计算机类专业“互联网＋”创新型精品教材

Python 数据分析

主　编　冯明卿　袁　帅　王晓燕

北京出版集团
北京出版社

Python 数据分析

主编 冯明卿 袁帅 王晓燕

北京出版集团
北京出版社

图书在版编目（CIP）数据

Python 数据分析 / 冯明卿，袁帅，王晓燕主编. 北京：北京出版社，2025. 8. -- ISBN 978-7-200-19642-9

Ⅰ. TP312.8

中国国家版本馆 CIP 数据核字第 2025CA3887 号

Python 数据分析

Python SHUJU FENXI

主　　编：冯明卿　袁　帅　王晓燕
出　　版：北京出版集团
　　北 京 出 版 社
地　　址：北京北三环中路 6 号
邮　　编：100120
网　　址：www.bph.com.cn
总 发 行：北京出版集团
经　　销：新华书店
印　　刷：定州启航印刷有限公司
版 印 次：2025 年 8 月第 1 版　2025 年 8 月第 1 次印刷
成品尺寸：185 毫米 ×260 毫米
印　　张：13
字　　数：292 千字
书　　号：ISBN 978-7-200-19642-9
定　　价：44.00 元

教材意见建议接收方式：010-58572341　邮箱：jiaocai@bphg.com.cn
如有印装质量问题，由本社负责调换
质量监督电话：010-82685218　010-58572341　010-58572393

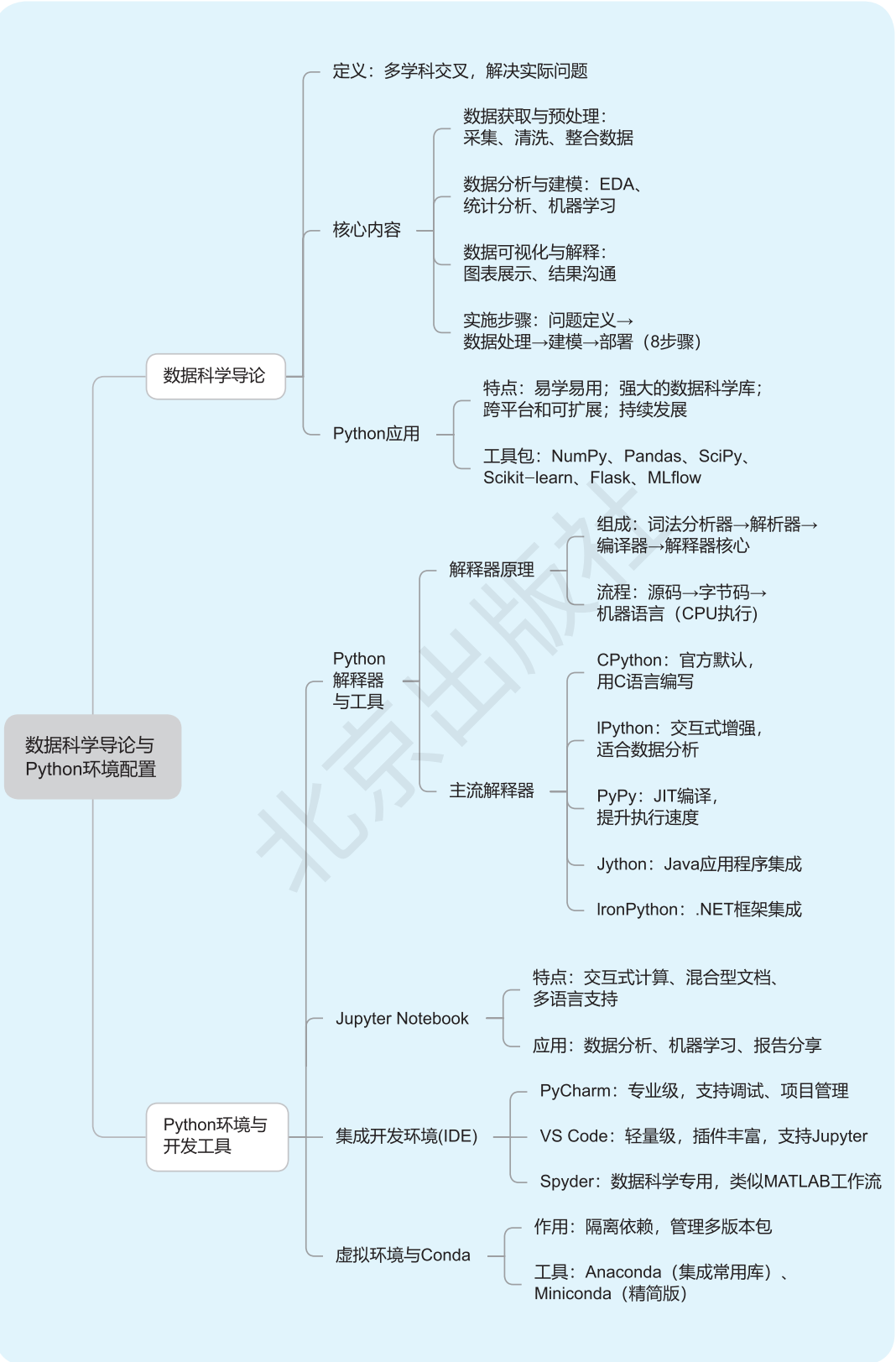
目录

项目一

数据科学导论与 Python 环境配置

数据科学是一门深度融合统计学、数学、计算机科学、领域专业知识和信息科学的多学科交叉型学科，其核心目标是通过系统化的方法从数据中提取知识或价值，以辅助决策并解决复杂的实际问题。其典型工作流程包括数据获取与清洗、探索性分析、特征工程、建模与验证、结果可视化及部署应用等多个阶段。在这一过程中，专业人员需综合运用数学统计模型、机器学习算法与高效的计算工具，同时必须始终关注数据伦理、隐私保护与相关合规要求，确保数据使用的合法性与安全性。

在工具层面，数据科学依赖丰富且持续演进的技术栈，其中，Python 凭借其简洁易读的语法、强大的社区支持和覆盖全流程的开源库生态，逐渐成为该领域的首选编程语言。例如，NumPy 和 SciPy 提供科学计算基础，Pandas 支持灵活的数据处理，Scikit－learn 包含常用的机器学习算法，而 Matplotlib、Seaborn 等库则用于生成多样的可视化图表。此外，Python 生态还集成了如 Jupyter Notebook 这样的交互式开发环境，便于实时探索与展示分析过程；PyCharm、VS Code 等 IDE 助力代码编写与调试；Conda 等环境管理工具则帮助维护项目依赖的独立性，保障复现性与协作效率。这些工具共同构成了从数据预处理、模型构建、评估优化到最终部署的完整支持体系，不仅能提升开发效率，也能保障团队协作与知识共享，这些都为数据科学在金融、医疗、教育、工业等众多行业持续发挥价值提供了核心驱动力。当然，在数据科学的技术实践中不断强调融入伦理意识与社会责任，也是确保数据科学在各行业中可持续发展和造福社会的关键。


数据科学导论与Python环境配置
数据科学导论
定义：多学科交叉，解决实际问题
核心内容
数据获取与预处理：采集、清洗、整合数据
数据分析与建模：EDA、统计分析、机器学习
数据可视化与解释：图表展示、结果沟通
实施步骤：问题定义→数据处理→建模→部署（8步骤）
Python应用
特点：易学易用；强大的数据科学库；跨平台和可扩展；持续发展
工具包：NumPy、Pandas、SciPy、Scikit-learn、Flask、MLflow
Python环境与开发工具
Python解释器与工具
解释器原理
组成：词法分析器→解析器→编译器→解释器核心
流程：源码→字节码→机器语言（CPU执行）
主流解释器
CPython：官方默认，用C语言编写
IPython：交互式增强，适合数据分析
PyPy：JIT编译，提升执行速度
Jython：Java应用程序集成
IronPython：.NET框架集成
Jupyter Notebook
特点：交互式计算、混合型文档、多语言支持
应用：数据分析、机器学习、报告分享
集成开发环境(IDE)
PyCharm：专业级，支持调试、项目管理
VS Code：轻量级，插件丰富，支持Jupyter
Spyder：数据科学专用，类似MATLAB工作流
虚拟环境与Conda
作用：隔离依赖，管理多版本包
工具：Anaconda（集成常用库）、Miniconda（精简版）

任务一 数据科学导论

任务描述

有段时期，村庄的庄稼产量突然下降，小红决定运用她所掌握的技能来找出原因。她收集了天气、土壤和作物生长的数据，并进行图表和模型分析，分析结果揭示干旱和不当种植时间是问题的根源。小红据此提出解决方案：调整种植时间和引入新的灌溉系统。方案被采纳后，庄稼产量很快恢复。这个故事反映出数据科学不仅是一门科学，更是解决问题的强大工具。本任务可以让我们认识并理解数据科学。

任务目标

1. 理解数据科学的概念，掌握数据科学的核心要素，包括大数据、数据分析、机器学习、统计推断等基本概念。
2. 能够完成数据获取、清洗、探索性分析到模型构建和验证的整体工作流程。
3. 培养数据思维，学会从数据角度思考问题。
4. 树立正确的数据价值观，恪守数据安全与隐私保护的职业道德规范。

知识链接

一、数据科学导论

数据科学是一门多学科交叉的综合性学科，它结合了统计学、计算机科学、数学、信息科学、领域专业知识以及有效的沟通技巧，其核心目标是通过系统性、科学的方法，处理、分析和解释数据，以解决实际问题、支持决策制定和推动创新。

数据科学包括数据获取与预处理、数据分析与建模、数据可视化与解释三部分内容。三部分内容介绍如下。

（一）数据获取与预处理

（1）数据采集。从各种数据源（如数据库、API、传感器、社交媒体、文件等）收集所需数据。

（2）数据清洗。去除数据中的错误、不完整、不一致或无关的记录，确保数据质量。

（3）数据整合与转换。将不同格式、结构的数据整合成统一的格式，进行必要的数据转换，如归一化、标准化、特征工程等，使其适应分析模型的要求。

（二）数据分析与建模

（1）探索性数据分析（EDA）。通过图表、统计摘要等手段对数据进行初步观察，发现模式、趋势、异常值等基本信息，为后续分析提供方向。

（2）统计分析。运用统计学方法（如假设检验、回归分析、聚类分析、时间序列分析等）对数据进行定量描述、推断和预测。

（3）机器学习与人工智能。构建和训练模型（如监督学习、无监督学习、强化学习模型）来自动从数据中学习规律，进行预测、分类、聚类、关联分析等任务。

（三）数据可视化与解释

（1）数据可视化。借助图表、仪表板、地图等视觉元素，将复杂数据转化为易于理解的图形，以便于洞察数据模式、传达分析结果和决策依据。

（2）结果解释与沟通。以清晰、简洁的语言解释数据分析结果，将其与业务问题或研究问题关联起来，为决策者、利益相关者或公众提供易于理解的洞察和建议。

二、数据科学项目的管理与实施

数据科学利器

数据科学项目的管理与实施一般包括以下八个步骤。

（1）问题定义与理解。明确项目目标，理解业务背景和需求，确定关键问题和成功指标。

（2）数据收集与整理。根据问题需求获取数据，进行数据清洗、整合与转换。

（3）探索性分析。对数据进行初步探索，发现潜在关系、异常值和数据分布特征。

（4）特征工程。根据问题特性创建、选择或转换有意义的特征变量。

（5）模型选择与构建。选择合适的统计模型或机器学习算法，训练模型并进行性能评估。

（6）模型验证与优化。使用交叉验证、网格搜索等方法优化模型参数，验证模型泛化能力。

（7）结果解释与可视化。将分析结果以易于理解的方式呈现，解释模型预测和发现的模式。

（8）部署与监控。将模型集成到业务流程或应用中，持续监控模型性能并进行必要的更新。

对数据科学的研究，业界也出现了很多工具和技术，包括编程语言及其对应的数据分析工具库、可视化工具、管理系统等。编程语言常见的有 Python、R、SQL 等，比较成熟的数据分析库包括 NumPy、Pandas、SciPy、Scikit－learn、TensorFlow、PyTorch 等，数据的可视化工具包括 Matplotlib、Seaborn、Tableau、Power BI 等，管理系统包括数据库管理系统（如关系型数据库、非关系型数据库、数据仓库等）、各类云环境平台与服务（如阿里云、华为云、百度云、腾讯云及自建私有云等）、代码的版本控制（如 git、svn 等）、项目管理工具（如禅道、jira 等）。

在数据科学实践中，还应重视数据隐私、安全、伦理与法律法规的合规性。要遵守数据最小化原则，通过数据脱敏、加密等措施保护个人敏感信息，要遵从国家数据保护法规，确保公平、透明、负责任地使用数据科学成果。

三、Python在数据科学中的应用

Python凭借其易用性、强大的数据科学库、跨平台与可扩展性以及活跃的社区，已成为数据科学领域事实上的标准语言。无论是数据的预处理、统计分析、可视化，还是机器学习、数据工程乃至模型部署，Python都能提供高效、灵活的解决方案，助力数据科学从业者完成各项任务。

