《Python程序设计基础教程（微课版）》

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

第8章 模块

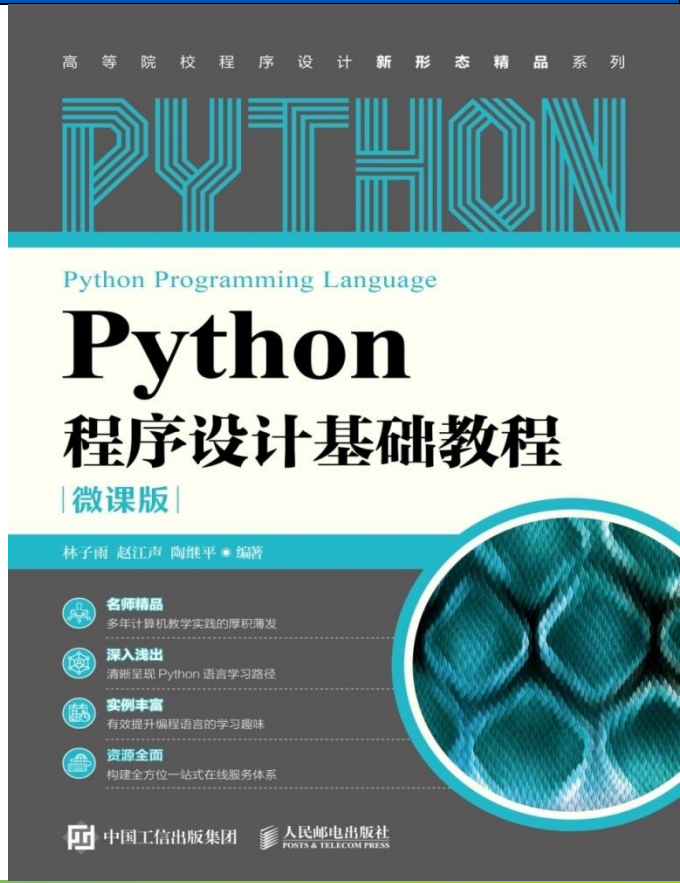
提纲

8.1创建和使用模块
8.2 模块搜索路径
8.3包
8.4 Python自带的标准模块
8.5 使用pip管理Python扩展模块

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



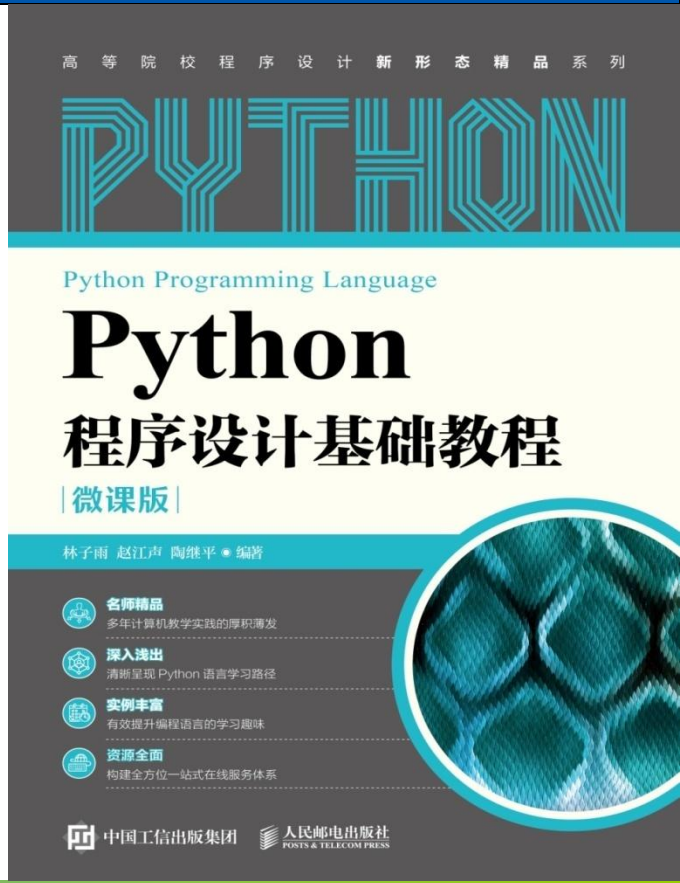
8.1创建和使用模块

8.1.1 创建模块
8.1.2 使用import语句导入模块
8.1.3 使用from…import语句导入模块

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



8.1.1 创建模块

在Python中模块分为以下几种：

（1）系统内置模块。例如sys、time、json模块等等；

（2）自定义模块。自定义模块是用户自己写的模块，对某段逻辑或某些函数进行封装后供其他函数调用。需要注意的是，自定义模块的命名一定不能和系统内置的模块重名，否则将不能再导入系统的内置模块。例如，自定义了一个sys.py模块后，就不能再使用系统的sys模块了；

