潍坊科技学院
Weifang University of Science and Technology

第9章 异常

授课老师：刘国旭

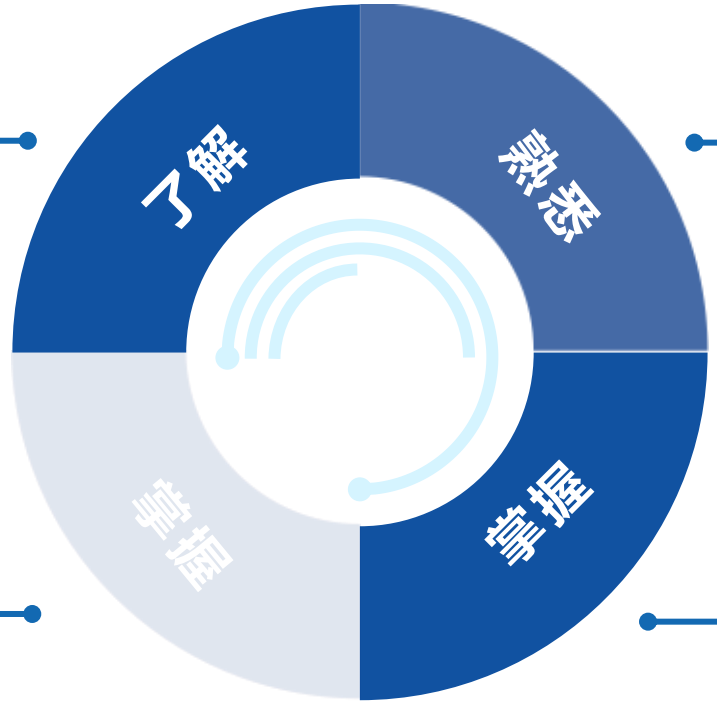
潍坊科技学院

- 异常的概念和类型
- 捕获异常的方式
- raise语句和assert语句
- 异常传递
- 自定义异常

学习目标

1 了解 异常、异常的捕获方式

2 熟悉 raise语句和assert语句



3 掌握 程序如何传递异常

4 掌握 自定义异常的定义与使用

目录页

目录页

9.1 异常概述

程序开发或运行时可能出现异常，开发人员和运维人员需要辨别程序的异常，明确这些异常是源于程序本身的设计问题，还是由外界环境的变化引起，以便有针对性地处理异常。

9.1.1 认识异常

程序运行出现异常时，若程序中没有设置异常处理功能，解释器会采用系统的默认方式处理异常，即返回异常信息、终止程序。

9.1 异常概述

异常信息中通常包含异常代码所在行号、异常的类型和异常的描述信息。

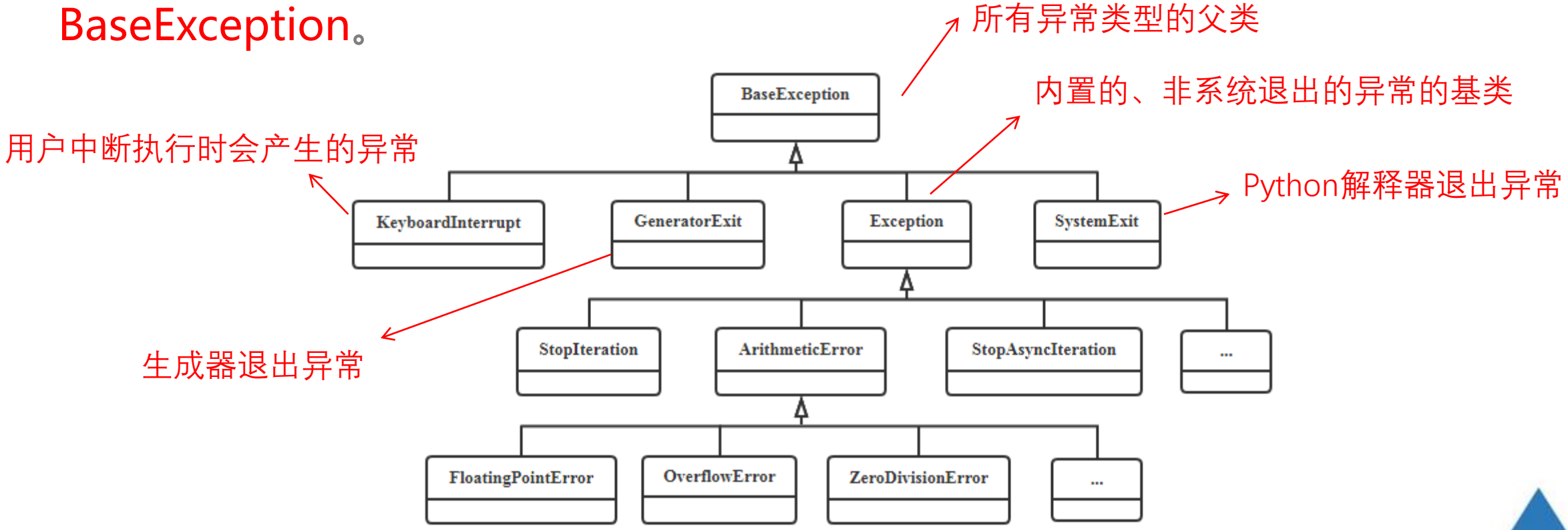
```
print(3/0)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-2-01a2217119a3> in <module>
----> 1 print(3/0)

ZeroDivisionError: division by zero
```

