《Python程序设计基础教程（微课版）》

http://dblab.xmu.edu.cn/post/python

第9章 异常处理

林子雨 博士/副教授

厦门大学计算机科学与技术系

E-mail: ziyulin@xmu.edu.cn

主页：http://dblab.xmu.edu.cn/linziyu

主讲教师

2017年度厦门大学奖教金获得者
2020年度厦门大学奖教金获得者

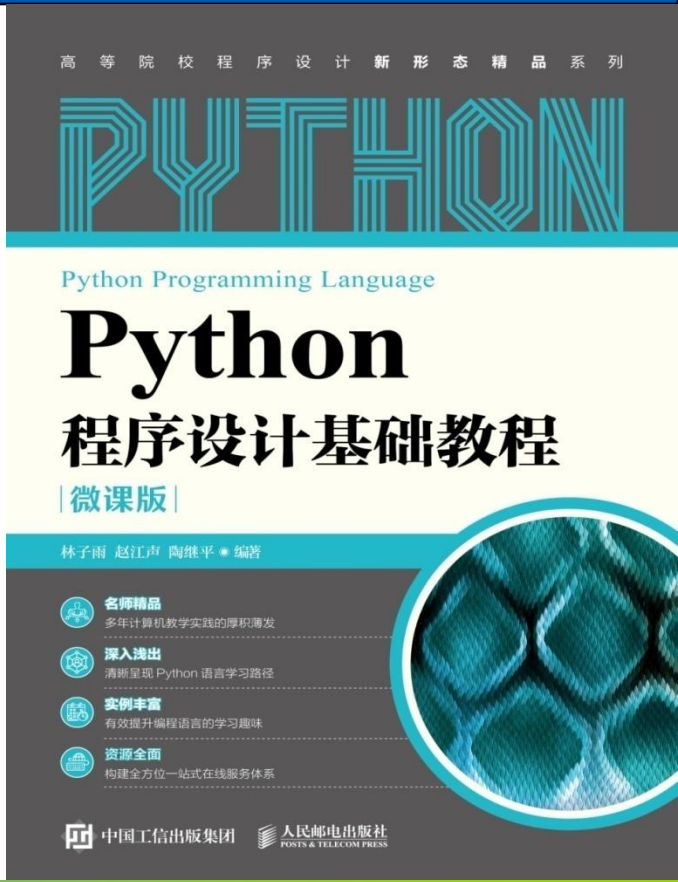
主讲教师：厦门大学 林子雨 博士/副教授
中国高校首个“**数字教师**”提出者和建设者
2009年7月从事教师职业以来
累计**免费**网络发布超过1500万字高价值教学和科研资料
网络浏览量超过1500万次

提纲

本PPT是如下教材的配套讲义：
《Python程序设计基础教程（微课版）》

厦门大学 林子雨,赵江声,陶继平 编著，人民邮电出版社

《 Python程序设计基础教程（微课版） 》教材官方网站：
http://dblab.xmu.edu.cn/post/python



9.1 异常的概念

•异常是程序运行过程中产生的错误。在程序解析时没有出现错误，语法正确但在运行期间出现错误的情况即为异常。

•引发异常的原因有很多，除以零、溢出异常、下标越界、不同类型的变量运算、内存错误等都会产生异常。

9.1 异常的概念

•以下为在交互式运行环境中执行的语句出现异常的例子：

```
>>> 1 /0                        #除以零
ZeroDivisionError: division by zero
>>> '2' + 2                     #类型错误
TypeError: can only concatenate str (not "int") to str
>>> 4 + var                     #变量未定义
NameError: name 'var' is not defined
>>> fp = open('file1.txt','r')       #文件不存在
FileNotFoundError: [Errno 2] No such file or directory: 'file1.txt'
>>> len(100)                    #类型错误
TypeError: object of type 'int' has no len()
```

9.1 异常的概念

以下为在IDLE中新建文件并运行出现异常的例子：

```
01  # exception01.py
02  la = [1,2,3]
03  la[3] = 100      #下标越界
04  print (la)
```

上面代码的执行后会报以下错误：

IndexError: list assignment index out of range

9.1 异常的概念

- 【例】执行代码引发异常。

```
01  # exception02.py
02  def sum(a,b):
03      sum = a + b
04      return sum
05
06  a = 'str'
07  b = 100
08  sum(a,b)     #类型错误
```

上面代码的执行后会报以下错误：
TypeError: can only concatenate str (not "int") to str

从上面的例子中可以看出，异常有不同的类型，其错误类型和描述显示在运行结果的信息中，例子中的错误类型包括ZeroDivisionError、NameError、FileNotFoundError、IndexError 和 TypeError等。