我们在本项目中，主要使用Python语言及其工具包完成我们的项目任务，主要是基于Python的以下特点。

（一）易学易用

Python具有简洁、清晰的语法，对新手友好，学习曲线平缓，使得非计算机背景的从业者也能快速上手进行数据分析。同时，Python有丰富的文档、教程、社区资源（如Stack Overflow、GitHub），能为学习和解决问题提供强大支持。

（二）拥有强大的数据科学库

Python拥有完善的数据科学生态系统，如NumPy、Pandas、Matplotlib、Scikit-learn等库几乎涵盖了数据科学全流程所需的功能，能极大地提高工作效率。

（三）跨平台与可扩展性

Python支持Windows、macOS、Linux等多种操作系统，且具有良好的可移植性。Python可通过Cython、Pybind11等工具与C/C++、Fortran等编译型语言无缝集成，实现高性能计算。

（四）活跃的社区与持续发展

Python社区庞大且活跃，不断涌现出新的数据科学相关项目与工具，这些确保了Python在数据科学领域的前沿地位。而且，国内外企业与学术界广泛采用Python进行数据科学工作，形成了丰富的实践案例与最佳实践典范。

针对数据科学的工作，Python在数据处理的各环节中，都有较成熟的工具包，能够帮助工程师快速地开展数据科学的相关工作。具体介绍如下。

1. 数据获取与预处理

（1）数据读取。Python提供了Pandas库，能够轻松读取各种数据格式（如CSV、Excel、JSON、SQL数据库等），并将数据转换为DataFrame对象，便于后续操作。

（2）数据清洗。Pandas提供了丰富的数据清洗功能，如处理缺失值、异常值、重复数据，以及进行数据类型转换、数据标准化等。

（3）数据整合。通过Pandas的合并、拼接等操作，可以方便地整合来自不同数据源的数据。

2. 数据分析

（1）统计分析。Python的NumPy、SciPy库提供了丰富的统计函数，用于计算描述性统计量、执行假设检验、拟合概率分布等。

（2）可视化。Matplotlib、Seaborn等库支持创建各种图表（如直方图、散点图、箱线

图等），用于数据可视化与探索性数据分析（EDA）。

（3）机器学习。Python 拥有强大的机器学习库，如 Scikit－learn、TensorFlow、PyTorch 等，支持各种监督学习、无监督学习、强化学习算法的训练与应用。

3. 数据工程

（1）数据管道。Python 的 Airflow、Luigi 等库可用于构建复杂的数据处理管道，以进行自动化数据清洗、转换、加载（ETL）等流程。

（2）大数据处理。借助 Apache Spark（PySpark）、Dask 等库，Python 可以处理大规模数据集，实现分布式计算。

4. 部署与运维

（1）API 开发。使用 Flask、Django 等 Web 框架，可以将数据科学模型封装为 RESTful API，便于与其他系统集成。

（2）模型服务化。借助 MLflow、Kubeflow 等工具，Python 支持模型的版本管理、部署、监控与更新。

任务实施

让我们以一个简单的数据科学任务——“预测房价”为例，来帮助理解数据科学导论中的一些基本概念。这个任务的具体目标是基于房屋的特征（如面积、卧室数量、位置等）预测其售价。具体步骤如下。

（1）定义问题。确定目标变量为房屋售价；确定特征变量包含面积、卧室数量、浴室数量、位置、年份等。

（2）数据收集。从公共数据源或房地产网站获取房价数据。

（3）数据清洗。检查缺失值，决定是删除还是填充，识别并处理异常值。

（4）数据探索。使用描述性统计分析来了解数据的基本特征；绘制图表，如直方图、箱线图，来可视化数据分布。

（5）特征工程。创建新特征，如从地址提取社区信息；对类别特征进行编码，如使用独热编码。

（6）模型选择。选择一个回归模型，如线性回归。

（7）模型训练。将数据分为训练集和测试集，使用训练集训练模型。

（8）模型评估。使用测试集评估模型性能；计算均方误差（MSE）或均方根误差（RMSE）。

（9）模型优化。调整模型参数，如正则化强度；尝试不同的模型，如决策树或随机森林。

（10）结果解释。解释模型的系数，了解每个特征对房价的影响。

（11）部署模型。将模型部署为一个 Web 应用，用户可以输入房屋特征来预测价格。

（12）监控和维护。定期检查模型的预测性能，确保其准确性。

（13）沟通和报告。准备报告，展示模型的预测能力和特征的重要性。

我们通过一段 Python 代码来实现上述流程。

```
    import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LinearRegression
    from sklearn.metrics import mean_squared_error
    # 加载数据
    data = pd.read_csv('housing_data.csv')
    # 数据清洗
    data = data.dropna()  # 删除缺失值
    data = data[data['Price'] < data['Price'].quantile(0.99)]  # 去除异常值
    # 特征工程
    data['TotalRooms'] = data['Bedrooms'] + data['Bathrooms']
    # 分割数据
    X = data[['TotalRooms', 'SquareFeet', 'Location']]
    y = data['Price']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
    # 模型训练
    model = LinearRegression()
    model.fit(X_train, y_train)
    # 模型评估
    y_pred = model.predict(X_test)
    mse = mean_squared_error(y_test, y_pred)print(f'Mean Squared Error:
{mse}')
    # 结果解释
    print(f'Model Coefficients: {model.coef_}')
```

现在我们还不需要看懂这些代码的含义，仅需直观上认识数据科学的处理过程，随着本书的学习，我们将更加深入地理解这些代码的逻辑和过程。

为了生成上述代码中的基础数据，可以编写一个代码来实现。

```
    import pandas as pdimport numpy as np
    # 设置随机种子以获得可重复的结果
    np.random.seed(42)
    # 生成随机数据
    data = {
        'Address': [f'{i} {np.random.choice(["Maple", "Oak", "Pine", "Beech",
"Cedar"])} St' for i in range(1, 101)],
        'Bedrooms': np.random.randint(1, 5, size=100),
        'Bathrooms': np.random.randint(1, 3, size=100),
        'SquareFeet': np.random.randint(500, 3000, size=100),
        'Location': np.random.choice(['Downtown', 'Suburban', 'Urban',
'Rural'], size=100),
        'Price': np.random.randint(200000, 800000, size=100),
        'YearBuilt': np.random.randint(1950, 2020, size=100)}
    # 创建DataFrame
    df = pd.DataFrame(data)
    # 保存为CSV文件
    df.to_csv('housing_data.csv', index=False)
```

思考与练习

一、选择题

1. 数据科学的核心要素不包括（　　）。

A. 大数据

B. 数据分析

C. 机器学习

D. 软件工程

2. 在数据科学的基本流程中，以下（　　）通常不是必要的。

A. 数据获取

B. 数据清洗

C. 数据加密

D. 模型构建

3. Python 在数据科学中的作用不包括（　　）。

A. 数据预处理

B. 统计分析

C. 编译操作系统

D. 机器学习

4. 在数据科学项目中，以下（　　）通常不是最后一步。

A. 部署与监控

B. 结果解释与可视化

C. 模型验证与优化

D. 问题定义与理解

二、填空题

1. 数据科学包括数据获取与预处理、数据分析与建模、数据可视化与解释三部分内容。其中，数据分析与建模通常包括 _____、_____、_____ 等步骤。

2. Python 的 _____ 库提供了丰富的数据清洗功能，如处理缺失值、异常值、重复数据。

3. 在 Python 中，使用 _____ 等库可以创建各种图表，用于数据可视化与探索性数据分析（EDA）。

三、思考题

1. 考虑到数据隐私和安全的重要性，数据科学家在处理敏感数据时应采取哪些措施？

2. 描述一个场景，说明如何使用 Python 及其数据科学库来解决一个实际的业务问题。

任务二 Python 环境配置

任务描述

Python 是数据科学的强大工具，需要搭建自己的 Python 环境。可使用 pip 安装数据处理库 NumPy、Pandas 和可视化库 Matplotlib，这些库可以让我们的编程如虎添翼。除了基础环境，还可以使用 Jupyter Notebook 方便地编写代码，导入数据，绘制图表。本任务，我们将完成 Python 的环境搭建。

任务目标

1. 能够独立完成 Python 程序的安装，并完成系统环境变量的配置。

2. 配置 Python 数据分析环境，能够独立安装和配置 Python 相关的数据分析环境，如 Anaconda，设置虚拟环境等。

3. 执行简单的数据分析任务：能够导入数据，进行基本的数据查看、筛选、统计和可视化操作。

4. 能够完成 Jupyter Notebook 等工具的安装，并熟悉工具的基本用法。

5. 将个人技术学习与国家科技发展需求紧密结合，以技术创新助力国家数字经济建设。

知识链接

一、Python解释器

（一）Python 解释器的定义

在计算科学中，编程语言是人类与计算机沟通的桥梁。而 Python 解释器是一种专门设计来理解和执行 Python 源代码的软件程序。源代码是以文本形式书写的 Python 程序，包含了一系列遵循 Python 语法规则的指令和表达式。由于 CPU 直接处理的是机器语言（即由 0 和 1 组成的二进制指令），因此，为了让 CPU 理解并执行 Python 代码，就需要有一个中间环节来进行翻译。这个翻译就是 Python 解释器。

（二）Python 解释器的构成

Python 解释器主要由以下几个关键部分构成。

（1）词法分析器（Lexer）。词法分析器也称为扫描器，负责将源代码分解成一系列不可再分的符号单元，称为词法标记（lexical tokens）。这些标记对应 Python 语言中的关键字、标识符、运算符、标点符号、字符串和数值等基本元素。词法分析是将源代码从字符流转换为结构化数据的第一步。

（2）解析器（Parser）。解析器接收词法分析器产生的词法标记流，并根据 Python 语言的语法规则构建抽象语法树（Abstract Syntax Tree, AST）。AST 是源代码的抽象语法结构的树状表现形式，它以层次化的方式表示了源代码的结构和逻辑。AST 每个节点代表一个编程构造（如函数定义、条件语句、赋值操作等），节点间的连接反映了这些构造之间的关系。解析阶段确保了源代码符合 Python 语言的语法规则。

（3）编译器（Compiler）。尽管 Python 通常被视为解释型语言，但其解释过程实际上包含了一个编译阶段。Python 解释器中的编译器将 AST 转换为一种中间形式，即 Python 字节码（Python Bytecode）。字节码是一种低级、平台无关的指令集，类似于汇编语言，但比机器语言更高级、更易理解。Python 字节码文件通常以 .pyc 或 .pyo（优化后的字节码）为扩展名。

（4）解释器核心（Interpreter Core）。解释器核心负责执行字节码。它包含一个虚拟机（Virtual Machine, VM），该 VM 逐条读取字节码指令并执行相应的操作。虚拟机内部有栈（用于存放数据和操作数）、寄存器（存储临时变量和状态信息）等结构，以模拟一个简单的计算环境。解释器在执行字节码时，会根据需要动态加载模块、处理异常、管理内存等。

（三）Python 语言的编译和执行流程

Python 在执行时，先将 .py 文件中的源代码编译成 Python 的 byte code（字节码），然后再由 Python 虚拟机来执行这些编译好的 byte code。这种机制的基本思想跟 Java、.NET 是一致的。

Python 语言的编译和执行流程如下。

（1）执行 source.py 后，将会先启动 Python 的解释器。

（2）Python 解释器的编译器会将 .py 源文件编译（解释）成字节码生成 PyCodeObject 字节码对象存放在内存中。

（3）Python 解释器的虚拟机将执行内存中的字节码对象转化为机器语言，虚拟机与操作系统交互，使机器语言在机器硬件上运行。

（4）运行结束后 python 解释器则将 PyCodeObject 写回到 pyc 文件中。当 python 程序第二次运行时，首先程序会在硬盘中寻找 pyc 文件，如果找到，则直接载入，否则就重复上面的过程。

（四）Python 解释器的种类

Python 解释器主要有以下几种。

（1）CPython。CPython 是官方默认且最广泛使用的 Python 解释器，用 C 语言编写。它实现了 Python 语言规范，并提供了 Python 标准库。大多数 Python 用户和第三方库都基于 CPython 进行开发和测试。

（2）IPython。IPython 是一个基于 CPython 的交互式解释器，增强了用户界面，提供了诸如自动补全、交互式调试、丰富的输出格式化等功能，常用于数据分析、科学计算等领域的交互式工作。

（3）PyPy。PyPy 采用了即时编译（JIT, Just－In－Time Compilation）技术，能够在运

行时将热点 Python 字节码动态编译为更高效的机器码，从而在某些场景下显著提升执行速度。PyPy 兼容 CPython 标准库，适用于对性能要求较高的应用。

（4）Jython。Jython 是运行在 Java 平台上的 Python 解释器，可以直接将 Python 代码编译为 Java 字节码并在 Java 虚拟机（JVM）上执行。这使得 Python 程序能够无缝地调用 Java 库，并与 Java 应用程序集成。

（5）IronPython。类似 Jython，IronPython 是专为 .NET 框架设计的 Python 解释器，允许 Python 代码编译为 .NET 的中间语言（CIL），在 CLR（Common Language Runtime）上执行。这使得 Python 能够与 .NET 生态系统中的其他语言和库交互。

二、Jupyter Notebook

安装完 Python，会自动包含一个 Jupyter Notebook 工具。Jupyter Notebook 是一款基于 Web 的交互式计算环境（如图 1－2－1），它允许用户创建和分享包含实时代码、方程式、可视化图表和文字注释的文档，特别适用于数据分析、机器学习、教育和研究等领域。Jupyter Notebook 起源于 IPython 项目，后来发展成为支持多种编程语言，通过内核系统（kernels）连接不同语言的运行环境。

