Python程序设计教案

<table>
<tr><td>本次授课内容</td><td colspan="2">11.1 异常的概念及常见表现形式
11.2 常用异常处理机构
11.3 断言语句与上下文管理语句</td></tr>
<tr><td>本次课的
教学目的</td><td colspan="2">了解异常基本概念及其常见表现形式
理解出现异常的各种原因和处理异常的必要性
掌握常用的异常处理结构</td></tr>
<tr><td>本次课教学
重点与难点</td><td colspan="2">异常处理机构的作用
断言语句和上下文管理语句的使用</td></tr>
<tr><td>教学方法
教学手段</td><td colspan="2">PPT、边讲边练</td></tr>
<tr><td rowspan="3">课堂教学
时间分配</td><td>教学内容</td><td>时间分配（分）</td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td>课堂教学设计</td><td colspan="2">简单介绍异常的概念、常见异常处理机构、断言语句以及上下文管理语句。</td></tr>
<tr><td>实　验</td><td colspan="2">教材中所有例题。</td></tr>
<tr><td>思考题及作业题</td><td colspan="2">本章所有课后习题。</td></tr>
<tr><td>备　注</td><td colspan="2"></td></tr>
<tr><td>教学后记</td><td colspan="2"></td></tr>
<tr><td></td><td colspan="2">第　一　节　课</td></tr>
</table>

课堂重点内容详解	常见的异常的表现形式：
	
	`>>> 2 / 0 #除0错误，也就是除数不能为0`
	`Traceback (most recent call last):`
	`  File "<pyshell#9>", line 1, in <module>`
	`  2 / 0`
	`ZeroDivisionError: division by zero`
	`>>> 'a' + 2 #操作数类型不支持，略去异常的详细信息`
	`TypeError: Can't convert 'int' object to str implicitly`
	`>>> [1, 2, 3].find(3) #属性错误，列表没有find()方法`
	`AttributeError: 'list' object has no attribute 'find'`
	`>>> 3 * 'Hello world #语法错误，后面缺少单引号`
	`SyntaxError: EOL while scanning string literal`
	`>>> {3, 4, 5} * 3 #操作数类型不支持`
	`TypeError: unsupported operand type(s) for *: 'set' and 'int'`
	`>>> id(5) #有可能有代码改变了id的含义`
	`TypeError: 'int' object is not callable`
	`>>> print(testStr) #变量名不存在`
	`NameError: name 'testStr' is not defined`
	`>>> fp = open(r'D:\test.data', 'rb') #文件不存在`
	`FileNotFoundError: [Errno 2] No such file or directory: 'D:\\test.data'`
	`>>> fp.close() #不存在文件对象fp，应该是之前打开失败`
	`NameError: name 'fp' is not defined`
	`>>> len(3) #参数类型不匹配`
	`TypeError: object of type 'int' has no len()`
	`>>> list(3) #参数类型不匹配`
	`TypeError: 'int' object is not iterable`
	`>>> import socket`
	`>>> sock = socket.socket()`
	`>>> sock.connect(('1.1.1.1', 80)) #无法连接远程主机`
	`TimeoutError: [WinError 10060]` 由于连接方在一段时间后没有正确答复或连接的主机没有反应，连接尝试失败。
	
	11.2.1 try...except...
	
	Python 异常处理结构中最简单的形式是 try...except...结构，其中 try 子句中的代码块包含可能会引发异常的语句，而 except 子句则用来捕捉相应的异常。该结构语法如下：
	
	`try:`
	`  #可能会引发异常的代码，先执行一下试试`
	`except Exception[ as reason]:`
	`  #如果try中的代码抛出异常并被except捕捉，就执行这里的代码`
	
	如果 try 子句中的代码引发异常并被 except 子句捕捉，就执行 except 子句的代码块；如果 try 中的代码块没有出现异常就继续往下执行异常处理结构后面的代码；如果出现异常但没有被 except 捕获，继续往外层抛出，如果所有层都没有捕获并处理该异常，程序崩溃并将该异常信息呈现给最终用户。
	
	11.2.2 try...except...else...

