Những tư duy cuối cùng

Học lập trình không phải là việc dễ dàng, nhưng nếu bạn đã đến được đây, bạn đang có một khởi đầu tốt. Bây giờ, tôi có một số gợi ý về cách bạn có thể tiếp tục học và áp dụng những gì bạn đã học.

Cuốn sách này được thiết kế như một phần giới thiệu chung về lập trình, vì vậy chúng tôi không tập trung vào các ứng dụng cụ thể. Tùy thuộc vào sở thích của bạn, có rất nhiều lĩnh vực mà bạn có thể áp dụng các kỹ năng mới của mình.

Nếu bạn quan tâm đến Khoa học Dữ liệu, có ba cuốn sách của tôi mà bạn có thể thích:

- Think Stats: Exploratory Data Analysis, O'Reilly Media, 2014.
- Think Bayes: Bayesian Statistics in Python, O'Reilly Media, 2021.
- Think DSP: Digital Signal Processing in Python, O'Reilly Media, 2016.

Nếu bạn quan tâm đến mô hình hóa vật lý và các hệ thống phức tạp, bạn có thể thích:

- Modeling and Simulation in Python: An Introduction for Scientists and Engineers, No Starch Press, 2023.
- Think Complexity: Complexity Science and Computational Modeling, O'Reilly Media, 2018.

Các cuốn sách này sử dụng NumPy, SciPy, pandas và các thư viện Python khác dành cho khoa học dữ liệu và tính toán khoa học.

Cuốn sách này cố gắng tìm sự cân bằng giữa các nguyên lý chung của lập trình và các chi tiết của Python. Kết quả là, nó không bao gồm tất cả các tính năng của ngôn ngữ Python. Để biết thêm về Python và những lời khuyên tốt về cách sử dụng nó, tôi khuyến nghị cuốn Fluent Python: Clear, Concise, and Effective Programming, phiên bản thứ hai của Luciano Ramalho, O'Reilly Media, 2022.

Sau khi giới thiệu về lập trình, một bước tiếp theo phổ biến là học về cấu trúc dữ liệu và thuật toán. Tôi có một tác phẩm đang trong quá trình hoàn thiện về chủ đề này, có tên **Data Structures and Information Retrieval in Python**. Một phiên bản điện tử miễn phí có sẵn từ Green Tea Press tại https://greenteapress.com.

Khi bạn làm việc với các chương trình phức tạp hơn, bạn sẽ gặp phải những thử thách mới. Bạn có thể thấy hữu ích khi ôn lại các phần trong cuốn sách này về việc gỡ lỗi. Cụ thể, hãy nhớ về Sáu R trong việc gỡ lỗi từ Chương 12: đọc, chạy, suy ngẫm, con vịt cao su, rút lui, và nghỉ ngơi.

Cuốn sách này gợi ý các công cụ giúp việc gỡ lỗi, bao gồm các hàm `print` và `repr`, hàm `structshape` trong Chương 11 -- và các hàm tích hợp `isinstance`, `hasattr`, và `vars` trong Chương 14.

Nó cũng gợi ý các công cụ để kiểm tra chương trình, bao gồm câu lệnh assert, mô-đun doctest, và mô-đun unittest. Việc bao gồm các bài kiểm tra trong chương trình của bạn là một trong những cách tốt nhất để ngăn ngừa và phát hiện lỗi, đồng thời tiết kiệm thời gian gỡ lỗi.

Tuy nhiên, loại gỡ lỗi tốt nhất là loại bạn không phải làm. Nếu bạn sử dụng quy trình phát triển gia tăng như đã mô tả trong Chương 6 -- và kiểm tra trong suốt quá trình — bạn sẽ gặp ít lỗi hơn và phát hiện chúng nhanh hơn khi chúng xảy ra. Ngoài ra, hãy nhớ về đóng gói và tổng quát hóa từ Chương 4, điều này đặc biệt hữu ích khi bạn phát triển mã trong các notebook Jupyter.

Trong suốt cuốn sách này, tôi đã gợi ý các cách sử dụng trợ lý ảo để giúp bạn học, lập trình và gỡ lỗi. Tôi hy vọng bạn thấy những công cụ này hữu ích.

Ngoài các trợ lý ảo như ChatGPT, bạn cũng có thể muốn sử dụng một công cụ như Copilot, giúp tự động hoàn thành mã khi bạn gõ. Ban đầu, tôi không khuyến nghị sử dụng những công cụ này vì chúng có thể gây quá tải cho người mới bắt đầu. Nhưng bạn có thể muốn khám phá chúng ngay bây giờ.

Việc sử dụng công cụ AI hiệu quả đòi hỏi một chút thử nghiệm và suy ngẫm để tìm ra một quy trình phù hợp với bạn. Nếu bạn thấy phiền phức khi phải sao chép mã từ ChatGPT vào Jupyter, bạn có thể sẽ thích những công cụ như Copilot. Tuy nhiên, công việc tư duy mà bạn làm để soạn thảo câu lệnh và giải thích phản hồi có thể có giá trị tương đương với mã mà công cụ tạo ra, theo cách tương tự như phương pháp gỡ lỗi **con vịt cao su**.

Khi bạn tích lũy kinh nghiệm lập trình, bạn có thể muốn khám phá các môi trường phát triển khác. Tôi nghĩ rằng các notebook Jupyter là một nơi tốt để bắt đầu, nhưng chúng còn khá mới và chưa phổ biến rộng rãi như các môi trường phát triển tích hợp (IDE) truyền thống. Đối với Python, các IDE phổ biến nhất bao gồm PyCharm và Spyder — và Thonny, thường được khuyến nghị cho người mới bắt đầu. Các IDE khác, như Visual Studio Code và Eclipse, cũng hỗ trợ các ngôn ngữ lập trình khác. Hoặc, như một lựa chọn đơn giản hơn, bạn có thể viết các chương trình Python bằng bất kỳ trình soạn thảo văn bản nào mà bạn thích.

Khi bạn tiếp tục hành trình lập trình của mình, bạn không cần phải đi một mình! Nếu bạn sống ở hoặc gần một thành phố, có thể sẽ có một nhóm người dùng Python mà bạn có thể tham gia. Những nhóm này thường rất thân thiện với người mới bắt đầu, vì vậy đừng lo lắng. Nếu không có nhóm gần bạn, bạn có thể tham gia các sự kiện trực tuyến. Ngoài ra, hãy chú ý đến các hội nghị Python khu vực.

Một trong những cách tốt nhất để cải thiện kỹ năng lập trình của bạn là học một ngôn ngữ khác. Nếu bạn quan tâm đến thống kê và khoa học dữ liệu, bạn có thể muốn học R. Tuy nhiên, tôi đặc biệt khuyến khích bạn học một ngôn ngữ hàm như Racket hoặc Elixir. Lập trình hàm yêu cầu một cách suy nghĩ khác, điều này thay đổi cách bạn nghĩ về các chương trình.

Chúc bạn may mắn!