计算机软件与新技术人才培养系列教材

软件技术

数据结构与算法
C语言程序设计
C#程序设计
Python程序设计实践教程
Java程序设计
Java Web应用开发
Java框架开发技术
Web前端开发
Web应用系统开发
HTML5与JavaScript程序设计
HTML5+CSS3网页设计技术
Android应用开发
iOS应用开发

大数据技术

Linux操作系统基础
Spark技术与应用
Python数据分析
Python数据工程师实战案例教程
大数据基础
数据库高级管理技术
大数据集群搭建维护与数据存储
大数据采集与数据处理
大数据可视化技术应用
数据挖掘应用
智能数据分析与应用

云计算技术

云计算基础
OpenStack系统架构与部署
云存储规划与部署
网络存储规划与实施
云安全技术与应用
服务器虚拟化技术项目式教程
虚拟化技术与应用
云数据中心架构与运维
公有云运维与应用实践
云平台建设与维护实战

人工智能技术

人工智能基础
机器学习
深度学习
人工智能算法设计
人工神经网络技术
智能视觉技术
智能算法容器化部署
自然语言处理技术
智能语音应用开发

读科学 得真知
科学普及出版社漫漫读微信号

ISBN 978-7-5236-1275-0

定价：45.00元

计算机软件与新技术人才培养系列教材

Python数据工程师实战案例教程

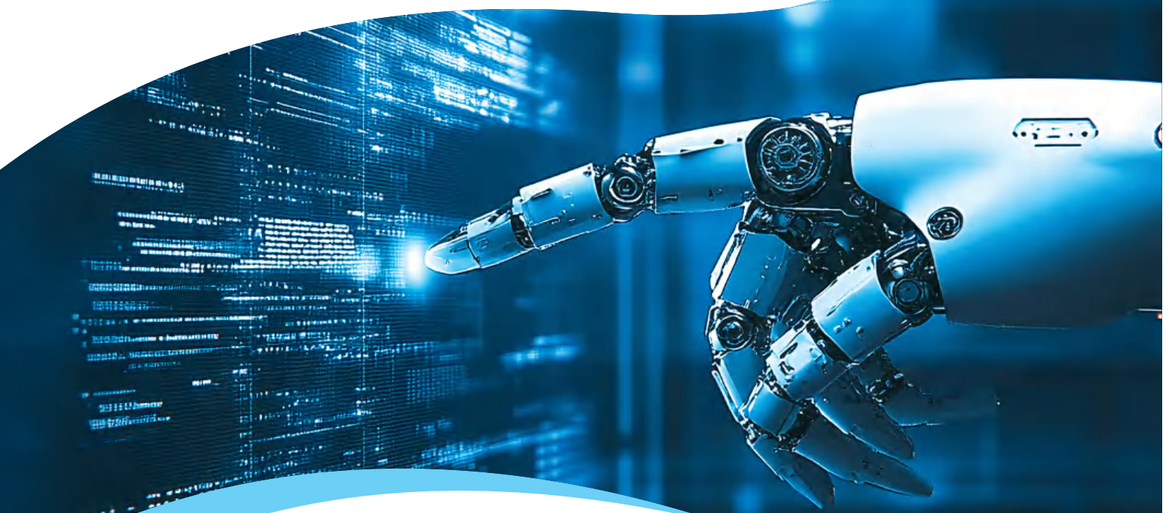
王克朝 张胜扬／主编

中国科学技术出版社

计算机软件与新技术人才培养系列教材

Python
数据工程师实战案例教程

王克朝　张胜扬／主编

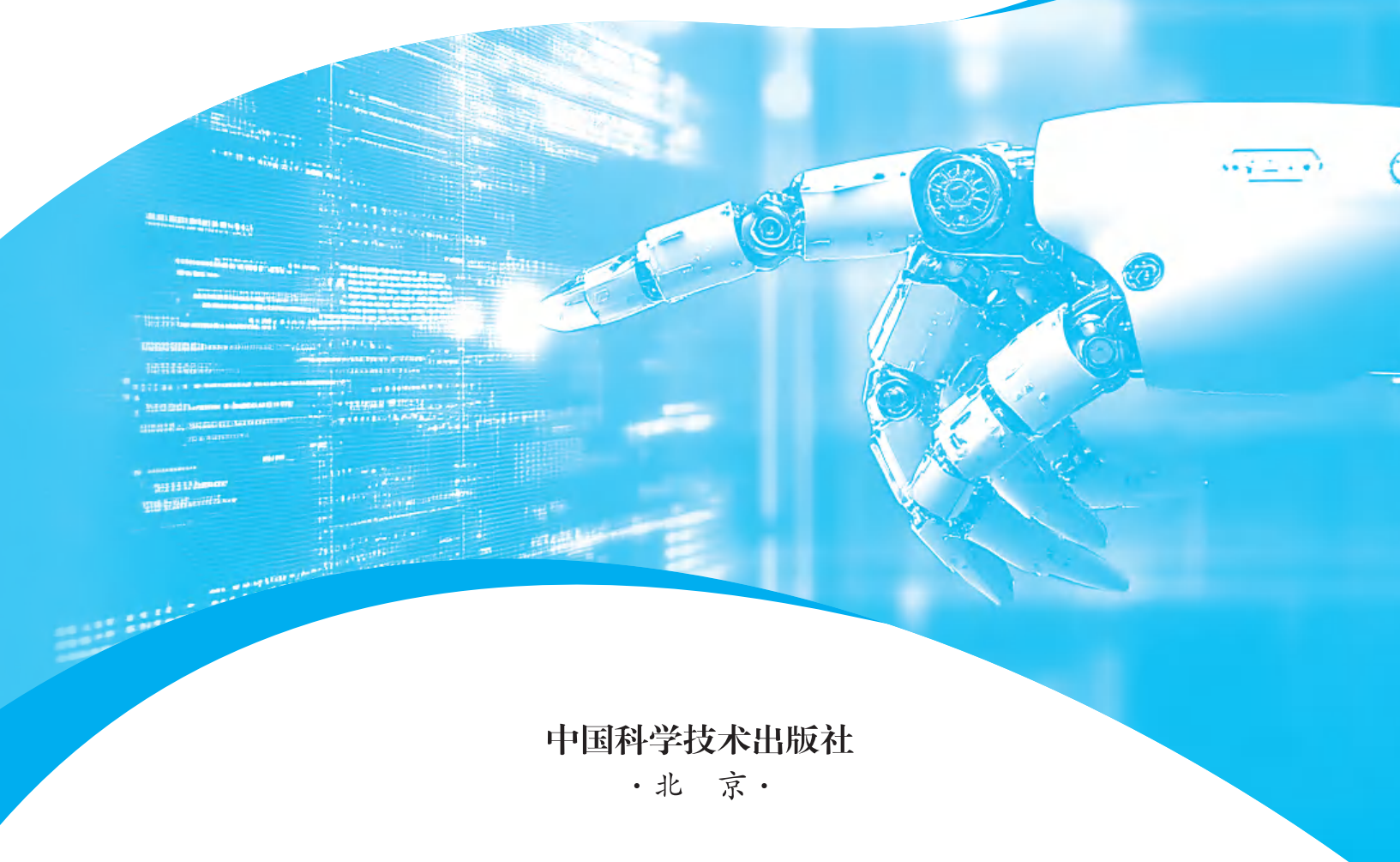


中国科学技术出版社
CHINA SCIENCE AND TECHNOLOGY PRESS

计算机软件与新技术人才培养系列教材

Python
数据工程师实战案例教程

王克朝　张胜扬／主编



中国科学技术出版社
·北　京·

图书在版编目（CIP）数据

Python 数据工程师实战案例教程 / 王克朝，张胜扬主编．-- 北京 ：中国科学技术出版社，2025．1．
（计算机软件与新技术人才培养系列教材）．-- ISBN 978-7-5236-1275-0

