#### 1.错误与异常

相信大家一定写了很多bug了，bug并不可怕，经验都是在bug中积累的，都有一段辛酸的历程。今天我们讲解错误与异常。

**重点**：掌握错误和异常，掌握处理异常的方式

#### 2.错误

当编写Python代码时，可能会遇到不同类型的错误。错误是指在代码编写过程中发生的问题，可以分为以下几类：

##### 2.1语法错误（Syntax Errors）

这种错误是由于代码不符合Python语法规则而引起的，通常在代码编译阶段就会被检测到。例如，缺少冒号、缩进错误、括号不匹配等。

if x = 5:  
 print("Hello, World!")

在这个示例中，=应该是赋值操作符，而不是在条件语句中使用的等于操作符。这会导致SyntaxError语法错误。

##### 2.2名称错误（Name Errors）

这种错误是由于使用了未定义的变量或函数而引起的。当Python无法找到与给定名称相关联的对象时，会引发这种错误。

y=1  
print(x)

在这个示例中，变量x没有定义，因此会引发NameError名称错误。

#### 3.异常

在 Python 中，异常是一个在程序执行过程中发生的错误事件，它会打断正常的指令流，因为某个操作无法正常完成。Python 提供了一套异常处理机制，允许程序捕获和处理这些错误，从而避免程序的非正常终止。  
Python 异常类继承自BaseException ，内置了许多标准异常，它们按类别组织成不同的继承层次结构。一些常见的内置异常包括：

* Exception：所有内置的非系统退出异常都派生自这个基类。
* ArithmeticError：所有算术相关的错误。
  + OverflowError：数值运算超出最大限制。
  + ZeroDivisionError：除数为零。

try:  
 result = 10 / 0  
except Exception as e:  
 print(e)

* AssertionError：断言语句失败。

try:  
 assert 2 + 2 == 5, "断言失败"  
except AssertionError as e:  
 print(e)

* AttributeError：尝试访问未知的对象属性。

try:  
 x = 10  
 print(x.length)  
except AttributeError:  
 print("对象没有 'length' 属性")

* ImportError：导入模块失败。

try:  
 import non\_existent\_module  
except ImportError:  
 print("导入模块失败")

* IndexError：序列中没有此索引。

try:  
 numbers = [1, 2, 3]  
 print(numbers[3])  
except IndexError:  
 print("索引超出范围")

* KeyError：字典中没有这个键。

try:  
 my\_dict = {"name": "John", "age": 30}  
 print(my\_dict["address"])  
except KeyError:  
 print("字典中没有 'address' 键")

* KeyboardInterrupt：用户中断执行。

try:  
 while True:  
 pass  
except KeyboardInterrupt:  
 print("用户中断执行")

* NameError：未声明/初始化对象（没有属性）。

try:  
 print(x)  
except NameError:  
 print("变量 'x' 未声明或未初始化")

* TypeError：操作或函数应用于不适当类型的对象。

try:  
 result = "hello" + 5  
except TypeError:  
 print("无效的操作：字符串不能与整数相加")

* ValueError：传入无效的参数。
* try:  
   num = int("abc")  
  except ValueError:  
   print("无效的参数：不能将字符串转换为整数")
* 异常处理使用 try、except、finally 和可选的 else 块。下面是一个异常处理的基本结构：

#### 4.异常处理模板

异常处理方法:  
在Python中，我们可以使用以下方法来处理异常:

使用try和except块来捕获和处理异常。在try块中放置可能引发异常的代码，然后在except块中处理异常。可以根据需要指定特定类型的异常或使用通用的except块来捕获所有异常。

try:  
 # 可能引发异常的代码  
except ExceptionType1:  
 # 处理特定类型的异常  
except ExceptionType2:  
 # 处理另一种类型的异常  
except:  
 # 处理其他类型的异常

可以使用finally块来执行无论是否发生异常都需要执行的清理操作。

try:  
 # 可能引发异常的代码  
except ExceptionType: #这个地方我通常写成Exception，捕获所有异常。  
 # 处理异常  
finally:  
 # 无论是否发生异常都会执行的代码

#### 5.自定义异常

除了使用Python内置的异常类型，我们还可以自定义异常来满足特定的需求。自定义异常可以继承自Python的Exception类，或者继承自Exception的子类。

以下是一个自定义异常的示例：

class MyCustomException(Exception):  
 def \_\_init\_\_(self, message):  
 self.message = message  
  
try:  
 age = int(input("请输入年龄: "))  
   
 if age < 0:  
 raise MyCustomException("年龄不能为负数")  
 elif age > 120:  
 raise MyCustomException("年龄不合理")  
 else:  
 print("年龄是:", age)  
  
except MyCustomException as e:  
 print("发生自定义异常:", e.message)

在上面的示例中，我们定义了一个名为MyCustomException的自定义异常类，它继承自Exception类。我们通过传递一个错误消息来初始化自定义异常对象。在try块中，我们接收用户输入的年龄，并根据一些条件判断是否引发自定义异常。如果年龄为负数或超过了120岁，我们会分别引发MyCustomException异常，并在except块中处理异常，打印出错误消息。