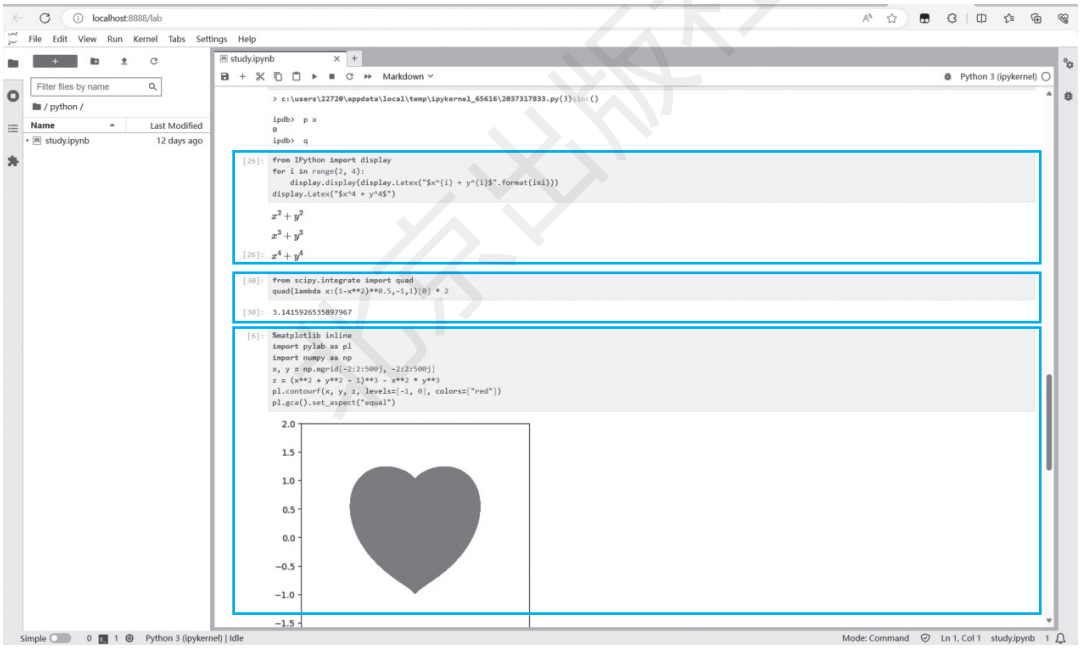


图 1－2－1　Jupyter Notebook 界面视图

Jupyter Notebook 有以下特点。

（一）交互式计算

用户可以编写代码并在浏览器中直接运行代码单元格，看到即时的输出结果。

（二）混合型文档

结合文本、代码、数学表达式（使用 LaTeX 语法）、图像、视频等多种元素，形成可读性强的叙述性报告或教学材料。

（三）支持多种编程语言

除了 Python，还可以支持 R、Julia、Scala、Ruby 等众多编程语言，只需安装对应的语言内核。

（四）动态可视化

能够轻松生成并嵌入图表和可视化结果，例如 matplotlib、seaborn 等库绘制的图表。

（五）可分享与协作

Notebooks 可以导出为多种格式（如 HTML、PDF 或纯文本），并且可以通过邮件、GitHub 或其他在线平台与他人分享和协作。

（六）版本控制友好

Notebooks 文件以 JSON 格式存储，扩展名为 .ipynb，这使得它们可以很容易地被纳入版本控制系统如 Git 进行版本管理和团队合作。

（七）灵活的插件系统

Jupyter Notebook 有一个丰富的插件生态系统，可以添加额外功能，如代码格式化、笔记本预览、主题定制等。

Jupyter Notebook 是一个强大的开放源码工具，能极大地提高数据科学家、工程师和教育工作者的工作效率，促进研究实践的开放透明。

三、基本IDE

Python 的集成开发环境（IDE）是专为 Python 编程设计的一体化软件套件，提供了代码编辑、调试、项目管理、版本控制、智能提示、交互式运行等多种功能，旨在提升开发效率和代码质量。以下是对几种常见 Python IDE 工具的简介。

（一）PyCharm

PyCharm 是由 JetBrains 公司开发的一个 Python IDE，有社区版和专业版之分，对学生来讲，社区版功能即可满足需要。功能包括代码编辑、调试、项目管理、版本控制（Git/SVN）、单元测试、代码审查、重构、数据库管理等；支持断点、步进、变量监视、内联调试、多进程 / 线程调试等高级功能；提供智能代码补全、实时错误检测、类型推断、代码导航、代码结构视图等；支持 Windows、macOS、Linux 操作系统。在科学计算与数据分析方面，集成了 Jupyter Notebook、支持交互式 Python Console、与数据科学库如 NumPy、Pandas、Matplotlib 等功能。

（二）Visual Studio Code with Python Extension

Visual Studio Code with Python Extension 是由微软开发的一个免费的 IDE，是一个基于 Electron 构建的通用代码编辑器，通过安装 Python 扩展实现 Python 开发环境；有丰富的插件市场，可根据个人需求扩展功能，不仅限于 Python。可支持 Windows、macOS、Linux 环境；提供语法高亮、代码片段、自动补全、错误检查、代码格式化（通过 black 等）、代码重构（通过 rope 等）等功能。在科学计算和数据分析方面，可通过 Jupyter Notebook 内核在 VS Code 中直接编辑和运行 Jupyter Notebook。支持 Python 代码

调试，包括断点、变量查看、调用堆栈等。

（三）Spyder

Spyder 是专为数据科学家和工程师设计的，集成 IPython Console、变量查看器、数据查看器等。Spyder 支持实时代码执行、变量查看、绘图等，类似于 MATLAB 的工作流；内置调试器，具备断点、步进、变量监视等调试功能；支持 Windows、macOS、Linux。基于 GPLv3 开源许可证发布，可免费使用。

四、虚拟机环境与Conda

我们在刚开始学习 Python 时，直接在 Windows 上安装一个 Python 就可以通过命令行执行 Python 程序。但随着学习深入，比如在学习深度学习的内容时，需要安装 keras、tensorflow 框架等，这时候如果盲目地安装，就容易出问题。因为每个框架依赖的包的版本不同，而我们只能在自己电脑上安装一个 Python 版本，无法应对不同框架不同 Python 版本的需求。为了解决这些问题，就需要运用虚拟机环境。最简单的例子：假设框架中需要安装 A 包，安装 A 包的前提条件是 B 包的 2.1 版本和 C 包的 1.2 版本都有，这时候你原来系统里的 B 包可能是 3.2 版本，为了装 A 包就需要降版本，但是系统里原来的 D 包需要 3.2 版本的 B 包，所以 B 包降了版本，可能导致 D 包无法使用。虚拟机环境 (Virtual Machine Environment) 是基于虚拟化技术构建的计算机模拟系统，通过软件抽象层隔离物理硬件资源，支持多系统或应用的并行运行与管理。其核心功能是模拟计算机软硬件平台或特定程序的执行环境。

虚拟机环境主要分为程序运行平台 (如 Java 虚拟机、.NET CLR)、完整计算机模拟器 (如 VirtualPC、VMware) 以及硬件系统仿真器 (如游戏机模拟器) 三类。它们通过虚拟硬盘文件 (如 VHD) 分配资源，可降低硬件成本并提升维护效率。

Conda 是一个开源环境管理系统和软件包管理系统，用于安装多个版本的软件包及其依赖项，并在它们之间轻松切换。它适用于 Linux，OS X 和 Windows，并且是为 Python 程序创建的，但可以打包和分发任何软件。Conda 可以配置隔离 Python 的虚拟环境。Conda 有两种发行版，分别是 Anaconda 和 Miniconda。Anaconda 包括常用的 pip 模块和图形化的管理工具等，Anaconda 具有丰富的工具包，软件也较大。Miniconda 则是精简版干净纯粹，安装后还可以根据需要增加安装各种工具。三者间的关系如图 1－2－2 所示。

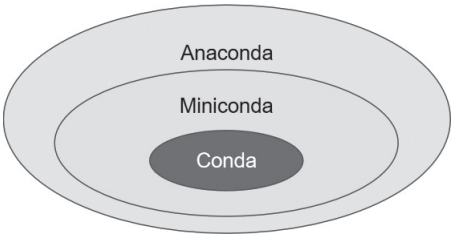


图 1–2–2　Conda 及不同版本的关系

任务实施

一、Python环境安装

Python 软件有 2.x 版和 3.x 版两个版本，这两个版本是不兼容的。由于 3.x 版越来越普及，我们的项目将采用 Python 3.12 版本，且基于 Windows 操作系统。

（1）下载 Python 安装包。打开 Python 官网，可以看到最新的稳定版本 Python。根据 CPU 的架构不同，对应的有多个不同的下载链接，CPU 的架构支持包括三种：64bit/32bit/Arm64。其中，64bit 和 32bit 是指 CPU 的位宽，Arm64 是专为基于 ARM 架构的 64 位处理器设计的，常见于移动设备、嵌入式系统以及部分服务器。

操作系统的位数可通过以下操作查看。右击“此电脑”点击“属性”，可在“系统信息”/“设备规格”中，查看“系统类型”，如图 1－2－3 所示。

图 1–2–3　查看操作系统的位数

根据已经查到的处理器信息，我们选择第一个“Download Windows installers（64bit）”点击下载（不支持 windows 7 及更老的操作系统），如图 1－2－4 所示。

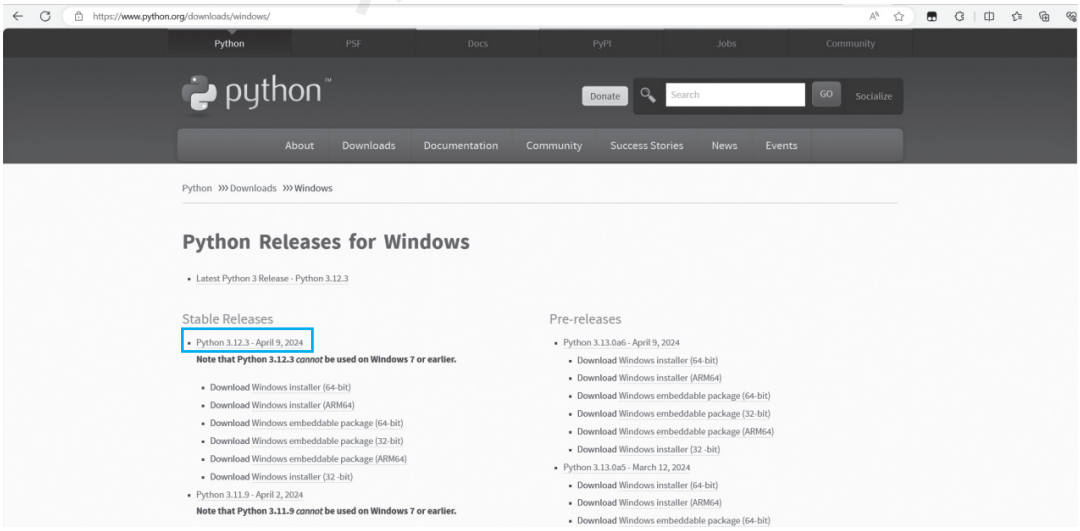


图 1–2–4　Python 下载页面

（2）打开安装包所在文件夹，浏览器会默认下载到“C:\Users\Administrator\Downloads”文件夹下，双击下载的安装包开始安装，如图 1－2－5 所示。

图 1–2–5　下载完成的 Python 安装包

（3）打开的界面中，选择“Add python.exe to PATH”，可以在安装时自动将安装路径设置到系统 PATH 中。安装时，可以选择“Customize installation”通过自定义模式更换安装目录和选项，也可以默认安装。此教材中我们选择默认安装，点击“Install Now”开始安装，如图 1－2－6、图 1－2－7 所示。

