作者：阿涵酱  
链接：https://www.zhihu.com/question/54513391/answer/945646205  
来源：知乎  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

**【纯干货！！！】花费了整整3天，整理出来的全网最实用Python面试大全，一共30道题目+答案的纯干货，**对于答主的辛勤整理，希望大家多多支持，建议 **点赞！！收藏！！**

**长文警告，全文共12000+字，涵盖Python面试可能遇到的所有问题，希望对大家有帮助，不过大家最好不要硬背，实战大于理论。祝大家面试顺利！**

对于机器学习算法工程师而言，Python是不可或缺的语言，它的优美与简洁令人无法自拔。那么你了解过Python编程面试题吗？从Python基础到网页爬虫你是否能全方位Hold住？本文主要从 Python 基础、高级语句、网页应用、数据库和测试等角度出发，可只关注自己需要的领域。

**1、谈谈对 Python 和其他语言的区别？**

**答：**Python 是一门语法简洁优美,功能强大无比,应用领域非常广泛,具有强大完备的第三方库，他是一门强类型的可移植、可扩展，可嵌入的解释型编程语言，属于动态语言。

* **和Java相比:**Python比Java要简单.Python是函数为一等公民的语言,而Java是类为一等公民的语言.Python是弱类型语言,而Java是强类型语言。
* **和C相比：**对于使用:Python的类库齐全并且使用简洁,很少代码实现的功能用C可能要很复杂。对于速度:Python的运行速度相较于C,绝对是很慢了.Python和CPython解释器都是C语言编写。

2、**谈谈Python 的特点和优点是什么?**

**答：**Python 是一门动态解释性的强类型定义语言：编写时无需定义变量类型；运行时变量类型强制固定；无需编译，在解释器环境直接运行。

* **解释性：**一个用编译型语言（如 C 或 C++）写的程序，可以从源文件转换到一个计算机使用的语言。这个过程主要通过编译器完成。当运行程序的时候，我们可以把程序从硬盘复制到内存中并且运行。而 Python 语言写的程序，则不需要编译成二进制代码，可以直接从源代码运行程序。在计算机内部，由 Python 解释器把源代码转换成字节码的中间形式，然后再把它翻译成计算机使用的机器语言并运行。
* **动态性：**在运行时可以改变其结构的语言 ：例如新的函数、对象、甚至代码可以被引进，已有的函数可以被删除或是其他结构上的变化。动态语言目前非常具有活力。Python便是一个动态语言，除此之外如 PHP 、 Ruby 、 JavaScript 等也都属于动态语言 。
* **面向对象：**面向对象编程简单来说就是基于对 类 和 对象 的使用，所有的代码都是通过类和对象来实现的编程就是面向对象编程！面向对象的三大特性：封装、继承、多态
* **语法简洁：**Python 是一种代表简单注意思想的语言，阅读一个良好的 Python 程序，即使是在 Python 语法要求非常严格的大环境下，给人的感觉也像是在读英语段落一样。换句话说，Python 编程语言最大的优点之一，是其具有伪代码的特质，它可以让我们在开发 Python 程序时，专注于解决问题，而不是搞明白语言本身。
* **开源：**Python 是开源的，简单地理解就是，用户使用 Python 进行开发和发布自己编写的程序，不需要支付任何费用，也不用担心版权问题，即使作为商业用途，Python 也是免费的。开源正在成为软件行业的一种发展趋势，现在有很多商业软件公司都开始将自己的产品变成开源的（例如 Java）。也许，Python 的开源正是它如此优秀的原因之一，因为会有这么一群人，他们希望看到一个更加优秀的 Python，从而为了这个目标，不断地对 Python 进行创造，不断地改进。
* **可扩展性：**Python 的可扩展性体现为它的模块，Python 具有脚本语言中最丰富和强大的类库，这些类库覆盖了文件 I/O、GUI、网络编程、数据库访问、文本操作等绝大部分应用场景。