9.1.2 异常的类型

Python程序运行出错时产生的每个异常类型都对应一个类，程序运行时出现的异常大多继承自Exception类，Exception类又继承了异常类的基类BaseException。



9.1.2 异常的类型

1. NameError

- NameError是程序中使用了未定义的变量时会引发的异常。
- 例如，访问一个未声明过的变量test，代码如下：

```
print(test)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-4ddfce83ccd5> in <module>
----> 1 print(test)

NameError: name 'test' is not defined
```

9.1.2 异常的类型

2. IndexError

- IndexError是程序越界访问时会引发的异常。
- 例如，使用索引0访问空列表num_list，代码如下：

```
num_list = []
num_list[0]
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-5-a0ace9c7cd31> in <module>
      1 num_list = []
----> 2 num_list[0]

IndexError: list index out of range
```

9.1.2 异常的类型

3. AttributeError

- AttributeError是使用对象访问不存在的属性时引发的异常。
- 例如，Car类中动态地添加了两个属性color和brand，使用Car类的对象依次访问color、brand属性及不存在的name属性，代码如下：

```
class Car(object):
    pass
car = Car()
car.color = "黑色"
car.brand = '五菱'
car.color
car.brand
car.name
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-6-5d112a938c09> in <module>
      6 car.color
      7 car.brand
----> 8 car.name

AttributeError: 'Car' object has no attribute 'name'
```

9.1.2 异常的类型

4.FileNotFoundError

- FileNotFoundError是未找到指定文件或目录时引发的异常。
- 例如，打开一个本地不存在的文件，代码如下：

```
file = open("test.txt")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)
<ipython-input-7-c7cceb9b5266> in <module>
----> 1 file = open("test.txt")

FileNotFoundError: [Errno 2] No such file or directory: 'test.txt'
```

目录页

9.2 异常捕获语句

Python既可以直接通过try-except语句实现简单的异常捕获与处理的功能，也可以将try-except语句与else或finally子句组合实现更强大的异常捕获与处理的功能。

9.2.1 使用try-except语句捕获异常

try-except语句的语法格式如下：

语法格式

```
try:
    可能出错的代码
except [异常类型 [as error]]:                    # 将捕获到的异常对象赋error
    捕获异常后的处理代码
```

9.2.1 使用try-except语句捕获异常

try-except语句可以捕获与处理程序的单个、多个或全部异常。

捕获单个异常

```
num_one = int(input("请输入被除数："))
num_two = int(input("请输入除数："))
try:
    print("结果为", num_one / num_two)
except ZeroDivisionError:
    print("出错了")
```