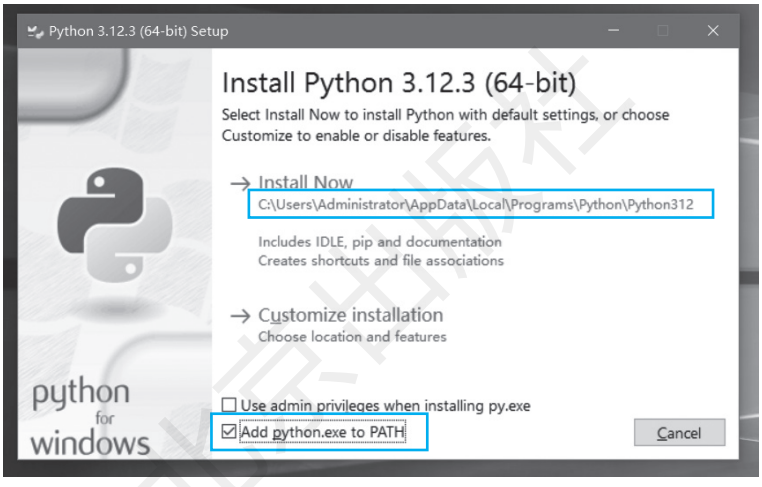


图 1–2–6　安装 Python

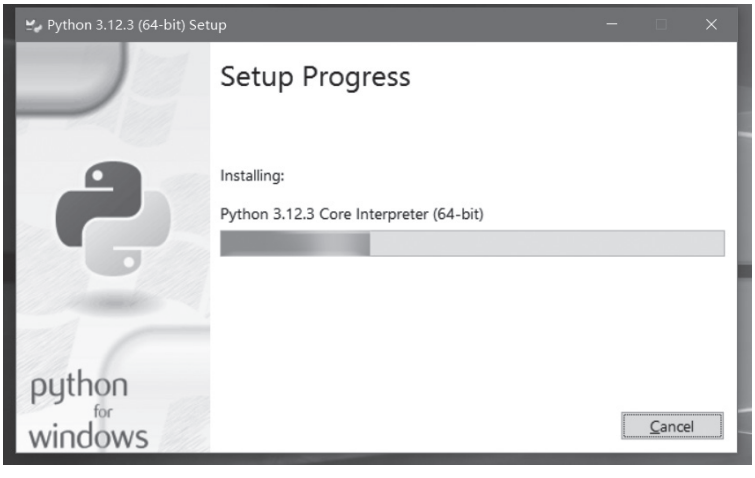


图 1–2–7　开始安装

（4）待安装界面完成后，进入提示“Setup was successful”，说明安装已经成功。点击“Disable path length limit”的按钮，然后点击 Close 关闭。“Disable path length limit”是指禁用系统的 Path 长度自动限制。更改计算机配置，以允许包括 Python 在内的程序绕过 260 个字符的“最大路径”限制，点击它然后确定权限即可，如图 1－2－8 所示。

至此，Python 环境已经安装完成，如图 1－2－9 所示。

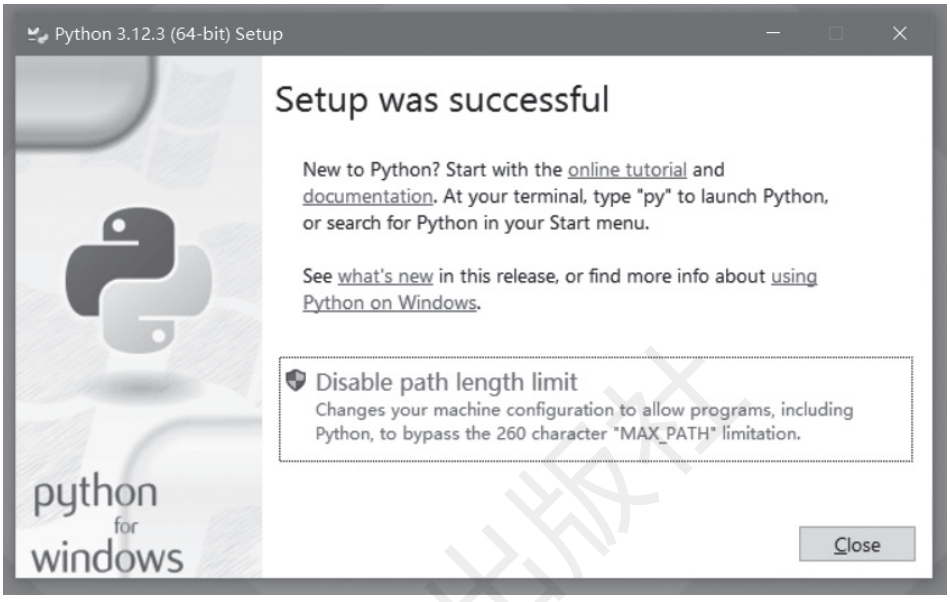


图 1–2–8　禁用系统的 Path 长度自动限制

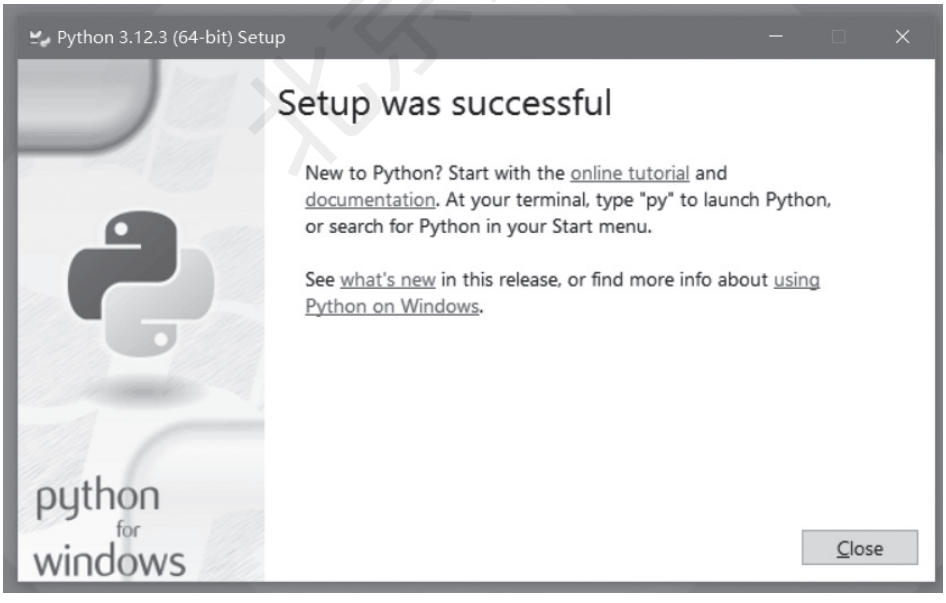


图 1–2–9　Python 安装完成

（5）使用组合键“win+R”，系统左下角弹出运行命令窗口，输入“cmd”后点确定，能启动命令提示行，如图 1－2－10 所示。

（6）在命令提示行中输入“python”，显示内容如图 1－2－11 所示，说明程序路径已经设置完成。

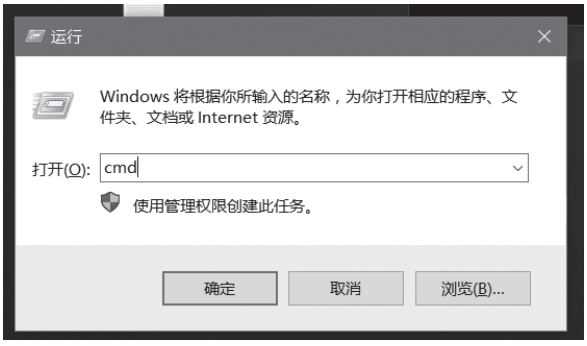


图 1–2–10　启动命令提示行

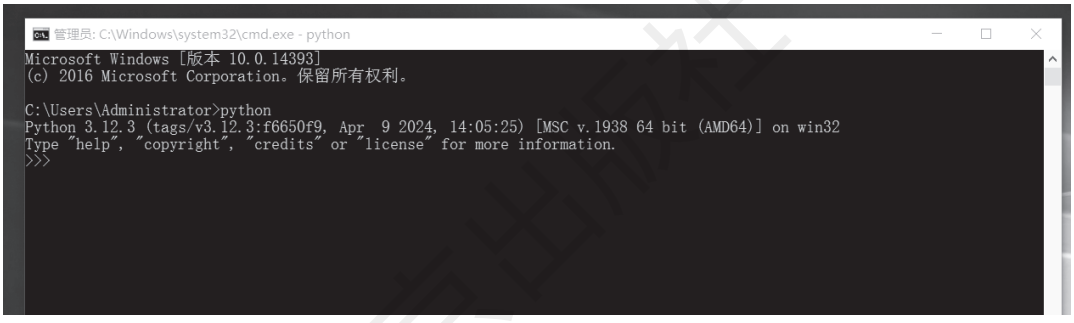
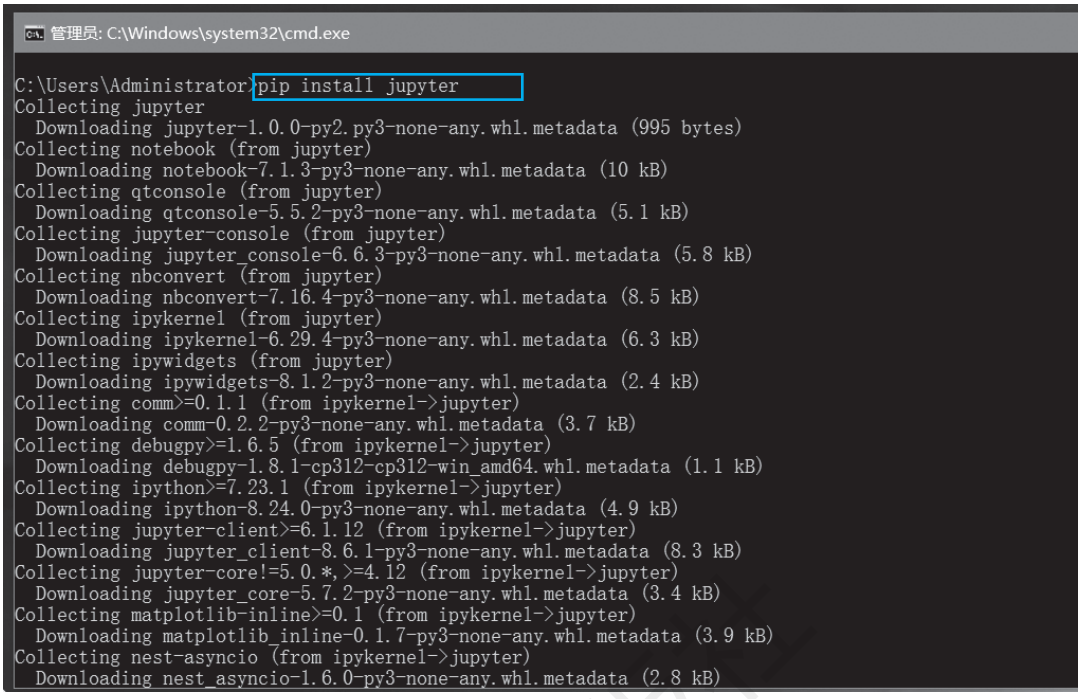


图 1–2–11　检查启动路径

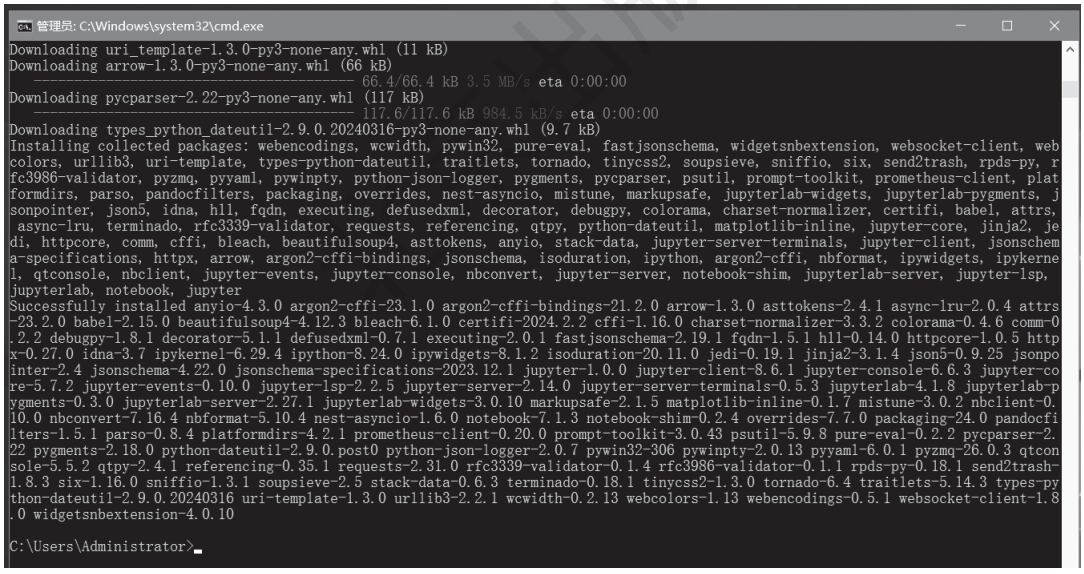
二、安装Jupyter Lab

Jupyter Lab 应用程序是一个服务器客户端应用程序，允许通过 Web 浏览器编辑和运行笔记本文档。Jupyter Lab 应用程序可以在不需要互联网访问的本地桌面上执行，也可以安装在远程服务器上并通过互联网访问。Jupyter Lab 具有简单易用、随写随看结果的特性，因而是数据分析师常用的工具。这里将具体介绍 Jupyter Lab 的使用方法以及一些技巧。

安装好 Python 程序后，在命令行执行“pip install jupyter”安装 Jupyter Lab 软件，如图 1－2－12 所示。



（a）



（b）

图 1-2-12　安装 Jupyter Lab

接下来我们介绍 Jupyter Lab 这个工具基本使用方法。

（一）启动程序

在本机的命令行界面，输入“jupyter lab”即可启动这个工具，如图 1－2－13 所示。启动后，会自动弹出浏览器窗口打开 Jupyter Lab。如果启动报错，需要检查 Path 中是否

