with与"上下文管理器"

笔记本：	Python提高-2		
创建时间：	2018/5/8 20:57	**更新时间：**	2018/5/15 8:21
作者：	ly		

with与"上下文管理器"

问题：
　　系统资源如文件、数据库连接、socket 而言，应用程序打开这些资源并执行完业务逻辑之后，必须做的一件事就是要关闭（断开）该资源。

- Python 程序打开一个文件，往文件中写内容，写完之后，就要关闭该文件。例如：你打开一个记事本txt，写入内容，而这时系统关机了，该内容不会被保存，因为之前打开文件只是在内存条中呈现，即：缓存，而没有存入到硬盘中。并且系统允许你打开的最大文件数量是有限的。
- 对于数据库，如果连接数过多而没有及时关闭的话，会报错，因为数据库连接是一种非常昂贵的资源，不可能无限制的被创建

以打开文件为例子

普通：

```
f = open("output", "w")   # 第一步打开文件（没有则新建）
f.write("hello python")    # 第二步写入内容
f.close()    # 关闭文件
```

存在问题：执行代码过程中会出现意外，而没有执行完毕，即文件打开未关闭

try版：

```
def m2():
    f = open("putput.txt", "w")
    try:
        f.write("hello python")
    except IOError:
        print("oops error")
    finally:                 # 不管上叙写入的代码是否执行，照常会执行，保证文件必须关闭
        f.close()
```

最终解决版：

```
def m3():
    with open ("output.txt", "w") as f:
        f.write("hello python")
```

一种更加简洁、优雅的方式就是用 with 关键字。

open 方法的返回值赋值给变量 f，当离开 with 代码块的时候，系统会自动调用 f.close() 方法

上下文管理

上下文(（context）理解

什么都要有头有尾，才能让人理解清楚

看，一篇文章，给你摘录一段，没前没后，你读不懂，因为有语境，就是语言环境存在，一段话说了什么，要通过上下文(文章的上下文)来推断。

上下文管理器

任何实现了__enter__() 和 __exit__() 方法的对象都可以称之为上下文管理器
例如：with关键字，在Python底层已经实现了上叙两个方法

结论

Python 提供了 with 语法用于简化资源操作的后续清除操作，是 try/finally 的替代方法，实现原理建立在上下文管理器之上。此外，Python 还提供了一个 contextmanager 装饰器，更进一步简化上下管理器的实现方式。