Python及其应用

主讲人：钱惠敏
E-mail: amandaqian@hhu.edu.cn

第9讲 错误类型和异常捕获

9.1 错误类型

9.2 语法错误

9.3 异常

9.4 捕获异常

9.1 错误类型

➢ 错误类型

✓ 语法错误（Syntax errors）

代码编译时的错误，不符合Python语言规则的代码会停止编译并返回错误信息

✓ 异常（Exceptions）

相较于语法错误，异常比较难发现，因为它只在代码运行时才会发生，如类型错误、数值错误、索引错误和属性错误等。

✓ 语法错误包含在异常基类中

9.2 语法错误

- 常见的语法错误（SyntaxError）
  - ✓ 缺少起始符号或结尾符号（括号、引号等）
  - ✓ 缩进错误
  - ✓ 关键词拼写错误

```
path='./Pokemon.csv
```

```
  File "<ipython-input-30-c3992727067c>", line 1
    path='./Pokemon.csv

SyntaxError: EOL while scanning string literal
```

9.2 语法错误(续1)

➢ 常见的语法错误（SyntaxError）

- ✓ 缺少起始符号或结尾符号（括号、引号等）
- ✓ 缩进错误
- ✓ 关键词拼写错误

```
series=[1,2,3,4]
for num in series:
    print (num)
      print  (num)
```

```
  File "<ipython-input-32-faa063183e22>", line 4
    print  (num)
    ^
IndentationError: unexpected indent
```

9.2 语法错误(续2)

- 常见的语法错误（SyntaxError）
  - ✓ 缺少起始符号或结尾符号（括号、引号等）
  - ✓ 缩进错误
  - ✓ 关键词拼写错误

```
for i in range(20):
    if i%2==0 and i%3:
        print (i)
    els:
        print(i)
```

```
  File "<ipython-input-39-a52daf9d3880>", line 4
    els:
       ^

SyntaxError: invalid syntax
```

9.3 异常

- 异常通常由以下问题引起：
  - ✓ 在定义函数之前就引用该函数
  - ✓ 调用不属于某个对象的方法或者属性
  - ✓ 试图将某个值转换为不恰当的数据类型

9.3 异常(续1)

- 六种典型的异常

①除零错误（ZeroDivisionError）：除数为0

②名称错误（NameError）：变量使用前未进行申明或者初

```
num=3
num/0
```

```
---------------------------------------
ZeroDivisionError
<ipython-input-40-60346af1ca64> in <mo
      1 num=3
----> 2 num/0

ZeroDivisionError: division by zero
```

```
variable_a
```

```
---------------------------------------
NameError                                 Tra
<ipython-input-42-3f13b4e447b4> in <module>
----> 1 variable_a

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

9.3 异常(续2)

- 六种典型的异常

③类型错误（TypeError）：某些函数或者方法只适用于特定的数据类型，如果对数据类型的操作不当，就会产生类型错误

```
num = 3
num + '3'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recer

<ipython-input-21-804b0101d483> in <module>()
      1 num = 3
----> 2 num + '3'

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

```
float('3')
float('hackdata')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call la

<ipython-input-23-8bd82f67428d> in <module>()
      1 float('3')
----> 2 float('hackdata')

ValueError: could not convert string to float: hackdata
```

9.3 异常(续3)

- 六种典型的异常

⑤索引错误（IndexError）：超出序列长度的索引操作

⑥属性错误（AttributeError）：方法或者属性不适用该对象

```
list_hack = ['h','a','c','k']
list_hack[4]
```

```
---------------------------------------------

IndexError                                Trace

<ipython-input-25-92bac5b43ef3> in <module>()
      1 l = ['h','a','c','k']
----> 2 l[4]

IndexError: list index out of range
```

```
tuple_hack = ('h','a','c')
tuple.append('k')
```

```
---------------------------------------------------------------------

AttributeError                            Traceback (most recent cal

<ipython-input-26-38f0e0e728dc> in <module>()
      1 tuple_hack = ('h','a','c')
----> 2 tuple.append('k')

AttributeError: 'tuple' object has no attribute 'append'
```

