Data Science – Khoa Học Dữ Liệu

Nguyễn Quản Bá Hồng*

Ngày 11 tháng 1 năm 2025

Tóm tắt nội dung

This text is a part of the series *Some Topics in Advanced STEM & Beyond*:
URL: https://nqbh.github.io/advanced_STEM/.
Latest version:

- *Data Science – Khoa Học Dữ Liệu.*
  PDF: URL: https://github.com/NQBH/advanced_STEM_beyond/blob/main/data_science/NQBH_data_science.pdf.
  TEX: URL: https://github.com/NQBH/advanced_STEM_beyond/blob/main/data_science/NQBH_data_science.tex.

Mục lục

1 Basic Data Science – Khoa Học Dữ Liệu Cơ Bản

Resources – Tài nguyên.

1. [McK22]. Wes McKinney. *Python for Data Analysis: Data Wrangling with pandas,* `NumPy` *& Jupyter*. [356 Amazon ratings][25357 Goodreads ratings]

   Amazon review. Get definitive handbook for manipulating, processing, cleaning, & crunching datasets in Python. Updated for Python 3.10 & `pandas` 1.4, 3e of this hand-on guide is packed with practical case studies that show you how to solve a broad set of data analysis problems effectively. Learn latest versions of pandas, NumPy, & Jupyter in process.

   Written by Wes McKinney, creator of Python `pandas` project, this book is a practical, modern introduction to data science tools in Python. Ideal for analysts new to Python & for Python programmers new to data science & scientific computing. Data files & related material are available on GitHub.

   - use Jupyter notebook & IPython shell for exploratory computing
   - Learn basic & advanced features in NumPy
   - Get started with data analysis tools in `pandas` library
   - Use flexible tools to load, clean, transform, merge, & reshape data
   - Create informative visualizations with matplotlib
   - Apply `pandas` groupby facility to slice, dice, & summarize datasets
   - Analyze & manipulative regular & irregular time series data
   - Learn how to solve real-world data analysis problems with thorough, detailed examples

   About the Author. Wes McKinney is a Nashville-based software developer & entrepreneur. After finishing his undergraduate degree in mathematics at MIT in 2007, he went on to do quantitative finance work at AQR Capital Management in Greenwich, CT. Frustrated by cumbersome data analysis tools, he learned Python & started building what would later become `pandas` project. He's now an active member of Python data community & is an advocate for Python use in data analysis, finance, & statistical computing applications.

   Wes was later cofounder & CEO of DataPad, whose technology assets & team were acquired by Cloudera in 2014. He has since become involved in big data technology, joining Project Management Committees for Apache Arrow & Apache Parquet projects in Apache Software Foundation. In 2018, he founded Ursa Labs, a not-for-profit organization focused Apache Arrow

*A Scientist & Creative Artist Wannabe. E-mail: nguyenquanbahong@gmail.com. Bến Tre City, Việt Nam.

development, in partnership with RStudio & 2 Sigma Investments. In 2021, he cofounded technology startup Voltron Data, where he currently works as Chief Technology Officer.

"With this new edition, Wes has updated his book to ensure it remains go-to resource for all things related to data analysis with Python & pandas. I cannot recommend this book highly enough." – Paul Barry, Lecturer & author of O'Reiley; *Head 1st Python*

Wes McKinney, cofounder & chief technology officer of Voltron Data, is an active member of Python data community & an advocate for Python use in data analysis, finance, & statistical computing applications. A graduate of MIT, he's also a member of project management committees for Apache Software Foundation's Apache Arrow & Apache Parquet projects.

Preface. 1e of this book was published in 2012, during a time when open source data analysis libraries for Python, especially pandas, were very new & developing rapidly. When time came to write 2e in 2016–2017, needed to update book not only for Python 3.6 (1e used Python 2.7) but also for many changes in `pandas` that had occurred over previous 5 years. 2022, there are fewer Python language changes (now at Python 3.10, with 3.11 coming out at end of 2022), but `pandas` has continued to evolve.

In 3e, goal: bring content up to date with current versions of Python, NumPy, pandas, & other projects, while also remaining relatively conservative about discussing newer Python projects having appeared in last few years. Since this book has become an important resource for many university courses & working professionals, try to avoid topics that are at risk of falling out of date within 1–2 year. That way paper copies won't be too difficult to follow in 2023 or 2024 or beyond.

A new feature of 3e: open access online version hosted on website https://wesmckinney.com/book, to serve as a resource & convenience for owners of print & digital editions. Intend to keep content reasonably up to date there, so if you paper paper book & run into sth that doesn't work properly, should check there for latest content changes.

Using Code Examples. Can find data files & related material for each chap in this book's GitHub repository at https://github.com/wesm/pydata-book, which is mirrored to Gitee (for those who cannot access GitHub) at https://gitee.com/wesmckinn/pydata-book.

This book is here to help get job done. In general, if example code is offered with this book, may use it in your programs & documentation. Do not need to contact for permission unless you're reproducing a significant portion of code. E.g., writing a program that uses several chunks of code from this book does not require permission. Selling or distributing examples from O'Reilly books does not require permission. Answering a question by citing this book & quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

Acknowledgments for 3e (2022). >1 decade since started writing 1e of this book & >15 years since originally started journey as a Python programmer. A lot has changed since then! Python has evolved from a relatively niche (ngách) language for data analysis to most popular & most widely used language powering plurality (if not majority!) of DS, ML, & AI work.

Have not been an active contributor to `pandas` open source project since 2013, but its worldwide developer community has continued to thrive, serving as a model of community-centric open source software development. Many "next-generation" Python projects that deal with tabular data are modeling their user interfaces directly after pandas, so project has proved to have an enduring influence on future trajectory of Python DS ecosystem.

Acknowledgments for 2e (2017). 5 years almost to day since completed manuscript for this book's 1e in Jul 2012. A lot has changed. Python community has grown immensely, & ecosystem of open source software around it has flourished.

This new edition of book would not exist if for tireless efforts of `pandas` core developers, who have grown project & its user community into 1 of cornerstones of Python DS ecosystem.

With open source software projects more thinly resourced than ever relative to size of user bases, it is becoming increasingly important for businesses to provide support for development of key open source projects. It's the right thing to do.

- 1. Preliminaries.
  - 1.1. What Is This Book About? This book is concerned with nuts & bolts of manipulating, processing, cleaning, & crunching (nhai giòn tan) data in Python. Goal: offer a guide to parts of Python programming language & its data-oriented library ecosystem & tools that will equip you to become an effective data analyst. While "data analysis" is in title of book, focus is specifically on Python programming, libraries, & tools as opposed to data analysis methodology. This is Python programming you need *for* data analysis.
    Sometime after Wes originally published this book in 2012, people started using term *data science* as an umbrella description for everything from simple descriptive statistics to more advanced statistical analysis & ML. Python open source ecosystem for doing data analysis (or DS) has also expanded significantly since then. There are now many other books which focus specifically on these more advanced methodologies. Hope: this book serves as adequate preparation to enable you to move on to a more domain-specific resource.

    Remark 1. *Some might characterize much of content of book as "data manipulation" as opposed to "data analysis." Also use terms* wrangling *or* munging *to refer to data manipulation.*

    What Kinds of Data? Primary focus is on *structured data*, a deliberately vague term that encompasses many different common forms of data, e.g.:
    - Tabular or spreadsheet-like data in which each column may be a different type (string, numeric, date, or otherwise). This includes most kinds of data commonly stored in relational databases or tab- or comma-delimited text files.

* Multidimensional arrays (matrices).
* Multiple tables of data interrelated by key columns (what would be primary or foreign keys for a SQL user).
* Evenly or unevenly spaced time series.

This is by no means a complete list. Even though it may not always be obvious, a large percentage of datasets can be transformed into a structured form that is more suitable for analysis & modeling. If not, it may be possible to extract features from a dataset into a structured form. E.g., a collection of news articles could be processed into a word frequency table, which could then be used to perform sentiment analysis.

Most users of spreadsheet programs like Microsoft Excel, perhaps most widely used data analysis tool in the world, will not be strangers to these kinds of data.

- 1.2. Why Python for Data Analysis? For many people, Python programming language has strong appeal. Since its 1st appearance in 1991, Python has become 1 of most popular interpreted programming languages, along with Perl, Ruby, & others. Python & Ruby have become especially popular since 2005 or so for building websites using their numerous web frameworks, like Rails (Ruby) & Django (Python). Such languages are often called *scripting* languages, as they can be used to quickly write small programs, or *scripts* to automate other tasks. I don't like term "scripting languages," as it carries a connotation that they cannot be used for building serious software. Among interpreted languages, for various historical & cultural reasons, Python has developed a large & active scientific computing & data analysis community. In last 20 years, Python has gone from a bleeding-edge or "at your own risk" scientific computing language to 1 of most important languages for DS, ML, & general software development in academia & industry.

  For data analysis & interactive computing & data visualization, Python will inevitably draw comparisons with other open source & commercial programming languages & tools in wide use, e.g. R, MATLAB, SAS, Stata, & others. In recent years, Python's improved open source libraries (e.g. `pandas` & scikit-learn) have made it a popular choice for data analysis tasks. Combined with Python's overall strength for general-purpose software engineering, it is an excellent option as a primary language for building data applications.

  * Python as Glue. Part of Python's success in scientific computing: ease of integrating C, C++, & FORTRAN code - -1 phần thành công của Python trong điện toán khoa học: dễ dàng tích hợp mã C, C++, & FORTRAN. Most modern computing environments share a similar set of legacy FORTRAN & C libraries for doing linear algebra, optimization, integration, fast Fourier transforms, & other such algorithms. Same story has held true for many companies & national labs that have used Python to glue together decades' worth of legacy software.

    Many programs consist of small portions of code where most of time is spent, with large amounts of "glue code" that doesn't run often. In many cases, execution time of glue code is significant; effort is most fruitfully invested in optimizing computational bottlenecks, sometimes by moving code to a lower-level language like C.
  * Solving "2-Language" Problem. In many organizations, common to research, prototype, & test new ideas using a more specialized computing language like SAS or R & then later port those ideas to be part of a larger production system written in, say, Java, C#, or C++. What people are increasingly finding: Python is a suitable language not only for doing research & prototyping but also for building production systems. *Why maintain 2 development environments when one will suffice?* Believe more & more companies will go down this path, as there are often significant organizational benefits to having both researchers & software engineers using same set of programming tools.

    Over last decade some new approaches to solving "2-language" problem have appeared, e.g. Julia programming language. Getting most out of Python in many cases *will* require programming in a low-level language like C or C++ & creating Python bindings to that code. I.e., "just-in-time" (JIT) compiler technology provided by libraries like Numba have provided a way to achieve excellent performance in many computational algorithms without having to leave Python programming environment.
  * **Why Not Python?** While Python is an excellent environment for building many kinds of analytical applications & general-purpose systems, there are a number of uses for which Python may be less suitable.

    As Python is an interpreted programming language, in general most Python code will run substantially slower than code written in a compiled language like Java or C++. As *programmer time* is often more valuable than *CPU time*, many are happy to make this trade-off. However, in an application with very low latency or demanding resource utilization requirements (e.g., a high-frequency trading systems), time spent programming in a lower-level (but also lower-productivity) language like C++ to achieve maximum possible performance might be time well spent.

    – Vì Python là ngôn ngữ lập trình được thông dịch, nhìn chung hầu hết mã Python sẽ chạy chậm hơn đáng kể so với mã được viết bằng ngôn ngữ biên dịch như Java hoặc C++. Vì *thời gian lập trình* thường có giá trị hơn *thời gian CPU*, nhiều người vui vẻ chấp nhận sự đánh đổi này. Tuy nhiên, trong một ứng dụng có độ trễ rất thấp hoặc yêu cầu sử dụng tài nguyên khắt khe (ví dụ: hệ thống giao dịch tần suất cao), thời gian dành cho việc lập trình bằng ngôn ngữ cấp thấp hơn (nhưng cũng có năng suất thấp hơn) như C++ để đạt được hiệu suất tối đa có thể là thời gian được sử dụng hợp lý.

    Python can be a challenging language for building highly concurrent, multithreaded applications, particularly applications with many CPU-bound threads. Reason for this: it has what is known as *global interpreter lock* (GIL), a mechanism that prevents interpreter from executing > 1 Python instruction at a time. Technical reasons for why GIL exists are beyond scope of this book. While it is true that in many big data processing applications, a cluster of computers may be required to process a dataset in a reasonable amount of time, there are still situations where a single-process, multithreaded system is desirable.

    This is not to say: Python cannot execute truly multithreaded, parallel code. Python C extensions that use native mutithreading (in C or C++) can run code in parallel without being impacted by GIL, as long as they do not need to

regularly interact with Python objects.

- 1.3. Essential Python Libraries. For those who are less familiar with Python data ecosystem & libraries used throughout book, a brief overview of some of them:
  - NumPy. NumPy, short for Numerical Python, has long been a cornerstone of numerical computing in Python. It provides data structures, algorithms, & library glue needed for most scientific applications involving numerical data in Python. `NumPy` contains, among other things:
    - A fast & efficient multidimensional array object `ndarray`
    - Functions for performing element-wise computations with arrays or mathematical operations between arrays
    - Tools for reading & writing array-based datasets to disk
    - Linear algebra operations, Fourier transform, & random number generation
    - A mature C API to enable Python extensions & native C or C++ code to access NumPy's data structures & computational facilities

    Beyond fast array-processing capabilities that `NumPy` adds to Python, 1 of its primary uses in data analysis is as a container for data to be passed between algorithms & libraries. For numerical data, `NumPy` arrays are more efficient for storing & manipulating data than the other built-in Python data structures. Also, libraries written in a lower-level language, e.g. C or FORTRAN, can operate on data stored in a `NumPy` array without copying data into some other memory representation. Thus, many numerical computing tools for Python either assume `NumPy` arrays as a primary data structure or else target interoperability with NumPy.
  - `pandas`. pandas provides high-level data structures & functions designed to make working with structured or tabular data intuitive & flexible. Since its emergence in 2010, it has helped enable Python to be a powerful & productive data analysis environment. Primary objects in `pandas` that will be used in this book are `DataFrame`, a tabular, column-oriented data structure with both row & column labels, & `Series`, a 1D labeled array object.
    `pandas` blends array-computing ideas of `NumPy` with kinds of data manipulation capabilities found in spreadsheets & relationship databases (e.g. SQL). It provides convenient indexing functionality to enable you to reshape, slice & dice, perform aggregations (thực hiện tổng hợp), & select subsets of data. Since data manipulation, preparation, & cleaning are such important skills in data analysis, `pandas` is 1 of primary focuses of this book.
    As a bit of background, McKinney started building `pandas` in early 2008 during his tenure at AQR Capital Management, a quantitative investment management firm. At time, McKinney had a distinct set of requirements that were not addressed by any single tool at his disposal:
    - Data structures with labeled axes supporting automatic or explicit data alignment – this prevents common errors resulting from misaligned data & working with differently indexed data coming from different sources
    - Integrated time series functionality
    - Same data structures handle both time series data & non-time series data
    - Arithmetic operations & reductions that preserve metadata
    - Flexible handling of missing data
    - Merge & other relational operations found in popular databases (e.g., SQL-based)

