引言

with 语句适用于对资源进行访问的场合，确保不管使用过程中是否发生异常都会执行必要的“清理”操作，释放资源，比如文件使用后自动关闭、线程中锁的自动获取和释放等。

术语

要使用 with 语句，首先要明白上下文管理器这一概念。有了上下文管理器，with 语句才能工作。

下面是一组与上下文管理器和with 语句有关的概念。

**上下文管理协议（Context Management Protocol）**：包含方法 \_\_enter\_\_() 和 \_\_exit\_\_()，支持

该协议的对象要实现这两个方法。

**上下文管理器（Context Manager）**：支持上下文管理协议的对象，这种对象实现了

\_\_enter\_\_() 和 \_\_exit\_\_() 方法。上下文管理器定义执行 with 语句时要建立的运行时上下文，

负责执行 with 语句块上下文中的进入与退出操作。通常使用 with 语句调用上下文管理器，

也可以通过直接调用其方法来使用。

**运行时上下文（runtime context）**：由上下文管理器创建，通过上下文管理器的 \_\_enter\_\_() 和

\_\_exit\_\_() 方法实现，\_\_enter\_\_() 方法在语句体执行之前进入运行时上下文，\_\_exit\_\_() 在

语句体执行完后从运行时上下文退出。with 语句支持运行时上下文这一概念。

**上下文表达式（Context Expression）**：with 语句中跟在关键字 with 之后的表达式，该表达式

要返回一个上下文管理器对象。

**语句体（with-body）**：with 语句包裹起来的代码块，在执行语句体之前会调用上下文管

理器的 \_\_enter\_\_() 方法，执行完语句体之后会执行 \_\_exit\_\_() 方法。

基本语法和工作原理

with 语句的语法格式如下：

清单 1. with 语句的语法格式

|  |  |
| --- | --- |
| 1  2 | with context\_expression [as target(s)]:      with-body |

这里 context\_expression 要返回一个上下文管理器对象，该对象并不赋值给 as 子句中的 target(s) ，如果指定了 as 子句的话，会将上下文管理器的 \_\_enter\_\_() 方法的返回值赋值给 target(s)。target(s) 可以是单个变量，或者由“()”括起来的元组（不能是仅仅由“,”分隔的变量列表，必须加“()”）。

Python 对一些内建对象进行改进，加入了对上下文管理器的支持，可以用于 with 语句中，比如可以自动关闭文件、线程锁的自动获取和释放等。假设要对一个文件进行操作，使用 with 语句可以有如下代码：

清单 2. 使用 with 语句操作文件对象

|  |  |
| --- | --- |
| 1  2  3  4 | with open(r'somefileName') as somefile:      for line in somefile:          print line          # ...more code |

这里使用了 with 语句，不管在处理文件过程中是否发生异常，都能保证 with 语句执行完毕后已经关闭了打开的文件句柄。如果使用传统的 try/finally 范式，则要使用类似如下代码：

清单 3. try/finally 方式操作文件对象

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | somefile = open(r'somefileName')  try:      for line in somefile:          print line          # ...more code  finally:      somefile.close() |

比较起来，使用 with 语句可以减少编码量。已经加入对上下文管理协议支持的还有模块 threading、decimal 等。