1. 选择题
2. 下列哪个语句在Python中是非法的？（B）

A．x=y=z=1 B. x=(y=z+1)

C. x,y=y,x D. x+=y

2. 下面哪个不是Python合法的标识符（B）

A . int32 B. 40XL C. self D. name

1. python不支持的数据类型（A）

A . char B. int C. float D. list

1. python序列类型不包括（C）

A. 列表 B. 字符串 C. 字典 D. 元组

1. 下列不是元组类型的是（C）

A. a=(1,2) B. a=tuple([1,2]) C . a=(1) D. a=”1”, ”2”, ”3”, ”4”

6. 下列等式中输出为False的是（C）

A. 1+1 is 2 B. 999+1 is 1000 C. 2.0\*3.0 ==3.0\*2.0 D. type(bool)==type(int)

1. 有两个元祖t1=(1,2),t2=(3,[4]),下列操作有误的是（**B**）

A. t3=t1+t2 B. t2[0]=5 C. t2[1].append(5) D. t1[:5]

8.下列哪种捕获异常的方式是错误的（D）

A. try….exxept…else B. try….finally…

C. try….exxept… D. try….raise

二、填空题

9.请填写下列方法运行的代码：

def f(x,l=[]):

for i in range(x):l.append(i\*i)

print(l)

print(f(3,[3,2,1]))的结果是[3, 2, 1, 0, 1, 4]

None

print(f(3)) 的结果是[0, 1, 4]

None

10.如何将字符串str=”hello world”反向输出，请用最简单最简洁的语句输出：str[::-1]

11.有一个列表a=[i for i in range(10)]请写出下列代码的输出结果

sums=sum(map(lambda x: x+3,a[1::2]))

print(s ums)---40

12.写出下列函数调用结果：

my\_list=[lambda:i for i in range(5)]

for l in my\_list:print(l())

结果是(换行用\t)4\t4\t\4\t4\t4

13.有一个列表b=[2,4,5,6]执行了下列操作

for i in b:

if not i%2:

b.remove(i)

print(b)结果是[4,5]

三、简答题

14.详细说说tuple、list、dict的用法，它们的特点

列表，元组，字典，都是可迭代对象

列表和元组都是序列，列表是可变的序列, 元组是不可变的序列；列表可以通过索引改变列表的元素， 元组不可以；列表可以通过切片赋值插入和删除数据，也可以改变数据， 元组不可以

字典是可变的容器，字典可以存储任意类型的数据；字典中的数据都是用键进行索引的；字典的存储是无序的；字典中的数据是以 键-值（key-value）对 的形式进行存储的；字典的键不能重复，且只能用不可变类型作为字典的键。

15.用自己的话说明迭代器和生成器，它们之间的关系

迭代器：指的是一个重复的过程，每一次重复称为一次迭代，并且每一次重复的结果是下一次重复的初始值

生成器：只要在函数体内出现yield关键字，那么再执行函数就不会执行函数代码，会得到一个结果，该结果就是生成器

16.什么是lambda函数？它有什么好处

lambda 表达式（又称匿名函数）

作用：创建一个匿名函数对象,同def类似，但不提供函数名

语法说明：

1. lambda 只是一个表达式 ,它用来创建一个函数对象

2. 当lambda表达式执行时，返回的是冒号(:)后 的表达式的值

3. lambda表达式创建的函数只能包含一条语句

4. lambda比函数简单，且可以随时创建和销毁，有利于减少程序的偶合度

17.Python里赋值（=）,浅拷贝，深拷贝的区别

答：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}

18.观察输出，用自己的话解释\*args,\*\*kwargs这两个参数是什么意思，我问为什么使用它？

def func(\*args,\*\*kwargs):print(args,kwargs)

l=[1,2,3]

t=(4,5,6)

d={'a':7,'b':8,'c':9}

func(1,2,3)#(1, 2, 3) {}

func(a=1,b=2,c=3)#() {'a': 1, 'c': 3, 'b': 2}

func=(1,2,3,a=1,b=2,c=3) #(1,2,3) {'a':1,'c':3,'b':2}

func(\*l,\*\*d)#(1, 2, 3) {'a': 7, 'c': 9, 'b': 8}

func(l,2,\*t)#([1, 2, 3], 2, 4, 5, 6) {}

func(q='winning',\*\*d)#() {'a': 7, 'c': 9, 'b': 8, 'q': 'winning'}

答案： 如果我们不确定要往函数中传入多少个参数，或者我们想往函数中以列表和元组 的形式传参数时，那就使要用\*args； 如果我们不知道要往函数中传入多少个关键词参数，或者想传入字典的值作为关 键词参数时，那就要使用\*\*kwargs。 args和kwargs这两个标识符是约定俗成的用法，你当然还可以用\*bob和\*\*billy， 但是这样就并不太妥。

