写：如何对查询命令进行优化；   
NoSQL了解么，和关系数据库的区别；redis有几种常用存储类型；

**Linux部分**

讲一下你常用的Linux/git命令和作用；   
查看当前进程是用什么命令，除了文件相关的操作外，你平时还有什么操作命令；   
（因为我本人Linux本身就很水，只会基本的操作，所以这部分面试官也基本没怎么问。。反正问了就大眼瞪小眼呗）

**django项目部分**

都是让简单的介绍下你在公司的项目，不管是不是后端相关的，主要是要体现出你干了什么；   
你在项目中遇到最难的部分是什么，你是怎么解决的；   
你看过django的admin源码么；看过flask的源码么；你如何理解开源；   
MVC / MTV；   
缓存怎么用；   
中间件是干嘛的；   
CSRF是什么，django是如何避免的；XSS呢；   
如果你来设计login，简单的说一下思路；   
session和cookie的联系与区别；session为什么说是安全的；   
uWSGI和Nginx的作用；   
（我发现基本不问django实现细节相关的东西。。或者问也问的很少，哎，之前准备的方向完全错了）

洋洋洒洒写了一个小时，发现还是有很多细节问题都已经忘了。。哎，自己的笔记做的还是不行。不过这些问题如果你能都答上来，起码我觉得你应该可以去试试面试初级web后端了，**good luck!**

后记：发现一个整理的特别棒的github仓库，比我写的强多了。共享给大家~：https://github.com/taizilongxu/interview\_python