1.

关于else，

else与for/while/try都可以搭配使用。

对while和for，只要不是因为break退出的，循环结束时都会执行else。

对于try，没有异常抛出时，才会运行else（else里面的异常不会由前面的except处理）。

2.

上下文管理器对象存在的目的是管理with语句。如同迭代器的存在是为了管理for语句。【虽然对上面两句话因果关系表示质疑，但是对应关系应该是没关系的】

3.

上下文管理器包含\_\_enter\_\_,\_\_exit\_\_两个方法。with语句开始时，会在上下文管理器对象上调用\_\_enter\_\_，with结束后，会在上下文管理器对象上调用\_\_exit\_\_（扮演finally）。

4.

with后面表达式的结果得到的是上下文管理器，\_\_enter\_\_方法返回self。

P371的例子【】值得注意，with没有定义新的作用域，所以as后面的变量在之后的代码里依然可以用，不过它在with外就被关闭了。

5.

【小知识】重复导入模块不会消耗很多资源。

6.

P372 15-3,15-4的例子很有意思，好像\_\_enter\_\_进入了一个状态，\_\_exit\_\_解除了这个状态。with不仅可以用作资源管理，也可用作去掉常规的设置和清理代码。

7.

contextlib里面有很多关于上下文管理器的工具，不过貌似暂时用不上（看不懂）。

8.

@contextmanager可以减少创建上下文管理器的代码量。只需要实现一个yield语句的生成器，生成\_\_enter\_\_方法返回的值。

关于使用方法，在P375的例子上，P375的例子的完整版在P377。

9.

P377，划线部分粘上

10.

P378，inplace的代码看起来很有吸引力，可以copy下来，链接在P378，