（3）第三方的开源模块。这部分模块可以通过“pip install”命令进行安装，有开源的代码。

8.1.1 创建模块

下面介绍如何创建自定义模块。新建一个rectangle.py文件，这个文件就可以看作是一个模块，其具体代码如下：

```
01        # rectangle.py
02        def area(length,width):
03            return length * width
04        def perimeter(length,width):
05            return (length + width) * 2
```

8.1.2 使用import语句导入模块

可以在程序中使用import语句导入已经创建的模块，语法格式如下：

```
import modulename [as alias]
```

其中，modulename为模块名称，[as alias]为可选项，可以使用该选项给模块起别名。
例如，在下面的程序中导入了上面定义的模块rectangle，并调用了模块里的area()函数：

```
01        # get_area.py
02        import rectangle
03        print("矩形的面积是：", rectangle.area(4,5))
```

上面程序的执行结果如下：
矩形的面积是：20

8.1.2 使用import语句导入模块

可以看出，在导入模块以后，如果要调用模块里面的变量、函数或者类时，需要在变量名、函数名或者类名前带上模块名作为前缀，比如rectangle.area(4,5)表示调用模块rectangle中的函数area(4,5)。

当一个模块名较长不方便记忆时，在导入模块时，也可以使用as关键字给模块取一个新的名字，实例如下：

```
01      # get_area1.py
02      import rectangle as m
03      print("矩形的面积是：",m.area(4,5))
```

8.1.2 使用import语句导入模块

还可以使用import语句来同时引入多个模块，语法如下：

import module1[, module2[,... moduleN]]

比如，假设已经创建了3个模块文件，分别是rectangle.py、circle.py和diamond.py，当需要同时引入这3个模块时，可以使用如下代码：
import rectangle, circle, diamond

8.1.3 使用from…import语句导入模块

•使用import语句导入模块时，每次在执行一条import语句的时候，都会创建一个新的命名空间，并在该命名空间中执行与.py文件相关的所有语句。
•如果我们不想在每次导入模块时都创建一个新的命名空间，而是将具体的定义导入到当前的命名空间中，这时可以使用from…import语句。
•这种导入方式可以减少程序员需要输入的代码量，因为在这种情况下，在调用模块里的变量、函数时，就不再需要使用模块名作为前缀。

•from…import语句的语法格式如下：

```
from modulename import member
```

其中，modulename表示要导入的模块的名称，member表示要导入的变量、函数或者类等，如果要导入全部定义，可以使用通配符“*”。

8.1.3 使用from…import语句导入模块

```
01      # get_area2.py
02      from rectangle import area,perimeter
03      print("矩形的面积是：",area(4,5))
04      print("矩形的周长是：",perimeter(4,5))
```

上面代码的执行结果如下：
矩形的面积是：20
矩形的周长是：18

可以看到，在使用from…import语句导入模块以后，不再需要使用前缀形式（比如rectangle.area(4,5)）来调用模块里面的函数，而是不加前缀直接调用函数，即直接使用area(4,5)和perimeter(4,5)。

8.1.3 使用from…import语句导入模块

由于上面这个程序导入了模块rectangle中的所有定义，因此，也可以使用通配符“*”，具体如下：

```
01        # get_area3.py
02        from rectangle import *
03        print("矩形的面积是：",area(4,5))
04        print("矩形的周长是：",perimeter(4,5))
```

8.2 模块搜索路径

模块其实就是一个文件，如果要执行文件，首先就需要找到文件的路径，如果模块的文件路径和执行文件不在一个目录下，我们就需要指定模块的路径。模块搜索路径就是指在导入模块时需要检索的目录。一般而言，在导入模块时，查找模块的顺序是：