Ⅰ．TP312.8

中国国家版本馆 CIP 数据核字第 2025C4R542 号

策划编辑	王晓义
责任编辑	徐君慧
装帧设计	世纪宏图
责任校对	焦　宁
责任印制	徐　飞

出　　版	中国科学技术出版社
发　　行	中国科学技术出版社有限公司
地　　址	北京市海淀区中关村南大街 16 号
邮　　编	100081
发行电话	010-62173865
传　　真	010-62173081
网　　址	http://www.cspbooks.com.cn

开　　本	889mm × 1194mm　1/16
字　　数	392 千字
印　　张	12.5
版　　次	2025 年 1 月第 1 版
印　　次	2025 年 1 月第 1 次印刷
印　　刷	北京荣玉印刷有限公司
书　　号	ISBN 978-7-5236-1275-0 / TP · 512
定　　价	45.00 元

编写委员会

主　编　王克朝　张胜扬

副主编　王　卓　李禹齐

前　言

在数字化浪潮汹涌澎湃的今天，数据已成为推动社会进步、助力企业腾飞的核心引擎。作为数字时代的关键生产要素，数据凭借其可复制、可共享的特性，正成为新质生产力发展的核心要素。数据工程师作为数据领域的先锋队，不仅要有坚实的理论基础，更要有丰富的实践经验和解决现实问题的能力。正是基于这样的认识，我们精心编写了这本《Python 数据工程师实战案例教程》。

本书的核心目标在于帮助学生系统地掌握数据工程师所需的核心技能，并通过一系列实战案例，将理论知识与实际应用深度融合，使学生能够在实践中学习、在挑战中成长。我们坚信，实践是检验真理的唯一标准，也是培养学生实际操作能力和问题解决能力的关键。同时，本书响应新时期国家对教育的新要求，落实立德树人根本任务，贯彻《高等学校课程思政建设指导纲要》和党的二十大精神，将专业知识与思政教育有机结合，实现价值引领、知识传授和能力培养紧密结合。

本书具有如下特点。

1. 实战引领，知行合一。本教材不仅介绍了数据工程师所需的基本理论和技能，更通过精心设计的实战案例，让学生能够亲手操作、亲身体验数据处理的每一个步骤。这些案例贴近实际，具有高度的可操作性和实用性，旨在引导学生将理论知识转化为实际操作能力，实现知行合一。

2. 注重实用性，强调应用。我们选取的案例均来源于实际项目或企业需求，旨在让学生在学习过程中了解数据工程师的实际工作内容及要求。这些实践案例使学生能够直观地感受到数据工程师职业的挑战与机遇，为他们未来的职业发展奠定基础。

3. 系统构建，全面覆盖。本教材从数据工程师的工作流程和技能体系出发，系统地介绍了数据采集、数据处理、数据分析、数据可视化等方面的知识和技能，同时通过全面的知识体系构建，使学生能够系统地掌握数据工程师所需的核心技能，为他们未来的职业发展提供支持。

4. 鼓励创新，激发潜能。在数据领域，新的技术和方法层出不穷。本教材在介绍经典理论和方法的同时，也密切关注最新的技术和趋势，鼓励学生在掌握基本技能的基础上，积极探索新的方法和思路。我们坚信，创新是推动数据科学领域发展的关键动力，也是培养学生核心竞争力的重要途径。

本书由王克朝、张胜扬担任主编，王卓、李禹齐担任副主编。具体编写分工如下：第 1、5、7 单元由王克朝编写，第 2、3 单元由王卓编写，第 4 单元由李禹齐编写，第 6 单元由张胜扬编写。汪曦翔负责本书代码的验证和文字的校对工作，王克朝和张胜扬负责最后的统稿。本书配备的课件、源码等教学资源可发邮件至 2393867076@qq.com 领取。

本书在编写过程中参考了部分优秀的文献，在此向有关作者表示衷心的感谢。同时，衷心感谢所有为本书撰写和出版付出辛勤努力的同人们，也感谢广大读者对本书的支持和关注。由于计算机技术飞速发展，书中不足之处恳请读者批评指正。

最后，我们衷心希望这本书能够成为学生学习数据工程师技能的宝贵资源，帮助他们通过实践锻炼成长为具备扎实理论基础和丰富实战经验的数据工程师。同时，我们也期待与广大师生和业界专家进行深入的交流和合作，共同推动数据科学领域的发展和进步。

目　录

单元 4　数据存储

单元 5　NumPy 基础

单元 6 Pandas 数据处理

单元

5

NumPy基础

单元导读

NumPy 是一个开源的 Python 科学计算库，是 Python 科学计算库的基础库，许多其他著名的科学计算库如 Pandas、Scikit-learn 等都要用到 NumPy 库的一些功能。它是在 1995 年诞生的 Python 库 Numeric 的基础上建立起来的，但真正促使 NumPy 发行的是 Python 的 Scipy 库。但 Scipy 中并没有合适的类似于 Numeric 中的对于基础数据对象进行处理的功能，于是，Scipy 的开发者将 Scipy 中的一部分和 Numeric 的设计思想结合，在 2005 年发行了 NumPy。它包含很多功能，如创建 n 维数组（矩阵）、对数组进行函数运算、数值积分等。

微课 NumPy

单元目标

知识目标

（1）了解 NumPy 的标准数据类型和 NumPy 数组对象以及一些基础知识，包括创建数组、数组基础和数组属性等内容。

（2）便捷地使用 NumPy，包括如何选取数组的某一部分（例如根据一组布尔值来选取）以及操纵 NumPy 对象的形态。

（3）通过 NumPy 中函数的一些基本模块解决数学运算，如加、减、乘、除等。

（4）运用 NumPy 的广播功能解决实际问题。

能力目标

（1）能够运用 Numpy 库解决实际问题，如数据分析、信号处理、图像处理等领域中的数值计算问题。

（2）能够根据问题的需求，选择合适的 Numpy 函数和方法，高效地实现数据处理和计算任务。

（3）能够分析和解决在使用 Numpy 库过程中遇到的错误和异常，提高程序的稳定性。

素质目标

（1）培养严谨的科学思维，通过学习和实践 Numpy 库，理解并掌握数值计算的基本原理和方法。

（2）培养团队合作精神和沟通能力，通过参与团队项目和讨论，学会与他人合作解决问题并分享经验。

（3）培养自我学习和持续学习的能力，通过查阅官方文档和参考书籍，不断更新和扩展自己的知识和技能。

5.1 安装 NumPy

NumPy 是基于 Python 的，因此在安装 NumPy 之前，需要先安装 Python。某些操作系统已经默认安装有 Python 环境，但仍需检查 Python 的版本是否与将要安装的 NumPy 版本兼容。Python 有很多种实现，包括一些商业化的实现和发行版。本小节将阐述 NumPy 的安装方法，在安装之前需要确保已有的环境准备为 Python3.6、PyCharm。安装 NumPy，打开 PyCharm，在 Terminal（终端）中输入以下代码：pip install NumPy。

最后，检查是否安装完成，创建 Python 文件，输入以下代码：

```
import numpy
# 检查 NumPy 版本
print(numpy.__version__)
```

运行后得到如图 5-1 所示结果，表示安装成功。

```
C:\Users\Administrator\PycharmProjects\Numpy\venv\Scripts\python.exe
1.26.4

Process finished with exit code 0
```

图 5-1　验证安装是否完成

5.2　NumPy 数组

NumPy 数组通常是由相同种类的元素组成的，即数组中的数据项的类型一致。这样有一个好处，由于数组元素的类型相同，所以能快速确定存储数据所需空间的大小。同时，NumPy 数组能够运用向量化运算来处理整个数组，速度较快；而 Python 的列表则通常需要借助循环语句遍历列表，运行效率相对来说较差。

5.2.1　理解 Python 中的数据类型

本节将学习 Python 中的简单数据类型，这些数据类型都在 Python 的信息处理过程中发挥着重要作用。这里将数据类型分为不可变数据类型和可变数据类型。

不可变数据类型的定义：当该数据类型的对应变量的值发生了改变，那么它对应的内存地址也会发生改变，就称不可变数据类型，具体有 4 个：数字（Number）、字符串（String）、布尔值（Bool）、元组（Tuple）。

