

## PHÂN CÔNG NHIỆM VỤ

Tên tài liệu dịch	Tên thành viên	Chương đảm nhận
Python for kids	Trần Thị Uyên Nhi	Mở đầu, chương 1, 2 (Trang 1-34)
	Phan Thị Anh Trang	Chương 3,4 (Trang 35-65)
Firewalls and Internet Security	Nguyễn Khánh Linh	Chương 15 (Trang 66-71)
	Tôn Huyền Kim Khánh	Chương 19 (Trang 72-76)

# PYTHON FOR KIDS

A Playful Introduction to Programming

JASON R. BRIGGS



**Python for Kids**

**Jason R. Briggs**

Published by No Starch Press

## Về tác giả

Jason R. Briggs là một lập trình viên kỳ cựu, bắt đầu hành trình học lập trình từ năm 8 tuổi với ngôn ngữ BASIC trên máy tính Radio Shack TRS-80. Trong suốt sự nghiệp của mình, ông đã làm việc chuyên nghiệp với vai trò nhà phát triển và kiến trúc sư hệ thống, đóng góp đáng kể cho cộng đồng công nghệ.

Briggs có một danh mục bài viết phong phú, từng là Biên tập viên Cộng tác cho *Java Developer's Journal* và có các bài viết được đăng trên *JavaWorld*, *ONJava*, và *ONLamp*. Cuốn sách đầu tay của ông, *Python for Kids*, thể hiện khả năng làm cho lập trình trở nên dễ tiếp cận và thú vị với các bạn trẻ.

Để liên lạc hoặc tìm hiểu thêm, bạn có thể truy cập trang web của Jason R. Briggs tại [jasonrbriggs.com](http://jasonrbriggs.com) hoặc gửi email đến [mail@jasonrbriggs.com](mailto:mail@jasonrbriggs.com).

## **Về họa sĩ minh họa**

Miran Lipovača là tác giả của cuốn sách *Learn You a Haskell for Great Good!*. Anh ấy yêu thích quyền anh, chơi guitar bass, và tất nhiên là cả vẽ tranh. Miran còn bị cuốn hút bởi những bộ xương nhảy múa và con số 71. Khi đi qua các cửa tự động, anh thường giả vờ rằng mình đang mở cửa bằng sức mạnh trí não.

## Lời cảm ơn

Viết lời cảm ơn giống như khi bạn bước lên sân khấu nhận giải thưởng, nhưng chợt nhận ra rằng danh sách những người cần cảm ơn lại dễ quên trong chiếc quần khác: bạn chắc chắn sẽ quên ai đó, và ngay sau đó nhạc nền sẽ vang lên để đẩy bạn rời khỏi sân khấu một cách nhanh chóng.

Vì vậy, dưới đây là danh sách (chắc chắn không đầy đủ) những người mà tôi vô cùng biết ơn vì đã giúp biến cuốn sách này trở nên tốt nhất có thể:

Cảm ơn đội ngũ của No Starch, đặc biệt là Bill Pollock, vì đã áp dụng tư duy “một đứa trẻ sẽ nghĩ gì” khi chỉnh sửa cuốn sách này. Khi bạn đã lập trình trong một thời gian dài, bạn dễ dàng quên mất rằng việc học những điều này khó khăn thế nào với người mới bắt đầu. Bill đã đóng vai trò không thể thiếu trong việc chỉ ra những phần thường bị bỏ qua và làm quá phức tạp. Cảm ơn Serena Yang, người quản lý sản xuất xuất sắc; hy vọng bạn chưa phải nhổ quá nhiều tóc khi xử lý hơn 300 trang mã nguồn để tô màu chính xác.

Một lời cảm ơn lớn gửi đến Miran Lipovača vì những minh họa tuyệt vời. Không chỉ tuyệt vời, mà còn vượt ngoài mong đợi! Thật đấy! Nếu tôi tự làm phần minh họa, có lẽ chúng ta chỉ có vài hình vẽ nguệch ngoạc chẳng giống thứ gì cả. Đây là một con gấu... ? Hay là một con chó... ? Không, khoan đã... đó có phải là một cái cây không?

Cảm ơn các nhà phê bình. Tôi xin lỗi nếu một số đề xuất của bạn cuối cùng không được áp dụng. Có thể bạn đúng, và tôi chỉ có thể đổ lỗi cho những thiếu sót cá nhân trong tính cách của mình khi mắc phải những sai lầm đáng tiếc. Đặc biệt cảm ơn Josh vì những đề xuất tuyệt vời và những phát hiện tinh tế. Và cũng xin lỗi Maria vì đôi khi phải xử lý những đoạn mã được định dạng không mấy chuẩn.

Cảm ơn vợ và con gái tôi, vì đã chịu đựng một người chồng và người cha dành nhiều thời gian dán mắt vào màn hình máy tính hơn bình thường.

Cảm ơn mẹ, người đã luôn khuyến khích tôi trong suốt những năm qua.

Và cuối cùng, cảm ơn cha, vì đã mua một chiếc máy tính từ những năm 1970 và chịu đựng một người muốn sử dụng nó như giống như anh ấy đã làm. Tất cả những điều này sẽ không thể xảy ra nếu không có ông.

## Tại sao nên học lập trình máy tính?

Lập trình thúc đẩy sự sáng tạo, tư duy logic và khả năng giải quyết vấn đề. Người lập trình có cơ hội tạo ra thứ gì đó từ con số không, sử dụng logic để biến các cấu trúc lập trình thành những thứ máy tính có thể thực thi, và khi mọi thứ không hoạt động như mong đợi, họ sẽ vận dụng kỹ năng giải quyết vấn đề để tìm ra nguyên nhân.

Lập trình là một hoạt động thú vị, đôi khi thách thức (và cũng có lúc gây nản lòng), nhưng các kỹ năng học được từ lập trình có thể rất hữu ích cả trong học tập và công việc, ngay cả khi sự nghiệp của bạn không liên quan trực tiếp đến máy tính. Và nếu không vì điều gì khác, lập trình là một cách tuyệt vời để dành một buổi chiều khi thời tiết bên ngoài ảm đạm.

## Tại sao là Python?

Python là một ngôn ngữ lập trình dễ học với nhiều tính năng hữu ích cho người mới bắt đầu. Mã nguồn trong Python dễ đọc hơn so với nhiều ngôn ngữ lập trình khác, và nó có một shell tương tác để bạn có thể nhập và chạy các chương trình ngay lập tức.

Ngoài cấu trúc ngôn ngữ đơn giản và shell tương tác, Python còn có một số tính năng hỗ trợ rất tốt cho quá trình học tập, đồng thời cho phép bạn tạo các hình ảnh động đơn giản để phát triển trò chơi của riêng mình. Một trong số đó là **module turtle**, lấy cảm hứng từ đồ họa Turtle (được sử dụng bởi ngôn ngữ lập trình Logo từ những năm 1960) và được thiết kế cho mục đích giáo dục. Một tính năng khác là **module tkinter**, một giao diện cho công cụ Tk GUI, cung cấp cách đơn giản để tạo các chương trình với đồ họa và hình ảnh động phức tạp hơn.

## Làm thế nào để học lập trình?

Như bất kỳ điều gì bạn thử lần đầu tiên, luôn luôn tốt nhất là bắt đầu từ những điều cơ bản. Hãy bắt đầu từ các chương đầu tiên và cố gắng không bỏ qua để nhảy ngay đến các



chương sau. Không ai có thể chơi một bản giao hưởng ngay lần đầu tiên cầm nhạc cụ. Các phi công tập sự cũng không thể lái máy bay trước khi hiểu rõ các điều khiển cơ bản. Và vận động viên thể dục dụng cụ (thường thì) cũng không thể thực hiện lộn nhào ngay lần thử đầu tiên. Nếu bạn học quá nhanh, các ý tưởng cơ bản không chỉ khó in sâu vào trí nhớ, mà bạn còn cảm thấy nội dung ở các chương sau phức tạp hơn thực tế. Khi học qua cuốn sách này, hãy thử từng ví dụ để hiểu cách chúng hoạt động. Hầu hết các chương đều có những câu đố lập trình ở cuối để bạn thực hành, giúp cải thiện kỹ năng lập trình của mình. Hãy nhớ rằng, bạn càng hiểu rõ các kiến thức cơ bản, việc tiếp thu các ý tưởng phức tạp sau này sẽ dễ dàng hơn.

Khi gặp khó khăn hoặc thử thách, nếu bạn gặp phải điều gì đó khó khăn hoặc quá thách thức, đây là vài lời khuyên hữu ích:

Chia nhỏ vấn đề: Hãy chia vấn đề lớn thành các phần nhỏ hơn. Tập trung vào việc hiểu từng đoạn mã hoặc một phần nhỏ của ý tưởng phức tạp thay vì cố gắng hiểu toàn bộ ngay một lúc.

Tạm nghỉ: Nếu cách trên không hiệu quả, hãy tạm ngưng. Nghỉ ngơi, ngủ một giấc, và quay lại sau. Đây là cách giải quyết vấn đề hiệu quả, đặc biệt đối với lập trình viên máy tính.

### **Ai nên đọc cuốn sách này?**

Cuốn sách này dành cho bất kỳ ai quan tâm đến lập trình máy tính, dù là trẻ em hay người lớn lần đầu tiên tiếp xúc với lập trình. Nếu bạn muốn học cách tự viết phần mềm của mình thay vì chỉ sử dụng các chương trình do người khác tạo ra, *Python for Kids* là một điểm khởi đầu tuyệt vời.

Trong các chương tiếp theo, bạn sẽ tìm thấy hướng dẫn: cài đặt Python. Bắt đầu với shell Python thực hiện các phép tính cơ bản, in văn bản ra màn hình, tạo danh sách, và thực hiện các thao tác điều khiển đơn giản như câu lệnh if và vòng lặp for (và hiểu chúng hoạt

động ra sao!). Tái sử dụng mã bằng cách sử dụng hàm. Tìm hiểu các kiến thức cơ bản về lớp và đối tượng. Làm quen với nhiều hàm và module tích hợp sẵn của Python.

Bạn sẽ học cách sử dụng **đồ họa turtle** cơ bản và nâng cao, cũng như module **tkinter** để vẽ trên màn hình máy tính. Các chương có kèm theo những câu đố lập trình với độ phức tạp khác nhau, giúp bạn củng cố kiến thức đã học bằng cách tự viết các chương trình nhỏ.