    Wanted to be able to do all of these things in 1 place, preferably in a language well suited to general-purpose software development. Python was a good candidate language for this, but at that time an integrated set of data structures & tools providing this functionality did not exist. As a result of having been built initially to solve finance & business analytics problems, `pandas` features especially deep time series functionality & tools well suited for working with time-indexed data generated by business processes.
    McKinney spent a large part of 2011 & 2012 expanding pandas's capabilities with some of former AQR colleagues, Adam Klein, Chang She. In 2013, stopped being as involved in day-to-day project development, & `pandas` has since become a fully community-owned & community-maintained project with well $>$ 2000 unique contributors around world.
    For users of R language for statistical computing, `DataFrame` name will be familiar, as object was named after similar R `data.frame` object. Unlike Python, data frames are built into R programming language & its standard library. As a result, many features found in `pandas` are typically either part of R core implementation or provided by add-on packages.
    pandas name itself is derived from *panel data*, an econometrics term for multidimensional structured datasets, & a play on phrase *Python data analysis*.
  - matplotlib. matplotlib is most popular Python library for producing plots & other 2D data visualizations. It was originally created by John D. Hunter & is now maintained by a large team of developers. It is designed for creating plots suitable for publication. While there are other visualization libraries available to Python programmers, matplotlib is still widely used & integrates reasonably well with rest of ecosystem. Think it is a safe choice as a default visualization tool.
  - `IPython & Jupyter`. IPython project began in 2001 as Fernando Pérez's side project to make a better interactive Python interpreter. Over subsequent 20 years it has become 1 of most important tools in modern Python data stack. While it does not provide any computational or data analytical tools by itself, IPython is designed for both interactive computing & software development work. It encourages an *execute-explore* workflow instead of typical *edit-compile-run* workflow of many other programming languages. It also provides integrated access to OS's shell & filesystem; this

reduces need to switch between a terminal window & a Python session in many cases. Since much of data analysis coding involves exploration, trial & error, & iteration, IPython can help you get job done faster.

In 2014, Fernando & IPython team announced Jupyter project, a broader initiative to design language-agnostic interactive computing tools. IPython web notebook became Jupyter notebook, with support now for > 40 programming languages. IPython system can now be used as a *kernel* (a programming language mode) for using Python with Jupyter. IPython itself has become a component of much broader Jupyter open source project, which provides a productive environment for interactive & exploratory computing. Its oldest & simplest "mode" is as an enhanced Python shell designed to accelerate writing, testing, & debugging of Python code. You can also use IPython system through Jupyter notebook.

Jupyter notebook system also allows you to author content in Markdown & HTML, providing you a means to create rich documents with code & text.

McKinney personally uses IPython & Jupyter regularly in Python work, whether running, debugging, or testing code. In accompanying book materials on GitHub, you will find Jupyter notebooks containing all code examples from each chap. If cannot access GitHub where you are, can try mirror on Gitee.

* `SciPy`. SciPy is a collection of packages addressing a number of foundational problems in scientific computing. Some of tools it contains in its various modules:
  - `scipy.integrate`: Numerical integration routines & differential equation solvers
  - `scipy.linalg`: Linear algebra routines & matrix decompositions extending beyond those provided in `numpy.linalg`
  - `scipy.optimize`: Function optimizers (minimizers) & root finding algorithms
  - `scipy.signal`: Signal processing tools
  - `scipy.sparse`: Sparse matrices & sparse linear system solvers
  - `scipy.special`: Wrapper around SPECFUN, a FORTRAN library implementing many common mathematical functions, e.g. `gamma` function
  - `scipy.stats`: Standard continuous & discrete probability distributions (density functions, samplers, continuous distribution functions), various statistical tests, & more descriptive statistics

  Together, `NumPy` & SciPy form a reasonably complete & mature computational foundation for many traditional scientific computing applications.
* `scikit-learn`: Since project's inception in 2007, scikit-learn has become premier general-purpose ML toolkit for Python programmers. As of this writing, > 2000 different individuals have contributed code to project. It includes submodels for such models as:
  - Classification: SVM, nearest neighbors, random forest, logistic regression, etc.
  - Regression: Lasso, ridge regression, etc.
  - Clustering: k-means, spectral clustering, etc.
  - Dimensionality reduction: PCA, feature selection, matrix factorization, etc.
  - Model selection: Grid search, cross-validation, metrics
  - Preprocessing: Feature extraction, normalization

  Along with pandas, statsmodels, & IPython, scikit-learn has been critical for enabling Python to be a productive DS programming language. While I won't be able to include a comprehensive guide to scikit-learn in this book, I will give a brief introduction to some of its models & how to use them with other tools presented in book.
* `statsmodels` is a statistical analysis package that was seeded by work from Stanford University statistics professor Jonathan Taylor, who implemented a number of regression analysis models popular in R programming language. Skipper Seabold & Josef Perktold formally created new statsmodels project in 2010 & since then have grown project to a critical mass of engaged users & contributors. Nathaniel Smith developed Patsy project, which provides a formula or model specification framework for statsmodels inspired by R's formula system.

  Compared with scikit-learn, statsmodels contains algorithms for classical (primarily frequentist) statistics & econometrics. This includes such submodules as:
  - Regression models: linear regression, generalized linear models, robust linear models, linear mixed effect models, etc.
  - Analysis of variance (ANOVA)
  - Time series analysis: AR, ARMA, ARIMA, VAR, & other models
  - Nonparametric methods: Kernel density estimation, kernel regression
  - Visualization of statistical model results

  statsmodels is more focused on statistical inference, providing uncertainty estimates & p-values for parameters. scikit-learn, by contrast, is more prediction focused.

  As with scikit-learn, give a brief introduction to statsmodels & how to use it with `NumPy` & pandas.
* `Other Packages`. In 2022, there are many other Python libraries which might be discussed in a book about DS. This includes some newer projects like TensorFlow or PyTorch, which have become popular for ML or AI work. Now that there are other books out there that focus more specifically on those projects, recommend using this book to build a foundation in general-purpose Python data wrangling. Then, you should be well prepared to move on to a more advanced resource that may assume a certain level of expertise.

- 1.4. Installation & Setup. Since everyone uses Python for different applications, there is no single solution for setting up Python & obtaining necessary add-on packages. Many readers will not have a complete Python development environment suitable for following along with this book, so here give detailed instructions to get set up on each OS. Use Miniconda, a minimal installation of conda package manager, along with conda-forge, a community-maintained software distribution based on conda. This book uses Python 3.10 throughout, but if read in future, welcome to install a newer version of Python.
  If for some reason these instructions become out-of-date by time you are reading this, can check website for book which I will endeavor to keep up to date with latest installation instructions.
  * `Miniconda on Windows.`
  * `GNU/Linux.` Linux details will vary a bit depending on Linux distribution type, but here give details for such distributions as Debian, Ubuntu, CentOS, & Fedora. Setup is similar to macOS with exception of how Miniconda is installed. Most readers will want to download default 64-bit installer file, which is for x86 architecture (but possible in future more users will have aarch64-based Linux machines). Installer is a shell script that must be executed in terminal. Then have a file named sth similar to `Miniconda3-latest-Linux-x86_64.sh`. To install it, execute this script with `bash`:

    ```
    $ bash Miniconda3-latest-Linux-x86_64.sh
    ```

    Remark 2. *Some Linux distributions have all required Python packages (although outdated versions, in some cases) in their package managers & can be installed using a tool like* `apt`*. Setup described here uses Miniconda, as it's both easily reproducible across distributions & simpler to upgrade packages to their latest versions.*

    Will have a choice of where to put Miniconda files. Recommend installing files in default location in home directory; e.g., `/home/$USER/miniconda` (with your username, naturally).
    Installer will ask if wish to modify shell scripts to automatically activate Miniconda. Recommend doing this (select "yes") as a matter of convenience.
    After completing installation, start a new terminal process & verify that you are picking up new Miniconda installation:

    ```
    (base) nqbh@nqbh-dell:~/advanced_STEM_beyond/data_science$ python
    Python 3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:27:36) [GCC 11.2.0] on linux
    Type "help", "copyright", "credits" or "license" for more information.
    >>>
    ```

    To exit Python shell, type `exit()` & press Enter or press Ctrl-D.
  * Miniconda on macOS.
  * Installing Necessary Packages. Have set up Miniconda on system, time to install main packages will be using in this book. 1st step: configure `conda-forge` as default package channel by running commands in a shell:

    ```
    (base) $ conda config --add channels conda-forge
    (base) $ conda config --set channel_priority strict
    ```

    Now create a new conda "environment" with `conda create` command using Python 3.10:

    ```
    (base) $ conda create -y -n pydata-book python=3.10

    (base) nqbh@nqbh-dell:~$ conda create -y -n pydata-book python=3.12.7
    Retrieving notices: done
    Channels:
    - conda-forge
    - defaults
    Platform: linux-64
    Collecting package metadata (repodata.json): done
    Solving environment: done

    ## Package Plan ##

    environment location: /home/nqbh/anaconda3/envs/pydata-book

    added / updated specs:
    - python=3.12.7


    The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    _libgcc_mutex-0.1          |      conda_forge           3 KB  conda-forge
    _openmp_mutex-4.5          |            2_gnu          23 KB  conda-forge
    ```

```
bzip2-1.0.8                |       h4bc722e_7         247 KB  conda-forge
ca-certificates-2024.12.14 |       hbcca054_0         153 KB  conda-forge
ld_impl_linux-64-2.43      |       h712a8e2_2         654 KB  conda-forge
libexpat-2.6.4             |       h5888daf_0          72 KB  conda-forge
libffi-3.4.2               |       h7f98852_5          57 KB  conda-forge
libgcc-14.2.0              |       h77fa898_1         829 KB  conda-forge
libgcc-ng-14.2.0           |       h69a702a_1          53 KB  conda-forge
libgomp-14.2.0             |       h77fa898_1         450 KB  conda-forge
liblzma-5.6.3              |       hb9d3cd8_1         109 KB  conda-forge
liblzma-devel-5.6.3        |       hb9d3cd8_1         368 KB  conda-forge
libnsl-2.0.1               |       hd590300_0          33 KB  conda-forge
libsqlite-3.47.2           |       hee588c1_0         853 KB  conda-forge
libuuid-2.38.1             |       h0b41bf4_0          33 KB  conda-forge
libxcrypt-4.4.36           |       hd590300_1          98 KB  conda-forge
libzlib-1.3.1              |       hb9d3cd8_2          60 KB  conda-forge
ncurses-6.5                |       he02047a_1         868 KB  conda-forge
openssl-3.4.0              |       h7b32b05_1         2.8 MB  conda-forge
pip-24.3.1                 |     pyh8b19718_2         1.2 MB  conda-forge
python-3.12.7              |hc5c86c4_0_cpython        30.1 MB  conda-forge
readline-8.2               |       h8228510_1         275 KB  conda-forge
setuptools-75.7.0          |     pyhff2d567_0         756 KB  conda-forge
tk-8.6.13                  |noxft_h4845f30_101         3.2 MB  conda-forge
tzdata-2024b               |       hc8b5060_0         119 KB  conda-forge
wheel-0.45.1               |     pyhd8ed1ab_1          61 KB  conda-forge
xz-5.6.3                   |       hbcc6ac9_1          23 KB  conda-forge
xz-gpl-tools-5.6.3         |       hbcc6ac9_1          33 KB  conda-forge
xz-tools-5.6.3             |       hb9d3cd8_1          88 KB  conda-forge
------------------------------------------------------------
Total:        43.4 MB

The following NEW packages will be INSTALLED:

_libgcc_mutex      conda-forge/linux-64::_libgcc_mutex-0.1-conda_forge
_openmp_mutex      conda-forge/linux-64::_openmp_mutex-4.5-2_gnu
bzip2              conda-forge/linux-64::bzip2-1.0.8-h4bc722e_7
ca-certificates    conda-forge/linux-64::ca-certificates-2024.12.14-hbcca054_0
ld_impl_linux-64   conda-forge/linux-64::ld_impl_linux-64-2.43-h712a8e2_2
libexpat           conda-forge/linux-64::libexpat-2.6.4-h5888daf_0
libffi             conda-forge/linux-64::libffi-3.4.2-h7f98852_5
libgcc             conda-forge/linux-64::libgcc-14.2.0-h77fa898_1
libgcc-ng          conda-forge/linux-64::libgcc-ng-14.2.0-h69a702a_1
libgomp            conda-forge/linux-64::libgomp-14.2.0-h77fa898_1
liblzma            conda-forge/linux-64::liblzma-5.6.3-hb9d3cd8_1
liblzma-devel      conda-forge/linux-64::liblzma-devel-5.6.3-hb9d3cd8_1
libnsl             conda-forge/linux-64::libnsl-2.0.1-hd590300_0
libsqlite          conda-forge/linux-64::libsqlite-3.47.2-hee588c1_0
libuuid            conda-forge/linux-64::libuuid-2.38.1-h0b41bf4_0
libxcrypt          conda-forge/linux-64::libxcrypt-4.4.36-hd590300_1
libzlib            conda-forge/linux-64::libzlib-1.3.1-hb9d3cd8_2
ncurses            conda-forge/linux-64::ncurses-6.5-he02047a_1
openssl            conda-forge/linux-64::openssl-3.4.0-h7b32b05_1
pip                conda-forge/noarch::pip-24.3.1-pyh8b19718_2
python             conda-forge/linux-64::python-3.12.7-hc5c86c4_0_cpython
readline           conda-forge/linux-64::readline-8.2-h8228510_1
setuptools         conda-forge/noarch::setuptools-75.7.0-pyhff2d567_0
tk                 conda-forge/linux-64::tk-8.6.13-noxft_h4845f30_101
tzdata             conda-forge/noarch::tzdata-2024b-hc8b5060_0
wheel              conda-forge/noarch::wheel-0.45.1-pyhd8ed1ab_1
xz                 conda-forge/linux-64::xz-5.6.3-hbcc6ac9_1
xz-gpl-tools       conda-forge/linux-64::xz-gpl-tools-5.6.3-hbcc6ac9_1
xz-tools           conda-forge/linux-64::xz-tools-5.6.3-hb9d3cd8_1
```

```
Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate pydata-book
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

After installation completes, activate environment with `conda activate`:

```
(base) nqbh@nqbh-dell:~$ conda activate pydata-book
(pydata-book) nqbh@nqbh-dell:~$
```

Remark 3. *Necessary to use* `conda activate` *to activate your environment each time you open a new terminal. Can see information about active conda environment at any time from terminal by running* `conda info`.

Now, install essential packages used throughout book (along with their dependencies) with `conda install`:

```
(pydata-book) $ conda install -y {\tt pandas} jupyter matplotlib

(pydata-book) nqbh@nqbh-dell:~$ conda install -y {\tt pandas} jupyter matplotlib
Channels:
- conda-forge
- defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: /home/nqbh/anaconda3/envs/pydata-book

added / updated specs:
- jupyter
- matplotlib
- pandas


The following packages will be downloaded:

package                    |            build
---------------------------|-----------------
alsa-lib-1.2.13            |       hb9d3cd8_0         547 KB  conda-forge
anyio-4.8.0                |     pyhd8ed1ab_0         113 KB  conda-forge
argon2-cffi-23.1.0         |     pyhd8ed1ab_1          18 KB  conda-forge
argon2-cffi-bindings-21.2.0|  py312h66e93f0_5          34 KB  conda-forge
arrow-1.3.0                |     pyhd8ed1ab_1          98 KB  conda-forge
asttokens-3.0.0            |     pyhd8ed1ab_1          28 KB  conda-forge
async-lru-2.0.4            |     pyhd8ed1ab_1          15 KB  conda-forge
attrs-24.3.0               |     pyh71513ae_0          55 KB  conda-forge
babel-2.16.0               |     pyhd8ed1ab_1         6.2 MB  conda-forge
beautifulsoup4-4.12.3      |     pyha770c72_1         115 KB  conda-forge
bleach-6.2.0               |     pyhd8ed1ab_3         129 KB  conda-forge
bleach-with-css-6.2.0      |       hd8ed1ab_3           6 KB  conda-forge
brotli-1.1.0               |       hb9d3cd8_2          19 KB  conda-forge
brotli-bin-1.1.0           |       hb9d3cd8_2          18 KB  conda-forge
brotli-python-1.1.0        |  py312h2ec8cdc_2         342 KB  conda-forge
cached-property-1.5.2      |       hd8ed1ab_1           4 KB  conda-forge
cached_property-1.5.2      |     pyha770c72_1          11 KB  conda-forge
cairo-1.18.2               |       h3394656_1         956 KB  conda-forge
```

```
certifi-2024.12.14          |     pyhd8ed1ab_0         158 KB  conda-forge
cffi-1.17.1                 |  py312h06ac9bb_0         288 KB  conda-forge
charset-normalizer-3.4.1    |     pyhd8ed1ab_0          46 KB  conda-forge
comm-0.2.2                  |     pyhd8ed1ab_1          12 KB  conda-forge
contourpy-1.3.1             |  py312h68727a3_0         270 KB  conda-forge
cycler-0.12.1               |     pyhd8ed1ab_1          13 KB  conda-forge
cyrus-sasl-2.1.27           |       h54b06d7_7         214 KB  conda-forge
dbus-1.13.6                 |       h5008d03_3         604 KB  conda-forge
debugpy-1.8.11              |  py312h2ec8cdc_0         2.5 MB  conda-forge
decorator-5.1.1             |     pyhd8ed1ab_1          14 KB  conda-forge
defusedxml-0.7.1            |     pyhd8ed1ab_0          23 KB  conda-forge
double-conversion-3.3.0     |       h59595ed_0          77 KB  conda-forge
entrypoints-0.4             |     pyhd8ed1ab_1          11 KB  conda-forge
exceptiongroup-1.2.2        |     pyhd8ed1ab_1          20 KB  conda-forge
executing-2.1.0             |     pyhd8ed1ab_1          28 KB  conda-forge
expat-2.6.4                 |       h5888daf_0         135 KB  conda-forge
font-ttf-dejavu-sans-mono-2.37|       hab24e00_0         388 KB  conda-forge
font-ttf-inconsolata-3.000  |       h77eed37_0          94 KB  conda-forge
font-ttf-source-code-pro-2.038|       h77eed37_0         684 KB  conda-forge
font-ttf-ubuntu-0.83        |       h77eed37_3         1.5 MB  conda-forge
fontconfig-2.15.0           |       h7e30c49_1         259 KB  conda-forge
fonts-conda-ecosystem-1     |                 0           4 KB  conda-forge
fonts-conda-forge-1         |                 0           4 KB  conda-forge
fonttools-4.55.3            |  py312h178313f_1         2.7 MB  conda-forge
fqdn-1.5.1                  |     pyhd8ed1ab_1          16 KB  conda-forge
freetype-2.12.1             |       h267a509_2         620 KB  conda-forge
graphite2-1.3.13            |    h59595ed_1003          95 KB  conda-forge
h11-0.14.0                  |     pyhd8ed1ab_1          51 KB  conda-forge
h2-4.1.0                    |     pyhd8ed1ab_1          51 KB  conda-forge
harfbuzz-10.1.0             |       h0b3b770_0         1.5 MB  conda-forge
hpack-4.0.0                 |     pyhd8ed1ab_1          29 KB  conda-forge
httpcore-1.0.7              |     pyh29332c3_1          48 KB  conda-forge
httpx-0.28.1                |     pyhd8ed1ab_0          62 KB  conda-forge
hyperframe-6.0.1            |     pyhd8ed1ab_1          17 KB  conda-forge
icu-75.1                    |       he02047a_0        11.6 MB  conda-forge
idna-3.10                   |     pyhd8ed1ab_1          49 KB  conda-forge
importlib-metadata-8.5.0    |     pyha770c72_1          28 KB  conda-forge
importlib_resources-6.5.2   |     pyhd8ed1ab_0          33 KB  conda-forge
ipykernel-6.29.5            |     pyh3099207_0         116 KB  conda-forge
ipython-8.31.0              |     pyh707e725_0         587 KB  conda-forge
ipywidgets-8.1.5            |     pyhd8ed1ab_1         111 KB  conda-forge
isoduration-20.11.0         |     pyhd8ed1ab_1          19 KB  conda-forge
jedi-0.19.2                 |     pyhd8ed1ab_1         824 KB  conda-forge
jinja2-3.1.5                |     pyhd8ed1ab_0         110 KB  conda-forge
json5-0.10.0                |     pyhd8ed1ab_1          31 KB  conda-forge
jsonpointer-3.0.0           |  py312h7900ff3_1          17 KB  conda-forge
jsonschema-4.23.0           |     pyhd8ed1ab_1          73 KB  conda-forge
jsonschema-specifications-2024.10.1|     pyhd8ed1ab_1          16 KB  conda-forge
jsonschema-with-format-nongpl-4.23.0|       hd8ed1ab_1           7 KB  conda-forge
jupyter-1.1.1               |     pyhd8ed1ab_1           9 KB  conda-forge
jupyter-lsp-2.2.5           |     pyhd8ed1ab_1          54 KB  conda-forge
jupyter_client-8.6.3        |     pyhd8ed1ab_1         104 KB  conda-forge
jupyter_console-6.6.3       |     pyhd8ed1ab_1          26 KB  conda-forge
jupyter_core-5.7.2          |     pyh31011fe_1          56 KB  conda-forge
jupyter_events-0.11.0       |     pyhd8ed1ab_0          22 KB  conda-forge
jupyter_server-2.15.0       |     pyhd8ed1ab_0         320 KB  conda-forge
jupyter_server_terminals-0.5.3|     pyhd8ed1ab_1          19 KB  conda-forge
jupyterlab-4.3.4            |     pyhd8ed1ab_0         6.9 MB  conda-forge
jupyterlab_pygments-0.3.0   |     pyhd8ed1ab_2          18 KB  conda-forge
jupyterlab_server-2.27.3    |     pyhd8ed1ab_1          48 KB  conda-forge
jupyterlab_widgets-3.0.13   |     pyhd8ed1ab_1         182 KB  conda-forge
keyutils-1.6.1              |       h166bdaf_0         115 KB  conda-forge
kiwisolver-1.4.7            |  py312h68727a3_0          69 KB  conda-forge
krb5-1.21.3                 |       h659f571_0         1.3 MB  conda-forge
```

```
lcms2-2.16                |       hb7c19ff_0         239 KB  conda-forge
lerc-4.0.0                |       h27087fc_0         275 KB  conda-forge
libblas-3.9.0             |26_linux64_openblas          16 KB  conda-forge
libbrotlicommon-1.1.0     |       hb9d3cd8_2          67 KB  conda-forge
libbrotlidec-1.1.0        |       hb9d3cd8_2          32 KB  conda-forge
libbrotlienc-1.1.0        |       hb9d3cd8_2         275 KB  conda-forge
libcblas-3.9.0            |26_linux64_openblas          16 KB  conda-forge
libclang-cpp19.1-19.1.6   |default_hb5137d0_0        19.6 MB  conda-forge
libclang13-19.1.6         |default_h9c6a7e4_0        11.3 MB  conda-forge
libcups-2.3.3             |       h4637d8d_4         4.3 MB  conda-forge
libdeflate-1.23           |       h4ddbbb0_0          71 KB  conda-forge
libdrm-2.4.124            |       hb9d3cd8_0         237 KB  conda-forge
libedit-3.1.20240808      | pl5321h7949ede_0         132 KB  conda-forge
libegl-1.7.0              |       ha4b6fd6_2          44 KB  conda-forge
libgfortran-14.2.0        |       h69a702a_1          53 KB  conda-forge
libgfortran5-14.2.0       |       hd5240d6_1         1.4 MB  conda-forge
libgl-1.7.0               |       ha4b6fd6_2         132 KB  conda-forge
libglib-2.82.2            |       h2ff4ddf_0         3.7 MB  conda-forge
libglvnd-1.7.0            |       ha4b6fd6_2         129 KB  conda-forge
libglx-1.7.0              |       ha4b6fd6_2          74 KB  conda-forge
libiconv-1.17             |       hd590300_2         689 KB  conda-forge
libjpeg-turbo-3.0.0       |       hd590300_1         604 KB  conda-forge
liblapack-3.9.0           |26_linux64_openblas          16 KB  conda-forge
libllvm19-19.1.6          |       ha7bfdaf_0        38.3 MB  conda-forge
libntlm-1.8               |       hb9d3cd8_0          33 KB  conda-forge
libopenblas-0.3.28        |pthreads_h94d23a6_1         5.3 MB  conda-forge
libopengl-1.7.0           |       ha4b6fd6_2          50 KB  conda-forge
libpciaccess-0.18         |       hd590300_0          28 KB  conda-forge
libpng-1.6.45             |       h943b412_0         283 KB  conda-forge
libpq-17.2                |       h3b95a9b_1         2.5 MB  conda-forge
libsodium-1.0.20          |       h4ab18f5_0         201 KB  conda-forge
libstdcxx-14.2.0          |       hc0a3c3a_1         3.7 MB  conda-forge
libstdcxx-ng-14.2.0       |       h4852527_1          53 KB  conda-forge
libtiff-4.7.0             |       hd9ff511_3         418 KB  conda-forge
libwebp-base-1.5.0        |       h851e524_0         420 KB  conda-forge
libxcb-1.17.0             |       h8a09558_0         387 KB  conda-forge
libxkbcommon-1.7.0        |       h2c5496b_1         579 KB  conda-forge
libxml2-2.13.5            |       h8d12d68_1         674 KB  conda-forge
libxslt-1.1.39            |       h76b75d6_0         248 KB  conda-forge
markupsafe-3.0.2          |  py312h178313f_1          24 KB  conda-forge
matplotlib-3.10.0         |  py312h7900ff3_0          16 KB  conda-forge
matplotlib-base-3.10.0    |  py312hd3ec401_0         7.8 MB  conda-forge
matplotlib-inline-0.1.7   |     pyhd8ed1ab_1          14 KB  conda-forge
mistune-3.1.0             |     pyhd8ed1ab_0          67 KB  conda-forge
munkres-1.1.4             |     pyh9f0ad1d_0          12 KB  conda-forge
mysql-common-9.0.1        |       h266115a_4         605 KB  conda-forge
mysql-libs-9.0.1          |       he0572af_4         1.3 MB  conda-forge
nbclient-0.10.2           |     pyhd8ed1ab_0          27 KB  conda-forge
nbconvert-core-7.16.5     |     pyhd8ed1ab_1         185 KB  conda-forge
nbformat-5.10.4           |     pyhd8ed1ab_1          99 KB  conda-forge
nest-asyncio-1.6.0        |     pyhd8ed1ab_1          11 KB  conda-forge
notebook-7.3.2            |     pyhd8ed1ab_0         8.6 MB  conda-forge
notebook-shim-0.2.4       |     pyhd8ed1ab_1          16 KB  conda-forge
numpy-2.2.1               |  py312h7e784f5_0         8.1 MB  conda-forge
openjpeg-2.5.3            |       h5fbd93e_0         335 KB  conda-forge
openldap-2.6.9            |       he970967_0         766 KB  conda-forge
overrides-7.7.0           |     pyhd8ed1ab_1          29 KB  conda-forge
packaging-24.2            |     pyhd8ed1ab_2          59 KB  conda-forge
pandas-2.2.3              |  py312hf9745cd_1        14.7 MB  conda-forge
pandocfilters-1.5.0       |     pyhd8ed1ab_0          11 KB  conda-forge
parso-0.8.4               |     pyhd8ed1ab_1          74 KB  conda-forge
pcre2-10.44               |       hba22ea6_2         930 KB  conda-forge
pexpect-4.9.0             |     pyhd8ed1ab_1          52 KB  conda-forge
pickleshare-0.7.5         |  pyhd8ed1ab_1004          11 KB  conda-forge
```