在这个结构中，如果 try 中的代码抛出了异常并且被 except 语句捕捉则执行相应的异常处理代码，这种情况下就不会执行 else 中的代码；如果 try 中的代码没有引发异常，则执行 else 块的代码。语法如下：

```
try:
    #可能会引发异常的代码
except Exception [ as reason]:
    #用来处理异常的代码
else:
    #如果 try 子句中的代码没有引发异常，就继续执行这里的代码
```

11.2.3 try...except...finally...

在这种结构中，无论 try 中的代码是否发生异常，也不管抛出的异常有没有被 except 语句捕获，finally 子句中的代码总是会得到执行。因此，finally 中的代码常用来做一些清理工作，例如释放 try 子句中代码申请的资源。该结构语法为：

```
try:
    #可能会引发异常的代码
except Exception [ as reason]:
    #处理异常的代码
finally:
    #无论 try 子句中的代码是否引发异常，都会执行这里的代码
```

11.2.4 可以捕捉多种异常的异常处理结构

在实际开发中，同一段代码可能会抛出多种异常，并且需要针对不同的异常类型进行相应的处理。为了支持多种异常的捕捉和处理，Python 提供了带有多个 except 的异常处理结构，一旦 try 子句中的代码抛出了异常，就按顺序依次检查与哪一个 except 子句匹配，如果某个 except 捕捉到了异常，其他的 except 子句将不会再尝试捕捉异常。该结构类似于多分支选择结构，语法格式为：

```
try:
    #可能会引发异常的代码
except Exception1:
    #处理异常类型 1 的代码
except Exception2:
    #处理异常类型 2 的代码
except Exception3:
    #处理异常类型 3 的代码
...
```

11.3 断言语句与上下文管理语句

断言语句 assert 也是一种比较常用的代码调试技术，常用来在程序的某个位置确认指定的条件必须满足，如果满足条件就继续执行后续的代码，否则就抛出异常。一般来说，通过

了严格测试的代码在正式发布之前会删除 assert 语句，这样可以适当提高程序运行速度。

上下文管理（context manager）语句 with 可以自动管理资源，不论因为什么原因（哪怕是代码引发了异常）跳出 with 块，总能保证文件被正确关闭，并且可以在代码块执行完毕后自动还原进入该代码块时的现场，常用于文件操作、数据库连接、网络通信连接和多线程、多进程同步等场合。其具体用法可以参考本书文件操作的有关章节。

第 二 节 课

例 11-1 编写程序，接收用户输入，并且要求用户必须输入整数，不接收其他类型的输入。

```
>>> while True:
    x = input('Please input:')
    try:
        x = int(x)
        print('You have input {0}'.format(x))
        break
    except Exception as e:
        print('Error.')
```

例 11-2 使用 try...except...else...结构改写例 11-1 的代码。

```
>>> while True:
    x = input('Please input:')
    try:
        x = int(x)
    except Exception as e:
        print('Error.')
    else:
        print('You have input {0}'.format(x))
        break
```

例 11-3 编写程序，接收一个文本文件的名字，预期该文件中只包含一个整数，要求输出该数字加 5 之后的结果。如果文件不存在就提示不存在；如果文件存在但内容格式不正确，就提示文件内容格式不正确。

```
filename = input('请输入一个文件名：')

try:
    fp = open(filename)          #尝试打开文件
    try:                          #尝试读取数据并计算和输出
        print(int(fp.read())+5)
    except:                       #读取文件或计算失败时执行的代码
        print('文件内容格式不正确。')
    finally:                      #确保文件能够关闭
        fp.close()
```

```
except:                                    #打开文件失败时执行的代码
    print('文件不存在')
```

例 11-4 使用异常处理结构捕获多种可能的异常。

```
>>> try:
    x = float(input('请输入被除数: '))
    y = float(input('请输入除数: '))
    z = x / y
except ZeroDivisionError:
    print('除数不能为零')
except ValueError:
    print('被除数和除数应为数值类型')
except NameError:
    print('变量不存在')
else:
    print(x, '/', y, '=', z)
```