9.2 内置异常类层次结构

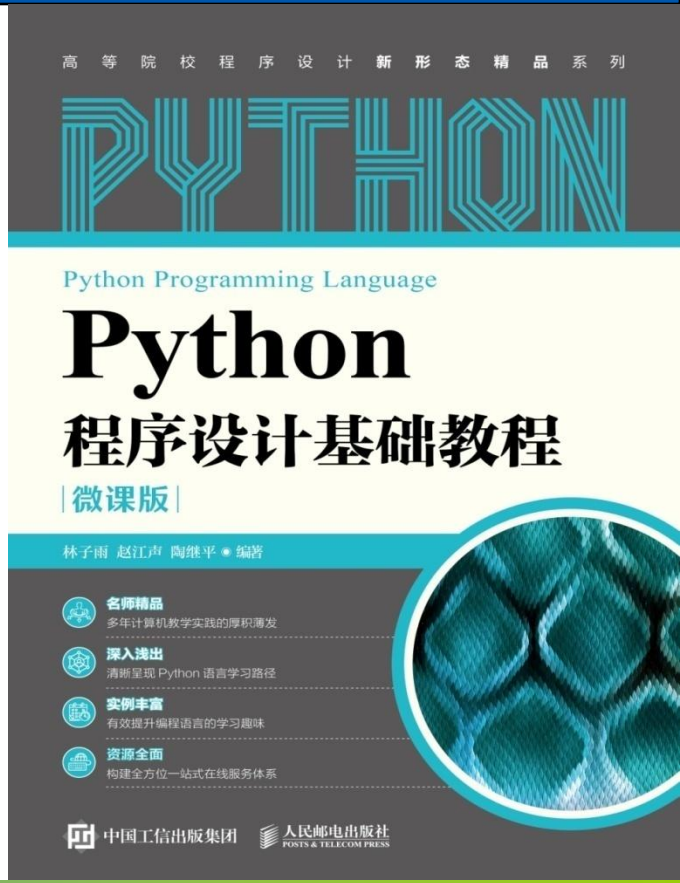
- Python中有很多内置的异常类，有着树状继承关系。（详见图9-1 Python内置异常类层次结构图）
- BaseException是所有内置异常类的基类。

9.2 内置异常类层次结构

```
•BaseException
• +-- SystemExit
• +-- KeyboardInterrupt
• +-- GeneratorExit
• +-- Exception
•      +-- StopIteration
•      +-- StopAsyncIteration
•      +-- ArithmeticError
•      |    +-- FloatingPointError
•      |    +-- OverflowError
•      |    +-- ZeroDivisionError
•      +-- AssertionError
•      +-- AttributeError
•      +-- BufferError
•      +-- EOFError
•      +-- ImportError
•      |    +-- ModuleNotFoundError
•      +-- LookupError
•      |    +-- IndexError
•      |    +-- KeyError
•      +-- MemoryError
•      +-- NameError
•      |    +-- UnboundLocalError
•      +-- OSError
•      |    +-- BlockingIOError
•      |    +-- ChildProcessError
•      |    +-- ConnectionError
•      |    |    +-- BrokenPipeError
•      |    |    +-- ConnectionAbortedError
•      |    |    +-- ConnectionRefusedError
•      |    |    +-- ConnectionResetError
•      |    +-- FileExistsError
|      +-- FileNotFoundError
|      +-- InterruptedError
|      +-- IsADirectoryError
|      +-- NotADirectoryError
|      +-- PermissionError
|      +-- ProcessLookupError
|      +-- TimeoutError
+-- ReferenceError
+-- RuntimeError
|    +-- NotImplementedError
|    +-- RecursionError
+-- SyntaxError
|    +-- IndentationError
|         +-- TabError
+-- SystemError
+-- TypeError
+-- ValueError
|    +-- UnicodeError
|         +-- UnicodeDecodeError
|         +-- UnicodeEncodeError
|         +-- UnicodeTranslateError
+-- Warning
     +-- DeprecationWarning
     +-- PendingDeprecationWarning
     +-- RuntimeWarning
     +-- SyntaxWarning
     +-- UserWarning
     +-- FutureWarning
     +-- ImportWarning
     +-- UnicodeWarning
     +-- BytesWarning
     +-- ResourceWarning
```

9.3 异常处理结构

9.3.1 try/except
9.3.2 try/except...else...
9.3.3 try/except...finally...
9.3.4 try/except...else...finally...