**3、说说Python解释器种类以及特点？**

**答：**Python是一门解释器语言，代码想运行，必须通过解释器执行，Python存在多种解释器，分别基于不同语言开发，每个解释器有不同的特点，但都能正常运行Python代码。

**Python解释器主要有以下几个:**

* **CPython：**官方版本的解释器：CPython。这个解释器是用C语言开发的，所以叫CPython。在命令行下运行python就是启动CPython解释器。CPython是使用最广且被的Python解释器。
* **IPython：**IPython是基于CPython之上的一个交互式解释器，也就是说，IPython只是在交互方式上有所增强，但是执行Python代码的功能和CPython是完全一样的。CPython用>>>作为提示符，而IPython用In [序号]:作为提示符。
* **PyPy：**PyPy是另一个Python解释器，它的目标是执行速度。PyPy采用JIT技术，对Python代码进行动态编译（注意不是解释），所以可以显著提高Python代码的执行速度。
* 绝大部分Python代码都可以在PyPy下运行，但是PyPy和CPython有一些是不同的，这就导致相同的Python代码在两种解释器下执行可能会有不同的结果。如果你的代码要放到PyPy下执行，就需要了解PyPy和CPython的不同点。
* **Jython：**Jython是运行在Java平台上的Python解释器，可以直接把Python代码编译成Java字节码执行。
* **IronPython：**IronPython和Jython类似，只不过IronPython是运行在微软.Net平台上的Python解释器，可以直接把Python代码编译成.Net的字节码。

在这些Python解释器中，使用广泛的是**CPython** 。

**4、说说Python面向对象三大特性?**

**答：**Python是一门面向对象的语言。面向对象都有三大特性：封装、继承、多态。

**下面分别来说说这三大特性:**

* **封装：**隐藏对象的属性和实现细节，仅对外提供公共访问方式。在python中用双下划线开头的方式将属性设置成私有的 。好处：1. 将变化隔离；2. 便于使用；3. 提高复用性；4. 提高安全性。
* **继承：**继承是一种创建新类的方式，在python中，新建的类可以继承一个或多个父类，父类又可称为基类或超类，新建的类称为派生类或子类。即一个派生类继承基类的字段和方法。继承也允许把一个派生类的对象作为一个基类对象对待。例如，有这样一个设计：一个Dog类型的对象派生自Animal类，这是模拟"是一个（is-a）"关系 。

python中类的继承分为：单继承和多继承

class ParentClass1: #定义父类

class ParentClass2: #定义父类

class SubClass1(ParentClass1): #单继承，基类是ParentClass1，派生类是SubClass

class SubClass2(ParentClass1,ParentClass2): #python支持多继承，用逗号分隔开多个继承的类

* **多态：**一种事物的多种体现形式，函数的重写其实就是多态的一种体现 。Python中，多态指的是父类的引用指向子类的对象 。实现多态的步骤：1、定义新的子类，2、重写对应的父类方法，3、使用子类的方法直接处理,不调用父类的方法。多态的好处：（1）增加了程序的灵活性（2）增加了程序可扩展性

**5、说说 Python 中有几种数据类型？**

**答：**Python 中主要有8种数据类型：number（数字）、string（字符串）、list（列表）、tuple（元组）、dict（字典）、set（集合）、Boolean（布尔值）、None（空值）。

**6、说说Python中xrange和range的区别？**

**答：**range()和xrange()都是在循环中使用，输出结果一样。

* range()返回的是一个list对象，而xrange返回的是一个生成器对象(xrange object)。
* xrange()则不会直接生成一个list，而是每次调用返回其中的一个值，内存空间使用极少。因而性能非常好，所以尽量用xrange吧。

在python3 中没有xrange，只有range。range和python2 中的xrange()一样。

**7、Python变量、函数、类的命名规则？**

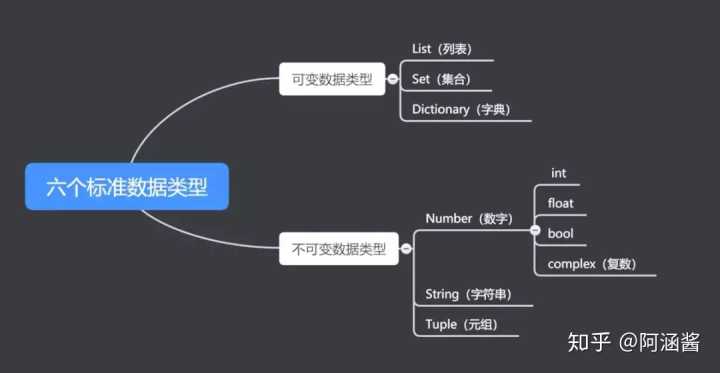
**答：**

* (1)不能以数字开头，不能出现中文。
* (2)命名以字母开头，包含数字，字母（区分大小写），下划线。
* (3)不能包含关键字，见名知意。

**8、说说Python可变与不可变数据类型的区别？**

**答：**Python中看可变与不可变数据类型，主要是看变量所指向的内存地址处的值是否会改变 。 Python 的六种标准数据类型：数字、字符串、列表、元组、字典、集合。

* 不可变数据（3个）：Number（数字）、String（字符串）、Tuple（元组）。
* 可变数据（3个）：List（列表）、Dictionary（字典）、Set（集合）。



