首先Hash用于校验程序在传输过程中是否被第三方（木马）修改，如果程序（字符）在传输过程中被修改hash值即发生变化，如果没有被修改，则 hash 值和原始的 hash 值吻合，只要验证 hash 值是否匹配即可验证程序是否带木马（病毒）。

简单的例子有：

name1='正常程序代码'

name2='正常程序代码带病毒'

print(hash(name1)) # 2403189487915500087

print(hash(name2)) # -8751655075885266653

Python产生hash值有两种方法。

1.Python可以使用内置函数hash(对象)生成hash值。hash () 函数可以应用于数字、字符串和对象，不能直接应用于 list、set、dictionary。

在hash()对对象使用时，所得的结果不仅和对象的内容有关，还和对象的 id()，也就是内存地址有关。

**🡪特别注意的，在Python2和Python3中，对同样的内容字符串生成的hash值是不一样的。**

**🡪Python3中，内置函数hash还采用了SIP，每一次会根据一个随机数进行生成。所以默认情况下每次启动Python对同样的内容字符串生成的hash值是不一样的。如果我们设置环境变量PYTHONHASHSEED=0， 关闭基于随机数。那么Python3每次对同样内容的字符串生成的hash值相同，但是和Python2也不会一样。**

2.

第二种方法是使用hashlib模块。 使用这个模块一般分为3步

创建一个哈希对象, 使用哈希算法命名的构造函数或通用构造函数hashlib.new(name[, data])

使用哈希对象调用update()方法填充这个对象,多次调用同一个对象的update方法将填充这个对象的字符串内容。

调用digest() 或 hexdigest()方法来获取摘要(加密结果)

注:

　　1.update () 方法需要接收的参数是一个字节对象 (字节字符串, 如:b"Hello,World!")，python3中对于str对象要先encode.

　　2.常用的一些算法主要有: SHA1, SHA224, SHA256, SHA384, SHA512, MD5等算法

　　3.SHA1算法比较早, 是不能抵抗暴力破解的

**hashlib模块提供一下常量属性**

　　hashlib.algorithms\_guaranteed

　　　　获取保证在所有平台上此模块支持的哈希算法名称的集合

　　hashlib.algorithms\_available

　　　　获取可以运行在Python解释器中的哈希算法名称的集合

**哈希对象常用的方法**

**h = hashlib.md5 () 或 h = hashlib.new("md5”) # md5可以替换为其他的哈希类型**

h.update(arg)

　　　　将字节对象arg填充到哈希对象中,arg通常为要加密的字符串

　　h.digest()

　　　　返回加密结果,它是一个字节对象,长度为 h.digest\_size, 包含的字节范围 0 ~ 255

　　h.hexdigest()

　　　　返回加密结果,它是一个字符串对象,长度为 h.digest\_size \* 2, 只包含16进制数字

**🡪特别注意的，在Python2和Python3中，通过hashlib采用同样的计算算法对同样的内容字符串生成的hash值是一样的。**