本PPT是如下教材的配套讲义：
《Python程序设计基础教程（微课版）》

厦门大学 林子雨,赵江声,陶继平 编著，人民邮电出版社

《 Python程序设计基础教程（微课版） 》教材官方网站：
http://dblab.xmu.edu.cn/post/python

9.3.1 try/except

•异常处理结构“try/except”是Python异常处理的基本形式。
•一般会把有可能引发异常的代码放在try子句中的代码块中，而except子句的代码块则是用来处理相应的异常。

图9-2 异常处理结构（try/except）

9.3.1 try/except

异常处理结构“try/except”的工作方式如下：

•□ 首先执行try子句代码块（在关键字try和except之间的语句）。如果没有产生异常，则忽略except子句，try子句执行后正常结束；

•□ 如果try子句代码块在执行过程中产生异常，则立即被捕获，同时跳出try子句代码块，进入except子句代码块，进行异常处理。在except子句的代码块中，可以根据try子句抛出的异常的不同类型，进行相应的处理。如果异常的类型与except之后的名称相同，则对应的except子句被执行；

•如果一个异常没有与except匹配，则该异常会传递给上层的try中（如果有的话）。

9.3.1 try/except

下面示例代码完成的任务是：不断接收用户的输入（要求输入整数，不接收其他类型的输入），并做除法运算。

```
01  # exception03.py
02  total = 100
03  while True:
04      x = input("请输入整数：")
05      try:
06          x = int(x)
07          val = total / x
08          print("==>您输入的整数为：",x, " **** 商为：",val)
09      except Exception as e:
10          print("Error! ",e)
```

9.3.1 try/except

上面代码的执行结果如下：

```
请输入整数：1
==>您输入的整数为： 1 **** 商为： 100.0
请输入整数：2
==>您输入的整数为： 2 **** 商为： 50.0
请输入整数：3str
Error!  invalid literal for int() with base 10: '3str'
请输入整数：0
Error!  division by zero
```

9.3.1 try/except

上面的结构为基本结构，仅能捕捉一种异常进行处理。如果需要针对多种不同的异常进行相应的处理，则需要增加多个except子句，语法格式如下：

```
try:
    #可能引发异常的代码块
except Exception1:
    #处理类型为Exception1的异常
except Exception2:
    #处理类型为Exception2的异常
except Exception3:
    #处理类型为Exception3的异常
…
```

这个处理结构的工作方式与基本结构稍有不同，在第2步，如果try子句代码块产生异常，则按顺序依次检查每个except后的异常类型，直到异常类型与某个except之后的名称相同，其对应的except子句被执行。其他的都相似。

9.3.2 try/except...else...

异常处理结构“try/except...else...”（如图9-3所示）是在“try/except”结构中增加else子句，注意else子句应该放在所有的except子句之后。

图9-3 异常处理结构（try/except...else...）

9.3.2 try/except...else...

异常处理结构“try/except...else...”的语法结构如下：

```
try:
    #可能引发异常的代码块
except Exception1:
    #处理类型为Exception1的异常
except Exception2:
    #处理类型为Exception2的异常
except Exception3:
    #处理类型为Exception3的异常
...
else:
    #如果try子句代码块没有引发异常，则继续执行else子句代码块
```

9.3.2 try/except...else...

•该处理结构的工作方式是：如果try子句代码块产生异常，则执行其后的一个或多个except子句，进行相应的异常处理，而不去执行else子句代码块；如果try子句代码块没有产生异常，则执行else子句代码块。

•这种结构的好处是不需要把过多的代码放在try子句中，而是放那些真的有可能产生异常的代码。

9.3.2 try/except...else...

具体实例如下：

```
01  # exception04.py
02  fname = "d:\\Temp\\info1.log"
03  try:
04      f = open(fname, 'r')
05  except IOError:
06      print('无法打开文件', fname)
07  else:
08      print(fname, '有', len(f.readlines()), '行。')
09      f.close()
```

上面代码执行后会报以下错误：
无法打开文件 d:\Temp\info1.log

9.3.3 try/except...finally...

异常处理结构“try/except...finally...”（如图9-4所示）是在“try/except”基本结构中增加finally子句，不管try子句代码块是否产生异常，也不管异常是否被except子句所捕获，都将执行finally子句代码块。

图9-4 异常处理结构（try/except...finally...）

9.3.3 try/except...finally...

异常处理结构“try/except...finally...”的语法结构如下：

```
try:
    #可能引发异常的代码块
except Exception1:
    #处理类型为Exception1的异常
except Exception2:
    #处理类型为Exception2的异常
except Exception3:
    #处理类型为Exception3的异常
...
finally:
    #无论try子句代码块是否产生异常，都会执行finally子句代码块
```

注意，该语法结构中可以没有except子句，此时finally子句都将执行；若try子句中产生了异常，则在finally子句执行后被抛出。

9.3.3 try/except...finally...