已经设置了 Python 的路径。

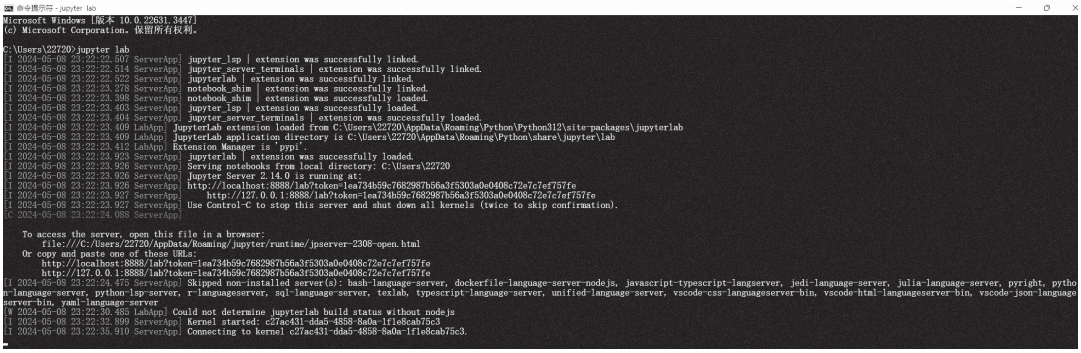


图 1–2–13　启动 Jupyter Lab

本地 Jupyter Lab 的默认 URL 为：http://localhost:8888/lab。自动浏览器打开的界面如图 1－2－14 所示。

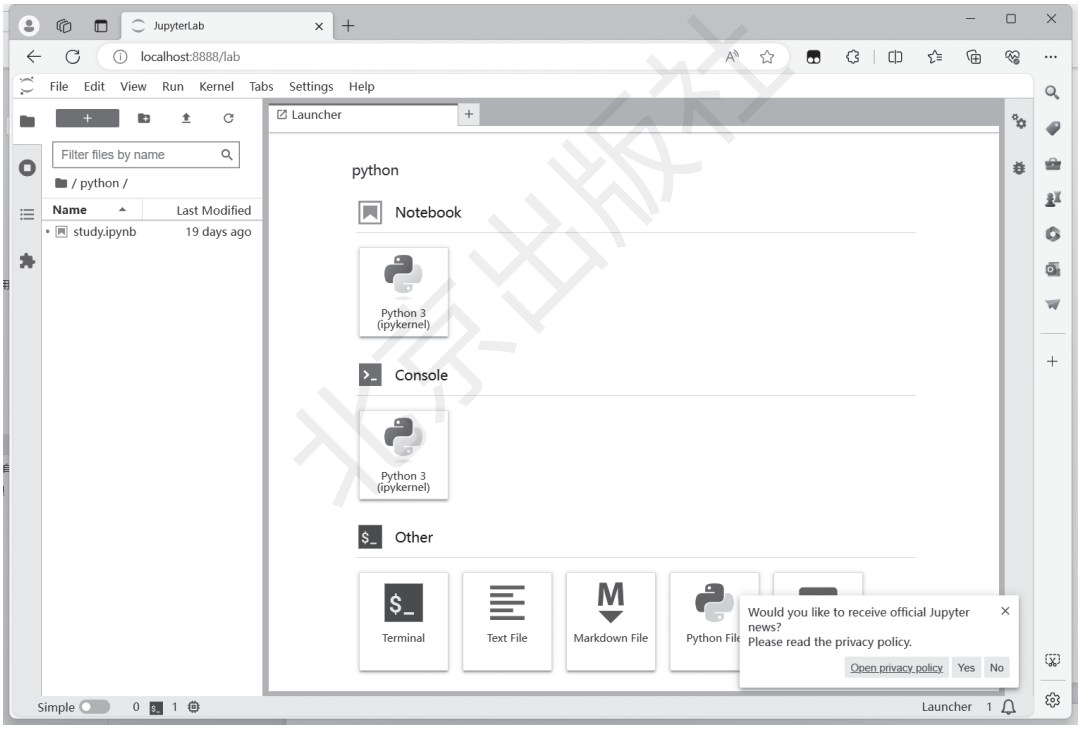


图 1–2–14　Jupyter Lab 界面

（二）启动器

右侧的选项卡称为启动器，你可以新建 Notebook、Console、Terminal 或者 Text 文本。当你创建新的 Notebook 或其他项目时，启动器会消失。如果您想新建文档，只需单击左侧圈里的“ +”按钮，如图 1－2－15 所示。

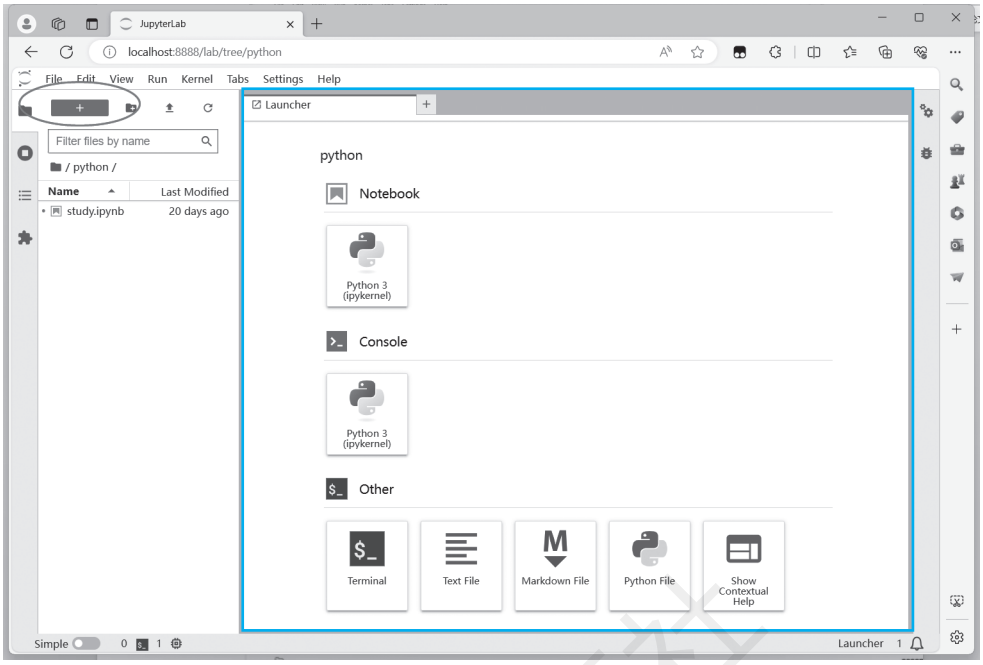


图 1–2–15　新建项目

（三）打开文档

在启动器中点击你想要打开的文档类型，即可打开相应文档，如图 1－2－16 所示。

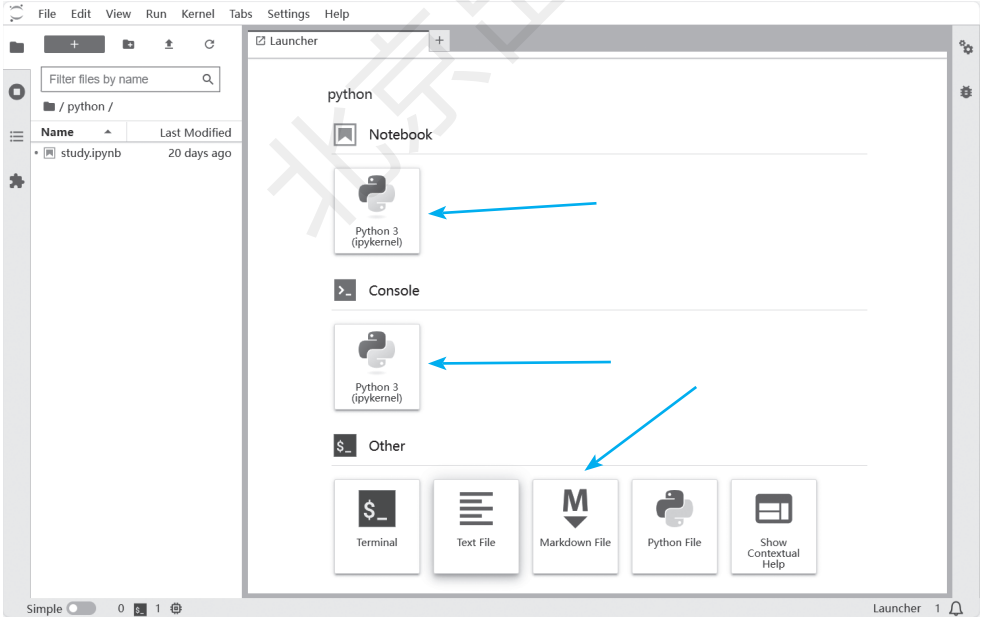


图 1–2–16　支持多种文档类型

单击左侧的“ +”按钮，可新建多种文档。右侧的标签栏中可以同时打开多种文件，如图 1－2－17 所示。

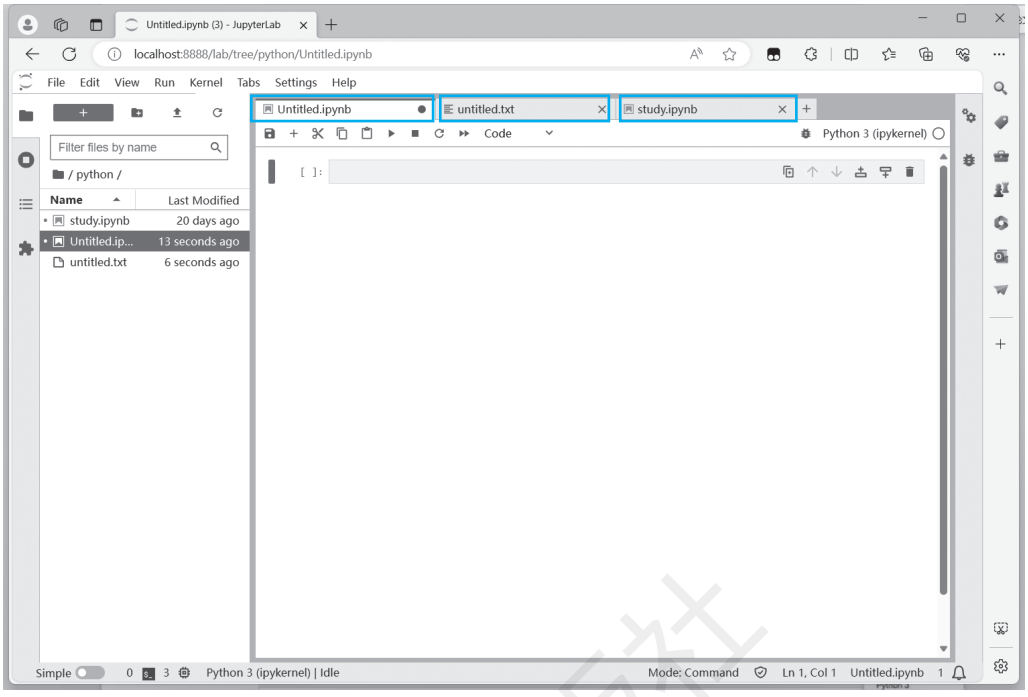


图 1–2–17　支持创建或打开多种文件

（四）支持多视图

当在一个 Notebook 里面写代码时，如果想要实时同步编辑文档并查看执行结果，可以新建该文档的多个视图，如图 1－2－18 所示。步骤：file → new view → for notebook。