Khi bạn đã nắm vững kiến thức lập trình cơ bản, cuốn sách sẽ hướng dẫn bạn viết các trò chơi của riêng mình. Bạn sẽ phát triển hai trò chơi đồ họa, học về phát hiện va chạm (collision detection), xử lý sự kiện (events), và các kỹ thuật hoạt hình khác nhau.

Hầu hết các ví dụ trong sách sử dụng **IDLE (Integrated DeveLopment Environment)** của Python. IDLE cung cấp: Tô sáng cú pháp giúp mã dễ đọc hơn. Chức năng sao chép và dán tương tự như các ứng dụng khác. Một cửa sổ soạn thảo để bạn lưu mã của mình và sử dụng lại sau. IDLE không chỉ là một môi trường tương tác cho việc thử nghiệm mà còn hoạt động như một trình soạn thảo văn bản. Mặc dù các ví dụ trong sách có thể hoạt động tốt với cửa sổ dòng lệnh tiêu chuẩn và trình soạn thảo văn bản thông thường, nhưng IDLE với cú pháp tô sáng và giao diện thân thiện hơn sẽ hỗ trợ bạn hiểu dễ dàng hơn. Vì vậy, chương đầu tiên sẽ hướng dẫn bạn cách thiết lập IDLE.

## **Nội dung sách này**

Dưới đây là tổng quan ngắn gọn về những gì bạn sẽ tìm thấy trong từng chương.

- **Chương 1** là phần giới thiệu về lập trình với hướng dẫn cài đặt Python lần đầu tiên.
- **Chương 2** giới thiệu về phép tính cơ bản và biến, và **Chương 3** mô tả một số kiểu dữ liệu cơ bản của Python, chẳng hạn như chuỗi, danh sách và tuple.
- **Chương 4** cung cấp trải nghiệm đầu tiên với module turtle, từ lập trình cơ bản đến việc điều khiển một con rùa (hình mũi tên) trên màn hình.

- **Chương 5** bao phủ các điều kiện và câu lệnh if, và **Chương 6** tiếp tục với vòng lặp for và while.
- **Chương 7** bắt đầu sử dụng và tạo hàm, và **Chương 8** giới thiệu về các lớp và đối tượng. Ở giai đoạn này, nội dung bắt đầu có chút phức tạp hơn.
- **Chương 9** trình bày hầu hết các hàm tích hợp sẵn của Python, và **Chương 10** tiếp tục với một vài module (các chức năng hữu ích) được cài đặt sẵn trong Python.
- **Chương 11** quay trở lại với module turtle khi người đọc thử nghiệm với các hình phức tạp hơn. **Chương 12** chuyển sang sử dụng module tkinter để tạo các hình ảnh đồ họa nâng cao hơn.
- **Chương 13** và **Chương 14** tạo ra game đầu tiên, “Bounce!”, dựa trên kiến thức từ các chương trước, và **Chương 15–18** tạo ra một game khác, “Mr. Stick Man Races for the Exit.” Các chương phát triển game có thể trở nên phức tạp hơn và đôi khi có thể gặp lỗi. Nếu gặp khó khăn, bạn có thể tải mã nguồn từ trang web hỗ trợ đi kèm (<http://python-for-kids.com/>) và so sánh với các ví dụ làm việc sẵn.
- Trong **Phụ lục A**, bạn sẽ tìm hiểu về PyGame và một số ngôn ngữ lập trình phổ biến khác.
- Cuối cùng, trong **Phụ lục B**, bạn sẽ tìm hiểu chi tiết về các từ khóa của Python, và trong **Bảng thuật ngữ**, bạn sẽ tìm thấy các định nghĩa về các thuật ngữ lập trình được sử dụng xuyên suốt cuốn sách này.

## Trang web hỗ trợ

Nếu bạn cần sự trợ giúp trong quá trình đọc sách, hãy thử trang web hỗ trợ tại <http://python-for-kids.com/>. Tại đây, bạn sẽ tìm thấy các bản tải xuống cho tất cả các ví dụ trong sách và nhiều bài toán lập trình khác. Bạn cũng sẽ tìm thấy giải pháp cho tất cả các bài toán lập trình trong sách trên trang web, phòng trường hợp bạn gặp khó khăn hoặc muốn kiểm tra lại công việc của mình.

**Chúc bạn vui vẻ!**

Hãy nhớ rằng, trong suốt quá trình học tập qua cuốn sách này, lập trình có thể rất vui vẻ. Đừng coi đây là công việc. Hãy nghĩ về lập trình như một cách để tạo ra những trò chơi hoặc ứng dụng thú vị mà bạn có thể chia sẻ với bạn bè hoặc người khác. Học lập trình là một bài tập tinh thần tuyệt vời và kết quả đạt được có thể rất xứng đáng. Nhưng hơn hết, dù bạn làm gì, hãy luôn tận hưởng và vui vẻ với hành trình này!

## **Phần I. Học lập trình**

### **Chương 1. Không Phải Tất Cả Những Con Rắn Đầu Trườn**

Một chương trình máy tính là tập hợp các hướng dẫn khiến máy tính thực hiện một loại hành động nào đó. Nó không phải là những phần cứng vật lý của máy tính-như dây điện, vi mạch, thẻ, ổ cứng và các linh kiện khác-mà là những thứ ẩn giấu đang chạy trên phần cứng đó. Một chương trình máy tính, mà tôi thường gọi là chương trình, là tập hợp các lệnh chỉ dẫn cho phần cứng vô tri kia phải làm gì. Phần mềm là một tập hợp các chương trình máy tính.

Không có chương trình máy tính, hầu như mọi thiết bị bạn sử dụng hàng ngày đều sẽ ngừng hoạt động hoặc trở nên kém hữu ích hơn rất nhiều. Các chương trình máy tính, ở một hình thức nào đó, điều khiển không chỉ máy tính cá nhân của bạn mà còn hệ thống chơi game, điện thoại di động và các hệ thống định vị GPS trong xe hơi. Phần mềm cũng điều khiển các thiết bị ít rõ ràng hơn như TV LCD và điều khiển từ xa của chúng, cũng như một số radio mới nhất, đầu đĩa DVD, lò nướng và một số tủ lạnh. Ngay cả động cơ ô tô, đèn giao thông, đèn đường, tín hiệu tàu hỏa, bảng quảng cáo điện tử và thang máy đều được điều khiển bởi các chương trình.

Chương trình giống như những suy nghĩ. Nếu bạn không có suy nghĩ, bạn có thể chỉ ngồi trên sàn nhà, nhìn chăm chăm và chảy nước dãi trước ngực. Suy nghĩ “đứng lên khỏi sàn” là một lệnh, hoặc mệnh lệnh, hướng dẫn cơ thể bạn đứng lên. Tương tự như vậy, các chương trình máy tính chỉ dẫn cho máy tính phải làm gì.

Nếu bạn biết cách viết chương trình máy tính, bạn có thể làm được rất nhiều điều hữu ích. Tất nhiên, bạn có thể chưa thể viết các chương trình điều khiển ô tô, đèn giao thông hoặc tủ lạnh của mình (ít nhất là chưa đầu tiên), nhưng bạn có thể tạo ra trang web, viết các trò chơi của riêng mình hoặc thậm chí tạo chương trình giúp bạn hoàn thành bài tập về nhà.

## Một Vài Lời Về Ngôn Ngữ

Giống như con người, máy tính sử dụng nhiều ngôn ngữ khác nhau để giao tiếp—trong trường hợp này, đó là ngôn ngữ lập trình. Ngôn ngữ lập trình đơn giản là cách mà con người và máy tính cùng hiểu để thực hiện các hướng dẫn.

Có những ngôn ngữ lập trình được đặt tên theo người (như Ada và Pascal), những ngôn ngữ được đặt tên bằng các chữ viết tắt (như BASIC và FORTRAN), và thậm chí một số ngôn ngữ được đặt tên theo các chương trình truyền hình, như Python. Đúng vậy, ngôn ngữ lập trình Python được đặt theo tên của chương trình hài Monty Python’s Flying Circus, chứ không phải rắn python.

## GHI CHÚ

Monty Python’s Flying Circus là một chương trình hài kịch thay thế của Anh, lần đầu tiên được phát sóng vào những năm 1970 và vẫn rất nổi tiếng đến ngày nay với một lượng lớn người hâm mộ. Chương trình có các đoạn hài như “The Ministry of Silly Walks,” “The Fish-Slapping Dance,” và “The Cheese Shop” (nơi không bán bất kỳ loại pho mát nào).

Một số điều về ngôn ngữ lập trình Python khiến nó cực kỳ hữu ích cho người mới bắt đầu. Quan trọng nhất, bạn có thể viết các chương trình đơn giản, hiệu quả một cách nhanh chóng với Python. Python không có nhiều ký hiệu phức tạp, như dấu ngoặc nhọn ({ }), dấu thăng (#), và dấu đô la (\$), vốn khiến các ngôn ngữ lập trình khác trở nên khó đọc hơn và kém thân thiện hơn với người mới.

## Cài Đặt Python

Cài đặt Python khá đơn giản. Dưới đây là các bước để cài đặt Python trên Windows 7, Mac OS X và Ubuntu. Khi cài đặt Python, bạn cũng sẽ thiết lập một shortcut cho chương trình IDLE, là môi trường phát triển tích hợp (Integrated Development Environment - IDE) cho phép bạn viết chương trình cho Python. Nếu Python đã được cài đặt trên máy

tính của bạn, hãy chuyển đến phần "Once You've Installed Python" (Khi bạn đã cài đặt Python).

## Cài đặt Python trên Windows 7

Để cài đặt Python cho hệ điều hành Microsoft Windows 7, hãy mở trình duyệt web và truy cập vào địa chỉ <http://www.python.org/>, sau đó tải xuống trình cài đặt mới nhất của Python phiên bản 3. Hãy tìm một phần trong menu có tiêu đề Quick Links, như hình dưới đây:



### Lưu ý

Phiên bản chính xác của Python bạn tải về không quan trọng, miễn là nó bắt đầu bằng số 3.

Sau khi tải xuống trình cài đặt Windows, hãy nhấp đúp vào biểu tượng của nó, sau đó làm theo hướng dẫn để cài đặt Python ở vị trí mặc định như sau:

1. Chọn **Install for All Users** (Cài đặt cho tất cả người dùng), sau đó nhấn **Next**.
2. Giữ nguyên thư mục cài đặt mặc định, nhưng hãy lưu ý tên của thư mục cài đặt (có thể là C:\Python31 hoặc C:\Python32). Nhấn **Next**.
3. Bỏ qua phần **Customize Python** trong quá trình cài đặt, sau đó nhấn **Next**.