（1）在当前目录（即执行的Python脚本文件所在目录）下查找；
（2）到环境变量PYTHONPATH下的每个目录中查找；
（3）到Python的默认安装目录下查找。

可以使用如下方式查找这些目录：

```
>>> import sys
>>> print(sys.path)
```

如果要导入的模块不在上述目录中，则需要通过三种方式把指定的目录添加到sys.path中，即函数添加、修改环境变量和增加.pth文件。

8.2 模块搜索路径

（1）函数添加

函数添加的语法格式如下：

```
import sys
sys.path.append("yourpath")
```

其中，yourpath就是要添加到sys.path中的目录。这种方式只是一次性的，只在执行当前文件的窗口中有效，也就是说，以后每次在新窗口中调用该模块时，都需要用这两个命令先把该模块的路径添加进去。

8.2 模块搜索路径

（2）修改环境变量

PYTHONPATH是Python搜索路径，默认情况下我们import的模块都会从PYTHONPATH里面寻找。对于不同的操作系统而言，设置环境变量PYTHONPATH的方法都不尽相同，具体设置方法可以参考相关书籍或网络资料。这里仅以Windows7操作系统为例介绍PYTHONPATH的设置方法。在Windows7操作系统中，在“计算机”图标上单击鼠标右键，在弹出的菜单中选择“属性”，再在出现的界面的左侧选择“高级系统设置”。在弹出的系统属性设置界面中（如图8-1所示），点击“环境变量”按钮。

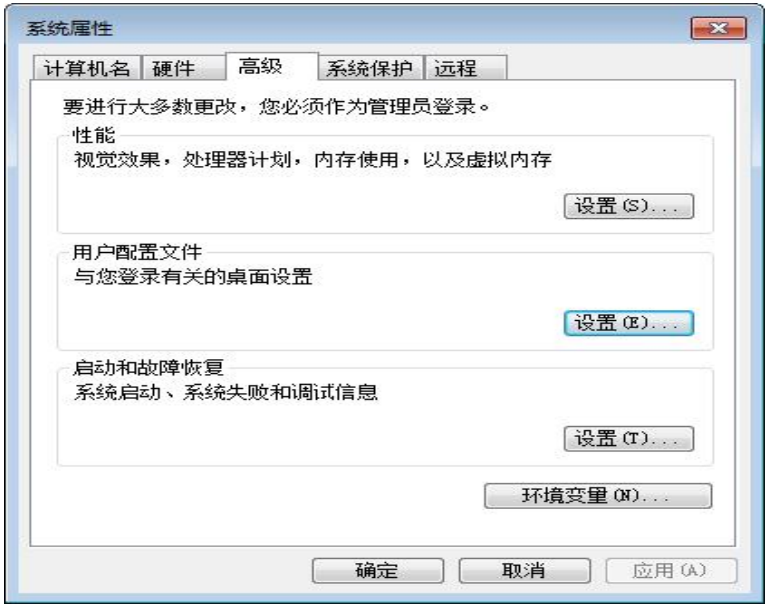


图 8-1 系统属性设置界面

8.2 模块搜索路径

（2）修改环境变量

然后，在弹出的环境变量设置界面中（如图8-2所示），在“系统变量”区域，点击“新建”按钮，在弹出的新建系统变量界面中（如图8-3所示），在“变量名”右侧的文本框中输入“PYTHONPATH”，在“变量值”右侧的文本框中输入模块文件所在的路径，比如“C:\Python38\mycode;”（注意，路径结尾用英文分号结束）。最后，点击“确定”按钮就可以完成设置。