```
pillow-11.1.0              |  py312h80c1187_0        40.8 MB  conda-forge
pixman-0.44.2              |       h29eaf8c_0         372 KB  conda-forge
pkgutil-resolve-name-1.3.10|     pyhd8ed1ab_2          10 KB  conda-forge
platformdirs-4.3.6         |     pyhd8ed1ab_1          20 KB  conda-forge
prometheus_client-0.21.1   |     pyhd8ed1ab_0          48 KB  conda-forge
prompt-toolkit-3.0.48      |     pyha770c72_1         264 KB  conda-forge
prompt_toolkit-3.0.48      |       hd8ed1ab_1           6 KB  conda-forge
psutil-6.1.1               |  py312h66e93f0_0         476 KB  conda-forge
pthread-stubs-0.4          |    hb9d3cd8_1002           8 KB  conda-forge
ptyprocess-0.7.0           |     pyhd8ed1ab_1          19 KB  conda-forge
pure_eval-0.2.3            |     pyhd8ed1ab_1          16 KB  conda-forge
pycparser-2.22             |     pyh29332c3_1         108 KB  conda-forge
pygments-2.19.1            |     pyhd8ed1ab_0         868 KB  conda-forge
pyparsing-3.2.1            |     pyhd8ed1ab_0          91 KB  conda-forge
pyside6-6.8.1              |  py312h91f0f75_0        10.4 MB  conda-forge
pysocks-1.7.1              |     pyha55dd90_7          21 KB  conda-forge
python-dateutil-2.9.0.post0|     pyhff2d567_1         217 KB  conda-forge
python-fastjsonschema-2.21.1|      pyhd8ed1ab_0         221 KB  conda-forge
python-json-logger-2.0.7   |     pyhd8ed1ab_0          13 KB  conda-forge
python-tzdata-2024.2       |     pyhd8ed1ab_1         139 KB  conda-forge
python_abi-3.12            |           5_cp312           6 KB  conda-forge
pytz-2024.1                |     pyhd8ed1ab_0         184 KB  conda-forge
pyyaml-6.0.2               |  py312h66e93f0_1         202 KB  conda-forge
pyzmq-26.2.0               |  py312hbf22597_3         369 KB  conda-forge
qhull-2020.2               |       h434a139_5         540 KB  conda-forge
qt6-main-6.8.1             |       h588cce1_2        49.2 MB  conda-forge
referencing-0.35.1         |     pyhd8ed1ab_1          41 KB  conda-forge
requests-2.32.3            |     pyhd8ed1ab_1          57 KB  conda-forge
rfc3339-validator-0.1.4    |     pyhd8ed1ab_1          10 KB  conda-forge
rfc3986-validator-0.1.1    |     pyh9f0ad1d_0           8 KB  conda-forge
rpds-py-0.22.3             |  py312h12e396e_0         346 KB  conda-forge
send2trash-1.8.3           |     pyh0d859eb_1          22 KB  conda-forge
six-1.17.0                 |     pyhd8ed1ab_0          16 KB  conda-forge
sniffio-1.3.1              |     pyhd8ed1ab_1          15 KB  conda-forge
soupsieve-2.5              |     pyhd8ed1ab_1          36 KB  conda-forge
stack_data-0.6.3           |     pyhd8ed1ab_1          26 KB  conda-forge
terminado-0.18.1           |     pyh0d859eb_0          22 KB  conda-forge
tinycss2-1.4.0             |     pyhd8ed1ab_0          28 KB  conda-forge
tomli-2.2.1                |     pyhd8ed1ab_1          19 KB  conda-forge
tornado-6.4.2              |  py312h66e93f0_0         821 KB  conda-forge
traitlets-5.14.3           |     pyhd8ed1ab_1         107 KB  conda-forge
types-python-dateutil-2.9.0.20241206|     pyhd8ed1ab_0          22 KB  conda-forge
typing-extensions-4.12.2   |       hd8ed1ab_1          10 KB  conda-forge
typing_extensions-4.12.2   |     pyha770c72_1          39 KB  conda-forge
typing_utils-0.1.0         |     pyhd8ed1ab_1          15 KB  conda-forge
unicodedata2-15.1.0        |  py312h66e93f0_1         360 KB  conda-forge
uri-template-1.3.0         |     pyhd8ed1ab_1          23 KB  conda-forge
urllib3-2.3.0              |     pyhd8ed1ab_0          98 KB  conda-forge
wayland-1.23.1             |       h3e06ad9_0         314 KB  conda-forge
wcwidth-0.2.13             |     pyhd8ed1ab_1          32 KB  conda-forge
webcolors-24.11.1          |     pyhd8ed1ab_0          18 KB  conda-forge
webencodings-0.5.1         |     pyhd8ed1ab_3          15 KB  conda-forge
websocket-client-1.8.0     |     pyhd8ed1ab_1          46 KB  conda-forge
widgetsnbextension-4.0.13  |     pyhd8ed1ab_1         877 KB  conda-forge
xcb-util-0.4.1             |       hb711507_2          19 KB  conda-forge
xcb-util-cursor-0.1.5      |       hb9d3cd8_0          20 KB  conda-forge
xcb-util-image-0.4.0       |       hb711507_2          24 KB  conda-forge
xcb-util-keysyms-0.4.1     |       hb711507_0          14 KB  conda-forge
xcb-util-renderutil-0.3.10 |       hb711507_0          17 KB  conda-forge
xcb-util-wm-0.4.2          |       hb711507_0          50 KB  conda-forge
xkeyboard-config-2.43      |       hb9d3cd8_0         380 KB  conda-forge
xorg-libice-1.1.2          |       hb9d3cd8_0          57 KB  conda-forge
xorg-libsm-1.2.5           |       he73a12e_0          27 KB  conda-forge
xorg-libx11-1.8.10         |       h4f16b4b_1         818 KB  conda-forge
```

```
xorg-libxau-1.0.12          |       hb9d3cd8_0          14 KB  conda-forge
xorg-libxcomposite-0.4.6    |       hb9d3cd8_2          13 KB  conda-forge
xorg-libxcursor-1.2.3       |       hb9d3cd8_0          32 KB  conda-forge
xorg-libxdamage-1.1.6       |       hb9d3cd8_0          13 KB  conda-forge
xorg-libxdmcp-1.1.5         |       hb9d3cd8_0          19 KB  conda-forge
xorg-libxext-1.3.6          |       hb9d3cd8_0          49 KB  conda-forge
xorg-libxfixes-6.0.1        |       hb9d3cd8_0          19 KB  conda-forge
xorg-libxi-1.8.2            |       hb9d3cd8_0          46 KB  conda-forge
xorg-libxrandr-1.5.4        |       hb9d3cd8_0          29 KB  conda-forge
xorg-libxrender-0.9.12      |       hb9d3cd8_0          32 KB  conda-forge
xorg-libxtst-1.2.5          |       hb9d3cd8_3          32 KB  conda-forge
xorg-libxxf86vm-1.1.6       |       hb9d3cd8_0          17 KB  conda-forge
yaml-0.2.5                  |       h7f98852_2          87 KB  conda-forge
zeromq-4.3.5                |       h3b0a872_7         328 KB  conda-forge
zipp-3.21.0                 |     pyhd8ed1ab_1          21 KB  conda-forge
zstandard-0.23.0            |  py312hef9b889_1         410 KB  conda-forge
zstd-1.5.6                  |       ha6fb4c9_0         542 KB  conda-forge
------------------------------------------------------------
Total:       295.3 MB

The following NEW packages will be INSTALLED:

alsa-lib           conda-forge/linux-64::alsa-lib-1.2.13-hb9d3cd8_0
anyio              conda-forge/noarch::anyio-4.8.0-pyhd8ed1ab_0
argon2-cffi        conda-forge/noarch::argon2-cffi-23.1.0-pyhd8ed1ab_1
argon2-cffi-bindi~ conda-forge/linux-64::argon2-cffi-bindings-21.2.0-py312h66e93f0_5
arrow              conda-forge/noarch::arrow-1.3.0-pyhd8ed1ab_1
asttokens          conda-forge/noarch::asttokens-3.0.0-pyhd8ed1ab_1
async-lru          conda-forge/noarch::async-lru-2.0.4-pyhd8ed1ab_1
attrs              conda-forge/noarch::attrs-24.3.0-pyh71513ae_0
babel              conda-forge/noarch::babel-2.16.0-pyhd8ed1ab_1
beautifulsoup4     conda-forge/noarch::beautifulsoup4-4.12.3-pyha770c72_1
bleach             conda-forge/noarch::bleach-6.2.0-pyhd8ed1ab_3
bleach-with-css    conda-forge/noarch::bleach-with-css-6.2.0-hd8ed1ab_3
brotli             conda-forge/linux-64::brotli-1.1.0-hb9d3cd8_2
brotli-bin         conda-forge/linux-64::brotli-bin-1.1.0-hb9d3cd8_2
brotli-python      conda-forge/linux-64::brotli-python-1.1.0-py312h2ec8cdc_2
cached-property    conda-forge/noarch::cached-property-1.5.2-hd8ed1ab_1
cached_property    conda-forge/noarch::cached_property-1.5.2-pyha770c72_1
cairo              conda-forge/linux-64::cairo-1.18.2-h3394656_1
certifi            conda-forge/noarch::certifi-2024.12.14-pyhd8ed1ab_0
cffi               conda-forge/linux-64::cffi-1.17.1-py312h06ac9bb_0
charset-normalizer conda-forge/noarch::charset-normalizer-3.4.1-pyhd8ed1ab_0
comm               conda-forge/noarch::comm-0.2.2-pyhd8ed1ab_1
contourpy          conda-forge/linux-64::contourpy-1.3.1-py312h68727a3_0
cycler             conda-forge/noarch::cycler-0.12.1-pyhd8ed1ab_1
cyrus-sasl         conda-forge/linux-64::cyrus-sasl-2.1.27-h54b06d7_7
dbus               conda-forge/linux-64::dbus-1.13.6-h5008d03_3
debugpy            conda-forge/linux-64::debugpy-1.8.11-py312h2ec8cdc_0
decorator          conda-forge/noarch::decorator-5.1.1-pyhd8ed1ab_1
defusedxml         conda-forge/noarch::defusedxml-0.7.1-pyhd8ed1ab_0
double-conversion  conda-forge/linux-64::double-conversion-3.3.0-h59595ed_0
entrypoints        conda-forge/noarch::entrypoints-0.4-pyhd8ed1ab_1
exceptiongroup     conda-forge/noarch::exceptiongroup-1.2.2-pyhd8ed1ab_1
executing          conda-forge/noarch::executing-2.1.0-pyhd8ed1ab_1
expat              conda-forge/linux-64::expat-2.6.4-h5888daf_0
font-ttf-dejavu-s~ conda-forge/noarch::font-ttf-dejavu-sans-mono-2.37-hab24e00_0
font-ttf-inconsol~ conda-forge/noarch::font-ttf-inconsolata-3.000-h77eed37_0
font-ttf-source-c~ conda-forge/noarch::font-ttf-source-code-pro-2.038-h77eed37_0
font-ttf-ubuntu    conda-forge/noarch::font-ttf-ubuntu-0.83-h77eed37_3
fontconfig         conda-forge/linux-64::fontconfig-2.15.0-h7e30c49_1
fonts-conda-ecosy~ conda-forge/noarch::fonts-conda-ecosystem-1-0
fonts-conda-forge  conda-forge/noarch::fonts-conda-forge-1-0
fonttools          conda-forge/linux-64::fonttools-4.55.3-py312h178313f_1
```

```
fqdn                conda-forge/noarch::fqdn-1.5.1-pyhd8ed1ab_1
freetype            conda-forge/linux-64::freetype-2.12.1-h267a509_2
graphite2           conda-forge/linux-64::graphite2-1.3.13-h59595ed_1003
h11                 conda-forge/noarch::h11-0.14.0-pyhd8ed1ab_1
h2                  conda-forge/noarch::h2-4.1.0-pyhd8ed1ab_1
harfbuzz            conda-forge/linux-64::harfbuzz-10.1.0-h0b3b770_0
hpack               conda-forge/noarch::hpack-4.0.0-pyhd8ed1ab_1
httpcore            conda-forge/noarch::httpcore-1.0.7-pyh29332c3_1
httpx               conda-forge/noarch::httpx-0.28.1-pyhd8ed1ab_0
hyperframe          conda-forge/noarch::hyperframe-6.0.1-pyhd8ed1ab_1
icu                 conda-forge/linux-64::icu-75.1-he02047a_0
idna                conda-forge/noarch::idna-3.10-pyhd8ed1ab_1
importlib-metadata conda-forge/noarch::importlib-metadata-8.5.0-pyha770c72_1
importlib_resourc~ conda-forge/noarch::importlib_resources-6.5.2-pyhd8ed1ab_0
ipykernel           conda-forge/noarch::ipykernel-6.29.5-pyh3099207_0
ipython             conda-forge/noarch::ipython-8.31.0-pyh707e725_0
ipywidgets          conda-forge/noarch::ipywidgets-8.1.5-pyhd8ed1ab_1
isoduration         conda-forge/noarch::isoduration-20.11.0-pyhd8ed1ab_1
jedi                conda-forge/noarch::jedi-0.19.2-pyhd8ed1ab_1
jinja2              conda-forge/noarch::jinja2-3.1.5-pyhd8ed1ab_0
json5               conda-forge/noarch::json5-0.10.0-pyhd8ed1ab_1
jsonpointer         conda-forge/linux-64::jsonpointer-3.0.0-py312h7900ff3_1
jsonschema          conda-forge/noarch::jsonschema-4.23.0-pyhd8ed1ab_1
jsonschema-specif~ conda-forge/noarch::jsonschema-specifications-2024.10.1-pyhd8ed1ab_1
jsonschema-with-f~ conda-forge/noarch::jsonschema-with-format-nongpl-4.23.0-hd8ed1ab_1
jupyter             conda-forge/noarch::jupyter-1.1.1-pyhd8ed1ab_1
jupyter-lsp         conda-forge/noarch::jupyter-lsp-2.2.5-pyhd8ed1ab_1
jupyter_client      conda-forge/noarch::jupyter_client-8.6.3-pyhd8ed1ab_1
jupyter_console     conda-forge/noarch::jupyter_console-6.6.3-pyhd8ed1ab_1
jupyter_core        conda-forge/noarch::jupyter_core-5.7.2-pyh31011fe_1
jupyter_events      conda-forge/noarch::jupyter_events-0.11.0-pyhd8ed1ab_0
jupyter_server      conda-forge/noarch::jupyter_server-2.15.0-pyhd8ed1ab_0
jupyter_server_te~ conda-forge/noarch::jupyter_server_terminals-0.5.3-pyhd8ed1ab_1
jupyterlab          conda-forge/noarch::jupyterlab-4.3.4-pyhd8ed1ab_0
jupyterlab_pygmen~ conda-forge/noarch::jupyterlab_pygments-0.3.0-pyhd8ed1ab_2
jupyterlab_server  conda-forge/noarch::jupyterlab_server-2.27.3-pyhd8ed1ab_1
jupyterlab_widgets conda-forge/noarch::jupyterlab_widgets-3.0.13-pyhd8ed1ab_1
keyutils            conda-forge/linux-64::keyutils-1.6.1-h166bdaf_0
kiwisolver          conda-forge/linux-64::kiwisolver-1.4.7-py312h68727a3_0
krb5                conda-forge/linux-64::krb5-1.21.3-h659f571_0
lcms2               conda-forge/linux-64::lcms2-2.16-hb7c19ff_0
lerc                conda-forge/linux-64::lerc-4.0.0-h27087fc_0
libblas             conda-forge/linux-64::libblas-3.9.0-26_linux64_openblas
libbrotlicommon     conda-forge/linux-64::libbrotlicommon-1.1.0-hb9d3cd8_2
libbrotlidec        conda-forge/linux-64::libbrotlidec-1.1.0-hb9d3cd8_2
libbrotlienc        conda-forge/linux-64::libbrotlienc-1.1.0-hb9d3cd8_2
libcblas            conda-forge/linux-64::libcblas-3.9.0-26_linux64_openblas
libclang-cpp19.1    conda-forge/linux-64::libclang-cpp19.1-19.1.6-default_hb5137d0_0
libclang13          conda-forge/linux-64::libclang13-19.1.6-default_h9c6a7e4_0
libcups             conda-forge/linux-64::libcups-2.3.3-h4637d8d_4
libdeflate          conda-forge/linux-64::libdeflate-1.23-h4ddbbb0_0
libdrm              conda-forge/linux-64::libdrm-2.4.124-hb9d3cd8_0
libedit             conda-forge/linux-64::libedit-3.1.20240808-pl5321h7949ede_0
libegl              conda-forge/linux-64::libegl-1.7.0-ha4b6fd6_2
libgfortran         conda-forge/linux-64::libgfortran-14.2.0-h69a702a_1
libgfortran5        conda-forge/linux-64::libgfortran5-14.2.0-hd5240d6_1
libgl               conda-forge/linux-64::libgl-1.7.0-ha4b6fd6_2
libglib             conda-forge/linux-64::libglib-2.82.2-h2ff4ddf_0
libglvnd            conda-forge/linux-64::libglvnd-1.7.0-ha4b6fd6_2
libglx              conda-forge/linux-64::libglx-1.7.0-ha4b6fd6_2
libiconv            conda-forge/linux-64::libiconv-1.17-hd590300_2
libjpeg-turbo       conda-forge/linux-64::libjpeg-turbo-3.0.0-hd590300_1
liblapack           conda-forge/linux-64::liblapack-3.9.0-26_linux64_openblas
libllvm19           conda-forge/linux-64::libllvm19-19.1.6-ha7bfdaf_0
```

