[**如何拿到半数面试公司Offer——我的Python求职之路**](http://www.cnblogs.com/Lands-ljk/p/5836492.html)

[**http://www.cnblogs.com/Lands-ljk/p/5836492.html**](http://www.cnblogs.com/Lands-ljk/p/5836492.html)

从八月底开始找工作，短短的 一星期多一些，面试了9家公司，拿到5份Offer，可能是因为我所面试的公司都是些创业性的公司吧，不过还是感触良多，因为学习Python的时间还很 短，没想到还算比较容易的找到了工作，就把这些天的面试经验和大家分享一下，希望为学习Python找工作的小伙伴们提供些许帮助。

笔者感觉面试最主要的两点：1.项目经验。 2.项目经验和招聘职位相符，这是最主要的，其他的都是锦上添花。

**自我介绍**

这是一道送分题，万年不变的第一个问题。不过有些小伙伴可能没有太在意，其实这个问题已经在面试官心中决定了你的去留意向。自我介绍的主要结构：个 人基本信息 + 基本技术构成 + 项目经验（具体项目以及在项目中的负责部分）+ 自我评价，其中的原则就是紧紧围绕招聘岗位的需求做介绍。在此之前要做好准备工作，看看招聘方具体需要什么方向的研发工程师。目前针对Python，拉勾 上的招聘多为自动化测试平台的设计与开发、数据的挖掘与清洗。单纯的web开发好像还没有，所以web方向的同学注意，多和运维以及自动化方面靠拢。

**二段式询问**

在面试的过程当中，在面试官提出问题的时候，往往会就问题本身引申出较深层次的问题。比如：你使用过with语句吗？我的回答是：with语句经常 适用于对资源进行访问的场合，确保在访问的过程中不管是否发生异常都会指执行必要的清理操作，比如文件的自动关闭以及线程中锁的自动获取与释放。面试官紧 接着问，那你知道为什么with语句能够使文件正确关闭，一下子把我问闷了，只能依稀记得with语句会开辟出一块独立环境来执行文件的访问，类似沙盒机 制。面试官对这个答案不置可否，算是勉强通过了。所以知其然更要知其所以然。在平时的学习中，多问一个为什么，面试的时候就不会太被动。

**不要给自己挖坑**

确保你在回答面试官的过程中，回答中的每个知识点都了然于胸，不然被问住，是很难堪的。我在回答web安全问题时，顺嘴说了SQL注入，面试官说既 然提到了SQL注入，那么你讲讲它的原理及解决方法吧！丢脸的是我竟然把XSS跨站注入攻击和SQL注入搞混了，场面也是有点尴尬。所以斟酌你说的每一句 话，聪明点的同学还可以引导面试官，让他问出自己想要被问的问题。

**必问到Redis，高并发解决办法**

面试了好多家公司，必然问道Redis了解多少，高并发的解决办法。笔者回答的都不是很好。

**这一年你学习了什么新的技能**

这是面试官在考察你是否对于新鲜技术抱有极大热忱。面试我的面试官无一例外都问到了这个问题。他们都希望能找一个不断学习，开括创新的年轻人。多浏览最新的技术资讯，选择一方面自己感兴趣的领域。

**你会选择创业公司还是像BAT那样的大公司，为什么？**

当然是看招聘方属于哪一个公司啦，不过问这种问题的一般都是创业公司。答案无非是：挑战大，享受挑战；创业公司具有无限成功的可能性，想随公司一起成长；

**为什么你要从上一家公司离职？**

这也是一个必问问题，找一个比较正当的理由，不要说什么公司零食太多胖了20斤，公司周别附近的外卖都吃腻了，真的别这样说…主要原则就是不要对前 公司抱有怨言，BOSS朝令夕改，PM不靠谱什么的，多寻找自身原因：公司发展比较稳定，但我还年轻，希望有更大的挑战和更多的学习机会。像这样就可以。

**描述一下你的上一家公司**

这个问题问到的几率不太大，不过也还是有三家公司问到过，招聘方主要想从上一家公司的具体经营规模以及主营业务来定位你的水平，知道招聘方的目的就可以从容应答。

**技术性问题**

非技术性的问题就是以上这么多，作为参考稍加准备，面试的时候就能对答如流。下面讲一下在面试中的技术性问题。个人感觉技术性的问题面试官问的没有特别多，一般考察2-3个，由浅到深。

1. 简述函数式编程

在函数式编程中，函数是基本单位，变量只是一个名称，而不是一个存储单元。除了匿名函数外，Python还使用fliter(),map(),reduce(),apply()函数来支持函数式编程。

