

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN



THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH
HỌC KỲ I, NĂM HỌC 2024 – 2025

**TÌM HIỂU JUPYTER BOOK
VÀ BIÊN SOẠN TÀI LIỆU
HƯỚNG DẪN LẬP TRÌNH TRÊN
THU VIỆN NUMPY VÀ PANDAS**

Giáo viên hướng dẫn:
TS. Nguyễn Bảo Ân

Sinh viên thực hiện:
Họ tên: Nguyễn Huỳnh Yến Như
MSSV:110122018
Lớp: DA22TTA

Trà Vinh, tháng 01 năm 2025

KHOA KỸ THUẬT VÀ CÔNG NGHỆ
BỘ MÔN CÔNG NGHỆ THÔNG TIN



THỰC TẬP ĐỒ ÁN CƠ SỞ NGÀNH
HỌC KỲ I, NĂM HỌC 2024 – 2025

**TÌM HIỂU JUPYTER BOOK
VÀ BIÊN SOẠN TÀI LIỆU
HƯỚNG DẪN LẬP TRÌNH TRÊN
THU VIỆN NUMPY VÀ PANDAS**

Giáo viên hướng dẫn:
TS. Nguyễn Bảo Ân

Sinh viên thực hiện:
Họ tên: Nguyễn Huỳnh Yến Như
MSSV:110122018
Lớp: DA22TTA

Trà Vinh, tháng 01 năm 2025

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Trà Vinh, ngày tháng năm
Giáo viên hướng dẫn
(Ký tên và ghi rõ họ tên)

NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG

Trà Vinh, ngày tháng năm

Thành viên hội đồng
(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Em xin gửi lời cảm ơn chân thành nhất đến thầy Nguyễn Bảo Ân đã hỗ trợ, hướng dẫn em trong suốt thời gian thực hiện đồ án này. Nhờ có sự hướng dẫn tận tình của thầy em đã hoàn thành đồ án một cách tốt nhất, hơn thế nữa những lời chỉ dẫn, góp ý của thầy đã giúp em có thêm nhiều kiến thức mới. Em biết ơn sự tận tình chỉ bảo và giải đáp mọi khó khăn của thầy giúp em vượt qua mọi thử thách trong thời gian thực hiện đồ án. Em sẽ luôn ghi nhớ và áp dụng những kiến thức quý báu mà thầy đã truyền đạt.

Một lần nữa em xin gửi lời cảm ơn sâu sắc đến thầy và hy vọng sẽ có cơ hội học hỏi thêm từ thầy trong những dự án sau này.

MỤC LỤC

TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH	7
MỞ ĐẦU	8
CHƯƠNG 1: TỔNG QUAN	10
1.1 Giới thiệu về Python và vai trò của thư viện NumPy, Pandas.....	10
1.2 Vai trò của tài liệu học lập trình tương tác	10
1.3 Lý do chọn Jupyter Book làm công cụ biên soạn	10
1.4 Vấn đề đặt ra và mục tiêu của nghiên cứu	11
CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT	12
2.1 Jupyter Book	12
2.1.1 Giới thiệu về Jupyter Book	12
2.1.1 Các tính năng chính của Jupyter Book	12
2.1.2 Cấu trúc của một dự án Jupyter Book	12
2.1.3 Cách tạo 1 dự án bằng Jupyter Book	13
2.2 Giới thiệu về Markdown cách hoạt động và cách sử dụng	14
2.2.1 Giới thiệu về Markdown	14
2.2.2 Cách hoạt động của Markdown	15
2.2.3 Cách sử dụng Markdown	15
2.2.4 Cú pháp của Markdown	16
2.3 Giới thiệu về ngôn ngữ YAML	19
2.3.1 Giới thiệu YAML	19
2.3.2 Collection types	19
2.3.3 Mẹo viết YAML	20
CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU	21
3.1 Tạo tài liệu với Jupyter Book	21
3.1.1 Cấu trúc tài liệu	21
3.1.2 Các file của tài liệu	21
3.2 Quá trình biên soạn	22
3.2.1 Tạo mục lục	22
3.2.2 Viết nội dung cho từng chương	24
CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU	27
4.1 Kết quả tài liệu trên Jupyter Book	27
4.1.1 Phản mục lục	27
4.1.2 Các trang tài liệu	28
4.2 Kết quả tài liệu xuất bằng PDF	30
4.3 Kết quả tài liệu trên Github	31
CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	32
5.1 Kết quả đạt được	32
5.2 Hướng phát triển	32
DANH MỤC TÀI LIỆU THAM KHẢO	33

DANH MỤC HÌNH ẢNH

Hình 2.1 Minh họa tạo chương	13
Hình 2.2 Minh họa file _toc.yml	14
Hình 2.3 Heading	17
Hình 2.4 Biểu diễn cặp khóa giá trị.....	19
Hình 2.5 Biểu diễn danh sách phần tử	20
Hình 2.6 Cấu trúc bảng đồ lồng nhau	20
Hình 3.1 Mô tả cấu trúc tài liệu.....	21
Hình 4.1 Mục lục Jupyter Book	27
Hình 4.2 Trang mục lục của tài liệu.....	28
Hình 4.3 Minh họa tài liệu	29
Hình 4.4 Minh họa các ví dụ của tài liệu	29
Hình 4.5 Các bảng của tài liệu	30
Hình 4.6 Bài tập và kết quả.....	30
Hình 4.8 Tài liệu trên GitHub	31

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Ý nghĩa
HTML	HyperText Markup Language
XHTML	eXtensible HyperText Markup Language
PDF	Portable Document Format
YAML	YAML Ain't Markup Language
JSON	JavaScript Object Notation

TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH

Đề tài "Tìm hiểu Jupyter Book và biên soạn tài liệu hướng dẫn lập trình Python trên thư viện NumPy và Pandas" tập trung nghiên cứu cách sử dụng Jupyter Book - một công cụ mã nguồn mở hỗ trợ xây dựng tài liệu học lập trình tương tác, dễ truy cập và cập nhật. Mục tiêu của đề tài là biên soạn một tài liệu hướng dẫn chi tiết nhằm hỗ trợ người học hiểu và ứng dụng các thư viện Python phổ biến, bao gồm NumPy và Pandas.

Phương pháp tiếp cận:

Nghiên cứu lý thuyết về Jupyter Book, NumPy và Pandas từ các tài liệu và khóa học trực tuyến.

Thực nghiệm bằng cách xây dựng một dự án tài liệu mẫu với Jupyter Book, kết hợp các đoạn mã minh họa, biểu đồ.

Cách giải quyết vấn đề:

Cài đặt và cấu hình Jupyter Book trên hệ thống cá nhân để hiểu cách hoạt động của công cụ này.

Phát triển các nội dung hướng dẫn cơ bản về lập trình Python với NumPy và Pandas, bao gồm: khái niệm và cú pháp cơ bản, các ứng dụng thực tế như xử lý dữ liệu, tính toán hiệu quả và phân tích thông tin.

Tích hợp mã nguồn Python, ví dụ minh họa và giải thích vào tài liệu Jupyter Book, đồng thời xuất bản tài liệu dưới dạng web hoặc file PDF để tiện cho người dùng.

Kết quả đạt được:

Tài liệu hướng dẫn lập trình Python với NumPy và Pandas được biên soạn thành công, bao gồm các phần như giới thiệu, thực hành cơ bản và các ví dụ ứng dụng thực tế.

Jupyter Book được ứng dụng hiệu quả để tạo ra tài liệu học lập trình tương tác, thân thiện với người dùng và dễ dàng chia sẻ qua internet.

MỞ ĐẦU

Lý do chọn đề tài:

Hiện nay, Python đã trở thành một trong những ngôn ngữ lập trình phổ biến nhất nhờ sự linh hoạt, dễ học và khả năng ứng dụng rộng rãi, đặc biệt trong lĩnh vực khoa học dữ liệu. Các thư viện NumPy và Pandas đóng vai trò quan trọng trong xử lý và phân tích dữ liệu, nhưng việc tìm kiếm tài liệu học tập rõ ràng và dễ tiếp cận vẫn là một thách thức với nhiều người mới bắt đầu.

Jupyter Book là một công cụ mã nguồn mở mạnh mẽ, giúp biên soạn tài liệu học tập tương tác, kết hợp lý thuyết và thực hành một cách hiệu quả. Tuy nhiên, tại Việt Nam, ứng dụng Jupyter Book để xây dựng tài liệu học lập trình vẫn còn hạn chế. Vì vậy, việc nghiên cứu và ứng dụng Jupyter Book để biên soạn tài liệu hướng dẫn lập trình Python trên NumPy và Pandas không chỉ đáp ứng nhu cầu học tập mà còn góp phần thúc đẩy phương pháp giảng dạy mới trong lĩnh vực khoa học dữ liệu.

Mục đích nghiên cứu

Tìm hiểu và ứng dụng công cụ Jupyter Book để biên soạn tài liệu học tập.

Cung cấp hướng dẫn lập trình Python cơ bản và nâng cao với các thư viện NumPy và Pandas.

Xây dựng một tài liệu học tập tương tác, dễ hiểu, và có thể truy cập trên nhiều nền tảng, hỗ trợ người học tự học và thực hành hiệu quả.

Đối tượng nghiên cứu

Công cụ Jupyter Book và các tính năng của nó trong việc biên soạn tài liệu học tập.

Các thư viện Python, cụ thể là NumPy và Pandas, cùng các ứng dụng của chúng trong xử lý và phân tích dữ liệu.

Phạm vi nghiên cứu

Nội dung nghiên cứu tập trung vào các khái niệm và ứng dụng cơ bản của NumPy và Pandas trong xử lý dữ liệu. Các nội dung nâng cao hơn sẽ được đề xuất cho các hướng phát triển trong tương lai.

CHƯƠNG 1: TỔNG QUAN

1.1 Giới thiệu về Python và vai trò của thư viện NumPy, Pandas

Python là một ngôn ngữ lập trình mạnh mẽ, dễ học và đã trở thành công cụ không thể thiếu trong lĩnh vực khoa học dữ liệu, học máy, và trí tuệ nhân tạo. Hai thư viện quan trọng là NumPy và Pandas được thiết kế để xử lý dữ liệu nhanh chóng, cung cấp các tính năng mạnh mẽ giúp người dùng thao tác và phân tích dữ liệu một cách hiệu quả.

NumPy: Cung cấp các công cụ mạnh mẽ cho việc tính toán số học, xử lý mảng n-dimensional và các phép toán ma trận.

Pandas: Tập trung vào xử lý dữ liệu theo dạng bảng (DataFrame), với các tính năng như truy vấn, lọc, và thao tác dữ liệu nhanh chóng.

Các thư viện này đã và đang được sử dụng rộng rãi trong các lĩnh vực như tài chính, kinh tế, y tế, và nghiên cứu khoa học.