**9、列举几个Python中的标准异常类？**



**10、说说Python中迭代器和生成器的区别？**

**答：**Python中生成器能做到迭代器能做的所有事，而且因为自动创建了\_\_iter\_\_()和next()方法，生成器显得特别简洁，而且生成器也是高效的，使用生成器表达式取代列表解析，同时节省内存。除了创建和保持程序状态的自动生成，当发生器终结时，还会自动跑出StopIterration异常。

* 列表、元组、字典、字符串都是可迭代对象。
* 数字、布尔值都是不可迭代的。

**11、说说Python字典以及基本操作？**

**答：**字典是 Python 提供的一种常用的数据结构，主要用于存放具有映射关系的数据 。比如保存某班同学的成绩单数据，张三：95分，李四：70分，王五：100分 ... ，因为姓名和成绩是有关联的，所以不能单独用两个列表来分别保存，这时候用字典来存储，再合适不过了 。

字典是一种可变的容器模型，它是通过一组键（key）值（value）对组成，这种结构类型通常也被称为映射，或者叫关联数组，也有叫哈希表的。每个key-value之间用“:”隔开，每组用“,”分割，整个字典用“{}”括起来 ，格式如下所示：

dictionary = {key1 : value1, key2 : value2 }

定义字典时，键前值后，键必须唯一性，值可以不唯一，如果键有相同，值则取最后一个；

值可以是任何的数据类型，但是键必须是不可变的数据类型（数字、字符串、元组）。想要访问字典中的值，只需要将键放入方括号里，如果用字典里没有的键访问数据，会输出错误 。

**12、说说Python种有几种字符串格式化？**

**答：**Python字符串格式化主要有两种方式：分别为占位符(%)和format方式 。文末还有2种要介绍，所以总共有4种 。

其中，占位符(%)方式比较老，而format方式是比较先进的，目前两者共存。占位符方式在Python2.x中用的比较广泛，随着Python3.x的使用越来越广，format方式使用的更加广泛。

**13、说说Python多线程与多进程的区别?**

* 1、多线程可以共享全局变量，多进程不能
* 2、多线程中，所有子线程的进程号相同；多进程中，不同的子进程进程号不同
* 3、线程共享内存空间；进程的内存是独立的
* 4、同一个进程的线程之间可以直接交流；两个进程想通信，必须通过一个中间代理来实现
* 5、创建新线程很简单；创建新进程需要对其父进程进行一次克隆
* 6、一个线程可以控制和操作同一进程里的其他线程；但是进程只能操作子进程

两者最大的不同在于：在多进程中，同一个变量，各自有一份拷贝存在于每个进程中，互不影响；而多线程中，所有变量都由所有线程共享 。

**14、说说Python中HTTP常见响应状态码？**

**答：**http协议是超文本传输协议，是用于从万维网服务器传输文本到本地浏览器的传送协议，是基于tcp/ip通信协议来传输数据的。

HTTP状态码（HTTP Status Code）是用以表示网页服务器超文本传输协议响应状态的3位数字代码。它由 RFC 2616 规范定义的，并得到 RFC 2518、RFC 2817、RFC 2295、RFC 2774 与 RFC 4918 等规范扩展。所有状态码的第一个数字代表了响应的五种状态之一。

**15、说说Python中猴子补丁是什么？**

**答：**在Ruby、Python等动态编程语言中，猴子补丁仅指在运行时动态改变类或模块，为的是将第三方代码打补丁在不按预期运行的bug或者feature上 。在运行时动态修改模块、类或函数，通常是添加功能或修正缺陷。猴子补丁在代码运行时内存中发挥作用，不会修改源码，因此只对当前运行的程序实例有效。因为猴子补丁破坏了封装，而且容易导致程序与补丁代码的实现细节紧密耦合，所以被视为临时的变通方案，不是集成代码的推荐方式。