下面的例子中，我们按顺序检查每个except后的异常类型，对try子句代码块产生的异常进行捕捉，如果某个except子句捕捉到，则执行其代码块进行相应处理。但无论是否产生异常，或者是否被捕捉到，finally子句都将执行。

```
01  # exception05.py
02  try:
03      val = 1/0
04      pass
05  except FloatingPointError as ex1:      #未能捕捉到异常
06      print("ex1:",ex1)
07  except ZeroDivisionError as ex2:       #捕捉到异常
08      print("ex2:",ex2)
09  finally:
10     print("都要处理finally子句！“)
```

代码的执行结果如下：

```
ex2: division by zero
都要处理finally子句！
```

9.3.3 try/except...finally...

特别地，如果try子句中产生的异常没有被except子句捕捉到，或者except子句或else子句中又产生新的异常，则这些异常会在finally子句执行后再次抛出。

```
01  # exception06.py
02  try:
03      val = 1/0
04      pass
05  except FloatingPointError as ex1:       #未能捕捉到异常
06      print("ex1:",ex1)
07
08  #except BaseException as BaseEx:    #可以捕捉到所有异常，此处注释掉
09  #    print("BaseEx:",BaseEx)
10
11  finally:
12     print("都要处理finally子句！")
```

代码的执行结果如下：

```
都要处理finally子句！
  （略去详细信息）
ZeroDivisionError: division by zero
```

9.3.4 try/except...else...finally...

完整的Python异常处理结构同时包括try子句、多个except子句、else子句和finally子句（如图9-5所示）。

图9-5 异常处理结构（try/except...else...finally...）

9.3.4 try/except...else...finally...

异常处理结构“try/except...else...finally...”的语法结构如下：

```
try:
    #可能引发异常的代码块
except Exception1:
    #处理类型为Exception1的异常
except Exception2:
    #处理类型为Exception2的异常
except Exception3:
    #处理类型为Exception3的异常
...
else:
    #如果try子句代码块没有引发异常，则继续执行else子句代码块
finally:
    #无论try子句代码块是否产生异常，都会执行finally子句代码块
```

9.3.4 try/except...else...finally...

```
01  # exception07.py
02  while True:
03      x = input("请输入整数类型的 除 数：")
04      y = input("请输入整数类型的被除数：")
05      try:
06          x = int(x)
07          y = int(y)
08          val = y / x
09      except TypeError:
10          print("TypeError")
11      except ZeroDivisionError:
12          print("ZeroDivisionError")
13      except Exception as e:
14          print("Error! ",e)
15      else:
16          print("No Error!")
17          print("x=",x," y=",y, " y/x=",val,"\n")
18      finally:
19          print("finally子句都要执行！ \n“)
```

9.3.4 try/except...else...finally...

上面代码的执行结果如下：
请输入整数类型的 除 数：1
请输入整数类型的被除数：100
No Error!
x= 1 y= 100 y/x= 100.0

finally子句都要执行！

请输入整数类型的 除 数：1str
请输入整数类型的被除数：100
Error! invalid literal for int() with base 10: '1str'
finally子句都要执行！

请输入整数类型的 除 数：0
请输入整数类型的被除数：100
ZeroDivisionError
finally子句都要执行！

9.4 抛出异常

Python中使用raise语句强制抛出指定的异常，具体实例如下：

```
>>> raise AttributeError('false attribute')
Traceback (most recent call last):
AttributeError: false attribute

>>> raise Exception
Traceback (most recent call last):
Exception

>>> raise Exception("AException")
Traceback (most recent call last):
Exception: AException
```

raise只有唯一的参数，就是要抛出的指定的异常。参数必须是一个异常的实例或者是派生自Exception的异常类。

9.5 断言

断言（assert）语句是利用异常来判断某个条件是否满足的一个常用的编程技巧。
assert语句用来判断一个表达式，如果该表达式为False时触发AssertionError异常。
以下为一个断言的例子：

```
>>> assert 1==1          #表达式为True，不会触发异常
>>> assert True          #表达式为True，不会触发异常

>>> assert 1==2          #表达式为False，触发AssertionError异常
AssertionError

>>> assert False         #表达式为False，触发AssertionError异常
AssertionError
```

断言的语法形式如下：

```
assert expression
```

或

```
assert expression,argument
```

9.5 断言

表9-1给出了assert语句与if语句的等价关系。

表9-1 assert语句与if语句的等价关系

assert语句	等价的if语句
assert expression	if not expression: raise AssertionError
assert expression,argument	if not expression: raise AssertionError(argument)

9.5 断言

下面的例子假设代码必须只能在unix下才能执行，由于当前运行环境为win32，所以assert语句会触发异常，try子句后的代码没有执行。

```
01  # exception08.py
02  import sys
03  print ("当前sys.platform : ",sys.platform)
04  try:
05      assert('unix' in sys.platform),"代码只能在unix下执行。"
06      print("unix下执行的代码")
07  except AssertionError as err:
08      print("%s : %s"%(err.__class__.__name__,err))
```