1.2 Vai trò của tài liệu học lập trình tương tác

Với nhu cầu học tập ngày càng tăng, các tài liệu học tập truyền thống đôi khi không đủ sức hấp dẫn và thiếu tính thực hành. Tài liệu học lập trình tương tác, chẳng hạn như tài liệu được biên soạn bằng Jupyter Book, cung cấp một cách tiếp cận mới: kết hợp giữa lý thuyết và thực hành ngay trong tài liệu, hỗ trợ học tập mọi lúc, mọi nơi với các nền tảng web, giúp người học dễ dàng thử nghiệm và thực hành trực tiếp trên nội dung tài liệu.

1.3 Lý do chọn Jupyter Book làm công cụ biên soạn

Jupyter Book là một nền tảng mã nguồn mở được thiết kế để tạo ra tài liệu học tập tương tác với nhiều ưu điểm:

Hỗ trợ Markdown và Jupyter Notebook: Cho phép trình bày nội dung theo định dạng đẹp mắt, đồng thời tích hợp mã nguồn chạy trực tiếp.

Khả năng mở rộng: Hỗ trợ tích hợp nhiều công cụ phân tích dữ liệu và trình bày đồ thị.

Xuất bản dễ dàng: Tài liệu có thể được xuất bản trực tuyến, dưới dạng HTML hoặc PDF.

Chính vì vậy, Jupyter Book là một công cụ lý tưởng để xây dựng tài liệu học lập trình về NumPy và Pandas.

1.4 Vấn đề đặt ra và mục tiêu của nghiên cứu

Mặc dù có nhiều tài liệu học tập về Python và các thư viện dữ liệu, nhưng nhiều tài liệu chỉ cung cấp lý thuyết mà thiếu thực hành tương tác, nội dung phân tán, không được tổ chức hệ thống, gây khó khăn cho người mới bắt đầu. Đề tài này hướng đến việc giải quyết các vấn đề trên bằng cách biên soạn tài liệu học tập sử dụng Jupyter Book, cung cấp một nguồn tài liệu dễ học, dễ hiểu và tích hợp các bài tập thực hành.

CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

2.1 Jupyter Book

2.1.1 Giới thiệu về Jupyter Book

Jupyter Book là một công cụ mã nguồn mở được phát triển bởi cộng đồng Project Jupyter. Nó được thiết kế để xây dựng tài liệu học thuật, giáo trình, hoặc sách hướng dẫn tương tác với các tính năng mạnh mẽ như: hỗ trợ nội dung động, bao gồm mã nguồn thực thi trực tiếp; tích hợp nội dung đa phương tiện như hình ảnh, biểu đồ và video; dễ dàng chia sẻ tài liệu qua nền tảng web hoặc xuất sang định dạng khác như PDF, HTML; Jupyter Book cho phép kết hợp lý thuyết và thực hành, tạo ra một trải nghiệm học tập phong phú và tương tác hơn cho người học.

2.1.1 Các tính năng chính của Jupyter Book

Hỗ trợ Markdown và Jupyter Notebook: Markdown cho phép trình bày văn bản theo phong cách đẹp mắt, hỗ trợ tiêu đề, danh sách, bảng, liên kết, và hình ảnh. Jupyter Notebook tích hợp các khối mã (code cells) và kết quả thực thi trực tiếp vào nội dung sách, giúp người đọc có thể thử nghiệm mã ngay trong tài liệu.

Cấu trúc tổ chức tài liệu linh hoạt: Jupyter Book hỗ trợ tổ chức nội dung dưới dạng chương, mục, và tiểu mục. Các tệp tài liệu có thể được liên kết thông qua cấu trúc thư mục rõ ràng, giúp dễ dàng điều hướng.

Hỗ trợ nhiều định dạng đầu ra: Jupyter Book có thể xuất tài liệu sang HTML (phổ biến để xuất bản trực tuyến), PDF (dễ dàng chia sẻ hoặc in ấn), EPUB (phù hợp để đọc trên các thiết bị di động).

Tích hợp công cụ tương tác: Interactive widgets (hỗ trợ các tiện ích tương tác như thanh trượt, hộp kiểm, và biểu đồ động), Live Code (người dùng có thể chạy mã trực tiếp trên tài liệu nếu tích hợp với dịch vụ như Binder hoặc JupyterHub).

2.1.2 Cấu trúc của một dự án Jupyter Book

Một dự án Jupyter Book điển hình bao gồm các thành phần sau:

_config.yml: Tệp cấu hình chính, dùng để tùy chỉnh tên sách, thông tin tác giả, định dạng đầu ra, và các cài đặt khác.

_toc.yml: Tệp định nghĩa mục lục, xác định cấu trúc các chương, mục của sách.

Content Files: Các tệp nội dung chính, được viết bằng định dạng Markdown (.md) hoặc Jupyter Notebook (.ipynb).

Assets Folder: Chứa các tài nguyên bổ sung như hình ảnh, dữ liệu, hoặc mã nguồn đi kèm tài liệu.

2.1.3 Cách tạo 1 dự án bằng Jupyter Book

Cài đặt Jupyter Book

Jupyter Book yêu cầu Python và pip để cài đặt. Hãy đảm bảo đã cài đặt Python (phiên bản 3.6 trở lên) trên hệ thống.

Mở terminal hoặc command prompt và chạy lệnh sau để cài đặt Jupyter Book:

```
pip install -U jupyter-book
```

Tạo cấu trúc sách

Sau khi cài đặt xong, cần khởi tạo cấu trúc cơ bản cho Jupyter Book:

Viết nội dung

Thêm một tệp chapter1.md vào thư mục mybook/ và viết nội dung

Markdown như sau:

```
❶ tongquan.md > abc # Chương 1: Tổng Quan về NumPy và Pandas
❷ # Chương 1: Tổng Quan về NumPy và Pandas
❸ NumPy và Pandas là hai thư viện mạnh mẽ trong Python, rất quan trọng cho việc phân tích dữ liệu, xử lý số liệu và khoa học dữ liệu. Cả hai đều được sử dụng rộng rãi trong cộng đồng khoa học dữ liệu và cung cấp các công cụ và hàm mạnh mẽ để làm việc với dữ liệu một cách hiệu quả và linh hoạt.
❹
❺ ## NumPy
❻ **NumPy (Numerical Python)** là thư viện cơ bản cho tính toán khoa học trong Python. Một số tính năng chính của NumPy bao gồm:
❼
```

Hình 2.1 Minh họa tạo chương

Cập nhật mục lục:

Mở tệp `_toc.yml` và thêm đường dẫn đến chương mới:

```
! _toc.yml > [ ]chapters > {} 3 > title
1   # _toc.yml
2
3   format: jb-book
4   root: intro
5
6   chapters:
7   | - file: mucluc.md
8   |   title: Mục lục
9   | - file: tongquan.md
10  |   title: Tổng quan về NumPy và Pandas
11  | - file: Numpy.md
12  |   title: NumPy Cơ Bản Mảng và Tính Toán Vector
13  | sections:
```

Hình 2.2 Minh họa file `_toc.yml`

Xây dựng sách

Khi đã hoàn thành nội dung, có thể xây dựng sách thành định dạng HTML bằng lệnh sau:

```
jupyter-book build mybook/
```

Quá trình này bao gồm:

Đọc các tệp Markdown và Jupyter Notebook.

Biên dịch và xuất tài liệu dưới dạng HTML.

Kiểm tra và xuất bản

Sau khi chạy lệnh trên, sách sẽ được xuất ra thư mục `_build/html`.

```
python -m http.server 8000
```

2.2 Giới thiệu về Markdown cách hoạt động và cách sử dụng

2.2.1 Giới thiệu về Markdown

Markdown là một ngôn ngữ đánh dấu. Ngôn ngữ đánh dấu rất đơn giản là một cách để làm cho một vài đoạn văn bản có ý nghĩa khác với các đoạn khác. Markdown đã được tạo ra vào năm 2004 bởi John Gruber với sự đóng góp đáng kể từ Aaron Swartz, với mục đích cho phép người sử dụng “để viết các định dạng văn bản đơn giản dễ đọc, và tùy chọn chuyển đổi nó thành các mã XHTML hợp lệ (hoặc HTML)”. Lấy cảm hứng từ các văn bản đơn giản trong email như `settext`, ngôn ngữ

được thiết kế để có thể đọc như - là, không nhìn nó được đánh dấu với thẻ hoặc cú pháp định dạng, không giống như các văn bản đã được định dạng với một ngôn ngữ Markup , chẳng hạn như HTML, trong đó có thẻ rõ ràng và cú pháp định dạng. Markdown là một cú pháp định dạng cho văn bản có thể được đọc bởi con người và có thể dễ dàng chuyển đổi sang HTML. Gruber đã viết một văn bản Perl - “Markdown.pl”, nó là một văn bản khác đi với XHTML hay HTML bằng việc thay thế các cú pháp thẻ bằng những cú pháp dễ dàng hơn nhưng chức năng thì tương tự nhau. Nó có thể được sử dụng như là một văn bản độc lập, như là một plugin cho Blosxom hay Movable Type, hoặc như một bộ lọc văn bản cho BBEdit. Các phần mềm sử dụng ngôn ngữ Markdown: GitHub, GitBook, Reddit, Diaspora, Stack Overflow, OpenStreetMap và các ứng dụng khác. Tệp tin: Một tài liệu Markdown là một file văn bản với phần mở rộng là .md. Có thể mở tệp tin markdown bằng một trình soạn thảo bất kỳ trên máy tính.[1]

2.2.2 Cách hoạt động của Markdown

Markdown hoạt động dựa trên việc sử dụng các ký hiệu định dạng đơn giản mà con người có thể đọc được ngay trong văn bản thuận túy. Khi mở file Markdown bằng phần mềm hỗ trợ, nội dung được chuyển đổi tự động sang HTML hoặc các giao diện khác thông qua một trình biên dịch tích hợp. Điều này giúp văn bản Markdown hiển thị đẹp mắt như giao diện web. Nếu không dùng trình hỗ trợ, file chỉ hiện dưới dạng văn bản thô, giống như khi mở bằng Notepad hoặc TextEdit. Chính sự đơn giản trong cú pháp và khả năng chuyển đổi linh hoạt đã khiến Markdown trở thành lựa chọn lý tưởng cho việc viết tài liệu và tạo nội dung trực tuyến.

2.2.3 Cách sử dụng Markdown

Để sử dụng Markdown, nội dung được soạn thảo trong trình chỉnh sửa văn bản bất kỳ và lưu với phần mở rộng **.md** hoặc **.markdown**. Các ký hiệu như dấu thẳng (#) để tạo tiêu đề, dấu sao (*) để làm danh sách, hoặc dấu ngoặc vuông kết hợp ngoặc đơn để chèn liên kết giúp định dạng nhanh chóng mà không cần giao diện phức tạp. Khi cần hiển thị nội dung dưới dạng HTML hoặc chuyển đổi sang định dạng khác như PDF, file Markdown được mở bằng phần mềm hỗ trợ như

Obsidian, Visual Studio Code hoặc MarkText, cho phép chỉnh sửa và xuất nội dung một cách trực quan.