Kết thúc quá trình này, bạn sẽ có một mục **Python 3** trong menu Start của mình.



Tiếp theo, làm theo các bước sau để thêm lối tắt Python 3 vào màn hình chính của bạn:

1. Nhấp chuột phải vào màn hình chính của bạn, rồi chọn **New ▶ Shortcut** từ menu bật lên.
2. Nhập vào ô nơi có dòng **Type the location of the item** (đảm bảo rằng thư mục bạn nhập vào là giống với thư mục bạn đã ghi lại trước đó):

`c:\Python32\Lib\idlelib\idle.pyw -n`

3. Hộp thoại của bạn sẽ trông như thế này:

`C:\Python32\Lib\idlelib\idle.pyw -n`

Hộp thoại của bạn sẽ trông như sau:





Nhấn **Next** để chuyển đến hộp thoại tiếp theo.

Nhập tên là **IDLE**, sau đó nhấn **Finish** để tạo lối tắt.

Bây giờ bạn có thể chuyển đến phần **Once You've Installed Python** trong **Once You've Installed Python** để bắt đầu với Python.

Nếu bạn đang sử dụng máy Mac, bạn sẽ tìm thấy một phiên bản Python đã được cài đặt sẵn, nhưng đó có thể là phiên bản cũ của ngôn ngữ này. Để chắc chắn rằng bạn đang sử dụng phiên bản mới nhất, hãy mở trình duyệt của bạn và truy cập vào <http://www.python.org/getit/> để tải xuống trình cài đặt mới nhất cho Mac.

## Cài đặt Python trên MAC OS X

Có hai phiên bản trình cài đặt khác nhau. Bạn nên tải xuống phiên bản phù hợp tùy thuộc vào phiên bản Mac OS X bạn đang sử dụng. (Để kiểm tra, hãy nhấp vào biểu tượng Apple trên thanh menu phía trên, sau đó chọn **About this Mac**.) Chọn trình cài đặt như sau:

- Nếu bạn đang sử dụng Mac OS X phiên bản từ 10.3 đến 10.6, hãy tải xuống phiên bản 32-bit của Python 3 dành cho i386/PPC.

- Nếu bạn đang sử dụng Mac OS X phiên bản 10.6 trở lên, hãy tải xuống phiên bản 64-bit/32-bit của Python 3 dành cho x86-64.

Sau khi tệp đã được tải xuống (nó sẽ có phần mở rộng tên tệp là .dmg), hãy nhấp đúp vào nó. Bạn sẽ thấy một cửa sổ hiển thị nội dung của tệp.



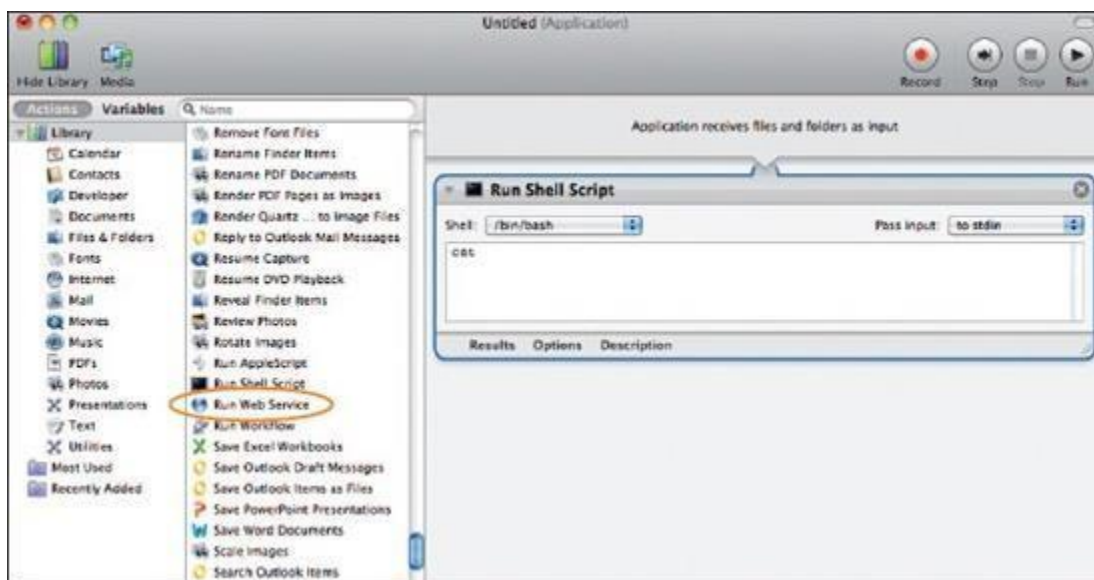
Trong cửa sổ này, hãy nhấp đúp vào tệp Python.mpkg, sau đó làm theo hướng dẫn để cài đặt phần mềm. Bạn sẽ được yêu cầu nhập mật khẩu quản trị viên cho máy Mac của bạn trước khi Python được cài đặt. (Không biết mật khẩu quản trị viên? Cha mẹ của bạn có thể cần nhập nó.)

Tiếp theo, bạn cần thêm một kịch bản vào màn hình desktop để khởi chạy ứng dụng IDLE của Python, như sau:

1. Nhấp vào biểu tượng Spotlight, biểu tượng kính lúp nhỏ ở góc trên cùng bên phải của màn hình.
2. Trong hộp văn bản xuất hiện, nhập Automator.
3. Nhấp vào ứng dụng có hình dáng như rô-bốt khi nó xuất hiện trong menu. Nó sẽ ở trong phần có tên là Top Hit hoặc trong Applications.
4. Khi Automator mở ra, chọn mẫu **Application**.



Nhấp **Choose** để tiếp tục. Trong danh sách các hành động, tìm **Run Shell Script** và kéo nó vào khung trống ở bên phải. Bạn sẽ thấy một cái gì đó tương tự như thế này:



Trong hộp văn bản, bạn sẽ thấy từ **cat**. Chọn từ đó và thay thế bằng đoạn văn bản sau (mọi thứ từ **open** đến **-n**):

```
open -a "/Applications/Python 3.2/IDLE.app" --args -n
```

Bạn có thể cần thay đổi thư mục tùy thuộc vào phiên bản Python bạn đã cài đặt.

Chọn **File** ► **Save**, và nhập **IDLE** làm tên.

Chọn **Desktop** từ hộp thoại **Where**, sau đó nhấn **Save**.

Bây giờ bạn có thể tiếp tục đến mục **Once You've Installed Python** trong *Once You've Installed Python* để bắt đầu với Python.

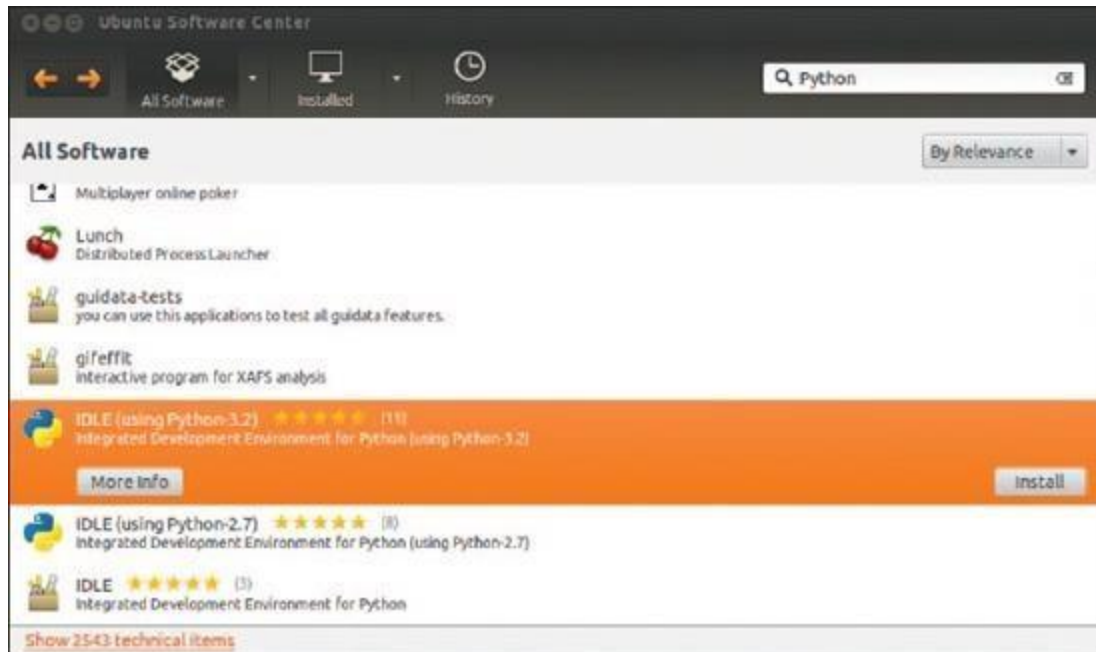
## Cài đặt Python trên Ubuntu

Python đã được cài đặt sẵn trên bản phân phối Ubuntu Linux, nhưng có thể là phiên bản cũ hơn. Hãy làm theo các bước sau để cài đặt Python 3 trên Ubuntu 12.x:

Nhấn vào nút của Trung tâm Phần mềm Ubuntu trong Thanh bên (đó là biểu tượng trông giống như một túi cam-nếu bạn không nhìn thấy nó, bạn có thể luôn nhấn vào biểu tượng Dash Home và nhập Software trong hộp thoại).

Nhập Python vào hộp tìm kiếm ở góc trên bên phải của Trung tâm Phần mềm.

Trong danh sách phần mềm hiển thị, chọn phiên bản mới nhất của IDLE, ví dụ như IDLE (using Python 3.2).



Nhấn **Cài đặt**.

Nhập mật khẩu quản trị viên của bạn để cài đặt phần mềm, sau đó nhấn **Xác thực**.  
(Không có mật khẩu quản trị viên? Bạn có thể yêu cầu cha mẹ nhập).

## LƯU Ý

Ở một số phiên bản của Ubuntu, bạn có thể chỉ nhìn thấy **Python (v3.2)** trong menu chính (thay vì IDLE)-bạn có thể cài đặt phiên bản này thay thế.