单个异常类型

9.2.1 使用try-except语句捕获异常

try-except语句可以捕获与处理程序的单个、多个或全部异常。

说明异常原因

```
num_one = int(input("请输入被除数："))
num_two = int(input("请输入除数："))
try:
    print("结果为", num_one / num_two)
except ZeroDivisionError as error:
    print("出错了，原因：", error)
```

9.2.1 使用try-except语句捕获异常

try-except语句可以捕获与处理程序的单个、多个或全部异常。

捕获多个异常

```
num_one = int(input("请输入被除数："))
num_two = int(input("请输入除数："))
try:
    print("结果为", num_one / num_two)
except (ZeroDivisionError, ValueError) as error:
    print("出错了，原因：", error)
```

多个异常类型

9.2.1 使用try-except语句捕获异常

try-except语句可以捕获与处理程序的单个、多个或全部异常。

捕获全部异常

```
num_one = int(input("请输入被除数："))
num_two = int(input("请输入除数："))
try:
    print("结果为", num_one / num_two)
except Exception as error:
    print("出错了，原因：", error)
```

异常类型设置为Exception或省略不写

9.2.2 异常结构中的else子句

else子句可以与try-except语句组合成try-except-else结构，若try监控的代码没有异常，程序会执行else子句后的代码。

语法格式

```
try:
    可能出错的代码
except [异常类型 [as error]]:                    # 将捕获到的异常对象赋值error
    捕获异常后的处理代码
else:
    未捕获异常后的处理代码
```

9.2.2 异常结构中的else子句

else子句可以与try-except语句组合成try-except-else结构，若try监控的代码没有异常，程序会执行else子句后的代码。

else子句示例

```
first_num = int(input("请输入被除数："))
second_num = int(input("请输入除数："))
try:
    res = first_num/second_num
except ZeroDivisionError as error:
    print('异常原因：',error)
else:
    print(res)
```

9.2.3异常结构中的finally子句

finally子句可以和try-except一起使用，语法格式如下：

语法格式

```
try:
    可能出错的代码
except [异常类型 [as error]]:                    # 将捕获到的异常对象赋值error
    捕获异常后的处理代码
finally:
    一定执行的代码
```

9.2.3异常结构中的finally子句

- 无论try子句监控的代码是否产生异常，finally子句都会被执行
- finally子句多用于预设资源的清理操作，如关闭文件、关闭网络连接

finally子句示例

```
try:
    file = open('./file.txt', mode='r', encoding='utf-8')
    print(file.read())
except FileNotFoundError as error:
    print(error)
finally:
    file.close()
    print('文件已关闭')
```

目录页

9.3 抛出异常

Python程序中的异常不仅可以自动触发异常，而且还可以由开发人员使用raise和assert语句主动抛出异常。

9.3.1 使用raise语句抛出异常

使用raise语句可以显式地抛出异常， raise语句的语法格式如下：

raise 语法格式

```
raise 异常类                    # 格式1：使用异常类名引发指定的异常
raise 异常类对象                # 格式2：使用异常类的对象引发指定的异常
raise                           # 格式3： 使用刚出现过的异常重新引发异常
```

示例：raise 异常类

```
raise IndexError
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-1-55a00e7db5b5> in <module>
----> 1 raise IndexError

IndexError:
```

9.3.1 使用raise语句抛出异常

使用raise语句可以显式地抛出异常， raise语句的语法格式如下：

raise 语法格式

```
raise 异常类                    # 格式1：使用异常类名引发指定的异常
raise 异常类对象                # 格式2：使用异常类的对象引发指定的异常
raise                           # 格式3： 使用刚出现过的异常重新引发异常
```

raise IndexError() 创建异常类对象

示例：raise 异常对象

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-2-7811308d6908> in <module>
----> 1 raise IndexError()

IndexError:
```

9.3.1 使用raise语句抛出异常

使用raise语句可以显式地抛出异常， raise语句的语法格式如下：

raise 语法格式

```
raise 异常类                    # 格式1：使用异常类名引发指定的异常
raise 异常类对象                # 格式2：使用异常类的对象引发指定的异常
raise                           # 格式3： 使用刚出现过的异常重新引发异常
```