**16、说说Python中的垃圾回收机制？**

**答：**垃圾回收机制（Garbage Collection:GC）基本是所有高级语言的标准配置之一了，在一定程度上，能优化编程语言的数据处理效率和提高编程软件开发软件的安全性能 。在python中的垃圾回收机制主要是以**引用计数**为主要手段以**标记清除**和**隔代回收**机制为辅的手段 。可以对内存中无效数据的自动管理！

**17、说说Python中有几种办法交换两个变量的值？**

**答：**交换两个变量的值方法，这个面试题如果只写一种当然很简单，没什么可以说的。今天这个面试是问大家有几种办法来实现交换两个变量的值 。

**方法一：**通过新添加中间变量temp的方式，这个方法是最简单的，每个语言都适用。

def swap(a,b):

temp = a

a = b

b = temp

print(a,b)

**方法二：**Python独有的方法，一行代码就能搞定，直接将两个变量放到元组中 。

def swap2(a,b):

a,b = b,a

print(a,b)

**方法三：**这个方法，是不是很少人想到了，采用加减法来交换 。我们不考虑效率，能达到交换的效果就行 。

def swap3(a, b):

a = a + b

b = a - b

a = a - b

print(a, b)

**方法四：**采用异或运算，这个是不是看起来比较高大上。通过按位异或运算来交换两变量的值，可以减少变量的定义，同时减少计算机对代码的解析时间。按位异或运算即计算机会先把十进制数转化为二进制数，并对二进制数进行从右到左用从1开始编数，然后比较两个二进制数值相同位置的数，如果相同结果为0，不同时结果为1 。 "1^1=0 1^0=1 0^0=0"

如： 1010

1111

则结果为 0101

def swap4(a,b):

a = a ^ b

b = a ^ b

a = a ^ b

print(a,b)

**18、说说Python中的6种位运算符？**

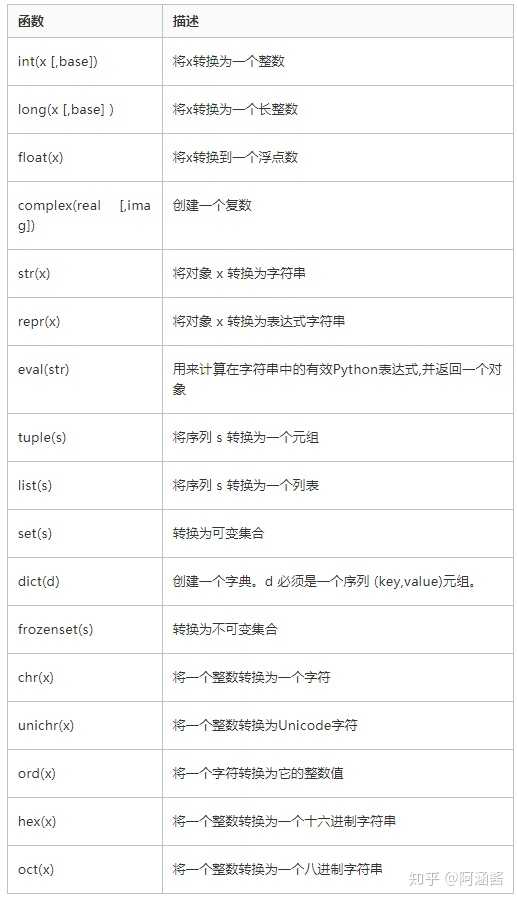
**答：**在Python中，按位运算符有左移运算符（<<）、右移运算符（>>）、按位与运算（&）、按位或运算（|）、按位取反运算（~）、异或运算符，其中按位取反运算符为单目运算符 。



**19、说说Python中的类型转换有哪些？**

**答：**在Python处理数据时，不可避免的要使用数据类型之间的转换。简单的诸如int、float、string之间的转换；更有数组array、列表list之间的转换。

以下是几个内置的函数可以执行数据类型之间的转换。这些函数返回一个新的对象，表示转换的值。



**20、Python中实现二分查找的2种方法？**

**答：**在Python实现二分查找法有两种方法，分别**用循环和递归方式**。

二分查找法：搜索过程从数组的中间元素开始，如果中间元素正好是要查找的元素，则搜索过程结束；如果某一特定元素大于或者小于中间元素，则在数组大于或小于中间元素的那一半中查找，而且跟开始一样从中间元素开始比较。如果在某一步骤数组为空，则代表找不到。这种搜索算法每一次比较都使搜索范围缩小一半。注意如果要想使用二分查找，前提必须是元素有序排列 。