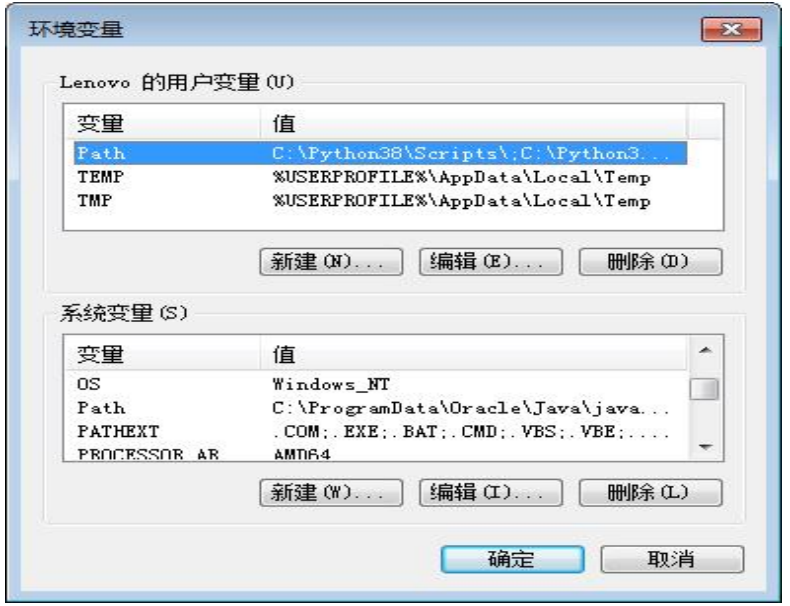


图8-2 环境变量设置界面

图8-3 新建系统变量界面

8.2 模块搜索路径

（3）增加.pth文件

推荐采用本方式。首先，执行如下命令找到site-packages文件路径：

```
>>> import site
>>> site.getsitepackages()
['C:\\Python38', 'C:\\Python38\\lib\\site-packages']
```

然后，在site-packages目录下添加一个路径文件，比如mypkpath.pth（文件名任意，只要后缀名是.pth即可）；最后，在该路径文件里面添加模块文件所在的目录名称。

8.3包

•为了组织好模块，Python会将多个模块分为包。包是一个分层次的文件目录结构，它定义了一个由模块、子包和子包下的子包等组成的 Python的应用环境。

•简单来说，包就是文件夹，但该文件夹下必须存在__init__.py文件，该文件的内容可以为空。__init__.py用于标识当前文件夹是一个包。图8-4给出了一个常见的包结构实例，其中，package_a是包的名称，module_a1.py和module_a2.py是模块的名称。

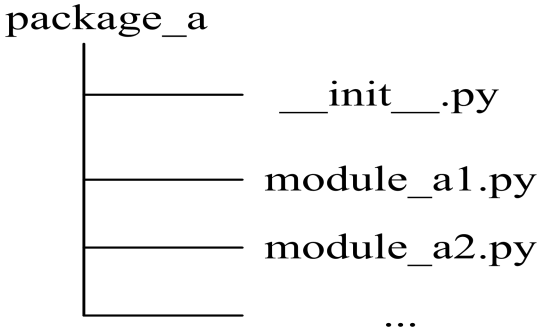


图8-4 一个常见的包结构实例

8.3.1 创建和使用包

这里创建一个名称为package_hello的包，在这个包里面有两个模块，分别是hello1.py和hello2.py。

hello1.py的代码如下：

```
01        def sayhello1():
02            print("Hello World")
```

hello2.py的代码如下：

```
01        def sayhello2():
02            print("Hello China")
```

8.3.1 创建和使用包

在package_hello目录下创建__init__.py文件，文件内容可以为空。
然后在package_hello同级目录下创建hello.py来调用package_hello包，
hello.py的代码如下：

```
01    from package_hello.hello1 import sayhello1
02    from package_hello.hello2 import sayhello2
03    sayhello1()
04    sayhello2()
```

运行hello.py以后的结果如下：
Hello World
Hello China

8.3.1 创建和使用包

在上面的代码中，我们采用了“from 包名.模块名 import 定义名”这种形式来加载模块。实际上，也可以采用“from 包名 import 模块名”这种形式来加载模块，实例如下：

```
01    from package_hello import hello1
02    from package_hello import hello2
03    hello1.sayhello1()
04    hello2.sayhello2()
```

8.3.2 作为主程序运行

•对于很多编程语言来说，程序都必须要有一个入口，比如C、C++、Java和C#等。C和C++都需要有一个main函数作为程序的入口，也就是程序的运行会从main函数开始。同样，Java和C#也必须要有一个包含Main方法的主类，作为程序入口。

•而Python则不同，它属于脚本语言，不像编译型语言那样先将程序编译成二进制再运行，而是动态地逐行解释运行，也就是从脚本第一行开始运行，没有统一的入口。一个Python源码文件（.py）除了可以被直接运行外，还可以作为模块（也就是库），被其他.py文件导入。