上面代码的执行结果如下：

```
当前sys.platform :  win32
AssertionError : 代码只能在unix下执行。
```

9.5 断言

•注意，assert语句一般只在开发的测试阶段使用。

•如果配置优化选项-o或--oo，Python将程序解析为字节码文件时，assert语句会被删除。

9.6 用户自定义异常

•通过前面的介绍，我们对Python内置的异常类有了初步的了解，其实Python还支持用户自定义异常类。用户自定义异常类需继承自Exception类，可以直接继承或者间接继承。

•下面的实例定义了一个名为CustomError的异常类，继承自Exception类，重载了__init__方法，并自定义了__GetStr__方法，返回该异常对象的字符串表示。

9.6 用户自定义异常

```
01  # exception09.py
02  class CustomError(Exception):
03      def __init__(self, value):
04          self.value = value
05          self.__class__.__name__ = "【用户自定义异常】" + self.__class__.__name__
06      def __GetStr__(self):
07          return repr(self)
08
09  try:
10      raise CustomError("hehe!")
11  except CustomError as e:
12      print(e.__class__.__name__,'已触发，值为：', e.value )
13      print(e.__GetStr__())
```

上面代码的执行结果如下：

【用户自定义异常】CustomError 已触发，值为： hehe!
【用户自定义异常】CustomError('hehe!')

9.6 用户自定义异常

此时在IDLE交互环境中执行抛出用户自定义异常，结果如下：

```
>>> raise CustomError("the error!")
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    raise CustomError("the error!")
CustomError: the error!
```

9.7 定义清理操作

在9.3节中，我们了解了异常处理的几种结构。如果仅有try子句，当输入代码“try: raise ZeroDivisionError”时，则或者等待继续输入，或者在程序解释时报语法错误。try子句需要有另一个可选子句，为其定义必须在所有的情况下执行的清理操作。例如：

```
>>> try:
        raise ZeroDivisionError
finally:
        print("百川东到海")
百川东到海
Traceback (most recent call last):
  File "<pyshell#5>", line 2, in <module>
    raise ZeroDivisionError
ZeroDivisionError
```

9.7 定义清理操作

上面这个实例中，try子句触发一个异常，在finally子句执行后被重新触发，没有得到有效处理。可见所有的异常都必须进行清理。

根据官方资料，下面列出一些异常发生时的较为复杂的情况：

•如果在执行 try 子句期间发生了异常，该异常可由一个 except 子句捕获并进行处理。如果异常没有被某个 except 子句所捕获处理，则该异常会在 finally 子句执行之后被重新引发；

•异常也可能在 except 或 else 子句执行期间发生，与上述相同，该异常会在 finally 子句执行之后被重新引发；

•如果在执行 try 语句时遇到一个 break、continue 或 return 语句，则 finally 子句将在执行 break、continue 或 return 语句之前被执行；

•如果 finally 子句中包含一个 return 语句，则整个异常处理结构所返回的值将来自 finally 子句的某个 return 语句的返回值，而非来自 try 子句的 return 语句的返回值。

9.7 定义清理操作

下面是一个具体实例：

```
>>> try:
        val = 1/0
except EOFError as e:
    print(e)
finally:
    print("都要处理finally子句！")
```

运行结果如下：

```
都要处理finally子句！
Traceback (most recent call last):
  File "<pyshell#0>", line 2, in <module>
    val = 1/0
ZeroDivisionError: division by zero
```

9.7 定义清理操作

在上面的实例中，try子句中产生了ZeroDivisionError异常，而except子句仅处理EOFError异常，并未捕获ZeroDivisionError异常，所以该异常在finally子句执行后被引发。该异常清理方式存在的问题，暴露出程序不够健壮，因此，可以将上例加以修改，将在try子句中各种类型的异常捕获，具体如下所示：

```
>>> try:
    val = 1/0
except EOFError as e:
    print(e)
except BaseException as baseEx:
    print(baseEx,"被捕获！")
finally:
    print("都要处理finally子句！“)
```

运行结果如下：
division by zero 被捕获！
都要处理finally子句！

9.7 定义清理操作

可以看出，上面这个实例对于try子句的异常的清理操作较为彻底。但是，如果异常在 except 或 else 子句执行期间发生，那么运行结果会是怎样呢？如在下例中，except子句已经捕获到ZeroDivisionError，在后续的处理过程中又抛出异常（抛出BufferError异常）：

```
>>> try:
    val = 1/0
except ZeroDivisionError as e:
    print("执行except语句块，捕获%s异常并处理"%(e.__class__.__name__))
    pass
    raise BufferError
except BaseException as baseEx:
    print(baseEx,"被捕获！")

finally:
    print("都要处理finally子句！")
```

运行结果如下：
执行except语句块，捕获ZeroDivisionError异常并处理
都要处理finally子句！
Traceback (most recent call last):
  File "<pyshell#13>", line 2, in <module>
    val = 1/0
ZeroDivisionError: division by zero