```
libntlm             conda-forge/linux-64::libntlm-1.8-hb9d3cd8_0
libopenblas         conda-forge/linux-64::libopenblas-0.3.28-pthreads_h94d23a6_1
libopengl           conda-forge/linux-64::libopengl-1.7.0-ha4b6fd6_2
libpciaccess        conda-forge/linux-64::libpciaccess-0.18-hd590300_0
libpng              conda-forge/linux-64::libpng-1.6.45-h943b412_0
libpq               conda-forge/linux-64::libpq-17.2-h3b95a9b_1
libsodium           conda-forge/linux-64::libsodium-1.0.20-h4ab18f5_0
libstdcxx           conda-forge/linux-64::libstdcxx-14.2.0-hc0a3c3a_1
libstdcxx-ng        conda-forge/linux-64::libstdcxx-ng-14.2.0-h4852527_1
libtiff             conda-forge/linux-64::libtiff-4.7.0-hd9ff511_3
libwebp-base        conda-forge/linux-64::libwebp-base-1.5.0-h851e524_0
libxcb              conda-forge/linux-64::libxcb-1.17.0-h8a09558_0
libxkbcommon        conda-forge/linux-64::libxkbcommon-1.7.0-h2c5496b_1
libxml2             conda-forge/linux-64::libxml2-2.13.5-h8d12d68_1
libxslt             conda-forge/linux-64::libxslt-1.1.39-h76b75d6_0
markupsafe          conda-forge/linux-64::markupsafe-3.0.2-py312h178313f_1
matplotlib          conda-forge/linux-64::matplotlib-3.10.0-py312h7900ff3_0
matplotlib-base     conda-forge/linux-64::matplotlib-base-3.10.0-py312hd3ec401_0
matplotlib-inline   conda-forge/noarch::matplotlib-inline-0.1.7-pyhd8ed1ab_1
mistune             conda-forge/noarch::mistune-3.1.0-pyhd8ed1ab_0
munkres             conda-forge/noarch::munkres-1.1.4-pyh9f0ad1d_0
mysql-common        conda-forge/linux-64::mysql-common-9.0.1-h266115a_4
mysql-libs          conda-forge/linux-64::mysql-libs-9.0.1-he0572af_4
nbclient            conda-forge/noarch::nbclient-0.10.2-pyhd8ed1ab_0
nbconvert-core      conda-forge/noarch::nbconvert-core-7.16.5-pyhd8ed1ab_1
nbformat            conda-forge/noarch::nbformat-5.10.4-pyhd8ed1ab_1
nest-asyncio        conda-forge/noarch::nest-asyncio-1.6.0-pyhd8ed1ab_1
notebook            conda-forge/noarch::notebook-7.3.2-pyhd8ed1ab_0
notebook-shim       conda-forge/noarch::notebook-shim-0.2.4-pyhd8ed1ab_1
numpy               conda-forge/linux-64::numpy-2.2.1-py312h7e784f5_0
openjpeg            conda-forge/linux-64::openjpeg-2.5.3-h5fbd93e_0
openldap            conda-forge/linux-64::openldap-2.6.9-he970967_0
overrides           conda-forge/noarch::overrides-7.7.0-pyhd8ed1ab_1
packaging           conda-forge/noarch::packaging-24.2-pyhd8ed1ab_2
pandas              conda-forge/linux-64::pandas-2.2.3-py312hf9745cd_1
pandocfilters       conda-forge/noarch::pandocfilters-1.5.0-pyhd8ed1ab_0
parso               conda-forge/noarch::parso-0.8.4-pyhd8ed1ab_1
pcre2               conda-forge/linux-64::pcre2-10.44-hba22ea6_2
pexpect             conda-forge/noarch::pexpect-4.9.0-pyhd8ed1ab_1
pickleshare         conda-forge/noarch::pickleshare-0.7.5-pyhd8ed1ab_1004
pillow              conda-forge/linux-64::pillow-11.1.0-py312h80c1187_0
pixman              conda-forge/linux-64::pixman-0.44.2-h29eaf8c_0
pkgutil-resolve-n~  conda-forge/noarch::pkgutil-resolve-name-1.3.10-pyhd8ed1ab_2
platformdirs        conda-forge/noarch::platformdirs-4.3.6-pyhd8ed1ab_1
prometheus_client   conda-forge/noarch::prometheus_client-0.21.1-pyhd8ed1ab_0
prompt-toolkit      conda-forge/noarch::prompt-toolkit-3.0.48-pyha770c72_1
prompt_toolkit      conda-forge/noarch::prompt_toolkit-3.0.48-hd8ed1ab_1
psutil              conda-forge/linux-64::psutil-6.1.1-py312h66e93f0_0
pthread-stubs       conda-forge/linux-64::pthread-stubs-0.4-hb9d3cd8_1002
ptyprocess          conda-forge/noarch::ptyprocess-0.7.0-pyhd8ed1ab_1
pure_eval           conda-forge/noarch::pure_eval-0.2.3-pyhd8ed1ab_1
pycparser           conda-forge/noarch::pycparser-2.22-pyh29332c3_1
pygments            conda-forge/noarch::pygments-2.19.1-pyhd8ed1ab_0
pyparsing           conda-forge/noarch::pyparsing-3.2.1-pyhd8ed1ab_0
pyside6             conda-forge/linux-64::pyside6-6.8.1-py312h91f0f75_0
pysocks             conda-forge/noarch::pysocks-1.7.1-pyha55dd90_7
python-dateutil     conda-forge/noarch::python-dateutil-2.9.0.post0-pyhff2d567_1
python-fastjsonsc~  conda-forge/noarch::python-fastjsonschema-2.21.1-pyhd8ed1ab_0
python-json-logger  conda-forge/noarch::python-json-logger-2.0.7-pyhd8ed1ab_0
python-tzdata       conda-forge/noarch::python-tzdata-2024.2-pyhd8ed1ab_1
python_abi          conda-forge/linux-64::python_abi-3.12-5_cp312
pytz                conda-forge/noarch::pytz-2024.1-pyhd8ed1ab_0
pyyaml              conda-forge/linux-64::pyyaml-6.0.2-py312h66e93f0_1
pyzmq               conda-forge/linux-64::pyzmq-26.2.0-py312hbf22597_3
```

```
qhull               conda-forge/linux-64::qhull-2020.2-h434a139_5
qt6-main            conda-forge/linux-64::qt6-main-6.8.1-h588cce1_2
referencing         conda-forge/noarch::referencing-0.35.1-pyhd8ed1ab_1
requests            conda-forge/noarch::requests-2.32.3-pyhd8ed1ab_1
rfc3339-validator   conda-forge/noarch::rfc3339-validator-0.1.4-pyhd8ed1ab_1
rfc3986-validator   conda-forge/noarch::rfc3986-validator-0.1.1-pyh9f0ad1d_0
rpds-py             conda-forge/linux-64::rpds-py-0.22.3-py312h12e396e_0
send2trash          conda-forge/noarch::send2trash-1.8.3-pyh0d859eb_1
six                 conda-forge/noarch::six-1.17.0-pyhd8ed1ab_0
sniffio             conda-forge/noarch::sniffio-1.3.1-pyhd8ed1ab_1
soupsieve           conda-forge/noarch::soupsieve-2.5-pyhd8ed1ab_1
stack_data          conda-forge/noarch::stack_data-0.6.3-pyhd8ed1ab_1
terminado           conda-forge/noarch::terminado-0.18.1-pyh0d859eb_0
tinycss2            conda-forge/noarch::tinycss2-1.4.0-pyhd8ed1ab_0
tomli               conda-forge/noarch::tomli-2.2.1-pyhd8ed1ab_1
tornado             conda-forge/linux-64::tornado-6.4.2-py312h66e93f0_0
traitlets           conda-forge/noarch::traitlets-5.14.3-pyhd8ed1ab_1
types-python-date~ conda-forge/noarch::types-python-dateutil-2.9.0.20241206-pyhd8ed1ab_0
typing-extensions   conda-forge/noarch::typing-extensions-4.12.2-hd8ed1ab_1
typing_extensions   conda-forge/noarch::typing_extensions-4.12.2-pyha770c72_1
typing_utils        conda-forge/noarch::typing_utils-0.1.0-pyhd8ed1ab_1
unicodedata2        conda-forge/linux-64::unicodedata2-15.1.0-py312h66e93f0_1
uri-template        conda-forge/noarch::uri-template-1.3.0-pyhd8ed1ab_1
urllib3             conda-forge/noarch::urllib3-2.3.0-pyhd8ed1ab_0
wayland             conda-forge/linux-64::wayland-1.23.1-h3e06ad9_0
wcwidth             conda-forge/noarch::wcwidth-0.2.13-pyhd8ed1ab_1
webcolors           conda-forge/noarch::webcolors-24.11.1-pyhd8ed1ab_0
webencodings        conda-forge/noarch::webencodings-0.5.1-pyhd8ed1ab_3
websocket-client    conda-forge/noarch::websocket-client-1.8.0-pyhd8ed1ab_1
widgetsnbextension conda-forge/noarch::widgetsnbextension-4.0.13-pyhd8ed1ab_1
xcb-util            conda-forge/linux-64::xcb-util-0.4.1-hb711507_2
xcb-util-cursor     conda-forge/linux-64::xcb-util-cursor-0.1.5-hb9d3cd8_0
xcb-util-image      conda-forge/linux-64::xcb-util-image-0.4.0-hb711507_2
xcb-util-keysyms    conda-forge/linux-64::xcb-util-keysyms-0.4.1-hb711507_0
xcb-util-renderut~ conda-forge/linux-64::xcb-util-renderutil-0.3.10-hb711507_0
xcb-util-wm         conda-forge/linux-64::xcb-util-wm-0.4.2-hb711507_0
xkeyboard-config    conda-forge/linux-64::xkeyboard-config-2.43-hb9d3cd8_0
xorg-libice         conda-forge/linux-64::xorg-libice-1.1.2-hb9d3cd8_0
xorg-libsm          conda-forge/linux-64::xorg-libsm-1.2.5-he73a12e_0
xorg-libx11         conda-forge/linux-64::xorg-libx11-1.8.10-h4f16b4b_1
xorg-libxau         conda-forge/linux-64::xorg-libxau-1.0.12-hb9d3cd8_0
xorg-libxcomposite conda-forge/linux-64::xorg-libxcomposite-0.4.6-hb9d3cd8_2
xorg-libxcursor     conda-forge/linux-64::xorg-libxcursor-1.2.3-hb9d3cd8_0
xorg-libxdamage     conda-forge/linux-64::xorg-libxdamage-1.1.6-hb9d3cd8_0
xorg-libxdmcp       conda-forge/linux-64::xorg-libxdmcp-1.1.5-hb9d3cd8_0
xorg-libxext        conda-forge/linux-64::xorg-libxext-1.3.6-hb9d3cd8_0
xorg-libxfixes      conda-forge/linux-64::xorg-libxfixes-6.0.1-hb9d3cd8_0
xorg-libxi          conda-forge/linux-64::xorg-libxi-1.8.2-hb9d3cd8_0
xorg-libxrandr      conda-forge/linux-64::xorg-libxrandr-1.5.4-hb9d3cd8_0
xorg-libxrender     conda-forge/linux-64::xorg-libxrender-0.9.12-hb9d3cd8_0
xorg-libxtst        conda-forge/linux-64::xorg-libxtst-1.2.5-hb9d3cd8_3
xorg-libxxf86vm     conda-forge/linux-64::xorg-libxxf86vm-1.1.6-hb9d3cd8_0
yaml                conda-forge/linux-64::yaml-0.2.5-h7f98852_2
zeromq              conda-forge/linux-64::zeromq-4.3.5-h3b0a872_7
zipp                conda-forge/noarch::zipp-3.21.0-pyhd8ed1ab_1
zstandard           conda-forge/linux-64::zstandard-0.23.0-py312hef9b889_1
zstd                conda-forge/linux-64::zstd-1.5.6-ha6fb4c9_0



Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
```

```
Executing transaction: done
```

Will be using some other packages, too, but these can be installed later once they are needed. There are 2 ways to install packages: with `conda install` & with `pip install`. `conda install` should always be preferred when using Miniconda, but some packages are not available through conda, so if `conda install $package_name` fails, try `pip install $package_name`.

Remark 4. *If want to install all of packages used in rest of book, can do that now by running:*

```
(pydata-book) nqbh@nqbh-dell:~$ conda install lxml beautifulsoup4 html5lib openpyxl \
requests sqlalchemy seaborn scipy statsmodels \
patsy scikit-learn pyarrow pytables numba
Channels:
- conda-forge
- defaults
Platform: linux-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: /home/nqbh/anaconda3/envs/pydata-book

added / updated specs:
- beautifulsoup4
- html5lib
- lxml
- numba
- openpyxl
- patsy
- pyarrow
- pytables
- requests
- scikit-learn
- scipy
- seaborn
- sqlalchemy
- statsmodels