2.2.4 Cú pháp của Markdown

Heading

Markdown hỗ trợ 2 kiểu viết tiêu đề tài liệu: Setext và ATX.

Với kiểu Setext: ta sử dụng ký tự = và - gạch chân dưới tiêu đề, sử dụng cho 2 thẻ h1 và h2.

Với kiểu ATX sử dụng ký tự # để biểu diễn các thẻ tiêu đề từ h1 tới h6.

Có thẻ sử dụng 1 thẻ đóng #

```
# Heading 1
# Heading1 #
Heading 1
=====

## Heading 2
## Heading ##
Heading 2
-----
## Heading 3
## Heading 3 ##
### Heading 4
### Heading 4 ###
#### Heading 5
#### Heading 5####
##### Heading 6
##### Heading 6 #####
```

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

Hình 2.3 Heading

Character styles

Tạo một đoạn văn bản

```
text
```

Tạo chữ in đậm

```
**Text**
```

Tạo chữ in nghiêng

```
_Text_
```

Tạo chữ in đậm và in nghiêng

```
** _text_ **
```

Tạo chữ trong hộp

```
`Text`
```

Tạo chữ gạch ngang

```
~~Text~~
```

Tạo văn bản trích dẫn

```
> text2
```

List

Để đánh dấu 1 danh sách không có thứ tự (unorder list) sử dụng dấu - hoặc + hoặc ***** trước mỗi dòng. Để đánh dấu một danh sách có thứ tự sử dụng các số thay vì các dấu như ở trên.

- Text1
 - Text2
-
1. Text1
 2. Text2

Links

Markdown hỗ trợ 2 kiểu chèn liên kết là: **inline** và **references**.

Để tạo một liên kết nội tuyến, sử dụng một tập hợp các dấu ngoặc đơn ngay sau ngoặc vuông. Bên trong ngoặc đặt URL muốn liên kết, cùng với một tiêu đề tùy chọn cho liên kết, bao bọc trong dấu ngoặc kép.

```
Markdown [Link] (http://example.com)
```

```
Markdown [Link] (.../link)
```

```
Link1[Link1] (1) Link2[Link2] (2)
```

```
[1]: http://example1.com
```

```
[2]: http://example2.com
```

Images

Hình ảnh trong Markdown tương tự như liên kết. Sự khác biệt là:

Các dấu ngoặc vuông phải được bắt đầu bằng một dấu chấm than

Và bên trong chúng có thể có một số văn bản thay thế. Mô tả về hình ảnh, nó sẽ được hiển thị nếu hình ảnh bị lỗi.

```
Picture: ! [Alt] (https://i.imgur.com/nhzMtLm.png)
```

Table

Number	Next number	Previous number
:-----	-----	-----
Five	Six	Four
Ten	Eleven	Nine
Seven	Eight	Six

Code và block

Có 2 loại code có thể viết trong markdown: inline code (code trong dòng) và code block (đoạn code riêng). sử dụng lần lượt 1 ký tự và 3 ký tự `.

2.3 Giới thiệu về ngôn ngữ YAML

2.3.1 Giới thiệu YAML

YAML, viết tắt của “YAML Ain't Markup Language”, là ngôn ngữ tuân tự hóa dữ liệu mà con người có thể đọc được. Ngôn ngữ này thường được sử dụng để tạo tệp cấu hình và hoạt động tốt với bất kỳ ngôn ngữ lập trình nào. YAML được thiết kế để dễ sử dụng và tương tác, khiến nó trở thành lựa chọn phổ biến trong số các nhà phát triển.

YAML là siêu tập nghiêm ngặt của JSON, một ngôn ngữ tuân tự hóa dữ liệu khác. Điều này có nghĩa là YAML có thể làm mọi thứ mà JSON có thể làm và hơn thế nữa. Không giống như JSON, YAML sử dụng thụt lề và xuống dòng để biểu thị cấu trúc, thay vì dựa vào dấu ngoặc và dấu ngoặc nhọn. Điều này làm cho các tệp YAML sạch hơn và dễ đọc hơn.

Cách viết YAML

Cấu trúc cơ bản của tệp YAML là một bản đồ. Có thể gọi đây là từ điển, băm hoặc đối tượng, tùy thuộc vào ngôn ngữ lập trình.

2.3.2 Collection types

Cách biểu diễn các cặp khóa-giá trị:

environment:

TEST_REPORTS: /tmp/test-reports

Hình 2.4 Biểu diễn cặp khóa giá trị

Danh sách các phần tử được biểu diễn như sau:

```
docker:
```

- image: ubuntu:20.04
- image: mongo:4.4
 - command: [mongod, --smallfiles]
- image: postgres:13

Hình 2.5 Biểu diễn danh sách phần tử

Cấu trúc bản lồng nhau

```
jobs:
```

```
build:
```

```
  docker:
```

- image: circleci/python:3.8

```
environment:
```

```
  NODE_ENV: development
```

```
steps:
```

- checkout

```
  - run:
```

```
    name: Install Dependencies
```

```
    command: pip install -r requirements.txt
```

Hình 2.6 Cấu trúc bảng đồ lồng nhau

2.3.3 Mẹo viết YAML

Thụt lề nhất quán: Luôn sử dụng khoảng trắng, không phải tab, để thụt lề. Điều này đảm bảo tính nhất quán và tránh lỗi cú pháp.

Cú pháp thích hợp: Đảm bảo dấu hai chấm (:) và dấu gạch ngang (-) được theo sau bởi một khoảng trắng để đảm bảo tính dễ đọc và chính xác.

Trích dẫn: Sử dụng dấu ngoặc kép cho các chuỗi chứa ký tự đặc biệt hoặc từ dành riêng để tránh hiểu sai.

Bình luận: Thêm bình luận bằng cách sử dụng ký tự # để giải thích cấu hình, giúp người khác (và chính bạn) dễ hiểu tệp YAML của bạn hơn

CHƯƠNG 3: HIỆN THỰC HÓA NGHIÊN CỨU

3.1 Tạo tài liệu với Jupyter Book

Chạy lệnh để khởi tạo dự án:

```
jupyter-book create Numpy
```

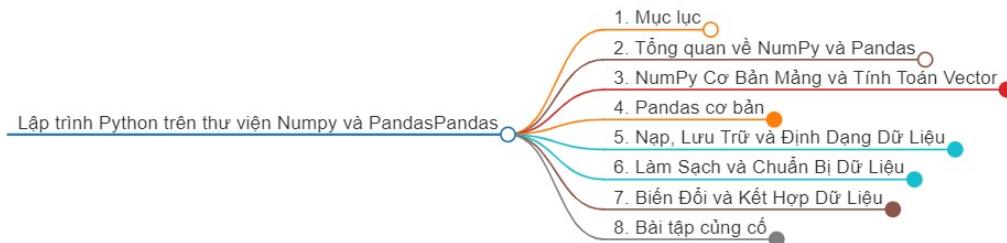
Di chuyển đến thư mục dự án:

```
cd D:/Do_an/Numpy
```

Chạy lệnh để tạo sách:

```
jupyter-book build .
```

3.1.1 Cấu trúc tài liệu



Hình 3.1 Mô tả cấu trúc tài liệu

3.1.2 Các file của tài liệu

mucluc.md: mục lục

Chương tổng quan:

- tongquan.md: tổng quan về Numpy và Pandas

Chương Numpy cơ bản:

- Numpy.md: Numpy cơ bản mảng và tính toán vector
- ndarray.md: đối tượng ndarray mảng nhiều chiều
- UniversalFunctions.md: hàm toàn cục và tối ưu hóa hiệu suất
- ArrayOriented.md: lập trình hướng mảng

- InputOutput.md: nhập/xuất dữ liệu
- Khac.md: các ứng dụng khác

Chương Pandas cơ bản:

- Pandas.md: Pandas cơ bản
- DataStructures.md: giới thiệu cấu trúc dữ liệu
- EssentialFunctionality.md: các chức năng cơ bản
- DescriptiveStatistics.md: thống kê và phân tích dữ liệu

Chương nạp, lưu trữ và định dạng dữ liệu:

- Format.md: nạp, lưu trữ và định dạng dữ liệu
- TextFormat.md: đọc và ghi dữ liệu văn bản
- DataFormat.md: định dạng dữ liệu nhị phân

Chương làm sạch và chuẩn bị dữ liệu:

- DataCleaning.md: làm sạch và chuẩn bị dữ liệu
- Handling.md: xử lý dữ liệu thiếu
- Transformation.md: biến đổi dữ liệu
- StringManipulation.md: xử lý chuỗi ký tự

Chương biến đổi và kết hợp dữ liệu

- DataWrangling.md: biến đổi và kết hợp dữ liệu
- Indexing.md: indexing phân cấp
- Combining.md: kết hợp dữ liệu
- Reshaping.md: biến đổi dữ liệu

Chương bài tập:

- BTNumPy.md: bài tập Numpy cơ bản
- BTPandas.md: bài tập Pandas cơ bản
- BTUongQuan.md: phân tích tương quan và hồi quy

3.2 Quá trình biên soạn

3.2.1 Tạo mục lục

Cập nhật file `_toc.yml` để tạo mục lục cho Jupyter Book

```
# _toc.yml  
  
format: jb-book
```

```
root: intro

chapters:
  - file: mucluc.md
    title: Mục lục
  - file: tongquan.md
    title: Tổng quan về NumPy và Pandas
  - file: Numpy.md
    title: NumPy Cơ Bản Mảng và Tính Toán Vector
  sections:
    - file: ndarray.md
      title: Đổi tượng ndarray Mảng Nhiều Chiều
    - file: UniversalFunctions.md
      title: Hàm Toàn Cục và Tối Ưu Hoá Hiệu Suất
    - file: ArrayOriented.md
      title: Lập Trình Hướng Mảng
    - file: InputOutput.md
      title: Nhập/Xuất Dữ Liệu
    - file: Khac.md
      title: Các Ứng Dụng Khác

  - file: Pandas.md
    title: Giới thiệu về Pandas
  sections:
    - file: DataStructures.md
      title: Giới Thiệu Cấu Trúc Dữ Liệu
    - file: EssentialFunctionality.md
      title: Các Chức Năng Cơ Bản
    - file: DescriptiveStatistics.md
      title: Thống Kê và Phân Tích Dữ Liệu

  - file: Format.md
    title: Nạp, Lưu Trữ và Định Dạng Dữ Liệu
  sections:
    - file: TextFormat.md
      title: Đọc và Ghi Dữ Liệu Văn Bản
    - file: DataFormat.md
      title: Định Dạng Dữ Liệu Nhị Phân

  - file: DataCleaning.md
    title: Làm Sạch và Chuẩn Bị Dữ Liệu
  sections:
```

```
- file: Handling.md
    title: Xử Lý Dữ Liệu Thiếu
- file: Transformation.md
    title: Biến Đổi Dữ Liệu
- file: StringManipulation.md
    title: Xử Lý Chuỗi Ký Tự

- file: DataWrangling.md
    title: Biến Đổi và Kết Hợp Dữ Liệu
    sections:
        - file: Indexing.md
            title: Indexing Phân Cấp
        - file: Combining.md
            title: Kết Hợp Dữ Liệu
        - file: Reshaping.md
            title: Biến Đổi Dữ Liệu
- file: bt.md
    title: Bài tập cùng cỗ
    sections:
        - file: BTNumpy.md
            title: Bài tập Numpy cơ bản
        - file: BTPandas.md
            title: Bài tập Pandas cơ bản
        - file: BTTuongQuan.md
            title: Phân tích tương quan và hồi quy
```

3.2.2 Viết nội dung cho từng chương

Chương 1: Tổng quan

Tệp tongquan.md cung cấp cái nhìn khái quát về thư viện Numpy và Pandas

Chương 2: Numpy cơ bản

Numpy.md: Giới thiệu về Numpy và các khái niệm cơ bản liên quan đến mảng và tính toán vector.

ndarray.md: Tìm hiểu về đối tượng ndarray - cấu trúc mảng nhiều chiều đặc trưng của Numpy.