➢ 异常层级

异常	描述
SyntaxError	语法错误
ZeroDivisionError	除数为0
NameError	尝试访问一个没有申明的变量
TypeError	对某种类型进行不恰当的操作
ValueError	传给函数的参数类型不正确，比如给 float() 函数传入字符串型变量
IndexError	索引超出序列范围
AttributeError	尝试访问未知的对象属性

```
BaseException
 +-- SystemExit
 +-- KeyboardInterrupt
 +-- GeneratorExit
 +-- Exception
      +-- StopIteration
      +-- StandardError
      |    +-- BufferError
      |    +-- ArithmeticError
      |    |    +-- FloatingPointError
      |    |    +-- OverflowError
      |    |    +-- ZeroDivisionError
      |    +-- AssertionError
      |    +-- AttributeError
      |    +-- EnvironmentError
      |    |    +-- IOError
      |    |    +-- OSError
      |    |         +-- WindowsError (Windows)
      |    |         +-- VMSError (VMS)
      |    +-- EOFError
      |    +-- ImportError
      |    +-- LookupError
      |    |    +-- IndexError
      |    |    +-- KeyError
      |    +-- MemoryError
      |    +-- NameError
      |    |    +-- UnboundLocalError
      |    +-- ReferenceError
      |    +-- RuntimeError
      |    |    +-- NotImplementedError
      |    +-- SyntaxError
      |    |    +-- IndentationError
      |    |         +-- TabError
      |    +-- SystemError
      |    +-- TypeError
      |    +-- ValueError
      |         +-- UnicodeError
      |              +-- UnicodeDecodeError
      |              +-- UnicodeEncodeError
      |              +-- UnicodeTranslateError
      |
      +-- Warning
           +-- DeprecationWarning
           +-- PendingDeprecationWarning
           +-- RuntimeWarning
           +-- SyntaxWarning
           +-- UserWarning
           +-- FutureWarning
           +-- ImportWarning
```

9.4 捕获异常

- 代码编写环境自带的高亮显示
  - ✓便于发现常规语法错误
  - ✓但难于发现异常

Jupyter Notebook

In [ ]:
```
import csv #引入csv模块

path = './Pokemon.csv'
f = open(path, 'r')

reader = csv.reader(f) #用csv.reader()方法生成reader对象
content = []  #用于数据存储

#for循环，从reader中逐个添加存储
for con in reader:
    content.append(con)
f.close()

print content[0]
print content[1]
```

Spyder

```
# -*- coding: utf-8 -*-
"""
Created on Thu Apr  2 22:37:58 2020

@author: qhmin
"""

########猜字游戏
import random
se=random.randint(1,10) #生成整型随机数
#print(se)
count=0 #记录猜测次数
#猜测次数小于3时，比较输入数据与生成随机数的大小，并做相应动作
while count<3:
    a=int(input("input the data:"))
    if a>se:
        print("the data is too big")
        count=count+1
    elif a<se:
        print("the data is too small")
        count+=1
    else:
        count+=1
        print("%d times is right" %count)
        break
else:
    print("you guessed two many times:%d"%se)

```

9.4 捕获异常(续1)

- 程序要遇到异常的时候，往往是直接中断，跳出执行。但是有些时候，我们需要在遇到异常的时候另外处理，而不是直接停止。

- 解决方法：
  - ✓try...except...语句
  - ✓try...except...else语句
  - ✓finally子句

9.4 捕获异常(续2)

✓ try...except...语句

`try...except...` 是捕获异常最常用的语句，具体语法如下：

```
try:
    # 正常运行代码
except:
    # 不正常运行时的处理
```

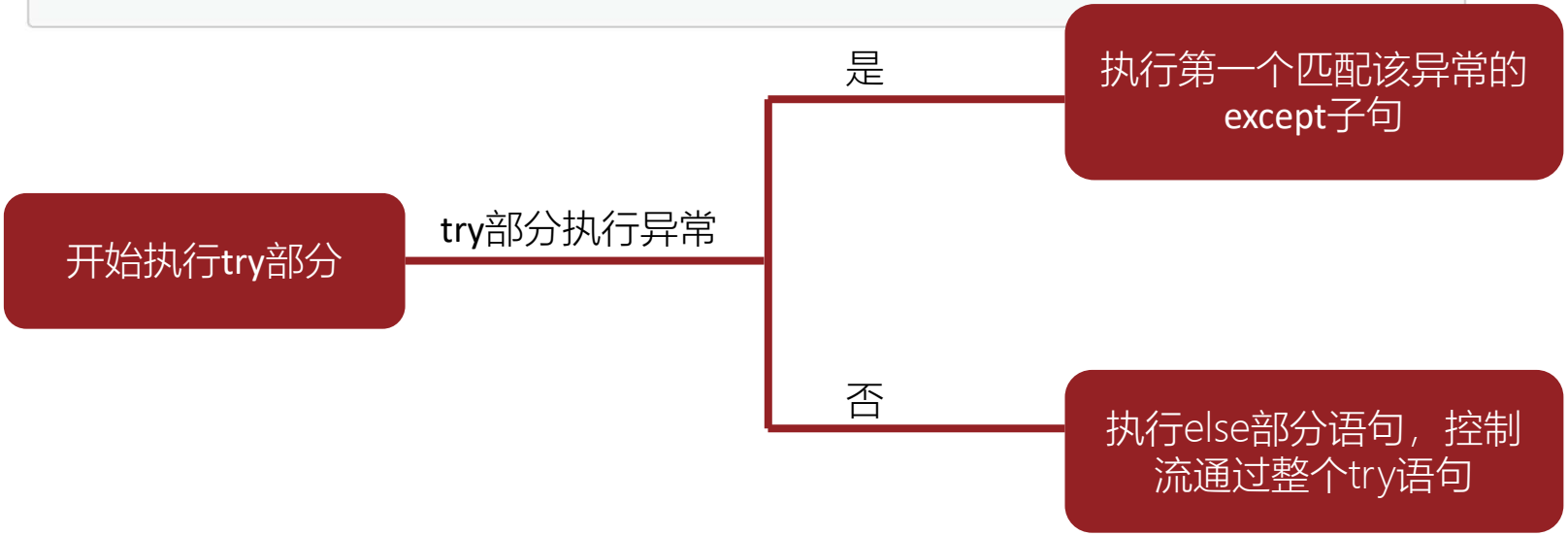
try 关键词内执行的是正常代码，当这部分代码出错的时候，会跳过错误代码后进入 except 关键词内部，执行此部分的代码