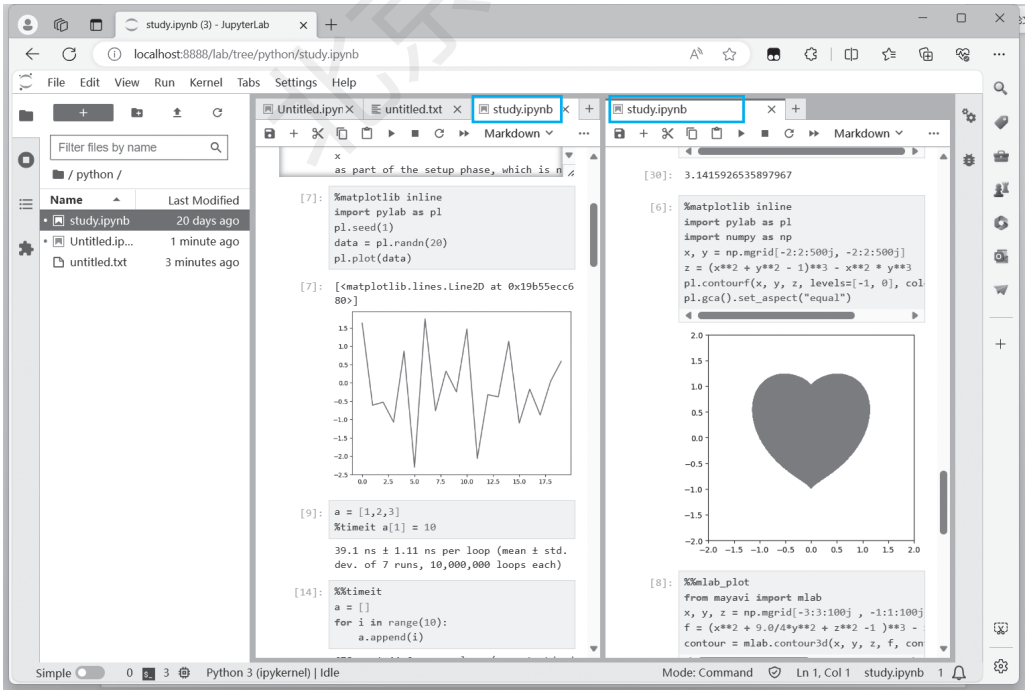


图 1–2–18　支持单文件多视图

Jupyter Lab 支持 Markdown、csv、geojson 地图、图片、pdf 等多种格式的文件，如图 1－2－19 所示。

图 1–2–19　支持多种格式

三、创建Notebook运行Python程序

（一）创建 Notebook 文件

在 Jupyter Lab 启动界面中，点击“Notebook”可以创建 Jupyter Notebook 文件，如图 1－2－20 所示。

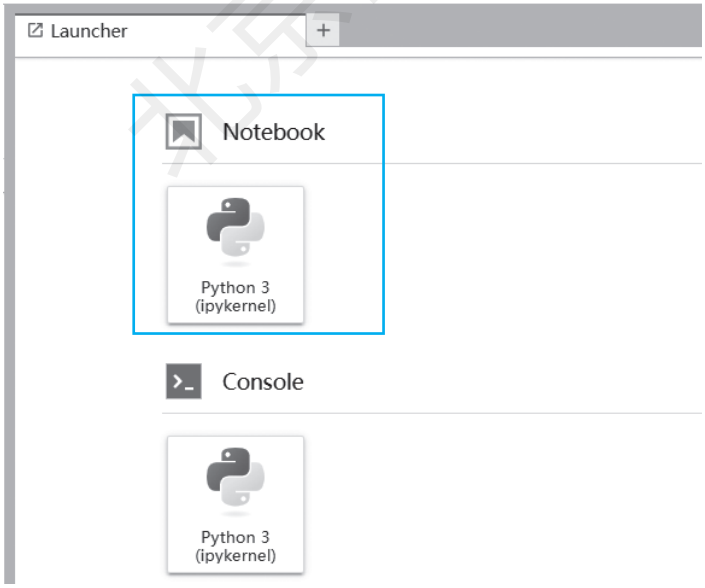


图 1–2–20　创建 Notebook 文件

（二）Notebook 文件的执行界面

Notebook 文件的执行界面及按钮介绍如图 1－2－21 所示。新建的文件名默认为

Untitled，可以右击文件标签选择“Rename Notebook”或者通过菜单栏“File → Rename Notebook”修改文件名。

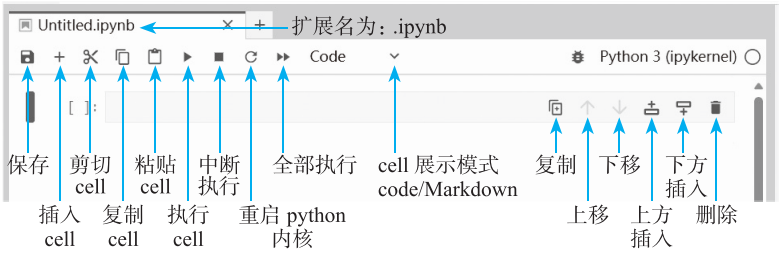


图 1–2–21　Notebook 执行界面

（三）交互模式

Notebook 界面提供了指令模式与单元编辑模式两种交互模式。在指令模式下，选定的单元格周围会显示为灰色边框，此模式主要用于管理单元格，操作涵盖单元的新增、删除、样式调整等全局任务。通过敲击回车键，用户能将界面切换至单元编辑模式，此时边框颜色转为绿色，界面上方的菜单栏右侧展示出笔形图标，这些标志着界面已进入内容编辑环节。如需回到指令模式，仅需按下 Esc 键即可实现模式切换。

（四）代码输入和运行区

工具栏下方是代码输入和运行区，支持多个 cell 以编写不同的代码并运行。在第一个单元格中，输入 Python 代码：“print（“hello world!”）”，点击运行按钮或按下“Shift+Enter”，即可看到执行结果，如图 1－2－22 所示。运行后，在其下方会出现一个新的 cell，可以继续编写其他代码。

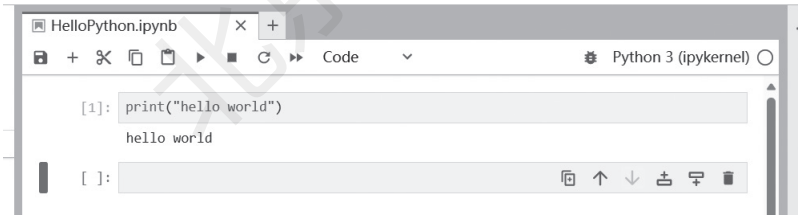


图 1–2–22　代码输入和运行区

图 1－2－23 中的一对 In Out 会话被视作一个代码单元，称为 cell。cell 左侧“[1]:”“[5]:”为自动生成的编号，表示执行已完成。某个 cell 的程序正在运行时，左侧会显示“*”符号。

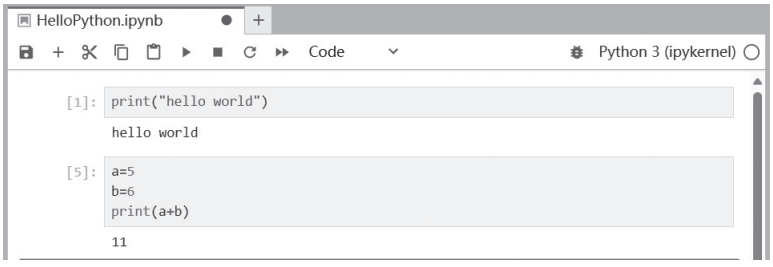


图 1–2–23　多对 In Out 会话

（五）常用快捷键操作

常用的 Notebook 快捷键操作如表 1－2－1 所示。

表 1–2–1　Notebook 快捷键

快捷键	含义
Shift+Enter	执行本单元代码，并跳转到下一单元
Ctrl+Enter	执行本单元代码，留在本单元
Y	cell 切换到 Code 模式
M	cell 切换到 Markdown 模式
双击 D	删除当前 cell
Z	回退
L	为当前 cell 加上行号
Tab 键	补全代码
Ctrl+/（Mac:CMD+/）	注释代码
Ctrl+Z （Mac:CMD+Z）	回退
Ctrl+Y （Mac:CMD+Y）	重做

四、Anaconda与虚拟环境管理

（一）Windows 操作系统安装

（1）下载 Anaconda 安装包。打开 Anaconda 官网，登录网站后，会看到登录 / 注册页面，如图 1－2－24 所示。我们选择跳过注册，进入下载页面。Anaconda 是跨平台的，有 Windows、macOS、Linux 版本，这里我们以 Windows 版本为例。点击图 1－2－25 中带有 Windows 图标的“Download”按钮，开始下载，Anaconda 安装包中含有 Python，下载页面也可以看到当前安装包包含的 Python 版本序号，如图 1－2－25 所示。

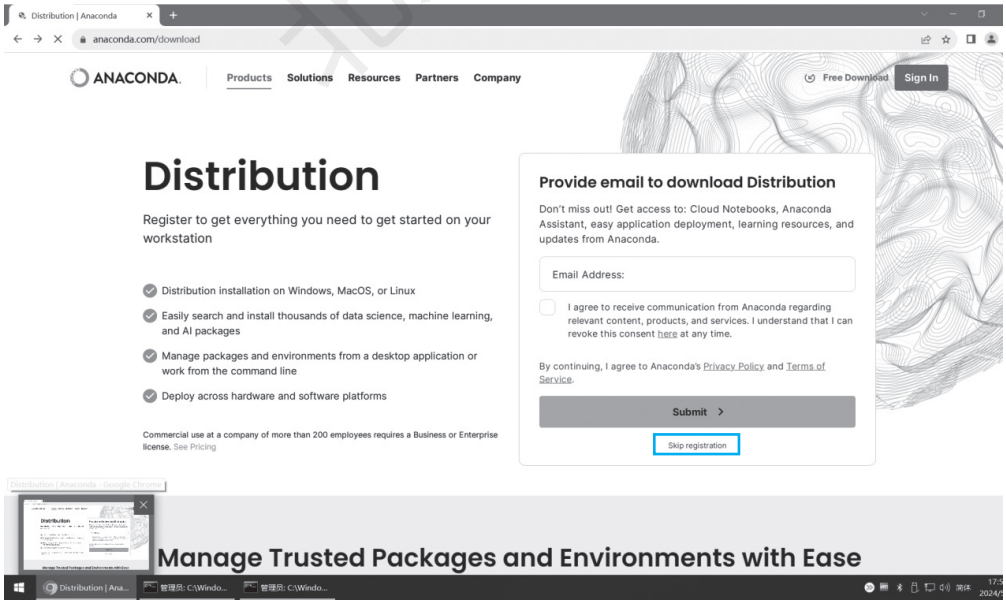


图 1–2–24　登录界面

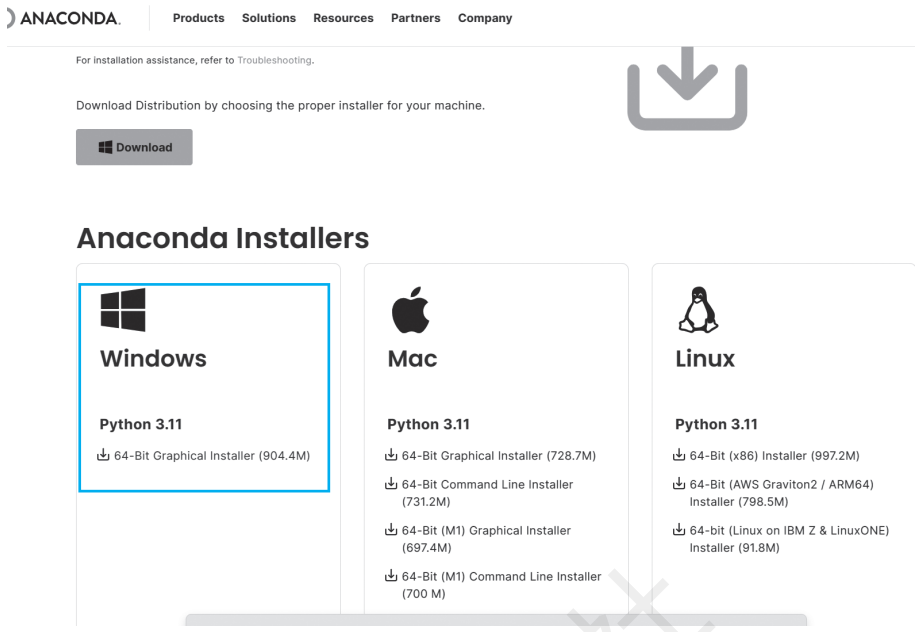


图 1–2–25　下载界面

（2）安装 Anaconda。下载完成后，会在本机下载目录看到安装包。双击下载好的文件，出现如图 1－2－26 所示的界面，点击“Next”，进入如图 1－2－27 所示界面，点击 I Agree，并进入下一界面。如果本机有多个用户，我们选择为本机的所有用户安装（All users），也可以选择仅为当前用户安装（Just me），这里我们选择默认的 All users 即可，如图 1－2－28 所示，继续点击“Next”。

如图 1－2－29 所示，软件默认是安装到 C:\ProgramData\anaconda3 文件夹下，也可以点击 Browse... ，选择想要安装的文件夹。这里要考虑个人电脑的 C 盘空间，如果空间不够大，建议安装到其他空间够大的磁盘。