可变数据类型的定义：当该数据类型的对应变量的值发生了改变，那么它对应的内存地址不发生改变，就称可变数据类型，具体有 3 个：列表（List）、集合（Set）、字典（Dictionary）。

1. Number

包括整形（int）、浮点型（float）和复数类型（complex）。

整型：用于表示没有小数部分的整数。在 Python 中可以使用 4 种进制表示整型，分别为二进制、八进制、十进制和十六进制。

浮点型：用于保存带有小数点的数值，Python 的浮点数一般以十进制形式表示，对于较大或较小的浮点数，可以使用科学计数法表示。

复数：由实部和虚部构成，其一般形式为 real+imagj，实部 real 和虚部的 imag 都是浮点型，其中虚部必须有虚数单位 j 或 J。

表 5-1 为数值类型实例。

表 5-1　数值类型实例

int	float	complex
10	0.0	3.14j
100	15.20	45.j
-786	-21.9	9.322e-36j
080	32.3e+18	.876j
-0490	-90.	-.6545+0J

续表

int	float	complex
-0x260	-32.54e100	3e+26J
0x69	70.2E-12	4.53e-7j

2. String

字符串是一种用来表示文本的数据类型，它是由符号或者数值组成的一个连续序列。Python 中的字符串用单引号或双引号括起来，同时使用反斜杠表示转义特殊字符。字符串的截取的语法格式如下：

```
变量 [ 头下标 : 尾下标 ]
```

其中，反斜杠可以用来转义，使用 r 可以让反斜杠不发生转义；字符串可以用 + 运算符连接在一起，用 * 运算符重复；Python 中的字符串有两种索引方式，从左往右以 0 开始，从右往左以 -1 开始；Python 中的字符串不能改变。

【例 5-1】字符串的应用实例。

```
str='China'
# 输出第一个到倒数第二个的所有字符
print(str[0:-1])
```

程序运行结果如下：

```
Chin
```

```
# 输出第三个开始到第四个的字符
print(str[2:4])
```

程序运行结果如下：

```
in
```

```
# 输出从第三个开始的后面所有字符
print(str[2:])
```

程序运行结果如下：

```
ina
```

```
# 连接字符
print(str+" 中国 ")
```

程序运行结果如下：

```
China 中国
```

3. Bool

Python 中的布尔类型有两种取值：True 和 False。True 和 False 都是关键字，表示布尔值。布尔类型可以用来控制程序的流程，比如判断某个条件是否成立，或者在某个条件满足时执行某段代码。

布尔类型特点：只有 True 和 False 两个值；可以和其他数据类型进行比较，比如数字、字符串等，在比较时，Python 会将 True 视为 1，False 视为 0；可以和逻辑运算符一起使用，包括 and、or 和 not，这些运算符可以用来组合多个布尔表达式，生成一个新的布尔值；布尔类型也可以被转换成其他数据类型，比如整数、浮点数和字符串，在转换时，True 会被转换成 1，False 会被转换成 0。

【例 5-2】bool 类型的应用实例。

```
a=True
b=False
```

```
# 比较运算符
print(2<3)
```

程序运行结果如下：

```
True
print(2==3)
```

程序运行结果如下：

```
False
```

4. Tuple

元组与列表类似，不同之处在于元组的元素不能修改。元组写在小括号里，元素之间用逗号隔开。同时，元组与字符串类似，可以被索引且下标索引从 0 开始，−1 为从末尾开始的位置。也可以进行截取。

其中，与字符串一样，元组的元素不能修改；元组也可以被索引和切片，方法一样；构造包含 0 或 1 个元素的元组的特殊语法规则；元组也可以使用 + 操作符进行拼接。

【例 5−3】元组的应用实例。

```
tup=(1, 9, 4, 9, 1, 9, 7, 8)
print(tup[0])
```

程序运行结果如下：

```
1
print(tup[0:4])
```

程序运行结果如下：

```
(1, 9, 4, 9)
# 修改元组元素的操作是非法的
tup[0]=11
```

程序运行结果如下：

```
Traceback(most recent call last):
  File"test.py",line1,in<module>
    tup[0]=11
TypeError:'tuple'object does not support item assignment
```

5. List

列表是 Python 中使用最频繁的数据类型，它可以完成大多数集合类的数据结构实现。列表中元素的类型可以不相同，支持数字、字符串甚至可以包含列表（所谓嵌套）。同时，列表是写在方括号之间、用逗号分隔开的元素列表。和字符串一样，列表同样可以被索引和截取，列表被截取后返回一个包含所需元素的新列表。

【例 5−4】列表的应用实例。

```
list=['China','teenage',' 不忘初心 ',' 砥砺奋进 ']
# 输出完整列表
print(list)
```

程序运行结果如下：

```
['China','teenage',' 不忘初心 ',' 砥砺奋进 ']
# 输出列表第一个元素
```

```
print(list[0])
```

程序运行结果如下：

```
China
```

```
# 从第二个开始输出到第三个元素
print(list[1:3])
```

程序运行结果如下：

```
['teenage',' 不忘初心 ']
```

```
# 输出从第三个元素开始的所有元素
print(list[2:])
```

程序运行结果如下：

```
[' 不忘初心 ',' 砥砺奋进 ']
```

6. Set

集合是由一个或数个形态各异的大小整体组成的，构成集合的事物或对象称作元素或是成员，基本功能是进行成员关系测试和删除重复元素。可以使用大括号或者 set() 函数创建集合。

其中，创建一个空集合必须用 set() 而不是大括号，因为大括号是用来创建一个空字典的。创建格式为：parame={value01,value02, ...} 或者 set(value)。

【例 5-5】集合的应用实例。

```
sites={'c','h','i','n','a','a'}
# 输出集合，重复的元素被自动去掉
print(sites)
```

程序运行结果如下：

```
{'c','h','i','n','a'}
```

```
# 成员测试
if 'c' in sites:
        print('c 在集合中 ')
   else:
        print('c 不在集合中 ')
```

程序运行结果如下：

```
c 在集合中
```

7. Dictionary

字典是 Python 中另一个非常有用的内置数据类型。其不同于列表，列表是有序的对象集合，而字典是无序的对象集合。两者之间的区别在于字典当中的元素是通过键来存取的，而不是通过偏移存取。字典是一种映射类型，用 {} 标识，它是一个无序的“键（key）：值（value）”的集合，且键（key）必须使用不可变类型。而在同一个字典中，键（key）必须是唯一的。

【例 5-6】字典的应用实例。

```
dict={}
dict['one']="China- 中国 "
dict[2]="student- 学生 "
tinydict={'name':'China','code':1}
# 输出键位 'one' 的值
```

```
print(dict['one'])
```

程序运行结果如下：

```
China- 中国
```

执行如下代码：

```
# 输出键为 2 的值
print(dict[2])
```

程序运行结果如下：

```
student- 学生
```

执行如下代码：

```
# 输出完整的字典
print(tinydict)
```

程序运行结果如下：

```
{'name': 'China', 'code': 1}
```

执行如下代码：

```
# 输出所有键
print(tinydict.keys())
```

程序运行结果如下：

```
dict_keys(['name', 'code'])
```

执行如下代码：

```
# 输出所有值
print(tinydict.values())
```

程序运行结果如下：

```
dict_values(['China', 1])
```

5.2.2　创建数组

在 Python 中，数组是一种有序的、可变的数据结构，可以存储不同类型的元素。然而，有时候需要在创建数组时指定它的类型，以便限制数组中元素的类型，提高程序的效率和可读性。有 5 种创建数组的通用机制。