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "<pyshell#13>", line 6, in <module>
    raise BufferError
BufferError

9.7 定义清理操作

•从上面例子中可以发现，finally子句在任何情况下都会被执行。但是，在处理完finally子句后，不但except子句中新产生的异常被引发，而且except子句已经捕获过的异常也被一并引发。

•这种方式下，异常被引发出来而非被except子句捕获，显得程序并不友好，总的来说还是程序健壮性不够，其解决的办法有如下两种：
（1）养成良好的编程习惯，在except子句中尽量只是显示信息或者打印日志，减少再次抛出异常的几率；
（2）如果实在有必要在except子句中增加复杂的业务逻辑，则可以再增加一重“try...finally...”结构，以确保不会再次引发异常。

9.8 返回值的取值选择

在异常处理结构中，还需要注意，对于不同的结构类型，其返回值的取值选择会随之不同，具体如下：

•对于“try/except”或“try/except...else...”，如果try子句和else子句都包含一个 return 语句，则整个异常处理结构所返回的值，将来自 try 子句的 return 语句的返回值，而非else子句的返回值，因为此时else子句不会被执行；

•对于“try...finally...”结构，如果try子句和 finally 子句中都包含一个 return 语句，则整个异常处理结构所返回的值，将来自 finally 子句的某个 return 语句的返回值，而非来自 try 子句的 return 语句的返回值。

9.8 返回值的取值选择

下面是一个“try/except...else...”结构中返回值的取值选择的例子。

```
01  # exception10.py
02  def num_return():
03      try:
04          return 1
05      except Exception as e:
06          print (e)
07          return 2
08      else:
09          return 3
10
11  print(num_return())
```

- 可见，对于“try/except”或“try/except...else...”结构，其返回值的取值选择一般由try子句的返回值决定，除非在return之前有异常产生。

程序的执行结果为：
1

9.8 返回值的取值选择

对于“try...finally...”结构的返回值的取值选择，请看下面这个实例：

```
01  # exception11.py
02  def bool_return():
03      try:
04          return True
05      finally:
06          return False
07
08  print(bool_return())
```

上面代码的执行结果如下：

```
False
```

9.8 返回值的取值选择

下面是一个更加复杂的例子：

```
01  # exception12.py
02  def integer_return():
03      try:
04          return 1
05      except Exception as e:
06          return 2
07      else:
08          return 3
09      finally:
10          return 4
11
12  print(integer_return())
```

•可见对于“try...finally...”结构，finally子句是必须执行的，如果finally子句中有return语句将要执行，则其他子句的return语句不会执行。

上面代码的执行结果如下：

```
4
```

附录A：主讲教师林子雨简介

主讲教师：林子雨

单位：厦门大学计算机科学与技术系
E-mail: ziyulin@xmu.edu.cn
个人网页：http://dblab.xmu.edu.cn/post/linziyu
数据库实验室网站：http://dblab.xmu.edu.cn

扫一扫访问个人主页

林子雨，男，1978年出生，博士（毕业于北京大学），全国高校知名大数据教师，现为厦门大学计算机科学系副教授，厦门大学信息学院实验教学中心主任，曾任厦门大学信息科学与技术学院院长助理、晋江市发展和改革局副局长。中国计算机学会数据库专业委员会委员，中国计算机学会信息系统专业委员会委员。国内高校首个“数字教师”提出者和建设者，厦门大学数据库实验室负责人，厦门大学云计算与大数据研究中心主要建设者和骨干成员，2013年度、2017年度和2020年度厦门大学教学类奖教金获得者，荣获2019年福建省精品在线开放课程、2018年厦门大学高等教育成果特等奖、2018年福建省高等教育教学成果二等奖、2018年国家精品在线开放课程。主要研究方向为数据库、数据仓库、数据挖掘、大数据、云计算和物联网，并以第一作者身份在《软件学报》《计算机学报》和《计算机研究与发展》等国家重点期刊以及国际学术会议上发表多篇学术论文。作为项目负责人主持的科研项目包括1项国家自然科学青年基金项目(No.61303004)、1项福建省自然科学青年基金项目(No.2013J05099)和1项中央高校基本科研业务费项目(No.2011121049)，主持的教改课题包括1项2016年福建省教改课题和1项2016年教育部产学协作育人项目，同时，作为课题负责人完成了国家发改委城市信息化重大课题、国家物联网重大应用示范工程区域试点泉州市工作方案、2015泉州市互联网经济调研等课题。中国高校首个“数字教师”提出者和建设者，2009年至今，“数字教师”大平台累计向网络免费发布超过1000万字高价值的研究和教学资料，累计网络访问量超过1000万次。打造了中国高校大数据教学知名品牌，编著出版了中国高校第一本系统介绍大数据知识的专业教材《大数据技术原理与应用》，并成为京东、当当网等网店畅销书籍；建设了国内高校首个大数据课程公共服务平台，为教师教学和学生学习大数据课程提供全方位、一站式服务，年访问量超过400万次，累计访问量超过1500万次。