**循环方式**

def binary\_search\_2(alist,item):

"""二分查找---循环版本"""

n = len(alist)

first = 0

last = n-1

while first <= last:

mid = (first + last)//2

if alist[mid] ==item:

return True

elif item < alist[mid]:

last = mid - 1

else:

first = mid + 1

return False

if \_\_name\_\_ == "\_\_main\_\_":

a = [1,5,6,10,11,13,18,37,99]

sorted\_list\_21 = binary\_search\_2(a, 18)

print(sorted\_list\_21) //True

sorted\_list\_22 = binary\_search\_2(a, 77)

print(sorted\_list\_22) //False

**递归方式**

def binary\_search(alist,item):

"""二分查找---递归实现"""

n = len(alist)

if n > 0:

mid = n//2 #数组长度的一半中间下标

if item == alist[mid] :

return True #查找成功

elif item < alist[mid]:

return binary\_search(alist[:mid],item)

else:

return binary\_search(alist[mid+1:], item)

else :

return False #失败

if \_\_name\_\_ == "\_\_main\_\_":

a = [1,5,6,10,11,13,18,37,99]

# print(a)

sorted\_list\_11 = binary\_search(a,37)

print(sorted\_list\_11)//True

sorted\_list\_12= binary\_search(a, 88)

print(sorted\_list\_12)//False

**21、说说Python中的lambda表达式？**

**答：**在Python中lambda表达式也叫匿名函数，即函数没有具体的名称。lambda表达式是Python中一类特殊的定义函数的形式，使用它可以定义一个匿名函数。与其它语言不同，Python的lambda表达式的函数体只能有单独的一条语句，也就是返回值表达式语句。

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 。lambda所表示的匿名函数的内容应该是很简单的，如果复杂的话，就重新定义一个函数了。lambda 表达式允许在一行代码中创建一个函数并传递。

lambda表达式在Python中的优点和缺点: 一方面，Lambda函数的减少了代码的行数，方便又简洁。另一方面，Lambda表达式有诸多限制，不能使用复杂逻辑。

add = lambda x, y : x+y

num =add(1,2)

print("x+y=",num)

# 结果为:x+y=3

**22、说说Python中的反射？**

**答：**在反射机制就是在运行时，动态的确定对象的类型，并可以通过字符串调用对象属性、方法、导入模块，是一种基于字符串的事件驱动。通过字符串的形式，去模块寻找指定函数，并执行。利用字符串的形式去对象（模块）中操作（查找/获取/删除/添加）成员。

Python是一门解释型语言，因此对于反射机制支持很好。在Python中支持反射机制的函数有getattr()、setattr()、delattr()、exec()、eval()、\_\_import\_\_，这些函数都可以执行字符串。

**23、说说Python删除list里的重复元素有几种方法？**

**答：**在Python中主要有5种方式，还没看答案，你能想起几种呢，面试笔试题经常碰到的一道题 。

**使用set函数：**set是定义集合的,无序，非重复

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

print(list(set(numList)))

#[1, 2, 3, 4, 5]

**先把list重新排序**，然后从list的最后开始扫描

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

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) #[1, 2, 4, 5]

**使用字典函数**

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

b={}

b=b.fromkeys(a)

c=list(b.keys())

print(c) #[1, 2, 4]

**append方式**

def delList(L):

L1 = []

for i in L:

if i not in L1:

L1.append(i)

return L1

print(delList([1, 2, 2, 3, 3, 4, 5])) #[1, 2, 3, 4, 5]

**count + remove方式**

def delList(L):

for i in L:

if L.count(i) != 1:

for x in range((L.count(i) - 1)):

L.remove(i)

return L

print(delList([1, 2, 2, 3, 3, 4]))#[1, 2, 3, 4]

**24、说说Python中的\_\_new\_\_和\_\_init\_\_的区别？**

**答：**在Python中\_\_new\_\_和\_\_init\_\_具有不同的功能。并且对于Python的新类和旧类而言功能也不同。

* \_\_new\_\_是在实例创建之前被调用的，因为它的任务就是创建实例然后返回该实例对象，是个静态方法。
* \_\_init\_\_是当实例对象创建完成后被调用的，然后设置对象属性的一些初始值，通常用在初始化一个类实例的时候。是一个实例方法。
* 主要区别在于：\_\_new\_\_是用来创造一个类的实例的，而\_\_init\_\_是用来初始化一个实例的。