UniversalFunctions.md: Khám phá các hàm toàn cục (ufuncs) và cách chúng tối ưu hóa hiệu suất tính toán.

ArrayOriented.md: Giới thiệu lập trình hướng mảng, một kỹ thuật quan trọng trong xử lý dữ liệu bằng Numpy.

InputOutput.md: Hướng dẫn cách nhập và xuất dữ liệu với Numpy.

Khac.md: Các ứng dụng và trường hợp sử dụng khác của Numpy.

Chương 3: Pandas cơ bản

Pandas.md: Tổng quan về Pandas và các tính năng cơ bản.

DataStructures.md: Giới thiệu các cấu trúc dữ liệu chính của Pandas, như Series và DataFrame.

EssentialFunctionality.md: Các chức năng cơ bản của Pandas giúp thao tác và phân tích dữ liệu.

DescriptiveStatistics.md: Các phương pháp thống kê và phân tích dữ liệu với Pandas.

Chương 4: Nạp, lưu trữ và định dạng dữ liệu

Format.md: Tổng quan về các phương pháp nạp, lưu trữ và định dạng dữ liệu.

TextFormat.md: Hướng dẫn đọc và ghi dữ liệu văn bản.

DataFormat.md: Định dạng dữ liệu nhị phân và các phương pháp xử lý liên quan.

Chương 5: Làm sạch và chuẩn bị dữ liệu

DataCleaning.md: Các kỹ thuật làm sạch và chuẩn bị dữ liệu cho phân tích.

Handling.md: Xử lý dữ liệu thiếu và các phương pháp khắc phục.

Transformation.md: Biến đổi dữ liệu để phù hợp với các yêu cầu phân tích.

StringManipulation.md: Các kỹ thuật xử lý chuỗi ký tự trong dữ liệu.

Chương 6: Biến đổi và kết hợp dữ liệu

DataWrangling.md: Các kỹ thuật biến đổi và kết hợp dữ liệu để phục vụ cho phân tích.

Indexing.md: Khám phá indexing phân cấp và các ứng dụng của nó.

Combining.md: Kết hợp dữ liệu từ nhiều nguồn khác nhau.

Reshaping.md: Các phương pháp biến đổi dữ liệu để phù hợp với các yêu cầu phân tích.

Chương 7: Bài tập

BTNumpy.md: Các bài tập về Numpy cơ bản để kiểm tra và củng cố kiến thức.

BTPandas.md: Các bài tập về Pandas cơ bản.

BTTuongQuan.md: Phân tích tương quan và hồi quy thông qua các bài tập thực hành.

CHƯƠNG 4: KẾT QUẢ NGHIÊN CỨU

4.1 Kết quả tài liệu trên Jupyter Book

4.1.1 Phản mục lục

Kết quả: Trang mục lục với đầy đủ các file chương chính và file nội dung có thể hiển thị mở rộng bằng các dấu mũi tên.

Chức năng: điều hướng tới các trang tài liệu



Hình 4.1 Mục lục Jupyter Book

Tương tự như phần mục lục của Jupyter Book trang mục lục có thể điều hướng đến các file chương và đây là mục lục xuất hiện khi xuất tài liệu:

The screenshot shows a table of contents for a Jupyter Book. At the top right are icons for search, download, and other navigation. Below the title 'Mục Lục' is a section titled 'Tổng quan về NumPy và Pandas' with a single item: 'Tổng quan về NumPy và Pandas'. This leads to a page with the same title and a list of sub-sections under 'NumPy Cơ Bản Mảng và Tính Toán Vector'. The sub-sections include: 'NumPy Cơ Bản Mảng và Tính Toán Vector' (with links to 'Đổi tượng ndarray Mảng Nhiều Chiều', 'Hàm Tính Cực và Tối Ưu Hóa Hiệu Suất', 'Lập Trình Hướng Mảng', 'Nhập/Xuất Dữ Liệu', and 'Các Ứng Dụng Khác'); 'Pandas cơ bản' (with links to 'Pandas cơ bản' and 'Giới Thiệu Cấu Trúc Dữ Liệu'); and a sidebar with 'Contents' and sections for 'Tổng quan về NumPy và Pandas', 'NumPy Cơ Bản Mảng và Tính Toán Vector', 'Pandas cơ bản', and 'Bài tập cung cấp'.

- [Tổng quan về NumPy và Pandas](#)

NumPy Cơ Bản Mảng và Tính Toán Vector

- [NumPy Cơ Bản Mảng và Tính Toán Vector](#)
 - [Đổi tượng ndarray Mảng Nhiều Chiều](#)
 - [Hàm Tính Cực và Tối Ưu Hóa Hiệu Suất](#)
 - [Lập Trình Hướng Mảng](#)
 - [Nhập/Xuất Dữ Liệu](#)
 - [Các Ứng Dụng Khác](#)

Pandas cơ bản

- [Pandas cơ bản](#)
 - [Giới Thiệu Cấu Trúc Dữ Liệu](#)

Hình 4.2 Trang mục lục của tài liệu

4.1.2 Các trang tài liệu

Đã biên soạn một bộ tài liệu chi tiết với cấu trúc rõ ràng bao gồm 7 chương:

1. Tổng quan về Python, NumPy và Pandas.
2. Hướng dẫn chi tiết về NumPy.
3. Hướng dẫn chi tiết về Pandas.
4. Xử lý và chuẩn bị dữ liệu thực tế.
5. Biến đổi và kết hợp dữ liệu.
6. Ứng dụng thực tế và ví dụ minh họa.
7. Bài tập thực hành nâng cao.

Tài liệu tương đối đầy đủ nội dung bao gồm phân lý thuyết, code ví dụ minh họa, các bảng và hình ảnh giúp người đọc dễ dàng áp dụng vào thực tiễn.

NumPy và Pandas là hai thư viện mạnh mẽ trong Python, rất quan trọng cho việc phân tích dữ liệu, xử lý số liệu và khoa học dữ liệu. Cá hai đều được sử dụng rộng rãi trong cộng đồng khoa học dữ liệu và cung cấp các công cụ và hàm mạnh mẽ để làm việc với dữ liệu một cách hiệu quả và linh hoạt.

NumPy

NumPy (Numerical Python) là thư viện cơ bản cho tính toán khoa học trong Python. Một số tính năng chính của NumPy bao gồm:

- **Mảng Nhiều Chiều (ndarray):** NumPy cung cấp một đối tượng mảng nhiều chiều rất hiệu quả, giúp lưu trữ và thao tác với dữ liệu số. Các mảng này có thể có nhiều chiều và kích thước khác nhau, và chúng hỗ trợ rất nhiều phép toán số học và thống kê.
- **Hàm Toàn Cực (Universal Functions):** NumPy cung cấp một bộ sưu tập phong phú các hàm toàn cục, cho phép thực hiện các phép toán trên toàn bộ mảng mà không cần phải viết vòng lặp.
- **Broadcasting:** Tính năng này cho phép các hoạt động trên mảng có hình dạng khác nhau mà không cần sao chép dữ liệu.
- Tích Hợp với Các Thư Viện Khác: NumPy tích hợp tốt với các thư viện khác như SciPy, Matplotlib, và Pandas, tạo thành một hệ sinh thái mạnh mẽ cho phân tích dữ liệu và tính toán khoa học.

Hình 4.3 Minh họa tài liệu

```
import numpy as np
arr = np.arange(10)
print(np.sqrt(arr)) # Căn bậc hai của các phần tử
print(np.exp(arr)) # Lũy thừa e của các phần tử
```

Kết quả:

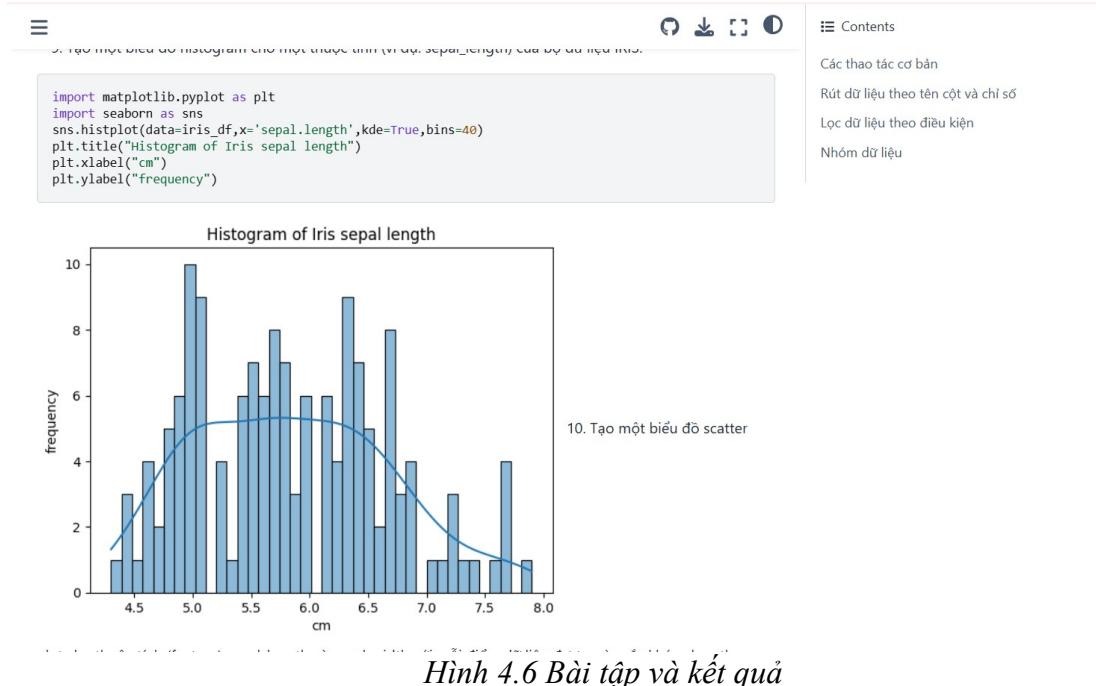
```
array([ 0. ,  1. ,  1.4142,  1.7321,  2. ,  2.2361,  2.4495,  2.6458,  2.8284,  3. ])
array([ 1. ,  2.7183,  7.3891, 20.0855, 54.5982, 148.4132, 403.4288, 1096.6332, 2980.958 , 8103.0839])
```

Các ufunc đơn như sqrt và exp thực hiện phép toán theo từng phần tử trong mảng.