1. 什么是匿名函数，匿名函数有什么局限性

匿名函数，也就是lambda函数，通常用在函数体比较简单的函数上。匿名函数顾名思义就是函数没有名字，因此不用担心函数名冲突。不过Python对匿名函数的支持有限，只有一些简单的情况下可以使用匿名函数。

1. 如何捕获异常，常用的异常机制有哪些？

如果我们没有对异常进行任何预防，那么在程序执行的过程中发生异常，就会中断程序，调用python默认的异常处理器，并在终端输出异常信息。

try...except...finally语句:当try语句执行时发生异常，回到try语句层，寻找后面是否有except语句。找到 except语句后，会调用这个自定义的异常处理器。except将异常处理完毕后，程序继续往下执行。finally语句表示，无论异常发生与 否，finally中的语句都要执行。

assert语句：判断assert后面紧跟的语句是True还是False，如果是True则继续执行print，如果是False则中断程序，调用默认的异常处理器，同时输出assert语句逗号后面的提示信息。

with语句：如果with语句或语句块中发生异常，会调用默认的异常处理器处理，但文件还是会正常关闭。

1. copy()与deepcopy()的区别

copy是浅拷贝，只拷贝可变对象的父级元素。 deepcopy是深拷贝，递归拷贝可变对象的所有元素。

1. 函数装饰器有什么作用（常考）

装饰器本质上是一个Python函数，它可以让其他函数在不需要做任何代码变动的前提下增加额外功能，装饰器的返回值也是一个函数对象。它经常用于 有切面需求的场景，比如：插入日志、性能测试、事务处理、缓存、权限校验等场景。有了装饰器，就可以抽离出大量与函数功能本身无关的雷同代码并继续重用。

1. 简述Python的作用域以及Python搜索变量的顺序

Python作用域简单说就是一个变量的命名空间。代码中变量被赋值的位置，就决定了哪些范围的对象可以访问这个变量，这个范围就是变量的作用域。 在Python中，只有模块（module），类（class）以及函数（def、lambda）才会引入新的作用域。Python的变量名解析机制也称 为 LEGB 法则：本地作用域（Local）→当前作用域被嵌入的本地作用域（Enclosing locals）→全局/模块作用域（Global）→内置作用域（Built-in）

1. 新式类和旧式类的区别,如何确保使用的类是新式类

为了统一类(class)和类型(type)，python在2.2版本引进来新式类。在2.1版本中，类和类型是不同的。

为了确保使用的是新式类，有以下方法：

放在类模块代码的最前面 \_\_metaclass\_\_ = type  
从内建类object直接或者间接地继承  
在python3版本中，默认所有的类都是新式类。

1. 简述\_\_new\_\_和\_\_init\_\_的区别

创建一个新实例时调用\_\_new\_\_,初始化一个实例时用\_\_init\_\_,这是它们最本质的区别。

new方法会返回所构造的对象，init则不会.

new函数必须以cls作为第一个参数，而init则以self作为其第一个参数.

1. Python垃圾回收机制(常考)

Python GC主要使用引用计数（reference counting）来跟踪和回收垃圾。在引用计数的基础上，通过“标记-清除”（mark and sweep）解决容器对象可能产生的循环引用问题，通过“分代回收”（generation collection）以空间换时间的方法提高垃圾回收效率。

1 引用计数

PyObject是每个对象必有的内容，其中ob\_refcnt就是做为引用计数。当一个对象有新的引用时，它的ob\_refcnt就会增加，当引用它的对象被删除，它的ob\_refcnt就会减少.引用计数为0时，该对象生命就结束了。

优点:

简单 实时性 缺点:

维护引用计数消耗资源 循环引用

2 标记-清除机制

基本思路是先按需分配，等到没有空闲内存的时候从寄存器和程序栈上的引用出发，遍历以对象为节点、以引用为边构成的图，把所有可以访问到的对象打上标记，然后清扫一遍内存空间，把所有没标记的对象释放。

3 分代技术

分代回收的整体思想是：将系统中的所有内存块根据其存活时间划分为不同的集合，每个集合就成为一个“代”，垃圾收集频率随着“代”的存活时间的增大而减小，存活时间通常利用经过几次垃圾回收来度量。

Python默认定义了三代对象集合，索引数越大，对象存活时间越长。

1. [Python中的@property有什么作用?如何实现成员变量的只读属性](mailto:Python%E4%B8%AD%E7%9A%84@property%E6%9C%89%E4%BB%80%E4%B9%88%E4%BD%9C%E7%94%A8?%E5%A6%82%E4%BD%95%E5%AE%9E%E7%8E%B0%E6%88%90%E5%91%98%E5%8F%98%E9%87%8F%E7%9A%84%E5%8F%AA%E8%AF%BB%E5%B1%9E%E6%80%A7)？

