# 面向对象编程

## 异常(高级)

**回顾异常相关的语句:**

* try-except 用来捕获异常通知
* try-finally 用来做一定要做的事情
* raise 用来发生异常通知
* assert 用来根据条件来发出AssertionError类型的异常通知

### with 语句

**语法:**

with 表达式1 [as 变量1], 表达式2 [as 变量2]:  
 语句块

**作用:**

使用于对资源进行访问的场合,确保使用过程中不管是否发生异常,都会执行必须的’清理’操作,并释放资源。如: 文件使用后自动关闭, 线程中锁的自动获取和释放等

**示例见:**

01\_with.py

**说明:**

能够用于with语句进行管理的对象必须是环境管理器

### 环境管理器:

* 1. 类内有\_\_enter\_\_ 和 \_\_exit\_\_实例方法的类被称为环境管理器
  2. 能够用with语句管理的对象必须是环境管理器
  3. \_\_enter\_\_方法将在进入with语句时被调用,并返回由 as变量管理的对象
  4. \_\_exit\_\_将在离开with语句时被调用,且可以用参数来判断在离开with语句时是否有异常发生并做出相应的处理

**示例:**

class A:  
 def \_\_enter\_\_(self):  
 print("已进入with语句")  
 return self # 返回的对象将由 as绑定  
 def \_\_exit\_\_(self, exc\_type, exc\_val, exc\_tb):  
 print("已离开with语句")

**详见:**

03\_context.py

### 对象的属性管理函数

* getattr(obj, name[, default]) 从一个对象得到对象的属性；getattr(x, ‘y’) 等同于x.y; 当属性不存在时,如果给出default参数,则返回default,如果没有给出default则产生一个AttributeError错误
* hasattr(obj, name) 用给定的name返回对象obj是否有此属性,此种做法可以避免在getattr(obj, name)时引发错误
* setattr(obj, name, value) 给对象obj的名为name的属性设置相应的值value, set(x, ‘y’, v) 等同于 x.y = v
* delattr(obj, name) 删除对象obj中的name属性, delattr(x, ‘y’) 等同于 del x.y