The following packages will be downloaded:

package                    |            build
---------------------------|-----------------
aws-c-auth-0.8.0           |      hb921021_15         105 KB  conda-forge
aws-c-cal-0.8.1            |       h1a47875_3          46 KB  conda-forge
aws-c-common-0.10.6        |       hb9d3cd8_0         231 KB  conda-forge
aws-c-compression-0.3.0    |       h4e1184b_5          19 KB  conda-forge
aws-c-event-stream-0.5.0   |      h7959bf6_11          53 KB  conda-forge
aws-c-http-0.9.2           |       hefd7a92_4         193 KB  conda-forge
aws-c-io-0.15.3            |       h831e299_5         154 KB  conda-forge
aws-c-mqtt-0.11.0          |      h11f4f37_12         190 KB  conda-forge
aws-c-s3-0.7.7             |       hf454442_0         111 KB  conda-forge
aws-c-sdkutils-0.2.1       |       h4e1184b_4          55 KB  conda-forge
aws-checksums-0.2.2        |       h4e1184b_4          71 KB  conda-forge
aws-crt-cpp-0.29.7         |       hd92328a_7         346 KB  conda-forge
aws-sdk-cpp-1.11.458       |       hc430e4a_4         2.9 MB  conda-forge
azure-core-cpp-1.14.0      |       h5cfcd09_0         337 KB  conda-forge
azure-identity-cpp-1.10.0  |       h113e628_0         227 KB  conda-forge
azure-storage-blobs-cpp-12.13.0|       h3cf044e_1         536 KB  conda-forge
azure-storage-common-cpp-12.8.0|       h736e048_1         146 KB  conda-forge
azure-storage-files-datalake-cpp-12.12.0|       ha633028_1         281 KB  conda-forge
blosc-1.21.6               |       he440d0b_1          47 KB  conda-forge
c-ares-1.34.4              |       hb9d3cd8_0         201 KB  conda-forge
c-blosc2-2.15.2            |       h3122c55_1         334 KB  conda-forge
```

```
et_xmlfile-2.0.0           |     pyhd8ed1ab_1          21 KB  conda-forge
gflags-2.2.2               |    h5888daf_1005         117 KB  conda-forge
glog-0.7.1                 |       hbabe93e_0         140 KB  conda-forge
greenlet-3.1.1             |  py312h2ec8cdc_1         232 KB  conda-forge
hdf5-1.14.4                |nompi_h2d575fe_105         3.8 MB  conda-forge
html5lib-1.1               |     pyhd8ed1ab_2          93 KB  conda-forge
joblib-1.4.2               |     pyhd8ed1ab_1         215 KB  conda-forge
libabseil-20240722.0       | cxx17_hbbce691_4         1.3 MB  conda-forge
libaec-1.1.3               |       h59595ed_0          35 KB  conda-forge
libarrow-18.1.0            |   hd595efa_7_cpu         8.4 MB  conda-forge
libarrow-acero-18.1.0      |   hcb10f89_7_cpu         598 KB  conda-forge
libarrow-dataset-18.1.0    |   hcb10f89_7_cpu         574 KB  conda-forge
libarrow-substrait-18.1.0  |   h08228c5_7_cpu         510 KB  conda-forge
libcrc32c-1.1.2            |       h9c3ff4c_0          20 KB  conda-forge
libcurl-8.11.1             |       h332b0f4_0         413 KB  conda-forge
libev-4.33                 |       hd590300_2         110 KB  conda-forge
libevent-2.1.12            |       hf998b51_1         417 KB  conda-forge
libgoogle-cloud-2.33.0     |       h2b5623c_1         1.2 MB  conda-forge
libgoogle-cloud-storage-2.33.0|       h0121fbd_1         766 KB  conda-forge
libgrpc-1.67.1             |       h25350d4_1         7.4 MB  conda-forge
libllvm14-14.0.6           |       hcd5def8_4        30.0 MB  conda-forge
libnghttp2-1.64.0          |       h161d5f1_0         632 KB  conda-forge
libparquet-18.1.0          |   h081d1f1_7_cpu         1.1 MB  conda-forge
libprotobuf-5.28.3         |       h6128344_1         2.8 MB  conda-forge
libre2-11-2024.07.02       |       hbbce691_2         205 KB  conda-forge
libssh2-1.11.1             |       hf672d98_0         297 KB  conda-forge
libthrift-0.21.0           |       h0e7cc3e_0         416 KB  conda-forge
libutf8proc-2.9.0          |       hb9d3cd8_1          80 KB  conda-forge
llvmlite-0.43.0            |  py312h374181b_1         3.3 MB  conda-forge
lxml-5.3.0                 |  py312he28fd5a_2         1.3 MB  conda-forge
lz4-c-1.10.0               |       h5888daf_1         163 KB  conda-forge
nomkl-1.0                  |       h5ca1d4c_0           4 KB  conda-forge
numba-0.60.0               |  py312h83e6fd3_0         5.4 MB  conda-forge
numexpr-2.10.2             |py312h6a710ac_100         191 KB  conda-forge
numpy-2.0.2                |  py312h58c1407_1         8.1 MB  conda-forge
openpyxl-3.1.5             |  py312h710cb58_1         680 KB  conda-forge
orc-2.0.3                  |       h12ee42a_2         1.1 MB  conda-forge
patsy-1.0.1                |     pyhd8ed1ab_1         182 KB  conda-forge
py-cpuinfo-9.0.0           |     pyhd8ed1ab_1          25 KB  conda-forge
pyarrow-18.1.0             |  py312h7900ff3_0          25 KB  conda-forge
pyarrow-core-18.1.0        |py312h01725c0_0_cpu         4.4 MB  conda-forge
pytables-3.10.2            |  py312hf8651a9_0         1.6 MB  conda-forge
re2-2024.07.02             |       h9925aae_2          26 KB  conda-forge
s2n-1.5.10                 |       hb5b8611_0         347 KB  conda-forge
scikit-learn-1.6.0         |  py312h7a48858_0        10.0 MB  conda-forge
scipy-1.15.0               |  py312h180e4f1_1        18.2 MB  conda-forge
seaborn-0.13.2             |       hd8ed1ab_3           7 KB  conda-forge
seaborn-base-0.13.2        |     pyhd8ed1ab_3         223 KB  conda-forge
snappy-1.2.1               |       h8bd8927_1          42 KB  conda-forge
sqlalchemy-2.0.36          |  py312h66e93f0_0         3.3 MB  conda-forge
statsmodels-0.14.4         |  py312hc0a28a1_0        11.5 MB  conda-forge
threadpoolctl-3.5.0        |     pyhc1e730c_0          23 KB  conda-forge
zlib-ng-2.2.3              |       h7955e40_0         106 KB  conda-forge
------------------------------------------------------------
Total:       138.7 MB

The following NEW packages will be INSTALLED:

aws-c-auth         conda-forge/linux-64::aws-c-auth-0.8.0-hb921021_15
aws-c-cal          conda-forge/linux-64::aws-c-cal-0.8.1-h1a47875_3
aws-c-common       conda-forge/linux-64::aws-c-common-0.10.6-hb9d3cd8_0
aws-c-compression  conda-forge/linux-64::aws-c-compression-0.3.0-h4e1184b_5
aws-c-event-stream conda-forge/linux-64::aws-c-event-stream-0.5.0-h7959bf6_11
aws-c-http         conda-forge/linux-64::aws-c-http-0.9.2-hefd7a92_4
```

```
aws-c-io              conda-forge/linux-64::aws-c-io-0.15.3-h831e299_5
aws-c-mqtt            conda-forge/linux-64::aws-c-mqtt-0.11.0-h11f4f37_12
aws-c-s3              conda-forge/linux-64::aws-c-s3-0.7.7-hf454442_0
aws-c-sdkutils        conda-forge/linux-64::aws-c-sdkutils-0.2.1-h4e1184b_4
aws-checksums         conda-forge/linux-64::aws-checksums-0.2.2-h4e1184b_4
aws-crt-cpp           conda-forge/linux-64::aws-crt-cpp-0.29.7-hd92328a_7
aws-sdk-cpp           conda-forge/linux-64::aws-sdk-cpp-1.11.458-hc430e4a_4
azure-core-cpp        conda-forge/linux-64::azure-core-cpp-1.14.0-h5cfcd09_0
azure-identity-cpp    conda-forge/linux-64::azure-identity-cpp-1.10.0-h113e628_0
azure-storage-blo~    conda-forge/linux-64::azure-storage-blobs-cpp-12.13.0-h3cf044e_1
azure-storage-com~    conda-forge/linux-64::azure-storage-common-cpp-12.8.0-h736e048_1
azure-storage-fil~    conda-forge/linux-64::azure-storage-files-datalake-cpp-12.12.0-ha633028_1
blosc                 conda-forge/linux-64::blosc-1.21.6-he440d0b_1
c-ares                conda-forge/linux-64::c-ares-1.34.4-hb9d3cd8_0
c-blosc2              conda-forge/linux-64::c-blosc2-2.15.2-h3122c55_1
et_xmlfile            conda-forge/noarch::et_xmlfile-2.0.0-pyhd8ed1ab_1
gflags                conda-forge/linux-64::gflags-2.2.2-h5888daf_1005
glog                  conda-forge/linux-64::glog-0.7.1-hbabe93e_0
greenlet              conda-forge/linux-64::greenlet-3.1.1-py312h2ec8cdc_1
hdf5                  conda-forge/linux-64::hdf5-1.14.4-nompi_h2d575fe_105
html5lib              conda-forge/noarch::html5lib-1.1-pyhd8ed1ab_2
joblib                conda-forge/noarch::joblib-1.4.2-pyhd8ed1ab_1
libabseil             conda-forge/linux-64::libabseil-20240722.0-cxx17_hbbce691_4
libaec                conda-forge/linux-64::libaec-1.1.3-h59595ed_0
libarrow              conda-forge/linux-64::libarrow-18.1.0-hd595efa_7_cpu
libarrow-acero        conda-forge/linux-64::libarrow-acero-18.1.0-hcb10f89_7_cpu
libarrow-dataset      conda-forge/linux-64::libarrow-dataset-18.1.0-hcb10f89_7_cpu
libarrow-substrait    conda-forge/linux-64::libarrow-substrait-18.1.0-h08228c5_7_cpu
libcrc32c             conda-forge/linux-64::libcrc32c-1.1.2-h9c3ff4c_0
libcurl               conda-forge/linux-64::libcurl-8.11.1-h332b0f4_0
libev                 conda-forge/linux-64::libev-4.33-hd590300_2
libevent              conda-forge/linux-64::libevent-2.1.12-hf998b51_1
libgoogle-cloud       conda-forge/linux-64::libgoogle-cloud-2.33.0-h2b5623c_1
libgoogle-cloud-s~    conda-forge/linux-64::libgoogle-cloud-storage-2.33.0-h0121fbd_1
libgrpc               conda-forge/linux-64::libgrpc-1.67.1-h25350d4_1
libllvm14             conda-forge/linux-64::libllvm14-14.0.6-hcd5def8_4
libnghttp2            conda-forge/linux-64::libnghttp2-1.64.0-h161d5f1_0
libparquet            conda-forge/linux-64::libparquet-18.1.0-h081d1f1_7_cpu
libprotobuf           conda-forge/linux-64::libprotobuf-5.28.3-h6128344_1
libre2-11             conda-forge/linux-64::libre2-11-2024.07.02-hbbce691_2
libssh2               conda-forge/linux-64::libssh2-1.11.1-hf672d98_0
libthrift             conda-forge/linux-64::libthrift-0.21.0-h0e7cc3e_0
libutf8proc           conda-forge/linux-64::libutf8proc-2.9.0-hb9d3cd8_1
llvmlite              conda-forge/linux-64::llvmlite-0.43.0-py312h374181b_1
lxml                  conda-forge/linux-64::lxml-5.3.0-py312he28fd5a_2
lz4-c                 conda-forge/linux-64::lz4-c-1.10.0-h5888daf_1
nomkl                 conda-forge/noarch::nomkl-1.0-h5ca1d4c_0
numba                 conda-forge/linux-64::numba-0.60.0-py312h83e6fd3_0
numexpr               conda-forge/linux-64::numexpr-2.10.2-py312h6a710ac_100
openpyxl              conda-forge/linux-64::openpyxl-3.1.5-py312h710cb58_1
orc                   conda-forge/linux-64::orc-2.0.3-h12ee42a_2
patsy                 conda-forge/noarch::patsy-1.0.1-pyhd8ed1ab_1
py-cpuinfo            conda-forge/noarch::py-cpuinfo-9.0.0-pyhd8ed1ab_1
pyarrow               conda-forge/linux-64::pyarrow-18.1.0-py312h7900ff3_0
pyarrow-core          conda-forge/linux-64::pyarrow-core-18.1.0-py312h01725c0_0_cpu
pytables              conda-forge/linux-64::pytables-3.10.2-py312hf8651a9_0
re2                   conda-forge/linux-64::re2-2024.07.02-h9925aae_2
s2n                   conda-forge/linux-64::s2n-1.5.10-hb5b8611_0
scikit-learn          conda-forge/linux-64::scikit-learn-1.6.0-py312h7a48858_0
scipy                 conda-forge/linux-64::scipy-1.15.0-py312h180e4f1_1
seaborn               conda-forge/noarch::seaborn-0.13.2-hd8ed1ab_3
seaborn-base          conda-forge/noarch::seaborn-base-0.13.2-pyhd8ed1ab_3
snappy                conda-forge/linux-64::snappy-1.2.1-h8bd8927_1
sqlalchemy            conda-forge/linux-64::sqlalchemy-2.0.36-py312h66e93f0_0
```

```
statsmodels         conda-forge/linux-64::statsmodels-0.14.4-py312hc0a28a1_0
threadpoolctl       conda-forge/noarch::threadpoolctl-3.5.0-pyhc1e730c_0
zlib-ng             conda-forge/linux-64::zlib-ng-2.2.3-h7955e40_0

The following packages will be DOWNGRADED:

numpy                                2.2.1-py312h7e784f5_0 --> 2.0.2-py312h58c1407_1


Proceed ([y]/n)? y


Downloading and Extracting Packages:

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

Question 1 (Downgrade NumPy)**.** *Why downgrade* `NumPy` *version?*
On Windows, substitute a carat `^` *for line continuation* `\` *used on Linux & macOS.*
Can update packages by using `conda update` command:

```
conda update package_name
```

`pip` also supports upgrades using `-upgrade` flag:

```
pip install --upgrade package_name
```

Have several opportunities to try out these commands throughout book.

Remark 5. *While can use both* `conda` *&* `pip` *to install packages, should avoid updating packages originally installed with conda using pip (& vice versa), as doing so can lead to environment problems. Recommend sticking to conda if can & falling back on pip only for packages that are unavailable with* `conda install`*.*

* Integrated Development Environments & Text Editors. When asked about standard development environment, almost always says "IPython plus a text editor." Typically write a program & iteratively test & debug each piece of it in IPython or Jupyter notebooks. Also useful to be able to play around with data interactively & visually verify that a particular set of data manipulations is doing right thing. Libraries like `pandas` & `NumPy` are designed to be productive to use in shell.
  When building software, however, some users may prefer to use a more richly featured integrated development environment (IDE) & rather than an editor like Emacs or Vim which provide a more minimal environment out of box. Some that you can explore:
  · PyDev (free), an IDE built on Eclipse platform
  · PyCharm from JetBrains (subscription-based for commercial users, free for open source developers)
  · Python Tools for Visual Studio (for Windows users)
  · Spyder (free), an IDE currently shipped with Anaconda
  · Komodo IDE (commercial)
  Due to popularity of Python, most text editors, like VS Code & Sublime Text 2, have excellent Python support.

