### 1.python with 如何使用？好处是什么？为什么可以起到关闭文件的作用？

**使用场景**：对于python在写入文件或者是读取文件的结束，需要手动关闭文件，极端的情况会出现too may open files 的错误，因为系统允许打开的文件数量是有限的。

**代码演变:**

**初级：**下面是不使用with的代码，会出现的问题是当使用readline出现异常的时候，下面的代码就不会被执行，因而文件就不会被关闭。

f = open("test.txt",'r')

f.readline()

f.close()

**进阶：**下面的代码使用了try进行异常捕捉，如果出现异常则会运行except代码，最后执行finally代码，该文件一定会被关闭，

f = open("test.txt",'r')

try:

f.readline()

except iOError:

print('error')

finally:

f.close()

**高级**：下面的代码系统会自动调用close方法，所以和进阶的作用是一样的，而with实现的原理就是实现了上下文管理器。

with open("test.txt",'r') as f:

f.readline()

**上下文管理器：**

任何实现了\_\_enter\_\_()和\_\_exit\_\_()方法的对象都称为上下文管理器。因此可以使用with关键字，显然file类实现了上下文管理器。

**手动写出file关闭文件功能：**即使在readline()函数出现异常，\_\_exit\_\_()也会被调用。

**class** file(object):  
 **def** \_\_init\_\_(self, filename, mode):  
 self.filename= filename  
 self.mode = mode  
   
 **def** \_\_enter\_\_(self):  
 print(**'entering'**)  
 self.f = open(self.filename, self.mode)  
 **return** self.f  
   
 **def** \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):  
 print(**'will exit'**)  
 self.f.close()

**if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 **with** file(**'test.txt'**,**'r'**) **as** f:  
 1/0  
 print(f.readline())

### 2. python \_\_name\_\_的作用？

**概念：**所有的模块都含有\_\_name\_\_属性，而\_\_name\_\_属性的值取决与如何使用这个模块，如一个模块为learning\_python.py

如果导入该模块：import learning\_python，则\_\_name\_\_属性的值为leanring\_python

如果运行该模块：python2 leanring\_python.py,则\_\_name\_\_属性的值为main

**作用：**所以可以添加如下代码,如果该模块被导入，则\_\_name\_\_为模块名字，则该if不会被执行，如果直接运行该模块，则\_\_name\_\_为\_\_main\_\_则该if会被运行。

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

do\_something()

### 3. python is和==的区别？

**前言：**python一切都是对象，而对象就会有三要素，id，value和type

**概念：**

is比较的是对象的id是否相同，也就是是否是同一个对象，是否指向同一个内存地址。

==比较的是对象的value值，

下面的结论只针对python2.7

**结论：**

当变量为数字，字符串，元组，列表，字典的时候，==和is不能互换使用，比较数值的时候尽量使用==，当比较两个对象id是否相同可以使用is

注：如果是数值，范围在[-5,256]则is和==的结果是一样的。