**25、说说Python中的help()和dir()函数？**

**答：**在Python中help()和dir()这两个函数都可以从Python解释器直接访问，并用于查看内置函数的合并转储。

* help()函数：help()函数用于显示文档字符串，还可以查看与模块，关键字，属性等相关的使用信息。
* dir()函数：dir()函数可以列出指定类或模块包含的全部内容（包括函数、方法、类、变量等）

**26、说说提高Python运行效率的技巧？**

**答：**不喜欢Python的人经常会吐嘈Python运行太慢。下面给大家介绍几种种提高python执行效率的方法 。

* **使用局部变量：**尽量使用局部变量代替全局变量：便于维护，提高性能并节省内存。一方面可以提高程序性能，局部变量查找速度更快;另一方面可用简短标识符替代冗长的模块变量，提高可读性。
* **使用较新的Python版本：**Python已经更新了很多个版本，每个版本的Python都会包含优化内容，使其运行速度优于之前的版本，所以大家记得经常更新版本哦！
* **先编译后调用：**使用eval()、exec()函数执行代码时，最好调用代码对象(提前通过compile()函数编译成字节码)，而不是直接调用str，可以避免多次执行重复编译过程，提高程序性能。正则表达式模式匹配也类似，也最好先将正则表达式模式编译成regex对象(通过re.complie()函数)，然后再执行比较和匹配。
* **采用生成器表达式替代列表解析：**列表解析会产生整个列表，对大量数据的迭代会产生负面效应。而生成器表达式则不会，其不会真正创建列表，而是返回一个生成器，在需要时产生一个值(延迟计算)，对内存更加友好。
* **关键代码使用外部功能包：**使用 C/C++ 或机器语言的外部功能包处理时间敏感任务，可以有效提高应用的运行效率。这些功能包往往依附于特定的平台，因此你要根据自己所用的平台选择合适的功能包 。比如下面四个功能包：Cython、Pylnlne、PyPy、Pyrex 。
* **在排序时使用键：**Python 含有许多古老的排序规则，这些规则在你创建定制的排序方法时会占用很多时间，而这些排序方法运行时也会拖延程序实际的运行速度。最佳的排序方法其实是尽可能多地使用键和内置的 sort() 方法。
* **优化算法时间：**算法的时间复杂度对程序的执行效率影响最大，在Python中可以通过选择合适的数据结构来优化时间复杂度，如list和set查找某一个元素的时间复杂度分别是O(n)和O(1)。不同的场景有不同的优化方式，总得来说，一般有分治，分支界限，贪心，动态规划等思想。
* **循环优化“**每种编程语言都会强调需要优化循环。当使用Python的时候，你可以依靠大量的技巧使得循环运行得更快。
* 技巧 1：减少循环内部不必要的计算
* 技巧 2：嵌套循环中，尽量减少内层循环的计算
* 技巧 3：尽量使用局部变量
* 技巧 4：使用 join() 连接字符串
* **交叉编译你的应用：**计算机其实并不理解用来创建现代应用程序的编程语言，计算机理解的是机器语言。所以我们可以用Python语言编写应用，再以C++这样的语言运行你的应用，这在运行的角度来说，是可行的。Nuitka是一款有趣的交叉编译器，能将你的Python代码转化成C++代码。这样，你就可以在native模式下执行自己的应用，而无需依赖于解释器程序。你会发现自己的应用运行效率有了较大的提高，但是这会因平台和任务的差异而有所不同。

**27、Python中的单例模式有几种实现方式?**

**答：**单例模式（Singleton Pattern）是一种常用的软件设计模式，该模式的主要目的是确保某一个类只有一个实例存在。在 Python 中，你可以想出几种种方法来实现单例模式呢？笔试题中，手写单例模式，也是经常碰到的，所以都要掌握下！

**1）使用模块实现：**Python 的模块就是天然的单例模式，因为模块在第一次导入时，会生成 .pyc 文件，当第二次导入时，就会直接加载 .pyc 文件，而不会再次执行模块代码。因此，我们只需把相关的函数和数据定义在一个模块中，就可以获得一个单例对象了。如果我们真的想要一个单例类，可以考虑这样做：

#test1.py

class Singleton(object):

def foo(self):

pass

singleton = Singleton()