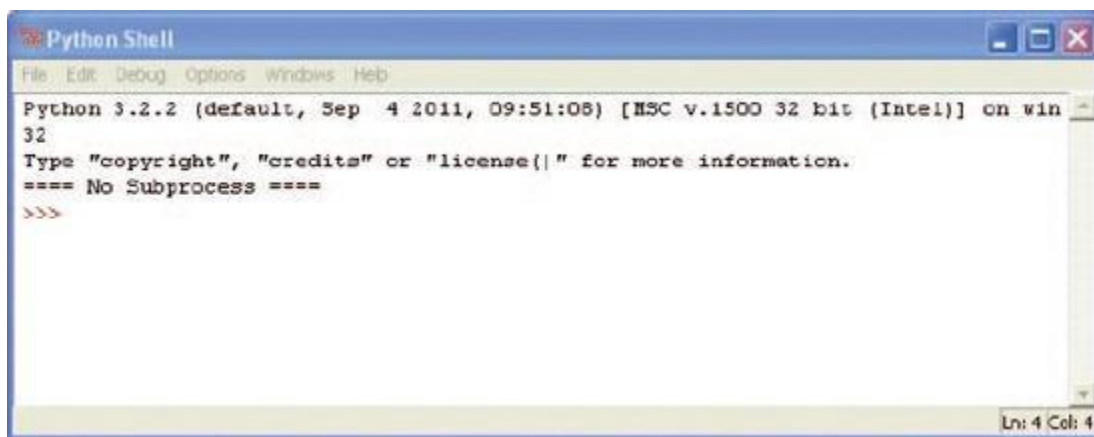
Giờ đây, bạn đã cài đặt phiên bản mới nhất của Python, hãy thử sử dụng nó.

## Khi bạn đã cài đặt Python



Bạn giờ đây sẽ có một biểu tượng trên màn hình nền Windows hoặc Mac OS X với tên IDLE. Nếu bạn sử dụng Ubuntu, trong menu Ứng dụng, bạn sẽ thấy một nhóm mới tên là Lập trình với ứng dụng IDLE (dùng Python 3.2) (hoặc phiên bản mới hơn).

Nhấp đúp vào biểu tượng hoặc chọn tùy chọn trong menu, và bạn sẽ thấy cửa sổ này:



Đây là Python shell, một phần của môi trường phát triển tích hợp của Python. Ba dấu ngoặc nhọn (>>>) gọi là dấu nhắc.

Hãy nhập một số lệnh tại dấu nhắc, bắt đầu với lệnh sau:

```
>>> print("Hello World")
```

Đảm bảo bao gồm cả dấu ngoặc kép (" "). Nhấn Enter trên bàn phím khi bạn đã nhập xong dòng lệnh. Nếu bạn nhập lệnh chính xác, bạn sẽ thấy một cái gì đó như thế này:

```
>>>
```

```
print("Hello World")
```

```
Hello World
```

```
>>>
```

Lời nhắc sẽ xuất hiện trở lại để cho bạn biết rằng Python shell đã sẵn sàng chấp nhận các lệnh khác. Chúc mừng! Bạn vừa tạo ra chương trình Python đầu tiên của mình. Từ `print` là một loại lệnh trong Python gọi là **hàm**, và nó sẽ in ra bất kỳ nội dung nào trong dấu ngoặc đơn ra màn hình. Nói cách khác, bạn đã đưa ra một lệnh cho máy tính để hiển thị từ "Hello World"—một lệnh mà cả bạn và máy tính đều có thể hiểu được.



## Lưu các chương trình Python của bạn

Chương trình Python sẽ không hữu ích lắm nếu bạn phải viết lại chúng mỗi lần muốn sử dụng, chưa kể đến việc in chúng ra để tham khảo. Điều này có thể ổn với các chương trình ngắn, nhưng với những chương trình lớn, như một trình xử lý văn bản, có thể chứa hàng triệu dòng mã. In ra toàn bộ, bạn sẽ có hơn 100.000 trang giấy. Hãy tưởng tượng mang chồng giấy khổng lồ đó về nhà—chỉ cần một cơn gió lớn thôi cũng đủ làm bạn gặp rắc rối!

May mắn thay, chúng ta có thể lưu các chương trình để sử dụng sau này. Để lưu một chương trình mới, hãy làm theo các bước sau:

1. **Mở IDLE** và chọn **File ▶ New Window**.

Một cửa sổ trống sẽ xuất hiện, với tiêu đề *Untitled* trên thanh menu.

2. Nhập đoạn mã sau vào cửa sổ mới:

```
print("Hello World")
```

Bây giờ, hãy làm theo các bước sau để lưu và chạy chương trình Python của bạn:

1. **Lưu chương trình:**

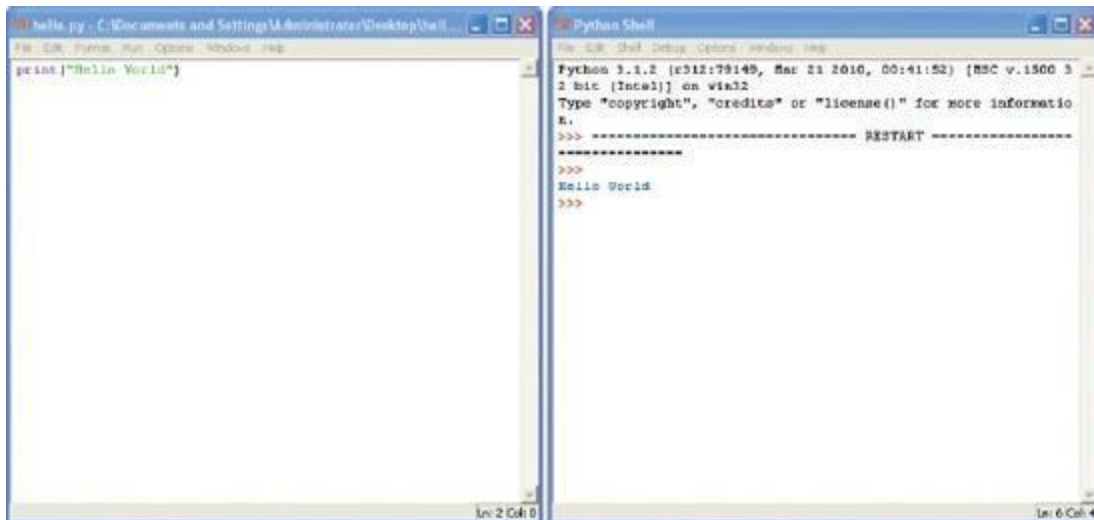
- Nhấn vào **File ▶ Save**.
- Khi được yêu cầu nhập tên tệp, hãy gõ **hello.py**, và lưu tệp này vào màn hình desktop của bạn.

2. **Chạy chương trình:**

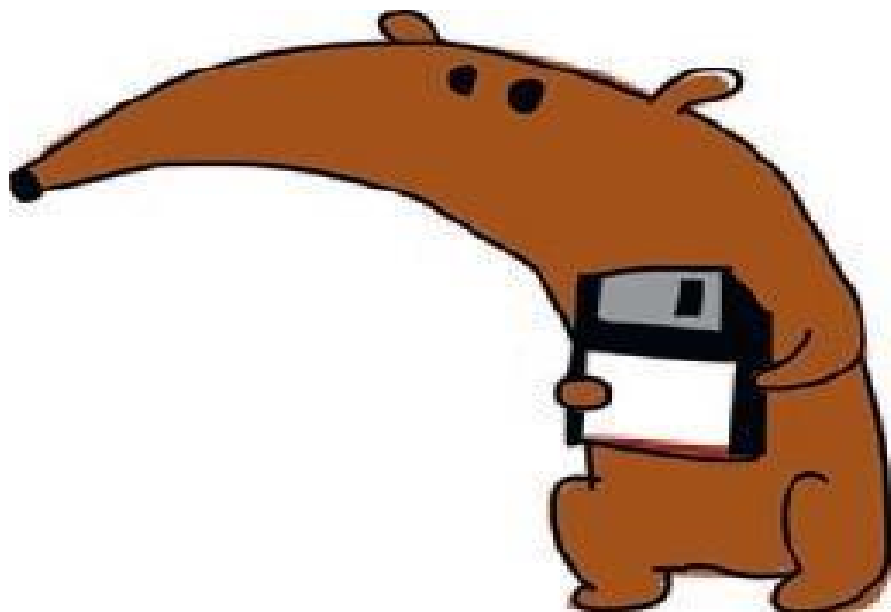
- Sau khi lưu, chọn **Run ▶ Run Module** hoặc nhấn phím tắt **F5** trên bàn phím.

Nếu mọi thứ diễn ra suôn sẻ, chương trình bạn đã lưu sẽ chạy, và bạn sẽ thấy kết quả hiển thị trong Python shell:





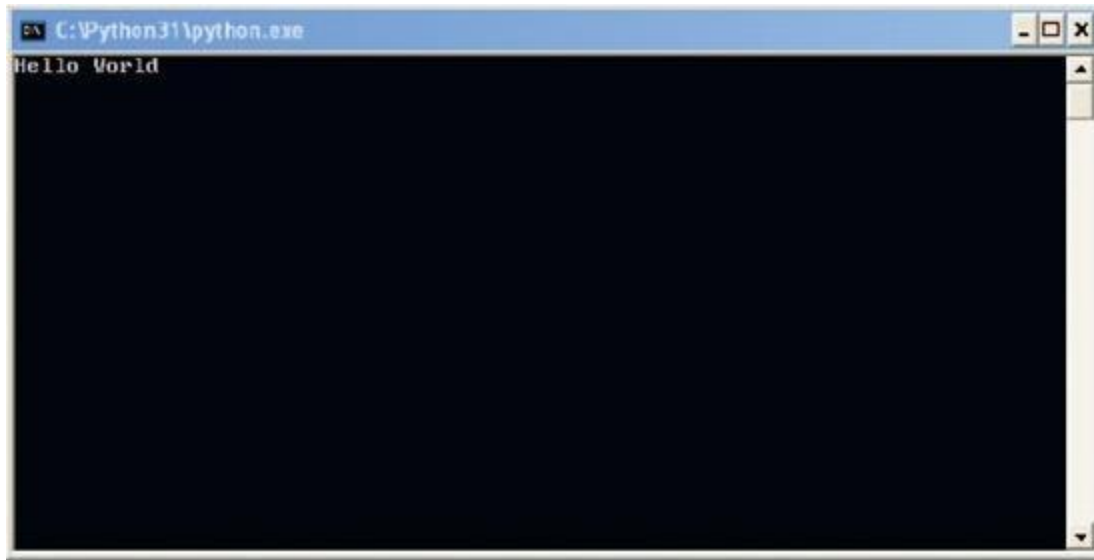
Bây giờ, nếu bạn đóng cửa sổ Python Shell nhưng giữ cửa sổ hello.py mở và sau đó chọn Run ► Run Module, cửa sổ Python Shell sẽ xuất hiện lại và chương trình của bạn sẽ chạy lần nữa. (Để mở lại Python Shell mà không chạy chương trình, hãy chọn Run ► Python Shell.)