附录B：大数据学习路线图

大数据学习路线图访问地址：http://dblab.xmu.edu.cn/post/10164/

附录C：林子雨大数据系列教材

林子雨大数据系列教材
用于导论课、专业课、实训课、公共课

了解全部教材信息：http://dblab.xmu.edu.cn/post/bigdatabook/

附录D：《大数据导论（通识课版）》教材

开设全校公共选修课的优质教材

本课程旨在实现以下几个培养目标：

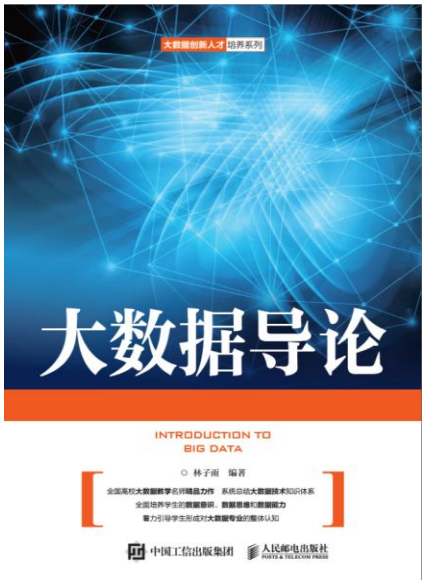
- 引导学生步入大数据时代，积极投身大数据的变革浪潮之中
- 了解大数据概念，培养大数据思维，养成数据安全意识
- 认识大数据伦理，努力使自己的行为符合大数据伦理规范要求
- 熟悉大数据应用，探寻大数据与自己专业的应用结合点
- 激发学生基于大数据的创新创业热情

高等教育出版社 ISBN:978-7-04-053577-8 定价：32元 版次：2020年2月第1版
教材官网：http://dblab.xmu.edu.cn/post/bigdataintroduction/

附录E：《大数据导论》教材

- 林子雨 编著《大数据导论》
- 人民邮电出版社，2020年9月第1版
- ISBN:978-7-115-54446-9 定价：49.80元

教材官网：http://dblab.xmu.edu.cn/post/bigdata-introduction/



开设大数据专业导论课的优质教材

扫一扫访问教材官网

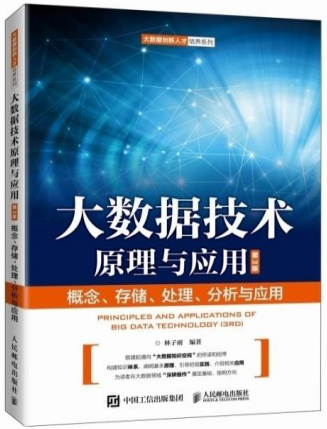
附录F：《大数据技术原理与应用（第3版）》教材

《大数据技术原理与应用——概念、存储、处理、分析与应用（第3版）》，由厦门大学计算机科学系林子雨博士编著，是国内高校第一本系统介绍大数据知识的专业教材。人民邮电出版社 ISBN:978-7-115-54405-6 定价：59.80元

全书共有17章，系统地论述了大数据的基本概念、大数据处理架构Hadoop、分布式文件系统HDFS、分布式数据 库HBase、NoSQL数据库、云数据库、分布式并行编程模型MapReduce、Spark、流计算、Flink、图计算、数据可视化以及大数据在互联网、生物医学和物流等各个领域的应用。在Hadoop、HDFS、HBase、MapReduce、Spark和Flink等重要章节，安排了入门级的实践操作，让读者更好地学习和掌握大数据关键技术。

本书可以作为高等院校计算机专业、信息管理等相关专业的大数据课程教材，也可供相关技术人员参考、学习、培训之用。
欢迎访问《大数据技术原理与应用——概念、存储、处理、分析与应用》教材官方网站：
http://dblab.xmu.edu.cn/post/bigdata3

扫一扫访问教材官网



附录G：《大数据基础编程、实验和案例教程（第2版）》

本书是与《大数据技术原理与应用（第3版）》教材配套的唯一指定实验指导书

大数据教材

1+1黄金组合

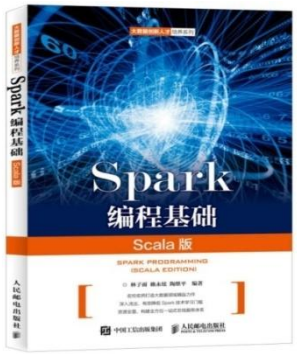
厦门大学林子雨编著

配套实验指导书

- 步步引导，循序渐进，详尽的安装指南为顺利搭建大数据实验环境铺平道路
- 深入浅出，去粗取精，丰富的代码实例帮助快速掌握大数据基础编程方法
- 精心设计，巧妙融合，八套大数据实验题目促进理论与编程知识的消化和吸收
- 结合理论，联系实际，大数据课程综合实验案例精彩呈现大数据分析全流程