（1）从其他 Python 结构（例如列表、元组）转换。

【例 5-7】列表转成数组。截取 2019 年至 2022 年科技活动的基本情况，以发表科技论文、出版科技著作和专利申请数为例。

```
import pandas as pd
import numpy as np
list1=[' 指标 ','2022','2021','2020','2019']
list2=[' 发表科技论文 ( 万篇 )', 215.00, 203.00, 195.17, 195.00]
list3=[' 出版科技著作 ( 种 )', 60930, 50580, 49634, 52067]
list4=[' 专利申请数 ( 项 )', 5364639, 5243592, 5194154, 4380468]
df1=pd.DataFrame(list1)
df2=pd.DataFrame(list2)
df3=pd.DataFrame(list3)
```

```
df4=pd.DataFrame(list4)
data=pd.concat([df1,df2,df3,df4],axis=1)
data.columns=[1, 2, 3, 4]
data=data.T
```

（2）内在的 NumPy 数组创建对象（例如 arange，1，0 等）。

（3）从磁盘读取标准或自定义格式的数组。

（4）通过使用字符串或缓冲区从原始字节创建数组。

（5）使用特殊的库函数创建。

5.2.3 从头创建数组

零个（形状）将创建一个数组，其中填充了具有指定形状的 0 个值。比如，zeros 和 ones 分别可以创建指定长度或形状的全 0 或全 1 数组。empty 可以创建一个没有任何具体值的数组。要用这些方法创建多维数组，只需传入一个表示形状的元组即可。其中，默认 dtype 是 float64。

【例 5-8】从头创建数组的应用实例。

```
import numpy as np
a=np.zeros((2, 3))
print(a)
```

程序运行结果如下：

```
[[0.0.0.]
[0.0.0.]]
```

5.2.4 NumPy 标准数据类型

dtype（数据类型）是一个特殊的对象，它含有数组将一块内存解释为特定数据类型所需要的信息。其中，NumPy 支持的数据类型比 Python 内置的类型要多很多，基本上可以和 C 语言的数据类型对应上，部分类型对应为 Python 内置的类型。

表 5-2 列举了常用 NumPy 基本类型。

表 5-2 NumPy 基本类型

名称	描述
bool_	布尔型数据类型（True 或者 False）
int_	默认的整数类型（类似于 C 语言中的 long，int32 或 int64）
intc	与 C 语言的 int 类型一样，一般是 int32 或 int 64
intp	用于索引的整数类型（类似于 C 语言的 ssize_t，一般情况下仍然是 int32 或 int64）
int8	字节（-128 ~ 127）
int16	整数（-32768 ~ 32767）
int32	整数（-2147483648 ~ 2147483647）
int64	整数（-9223372036854775808 ~ 9223372036854775807）
uint8	无符号整数（0 ~ 255）
uint16	无符号整数（0 ~ 65535）

续表

名称	描述
uint32	无符号整数（0 ~ 4294967295）
uint64	无符号整数（0 ~ 18446744073709551615）
float_	float64 类型的简写
float16	半精度浮点数，包括 1 个符号位，5 个指数位，10 个尾数位
float32	单精度浮点数，包括 1 个符号位，8 个指数位，23 个尾数位
float64	双精度浮点数，包括 1 个符号位，11 个指数位，52 个尾数位
complex_	complex128 类型的简写，即 128 位复数
complex64	复数，表示双 32 位浮点数（实数部分和虚数部分）
complex128	复数，表示双 64 位浮点数（实数部分和虚数部分）

5.2.5　NumPy 数组基础

NumPy 代表数值 Python，是一个用于处理数组的 Python 库。在 Python 中，使用列表来创建数组，但处理起来很慢。NumPy 数组是一个强大的 n 维数组对象，在线性代数、傅里叶变换和随机数方面有应用。它提供的数组对象比传统的 Python 列表快得多。

【例 5-9】一维数组。

```
import numpy as np
# 创建列表
list=[1, 2, 3, 4]
# 创建数组
sample_array=np.array(list)
print("List in python:",list)
print("NumPy Array in python:",sample_array)
```

程序运行结果如下：

```
List in python :[1, 2, 3, 4]
NumPy Array in python:[1 2 3 4]
```

多维数组中的数据以多维形式存储，如图 5-2 所示为多维数组数据存储形式。

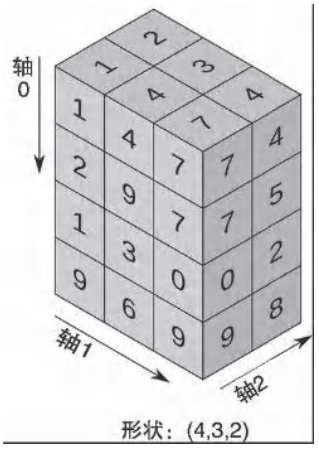


图 5-2　多维数组数据存储形式

【例 5-10】多维数组数据存储。

```
import numpy as np
# 创建列表
list_1=[1, 2, 3, 4]
list_2=[5, 6, 7, 8]
list_3=[9, 10, 11, 12]
# 创建数组
sample_array=np.array([list_1,list_2,list_3])
print("NumPy multi dimensional array in python\n",sample_array)
```

程序运行结果如下：

```
NumPy multi dimensional array in python
 [[ 1 2 3 4]
 [ 5 6 7 8]
 [ 9 10 11 12]]
```

5.2.6 NumPy 数组的属性

NumPy 数组的维数称为秩（rank）。秩就是轴的数量，即数组的维度，一维数组的秩为 1，二维数组的秩为 2，以此类推。在 NumPy 中，每一个线性的数组称为是一个轴（axis），也就是维度（dimension）。比如说，二维数组相当于是两个一维数组，其中第一个一维数组中每个元素又是一个一维数组。所以一维数组就是 NumPy 中的轴，第一个轴相当于是底层数组，第二个轴是底层数组里的数组。很多时候可以声明 axis。例如 axis=0，表示沿着第 0 轴进行操作，即对每一列进行操作；axis=1，表示沿着第 1 轴进行操作，即对每一行进行操作。

NumPy 的数组中比较重要的 ndarray 对象属性如表 5-3 所示。

表 5-3　NumPy 的数组中 ndarray 对象属性

属性	说明
ndarray.ndim	秩，即轴的数量或维度的数量
ndarray.shape	数组的维度，对于矩阵，n[①] 行 m 列
ndarray.size	数组元素的总个数，相当于 .shape 中 n*m 的值
ndarray.dtype	ndarray 对象的元素类型
ndarray.itemsize	ndarray 对象中每个元素的大小，以字节为单位
ndarray.flags	ndarray 对象的内存信息
ndarray.real	ndarray 元素的实部
ndarray.imag	ndarray 元素的虚部
ndarray.data	包含实际数组元素的缓冲区，由于一般通过数组的索引获取元素，所以通常不需要使用这个属性

5.2.7 数组索引

NumPy 数组索引是一个大话题，有很多种方式可以选中数据中的子集或者某个元素，主要有基础

① 为与代码显示格式一致，本书变量统一使用正体。

索引、切片索引、布尔索引和数组索引四种方式，如表 5-4 所示。

表 5-4　数组索引

索引方式	使用场景	视图或副本
基础索引	获取单个元素	视图
切片索引	获取子数组	视图
布尔索引	根据比较操作，获取数组元素	副本
数组索引	传递索引数组，更加快速、灵活地获取子数据集	副本

（1）基础索引：在一维数组中，可以使用中括号指定索引获取第 i 个值（从 0 开始计数），但是对于多维的数组，每个索引值对应的元素不再是一个数值。例如，在一个二维数组中，将得到一个一维数组。单个元素可以继续索引，或者传递一个索引的列表选择单个元素。

【例 5-11】基础索引的运行实例如下。

```
import numpy as np
arr=np.arange(9).reshape((3, 3))
print(arr[0])
```

程序运行结果如下：

```
[0 1 2]
```

运行如下代码：

```
print(arr[0][0])
```

程序运行结果如下：

```
0
```

运行如下代码：

```
print(arr[0, 0])
```

程序运行结果如下：

```
0
```

使用这种基础索引方式获取的数据为原数组的视图，修改视图数值，也会修改原数组数值。

（2）切片索引：NumPy 切片与 Python 列表的标准切片语法相同，通过切片可以灵活地得到一个子数组。与基础索引一样，切片索引也是原数组的视图，修改数据也会作用于原数组。