8.3.2 作为主程序运行

不管是直接运行还是被导入，.py文件的最顶层代码都会被运行（Python用缩进来区分代码层次）。而当一个.py文件作为模块被导入时，我们有时候可能不希望一部分代码被运行，这时，可以在代码中采用如下语句来达到目的：

```
if __name__ == '__main__':
```

其中，“__name__”是内置变量，可用于表示当前模块的名字。如果一个.py文件（模块）被直接运行时，则其没有包结构，其“__name__”值为“__main__”，即模块名为“__main__”。当.py文件以模块形式被导入时，“__name__”值就不是“__main__”，因此，if __name__ == '__main__'之下的代码块就不会被运行。

8.3.2 作为主程序运行

现在我们对8.3.1节中的实例做适当修改。创建一个名称为package_hello1的包，在这个包里面有两个模块，分别是hello1.py和hello2.py。

hello1.py的代码如下：

```
01        welcome1 = "Hello World"
02        def sayhello1():
03          print(welcome1)
04        if __name__ == '__main__':
05          print("你好，世界")
```

hello2.py的代码如下：

```
01        welcome2 = "Hello China"
02        def sayhello2():
03          print(welcome2)
04        if __name__ == '__main__':
05          print("你好，中国")
```

8.3.2 作为主程序运行

在package_hello1目录下创建__init__.py文件，文件内容可以为空。
然后在package_hello1同级目录下创建hello_1.py来调用package_hello1包，hello_1.py的代码如下：

```
01      from package_hello1 import hello1
02      from package_hello1 import hello2
03      hello1.sayhello1()
04      hello2.sayhello2()
```

单独执行hello1.py的结果如下：
你好，世界
单独执行hello2.py的结果如下：
你好，中国
执行hello_1.py的结果如下：
Hello World
Hello China

8.3.2 作为主程序运行

•可以看出，当直接执行hello1.py时，“__name__”的值是“__main__”，if __name__ == '__main__'后面的语句块会被执行，因此，会输出“你好，世界”。

•当执行hello_1.py时，hello1.py会被导入到hello_1.py中运行，这时“__name__”的值不等于“__main__”，因此，if __name__ == '__main__'后面的语句块不会被执行，也就不会输出“你好，世界”。

8.4 Python自带的标准模块

Python自带了很多实用的模块，称为“标准模块”或者“标准库”，我们可以直接使用import语句把这些模块导入到Python文件中使用。表8-1给出了Python常用的内置标准模块。

表8-1 Python常用的内置标准模块

模块名	功能
calendar	提供与日期相关的各种函数的标准库
datetime	提供与日期、时间相关的各种函数的标准库
decimal	用于定点和浮点运算
json	用于使用JSON序列化和反序列化对象
logging	提供了配置日志信息的功能
math	提供了许多对浮点数的数学运算函数
os	提供了对文件和目录进行操作的标准库
random	提供随机数功能的标准库
re	提供了基于正则表达式的字符串匹配功能
sys	提供对解释器使用或维护的一些变量的访问以及与解释器交互的函数
shutil	高级的文件、文件夹、压缩包处理模块
time	提供与时间相关的各种函数的标准库
urllib	请求URL连接的标准库

8.5 使用pip管理Python扩展模块

•Python的强大之处在于它拥有非常丰富的第三方模块（或第三方库），可以帮我们方便、快捷地实现网络爬虫、数据清洗、数据可视化和科学计算等功能。

•为了便于用于安装和管理第三方库和软件，Python提供了一个扩展模块（或扩展库）管理工具pip，Python3.8.7在安装的时候会默认安装pip。

8.5 使用pip管理Python扩展模块

pip之所以能够成为最流行的扩展模块管理工具，并不是因为它被Python官方作为默认的扩展模块管理器，而是因为它自身有很多优点，主要包括：

•pip提供了丰富的功能，包括扩展模块的安装和卸载，以及显示已经安装的扩展模块；

•pip能够很好地支持虚拟环境；

•pip可以集中管理依赖；

•pip能够处理二进制格式；

•pip是先下载后安装，如果安装失败，也会清理干净，不会留下一个中间状态。

8.5 使用pip管理Python扩展模块

pip提供的命令不多，但是都很实用。表8-2给出了常用pip命令的使用方法。

表8-2 常用pip命令的使用方法

pip命令	说明
pip install SomePackage	安装SomePackage模块
pip list	列出当前已经安装的所有模块
pip install --upgrade SomePackage	升级SomePackage模块
pip uninstall SomePackage	卸载SomePackage模块