Sau khi chạy mã, bạn sẽ thấy một biểu tượng mới trên màn hình máy tính với nhãn hello.py. Nếu bạn nhấp đúp vào biểu tượng này, một cửa sổ màu đen sẽ xuất hiện trong chốc lát rồi biến mất. Chuyện gì đã xảy ra?

Bạn đang thấy bảng điều khiển dòng lệnh của Python (tương tự như shell) khởi động, in ra “Hello World,” và sau đó thoát ra.

Đây là những gì bạn sẽ thấy nếu bạn có khả năng quan sát siêu tốc như siêu anh hùng và có thể nhìn thấy cửa sổ trước khi nó đóng lại:



Ngoài các menu, bạn cũng có thể sử dụng phím tắt để tạo một cửa sổ shell mới, lưu tệp và chạy chương trình:

- Trên Windows và Ubuntu, sử dụng **Ctrl + N** để tạo một cửa sổ shell mới, sử dụng **Ctrl + S** để lưu tệp sau khi hoàn tất chỉnh sửa, và nhấn **F5** để chạy chương trình.
- Trên Mac OS X, sử dụng **⌘ + N** để tạo một cửa sổ shell mới, sử dụng **⌘ + S** để lưu tệp, và giữ phím chức năng (fn) rồi nhấn **F5** để chạy chương trình.

## Bạn đã học được gì?

Chúng ta bắt đầu một cách đơn giản trong chương này với một ứng dụng Hello World—chương trình gần như mọi người đều bắt đầu khi học lập trình máy tính. Trong chương tiếp theo, chúng ta sẽ làm thêm một số điều hữu ích khác với Python shell.

## Chương 2. Tính Toán và Biến

Bây giờ bạn đã cài đặt Python và biết cách khởi động Python shell, bạn đã sẵn sàng để bắt đầu sử dụng nó. Chúng ta sẽ bắt đầu với một số tính toán đơn giản và sau đó chuyển

sang các biến. Các biến là cách lưu trữ dữ liệu trong chương trình máy tính, và chúng có thể giúp bạn viết những chương trình hữu ích.

## Tính Toán với Python

Thông thường, khi được yêu cầu tìm tích của hai số như  $8 \times 3.57$ , bạn sẽ sử dụng máy tính hoặc bút và giấy. Vậy bạn có thể sử dụng Python shell để thực hiện phép tính của mình không? Hãy thử nhé.

Khởi động Python shell bằng cách nhấp đúp vào biểu tượng IDLE trên màn hình desktop của bạn hoặc nếu bạn đang sử dụng Ubuntu, hãy nhấp vào biểu tượng IDLE trong menu Ứng dụng. Tại dấu nhắc lệnh, nhập biểu thức sau:

```
>>> 8 * 3.57
```

```
28.56
```

Hãy để ý rằng khi nhập một phép toán nhân trong Python, bạn sử dụng ký hiệu dấu hoa thị (\*) thay vì dấu nhân (×).

Giả sử bạn đang đào bới sân sau và phát hiện ra một túi đựng 20 đồng tiền vàng. Ngày hôm sau, bạn lén lút xuống tầng hầm và cho những đồng tiền đó vào máy sao chép của ông bạn (rất may là bạn chỉ vừa vặn để chứa 20 đồng tiền). Bạn nghe thấy một tiếng rít và một tiếng bíp, và sau vài giờ, một số đồng tiền khác được bắn ra.

Vậy nếu bạn làm điều này hàng ngày trong suốt một năm, bạn sẽ có bao nhiêu đồng tiền trong kho báu của mình? Trên giấy, các phương trình có thể trông như thế này:

$$10 \times 365 = 3650$$

$$20 + 3650 = 3670$$

Đúng vậy, dễ dàng để thực hiện các phép toán này trên máy tính hoặc trên giấy, nhưng chúng ta cũng có thể thực hiện tất cả các phép toán này bằng Python shell. Đầu tiên, chúng ta nhân 10 đồng tiền với 365 ngày trong một năm để được 3650. Sau đó, chúng ta cộng thêm 20 đồng tiền ban đầu để có tổng cộng 3670 đồng tiền.

```
>>> 10 * 365
```

```
3650
```

```
>>> 20 + 3650
```

```
3670
```

Nếu một con quạ phát hiện ra số vàng lấp lánh trong phòng bạn và mỗi tuần bay vào để ăn cắp ba đồng tiền, thì chúng ta sẽ thực hiện phép toán như sau trong shell Python:

```
>>> 3 * 52
```

```
156
```

```
>>> 3670 - 156
```

```
3514
```

Đầu tiên, chúng ta nhân 3 đồng tiền với 52 tuần trong một năm. Kết quả là 156. Chúng ta trừ con số đó khỏi tổng số tiền (3670), cho thấy rằng chúng ta sẽ có 3514 đồng tiền còn lại vào cuối năm.

Đây là một chương trình rất đơn giản. Trong cuốn sách này, bạn sẽ học cách mở rộng những ý tưởng này để viết các chương trình hữu ích hơn.

Các toán tử cơ bản trong Python cho phép thực hiện các phép toán số học như cộng, trừ, nhân, chia và nhiều phép toán khác như trình bày trong Bảng 2-1.

Bảng 2-1. Các Toán Tử Python Cơ Bản

Biểu tượng	Hoạt động	Ví dụ
+	Cộng	$a + b$
—	Trừ	$a - b$
*	Nhân	$a * b$
/	Chia	$a / b$

Dấu gạch chéo (/) được sử dụng cho phép chia vì nó tương tự như dấu chia mà bạn sử dụng khi viết phân số. Ví dụ, nếu bạn có 100 hải tặc và 20 thùng lớn và muốn tính xem bạn có thể giấu bao nhiêu hải tặc trong mỗi thùng, bạn có thể chia 100 hải tặc cho 20

thùng ( $100 \div 20$ ) bằng cách nhập  $100 / 20$  vào Python shell. Chỉ cần nhớ rằng dấu gạch chéo là dấu mà đỉnh nghiêng về phía phải.



### Thứ tự ưu tiên của các phép toán

Chúng ta sử dụng dấu ngoặc đơn trong ngôn ngữ lập trình để kiểm soát thứ tự thực hiện các phép toán. Một phép toán là bất kỳ thao tác nào sử dụng một toán tử. Phép nhân và phép chia có thứ tự cao hơn phép cộng và phép trừ, có nghĩa là chúng sẽ được thực hiện trước. Nói cách khác, nếu bạn nhập một phương trình trong Python, phép nhân hoặc chia sẽ được thực hiện trước phép cộng hoặc trừ.

Ví dụ, trong phương trình sau, hai số 30 và 20 được nhân trước, sau đó số 5 được cộng vào tích của chúng.

```
>>> 5 + 30 * 20
```

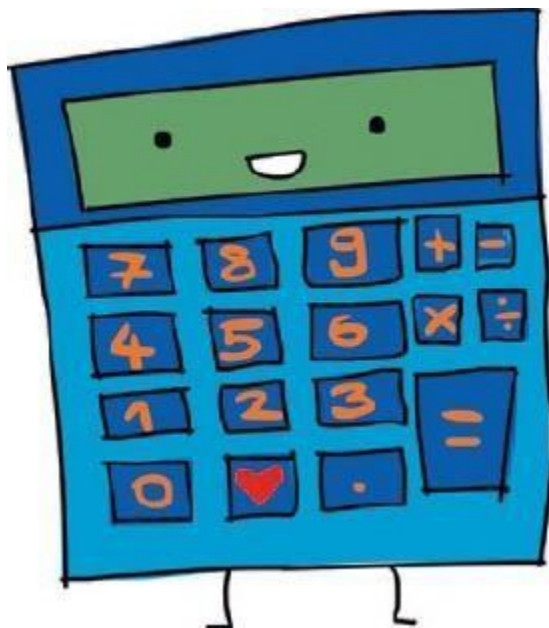
```
605
```

Đây là cách khác để nói, "nhân 30 với 20, và sau đó cộng 5 vào kết quả." Kết quả là 605. Chúng ta có thể thay đổi thứ tự thực hiện bằng cách thêm dấu ngoặc xung quanh hai số đầu tiên, như sau:

```
>>> (5 + 30) * 20
```

700

Kết quả của phương trình này là 700 (không phải 605) vì dấu ngoặc báo cho Python thực hiện phép toán trong dấu ngoặc trước, và sau đó thực hiện phép toán bên ngoài dấu ngoặc. Ví dụ này có nghĩa là "cộng 5 với 30, và sau đó nhân kết quả với 20."



Dấu ngoặc có thể được lồng nhau, nghĩa là có thể có dấu ngoặc bên trong dấu ngoặc, như thế này:

```
>>> ((5 + 30) * 20) / 10
```

70.0

Trong trường hợp này, Python sẽ thực hiện các dấu ngoặc bên trong trước, sau đó là các dấu ngoặc ngoài và cuối cùng là toán tử chia. Nói cách khác, phương trình này đang nói rằng "thêm 5 vào 30, sau đó nhân kết quả với 20 và chia kết quả đó cho 10." Đây là những gì xảy ra:

- Thêm 5 vào 30 sẽ cho kết quả là 35.
- Nhân 35 với 20 sẽ cho kết quả là 700.
- Chia 700 cho 10 sẽ cho kết quả cuối cùng là 70.

Nếu không sử dụng dấu ngoặc, kết quả sẽ hơi khác một chút:

```
>>> 5 + 30 * 20 / 10
```

```
65.0
```

Trong trường hợp này, 30 sẽ được nhân với 20 trước (cho kết quả là 600), sau đó 600 sẽ được chia cho 10 (cho kết quả là 60). Cuối cùng, 5 được thêm vào để có kết quả là 65.