Hình 4.4 Minh họa các ví dụ của tài liệu

Phương thức	Mô tả
<code>count</code>	Trả về số lần xuất hiện không trùng lặp của chuỗi con trong chuỗi.
<code>endswith</code>	Trả về <code>True</code> nếu chuỗi kết thúc bằng hậu tố.
<code>startswith</code>	Trả về <code>True</code> nếu chuỗi bắt đầu bằng tiền tố.
<code>join</code>	Sử dụng chuỗi làm dấu phân cách để nối một dãy các chuỗi khác.
<code>index</code>	Trả về vị trí của ký tự đầu tiên trong chuỗi con nếu tìm thấy trong chuỗi; đưa ra <code>ValueError</code> nếu không tìm thấy.
<code>find</code>	Trả về vị trí của ký tự đầu tiên của lần xuất hiện đầu tiên của chuỗi con trong chuỗi; giống như <code>index</code> , nhưng trả về <code>-1</code> nếu không tìm thấy.
<code>rfind</code>	Trả về vị trí của ký tự đầu tiên của lần xuất hiện cuối cùng của chuỗi con trong chuỗi; trả về <code>-1</code> nếu không tìm thấy.
<code>replace</code>	Thay thế các lần xuất hiện của chuỗi này bằng chuỗi khác.
<code>strip</code> , <code>rstrip</code> , <code>lstrip</code>	Loại bỏ khoảng trắng, bao gồm cả dòng mới; tương đương với <code>x.strip()</code> (và <code>rstrip</code> , <code>lstrip</code> , tương ứng).

Hình 4.5 Các bảng của tài liệu



Hình 4.6 Bài tập và kết quả

4.2 Kết quả tài liệu xuất bằng PDF

Mục Lục

Print to PDF ▶

Contents

- Tổng quan về NumPy và Pandas
- NumPy Cơ Bản Mảng và Tính Toán Vector
- Pandas cơ bản
- Nạp, Lưu Trữ và Định Dạng Dữ Liệu
- Làm Sạch và Chuẩn Bị Dữ Liệu
- Biến Đổi và Kết Hợp Dữ Liệu
- Bài tập củng cố

Tổng quan về NumPy và Pandas

- [Tổng quan về NumPy và Pandas](#)

NumPy Cơ Bản Mảng và Tính Toán Vector

- [NumPy Cơ Bản Mảng và Tính Toán Vector](#)
 - [Đối tượng ndarray Mảng Nhiều Chiều](#)
 - [Hàm Toàn Cục và Tối Ưu Hoá Hiệu Suất](#)
 - [Lập Trình Hướng Mảng](#)
 - [Nhập/Xuất Dữ Liệu](#)
 - [Các Ứng Dụng Khác](#)

Pandas cơ bản

- [Pandas cơ bản](#)
 - [Giới Thiệu Cấu Trúc Dữ Liệu](#)
 - [Các Chức Năng Cơ Bản](#)
 - [Thống Kê và Phân Tích Dữ Liệu](#)

Nạp, Lưu Trữ và Định Dạng Dữ Liệu

- [Nạp, Lưu Trữ và Định Dạng Dữ Liệu](#)
 - [Đọc và Ghi Dữ Liệu Văn Bản](#)
 - [Định Dạng Dữ Liệu Nhị Phân](#)

Làm Sạch và Chuẩn Bị Dữ Liệu

- [Làm Sạch và Chuẩn Bị Dữ Liệu](#)
 - [Xử Lý Dữ Liệu Thiếu](#)
 - [Biến Đổi Dữ Liệu](#)
 - [Xử Lý Chuỗi Ký Tự](#)

Biến Đổi và Kết Hợp Dữ Liệu

- [Biến Đổi và Kết Hợp Dữ Liệu](#)
 - [Indexing Phân Cấp](#)
 - [Kết Hợp Dữ Liệu](#)
 - [Biến Đổi Dữ Liệu](#)

Bài tập củng cố

1. [Bài tập Numpy cơ bản](#)
2. [Bài tập Pandas cơ bản](#)
3. [Phân tích tương quan và hồi quy](#)

Đổi Tượng Mảng N Chiều NumPy: Một Đổi Tượng Mảng Đa Chiều

Print to PDF

Contents

- Ví Dụ Cơ Bản Về Mảng NumPy
- Tạo ndarrays
- Bảng 2-1. Các Hàm Tạo Mảng
- Các Kiểu Dữ Liệu cho ndarrays
- Bảng 2-2. Các Kiểu Dữ Liệu của NumPy
- Chuyển Đổi Kiểu Dữ Liệu
- Các Phép Toán với Mảng NumPy
- Indexing và Slicing Cơ Bản
- ndexing với Slices
- Boolean Indexing
- Fancy Indexing
- Transposing Arrays và Swapping Axes

Một trong những tính năng chính của **NumPy** là đổi tượng mảng N chiều, hay **ndarray**, là một container nhanh chóng và linh hoạt cho các tập dữ liệu lớn trong Python. Các mảng cho phép bạn thực hiện các phép toán toán học trên toàn bộ khối dữ liệu bằng cách sử dụng cú pháp tương tự như các phép toán tương đương giữa các phần tử vô hướng.

Ví Dụ Cơ Bản Về Mảng NumPy

Để cung cấp cho bạn một cái nhìn tổng quan về cách **NumPy** cho phép thực hiện các phép toán theo lô với cú pháp tương tự như các giá trị vô hướng trên các đối tượng tích hợp sẵn của Python, tôi sẽ nhập **NumPy** và tạo ra một mảng nhỏ dữ liệu ngẫu nhiên:

```
import numpy as np

# Tạo một số dữ liệu ngẫu nhiên
data = np.random.randn(2, 3)
print(data)
```

```
array([[-0.2047,  0.4789, -0.5194],
       [-0.5557,  1.9658,  1.3934]])
```

```
# Nhân tất cả các phần tử với 10
print(data * 10)

# Cộng các phần tử tương ứng trong mỗi hàng
print(data + data)
```

```
array([[ -2.0471,   4.7894,  -5.1944],
       [ -5.5573,  19.6578,  13.9341]])

array([[ -0.4094,   0.9579,  -1.0389],
       [ -1.1115,   3.9316,   2.7868]])
```

Trong ví dụ đầu tiên, tất cả các phần tử trong mảng đã được nhân với 10. Trong ví dụ thứ hai, các giá trị tương ứng trong mỗi "ô" trong mảng đã được cộng lại với nhau.

Một ndarray là một container đa chiều tổng quát cho dữ liệu đồng nhất, tức là tất cả các phần tử phải cùng loại. Mỗi mảng có:

shape: Một tuple chỉ ra kích thước của mỗi chiều.

dtype: Một đối tượng mô tả kiểu dữ liệu của mảng.

```
print(data.shape) # (2, 3)
print(data.dtype) # dtype('float64')
```

Tạo ndarrays

Cách dễ nhất để tạo một mảng là sử dụng hàm `array`. Hàm này chấp nhận bất kỳ đối tượng giống như chuỗi nào (bao gồm cả các mảng khác) và tạo ra một mảng **NumPy** mới chứa dữ liệu đã được truyền vào.

```
data1 = [6, 7.5, 8, 0, 1]
arr1 = np.array(data1)
print(arr1)
```

```
array([ 6., 7.5, 8., 0., 1.])
```

Các chuỗi lồng nhau, như một danh sách các danh sách có chiều dài bằng nhau, sẽ được chuyển đổi thành một mảng đa chiều:

```
data2 = [[1, 2, 3, 4], [5, 6, 7, 8]]
arr2 = np.array(data2)
print(arr2)
```

Output:

```
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

Vì data2 là một danh sách các danh sách, mảng NumPy arr2 có hai chiều với hình dạng được suy diễn từ dữ liệu. Chúng ta có thể xác nhận điều này bằng cách kiểm tra các thuộc tính `ndim` và `shape`:

```
print(arr2.ndim) # Số chiều của mảng
print(arr2.shape) # Hình dạng của mảng
```

Output:

```
2
(2, 4)
```

Trừ khi được chỉ định rõ ràng (sẽ được đề cập sau), np.array cố gắng suy diễn một kiểu dữ liệu tốt cho mảng mà nó tạo ra. Kiểu dữ liệu được lưu trữ trong một đối tượng metadata dtype đặc biệt; ví dụ, trong hai ví dụ trên, chúng ta có:

```
print(arr1.dtype) # dtype('float64')
print(arr2.dtype) # dtype('int64')
```

Ngoài np.array, còn có một số hàm khác để tạo ra các mảng mới. Ví dụ, zeros và ones tạo ra các mảng chứa toàn số 0 hoặc 1, tương ứng, với một chiều dài hoặc hình dạng nhất định. empty tạo ra một mảng mà không khởi tạo các giá trị của nó thành bất kỳ giá trị cụ thể nào. Để tạo ra một mảng có chiều cao hơn với các phương pháp này, hãy truyền một tuple cho hình dạng:

```
np.zeros(10)
# Output: array([ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.])

np.zeros((3, 6))
# Output: array([[ 0.,  0.,  0.,  0.,  0.,  0.],
#                [ 0.,  0.,  0.,  0.,  0.,  0.],
#                [ 0.,  0.,  0.,  0.,  0.,  0.]))

np.empty((2, 3, 2))
# Output: array([[[ 0.,  0.],
#                  [ 0.,  0.],
#                  [ 0.,  0.]],
#                 [[ 0.,  0.],
#                  [ 0.,  0.],
#                  [ 0.,  0.]]])
```

Lưu ý: Không an toàn khi giả định rằng np.empty sẽ trả về một mảng toàn số 0. Trong một số trường hợp, nó có thể trả về các giá trị "rác" chưa được khởi tạo.

arange là một phiên bản mảng của hàm range tích hợp sẵn trong Python:

```
np.arange(15)
# Output: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9,  10,  11,  12,  13,  14])
```

Bảng 2-1. Các Hàm Tạo Mảng

Hàm	Mô tả
<code>array</code>	Chuyển đổi dữ liệu đầu vào (danh sách, tuple, mảng hoặc loại chuỗi khác) thành một <code>ndarray</code> bằng cách suy diễn một <code>dtype</code> hoặc chỉ định rõ ràng một <code>dtype</code> ; sao chép dữ liệu đầu vào theo mặc định.
<code>asarray</code>	Chuyển đổi đầu vào thành <code>ndarray</code> , nhưng không sao chép nếu đầu vào đã là một <code>ndarray</code> .
<code>arange</code>	Giống như hàm <code>range</code> tích hợp nhưng trả về một <code>ndarray</code> thay vì một danh sách.
<code>ones</code> , <code>ones_like</code>	Tạo ra một mảng chứa toàn số 1 với hình dạng và <code>dtype</code> đã cho; <code>ones_like</code> nhận một mảng khác và tạo ra một mảng chứa toàn số 1 có cùng hình dạng và <code>dtype</code> .
<code>zeros</code> , <code>zeros_like</code>	Giống như <code>ones</code> và <code>ones_like</code> nhưng tạo ra các mảng chứa toàn số 0.
<code>empty</code> , <code>empty_like</code>	Tạo ra các mảng mới bằng cách cấp phát bộ nhớ mới, nhưng không điền vào bất kỳ giá trị nào như <code>ones</code> và <code>zeros</code> .
<code>full</code> , <code>full_like</code>	Tạo ra một mảng có hình dạng và <code>dtype</code> đã cho với tất cả các giá trị được đặt thành “giá trị điền” đã chỉ định; <code>full_like</code> nhận một mảng khác và tạo ra một mảng đã điền có cùng hình dạng và <code>dtype</code> .
<code>eye</code> , <code>identity</code>	Tạo ra một ma trận đơn vị vuông $N \times N$ (1s trên đường chéo và 0s ở những nơi khác).

dịch theo phong cách học thuật để tôi có thể viết vào tài liệu Chắc chắn rồi! Dưới đây là bản dịch theo phong cách học thuật:

Các Kiểu Dữ Liệu cho ndarrays

Kiểu dữ liệu (hay `dtype`) là một đối tượng đặc biệt trong NumPy chứa thông tin cần thiết để diễn giải một khối bộ nhớ dưới dạng một loại dữ liệu cụ thể. Ví dụ:

```
In [33]: arr1 = np.array([1, 2, 3], dtype=np.float64)
In [34]: arr2 = np.array([1, 2, 3], dtype=np.int32)
In [35]: arr1.dtype
Out[35]: dtype('float64')
In [36]: arr2.dtype
Out[36]: dtype('int32')
```

dtypes là yếu tố quan trọng cho sự linh hoạt của NumPy trong việc tương tác với dữ liệu từ các hệ thống khác nhau. Chúng cung cấp một ánh xạ trực tiếp đến đại diện đĩa hoặc bộ nhớ cơ bản, giúp dễ dàng đọc và ghi các luồng dữ liệu nhị phân vào đĩa và kết nối với mã viết bằng các ngôn ngữ cấp thấp như C hoặc Fortran.

Các kiểu dữ liệu số được đặt tên theo cách bao gồm tên kiểu như float hoặc int, tiếp theo là số bit mỗi phần tử chiếm dụng. Ví dụ, giá trị số thực với độ chính xác kép chuẩn (được sử dụng trong đối tượng float của Python) chiếm 8 byte hoặc 64 bit, được biết đến trong NumPy là float64.

Bảng 2-2. Các Kiểu Dữ Liệu của NumPy

Kiểu	Mã Kiểu	Mô Tả
<code>int8</code> , <code>uint8</code>	<code>i1</code> , <code>u1</code>	Số nguyên 8-bit có dấu và không dấu (1 byte)
<code>int16</code> , <code>uint16</code>	<code>i2</code> , <code>u2</code>	Số nguyên 16-bit có dấu và không dấu
<code>int32</code> , <code>uint32</code>	<code>i4</code> , <code>u4</code>	Số nguyên 32-bit có dấu và không dấu
<code>int64</code> , <code>uint64</code>	<code>i8</code> , <code>u8</code>	Số nguyên 64-bit có dấu và không dấu
<code>float16</code>	<code>f2</code>	Số thực độ chính xác nửa
<code>float32</code>	<code>f4</code> , <code>f</code>	Số thực độ chính xác đơn (tương thích với <code>float</code> trong C)
<code>float64</code>	<code>f8</code> , <code>d</code>	Số thực độ chính xác gấp đôi (tương thích với <code>double</code> trong C) và <code>float</code> trong Python
<code>float128</code>	<code>f16</code> , <code>g</code>	Số thực độ chính xác mở rộng
<code>complex64</code> , <code>complex128</code> , <code>complex256</code>	<code>c8</code> , <code>c16</code> , <code>c32</code>	Số phức với hai số thực 32, 64, hoặc 128 bit
<code>bool</code>	<code>?</code>	Kiểu boolean, lưu trữ giá trị <code>True</code> và <code>False</code>
<code>object</code>	<code>o</code>	Kiểu đối tượng Python, có thể lưu bất kỳ đối tượng Python nào
<code>string_</code>	<code>s</code>	Kiểu chuỗi ASCII cố định (1 byte mỗi ký tự); ví dụ, để tạo một kiểu chuỗi với độ dài 10, sử dụng <code>'S10'</code>
<code>unicode_</code>	<code>U</code>	Kiểu Unicode cố định, số byte phụ thuộc vào nền tảng; cách xác định giống như <code>string_</code> (ví dụ, <code>'U10'</code>)

Chuyển Đổi Kiểu Dữ Liệu

Có thể chuyển đổi hoặc ép kiểu một mảng từ kiểu này sang kiểu khác bằng cách sử dụng phương thức `astype` của `ndarray`:

```
In [37]: arr = np.array([1, 2, 3, 4, 5])
In [38]: arr.dtype
Out[38]: dtype('int64')
In [39]: float_arr = arr.astype(np.float64)
In [40]: float_arr.dtype
Out[40]: dtype('float64')
```

Trong ví dụ trên, các số nguyên được chuyển đổi thành số thực. Nếu chuyển đổi số thực

```
```python
In [41]: arr = np.array([3.7, -1.2, -2.6, 0.5, 12.9, 10.1])
In [42]: arr
Out[42]: array([3.7, -1.2, -2.6, 0.5, 12.9, 10.1])
In [43]: arr.astype(np.int32)
Out[43]: array([3, -1, -2, 0, 12, 10], dtype=int32)
```

Nếu có một mảng chuỗi đại diện cho các số, có thể dùng `astype` để chuyển đổi chúng thành dạng số:

```
In [44]: numeric_strings = np.array(['1.25', '-9.6', '42'], dtype=np.string_)
In [45]: numeric_strings.astype(float)
Out[45]: array([1.25, -9.6 , 42.])
```

Quan trọng cần lưu ý khi sử dụng kiểu `numpy.string_`, vì dữ liệu chuỗi trong NumPy có kích thước cố định và có thể bị cắt mà không có cảnh báo. pandas có hành vi trực quan hơn khi xử lý dữ liệu không phải số.

## Các Phép Toán với Mảng NumPy

Mảng (array) rất quan trọng vì chúng cho phép bạn thể hiện các thao tác hàng loạt trên dữ liệu mà không cần viết bất kỳ vòng lặp nào. Người dùng NumPy gọi điều này là vectorization. Bất kỳ phép toán số học nào giữa các mảng có cùng kích thước đều áp dụng phép toán từng phần tử:

```
arr = np.array([[1., 2., 3.], [4., 5., 6.]])
arr
```

Kết quả:

```
array([[1., 2., 3.],
 [4., 5., 6.]])
```

Phép nhân phần tử từng phần tử giữa các mảng:

```
arr * arr
```

Kết quả:

```
array([[1., 4., 9.],
 [16., 25., 36.]])
```

Phép trừ phần tử từng phần tử giữa các mảng:

```
arr - arr
```

Kết quả:

```
array([[0., 0., 0.],
 [0., 0., 0.]])
```

Các phép toán số học với các số vô hướng (scalars) sẽ lan truyền đối số vô hướng đến từng phần tử trong mảng:

```
1 / arr
```

Kết quả:

```
array([[1. , 0.5 , 0.3333],
 [0.25, 0.2 , 0.1667]])
```

Phép khai căn từng phần tử trong mảng:

```
arr ** 0.5
```

Kết quả:

```
array([[1. , 1.4142, 1.7321],
 [2. , 2.2361, 2.4495]])
```

So sánh giữa các mảng có cùng kích thước sẽ tạo ra các mảng boolean:

```
arr2 = np.array([[0., 4., 1.], [7., 2., 12.]])
arr2
```

Kết quả:

```
array([[0., 4., 1.],
 [7., 2., 12.]])
```

```
arr2 > arr
```

Kết quả:

```
array([[False, True, False],
 [True, False, True]], dtype=bool)
```

Các phép toán giữa các mảng có kích thước khác nhau được gọi là broadcasting

## Indexing và Slicing Cơ Bản

Indexing mảng NumPy là một chủ đề phong phú, vì có nhiều cách chọn một tập con dữ liệu hoặc các phần tử riêng lẻ. Các mảng một chiều rất đơn giản; trên bề mặt, chúng hoạt động tương tự như các danh sách Python:

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

Kết quả:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Truy cập phần tử tại chỉ mục 5:

```
arr[5]
```

Kết quả:

```
5
```

Truy cập slice từ chỉ mục 5 đến 8:

```
arr[5:8]
```

Kết quả:

```
array([5, 6, 7])
```

Gán giá trị vô hướng cho một slice:

```
arr[5:8] = 12
arr
```

Kết quả:

```
array([0, 1, 2, 3, 4, 12, 12, 12, 8, 9])
```

Như có thể thấy, nếu gán một giá trị vô hướng cho một slice, giá trị đó sẽ được truyền (hoặc phát tán) cho toàn bộ lựa chọn. Một điểm khác biệt quan trọng so với các danh sách tích hợp của Python là các slice của mảng là các view trên mảng gốc. Điều này có nghĩa là dữ liệu không được sao chép, và bất kỳ sửa đổi nào đối với view sẽ được phản ánh trong mảng nguồn.

Để minh họa điều này, đầu tiên tạo một slice của arr:

```
arr_slice = arr[5:8]
arr_slice
```

Kết quả:

```
array([12, 12, 12])
```

Bây giờ, khi thay đổi các giá trị trong arr\_slice, các thay đổi này sẽ được phản ánh trong mảng gốc arr:

```
arr_slice[1] = 12345
arr
```

Kết quả:

```
array([0, 1, 2, 3, 4, 12, 12345, 12, 8, 9])
```

Slice không có chỉ mục [:] sẽ gán cho tất cả các giá trị trong một mảng:

```
arr_slice[:] = 64
arr
```

Kết quả:

```
array([0, 1, 2, 3, 4, 64, 64, 64, 8, 9])
```

Nếu mới sử dụng NumPy, có thể ngạc nhiên bởi điều này, đặc biệt nếu đã sử dụng các ngôn ngữ lập trình mảng khác sao chép dữ liệu một cách chủ động hơn. Vì NumPy được thiết kế để làm việc với các mảng rất lớn, có thể tưởng tượng các vấn đề về hiệu suất và bộ nhớ nếu NumPy cứ luôn sao chép dữ liệu.