四、编程题

19.将一个正整数分解质因数。例如输入90，打印90=2\*3\*3\*5

n = num = int(input('请输入一个数字：')) #用num保留初始值

f = [] #存放质因数的列表

for j in range(int(num/2)+1): #判断次数仅需该数字的一半多1次

for i in range(2, n):

t = n % i #i不能是n本身

if t == 0: #若能整除

f.append(i) #则表示i是质因数

n = n//i #除以质因数后的n重新进入判断，注意应用两个除号，使n保持整数

break #找到1个质因数后马上break，防止非质数却可以整除的数字进入质因数列表

if len(f) == 0: #若一个质因数也没有

print('该数字没有任何质因数。')

else: #若至少有一个质因数

f.append(n) #此时n已被某个质因数整除过，最后一个n也是其中一个质因数

f.sort() #排下序

print('%d=%d' % (num, f[0]), end='')

for i in range(1,len(f)):

print('\*%d' % f[i], end='')

20.编写一个队列类Queue(先进先出)，实现队列enqueue,出列dequeue,

队列有多少个元素size,添加一个队列extend,清空队列clear

是否为空isEmpty,打印全部元素等方法showinfo(该队列可以初始时赋值)

1. 解释什么是栈溢出，在什么情况下可能出现
2. 简述CPython的内存管理机制

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， Python 开发学院整理-面试题 18-AID 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。

1. 列举你知道的Python的魔法方法及用途

|  |  |
| --- | --- |
| \_\_new\_\_(cls[, ...]) | 1. \_\_new\_\_ 是在一个对象实例化的时候所调用的第一个方法 2. 它的第一个参数是这个类，其他的参数是用来直接传递给 \_\_init\_\_ 方法 3. \_\_new\_\_ 决定是否要使用该 \_\_init\_\_ 方法，因为 \_\_new\_\_ 可以调用其他类的构造方法或者直接返回别的实例对象来作为本类的实例，如果 \_\_new\_\_ 没有返回实例对象，则 \_\_init\_\_ 不会被调用 4. \_\_new\_\_ 主要是用于继承一个不可变的类型比如一个 tuple 或者 string |
| \_\_init\_\_(self[, ...]) | 构造器，当一个实例被创建的时候调用的初始化方法 |
| \_\_del\_\_(self) | 析构器，当一个实例被销毁的时候调用的方法 |
| \_\_call\_\_(self[, args...]) | 允许一个类的实例像函数一样被调用：x(a, b) 调用 x.\_\_call\_\_(a, b) |
| \_\_len\_\_(self) | 定义当被 len() 调用时的行为 |
| \_\_repr\_\_(self) | 定义当被 repr() 调用时的行为 |
| \_\_str\_\_(self) | 定义当被 str() 调用时的行为 |
|  |  |
| \_\_getattr\_\_(self, name) | 定义当用户试图获取一个不存在的属性时的行为 |
| \_\_getattribute\_\_(self, name) | 定义当该类的属性被访问时的行为 |
| \_\_setattr\_\_(self, name, value) | 定义当一个属性被设置时的行为 |
| \_\_delattr\_\_(self, name) | 定义当一个属性被删除时的行为 |
| \_\_len\_\_(self) | 定义当被 len() 调用时的行为（返回容器中元素的个数） |
| \_\_getitem\_\_(self, key) | 定义获取容器中指定元素的行为，相当于 self[key] |
| \_\_setitem\_\_(self, key, value) | 定义设置容器中指定元素的行为，相当于 self[key] = value |
| \_\_delitem\_\_(self, key) | 定义删除容器中指定元素的行为，相当于 del self[key] |
| \_\_iter\_\_(self) | 定义当迭代容器中的元素的行为 |
| \_\_reversed\_\_(self) | 定义当被 reversed() 调用时的行为 |
| \_\_contains\_\_(self, item) | 定义当使用成员测试运算符（in 或 not in）时的行为 |

重载运算……

1. 已知以下list：

list1=[{'mm':2},{'mm':1},{'mm':4},{'mm':3},{'mm':3}]

1. 把list1中的元素按mm的值排序
2. 获取list1中第一个mm的值等于x的元素
3. List1[::4]输出的是什么

[{'mm': 2}, {'mm': 3}]

1. 简述你对GIL的理解

线程全局锁(Global Interpreter Lock),即 Python 为了保证线程安全而采取 的独立线程运行的限制,说白了就是一个核只能在同一时间运行一个线程.