## CẢNH BÁO

Hãy nhớ rằng, phép nhân và phép chia luôn được thực hiện trước phép cộng và phép trừ, trừ khi có dấu ngoặc được sử dụng để kiểm soát thứ tự các phép toán.

## Biến giống như nhãn

Biến trong lập trình mô tả một nơi để lưu trữ thông tin như số, văn bản, danh sách số hoặc văn bản, v.v. Một cách khác để nhìn vào biến là nó giống như một nhãn cho một thứ gì đó.

Ví dụ, để tạo một biến có tên là fred, chúng ta sử dụng dấu bằng (=) và sau đó chỉ định cho Python thông tin mà biến này sẽ làm nhãn cho. Ở đây, chúng ta tạo biến fred và chỉ định rằng nó gán cho số 100 (lưu ý rằng điều này không có nghĩa là một biến khác không thể có cùng giá trị):

```
>>> fred = 100
```

Để biết một biến gán giá trị nào, bạn nhập print trong shell, tiếp theo là tên biến trong dấu ngoặc, như sau:

```
>>> print(fred)
```

```
100
```

Chúng ta cũng có thể yêu cầu Python thay đổi biến fred để gán một giá trị khác. Ví dụ, đây là cách thay đổi fred thành số 200:

```
>>> fred = 200
```

```
>>> print(fred)
```

```
200
```

Ở dòng đầu tiên, chúng ta xác định rằng fred gán cho số 200. Ở dòng thứ hai, chúng ta kiểm tra fred để xác nhận sự thay đổi. Python sẽ in kết quả ở dòng cuối cùng.

Chúng ta cũng có thể sử dụng nhiều nhân (nhiều biến) cho cùng một mục:

```
>>> fred = 200
```

```
>>> john = fred
```

```
>>> print(john)
```

```
200
```

Trong ví dụ này, chúng ta đang yêu cầu Python gán cho tên (hoặc biến) john tương ứng với cùng một giá trị mà fred đã gán.

Dĩ nhiên, fred có thể không phải là một tên biến hữu ích vì nó có thể không cung cấp thông tin hữu ích về mục đích sử dụng của biến. Hãy gọi biến của chúng ta là `number_of_coins` thay vì fred, như sau:

```
>>> number_of_coins = 200
```

```
>>> print(number_of_coins)
```

```
200
```

Điều này giúp rõ ràng rằng chúng ta đang nói về 200 đồng xu. Tên biến có thể bao gồm chữ cái, số và ký tự gạch dưới (`_`), nhưng không thể bắt đầu bằng số.

Bạn có thể sử dụng bất kỳ thứ gì từ một chữ cái (như a) đến các câu dài cho tên biến. (Biến không thể chứa khoảng trắng, vì vậy hãy sử dụng ký tự gạch dưới để phân tách từ.) Thịnh thoảng, nếu bạn đang thực hiện điều gì đó nhanh chóng, một tên biến ngắn là tốt nhất. Tên mà bạn chọn nên phụ thuộc vào mức độ ý nghĩa mà bạn cần tên biến mang lại.

Giờ bạn đã biết cách tạo biến, hãy cùng tìm hiểu cách sử dụng chúng.

## Sử dụng biến

Nhớ lại phương trình của chúng ta để tính số đồng xu bạn sẽ có vào cuối năm nếu bạn có thể tạo ra đồng xu mới một cách kỳ diệu với sáng chế điên rồ của ông bạn trong tầng hầm? Chúng ta có phương trình này:

```
>>> 20 + 10 * 365
```

```
3670
```

```
>>> 3 * 52
```



156

```
>>> 3670 - 156
```

3514

Chúng ta có thể chuyển nó thành một dòng lệnh duy nhất:

```
>>> 20 + 10 * 365 - 3 * 52
```

3514

Bây giờ, nếu chúng ta biến các số thành biến? Hãy thử nhập vào:

```
>>> found_coins = 20
```

```
>>> magic_coins = 10
```

```
>>> stolen_coins = 3
```

Các mục nhập này tạo ra các biến `found_coins`, `magic_coins` và `stolen_coins`. Bây giờ, chúng ta có thể viết lại phương trình như sau:

```
>>> found_coins + magic_coins * 365 - stolen_coins * 52
```

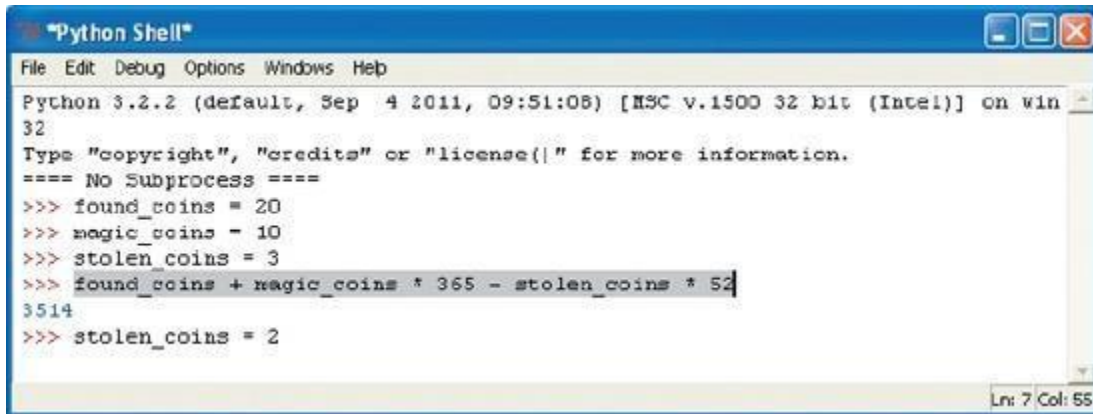
3514

Bạn có thể thấy rằng điều này cho chúng ta cùng một kết quả. Nhưng đây là phép màu của biến. Thử tưởng tượng bạn đặt một con bù nhìn trong cửa sổ, và con quạ chỉ ăn trộm hai đồng xu thay vì ba đồng. Khi sử dụng biến, chúng ta chỉ cần thay đổi biến để giữ số mới, và nó sẽ thay đổi ở mọi nơi nó được sử dụng trong phương trình. Chúng ta có thể thay đổi biến `stolen_coins` thành 2 bằng cách nhập như sau:



```
>>> stolen_coins = 2
```

Chúng ta sau đó có thể sao chép và dán phương trình để tính toán lại kết quả, như sau: Chọn văn bản cần sao chép bằng cách nhấp chuột và kéo từ đầu đến cuối dòng, như được hiển thị ở đây:

A screenshot of a Python Shell window titled "Python Shell". The window has a menu bar with "File", "Edit", "Debug", "Options", "Windows", and "Help". The status bar at the bottom right shows "Ln: 7 Col: 55". The main text area contains the following text:

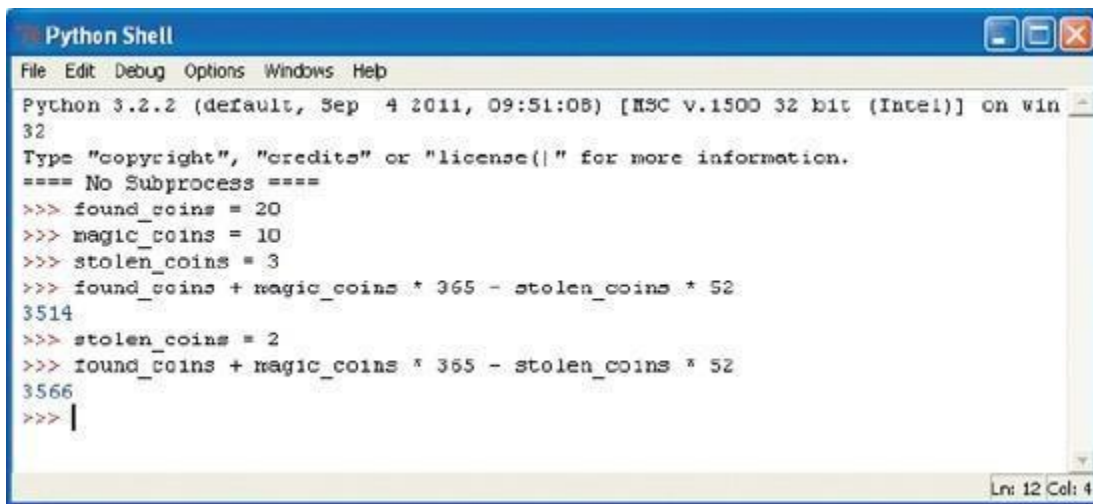
```
Python 3.2.2 (default, Sep 4 2011, 09:51:08) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
==== No Subprocess ====
>>> found_coins = 20
>>> magic_coins = 10
>>> stolen_coins = 3
>>> found_coins + magic_coins * 365 - stolen_coins * 52
3514
>>> stolen_coins = 2
```

Giữ phím Ctrl (hoặc nếu bạn đang sử dụng Mac, giữ phím Cmd) và nhấn C để sao chép văn bản đã chọn. (Bạn sẽ thấy điều này viết là Ctrl-C từ giờ trở đi.)

Nhấn vào dòng lệnh cuối cùng (sau khi `stolen_coins = 2`).

Giữ phím Ctrl và nhấn V để dán văn bản đã chọn. (Bạn sẽ thấy điều này viết là Ctrl-V từ giờ trở đi.)

Nhấn Enter để xem kết quả mới.

A screenshot of a Python Shell window titled "Python Shell". The window has a menu bar with "File", "Edit", "Debug", "Options", "Windows", and "Help". The status bar at the bottom right shows "Ln: 12 Col: 4". The main text area contains the following text:

```
Python 3.2.2 (default, Sep 4 2011, 09:51:08) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
==== No Subprocess ====
>>> found_coins = 20
>>> magic_coins = 10
>>> stolen_coins = 3
>>> found_coins + magic_coins * 365 - stolen_coins * 52
3514
>>> stolen_coins = 2
>>> found_coins + magic_coins * 365 - stolen_coins * 52
3566
>>> |
```

Có phải dễ dàng hơn rất nhiều so với việc gõ lại toàn bộ phương trình không? Đúng vậy.