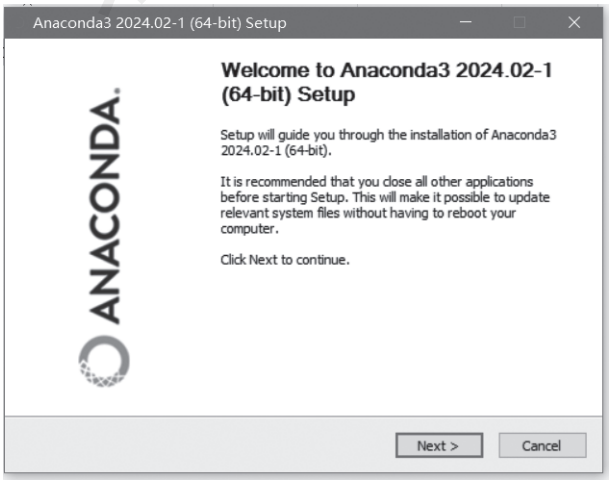


图 1–2–26　启动安装

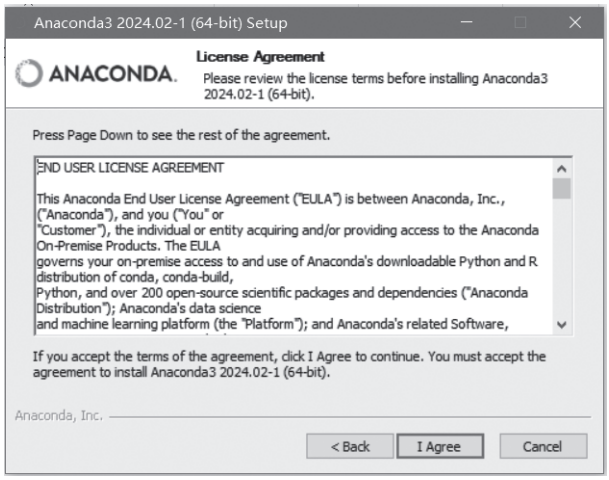


图 1–2–27　同意协议

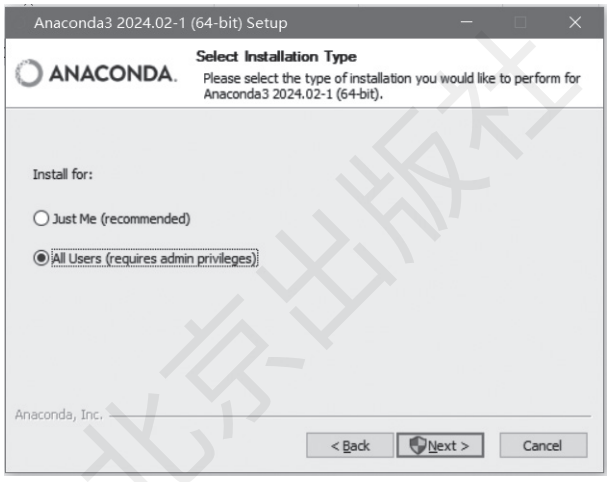


图 1–2–28　为所有用户安装

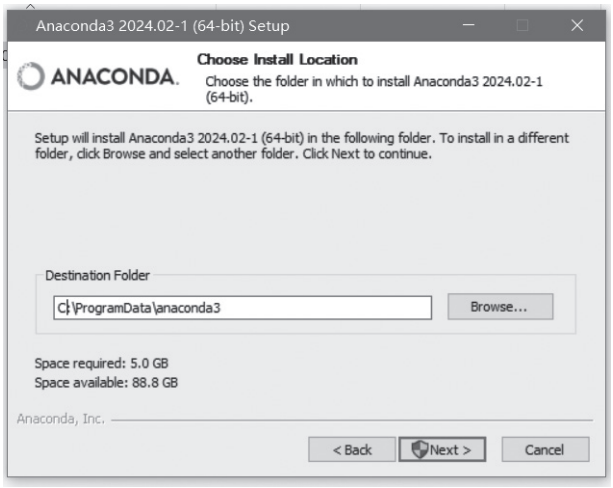


图 1–2–29　选择安装路径

点击“Next”，图 1－2－30 中是创建快捷方式、注册 Python 到本机环境、安装完毕后删除安装过程缓存文件几个选项，保持默认即可。点击“Install”开始安装。等待安装结束后，会让你选择是否安装 Vs Code，我们这里选择“暂时不安装 Vs Code”。点击“Next”直到安装成功，如图 1－2－31 所示。

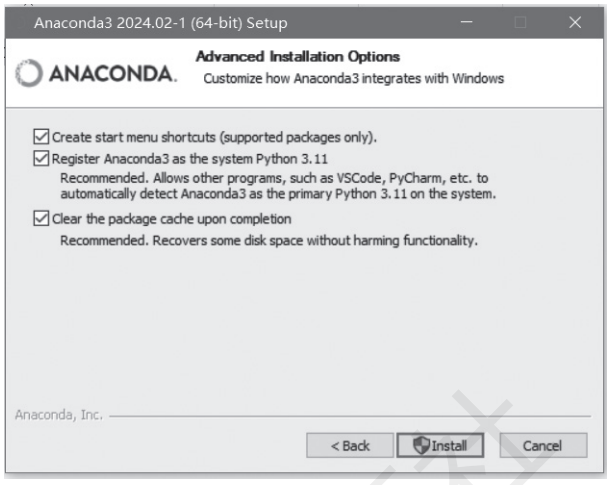


图 1–2–30　安装选项

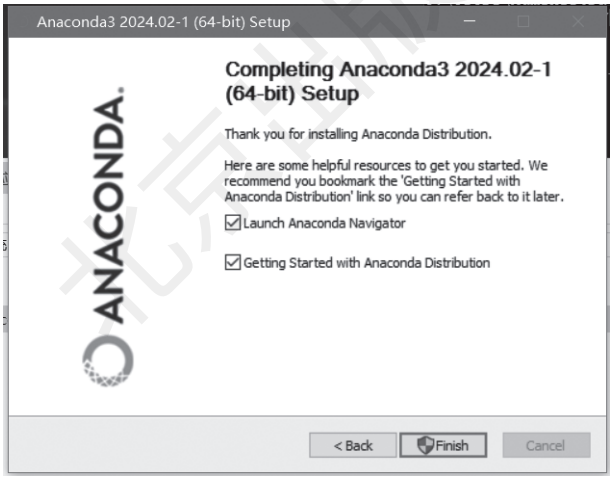


图 1–2–31　完成安装

（二）启动 Anaconda 并创建虚拟环境

安装完成后，启动 Anaconda，界面如图 1－2－32 所示。我们知道 Conda 是一个开源环境管理系统和软件包管理系统，所以在 home 界面能看到各种软件包。点击左侧菜单中的“Environments”，可以看到虚拟机环境的管理页面，如图 1－2－33 所示。安装完成后，默认会有一个 base（root）的环境，右侧可以看到这个环境内已经安装了比较多的软件，包括我们要用到的 NumPy 和 Pandas。如果没有，可以在右侧顶部选择“uninstalled”，查找这两个软件并安装，如图 1－2－34、图 1－2－35 所示。

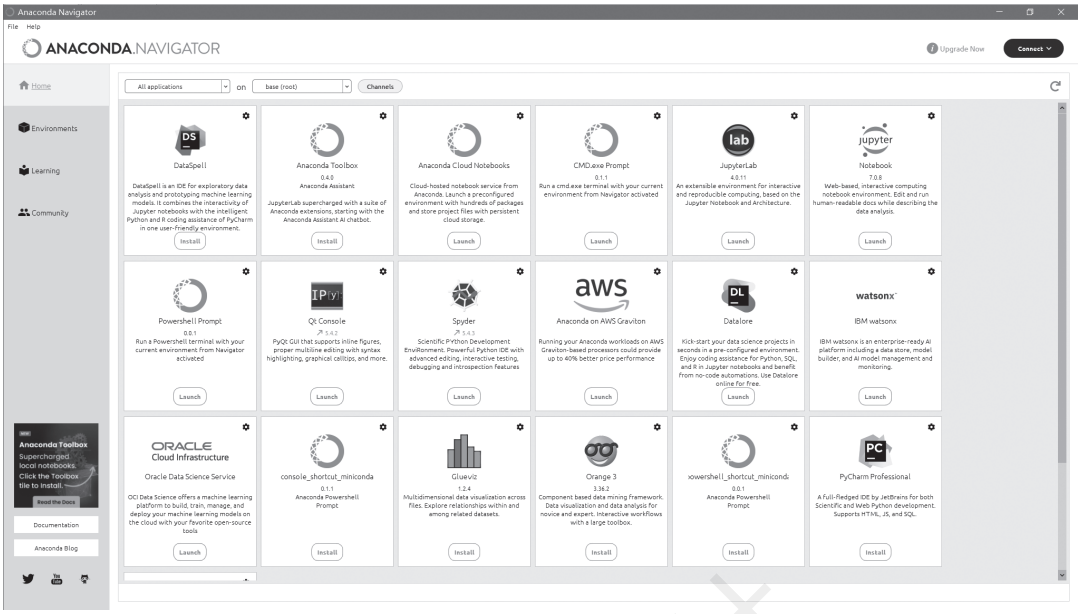


图 1–2–32　启动 Anaconda

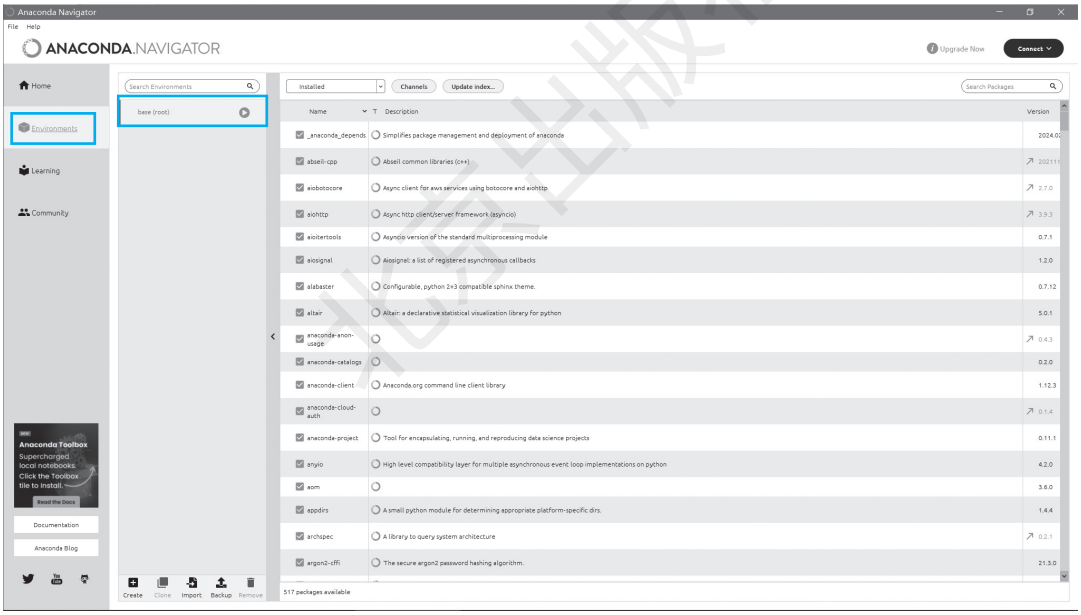


图 1–2–33　默认 base（root）环境

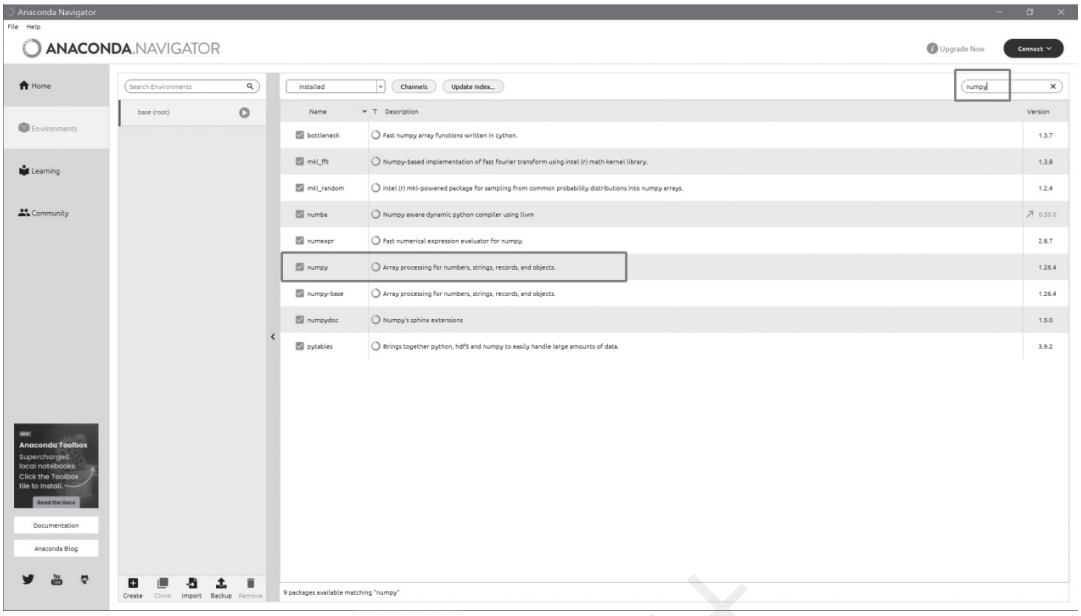


图 1–2–34　安装 NumPy

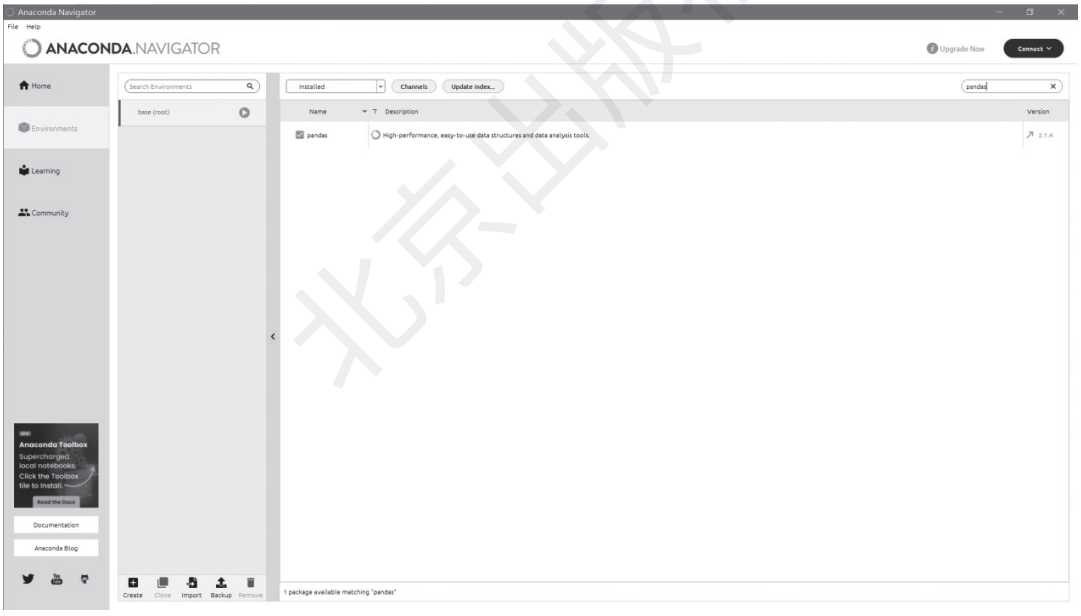


图 1–2–35　安装 Pandas

前面我们讲过，Anaconda 是 Conda 的发行版，在本书中的主要作用是可以创建多个虚拟环境，并在不同虚拟环境中安装不同版本的软件。我们在 Anaconda 界面的 Environments 页面中，点击下方的“create”按钮，就可以创建一个 base 以外的新环境，如图 1－2－36 所示。