（3）布尔索引：布尔索引主要是利用的布尔数组，数组的比较操作（比如 ==、>=）也是可以向量化的，这些比较运算的结果是一个布尔数据类型的数组。该数组与原数组的长度和形状都是一致的，可以获取到对应位置为 True 的值。

（4）数组索引：数组索引在概念上很简单，通过一个数组来一次性获取多个元素。

注意：使用数组索引，结果的形状不是与被索引的数组的形状一致；操作重复的索引可以会产生一些出乎意料的结果。

5.2.8　数组的变形

NumPy 中的数组变形有 reshape() 和 resize() 两种方法，它们之间的区别是：reshape() 不改变原始数组，只返回改变后的数组；而 resize() 方法不返回任何东西，直接改变原始数组。代码格式如下所示：

```
numpy.reshape（a,newshape, order='C'）
```

其中 a 表示 array_like 要重新形成的数组。newshape 表示 int 或 tuple 的整数。新的形状应该与原始形状兼容。如果是整数，则结果将是该长度的 1−D 数组。一个形状维度可以是 −1。在这种情况下，从数组的长度和其余维度推断该值。order 表示的是数组排序的方向不同，有 {'C'，'F'，'A'} 形式。使用此索引顺序读取 a 的元素，并使用此索引顺序将元素放置到重新形成的数组中。其中 'C' 意味着使用 C 样索引顺序读取 / 写入元素，最后一个轴索引变化最快，回到第一个轴索引变化最慢。'F' 意味着使用 Fortran 样索引顺序读取 / 写入元素，第一个索引变化最快，最后一个索引变化最慢。

注意：'C' 和 'F' 选项不考虑底层数组的内存布局，而只是参考索引的顺序。

'A' 意味着在 Fortran 类索引顺序中读 / 写元素，如果 a 是 Fortran 在内存中连续的，如果不是则为 C 样顺序。

返回一个具有指定形状的新数组。如果新数组比原始数组大，则新数组将填充 a 的重复副本。代码格式如下所示：

```
numpy.resize（a,new_shape）
```

其中参数 a 为 array_like 要调整大小的数组。new_shape 为 int 或 int 的元组调整大小数组的形状。返回值 reshaped_array 为 ndarray。

新数组由旧数组中的数据组成，必要时重复以填充所需数量的元素。数据在 C−order 中的数组上重复迭代。

注意：当数组的总大小不变时，应该使用 reshape。

在大多数其他情况下，索引（以减小大小）或填充（以增加大小）可能是更合适的解决方案。其中，此函数不单独考虑轴，即它不应用插值 / 外插。它用所需数量的元素填充返回数组，迭代 C−order 中的 a，不考虑轴（如果新形状更大，则从头开始循环）。因此，此函数不适用于调整图像或每个轴代表单独且不同实体的数据的大小。

【例 5−12】numpy.resize 运行实例。

```
import numpy as np
a=np.array([[0, 1],[2, 3]])
b=np.resize(a,(2, 3))
c=np.resize(a,(1, 4))
d=np.resize(a,(2, 4))
print(b)
```

程序运行结果如下：

```
[[0 1 2]
[3 0 1]]
```

运行如下代码：

```
print(c)
```

程序运行结果如下：

```
[[0 1 2 3]]
```

运行如下代码：

```
print(d)
```

程序运行结果如下：

```
[[0 1 2 3]
```

```
[0 1 2 3]]
```

【例 5-13】在各种数据计算中经常用到的改变数组形状的函数 reshape（ ）。

```
import numpy as np
arrayA=np.arange(8)
a=np.reshape(arrayA,(2, 4))
print(a)
```

程序运行结果如下：

```
[[0 1 2 3]
[4 5 6 7]]
```

这里它把一个有 8 个元素的向量，转换成了形状为（4, 2）的一个矩阵。因为转换前后的元素数目一样，所以能够成功地进行转换，假如前后数目不一样的话，就会有错误 ValueError 报出。

5.2.9　数组的拼接和分裂

1. 数组的拼接

concatenate() 数组拼接方法是将多个数组沿某个坐标轴 axis 按 a1,a2,……顺序进行拼接。函数格式为 numpy.concatenate((a1,a2,…),axis=0)。其中，(a1,a2,…) 或 [a1,a2,…] 表示以元组或列表的形式将需要拼接的数组进行打包；a1,a2,…表示是需要拼接的多个数组；axis 表示指定拼接数组所沿的某个维度，默认为 0。其函数的返回值为拼接后的数组（副本）。

concatenate() 数组的拼接条件是 a1,a2,…需要在非拼接的坐标轴上拥有相同的 shape，否则会出错。其中，axis 为多维数组的某个维度或坐标轴。为简便起见，后面所述的轴等价于坐标轴。类似于数组元素的索引是用于访问数组的某个元素的。在 NumPy 中，axis 用于访问数组的某个轴，从 0 开始，最大到 ndim−1，共有 ndim 个轴。N 维数组可以理解 N 维空间，N 维空间拥有 N 个坐标轴。NumPy 可通过 axis 来索引某个具体的坐标轴。比如二维数组，它有 2 个维度 / 轴。第 1 维度对应 axis=0 轴，第 2 维度对应 axis=1 轴。多维数组的切片操作，本质上是沿着数组的每个轴分别做一维的切片操作，然后再将操作结果进行组合。

【例 5-14】两个一维数组的拼接。

```
import numpy as np
x=np.array([1921, 1949, 1978])
y=np.array([2015, 2020, 2025])
print(np.concatenate([x,y]))
```

程序运行结果如下：

```
[1921 1949 1978 2015 2020 2023]
```

vstack() 数组拼接方法是沿着第 0 轴（垂直方向）拼接数组，等价于 axis=0 的 concatenate()，v 代表 vertically。函数格式为 numpy.vstack((a1,a2,…))。其中，(a1,a2,…) 表示以元组的形式将需要拼接的数组进行打包；a1,a2, …表示是需要拼接的多个数组。其函数的返回值为拼接后的数组（副本）。然而，垂直方向对于超过二维的数组很难想象，因此，将这方法理解为沿着第 0 轴拼接数组。vstack() 数组的拼接条件是非拼接的 shape 必须要一致。

【例 5-15】vstack() 数组的拼接。

```
x1=np.array([1921, 1949, 1978])
```

```
x2=np.array([2015, 2020, 2023])
print(x1.shape,x2.shape)
```

程序运行结果如下：

```
(3,)(3,)
```

运行如下代码：

```
x3=np.vstack((x1,x2))
x4=np.concatenate((x1,x2),axis=0)
print('vstack:',x3.shape)
```

程序运行结果如下：

```
vstack:(2, 3)
```

运行如下代码：

```
print(x3)
```

程序运行结果如下：

```
[[1921 1949 1978]
[2015 2020 2023]]
```

运行如下代码：

```
print(x4)
```

程序运行结果如下：

```
[1921 1949 1978 2015 2020 2023]
```

hstack() 快捷数组拼接方法是沿着第 1 轴（水平方向）拼接数组，等价于 axis=1 的 concatenate()，h 代表 horizontally。函数格式为 numpy.hstack((a1,a2, …))。其中，(a1,a2,…) 表示以元组的形式将需要拼接的数组进行打包；a1,a2, …是表示需要拼接的多个数组。其函数的返回值为拼接后的数组（副本）。然而，水平方向对于超过二维的数组很难想象，因此，将这方法理解为沿着第 1 轴拼接数组。hstack（）快捷数组的拼接条件是非拼接的 shape 必须要一致。

2. 数组的分裂

split() 数组分裂方法是将一个数组分裂为多个子数组。函数格式为 numpy.split(ary,indices_or_sections,axis=0)。其中，ary 表示需要分裂的 NumPy 数组；indices_or_sections 表示分裂规则。如果输入是整数 N，那么指明原数组被等分为 N 份。另外，整数 N 必需要能整除 axis 轴方向的分裂数组 shape，否则会出错。如果输入是列表，则列表元素代表被分裂的索引位置，axis 指明沿哪个轴进行分裂，默认为 0。其函数的返回值为一组按规则分裂得到的含 N 个子数组集合。进一步，可以对返回的子数组集合进行解包操作，解包所需的子数组数量必须刚好等于 N，否则会出错。