Bạn có thể thử thay đổi các biến khác, sau đó sao chép (ctrl-C) và dán (ctrl-V) phép tính để thấy được hiệu quả của những thay đổi của mình. Ví dụ, nếu bạn làm cho máy của ông

nội rung lắc đúng lúc, và nó phun ra thêm 3 đồng mỗi lần, bạn sẽ thấy rằng bạn sẽ có 4661 đồng vào cuối năm.

```
>>> magic_coins = 13
```

```
>>> found_coins + magic_coins * 365 - stolen_coins * 52
```

```
4661
```

Chắc chắn, việc sử dụng biến cho một phương trình đơn giản như thế này vẫn còn hữu ích chỉ ở mức độ vừa phải. Chúng ta vẫn chưa đạt đến mức độ thực sự hữu ích.

Tạm thời, hãy nhớ rằng biến là một cách để gán nhãn cho các thứ để bạn có thể sử dụng chúng sau này.

### **Bạn học được gì**

Trong chương này, bạn đã học cách thực hiện các phương trình đơn giản bằng cách sử dụng toán tử Python và cách sử dụng dấu ngoặc đơn để kiểm soát thứ tự thực hiện các phép toán (thứ tự mà Python đánh giá các phần của phương trình). Sau đó, chúng ta đã tạo ra các biến để gán giá trị và sử dụng những biến đó trong các phép tính của mình.



Trong chương 2, chúng ta đã thực hiện một số phép tính cơ bản với Python và bạn đã tìm hiểu về biến. Trong chương này, chúng ta sẽ làm việc với các thành phần khác trong chương trình Python: chuỗi, danh sách, tuple, và map. Bạn sẽ sử dụng chuỗi để hiển thị thông báo trong chương trình của mình (như các thông điệp “Get Ready” và “Game Over” trong trò chơi). Bạn cũng sẽ khám phá cách các danh sách, tuple, và map được sử dụng để lưu trữ tập hợp các thứ.

### Chuỗi

Trong thuật ngữ lập trình, chúng ta thường gọi văn bản là chuỗi. Khi bạn nghĩ về chuỗi như một tập hợp các chữ cái, thuật ngữ này có vẻ hợp lý. Tất cả các chữ cái, số, và ký tự trong cuốn sách này có thể được xem là một chuỗi. Tên của bạn có thể là một chuỗi, và địa chỉ của bạn cũng vậy. Thậm chí, chương trình Python đầu tiên chúng ta tạo trong Chương 1 đã sử dụng một chuỗi: “Hello World.”



### TẠO CHUỖI

Trong Python, chúng ta tạo chuỗi bằng cách đặt dấu ngoặc kép xung quanh văn bản. Ví dụ: chúng ta có thể sử dụng biến fred (vốn không hữu ích lắm) từ Chương 2 và gán cho nó một chuỗi để làm nhãn, như sau:

---

```
fred = "Why do gorillas have big nostrils? Big fingers!!"
```

---

Sau đó, để xem nội dung bên trong fred, chúng ta có thể nhập print(fred), như sau:

---

```
>>> print(fred)
```

```
Why do gorillas have big nostrils? Big fingers!!
```

---

Bạn cũng có thể sử dụng dấu nháy đơn để tạo một chuỗi, như sau:

---

```
>>> fred = 'What is pink and fluffy? Pink fluff!!'
```

```
>>> print(fred)
```

```
What is pink and fluffy? Pink fluff!!
```

---

Tuy nhiên, nếu bạn cố gắng nhập nhiều hơn một dòng văn bản cho chuỗi của mình chỉ bằng một dấu nháy đơn (') hoặc dấu nháy kép ("), hoặc nếu bạn bắt đầu bằng một loại dấu nháy và kết thúc bằng loại khác, bạn sẽ nhận được thông báo lỗi trong Python shell. Ví dụ, hãy nhập dòng sau:

---

```
>>> fred = "How do dinosaurs pay their bills?"
```

---

Bạn sẽ thấy kết quả như sau:

---

```
SyntaxError: EOL while scanning string literal
```

---

Đây là một thông báo lỗi phản nản về cú pháp vì bạn đã không tuân thủ quy tắc kết thúc chuỗi bằng dấu nháy đơn hoặc dấu nháy kép.

Cú pháp (syntax) có nghĩa là cách sắp xếp và thứ tự của từ trong một câu, hoặc trong trường hợp này, là cách sắp xếp và thứ tự của từ và ký hiệu trong chương trình. Vì vậy, `SyntaxError` có nghĩa là bạn đã làm gì đó không đúng theo thứ tự mà Python mong đợi, hoặc bạn đã bỏ sót điều gì đó mà Python đang mong đợi.

Giải pháp: Để sử dụng nhiều dòng trong chuỗi (được gọi là chuỗi nhiều dòng), bạn dùng ba dấu nháy đơn (`"""`) và nhấn Enter giữa các dòng, như sau:

---

```
>>> fred = """How do dinosaurs pay their bills?
With tyrannosaurus checks!"""
```

---

Bây giờ, hãy in nội dung của `fred` để xem liệu điều này có hoạt động không:

---

```
>>> print(fred)
How do dinosaurs pay their bills?
With tyrannosaurus checks!
```

---

## XỬ LÝ VẤN ĐỀ VỚI CHUỖI

Bây giờ, hãy xem xét một ví dụ phức tạp về chuỗi, điều này sẽ khiến Python hiển thị thông báo lỗi:

---

```
>>> silly_string = 'He said, "Aren't can't shouldn't wouldn't."'
SyntaxError: invalid syntax
```

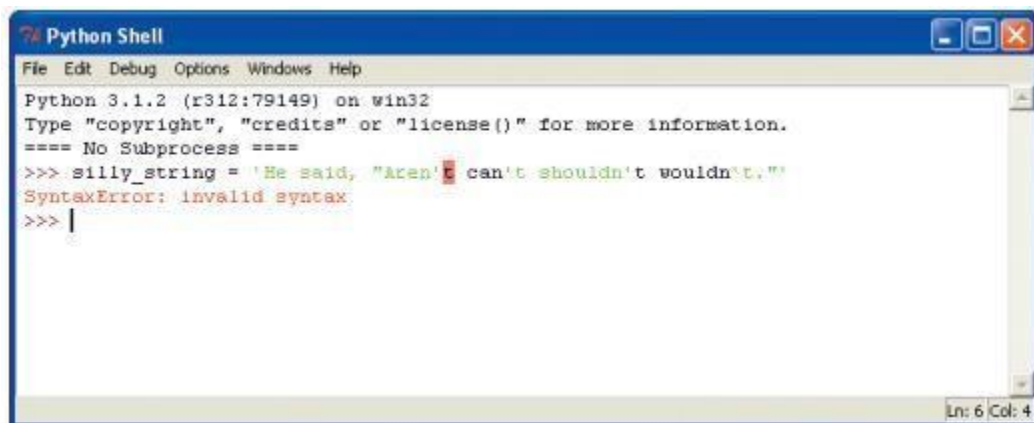
---

Trong dòng đầu tiên, chúng ta cố gắng tạo một chuỗi (được định nghĩa là biến `silly_string`) được bao quanh bởi dấu nháy đơn, nhưng lại chứa sự kết hợp của dấu nháy đơn trong các từ *can't*, *shouldn't*, và *wouldn't*, cùng với dấu nháy kép. Thật hỗn loạn!

Hãy nhớ rằng Python không thông minh như con người, vì vậy tất cả những gì nó thấy là một chuỗi chứa **He said**, **"Aren**, sau đó là một loạt ký tự khác mà nó không mong đợi. Khi Python gặp một dấu nháy (dấu nháy đơn hoặc kép), nó sẽ mong đợi chuỗi bắt đầu ngay sau dấu nháy đầu tiên và kết thúc sau dấu nháy phù hợp tiếp theo (dấu nháy đơn hoặc kép) trên cùng một dòng.

Trong trường hợp này, điểm bắt đầu chuỗi là dấu nháy đơn trước từ **He**, và điểm kết thúc chuỗi (theo Python) là dấu nháy đơn sau chữ **n** trong từ **Aren**.

IDLE sẽ làm nổi bật vị trí mà lỗi xảy ra:



Dòng cuối cùng trong IDLE cho chúng ta biết loại lỗi đã xảy ra—trong trường hợp này là lỗi cú pháp (**syntax error**).

Sử dụng dấu nháy kép thay vì dấu nháy đơn vẫn sẽ dẫn đến lỗi:

---

```
>>> silly_string = "He said, \"Aren't can't shouldn't wouldn't.\""
```

```
SyntaxError: invalid syntax
```

---

Ở đây, Python thấy một chuỗi được giới hạn bởi dấu nháy kép, chứa các ký tự **He said**, (và một khoảng trắng). Mọi thứ sau chuỗi đó (từ **Aren't** trở đi) gây ra lỗi.

```
Python Shell
File Edit Debug Options Windows Help
Python 3.1.2 (r312:79149) on win32
Type "copyright", "credits" or "license()" for more information.
==== No Subprocess ====
>>> silly_string = 'He said, "Aren't can't shouldn't wouldn't."'
SyntaxError: invalid syntax
>>> silly_string = "He said, "Aren't can't shouldn't wouldn't.""
SyntaxError: invalid syntax
>>> |
```

Điều này là vì, từ quan điểm của Python, tất cả những thứ bổ sung đó không nên có ở đó. Python tìm kiếm dấu nháy phù hợp tiếp theo và không biết bạn muốn nó làm gì với bất kỳ thứ gì theo sau dấu nháy đó trên cùng một dòng.

Giải pháp cho vấn đề này là sử dụng chuỗi nhiều dòng, mà chúng ta đã học trước đó, bằng cách dùng ba dấu nháy đơn (""") giúp chúng ta kết hợp cả dấu nháy đơn và dấu nháy kép trong chuỗi mà không gây lỗi. Thực tế, nếu sử dụng ba dấu nháy đơn, chúng ta có thể đặt bất kỳ sự kết hợp nào của dấu nháy đơn và dấu nháy kép vào trong chuỗi (miễn là không cố gắng đặt ba dấu nháy đơn vào trong chuỗi). Đây là phiên bản chuỗi không có lỗi của chúng ta:

```
silly_string = """He said, "Aren't can't shouldn't
wouldn't."""
```



Nhưng khoan đã, còn nhiều điều nữa. Nếu bạn thực sự muốn sử dụng dấu nháy đơn hoặc dấu nháy kép để bao quanh một chuỗi trong Python, thay vì dùng ba dấu nháy đơn, bạn có thể thêm một dấu gạch chéo

ngược (\) trước mỗi dấu nháy trong chuỗi. Đây gọi là escaping. Đây là cách để nói với Python, "Vâng, tôi biết tôi có dấu nháy trong chuỗi của mình, và tôi muốn bạn bỏ qua chúng cho đến khi bạn thấy dấu nháy kết thúc."