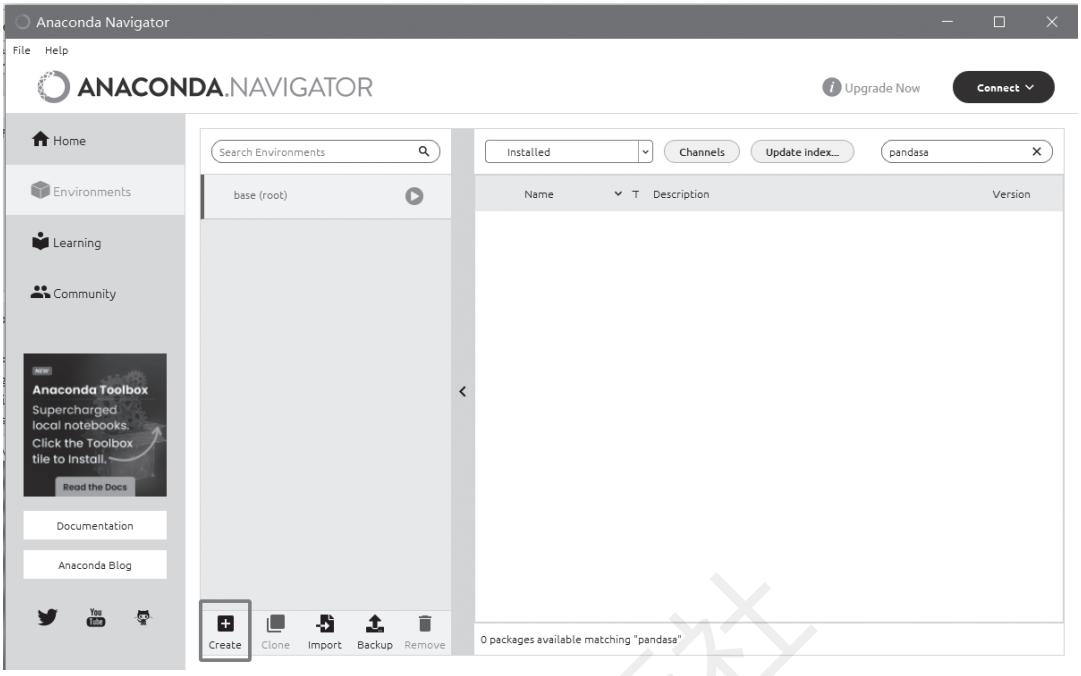


图 1–2–36　创建虚拟环境

在弹窗中输入环境名称，并选择 Python 版本，此处我们有别于 base，选择 3.10.14 版本，如图 1－2－37 所示。然后等待一段时间，即可完成 studyPy 环境的搭建。完成创建后，可看到列表中出现了两个环境，分别是“base”和“studyPy”。

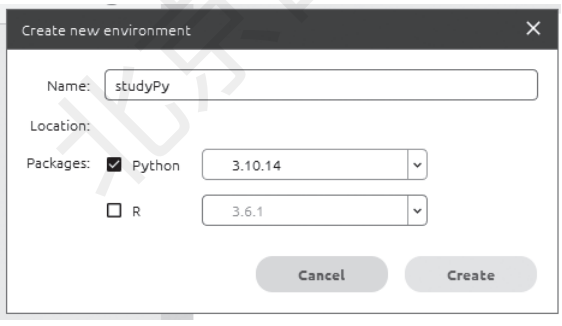


图 1–2–37　新虚拟环境选择 Python 版本

点击“studyPy”环境的启动按钮，在弹框中点击“Open Terminal”启动虚拟环境，如图 1－2－38 所示。在弹出的命令行窗口中，可以看到在路径前出现了（studyPy），说明当前在 studyPy 环境，如图 1－2－39 所示。输入“python －－version”，可看到版本为 Python 3.10.14，如图 1－2－40 所示。切换到 boot 中，再次输入“python －－version”，如图 1－2－41 所示，studyPy 虚拟环境的 Python 版本与 base 虚拟环境的 Python 版本是不同的。

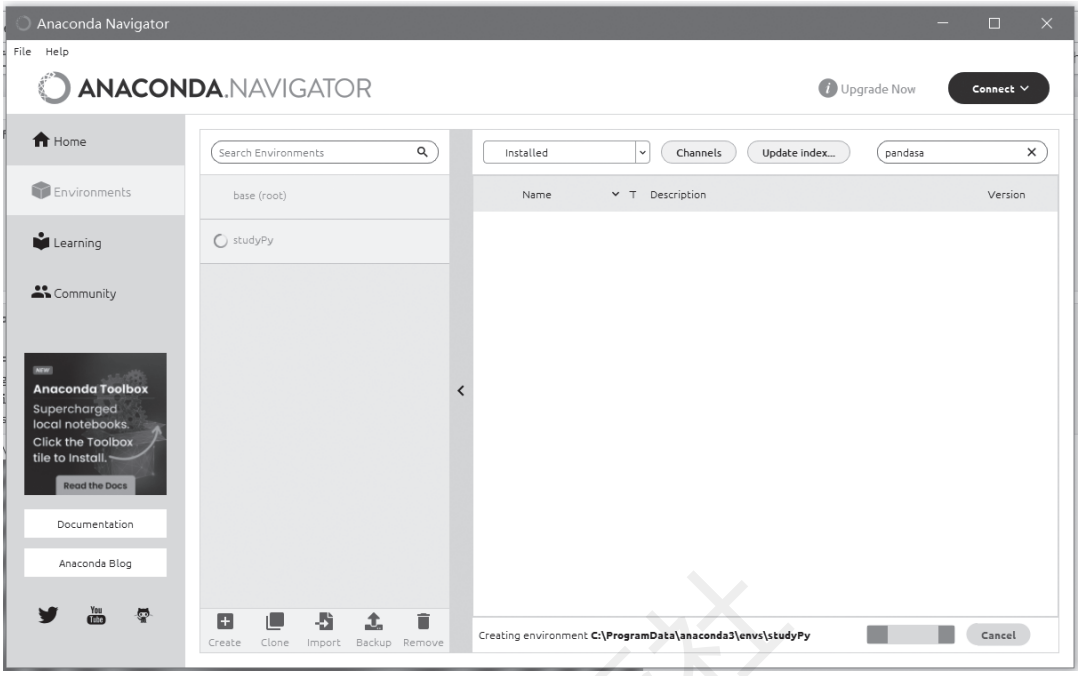


图 1–2–38 安装虚拟环境

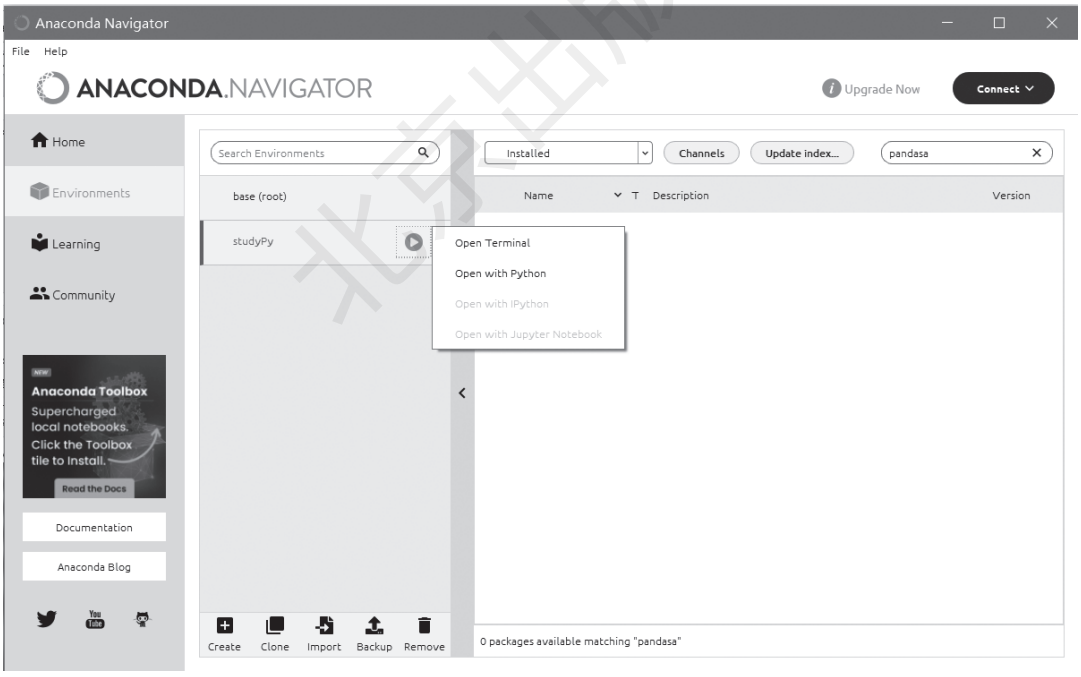


图 1–2–39 启动虚拟环境

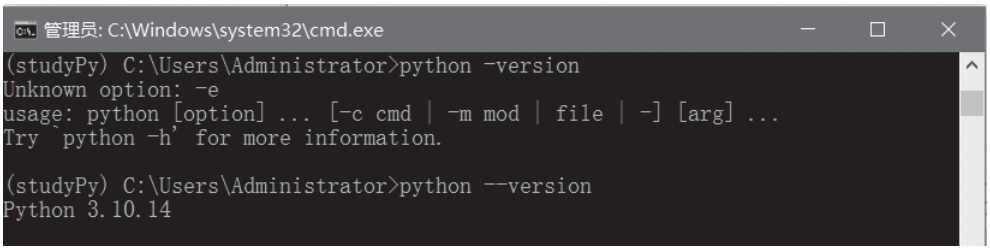


图 1–2–40　查看 studyPy 虚拟环境的 Python 版本

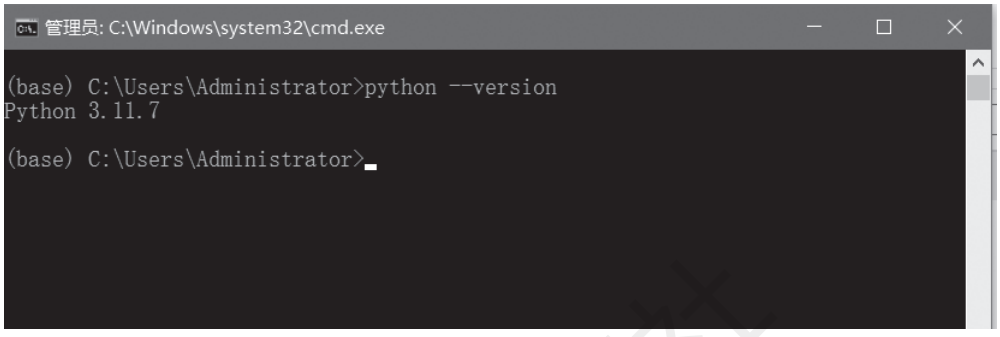


图 1–2–41　查看 base 虚拟环境的 Python 版本

思考与练习

一、选择题

1. Python 解释器的（　　）负责将源代码分解成词法标记。

A. 解释器核心

B. 编译器

C. 词法分析器

D. 解析器

2. Python 字节码通常保存（　　）的文件。

A. .py

B. .pyc

C. .pyo

D. .pyp

3.（　　）不是 Python 解释器。

A. CPython

B. IPython

C. PyPy

D. Java

4. Jupyter Notebook 支持的文档类型不包括（　　）。

A. 文本

B. 代码

C. 图像

D. 音频

5. Visual Studio Code 通过（　　）方式支持 Python 开发。

A. 内置的 Python 环境

B. Python 扩展

C. 专用的 Python 插件

D. 集成的 Python 解释器

二、填空题

1. Python 解释器中的 _____ 将 AST 转换为 Python 字节码。

2. 当 Python 程序第二次运行时，如果硬盘中存在 _____ 文件，则直接载入。

3. Jupyter Notebook 的文档可以导出为多种格式，包括 HTML、PDF 或 _____。

4. Conda 是一个开源环境管理系统和 _____ 系统，用于安装软件包及其依赖项。

三、思考题

1. 为什么在数据科学项目中，使用虚拟环境（如 Conda 环境）是一个好的实践？

2. 描述 Conda 与虚拟环境的关系，并解释为什么它对于数据科学和机器学习项目很有用。

四、编程题

1. 编写一个 Python 脚本，该脚本读取一个文本文件的内容，并打印出该文本文件中的单词列表。

2. 编写一个 Python 函数，该函数接受一个字符串作为输入，然后使用 Python 内置的 ast 模块来解析这个字符串作为 Python 表达式，并打印出表达式的 AST 节点类型。