例如，Matplotlib是最著名的Python绘图库，它提供了一整套和Matlab相似的API，十分适合交互式地进行制图，可以使用如下命令安装Matplotlib：

```
> pip install matplotlib
```

安装成功以后，使用如下命令就可以看到安装的Matplotlib：

```
> pip list
```

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：

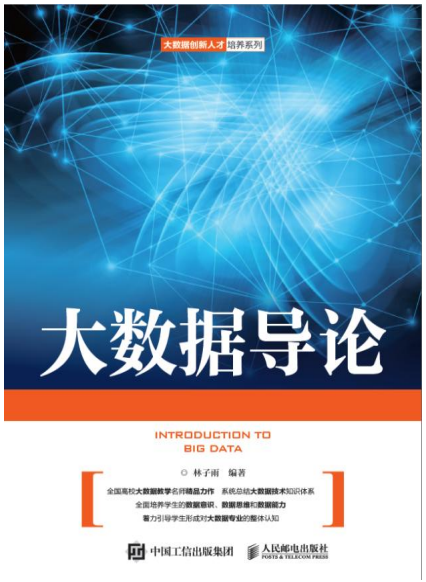
- 引导学生步入大数据时代，积极投身大数据的变革浪潮之中
- 了解大数据概念，培养大数据思维，养成数据安全意识
- 认识大数据伦理，努力使自己的行为符合大数据伦理规范要求
- 熟悉大数据应用，探寻大数据与自己专业的应用结合点
- 激发学生基于大数据的创新创业热情

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

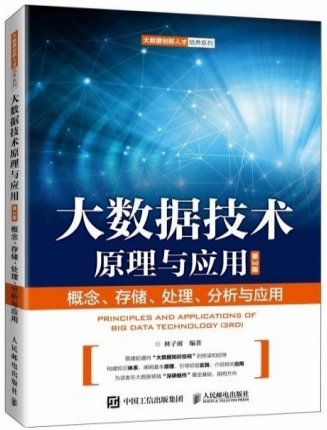
附录F：《大数据技术原理与应用（第3版）》教材

《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

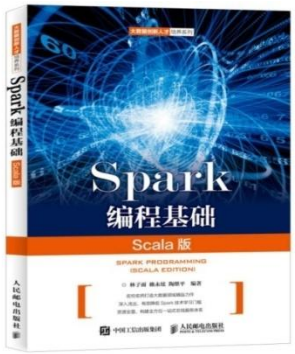
厦门大学林子雨编著

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

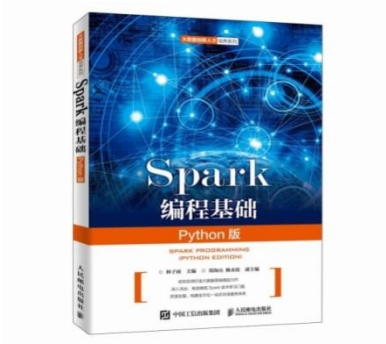
披荆斩棘，在大数据丛林中开辟学习捷径
填沟削坎，为快速学习Spark技术铺平道路
深入浅出，有效降低Spark技术学习门槛
资源全面，构建全方位一站式在线服务体系

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/

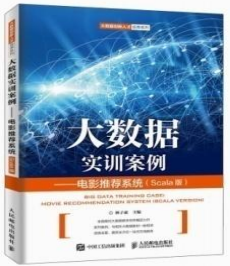
扫一扫访问平台主页

扫一扫观看3分钟FLASH动画宣传片

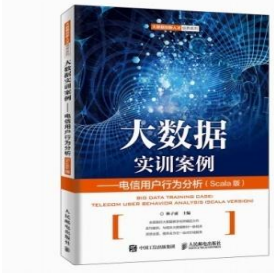
附录K：高校大数据实训课程系列案例教材

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

Department of Computer Science, Xiamen University, 2022