【例 5-16】split() 数组分裂。

```
x=np.array([2006, 2010, 2011, 2015, 2016, 2020, 2021, 2025])
print(x)
```

程序运行结果如下：

```
[2006 2010 2011 2015 2016 2020 2021 2025]
```

运行如下代码：

```
# 等分为 4 份
x_split=np.split(x, 4)
print(x_split)
```

程序运行结果如下：

```
[array([2006, 2010]),array([2011, 2015]),array([2016, 2020]),array([2021, 2025])]
```

vsplit() 数组分裂方法是沿第 0 轴方向对数组进行分裂。函数格式为 numpy.vsplit(ary, indices_or_sections)，其中参数描述与 numpy.split() 函数一致。其函数返回值为一组沿第 0 轴方向按规则分裂得到的含 N 个子数组集合。

hsplit() 数组分裂方法是沿第 1 轴方向对数组进行分裂。函数格式为 numpy.hsplit(ary,indices_or_sections)，其中参数描述与 numpy.split() 函数一致。其函数返回值为一组沿第 1 轴方向按规则分裂得到的含 N 个子数组集合。

【例 5-17】vsplit() 数组和 hsplit() 数组分裂方法。

```
grid=np.arange(16).reshape((4, 4))
print(grid)
```

程序运行结果如下：

```
[[ 0 1 2 3]
 [ 4 5 6 7]
 [ 8 9 10 11]
 [12 13 14 15]]
```

运行如下代码：

```
upper,lower=np.vsplit(grid,[2])
print(upper)
```

程序运行结果如下：

```
[[0 1 2 3]
 [4 5 6 7]]
```

运行如下代码：

```
print(lower)
```

程序运行结果如下：

```
[[ 8 9 10 11]
  [12 13 14 15]]
```

运行如下代码：

```
left,right=np.hsplit(grid,[2])
print(left)
```

程序运行结果如下：

```
[[ 0 1]
  [ 4 5]
  [ 8 9]
  [12 13]]
```

运行如下代码：

```
print(right)
```

程序运行结果如下：

```
[[ 2 3]
  [ 6 7]
  [10 11]
```

```
[14 15]]
```

5.3 NumPy 数组的计算

NumPy 是 Python 数据科学和数值计算领域的重要工具之一。它提供了多维数组和各种数学函数，使得处理数据和进行科学计算变得更加高效和便捷。在数据科学和数值计算的领域，NumPy 是不可或缺的利器。本节介绍 NumPy 数组的计算。

5.3.1 NumPy 的通用函数

NumPy 通用函数，也可以称为 ufunc，是一种在 ndarray 数据中对逐个元素进行操作的函数。某些简单函数接收一个或多个标量数值，并产生一个或多个标量结果，而通用函数就是对这样的简单函数的向量化封装。

【例 5-18】一元通用函数。

```
arr=np.arange(10)
print(arr)
```

程序运行结果如下：

```
[0 1 2 3 4 5 6 7 8 9]
```

运行如下代码：

```
print(np.sqrt(arr))
```

程序运行结果如下：

```
[0.       1.       1.41421356 1.73205081 2.       2.23606798
2.44948974 2.64575131 2.82842712 3.       ]
```

运行如下代码：

```
print(np.exp(arr))
```

程序运行结果如下：

```
[1.00000000e+00 2.71828183e+00 7.38905610e+00 2.00855369e+01
 5.45981500e+01 1.48413159e+02 4.03428793e+02 1.09663316e+03
 2.98095799e+03 8.10308393e+03]
```

表 5-5 为一元通用函数及其描述。

表 5-5　一元通用函数

函数名	描述
abs、fabs	逐个元素地计算整数、浮点数或复数的绝对值
sqrt	计算每个元素的平方根（与 arr ** 0.5 相等）
square	计算每个元素的平方（与 arr ** 2 相等）
exp	计算每个元素的自然指数值 e^x 次方
log、log10、log2、log1p	分别对应自然对数（以 e 为底）、对数以 10 为底、对数以 2 为底、log（1+x）
sign	计算每个元素的符号值：1（正数）、0（0）、-1（负数）
ceil	计算每个元素的最高整数值（即大于等于给定数值的最小整数）
floor	计算每个元素的最小整数值（即小于等于给定整数的最大整数）
rint	将元素保留到整数位，并保持 dtype

续表

函数名	描述
modf	分别将数组的小数部分与整数部分按数组形式返回
isnan	返回数组元素是否是一个 NaN（非数值），形式为布尔值数组
isfinite、isinf	分别返回数组中的元素是否有限（非 inf、非 NaN）、是否无限的，形式为布尔值数组
cos、cish、sin、sinh、tan、tanh	常规三角函数及双曲三角函数
arccos、arccosh、arcsin、arcsinh、arctan、arctanh	反三角函数
logical_not	对数组元素按位取反

以上都是一元（unary）ufunc。另外一些，比如 add 或 maximum 则会接收两个数值并返回一个数组作为结果，因此称为二元通用函数。表 5-6 为二元通用函数及其对应描述。

表 5-6　二元通用函数

函数名	描述
add	将数组的对应元素相加
subtract	在第二个数组中，将第一个数组中包含的元素去除
multiply	将数组的对应元素相乘
divide,floor_divide	除或整除（放弃余数）
power	将第二个数组的元素作为第一个数组对应元素的幂次方
maximum	逐个元素计算最大值，fmax 忽略 NaN
minimum	逐个元素计算最小值，fmin 忽略 NaN

5.3.2　数组值求和

在 NumPy 中，可以使用 numpy.cumsum() 函数来求数组的累加和。这个函数会返回一个新的数组，其中的元素是原数组中对应元素及其之前所有元素的和。其函数格式为：

```
numpy.cumsum(a,axis=None,dtype=None,out=None)
```

按照所给定的轴参数返回元素的梯形累计和：axis=0，按照行累加；axis=1，按照列累加。

np.sum() 函数可以计算数组中所有元素的总和。该函数可以接受多个参数，其中最重要的一个参数是 axis，用于指定沿哪个轴进行求和。默认情况下，np.sum() 函数会对整个数组进行求和。

【例 5-19】np.sum() 函数求和。

```
import numpy as np
arr = np.array([[1, 2],[3, 4]])
sum1=np.sum(arr)
sum2=np.sum(arr,axis=0)
sum3=np.sum(arr,axis=1)
print(sum1)
```

程序运行结果如下：

```
10
```

运行以下代码：

```
print(sum2)
```

程序运行结果如下：

```
[4 6]
```

运行以下代码：

```
print(sum3)
```

程序运行结果如下：

```
[3 7]
```

在上面的示例中，首先，创建了一个包含 4 个元素的二维数组，使用 np.sum() 函数计算了该数组中所有元素的总和，并将结果赋值给变量 sum1。然后，使用 axis 参数指定沿行或列的方向进行求和，并将结果分别赋值给 sum2 和 sum3 变量。最后，使用 print() 函数将结果输出。

5.3.3 最大值和最小值

在 NumPy 中，求数组最大值和最小值的函数通常是 np.max() 和 np.min()。但是，np.amax() 和 np.amin() 以及 np.nanmax() 以及 np.nanmin() 这两个函数也可以用来求数组的最大值和最小值，还有 np.argmax() 和 np.argmin() 函数也可以用来求数组的最值。它们在某些特定情况下有所不同。

（1）np.max() 和 np.min()。np.max() 函数返回数组中的最大值。如果数组中有 NaN 值（Not a Number，表示缺失或无效数据），np.max() 会将其视为最大值，因此会返回除 NaN 之外的最大值。np.min() 函数返回数组中的最小值。如果数组中有 NaN 值，np.min() 会将其视为最小值返回。

