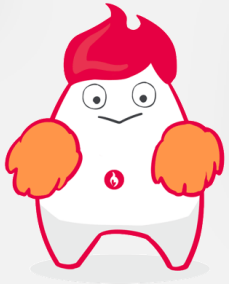
初探异常



本节课内容

- 什么是异常与异常处理
- 异常的语法结构

什么是异常与异常处理

- 异常就是错误
- 异常会导致程序崩溃并停止运行
- 能监控并捕获到异常，将异常部位的程序进行修理使得程序继续正常运行

异常的语法

```
try:
    <代码块1>  被try关键字检查并保护的业务代码

except <异常的类型>:
    <代码块2>  # 代码块1出现错误后执行的代码块
```

异常的语法

```
In [1]: 1 / 0
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-1-bc757c3fda29> in <module>
----> 1 1 / 0

ZeroDivisionError: division by zero
```

异常的语法

```
In [4]: try:
   ...:     1 / 0
   ...: except:
   ...:     print('0不能被1整除')
   ...:     print('程序继续执行')
   ...:
0不能被1整除
程序继续执行
```

捕获通用异常

- 无法确定是哪种异常的情况下使用的捕获方法

```
try:
    <代码块>
except  Exception  as e:
    <异常代码块>
```

捕获通用异常

```
In [3]: try:
   ...:     1 / 0
   ...: except Exception as e:
   ...:     print('try代码块中的语句出错了，具体错误是:{}'.format(e))
   ...:
try代码块中的语句出错了，具体错误是:division by zero
```

捕获具体异常

- 无法确定是哪种异常的情况下使用的捕获方法
- except <具体的异常类型>as e

ZeroDivisionError 是python内置的具体异常：0不能被整除

```
In [7]: try:
   ...:     1 / 0
   ...: except ZeroDivisionError as e:
   ...:     print(e)
   ...:
division by zero
```

捕获多个异常1

```
try:
    print( ‘try start’ )
    res = 1 / 0
    print( ‘try finish’ )
except ZeroDivisionError as e:
    print(e)
except Exception as e:   # 可以有多个except
    print( ‘ this is a public except, bug is :%s’ % e)
```

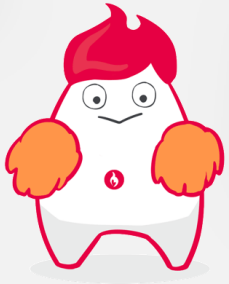
当except代码块有多个的时候，当捕获到第一个后，不会继续往下捕获

捕获多个异常2

```
try:
    print(‘try start’)
    res = 1 / 0
    print(‘try finish’)
except (ZeroDivisionError, Exception) as e:
    print(e)
```

当except代码后边的异常类型使用元组包裹起来，捕获到哪种抛哪种

Python中的异常类型



本节课内容

异常类型集合

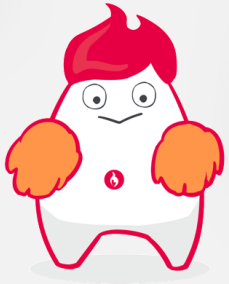
异常类型集合

异常名称	说明
Exception	通用异常类型（基类）
ZeroDivisionError	不能整出0
AttributeError	对象没有这个属性
IOError	输入输出操作失败
IndexError	没有当前的索引

异常类型集合

异常名称	说明
KeyError	没有这个健值（key）
NameError	没有这个变量（未初始化对象）
SyntaxError	Python语法错误
SystemError	解释器的系统错误
ValueError	传入的参数错误

异常中的finally



本节课内容

finally的功能与用法

finally的功能

- 无论是否发生异常，一定会执行的代码块
- 在函数中，即便在try或except中进行了return也依然会执行finally语法块
- try语法至少要伴随except或finally中的一个

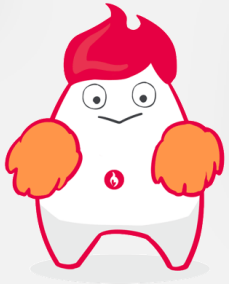
finally的用法

```
try:
    <代码块1>
except:
    <代码块2>
finally:
    <代码块3>
```

finally的历史

在python2.5之前的版本，finally需要独立使用，不可以和try配合，之后才演变成现在的模式

异常中的finally



本节课内容

- 自定义抛出异常 raise
- 自定义异常类

自定义抛出异常函数 --raise

将信息以报错的形式抛出

自定义抛出异常函数 --raise

用法：

raise 异常类型(message)

参数：

message： 错误信息

返回值：

无返回值

自定义抛出异常函数 --raise

```
In [12]: raise ValueError('主动抛出一个异常')
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-12-0b195faebc87> in <module>
----> 1 raise ValueError('主动抛出一个异常')

ValueError: 主动抛出一个异常
```

自定义异常类

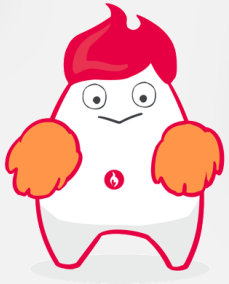
- 继承基类—Exception
- 在构造函数中定义错误信息

```
In [13]: class NewError(Exception):
    ...:     def __init__(self, message):
    ...:         self.message = message
    ...:

In [14]: def test():
    ...:     raise NewError('this is a bug')
    ...:

In [15]: try:
    ...:     test()
    ...: except Exception as e:
    ...:     print(e)
    ...:
this is a bug
```

断言



本节课内容

断言的功能与用法

断言的功能--assert

用于判断一个表达式，在表达式条件为 false 的时候触发异常

断言的用法--assert

用法：

assert expression

参数：

expression： 表达式，一般是判断相等，或者判断是某种数据类型的bool判断的语句

返回值：

无返回值

断言的用法--assert

```
In [16]: assert 1 == 1

In [17]: assert 1 > 2
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-17-f53b9196f459> in <module>
----> 1 assert 1 > 2

AssertionError:
```

什么是bug

bug定义

- 程序中出现的错误，但又没有通过异常去捕获，以至于直接抛出，导致程序的崩溃

bug一词的由来

- bug 指的是 小虫
- 飞入计算机中导致机器停止