- 1.5. Community & Conferences. Outside of an internet search, various scientific & data-related Python mailing lists are generally helpful & responsive to questions. Some to take a look at include:
  * pydata: A Google Group list for questions related to Python for data analysis & pandas
  * pystatsmodels: For statsmodels or pandas-related questions
  * Mailing list for scikit-learn `scikit-learn@python.org` & ML in Python, generally
  * numpy-discussion: For NumPy-related questions
  * scipy-user: For general SciPy or scientific Python questions

  Deliberately did not post URLs for these in case they change. They can be easily located via an internet search.
  Each year many conferences are held all over world for Python programmers. If would like to connect with other Python programmers who share interests, encourage to explore attending one, if possible. Many conferences have financial support available for those who cannot afford admission or travel to conference. Some to consider:
  * PyCon & EuroPython: 2 main general Python conferences in North America & Europe, resp.
  * SciPy & EuroSciPy: Scientific-computing-oriented conferences in North America & Europe, resp.
  * SciPy & EuroSciPy: Scientific-computing-oriented conferences in North America & Europe, resp.
  * PyData: A worldwide series of regional conferences a targeted at DS & data analysis use cases
  * International & regional PyCon conferences (see https://pycon.org for a complete listing)

- 1.6. Navigating This Book. If have never programmed in Python before, will want to spend some time in Chaps. 2–3, where have placed a condensed tutorial on Python language features & IPython shell & Jupyter notebooks. These things are prerequisite knowledge for remainder of book. If have Python experience already, may instead choose to skim or skip these chaps.
  Next, give a short introduction to key features of NumPy, leaving more advanced `NumPy` use for Appendix A. Then, introduce `pandas` & devote rest of book to data analysis topics applying `pandas, NumPy, matplotlib` (for visualization). Have structured material in an incremental fashion, though there is occasionally some minor crossover between chaps, with a few cases where concepts are used that haven't been introduced yet.
  While readers may have many different end goals for their work, tasks required generally fall into a number of different broad groups:
  * *Interacting with outside world*: Reading & writing with a variety of file formats & data stores
  * *Preparation*: Cleaning, munging, combining, normalizing, reshaping, slicing & dicing, & transforming data for analysis
  * *Transformation*: Applying mathematical & statistical operations to groups of datasets to derive new datasets (e.g., aggregating a large table by group variables)
  * *Modeling & computation*: Connecting your data to statistical models, ML algorithms, or other computational tools
  * *Presentation*: Creating interactive or static graphical visualizations or textual summaries
  * Code Examples. Most of code examples in book are shown with input & output as it should appear executed in IPython shell or in Jupyter notebooks:

    ```
    In [5]: CODE EXAMPLE
    Out[5]: OUTPUT
    ```

    When see a code example like this, intent is for you to type example code in `In` block in your coding environment & execute it by pressing `Enter` key (or `Shift-Enter` in Jupyter). Should see output similar to what is shown in `Out` block.
    Changed default console output settings in `NumPy` & `pandas` to improve readability & brevity throughout book. E.g., may see more digits of precision printed in numeric data. To exactly match output shown in book, can execute following Python code before running code examples:

    ```
    import numpy as np
    import pandas as pd
    pd.options.display.max_columns = 20
    pd.options.display.max_rows = 20
    pd.options.display.max_colwidth = 80
    np.set_printoptions(precision=4, suppress=True)
    ```

  * Data Examples. Datasets for examples in each chap are hosted in https://github.com/wesm/pydata-book (or in https://gitee.com/wesmckinn/pydata-book if cannot access GitHub). Can download this data either by using Git version control system on command line or by downloading a zip file of repository from website. If you run into problems, navigate to book website https://wesmckinney.com/book for up-to-date instructions about obtaining book materials. If download a zip file containing example datasets, must then fully extract contents of zip file to a directory & navigate to that directory from terminal before proceeding with running book's code examples:

    ```
    $ pwd
    /home/wesm/book-materials
    $ ls
    appa.ipynb ch05.ipynb ch09.ipynb ch13.ipynb  README.md
    ch02.ipynb ch06.ipynb ch10.ipynb COPYING     requirements.txt
    ch03.ipynb ch07.ipynb ch11.ipynb datasets
    ch04.ipynb ch08.ipynb ch12.ipynb examples
    ```

    Have made every effort to ensure: GitHub repository contains everything necessary to reproduce examples, but may have made some mistakes or omissions.
  * Import Conventions. Python community has adopted a number of naming conventions for commonly used modules:

    ```
    import numpy as np
    import matplotlib.pyplot as plt
    import pandas as pd
    import seaborn as sns
    import statsmodels as sm
    ```

    I.e., when see `np.arrange`, this is a reference to `arrange` function in NumPy. This is done because it's considered bad practice in Python software development to import everything `from numpy import *` from a large package like NumPy.
- 2. Python Language Basics, IPython, & Jupyter Notebooks. When wrote 1e of this book in 2011–2012, there were fewer resources available for learning about doing data analysis in Python. This was partially a chicken-&-egg problem; many

libraries that we know take for granted, like `pandas, scikit-learn, statsmodels`, were comparatively immature back then. Now in 2022, there is now a growing literature on DS, data analysis, & ML, supplementing prior works on general-purpose scientific computing geared toward computational scientists, physicists, & professionals in other research fields. There are also excellent books about learning Python programming language itself & becoming an effective software engineer.

As this book is intended as an introductory text in working with data in Python, feel valuable to have a self-contained overview of some of most important features of Python's built-in data structures & libraries from perspective of data manipulation. So, will only present roughly enough information in Chaps. 2–3 to enable you to follow along with rest of book.

Much of this book focuses on table-based analytics & data preparation tools for working with datasets that are small enough to fit on your personal computer. to use these tools you must sometimes do some wrangling to arrange messy data into a more nicely tabular (or *structured*) form. Fortunately, Python is an ideal language for doing this. Greater your facility with Python language & its built-in data types, easier it will be for you to prepare new datasets for analysis.

Some of tools in this book are best explored from a live IPython or Jupyter session. Once learn how to start up IPython & Jupyter session. Once learn how to start up IPython & Jupyter, recommend: follow along with examples so can experiment & try different things. As with any keyboard-driven console-like environment, developing familiarity with common commands is also part of learning curve.

Remark 6. *There are introductory Python concepts that this chap does not cover, like classes & object-oriented programming, which may find useful in your foray ()cuộc đột kích, cướp phá, xâm lược) into data analysis in Python.*
To deepen Python language knowledge, recommend: supplement this chap with official Python tutorial & potentially 1 of many excellent books on general-purpose Python programming. Some recommendations to get you started include:

- *Python Cookbook, 3e, by* David Beazley, Brian K. Jones
- *Fluent Python by* Luciano Ramalho
- *Effective Python, 2e, by* Brett Slatkin

- 2.1. Python Interpreter. Python is an *interpreted* language. Python interpreter runs a program by executing 1 statement at a time. Standard interactive Python interpreter can be invoked on command line with `python` command:

```
(pydata-book) nqbh@nqbh-dell:~$ python
Python 3.12.7 | packaged by conda-forge | (main, Oct  4 2024, 16:05:46) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

  `>>>` is *prompt* after which you'll type code expressions. To exit Python interpreter, can either type `exit()` or press Ctrl-D (works on Linux & macOS only).
  Running Python programs is as simple as calling `python` with a `.py` file as its 1st argument.
  While some Python programmers execute all of their Python code in this way, those doing data analysis or scientific computing make use of IPython, an enhanced Python interpreter, or Jupyter notebooks, web-based code notebooks originally created within IPython project. Give an introduction to using IPython & Jupyter in this chap & have included a deeper look at IPython functionality in Appendix A. When use `%run` command, IPython executes code in specified file in same process, enabling to explore results interactively when it's done:

```
(pydata-book) nqbh@nqbh-dell:~$ ipython
Python 3.12.7 | packaged by conda-forge | (main, Oct  4 2024, 16:05:46) [GCC 13.3.0]
IPython 8.31.0 -- An enhanced Interactive Python. Type '?' for help.
Hello world

In [2]:
```

  Default IPython prompt adopts numbered `In [2]:` style, cf. standard `>>>` prompt.
- 2.2. IPython Basics. Run with IPython shell & Jupyter notebook, & introduce to some of essential concepts.
  * Running IPython Shell. Can launch IPython shell on command line just like launching regular Python interpreter except with `ipython` command `ipython`. You can execute arbitrary Python statements by typing them & pressing Return (or Enter). When type just a variable into IPython, it renders a string representation of object:

```
In [5]: import numpy as np

In [6]: data = [np.random.standard_normal() for i in range(7)]

In [7]: data
Out[7]:
[-0.960515015233981,
0.29199995965351516,
0.656773965407049,
1.0319443387105414,
-0.15623460611892206,
```

```
-0.17214640580390445,
0.5260760636382895]
```

1st 2 lines are Python code statements; 2nd statement creates a variable named `data` that refers to a newly created Python dictionary. Last line prints value of `data` in console.
Many kinds of Python objects are formatted to be more readable, or *pretty-printed*, which is distinct from normal printing with `print`. If printed above `data` variable in standard Python interpreter, it would be much less readable:

```
>>> import numpy as np
>>> data = [np.random.standard_normal() for i in range(7)]
>>> print(data)
>>> data
[-0.5767699931966723, -0.1010317773535111, -1.7841005313329152,
-1.524392126408841, 0.22191374220117385, -1.9835710588082562,
-1.6081963964963528]
```

IPython also provides facilities to execute arbitrary blocks of code (via a somewhat glorified copy-&-paste approach) & whole Python scripts. Can also use Jupyter notebook to work with larger blocks of code.

* Running Jupyter Notebook. 1 of major components of Jupyter project is *notebook*, a type of interactive document for code, text (including Markdown), data visualizations, & other output. Jupyter notebook interacts with *kernels*, which are implementations of Jupyter interactive computing protocol specific to different programming languages. Python Jupyter kernel uses IPython system for its underlying behavior.
To start up Jupyter, run command `jupyter notebook` in a terminal:

```
$ jupyter notebook
[I 15:20:52.739 NotebookApp] Serving notebooks from local directory:
/home/wesm/code/pydata-book
[I 15:20:52.739 NotebookApp] 0 active kernels
[I 15:20:52.739 NotebookApp] The Jupyter Notebook is running at:
http://localhost:8888/?token=0a77b52fefe52ab83e3c35dff8de121e4bb443a63f2d...
[I 15:20:52.740 NotebookApp] Use Control-C to stop this server and shut down
all kernels (twice to skip confirmation).
Created new window in existing browser session.
To access the notebook, open this file in a browser:
file:///home/wesm/.local/share/jupyter/runtime/nbserver-185259-open.html
Or copy and paste one of these URLs:
http://localhost:8888/?token=0a77b52fefe52ab83e3c35dff8de121e4...
or http://127.0.0.1:8888/?token=0a77b52fefe52ab83e3c35dff8de121e4...
```

NQBH's:

```
(base) nqbh@nqbh-dell:~$ conda activate pydata-book
(pydata-book) nqbh@nqbh-dell:~$ jupyter notebook
[I 2025-01-10 15:13:58.486 ServerApp] jupyter_lsp | extension was successfully linked.
[I 2025-01-10 15:13:58.488 ServerApp] jupyter_server_terminals | extension was successfully linked.
[I 2025-01-10 15:13:58.491 ServerApp] jupyterlab | extension was successfully linked.
[I 2025-01-10 15:13:58.493 ServerApp] notebook | extension was successfully linked.
[I 2025-01-10 15:13:58.610 ServerApp] notebook_shim | extension was successfully linked.
[I 2025-01-10 15:13:58.622 ServerApp] notebook_shim | extension was successfully loaded.
[I 2025-01-10 15:13:58.623 ServerApp] jupyter_lsp | extension was successfully loaded.
[I 2025-01-10 15:13:58.624 ServerApp] jupyter_server_terminals | extension was successfully loaded.
[I 2025-01-10 15:13:58.624 LabApp] JupyterLab extension loaded from /home/nqbh/anaconda3/envs/pydat
[I 2025-01-10 15:13:58.624 LabApp] JupyterLab application directory is /home/nqbh/anaconda3/envs/py
[I 2025-01-10 15:13:58.624 LabApp] Extension Manager is 'pypi'.
[I 2025-01-10 15:13:58.661 ServerApp] jupyterlab | extension was successfully loaded.
[I 2025-01-10 15:13:58.663 ServerApp] notebook | extension was successfully loaded.
[I 2025-01-10 15:13:58.663 ServerApp] Serving notebooks from local directory: /home/nqbh
[I 2025-01-10 15:13:58.663 ServerApp] Jupyter Server 2.15.0 is running at:
[I 2025-01-10 15:13:58.663 ServerApp] http://localhost:8888/tree?token=6f7ecdf66d339bf97d3a73aa22ce
[I 2025-01-10 15:13:58.663 ServerApp]     http://127.0.0.1:8888/tree?token=6f7ecdf66d339bf97d3a73aa
[I 2025-01-10 15:13:58.663 ServerApp] Use Control-C to stop this server and shut down all kernels (
[C 2025-01-10 15:13:58.684 ServerApp]

To access the server, open this file in a browser:
file:///home/nqbh/.local/share/jupyter/runtime/jpserver-73029-open.html
Or copy and paste one of these URLs:
```

```
http://localhost:8888/tree?token=6f7ecdf66d339bf97d3a73aa22ceff2f9eb3957d2aa3d4a6
http://127.0.0.1:8888/tree?token=6f7ecdf66d339bf97d3a73aa22ceff2f9eb3957d2aa3d4a6
[I 2025-01-10 15:13:58.695 ServerApp] Skipped non-installed server(s): bash-language-server, docker
Gtk-Message: 15:13:58.798: Not loading module "atk-bridge": The functionality is provided by GTK na
[73110, Main Thread] WARNING: GTK+ module /snap/firefox/5561/gnome-platform/usr/lib/gtk-2.0/modules
GTK+ 2.x symbols detected. Using GTK+ 2.x and GTK+ 3 in the same process is not supported.: 'glib w

(firefox:73110): Gtk-WARNING **: 15:13:58.845: GTK+ module /snap/firefox/5561/gnome-platform/usr/li
GTK+ 2.x symbols detected. Using GTK+ 2.x and GTK+ 3 in the same process is not supported.
Gtk-Message: 15:13:58.845: Failed to load module "canberra-gtk-module"
[73110, Main Thread] WARNING: GTK+ module /snap/firefox/5561/gnome-platform/usr/lib/gtk-2.0/modules
GTK+ 2.x symbols detected. Using GTK+ 2.x and GTK+ 3 in the same process is not supported.: 'glib w