林子雨编著《大数据基础编程、实验和案例教程（第2版）》
清华大学出版社 ISBN:978-7-302-55977-1 定价：69元 2020年10月第2版

附录H：《Spark编程基础（Scala版）》



《Spark编程基础（Scala版）》

厦门大学 林子雨，赖永炫，陶继平 编著

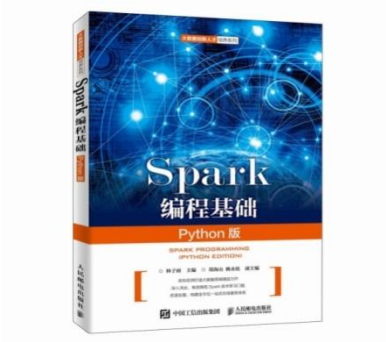
披荆斩棘，在大数据丛林中开辟学习捷径
填沟削坎，为快速学习Spark技术铺平道路
深入浅出，有效降低Spark技术学习门槛
资源全面，构建全方位一站式在线服务体系

人民邮电出版社出版发行，ISBN:978-7-115-48816-9

教材官网：http://dblab.xmu.edu.cn/post/spark/

本书以Scala作为开发Spark应用程序的编程语言，系统介绍了Spark编程的基础知识。全书共8章，内容包括大数据技术概述、Scala语言基础、Spark的设计与运行原理、Spark环境搭建和使用方法、RDD编程、Spark SQL、Spark Streaming、Spark MLlib等。本书每个章节都安排了入门级的编程实践操作，以便读者更好地学习和掌握Spark编程方法。本书官网免费提供了全套的在线教学资源，包括讲义PPT、习题、源代码、软件、数据集、授课视频、上机实验指南等。

附录I：《Spark编程基础（Python版）》



《Spark编程基础（Python版）》

厦门大学 林子雨，郑海山，赖永炫 编著

披荆斩棘，在大数据丛林中开辟学习捷径
填沟削坎，为快速学习Spark技术铺平道路
深入浅出，有效降低Spark技术学习门槛
资源全面，构建全方位一站式在线服务体系

人民邮电出版社出版发行，ISBN:978-7-115-52439-3
教材官网： http://dblab.xmu.edu.cn/post/spark-python/

本书以Python作为开发Spark应用程序的编程语言，系统介绍了Spark编程的基础知识。全书共8章，内容包括大数据技术概述、Spark的设计与运行原理、Spark环境搭建和使用方法、RDD编程、Spark SQL、Spark Streaming、Structured Streaming、Spark MLlib等。本书每个章节都安排了入门级的编程实践操作，以便读者更好地学习和掌握Spark编程方法。本书官网免费提供了全套的在线教学资源，包括讲义PPT、习题、源代码、软件、数据集、上机实验指南等。

附录J：高校大数据课程公共服务平台

高校大数据课程

公 共 服 务 平 台

http://dblab.xmu.edu.cn/post/bigdata-teaching-platform/

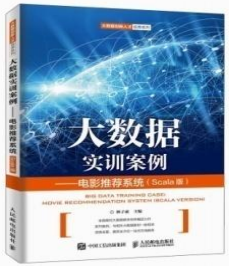
扫一扫访问平台主页

扫一扫观看3分钟FLASH动画宣传片

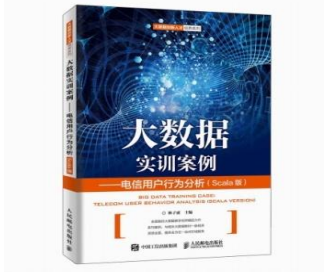
附录K：高校大数据实训课程系列案例教材

为了更好满足高校开设大数据实训课程的教材需求，厦门大学数据库实验室林子雨老师团队联合企业共同开发了《高校大数据实训课程系列案例》，目前已经完成开发的系列案例包括：

《电影推荐系统》（已经于2019年5月出版）
《电信用户行为分析》（已经于2019年5月出版）
《实时日志流处理分析》
《微博用户情感分析》
《互联网广告预测分析》
《网站日志处理分析》



系列案例教材将于2019年陆续出版发行，教材相关信息，敬请关注网页后续更新！
http://dblab.xmu.edu.cn/post/shixunkecheng/



扫一扫访问大数据实训课程系列案例教材主页


Thank You!