1. 简述以下内置函数的用法
2. 简述多线程，多进程，协程之间的区别

进程：是资源分配的最小单位，创建和销毁开销较大；

线程：是CPU调度的最小单位，开销小，切换速度快；

协程：轻量级的线程，又称之为 纤程，是一种用户态的轻量级线程，实际上只有一个单线程完成。

操作系统将CPU时间片分配给多个线程，每个线程在指定放到时间片内完成。操作系统不断从一个线程切换到另一个线程执行，宏观上看就好像是多个线程一起执行。

Python中由于全局锁 (GIL)的存在导致，同一时间只有一个获得GIL的线程在跑，其他线程则处于等待状态，这导致了多线程只是在做分时切换，并不能利用多核。

多线程与多进程的区别：（1）多进程中同一个变量各自有一份拷贝在每个进程中，互不影响；（2）多线程中，所有变量都由所有线程共享，任何一个变量都可被任何一个线程修改。线程之间共享数据的最大危险在于多个线程同时更改一个变量，把内容改乱。

协程优点：创建协程资源消耗非常少，协程的优点是可以用作IO高并发处理

无需上下文切换的开销，没有临界资源的争夺，保证对数据的原子操作，缺点是无法利用多核。

进程拥有自己独立的堆和栈，既不共享堆，也不共享栈，进程由操作系统调度

线程拥有自己独立的栈，和共享的堆，不共享栈，线程也是由操作系统调度

协程和线程一样共享堆，不共享栈，协程由程序员在协程的代码里显示调度

协程避免了无意义的调度，由此可以提高性能，但也因此，程序员必须自己承担调度的责任，同时，协程也失去了标准线程使用多cpu的能力

说协程是进程和线程的升级版,进程和线程都面临着内核态和用户态的切 换问题而耗费许多切换时间,而协程就是用户自己控制切换的时机,不再需要陷 入系统的内核态.

1. 简述COOKIE和SESSION的区别与联系

cookie 和session 的区别：  
1、cookie数据存放在客户的浏览器上，session数据放在服务器上。  
2、cookie不是很安全，别人可以分析存放在本地的COOKIE并进行[COOKIE欺骗](https://www.baidu.com/s?wd=COOKIE%E6%AC%BA%E9%AA%97&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y4uHm4n1ndnW99nHTkmWb10ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWD1rHcLnj0dnWbdPjf1nHb3r0" \t "_blank),考虑到安全应当使用session。  
3、session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能,考虑到减轻服务器性能方面，应当使用COOKIE。  
4、单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。  
cookie 和session 的联系：  
session是通过cookie来工作的  
session和cookie之间是通过$\_COOKIE['PHPSESSID']来联系的，通过$\_COOKIE['PHPSESSID']可以知道session的id，从而获取到其他的信息

1. 简述什么是浏览器的同源策略

所谓同源是指，域名，协议，端口相同。不同源的客户端脚本在没有明确授权的情况下，不能读写对方的资源

10.git commit –amend有何用处

比方说，你的代码已经提交到git库，leader审核的时候发现有个Java文件代码有点问题，于是让你修改，通常有2种方法：方法1：leader 将你提交的所有代码 abandon掉，然后你回去 通过git reset …将代码回退到你代码提交之前的版本，然后你修改出问题的Java文件，然后 git add xx.java xxx.java -s -m “Porject : 1.修改bug…”   
最后通过 git push origin HEAD:refs/for/branches方法2：leader不abandon代码，你回去之后，修改出问题的Java文件，修改好之后，git add 该出问题.java   
然后 git commit –amend –no-edit,   
最后 git push origin HEAD:refs/for/branches

11.git如何查看某次提交修改的内容

我们首先可以git log显示历史的提交列表：

之后我们用git show <commit-hashId> 便可以显示某次提交的修改内容

同样 git show <commit-hashId> filename 可以显示某次提交的某个内容的修改信息

12.git如何比较两个commit的区别

git diff commit-id-1 commit-id-2 > d:/diff.txt

结果文件diff.txt中：

"-"号开头的表示 commit-id-2 相对 commit-id-1 减少了的内容。  
"+"号开头的表示 commit-id-2 相对 commit-id-1 增加了的内容。

13.git如何把分支A上某个commit应用到分支B上

1. 执行git log -3 --graph A，查看A分支下的commit:    注：commit 后面的hash值代表某个commit，这里把”82f1fb7138c5860cc775b4b5ea71c5d19c4e6497“这个commit提交到B。

2. 执行git checkout B，切换到B分支;

3. 执行 git cherry-pick 82f1fb7138c5860cc775b4b5ea71c5d19c4e6497，该commit便被提交到了B分支;

4. git push     //注：将该commit推到远程服务器

14.如何查看linux系统的启动时间，磁盘使用量，内存使用量

top

**HTTP协议**

1. 请列举常见的HTTP头及其作用

Accept:指浏览器或其他客户可以接爱的MIME文件格式。可以根据它判断并返回适当的文件格式。

Accept-Charset：指出浏览器可以接受的字符编码。英文浏览器的默认值是ISO-8859-1.