#test2.py

from test1 import singleton

**2）用\_\_new\_\_特殊方法实现**

class Singleton:

def \_\_new\_\_(cls, \*args, \*\*kwargs):

if not hasattr(cls, '\_instance'):

cls.\_instance = super(Singleton, cls).\_\_new\_\_(cls)

return cls.\_instance

def \_\_init\_\_(self, name):

self.name = name

s1 = Singleton('IT圈')

s2= Singleton('程序IT圈')

print(s1 == s2) # True

**3）使用装饰器实现**

def singleton(cls):

\_instance = {}

def inner(\*args, \*\*kwargs):

if cls not in \_instance:

\_instance[cls] = cls(\*args, \*\*kwargs)

return \_instance[cls]

return inner

@singleton

class Singleton:

def \_\_init\_\_(self, name):

self.name = name

s1 = Singleton('IT圈')

s2= Singleton('程序IT圈')

print(s1 == s2) # True

**4）类装饰器实现**

class Singleton:

def \_\_init\_\_(self, cls):

self.\_cls = cls

self.\_instance = {}

def \_\_call\_\_(self, \*args):

if self.\_cls not in self.\_instance:

self.\_instance[self.\_cls] = self.\_cls(\*args)

return self.\_instance[self.\_cls]

@Singleton

class Singleton:

def \_\_init\_\_(self, name):

self.name = name

s1 = Singleton('IT圈')

s2= Singleton('程序IT圈')

print(s1 == s2) # True

**5）使用元类实现方式**

class Singleton1(type):

def \_\_init\_\_(self, \*args, \*\*kwargs):

self.\_\_instance = None

super(Singleton1, self).\_\_init\_\_(\*args, \*\*kwargs)

def \_\_call\_\_(self, \*args, \*\*kwargs):

if self.\_\_instance is None:

self.\_\_instance = super(Singleton1, self).\_\_call\_\_(\*args, \*\*kwargs)

return self.\_\_instance

class Singleton(metaclass=Singleton1):

def \_\_init\_\_(self, name):

self.name = name

s1 = Singleton('IT圈')

s2= Singleton('程序IT圈')

print(s1 == s2) # True

**28、Python实现自省的方法有哪些？**

**答：**自省是一种自我检查行为。在计算机编程中，自省是指这种能力：检查某些事物以确定它是什么、它知道什么以及它能做什么。自省向程序员提供了极大的灵活性和控制力。

说的更简单直白一点：自省就是面向对象的语言所写的程序在运行时，能够知道对象的类型。一句可以概况为：**运行时能够获知对象的类型。**

Python实现自省有很多方法，常用的有

* type()，判断对象类型
* dir()， 带参数时获得该对象的所有属性和方法；不带参数时，返回当前范围内的变量、方法和定义的类型列表
* help() , 用于查看函数或模块用途的详细说明
* isinstance()，判断对象是否是已知类型
* issubclass()，判断一个类是不是另一个类的子类
* hasattr()，判断对象是否包含对应属性
* getattr()，获取对象属性
* setattr()， 设置对象属性
* id(): 用于获取对象的内存地址
* callable()：判断对象是否可以被调用。

**29、简述一下爬虫的步骤？**

* 确定需求；
* 确定资源；
* 通过url获取网站的返回数据；
* 定位数据；
* 存储数据。

**30、遇到的反爬虫策略以及解决方法?**

* 通过headers反爬虫：自定义headers，添加网页中的headers数据。
* 基于用户行为的反爬虫(封IP)：可以使用多个代理IP爬取或者将爬取的频率降低。
* 动态网页反爬虫(JS或者Ajax请求数据)：动态网页可以使用 selenium + phantomjs 抓取。
* 对部分数据加密处理(数据乱码):找到加密方法进行逆向推理。

**常见的反爬虫和应对方法有：**

* 基于用户行为，同一个ip段时间多次访问同一页面 利用代理ip，构建ip池
* 请求头里的user-agent 构建user-agent池（操作系统、浏览器不同，模拟不同用户）
* 动态加载（抓到的数据和浏览器显示的不一样），js渲染 模拟ajax请求，返回json形式的数据
* selenium / webdriver 模拟浏览器加载
* 对抓到的数据进行分析
* 加密参数字段 会话跟踪【cookie】 防盗链设置【Referer】

以后遇到更多的问题会加更，祝大家面试顺利！

共勉。