（2）np.amax() 和 np.amin()。这两个函数与 np.max() 和 np.min() 在功能上是相同的，都是用来求数组的最大值和最小值。它们之间的区别在于 np.amax() 和 np.amin 可以接受一个 axis 参数来沿着指定轴操作。如果不指定 axis，则默认对整个数组操作。

（3）np.nanmax() 和 np.nanmin()。这两个函数与 np.max() 和 np.min 类似，但它在计算时会忽略数组中的 NaN 值，只考虑非 NaN 元素来计算最大值和最小值。

（4）np.argmax() 和 np.argmin()。这两个函数是返回数组中的最大值和最小值的索引值的函数，常用于分类器中，直接可以表示出分类的索引的数组的最大值和最小值或者是沿轴返回忽略任何 NaN 的数组的最大值和最小值。这两个函数的参数为通用的，其函数格式：

```
np.argmax/min(a,axis=None,out=None)
```

其中，a 即为输入的数组；axis 默认为 None，即输入的数组为平坦数组（即 reshape 为一维数组），axis=0 为列向量输入，axis=1 为行向量输入；如果提供 out 参数，则会插入输入数组当中。

在实际应用中，如果在处理包含 NaN 值的数组时，通常推荐使用 np.nanmax() 和 np.nanmin()，因为它们能够正确地处理 NaN 值，并只考虑非 NaN 元素来找到最大值和最小值。如果不关心 NaN 值，或者想要包括它们在内，那么可以使用 np.max() 和 np.min()。而 np.argmax() 和 np.argmin() 则是用来找到最大和最小元素的索引。

【例 5-20】求解最大值和最小值。

```
a=np.arange(6).reshape(2, 3)+10
print(a)
```

程序运行结果如下：

```
[[10 11 12]
 [13 14 15]]
```

```
print(np.argmax(a))
```

程序运行结果如下：

```
5
```

运行如下代码：

```
print(np.argmax(a,axis=0))
```

程序运行结果如下：

```
[1 1 1]
```

运行如下代码：

```
print(np.argmax(a,axis=1))
```

程序运行结果如下：

```
[2 2]
```

5.4　广播

NumPy 拥有在算术运算期间处理不同形状的数组的能力。对数组的算术运算通常在相应的元素上进行。在 NumPy 中当数组进行运算时，如果两个数组的形状相同，那么两个数组相乘就是两个数组的对应位相乘，这时要求维数相乘，并且各维度的长度相同。但是当运算中两个数组的形状不同时，NumPy 将会自动触发广播机制，所以要了解 NumPy 的广播机制，才能更好地进行数组的运算。

5.4.1　广播的介绍

广播（Broadcasting）是 NumPy 中用于处理不同形状数组间运算的一种强大机制。它允许 NumPy 在执行算术运算时处理不同形状的数组，而无须显式地重塑或复制数组。广播规则允许在某些情况下对不同形状的数组进行数学运算，而不需要改变它们的形状。

【例 5-21】如果两个数组 a 和 b 形状相同，即满足 a.shape==b.shape，那么 a*b 的结果就是 a 与 b 数组对应位相乘。这要求维数相同，且各维度的长度相同。

```
import numpy as np
a=np.array([1, 2, 3, 4])
b=np.array([10, 20, 30, 40])
c=a*b
print(c)
```

程序运行结果如下：

```
[ 10 40 90 160]
```

【例 5-22】当运算中的 2 个数组的形状不同时，NumPy 将自动触发广播机制。

```
import numpy as np
a=np.array([[0, 0, 0],
            [10, 10, 10],
            [20, 20, 20],
            [30, 30, 30]])
b=np.array([0, 1, 2])
print(a+b)
```

程序运行结果如下：

```
[[ 0 1 2]
 [10 11 12]
 [20 21 22]
 [30 31 32]]
```

图 5-3 展示了数组 b 如何通过广播来与数组 a 兼容。

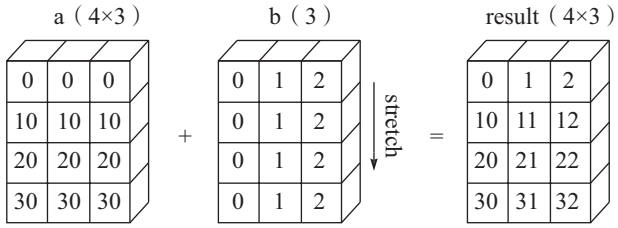


图 5-3　数组 b 通过广播与数组 a 兼容

【例 5-23】4 × 3 的二维数组与长为 3 的一维数组相加，等效于把数组 b 在二维上重复 4 次再运算。

```
import numpy as np
a=np.array([[0, 0, 0],
            [10, 10, 10],
            [20, 20, 20],
            [30, 30, 30]])
b=np.array([1, 2, 3])
# 重复 b 的各个维度
bb=np.tile(b,(4, 1))
print(a+bb)
```

程序运行结果如下：

```
[[ 1 2 3]
 [11 12 13]
 [21 22 23]
 [31 32 33]]
```

5.4.2　广播的规则

（1）让所有输入数组都向其中形状最长的数组看齐，形状中不足的部分都通过在前面加 1 补齐。

（2）输出数组的形状是输入数组形状的各个维度上的最大值。

（3）如果输入数组的某个维度和输出数组的对应维度的长度相同或者其长度为 1 时，这个数组能够用来计算，否则出错。

（4）当输入数组的某个维度的长度为 1 时，沿着此维度运算时都用此维度上的第一组值。

（5）对两个数组，分别比较它们的每一个维度（若其中一个数组没有当前维度则忽略），满足如下条件：数组拥有相同形状，当前维度的值相等，当前维度的值有一个是 1。若条件不满足，抛出 "ValueError: frames are not aligned" 异常。

5.4.3　广播的实际应用

Numpy 广播的实际应用主要体现在对多维数组的计算上，允许不同形状的数组进行计算。在进行计算时，如果两个数组的形状不同，Numpy 会自动地将较小的数组按照广播的原则调整到较大数组的大小，以使它们拥有兼容的形状，从而简化数组的操作。NumPy 通常与 Scipy（Scientific Python）和 Matplotlib（绘图库）一起使用，这种组合广泛用于替代 MatLab，是一个强大的科学计算环境，有助于通过 Python 学习数据科学或者机器学习。

1. 用 NumPy 广播实现相似度计算

代码格式如下：

```
D=np.sqrt(((U[:,np.newaxis,:]-U[np.newaxis,:,:])**2).sum(axis=2))
```

要利用广播机制，就得想办法把二维的矩阵 U 投射到高维空间，造成某个维度上元素个数的不相等。“ np.newaxis”是 NumPy 的一个常量，用于创建一个新的维度。“ U[:,np.newaxis,:]”和“ U[np.newaxis,:,:]”分别将矩阵 U 这个二维平面投射到三维空间中两个不同的方向。这两个平面相减（减法是一种 Element−wise 计算）就会触发广播机制，从而得到一个三维的差值数组（即三维张量 Tensor）。“**2”是对这些差值求平方，“sum(axis=2)”是将平方值沿着第三维度加到一起，再经过“np.sqrt”开方就得到了正确的结果。

2. 数组归一化

在 NumPy 中，广播可以用于数组归一化，即使不同形状的数组能够参与归一化运算。数组归一化通常是将数组元素的值转换到一个标准范围内，比如 [0,1] 或 [−1,1]。这对于许多机器学习算法和数据处理任务是非常重要的。

代码格式如下：

```
X=np.random.random((10, 3))
# 计算每一列的平均值
Xmean=X.mean(0)
Xmean
# 实现归一化
X_centered=X-Xmean
# 检验每一列的均值是否为 0
X_centered.mean（0）
```