@property装饰器就是负责把一个方法变成属性调用，通常用在属性的get方法和set方法，[通过设置@property可以实现实例成员变量的直接访问](mailto:%E9%80%9A%E8%BF%87%E8%AE%BE%E7%BD%AE@property%E5%8F%AF%E4%BB%A5%E5%AE%9E%E7%8E%B0%E5%AE%9E%E4%BE%8B%E6%88%90%E5%91%98%E5%8F%98%E9%87%8F%E7%9A%84%E7%9B%B4%E6%8E%A5%E8%AE%BF%E9%97%AE)，又保留了参数的检查。另外通过设置get方法而不定义set方法可以实现成员变量的只读属性。

1. \*args and \*\*kwargs

\*args代表位置参数，它会接收任意多个参数并把这些参数作为元组传递给函数。\*\*kwargs代表的关键字参数，允许你使用没有事先定义的参数名，另外，位置参数一定要放在关键字参数的前面。

1. 有用过with statement吗？它的好处是什么？具体如何实现？

with语句适用于对资源进行访问的场合，确保不管使用过程中是否发生异常都会执行必要的“清理”操作，释放资源，比如文件使用后自动关闭、线程中锁的自动获取和释放等。

1. what will be the output of the code below? explain your answer
2. def extend\_list(val, list=[]):
3. list.append(val)
4. return list
5. list1 = extend\_list(10)
6. list2 = extend\_list(123, [])
7. list3 = extend\_list('a')
8. print(list1) # list1 = [10, 'a']
9. print(list2) # list2 = [123, []]
10. print(list3) # list3 = [10, 'a']
11. class Parent(object):
12. x = 1
13. class Child1(Parent):
14. pass
15. class Child2(Parent):
16. pass
17. print(Parent.x, Child1.x, Child2.x) # [1,1,1]
18. Child1.x = 2
19. print(Parent.x, Child1.x, Child2.x) # [1,2,1]
20. Partent.x = 3

print(Parent.x, Child1.x, Child2.x) # [3,2,3]

1. 在一个二维数组中，每一行都按照从左到右递增的顺序排序，每一列都按照从上到下递增的顺序排序。请完成一个函数，输入这样的一个二维数组和一个整数，判断数组中是否含有该整数。
2. arr = [[1,4,7,10,15], [2,5,8,12,19], [3,6,9,16,22], [10,13,14,17,24], [18,21,23,26,30]]
3. def getNum(num, data=None):
4. while data:
5. if num > data[0][-1]:
6. del data[0]
7. print(data)
8. getNum(num, data=None)
9. elif num < data[0][-1]:
10. data = list(zip(\*data))
11. del data[-1]
12. data = list(zip(\*data))
13. print(data)
14. getNum(num, data=None)
15. else:
16. return True
17. data.clear()
18. return False
19. if \_\_name\_\_ == '\_\_main\_\_':

print(getNum(18, arr))

1. 获取最大公约数、最小公倍数
2. a = 36
3. b = 21
4. def maxCommon(a, b):
5. while b: a,b = b, a%b
6. return a
7. def minCommon(a, b):
8. c = a\*b
9. while b: a,b = b, a%b
10. return c//a
11. if \_\_name\_\_ == '\_\_main\_\_':
12. print(maxCommon(a,b))

print(minCommon(a,b))

1. 获取中位数
2. def median(data):
3. data.sort()
4. half = len(data) // 2
5. return (data[half] + data[~half])/2
6. l = [1,3,4,53,2,46,8,42,82]
7. if \_\_name\_\_ == '\_\_main\_\_':

print(median(l))

1. 输入一个整数，输出该数二进制表示中1的个数。其中负数用补码表示。
2. def getOneCount(num):
3. if num > 0:
4. count = b\_num.count('1')
5. print(b\_num)
6. return count
7. elif num < 0:
8. b\_num = bin(~num)
9. count = 8 - b\_num.count('1')
10. return count
11. else:
12. return 8
13. if \_\_name\_\_ == '\_\_main\_\_':
14. print(getOneCount(5))
15. print(getOneCount(-5))

print(getOneCount(0))

以上就是我面试过程中所被问到的问题，算法题还是比较少的，也只有2家公司要求写算法，数据结构似乎被问到的不是特别多，就问到了一个B+树的结构。数据库问到的是索引相关的优化。稍微有些基础的都能回答上来，但是最好可以深层次的探讨一下。

本文只做抛砖引玉之用，有些见解还不是特别成熟，希望可以为学习Python找工作的伙伴们提供一些帮助，面试过程当中最重要的一点是放平心态，求职过程是双方的，不需要太过紧张，把自己掌握的知识充分表达出来就好。只要你是匹千里马，迟早会被伯乐牵出来遛一遛的。