```
raise IndexError('索引下标超出范围')   指定异常的具体信息
```

示例：raise 异常对象

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-3-3f8088ae2f45> in <module>
----> 1 raise IndexError('索引下标超出范围')

IndexError: 索引下标超出范围
```

9.3.1 使用raise语句抛出异常

使用raise语句可以显式地抛出异常， raise语句的语法格式如下：

raise 语法格式

```
raise 异常类                    # 格式1：使用异常类名引发指定的异常
raise 异常类对象                # 格式2：使用异常类的对象引发指定的异常
raise                           # 格式3： 使用刚出现过的异常重新引发异常
```

示例：重新引发异常

```
try:
    raise IndexError
except:
    raise
```

9.3.2 使用assert语句抛出异常

assert语句又称为断言语句，其语法格式如下所示：

assert语法格式

```
assert 表达式[，异常信息]
```

示例

```
num_one = int(input("请输入被除数："))
num_two = int(input("请输入除数："))
assert num_two != 0, '除数不能为0'  # assert语句判定num_two不等于0
result = num_one / num_two
print(num_one, '/', num_two, '=', result)
```

9.3.2 使用assert语句抛出异常

示例

```
num_one = int(input("请输入被除数："))
num_two = int(input("请输入除数："))
assert num_two != 0, '除数不能为0'  # assert语句判定num_two不等于0
result = num_one / num_two
print(num_one, '/', num_two, '=', result)
```

```
请输入被除数：4
请输入除数：0
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-4-e51416c95150> in <module>
      1 num_one = int(input("请输入被除数："))
      2 num_two = int(input("请输入除数："))
----> 3 assert num_two != 0, '除数不能为0'  # assert语句判定num_two不等于0
      4 result = num_one / num_two
      5 print(num_one, '/', num_two, '=', result)

AssertionError: 除数不能为0
```

9.3.3 异常的传递

如果程序中的异常没有被处理，默认情况下会将该异常传递到上一级，如果上一级仍然没有处理异常，那么会继续向上传递，直至异常被处理或程序崩溃。

9.3.3 异常的传递

```
def get_width():
    print(“get_width开始执行”)
    num = int(input(“请输入除数：”))
    width_len = 10/num
    print(“get_width执行结束”)
    return width_len
def calc_area():
    print(“calc_area开始执行”)
    width_len = get_width()
    print (“calc_area执行结束”)
    return width_len*width_len
```

示例

```
def show_area():
    try:
        print(“show_area开始执行”)
        area_val = calc_area()
        print(f”正方形的面积是：{area_val}”)
        print(“show_area执行结束”)
    except ZeroDivisionError as e:
        print(f”捕捉到异常：{e}”)


show_area()
```

目录页

9.4 自定义异常

有时我们需要自定义异常类，以满足当前程序的需求。自定义异常的方法比较简单，只需要创建一个继承Exception类或Exception子类的类（类名一般以“Error”为结尾）即可。

9.4 自定义异常

示例

```
class ShortInputError(Exception):
    '''自定义异常类'''
    def __init__(self, length, atleast):
        self.length = length                # 输入的密码长度
        self.atleast = atleast              # 限制的密码长度
try:
    text = input("请输入密码：")
    if len(text) < 3:
        raise ShortInputError(len(text), 3)
except ShortInputError as result:
    print("ShortInputError：输入的长度是%d，长度至少应是 % d" %
          (result.length, result.atleast))
else:
    print("密码设置成功")
```

目录页

9.5.1 头像格式检测

假设某网站只允许用户上传jpg、png和jpeg格式的文件，本实例要求编写代码，通过异常捕获语句实现用户上传头像格式检测的功能。

9.5.2 商品数量检测

用户在进行网购时，需要同时选择商品及数量，只有输入的商品数量不小于1才符合规则，小于1则提示错误信息并设为默认值1。

本实例要求编写代码，实现具有检测商品数量是否符合规则的程序。

9.6 本章小结

本章主要讲解了Python异常的相关知识，包括异常概述、异常捕获语句、抛出异常和自定义异常，同时结合实训案例演示了异常的用法。通过本章的学习，希望大家掌握如何处理异常。