(firefox:73110): Gtk-WARNING **: 15:13:58.846: GTK+ module /snap/firefox/5561/gnome-platform/usr/li
GTK+ 2.x symbols detected. Using GTK+ 2.x and GTK+ 3 in the same process is not supported.
Gtk-Message: 15:13:58.846: Failed to load module "canberra-gtk-module"
```

On many platforms, Jupyter will automatically open in default web browser (unless start it with `-no-browser`). Otherwise, can navigate to HTTP address printed when started notebook, here http://localhost:8888/?token=0a77b52fefe52ab83e3c35dff8de121e4bb443a63f2d3055. See Fig. 2.1: Jupyter notebook landing page for what this looks like in Google Chrome.

Remark 7. *Many people use Jupyter as a local computing environment, but it can also be deployed on servers & accessed remotely. Won't cover those details here, but encourage to explore this topic on internet if it's relevant to your needs.*

To create a new notebook, click `New` button & select `Python 3` option. Should see sth like Fig. 2.2: Jupyter new notebook view. If this is 1st time, try clicking on empty code "cell" & entering a line of Python code. Then press Shift-Enter to execute it.

When save notebook (see `Save & Checkpoint` under notebook `File` menu), it creates a file with extension `.ipynb`: a self-contained file format containing all of content (including any evaluated code output) currently in notebook. These can be loaded & edited by other Jupyter users.

To rename an open notebook, click on notebook title at top of page & type new title, pressing `Enter` when finished.

To load an existing notebook, put file in same directory where started notebook process (or in a subfolder within it), then click name from landing page. Can try it out with notebooks from `wesm/pydata-book` repository on GitHub Fig. 2.3: Jupyter example view for an existing notebook.

When want to close a notebook, click File menu & select `Close & Halt`. If simply close browser tab, Python process associated with notebook will keep running in background.

While Jupyter notebook may feel like a distinct experience from IPython shell, nearly all of commands & tools in this chap can be used in either environment.

* Tab Completion. On surface, IPython shell looks like a cosmetically different version (phiên bản khác biệt về mặt thẩm mỹ) of standard terminal Python interpreter (invoked with `python`). 1 of major improvements over standard Python shell is *tab completion*, found in many IDEs or other interactive computing analysis environments. While entering expressions in shell, pressing `Tab` key will search namespace for any variables (objects, functions, etc.) matching characters you have typed so far & show results in a convenient drop-down menu:

```
In [1]: an_apple = 27
In [2]: an_example = 42
In [3]: an<Tab>
an_apple an_example  any
```

Note: IPython displayed both of 2 variables I defined, as well as built-in function `any`. Also, you can also complete methods & attributes on any object after typing a period:

```
In [3]: b = [1, 2, 3]

In [4]:  b.<Tab>
append()  count()   insert()  reverse()
clear()   extend()  pop()     sort()
copy()    index()   remove()
```

Same is true for modules:

```
In [1]:  import datetime

In [2]: datetime.
date          MAXYEAR       timedelta     UTC
datetime      MINYEAR       timezone
```

```
datetime_CAPI time          tzinfo
```

Remark 8. *IPython by default hides methods & attributes starting with underscores, e.g. magic methods & internal "private" methods & attributes, in order to avoid cluttering display (& confusing novice users – gây bối rối cho người dùng mới!). These, too, can be tab-completed, but must 1st type an underscore to see them. If prefer to always see such methods in tab completion, can change this setting in IPython configuration. See IPython documentation to find out how to do this.*

Tab completion works in many contexts outside of searching interactive namespace & completng object or module attributes. When typing anything that looks like a file path (even in a Python string), pressing Tab key will complete anything on your computer's filesystem matching what you've typed.
Combined with `%run` command, this functionality can save you many keystrokes.
Another area where tab completion saves time is in completion of function keyword arguments (including = sign!) Fig. 2.4: Autocomplete function keywords in a Jupyter notebook.
Have a closer look at functions in a little bit:

* Introspection. Using a question mark `?` before or after a variable will display some general information about object:

```
In [12]: b?
Type:        list
String form: [1, 2, 3]
Length:      3
Docstring:
Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list.
The argument must be an iterable if specified.

In [13]: ?b
Type:        list
String form: [1, 2, 3]
Length:      3
Docstring:
Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list.
The argument must be an iterable if specified.

In [14]: print?
Signature: print(*args, sep=' ', end='\n', file=None, flush=False)
Docstring:
Prints the values to a stream, or to sys.stdout by default.

sep
string inserted between values, default a space.
end
string appended after the last value, default a newline.
file
a file-like object (stream); defaults to the current sys.stdout.
flush
whether to forcibly flush the stream.
Type:      builtin_function_or_method
```

This is referred to as *object introspection*. If object is a function or instance method, docstring, if defined, will also be shown. Suppose we'd written following function (which you can reproduce in IPython or Jupyter):

```
def add_numbers(a, b):
    """
    Add two numbers together
    Returns
    -------
    the_sum : type of arguments
    """
return a + b
```

Then using `?` shows us docstring:

```
In [6]: add_numbers?
```

```
Signature: add_numbers(a, b)
Docstring:
Add two numbers together
Returns
-------
the_sum : type of arguments
File:    <ipython-input-9-6a548a216e27>
Type:    function
```

`?` has a final usage, which is for searching IPython namespace in a manner similar to standard Unix or Windows command line. A number of characters combined with wildcard `*` will show all names matching wildcard expression. E.g., could get a list of all functions in top-level NumPy namespace containing `load`: [Missing line: `np.loads` cf. book]

```
In [1]: import numpy as np

In [2]: np.*load*?
np.__loader__
np.load
np.loadtxt
```

- 2.3. Python Language Basics. Give an overview of essential Python programming concepts & language mechanics. In Chap. 3, go into more detail about Python data structures, functions, & other built-in tools.
  - Language Semantics. Python language design is distinguished by its emphasis on readability, simplicity, & explicitness. Some people go so far as to liken it to "executable pseudocode."
    - `Indentation, not braces.` Python uses whitespace (tabs or spaces) to structure code instead of using braces as in many other languages like R, C++, Java, & Perl. Consider a `for` loop from a sorting algorithm:

```
for x in array:
    if x < pivot:
        less.append(x)
    else:
        greater.append(x)
```

      A colon denotes start of an indented code block after which all of code must be indented by same amount until end of block.
      Love it or hate it, significant whitespace is a fact of life for Python programmers. While it may seem foreign at 1st, will hopefully grow accustomed to it in time.

      Remark 9. *Strong recommend using* 4 spaces *as your default indentation & replacing tabs with 4 spaces. Many text editors have a setting that will replace tab stops with spaces automatically insert 4 spaces on new lines following a colon & replace tabs by 4 spaces.*

      As you can see by now, Python statements also do not need to be terminated by semicolons. Semicolons can be used, however, to separate multiple statements on a single line:

```
a = 5; b = 6; c = 7
```

      Putting multiple statements on 1 line is generally discouraged in Python as it can make code less readable.
    - Everything is an object. An important characteristic of Python language is consistency of its *object model*. Every number, string, data structure, function, class, module, & so on exists in Python interpreter in its own "box," which is referred to as a *Python object*. Each object has an associated *type* (e.g., *integer, string*, or *function*) & internal data. In practice this makes language very flexible, as even functions can be treated like any other object.
    - Comments. Any text preceded by hash mark (pound sign) `#` is ignored by Python interpreter. Often used to add comments to code. At times may also want to exclude certain blocks of code without deleting them. 1 solution: *comment out* code:

```
results = []
for line in file_handle:
    # keep the empty lines for now
    # if len(line) == 0:
    #   continue
    results.append(line.replace("foo", "bar"))
```

      Comments can also occur after a line of executed code. While some programmers prefer comments to be placed in line preceding a particular line of code, this can be useful at times:

```
print("Reached this line") # Simple status report
```

· Function & object method calls. Call functions using parentheses & passing 0 or more arguments, optionally assigning returned value to a variable:

```
result = f(x, y, z)
g()
```

Almost every object in Python has attached functions, known as *methods*, that have access to object's internal contents. Can call them using following syntax:

```
obj.some_method(x, y, z)
```

Functions can take both *positional & keyword* arguments:

```
result = f(a, b, c, d=5, e="foo")
```

· Variables & argument passing. When assigning a variable (or *name*) in Python, you are creating a *reference* to object shown on RHS of equals sign. In practical terms, consider a list of integers:

```
In [8]: a = [1, 2, 3]
```

Suppose: assign `a` to a new variable `b`:

```
In [9]: b = a

In [10]: b
Out[10]: [1, 2, 3]
```

In some languages, assignment if `b` will cause data `[1, 2, 3]` to be copied. In Python, `a` & `b` actually now refer to same object, original list `[1, 2, 3]` (see Fig. 2.5: 2 references for same object for a mock-up). Can prove this by appending an element to `a` & then examining `b`:

```
In [11]: a.append(4)

In [12]: b
Out[12]: [1, 2, 3, 4]
```

Understanding semantics of references in Python, & when, how, & why data is copied, is especially critical when you are working with larger datasets in Python.
– Hiểu được ngữ nghĩa của các tham chiếu trong Python, & khi nào, như thế nào, & lý do tại sao dữ liệu được sao chép, đặc biệt quan trọng khi bạn làm việc với các tập dữ liệu lớn hơn trong Python.

Remark 10. *Assignment is also referred to as* binding*, as we are binding a name to an object. Variable names that have been assigned may occasionally be referred to as bound variables.*

When pass objects as arguments to a function, new local variables are created referencing original objects without any copying. If bind a new object to a variable inside a function, that will not overwrite a variable of same name in "scope" outside of function ("parent scope"). Therefore possible to alter internals of a mutable argument. Suppose had following function:

```
In [13]: def append_element(some_list, element):
   ....:     some_list.append(element)
```

Then have:

```
In [14]: data = [1, 2, 3]

In [15]: append_element(data, 4)

In [16]: data
Out[16]: [1, 2, 3, 4]
```

· Dynamic references, strong types. Variables in Python have no inherent type associated with them; a variable can refer to a different type of object simply by doing an assignment. There is no problem with following:

```
In [17]: a = 5

In [18]: type(a)
Out[18]: int

In [19]: a = "foo"

In [20]: type(a)
```

```
Out[20]: str
```

Variables are names for objects within a particular namespace; type information is stored in object itself. Some observers might hastily conclude: Python is not a "typed language." Wrong: consider:

```
In [21]: "5" + 5
---------------------------------------------------------------------------
TypeError
Traceback (most recent call last)
<ipython-input-21-7fe5aa79f268> in <module>
----> 1 "5" + 5
TypeError: can only concatenate str (not "int") to str
```

In some languages, string `'5'` might get implicitly converted (or *cast*) to an integer, thus yielding 10. In other languages integer `5` might be cast to a string, yielding concatenated string `'55'`. In Python, such implicit casts are not allowed. In this regard, say: Python is a *strongly typed* language, i.e., every object has a specific type (or *class*), & implicit conversions will occur only in certain permitted circumstances, e.g.:

```
In [22]: a = 4.5

In [23]: b = 2

# String formatting, to be visited later
In [24]: print(f"a is {type(a)}, b is {type(b)}")
a is <class 'float'>, b is <class 'int'>

In [25]: a / b
Out[25]: 2.25
```

Here, even though `b` is an integer, it is implicitly converted to a float for division operation.
Knowing type of an object is important, & useful to be able to write functions that can handle many different kinds of input. Can check: an object is an instance of a particular type using `isinstance` function:

```
In [26]: a = 5

In [27]: isinstance(a, int)
Out[27]: True
```

`isinstance` can accept a type of types if want to check: an object's type is among those present in tuple:

```
In [28]: a = 5; b = 4.5

In [29]: isinstance(a, (int, float))
Out[29]: True

In [30]: isinstance(b, (int, float))
Out[30]: True
```

- Attributes & methods. Objects n Python typically have both attributes (other Python objects stored "inside" object) & methods (functions associated with an object that can have access to object's internal data). Both of them are accessed via syntax `obj.attribute_name`:

```
In [1]: a = "foo"

In [2]: a.<Press Tab>
capitalize()   encode()       format()       isalpha()      isidentifier() isspace()      ljust()
casefold()     endswith()     format_map()   isascii()      islower()      istitle()      lower()
center()       expandtabs()   index()        isdecimal()    isnumeric()    isupper()      lstrip()
count()        find()         isalnum()      isdigit()      isprintable()  join()         maketrans()
```

Attributes & methods can also be accessed by name via `getattr` function:

```
In [32]: getattr(a, "split")
Out[32]: <function str.split(sep=None, maxsplit=-1)>
```

While will not extensively use functions `getattr` & related functions `hasattr` & `setattr` in this book, they can be used very effectively to write generic, reusable code.

- Duck typing. Often may not care about type of an object but rather only whether it has certain methods or behavior. This is sometimes called *duck typing*, after saying "If it walks like a duck & quacks like a duck, then it's a duck." E.g., you can verify: an object is iterable if it implements *iterator protocol*. For many objects, this means it has an `__iter__` "magic method," though an alternative & better way to check is to try using `iter` function:

```
In [33]: def isiterable(obj):
   ....:     try:
   ....:         iter(obj)
   ....:         return True
   ....:     except TypeError: # not iterable
   ....:         return False
```

  This function would return `True` for strings as well as most Python collection types:

```
In [34]: isiterable("a string")
Out[34]: True

In [35]: isiterable([1, 2, 3])
Out[35]: True

In [36]: isiterable(5)
Out[36]: False
```

- Imports. In Python, a *module* is simply a file with `.py` extension containing Python code. Suppose had following module:

```
# some_module.py
PI = 3.14159

def f(x):
    return x + 2

def g(a, b):
    return a + b
```

  If wanted to access variables & functions defined in `some_module.py`, from another file in same directory, could do:

```
import some_module
result = some_module.f(5)
pi = some_module.PI
```

  Or alternately:

```
from some_module import g, PI
result = g(5, PI)
```

  By using `as` keyword, can give imports different variable names:

```
import some_module as sm
from some_module import PI as pi, g as gf

r1 = sm.f(pi)
r2 = gf(6, pi)
```

- Binary operators & comparisons. Most of binary math operations & comparisons use familiar mathematical syntax used in other programming languages:

```
In [37]: 5 - 7
Out[37]: -2

In [38]: 12 + 21.5
Out[38]: 33.5

In [39]: 5 <= 2
Out[39]: False
```

  See Table 2.1: Binary operators for all of available binary operators.

- 3. Built-In Data Structures, Functions, & Files.

- 4. `NumPy` Basics: Arrays & Vectorized Computation.
- 5. Getting Started with pandas.
- 6. Data Loading, Storage, & File Formats.
- 7. Data Cleaning & Preparation.
- 8. Data Wrangling: Join, Combine, & Reshape.
- 9. Plotting & Visualization.
- 10. Data Aggregation & Group Operations.
- 11. Time Series.
- 12. Introduction to Modeling Libraries in Python.
- 13. Data Analysis Examples.
- A. Advanced NumPy.
- B. More on IPython System.

2 Miscellaneous

Tài liệu

[McK22] Wes McKinney. *Python for Data Analysis: Data Wrangling with pandas, NumPy, & Jupyter*. 3rd edition. O'Reilly Media Publisher, 2022, p. 579.