Nếu muốn một bản sao của một slice của một ndarray thay vì một view, cần sao chép mảng một cách rõ ràng, chẳng hạn như arr[5:8].copy().

Với các mảng nhiều chiều, có nhiều tùy chọn hơn. Trong một mảng hai chiều, các phần tử tại mỗi chỉ mục không còn là các số vô hướng mà thay vào đó là các mảng một chiều:

```
arr2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
arr2d[2]
```

Kết quả:

```
array([7, 8, 9])
```

Do đó, các phần tử riêng lẻ có thể được truy cập đệ quy. Nhưng điều đó hơi phức tạp, vì vậy có thể truyền một danh sách chỉ mục phân tách bằng dấu phẩy để chọn các phần tử riêng lẻ. Vì vậy, những cách sau là tương đương:

```
arr2d[0][2]
```

Kết quả:

```
3
```

```
arr2d[0, 2]
```

Kết quả:

```
3
```

Trong các mảng nhiều chiều, nếu bỏ qua các chỉ mục sau, đối tượng trả về sẽ là một ndarray có chiều thấp hơn bao gồm tất cả dữ liệu theo các chiều cao hơn. Vì vậy, trong mảng  $2 \times 2 \times 3$  arr3d:

```
arr3d = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
arr3d
```

Kết quả:

```
array([[[1, 2, 3],
 [4, 5, 6]],
 [[7, 8, 9],
 [10, 11, 12]]])
```

arr3d[0] là một mảng  $2 \times 3$ :

```
arr3d[0]
```

Kết quả:

```
array([[1, 2, 3],
 [4, 5, 6]])
```

Cả các giá trị vô hướng và mảng đều có thể được gán cho arr3d[0]:

```
old_values = arr3d[0].copy()
arr3d[0] = 42
arr3d
```

Kết quả:

```
array([[[42, 42, 42],
 [42, 42, 42]],
 [[7, 8, 9],
 [10, 11, 12]]])
```

Khôi phục lại các giá trị cũ:

```
arr3d[0] = old_values
arr3d
```

Kết quả:

```
array([[[1, 2, 3],
 [4, 5, 6]],
 [[7, 8, 9],
 [10, 11, 12]]])
```

Tương tự, arr3d[1, 0] cho tất cả các giá trị có chỉ số bắt đầu bằng (1, 0), hình thành một mảng một chiều:

```
arr3d[1, 0]
```

Kết quả:

```
array([7, 8, 9])
```

Biểu thức này giống như thể đã truy cập chỉ mục trong hai bước:

```
x = arr3d[1]
x
```

Kết quả:

```
array([[7, 8, 9],
 [10, 11, 12]])
```

```
x[0]
```

Kết quả:

```
array([7, 8, 9])
```

Lưu ý rằng trong tất cả các trường hợp này, nơi các phần của mảng đã được chọn, các mảng trả về là các view.

## ndexing với Slices

Giống như các đối tượng một chiều như danh sách Python, ndarray có thể được cắt (slice) bằng cú pháp quen thuộc:

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

Kết quả:

```
array([0, 1, 2, 3, 4, 64, 64, 64, 8, 9])
```

Truy cập một slice từ chỉ mục 1 đến 6:

```
arr[1:6]
```

Kết quả:

```
array([1, 2, 3, 4, 64])
```

Xem xét mảng hai chiều từ trước đó, arr2d. Cắt mảng này hơi khác một chút:

```
arr2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
arr2d
```

Kết quả:

```
array([[1, 2, 3],
 [4, 5, 6],
 [7, 8, 9]])
```

Cắt để lấy hai hàng đầu tiên:

```
arr2d[:2]
```

Kết quả:

```
array([[1, 2, 3],
 [4, 5, 6]])
```

Như thấy, nó đã cắt dọc theo trục 0, trục đầu tiên. Một slice, do đó, chọn một phạm vi các phần tử dọc theo một trục. Có thể đọc biểu thức arr2d[:2] là “chọn hai hàng đầu tiên của arr2d.”

Có thể truyền nhiều slices giống như truyền nhiều chỉ mục:

```
arr2d[:2, 1:]
```

Kết quả:

```
array([[2, 3],
 [5, 6]])
```

Khi cắt như thế này, luôn nhận được các view mảng có cùng số chiều. Bằng cách kết hợp các chỉ mục số nguyên và slices, có thể nhận được các slices có số chiều thấp hơn.

Ví dụ, có thể chọn hàng thứ hai nhưng chỉ các cột đầu tiên như sau:

```
arr2d[1, :2]
```

Kết quả:

```
array([4, 5])
```

Tương tự, có thể chọn cột thứ ba nhưng chỉ hai hàng đầu tiên như sau:

```
arr2d[:2, 2]
```

Kết quả:

```
array([3, 6])
```

Lưu ý rằng dấu hai chấm tự nó có nghĩa là lấy toàn bộ trực, vì vậy có thể chỉ cắt các trực chiều cao hơn bằng cách làm như sau:

```
arr2d[:, :1]
```

Kết quả:

```
array([[1],
 [4],
 [7]])
```

Tất nhiên, việc gán cho một biểu thức slice sẽ gán cho toàn bộ lựa chọn:

```
arr2d[::2, 1:] = 0
arr2d
```

Kết quả:

```
array([[1, 0, 0],
 [4, 0, 0],
 [7, 8,]])
```

## Boolean Indexing

Hãy xem xét một ví dụ trong đó có một số dữ liệu trong mảng và một mảng các tên có tên trùng lặp. Sử dụng hàm randn trong numpy.random để tạo ra một số dữ liệu phân phối chuẩn ngẫu nhiên:

```
names = np.array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe'])
data = np.random.randn(7, 4)
names
```

Kết quả:

```
array(['Bob', 'Joe', 'Will', 'Bob', 'Will', 'Joe', 'Joe'], dtype='<U4')
```

```
data
```

Kết quả:

```
array([[0.0929, 0.2817, 0.769 , 1.2464],
 [1.0072, -1.2962, 0.275 , 0.2289],
 [1.3529, 0.8864, -2.0016, -0.3718],
 [1.669 , -0.4386, -0.5397, 0.477],
 [3.2489, -1.0212, -0.5771, 0.1241],
 [0.3026, 0.5238, 0.0009, 1.3438],
 [-0.7135, -0.8312, -2.3702, -1.8608]])
```

Giả sử mỗi tên tương ứng với một hàng trong mảng dữ liệu và muốn chọn tất cả các hàng có tên tương ứng là 'Bob'. Giống như các phép toán số học, so sánh (chẳng hạn như ==) với các

mảng cũng được vector hóa. So sánh names với chuỗi 'Bob' cho ra một mảng boolean:

```
names == 'Bob'
```

Kết quả:

```
array([True, False, False, True, False, False, False], dtype=bool)
```

Mảng boolean này có thể được truyền khi indexing mảng:

```
data[names == 'Bob']
```

Kết quả:

```
array([[0.0929, 0.2817, 0.769 , 1.2464],
 [1.669 , -0.4386, -0.5397, 0.477]])
```

Mảng boolean phải có cùng độ dài với trực mảng đang được indexing. Cũng có thể kết hợp mảng boolean với slices hoặc số nguyên (hoặc chuỗi số nguyên; sẽ nói thêm về điều này sau).

Việc chọn bằng boolean sẽ không thất bại nếu mảng boolean không đúng độ dài, vì vậy cần cẩn thận khi sử dụng tính năng này.

Trong các ví dụ này, chọn từ các hàng nơi names == 'Bob' và cũng indexing các cột:

```
data[names == 'Bob', 2:]
```

Kết quả:

```
array([[0.769 , 1.2464],
 [-0.5397, 0.477]])
```

```
data[names == 'Bob', 3]
```

Kết quả:

```
array([1.2464, 0.477])
```

Để chọn tất cả ngoại trừ 'Bob', có thể sử dụng != hoặc phủ định điều kiện sử dụng ~:

```
names != 'Bob'
```

Kết quả:

```
array([False, True, True, False, True, True, True], dtype=bool)
```

```
data[~(names == 'Bob')]
```

Kết quả:

```
array([[1.0072, -1.2962, 0.275 , 0.2289],
 [1.3529, 0.8864, -2.0016, -0.3718],
 [3.2489, -1.0212, -0.5771, 0.1241],
 [0.3026, 0.5238, 0.0009, 1.3438],
 [-0.7135, -0.8312, -2.3702, -1.8608]])
```

Toán tử ~ có thể hữu ích khi muốn đảo ngược một điều kiện tổng quát:

```
cond = names == 'Bob'
data[~cond]
```

Kết quả:

```
array([[1.0072, -1.2962, 0.275 , 0.2289],
 [1.3529, 0.8864, -2.0016, -0.3718],
 [3.2489, -1.0212, -0.5771, 0.1241],
 [0.3026, 0.5238, 0.0009, 1.3438],
 [-0.7135, -0.8312, -2.3702, -1.8608]])
```

Để chọn hai trong ba tên để kết hợp nhiều điều kiện boolean, sử dụng các toán tử số học boolean như & (và) và | (hoặc):

```
mask = (names == 'Bob') | (names == 'Will')
mask
```

Kết quả:

```
array([True, False, True, True, True, False, False], dtype=bool)
```

```
data[mask]
```

Kết quả:

```
array([[0.0929, 0.2817, 0.769 , 1.2464],
 [1.3529, 0.8864, -2.0016, -0.3718],
 [1.669 , -0.4386, -0.5397, 0.477],
 [3.2489, -1.0212, -0.5771, 0.1241]])
```

Việc chọn dữ liệu từ một mảng bằng boolean indexing luôn tạo ra một bản sao của dữ liệu, ngay cả khi mảng trả về không thay đổi.

Các từ khóa and và or của Python không hoạt động với các mảng boolean. Sử dụng & (và) và | (hoặc) thay thế.