9.4 捕获异常(续3)

✓ try...except...else语句

当在 try...except... 后加入的 else 指，当程序没发生错误时执行的部分

```
try:
    # 正常运行
except(Exception1, Exception2, ...),e:
    # 发生Exception1, Exception2,...时的处理方式
else:
    # 正确时执行
```



9.4 捕获异常(续4)

✓ finally子句

finally 语句是指，无论程序运行对或错，都会执行的部分

```
try:
    # 正常运行
except(Exception1, Exception2, ...),e:
    # 发生Exception1, Exception2,...时的处理方式
else:
    # 正确时执行
finally:
    # 无论对错都执行
```

9.4 捕获异常(续5)

- assert语句
  - 当Expression部分为True时，则正确执行，程序继续下去；当判断为False时，则抛出后面的e错误提示。
  - 在大型的项目中，assert常被用来作为“防御性编程”

```
assert Expression, e
```

9.4 捕获异常(续6)

- with语句

有时候打开了文件却忘记关闭，或者是在读取文件过程出错，那么"with"语句能够很好解决关于文件读取、写入的问题

```
with open() as f:
    content = f.readlines()
```

上面的语句等价于

```
f=open()
content=f.readlines()#代码块
f.close()
```

9.4 捕获异常(续7)

- 自主控制异常：用户自定义异常
  - ✓ 自定义异常的原因
    - Python提供的内建异常不够用
    - 可以预估某个错误的产生
  - ✓ 定义异常类
    - 继承于Exception类，由它开始扩展

```
class MyError(Exception):
    pass

raise MyError(u'something error')
```

```
---------------------------------------------------------------------------
--
MyError                                   Traceback (most recent call las
t)
<ipython-input-4-d538cdc34e5a> in <module>()
      2     pass
      3
----> 4 raise MyError(u'something error')

MyError: something error
```

- ✓ 自主定义的NotIntError异常类，捕获非整型错误

```
class NotIntError(Exception):

    def __init__(self, error):
        self.error = error


a = [1, 2, "", 4, 5, "a", ['1', '2', '3']]

for i in range(len(a)):

    try:
        if type(a[i]) != int:
            raise NotIntError("错误")

    except NotIntError,e:
        print e.error

    finally:
        print a[i]
```

9.4 捕获异常(续8)

- 使用异常代替返回状态码
  - ✓ 编写工具类函数时，函数处理流程会产生很多状态
  - ✓ 用返回值代表函数处理状态，调用者需要去理解每个状态码的意义，存在学习成本

```
# 定义函数
def write(content):
    if isinstance(content, str):
        f = open("file.txt", 'w')
        try:
            f.write(content)
        except Exception:
            return -2     # 写入文件失败
        else:
            return 0      # 写入文件成功
        finally:
            f.close()
    else:
        return -1     # 输入类型错误
```

```
# 调用函数
import logging # 打印日志信息
result = write(2)
if result == -1:
    logging.error(u"type error")
elif result == -2:
    logging.error(u"write error")
else:
    logging.info("ok")
```

```
ERROR:root:type error
```

9.4 捕获异常(续9)

- 异常处理与流程控制
  - 异常处理应该与正常流程控制分离

异常处理搞乱了代码逻辑

```
try:
    action_a()
    action_b()
    action_c()
except ActionException as e:
    loggging.error(e)
else:
    action_d()
```

将异常代码块抽离到另外的函数中

```
def action_executor():
    action_a()
    action_b()
    action_c()

def action():
    try:
        action_executor()
    except ActionException as e:
        logging.error(e)

action()
action_d()
```