首先，创建了一个形状为 (10,3) 的随机数组 X。接下来，计算了 X 的每一列的平均值，mean(0) 会沿着第一个轴（即行）计算平均值，从而得到每一列的平均值并输出。然后，Xmean 来中心化 X，即让每一列的均值变为 0，它利用了 NumPy 的广播机制。因为 Xmean 是一个一维数组，它的形状与 X 的列数相匹配，所以 X−Xmean 会逐元素地从 X 中减去对应的列均值。最后，验证中心化后的 X_centered 的每一列均值是否为 0，如果 X_centered 的计算是正确的，那么这行代码的输出应该是一个接近 0 的数组，表示每一列的均值已经被成功地减去了。

3. 画一个二维函数

要使用 NumPy 的广播机制来绘制一个二维数组，首先需要创建一个二维数组，然后使用 matplotlib 等绘图库来显示这个数组。这里使用 matplotlib 来可视化一个简单的二维数组，展示如何通

过 NumPy 的广播特性来处理数据。虽然广播本身并不直接用于绘图，但可以用来准备要绘制的数据。

【例 5-24】画一个二维函数。

```
import numpy as np
import matplotlib.pyplot as plt
x=np.linspace(0, 5, 50)
y=np.linspace(0, 5, 50)[:, np.newaxis]
z=np.sin(x)**10+np.cos(10+y*x)*np.cos(x)
plt.imshow(z,origin='lower',extent=[0, 5, 0, 5],
           cmap='viridis')
plt.colorbar()
plt.show()
```

二维函数程序运行结果如图 5-4 所示。

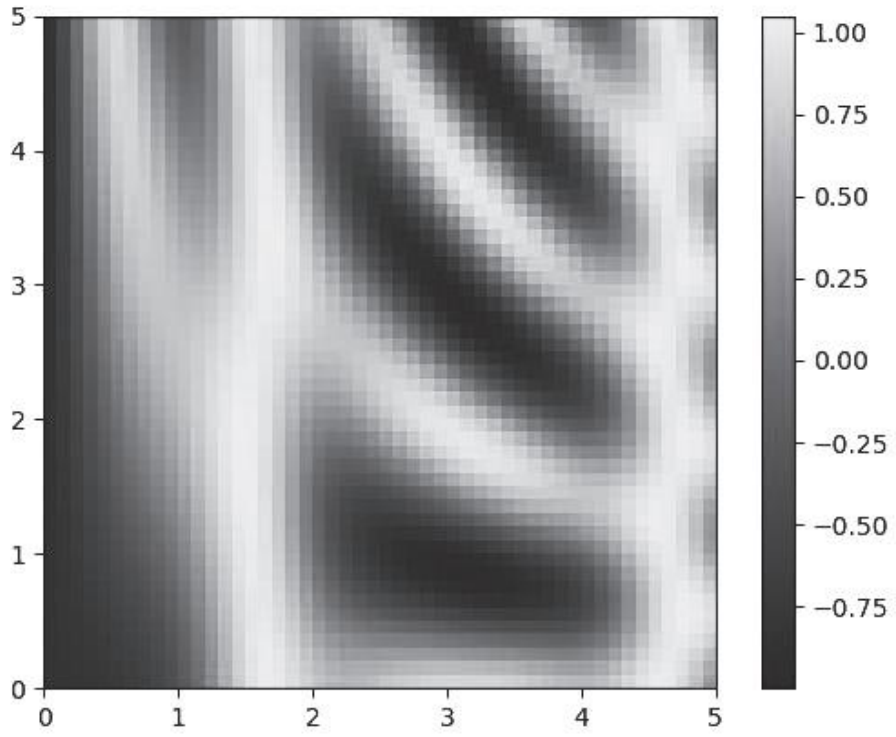


图 5-4　二维函数结果

5.5　实战案例

数组（NumPy 数组，也称 ndarray）是 NumPy 库的核心功能，也是重要的特性之一。NumPy 数组是一个多维数组对象，用于存储同一类型的数据（如整数、浮点数、复数、字符串等）的集合。相比 Python 的内置列表（list），NumPy 数组在存储和处理大量数据时具有更高的效率和性能。本实战案例是使用 NumPy 分别记录学生的姓名、班级和成绩。在这个场景中，将应用 NumPy 中的记录数组（又称之为结构化数组）。这类数组可以为每一列指定数据类别（如 str 字符串、int 整数、bool 布尔运算等）和名称，让数据筛选变得更简单强大。

【例 5-25】列出低于及格分数的学生姓名。

```
import numpy as np
data=np.array([
    (" 丁一 ", 1, 87),
    (" 刘二 ", 2, 68),
    (" 张三 ", 3, 72),
    (" 李四 ", 3, 55),
    (" 王五 ", 3, 93),
    (" 赵六 ", 2, 81),
    (" 孙七 ", 1, 75),
    (" 周八 ", 1, 88),
    (" 吴九 ", 2, 64),
    (" 郑十 ", 2, 49)
],dtype=[(" 姓名 ",str, 10),(" 班级 ",int),(" 分数 ",int)])
fail=data[data[" 分数 "]<60][" 姓名 "]
print(fail)
```

程序运行结果如下：

```
[' 李四 ' ' 郑十 ']
```

【例 5-26】列出分数最高的学生姓名、班级及分数。

```
top=data[data[" 分数 "]==np.max(data[" 分数 "])]
print(top)
```

程序运行结果如下：

```
[(' 王五 ', 3, 93)]
```

【例 5-27】各班级的学生姓名、班级及分数，需按分数大小逆序排列。

```
top=data[data[" 分数 "]==np.max(data[" 分数 "])]
print(top)
```

程序运行结果如下：

```
[(' 周八 ', 1, 88)(' 丁一 ', 1, 87)(' 孙七 ', 1, 75)]
[(' 赵六 ', 2, 81)(' 刘二 ', 2, 68)(' 吴九 ', 2, 64)(' 郑十 ', 2, 49)]
[(' 王五 ', 3, 93)(' 张三 ', 3, 72)(' 李四 ', 3, 55)]
```

单元小结

在本单元中介绍了 NumPy 库的基本概念和用法。其中，NumPy 是 Python 中流行的科学计算库之一，提供了高性能的多维数组对象和各种数学函数，且核心数据结构是 ndarray。它是一个多维数组对象，可以存储任意类型的数据。不仅如此，NumPy 还提供了丰富的数组操作函数，包括数组创建、切片、索引、转置、广播和聚合等。它支持各种数学函数，包括基本的算术运算、三角函数、指数和对数函数、矩阵运算等。NumPy 还提供了多种排序、搜索和统计函数，以及随机数生成函数。NumPy 可以与其他 Python 库集成，例如 Matplotlib、Pandas 和 Scikit-learn 等，使得科学计算更加便捷和高效。故 NumPy 是 Python 中必不可少的科学计算库，它提供了高效、灵活和可扩展的多维数组对象和数学函数。

单元练习

一、单选题

1. 对于 DataFrame 对象，以下说法错误的是（　　）。

A. DataFrame 对象是一个表格型的数据结构

B. DataFrame 对象的列是有序的

C. DataFrame 对象列与列之间的数据类型可以互不相同

D. DataFrame 对象每一行都是一个 Series 对象

2. 文件写操作时，writelines 方法的参数不可以是（　　）。

A. 列表　　B. 元组

C. 字典　　D. 整数

3. wordcloud.WordCloud 函数中的参数 mask 是用来设置（　　）的。

A. 词云的颜色　　B. 词云的尺寸

C. 词云的遮置形状　　D. 词云的坐标

4. 不是 NumPy 数组创建的函数是（　　）。

A. array　　B. ones_like

C. eye　　D. reshape

5. 假设有命令如下：

```
import NumPy as np
bArray=np.array([[1, 2, 3],[4, 5, 6]])
```

则 bArray.ndim 的结果是（　　）。

A. 1　　B. 2

C. 3　　D. 4

二、程序设计题

1. 使用 tile 函数创建一个棋盘。

2. 对一个 6×7×8 的数组，找出第 100 个元素的下标。

3. 创建一个二维数组，边为 1，其余为 0。

4. 使用生成器创建一个大小为 10 的数组。

5. 创建一个大小为 10 的数组并排序。