Gán giá trị với các mảng boolean hoạt động theo cách hợp lý. Để đặt tất cả các giá trị âm trong dữ liệu về 0, chỉ cần làm:

```
data[data < 0] = 0
data
```

Kết quả:

```
array([[0.0929, 0.2817, 0.769 , 1.2464],
 [1.0072, 0. , 0.275 , 0.2289],
 [1.3529, 0.8864, 0. , 0.],
 [1.669 , 0. , 0. , 0.477],
 [3.2489, 0. , 0. , 0.1241],
 [0.3026, 0.5238, 0.0009, 1.3438],
 [0. , 0. , 0. , 0.]])
```

Gán toàn bộ hàng hoặc cột bằng cách sử dụng một mảng boolean một chiều cũng dễ dàng:

```
data[names != 'Joe'] = 7
data
```

Kết quả:

```
array([[7. , 7. , 7. , 7.],
 [1.0072, 0. , 0.275 , 0.2289],
 [7. , 7. , 7. , 7.],
 [7. , 7. , 7. , 7.],
 [7. , 7. , 7. , 7.],
 [0.3026, 0.5238, 0.0009, 1.3438],
 [0. , 0. , 0. , 0.]])
```

Như sẽ thấy sau này, các loại thao tác này trên dữ liệu hai chiều rất tiện lợi khi sử dụng pandas.

## Fancy Indexing

Fancy indexing là thuật ngữ được NumPy sử dụng để mô tả việc indexing bằng cách sử dụng các mảng số nguyên. Giả sử có một mảng kích thước  $8 \times 4$ :

```
arr = np.empty((8, 4))
for i in range(8):
 arr[i] = i
arr
```

Kết quả:

```
array([[0., 0., 0., 0.],
 [1., 1., 1., 1.],
 [2., 2., 2., 2.],
 [3., 3., 3., 3.],
 [4., 4., 4., 4.],
 [5., 5., 5., 5.],
 [6., 6., 6., 6.],
 [7., 7., 7., 7.]])
```

Để chọn một tập hợp các hàng theo thứ tự cụ thể, chỉ cần truyền một danh sách hoặc ndarray các số nguyên xác định thứ tự mong muốn:

```
arr[[4, 3, 0, 6]]
```

Kết quả:

```
array([[4., 4., 4., 4.],
 [3., 3., 3., 3.],
 [0., 0., 0., 0.],
 [6., 6., 6., 6.]])
```

Sử dụng chỉ số âm để chọn các hàng từ cuối:

```
arr[[-3, -5, -7]]
```

Kết quả:

```
array([[5., 5., 5., 5.],
 [3., 3., 3., 3.],
 [1., 1., 1., 1.]])
```

Truyền nhiều mảng chỉ mục sẽ làm điều gì đó hơi khác; nó chọn một mảng một chiều của các phần tử tương ứng với mỗi bộ chỉ mục:

```
arr = np.arange(32).reshape((8, 4))
arr
```

Kết quả:

```
array([[0, 1, 2, 3],
 [4, 5, 6, 7],
 [8, 9, 10, 11],
 [12, 13, 14, 15],
 [16, 17, 18, 19],
 [20, 21, 22, 23],
 [24, 25, 26, 27],
 [28, 29, 30, 31]])
```

```
arr[[1, 5, 7, 2], [0, 3, 1, 2]]
```

Kết quả:

```
array([4, 23, 29, 10])
```

Trong trường hợp này, các phần tử  $(1, 0)$ ,  $(5, 3)$ ,  $(7, 1)$ , và  $(2, 2)$  đã được chọn. Bất kể mảng có bao nhiêu chiều (ở đây, chỉ có 2), kết quả của fancy indexing luôn là một chiều.

Hành vi của fancy indexing trong trường hợp này hơi khác so với những gì một số người dùng có thể mong đợi (bao gồm cả bản thân), đó là vùng hình chữ nhật được hình thành bằng cách chọn một tập hợp các hàng và cột của ma trận. Đây là một cách để đạt được điều đó:

```
arr[[1, 5, 7, 2]][:, [0, 3, 1, 2]]
```

Kết quả:

```
array([[4, 7, 5, 6],
 [20, 23, 21, 22],
 [28, 31, 29, 30],
 [8, 11, 9, 10]])
```

Lưu ý rằng fancy indexing, không giống như slicing, luôn sao chép dữ liệu vào một mảng mới.

## Transposing Arrays và Swapping Axes

Transposing là một dạng đặc biệt của reshaping mà tương tự trả về một view trên dữ liệu gốc mà không sao chép bất cứ thứ gì. Các mảng có phương thức transpose và thuộc tính đặc biệt T:

```
arr = np.arange(15).reshape((3, 5))
arr
```

Kết quả:

```
array([[0, 1, 2, 3, 4],
 [5, 6, 7, 8, 9],
 [10, 11, 12, 13, 14]])
```

```
arr.T
```

Kết quả:

```
array([[0, 5, 10],
 [1, 6, 11],
 [2, 7, 12],
 [3, 8, 13],
 [4, 9, 14]])
```

Khi thực hiện các phép toán ma trận, việc làm này thường rất hữu ích — ví dụ, khi tính tích ma trận bên trong bằng cách sử dụng np.dot:

```
arr = np.random.randn(6, 3)
arr
```

Kết quả:

```
array([[-0.8608, 0.5601, -1.2659],
 [0.1198, -1.0635, 0.3329],
 [-2.3594, -0.1995, -1.542],
 [-0.9707, -1.307 , 0.2863],
 [0.378 , -0.7539, 0.3313],
 [1.3497, 0.0699, 0.2467]])
```

```
np.dot(arr.T, arr)
```

Kết quả:

```
array([[9.2291, 0.9394, 4.948],
 [0.9394, 3.7662, -1.3622],
 [4.948 , -1.3622, 4.3437]])
```

Đối với các mảng có chiều cao hơn, transpose sẽ chấp nhận một bộ số trực để hoán đổi các trực:

```
arr = np.arange(16).reshape((2, 2, 4))
arr
```

Kết quả:

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

```
arr.transpose((1, 0, 2))
```

Kết quả:

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

Ở đây, các trục đã được sắp xếp lại với trục thứ hai trước, trục đầu tiên thứ hai, và trục cuối cùng không thay đổi.

Transposing đơn giản với .T là một trường hợp đặc biệt của swapping axes. ndarray có phương thức swapaxes, phương thức này nhận một cặp số trục và hoán đổi các trục được chỉ định để sắp xếp lại dữ liệu:

```
arr
```

Kết quả:

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

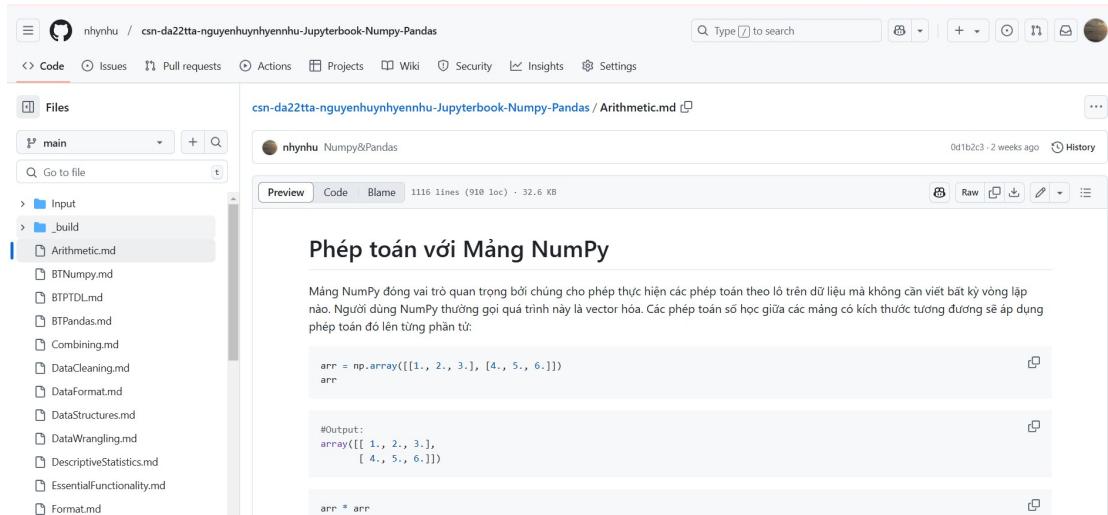
```
arr.swapaxes(1, 2)
```

Kết quả:

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

swapaxes tương tự trả về một view trên dữ liệu mà không tạo ra một bản sao.

## 4.3 Kết quả tài liệu trên Github



Hình 4.7 Tài liệu trên GitHub

## CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 5.1 Kết quả đạt được

Sau quá trình nghiên cứu và thực hiện, tài liệu học tập về Thống kê và Phân tích dữ liệu đã được xây dựng hoàn chỉnh với 7 chương, bao gồm lý thuyết chi tiết, ví dụ minh họa và bài tập thực hành. Nội dung tài liệu tập trung vào các nội dung lý thuyết từ cơ bản đến nâng cao. Các ví dụ mã nguồn sử dụng thư viện Python như NumPy, Pandas và Matplotlib đã được tích hợp, cho phép thực hành trực tiếp và kiểm tra kết quả ngay trên tài liệu. Bên cạnh đó, tài liệu được biên soạn bằng Jupyter Book với tính năng tương tác cao, hỗ trợ hiển thị biểu đồ, bảng dữ liệu và kết quả trực quan, giúp tăng trải nghiệm học tập. Tài liệu cũng đã được xuất bản online, hỗ trợ truy cập dễ dàng trên nhiều thiết bị và có thể chuyển đổi sang các định dạng PDF để sử dụng offline.

### 5.2 Hướng phát triển

Trong tương lai, tài liệu học tập sẽ được mở rộng bằng cách bổ sung thêm nhiều bài tập thực hành đa dạng, từ cơ bản đến nâng cao, nhằm củng cố kiến thức và kỹ năng phân tích dữ liệu cho người học. Ngoài ra, các bài tập sẽ được tích hợp chặt chẽ với nội dung lý thuyết và ví dụ minh họa để tạo sự liên mạch trong quá trình học tập. Tài liệu sẽ được cập nhật thêm các bài tập mở rộng với dữ liệu thực tế từ nhiều lĩnh vực như tài chính, y tế và khoa học xã hội để giúp người học ứng dụng kiến thức vào các tình huống cụ thể. Các bài tập tương tác sẽ được phát triển, cho phép người học chạy mã trực tiếp trên nền tảng Jupyter Book và kiểm tra kết quả ngay lập tức, qua đó tăng cường khả năng thực hành và thử nghiệm.

## DANH MỤC TÀI LIỆU THAM KHẢO

- [1] Jupyter Book, "Jupyter Book Documentation," *Jupyter Book: Interactive books for teaching and learning*, [Online]. Available: <https://jupyterbook.org/en/stable/intro.html>. [Accessed: Jan. 8, 2025].
- [2] Thaycacac. (2025, January 8). *Markdown là gì?* Viblo. <https://viblo.asia/p/markdown-la-gi-Ljy5VX9VZra>
- [3] W. McKinney, *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2017.
- [4] Sharma, A. (2021, April 9). *What is YAML? A beginner's guide.* CircleCI. <https://circleci.com/blog/what-is-yaml-a-beginner-s-guide/>
- [5] W3Schools.com. (n.d.). <https://www.w3schools.com/python/>