Python try except异常处理详解

Python 提供了try except语句捕获并处理异常，该异常处理语句的基本语法结构如下：

```
try:
    可能产生异常的代码块
except [(Error1, Error2, ...) [as e]]:
    处理异常的代码块1
except [(Error3, Error4, ...) [as e]]:
    处理异常的代码块2
```

该格式中，[] 括起来的部分可以使用，也可以省略；(Error1,Error2,...) 、(Error3,Error4,...) 表示各自的 except 块可以处理异常的具体类型；[as e] 表示将异常类型赋值给变量 e（方便在 except 块中调用异常类型）。

注意，except 后面也可以不指定具体的异常名称，这样的话，表示要捕获所有类型的异常。

另外，从 try except 的基本语法格式可以看出，try 块仅有一个，但 except 代码块可以有多个，这是为了针对不同的异常类型提供不同的异常处理方式。当程序发生不同的意外情况时，会对应不同的异常类型，Python 解释器就会根据该异常类型来决定使用哪个 except 块来处理该异常。

通过在 try 块后提供多个 except 块可以无须在异常处理块中使用 if 判断异常类型，但依然可以针对不同的异常类型提供相应的处理逻辑，从而提供更细致、更有条理的异常处理逻辑。

try except 语句的执行流程如下：

1. 首先执行 try 中的代码块，如果执行过程中出现异常，系统会自动生成一个异常对象，该异常对象会提交给 Python 解释器，此过程被称为引发异常。
2. 当 Python 解释器收到异常对象时，会寻找能处理该异常对象的 except 块，如果找到合适的 except 块，则把该异常对象交给该 except 块处理，这个过程被称为捕获异常。如果 Python 解释器找不到捕获异常的 except 块，则程序运行终止，Python 解释器也将退出。

事实上，不管程序代码块是否处于 try 块中，甚至包括 except 块中的代码，只要执行该代码块时出现了异常，系统总会自动生成一个 Error 对象。如果程序没有为这段代码定义任何的 except 块，则 Python 解释器无法找到处理该异常的 except 块，程序就会停止运行；反之，如果程序发生异常，并且该异常经 try 捕获并由 except 处理完成，则程序会继续执行。

举个例子：

```
try:
    a = int(input("输入被除数："))
    b = int(input("输入除数："))
    c = a / b
    print("您输入的两个数相除的结果是：", c )
```

```
except (ValueError, ArithmeticError):
    print("程序发生了数字格式异常、算术异常之一")
except :
    print("未知异常")
print("程序继续运行")
```

程序运行结果为：

```
输入被除数：a
程序发生了数字格式异常、算术异常之一
程序继续运行
```

上面程序中，第 6 行代码使用了（ValueError, ArithmeticError）来指定所捕获的异常类型，这就表明该 except 块可以同时捕获这 2 种类型的异常；第 8 行代码只有 except 关键字，并未指定具体要捕获的异常类型，这种省略异常类的 except 语句也是合法的，它表示可捕获所有类型的异常，一般会作为异常捕获的最后一个 except 块。

除此之外，由于 try 块中引发了异常，并被 except 块成功捕获，因此程序才可以继续执行，才有了“程序继续运行”的输出结果。

访问异常信息

如果程序需要在 except 块中访问异常对象的相关信息，可以通过为 except 块添加as a来实现。当 Python 解释器决定调用某个 except 块来处理该异常对象时，会将异常对象赋值给 except 块后的异常变量，程序即可通过该变量来获得异常对象的相关信息。

所有的异常对象都包含了如下几个常用属性和方法：

- args：该属性返回异常的错误编号和描述字符串。
- errno：该属性返回异常的错误编号。
- strerror：该属性返回异常的描述字符串。
- with_traceback()：通过该方法可处理异常的传播轨迹信息。

下面例子演示了程序如何访问异常信息：

```
def foo():
    try:
        fis = open("a.txt");
    except Exception as e:
        # 访问异常的错误编号和详细信息
        print(e.args)
        # 访问异常的错误编号
        print(e.errno)
        # 访问异常的详细信息
        print(e.strerror)
foo()
```

从上面程序可以看出，如果要访问异常对象，只要在单个异常类或异常类元组（多异常捕获）之后使用 as 再加上异常变量即可。

在 Python 的早期版本中，直接在单个异常类或异常类元组（多异常捕获）之后添加异常变量，中间用逗号隔开即可。

上面程序调用了 Exception 对象的 args 属性（该属性相当于同时返回 errno 属性和 strerror 属性）访问异常的错误编号和详细信息。运行上面程序，会看到如下运行结果：

```
(2, 'No such file or directory')
2
No such file or directory
```

从上面的运行结果可以看到，由于程序尝试打开的文件不存在，因此引发的异常错误编号为 2，异常详细信息为：No such file or directory。