注：答案一般在网上都能够找到。  
1.对if \_\_name\_\_ == 'main'的理解陈述  
2.python是如何进行内存管理的？  
3.请写出一段Python代码实现删除一个list里面的重复元素  
4.Python里面如何拷贝一个对象？（赋值，浅拷贝，深拷贝的区别）  
5.介绍一下except的用法和作用？  
6.Python中\_\_new\_\_与\_\_init\_\_方法的区别  
7.常用的网络数据爬取方法  
8.遇到过得反爬虫策略以及解决方法  
9.urllib 和 urllib2 的区别  
10.设计一个基于session登录验证的爬虫方案  
11.列举网络爬虫所用到的网络数据包，解析包  
12.熟悉的爬虫框架  
13.Python在服务器的部署流程，以及环境隔离  
14.Django 和 Flask 的相同点与不同点，如何进行选择？  
15.写一个Python中的单例模式  
16.Linux部署服务脚本命令(包括启动和停止的shell脚本)  
17.你用过多线程和异步嘛？除此之外你还用过什么方法来提高爬虫效率？  
18.POST 与 GET的区别

1. 对if \_\_name\_\_ == 'main'的理解陈述  
   \_\_name\_\_是当前模块名，当模块被直接运行时模块名为\_\_main\_\_，也就是当前的模块，当模块被导入时，模块名就不是\_\_main\_\_，即代码将不会执行。
2. python是如何进行内存管理的？  
   a、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，Python内部记录了对象有多少个引用，即引用计数，当对象被创建时就创建了一个引用计数，当对象不再需要时，这个对象的引用计数为0时，它被垃圾回收。  
   b、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。  
   c、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。
3. 请写出一段Python代码实现删除一个list里面的重复元素
4. # 1.使用set函数
5. **list** = [1, 3, 4, 5, 51, 2, 3]
6. set(**list**)
7. # 2.使用字典函数，
8. >>> a = [1, 2, 4, 2, 4, 5, 6, 5, 7, 8, 9, 0]
9. >>> b = {}
10. >>> b = b.fromkeys(a)
11. >>> c = **list**(b.keys())

>>> c

1. 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}
2. 介绍一下except的用法和作用？  
   try…except…except…else…  
   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。
3. Python中\_\_new\_\_与\_\_init\_\_方法的区别  
   \_\_new\_\_:它是创建对象时调用，会返回当前对象的一个实例，可以用\_\_new\_\_来实现单例  
   \_\_init\_\_:它是创建对象后调用，对当前对象的一些实例初始化，无返回值
4. 常用的网络数据爬取方法
   * 正则表达式
   * Beautiful Soup
   * Lxml
5. 遇到过得反爬虫策略以及解决方法  
   1.通过headers反爬虫  
   2.基于用户行为的发爬虫：(同一IP短时间内访问的频率)  
   3.动态网页反爬虫(通过ajax请求数据，或者通过JavaScript生成)  
   4.对部分数据进行加密处理的(数据是乱码)  
   解决方法：  
   对于基本网页的抓取可以自定义headers,添加headers的数据  
   使用多个代理ip进行抓取或者设置抓取的频率降低一些，  
   动态网页的可以使用selenium + phantomjs 进行抓取  
   对部分数据进行加密的，可以使用selenium进行截图，使用python自带的pytesseract库进行识别，但是比较慢最直接的方法是找到加密的方法进行逆向推理。
6. urllib 和 urllib2 的区别  
   urllib 和urllib2都是接受URL请求的相关模块，但是urllib2可以接受一个Request类的实例来设置URL请求的headers，urllib仅可以接受URL。urllib不可以伪装你的User-Agent字符串。  
   urllib提供urlencode()方法用来GET查询字符串的产生，而urllib2没有。这是为何urllib常和urllib2一起使用的原因。
7. 设计一个基于session登录验证的爬虫方案
8. 列举网络爬虫所用到的网络数据包，解析包
   * 网络数据包 urllib、urllib2、requests
   * 解析包 re、xpath、beautiful soup、lxml
9. 熟悉的爬虫框架  
   Scrapy框架 根据自己的实际情况回答
10. Python在服务器的部署流程，以及环境隔离
11. Django 和 Flask 的相同点与不同点，如何进行选择？
12. 写一个Python中的单例模式
13. **class** **Singleton**(object):
14. \_instance = **None**
15. **def** **\_\_new\_\_**(cls, \*args, \*\*kw):
16. **if** **not** cls.\_instance:
17. cls.\_instance = super(Singleton, cls).\_\_new\_\_(cls, \*args, \*\*kw)
18. **return** cls.\_instance
19. **class** **MyClass**(Singleton):
20. a = 1
22. one = MyClass()
23. two = MyClass()
24. id(one) = id(two)
25. **>>> True**
26. Linux部署服务脚本命令(包括启动和停止的shell脚本)
27. 你用过多线程和异步嘛？除此之外你还用过什么方法来提高爬虫效率？
    * scrapy-redis 分布式爬取
    * 对于定向爬取可以用正则取代xpath
28. POST与 GET的区别
    * GET数据传输安全性低，POST传输数据安全性高，因为参数不会被保存在浏览器历史或web服务器日志中；
    * 在做数据查询时，建议用GET方式；而在做数据添加、修改或删除时，建议用POST方式；
    * GET在url中传递数据，数据信息放在请求头中；而POST请求信息放在请求体中进行传递数据；
    * GET传输数据的数据量较小，只能在请求头中发送数据，而POST传输数据信息比较大，一般不受限制；
    * 在执行效率来说，GET比POST好
29. 什么是lambda函数？它有什么好处?  
    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：首要用途是指点短小的回调函数
30. lambda [arguments]:expression
31. **>>**> a=lambdax,y:x+y

**>>**> a(3,11)