Accept-Language：指出浏览器可以接受的语言种类，如en或en-us，指英语。

Accept-Encoding：指出浏览器可以接受的编码方式。编码方式不同于文件格式，它是为了压缩文件并加速文件传递速度。浏览器在接收到Web响应之后先解码，然后再检查文件格式。

Cache-Control：设置关于请求被代理服务器存储的相关选项。一般用不到。

Connection：用来告诉服务器是否可以维持固定的HTTP连接。HTTP/1.1使用Keep-Alive为默认值，这样，当浏览器需要多个文件时(比如一个HTML文件和相关的图形文件)，不需要每次都建立连接。

Content-Type：用来表名request的内容类型。可以用HttpServletRequest的getContentType()方法取得。

Cookie：浏览器用这个属性向服务器发送Cookie。Cookie是在浏览器中寄存的小型数据体，它可以记载和服务器相关的用户信息，也可以用来实现会话功能。

1. 请列举常见的HTTP状态码响应码及其意义

状态代码有三位数字组成，第一个数字定义了响应的类别，且有五种可能取值：

1xx：指示信息–表示请求已接收，继续处理

2xx：成功–表示请求已被成功接收、理解、接受

3xx：重定向–要完成请求必须进行更进一步的操作

4xx：客户端错误–请求有语法错误或请求无法实现

5xx：服务器端错误–服务器未能实现合法的请求

常见状态代码、状态描述、说明：

200 OK //客户端请求成功

400 Bad Request //客户端请求有语法错误，不能被服务器所理解

401 Unauthorized //请求未经授权，这个状态代码必须和WWW-Authenticate报头域一起使用

403 Forbidden //服务器收到请求，但是拒绝提供服务

404 Not Found //请求资源不存在，eg：输入了错误的URL

500 Internal Server Error //服务器发生不可预期的错误

503 Server Unavailable //服务器当前不能处理客户端的请求，一段时间后可能恢复正常

1. 请简述对REST API设计规范的理解

REST API是使用统一资源标识符（url）来搜寻资源 有七大设计原则：

1. url结尾不应包含（/）

2. 正斜杠分隔符（/）必须用来指示层级关系

3. 应使用连字符（-）来提高url的可读性

4. 不得在url中使用下划线（\_）

5. url路径中首选小写字母

6. 文件扩展名中不应包含在url中

7. 端点名称是单数（但是实际上为了保持url格式的一致性建议使用复数形式）

正在构建的服务中的每个资源将至少有一个url标识它，这个url最好是有意义的，且能充分藐视资源。url应遵循可预测的层次结构，用来提高其可理解性，可用性：可预测的意义在于他们是一致的，他的层次结构在数据关系上时有意义的。

REST API是使用者编写的，url的名称和结构应该能够向使用者传达更清晰的含义。通过遵循上述规则，您将创建一个更清晰的REST API与更友好的客户端。

1. 请简述HTTP缓存机制

简单来说就是把一个已经请求过的 Web 资源（如 html 页面，图片， js，数据等）拷贝一份副本储存在浏览器中。缓存会根据进来的请求保存输出内 容的副本。当下一个请求来到的时候，如果是相同的 URL，缓存会根据缓存机制 决定是直接使用副本响应访问请求，还是向源服务器再次发送请求。比较常见的 就是浏览器会缓存访问过网站的网页，当再次访问这个 URL 地址的时候，如果网 页没有更新，就不会再次下载网页，而是直接使用本地缓存的网页。只有当网站 明确标识资源已经更新，浏览器才会再次下载网页

好处：

减少请求次数，减小服务器压力,本地数据读取速度更快，让页面不会空白几百毫秒,在无网络的情况下提供数据。

**其他**

1. 请列举经常访问的技术网站或博客

知乎 http://www.zhihu.com/  
博客园 http://www.cnblogs.com/  
开源中国社区 http://www.oschina.net/  
红黑联盟 http://www.2cto.com/

1. 请列举最近关注的一些技术
2. 请列举你认为不懂得一些技术书籍和你最近在看的书籍（不限于技术）
3. 请列举你阅读过源码的一些项目
4. 请给出你对这份笔试题的看法