Việc sử dụng escaping có thể làm cho chuỗi khó đọc hơn, vì vậy tốt hơn là nên sử dụng chuỗi nhiều dòng. Tuy nhiên, bạn vẫn có thể gặp phải những đoạn mã sử dụng escaping, vì vậy hiểu được lý do tại sao có dấu gạch chéo ngược là điều tốt.



Dưới đây là một vài ví dụ về cách escaping hoạt động:

---

```
(1) >>> single_quote_str = 'He said, "Aren\'t can\'t shouldn\'t wouldn\'t."'
(2) >>> double_quote_str = "He said, \'Aren\'t can\'t shouldn\'t wouldn\'t.\'"

>>> print(single_quote_str)

He said, "Aren't can't shouldn't wouldn't."

>>> print(double_quote_str)

He said, "Aren't can't shouldn't wouldn't."
```

---

Đầu tiên, tại **1**, chúng ta tạo một chuỗi với dấu nháy đơn, sử dụng dấu gạch chéo ngược () trước dấu nháy đơn bên trong chuỗi đó. Tại **2**, chúng ta tạo một chuỗi với dấu nháy kép và sử dụng dấu gạch chéo ngược () trước các dấu nháy kép trong chuỗi. Trong các dòng tiếp theo, chúng ta in các biến mà chúng ta vừa tạo. Lưu ý rằng ký tự gạch chéo ngược không xuất hiện trong các chuỗi khi chúng ta in chúng ra.

## NHÚNG GIÁ TRỊ VÀO CHUỖI

Nếu bạn muốn hiển thị một thông báo sử dụng nội dung của một biến, bạn có thể nhúng các giá trị vào chuỗi bằng cách sử dụng **%s**, đây giống như một dấu hiệu cho một giá trị mà bạn muốn thêm vào sau. (Việc nhúng giá trị là cách nói của lập trình viên để "chèn giá trị vào"). Ví dụ, để Python tính toán hoặc lưu trữ số điểm bạn ghi được trong một trò chơi, và sau đó thêm nó vào một câu như "I scored \_points," bạn sử dụng **%s** trong câu thay cho giá trị, và sau đó thông báo cho Python biết giá trị đó, như sau:

---

```
>>> myscore = 1000

>>> message = 'I scored %s points'

>>> print(message % myscore)

I scored 1000 points
```

---

Ở đây, chúng ta tạo biến `myscore` với giá trị 1000 và biến `message` với một chuỗi chứa các từ "I scored %s points", trong đó %s là một dấu chấm cho số điểm. Ở dòng tiếp theo, chúng ta gọi `print(message)` với ký tự % để thông báo cho Python thay thế %s bằng giá trị được lưu trữ trong biến `myscore`. Kết quả khi in thông điệp này sẽ là "I scored 1000 points". Chúng ta không cần phải sử dụng một biến cho giá trị này. Ta cũng có thể thực hiện ví dụ tương tự và chỉ cần sử dụng `print(message % 1000)`.

Chúng ta cũng có thể truyền vào các giá trị khác cho dấu chấm %s, sử dụng các biến khác nhau, như trong ví dụ sau:

---

```
>>> joke_text = '%s: a device for finding furniture in the dark'
```

```
>>> bodypart1 = 'Knee'
```

```
>>> bodypart2 = 'Shin'
```

```
>>> print(joke_text % bodypart1)
```

```
Knee: a device for finding furniture in the dark
```

```
>>> print(joke_text % bodypart2)
```

```
Shin: a device for finding furniture in the dark
```

---

Ở đây, chúng ta tạo ba biến. Biến đầu tiên, `joke_text`, bao gồm chuỗi với dấu %s. Các biến khác là `bodypart1` và `bodypart2`. Chúng ta có thể in biến `joke_text`, và một lần nữa sử dụng toán tử % để thay thế nó bằng nội dung của các biến `bodypart1` và `bodypart2`, từ đó tạo ra các thông điệp khác nhau.

Bạn cũng có thể sử dụng nhiều dấu chấm (placeholders) trong một chuỗi, như sau:



---

```
>>> nums = 'What did the number %s say to the number %s? Nice belt!!' >>> print(nums % (0, 8))
```

```
What did the number 0 say to the number 8? Nice belt!!
```

---

Khi sử dụng nhiều dấu chấm (placeholders), hãy chắc chắn rằng bạn đóng gói các giá trị

thay thế trong dấu ngoặc đơn, như trong ví dụ. Thứ tự của các giá trị là thứ tự mà chúng sẽ được sử dụng trong chuỗi.

## NHÂN CHUỖI

Cái gì là 10 nhân với 5? Đáp án là 50, tất nhiên rồi. Nhưng 10 nhân với **a** thì sao? Đây là câu trả lời của Python:

---

```
>>> print(10 * 'a')
```

```
aaaaaaaaaa
```

---

Các lập trình viên Python có thể sử dụng phương pháp này để căn chỉnh các chuỗi với một số khoảng trắng cụ thể khi hiển thị thông báo trong shell, ví dụ như vậy. Vậy còn việc in một chữ cái trong shell thì sao? Bạn có thể chọn **File -> New Window** và nhập mã sau:

---

```
spaces = ' ' * 25
```

```
print('%s 12 Butts Wynd' % spaces)
```

```
print('%s Twinklebottom Heath' % spaces)
```

```
print('%s West Snoring' % spaces)
```

```
print()
```

```
print()
```

```
print('Dear Sir')
```

```
print()
```

```
print('I wish to report that tiles are missing from the')
```

```
print('outside toilet roof.')
```

```
print('I think it was bad wind the other night that blew them away.')
```

```
print()
```

```
print('Regards')
```

```
print('Malcolm Dithering')
```

---

Khi bạn đã nhập mã vào cửa sổ shell, chọn **File -> Save As** và đặt tên cho tệp của bạn là myletter.py.

*Từ bây giờ, khi bạn thấy Save As: somefilename.py ở trên một đoạn mã, bạn sẽ biết rằng bạn cần chọn **File -> New Window**, nhập mã vào cửa sổ xuất hiện và sau đó lưu lại như chúng ta đã làm trong ví dụ này.*

Trong dòng đầu tiên của ví dụ này, chúng ta tạo biến spaces bằng cách nhân một ký tự khoảng trắng với 25. Sau đó, chúng ta sử dụng biến đó trong ba dòng tiếp theo để căn chỉnh văn bản sang phía bên phải của shell. Bạn có thể thấy kết quả của các câu lệnh print dưới đây:



```
Python Shell
File Edit Debug Options Windows Help
>>>
12 Butts Wynd
Twinklebottom Heath
West Snoring

Dear Sir

I wish to report that tiles are missing from the
outside toilet roof.
I think it was bad wind the other night that blew
them away.

Regards
Malcolm Dithering
>>>
>>>
>>>
>>>
>>>
Ln: 68 Col: 4
```

Ngoài việc sử dụng phép nhân để căn chỉnh, chúng ta cũng có thể sử dụng nó để làm đầy màn hình với những thông báo gây phiền phức. Hãy thử ví dụ này cho chính bạn:

---

```
>>> print(1000 * 'snirt')
```

---

## DANH SÁCH MẠNH HƠN CHUỖI

“Spider legs, toe of frog, eye of newt, bat wing, slug butter, and snake dandruff” không phải là một danh sách mua sắm bình thường (trừ khi bạn là một phù thủy), nhưng chúng ta sẽ sử dụng nó như là ví dụ đầu tiên về sự khác biệt giữa chuỗi và danh sách.

Chúng ta có thể lưu trữ danh sách các món đồ này trong biến `wizard_list` bằng cách sử dụng một chuỗi như sau:



```
>>> wizard_list = 'spider legs, toe of frog, eye of newt, bat wing, slug butter, snake dandruff'
```

```
>>> print(wizard_list)
```

```
spider legs, toe of frog, eye of newt, bat wing, slug butter, snake dandruff
```

---

Tuy nhiên, chúng ta cũng có thể tạo một danh sách, một loại đối tượng Python "ma thuật" mà chúng ta có thể thao tác. Đây là cách các món đồ này sẽ trông như thế nào khi được viết dưới dạng danh sách:

```
>>> wizard_list = ['spider legs', 'toe of frog', 'eye of newt', 'bat wing', 'slug butter', 'snake dandruff']
```

```
>>> print(wizard_list)
```

```
['spider legs', 'toe of frog', 'eye of newt', 'bat wing', 'slug butter', 'snake dandruff']
```

---

Việc tạo một danh sách mất nhiều công sức hơn so với việc tạo một chuỗi, nhưng danh sách lại hữu ích hơn vì chúng ta có thể thao tác với nó. Ví dụ, chúng ta có thể in món đồ thứ ba trong `wizard_list` (eye of newt) bằng cách nhập vị trí của nó trong danh sách (gọi là chỉ số vị trí) trong dấu ngoặc vuông (`[]`), như sau:

---

```
>>> print(wizard_list[2])
```

```
eye of newt
```

---

Chà? Nó không phải là món thứ ba trong danh sách sao? Đúng vậy, nhưng danh sách bắt đầu từ chỉ số vị trí 0, vì vậy món thứ nhất có chỉ số 0, món thứ hai có chỉ số 1, và món thứ ba có chỉ số 2. Điều này có thể không hợp lý với con người, nhưng lại hợp lý với máy tính.

Chúng ta cũng có thể dễ dàng thay đổi một món đồ trong danh sách hơn là trong một chuỗi. Có thể thay vì "eye of newt", chúng ta cần "snail tongue". Đây là cách chúng ta sẽ làm điều đó với danh sách:

---

```
>>> wizard_list[2] = 'snail tongue'
```

```
>>> print(wizard_list)
```

```
['spider legs', 'toe of frog', 'snail tongue', 'bat wing', 'slug butter', 'snake dandruff']
```

---

Điều này thay thế món đồ ở vị trí chỉ số 2 (trước đây là "eye of newt") thành "snail tongue". Một tùy chọn khác là hiển thị một phần của các mục trong danh sách. Chúng ta làm điều này bằng cách sử dụng dấu hai chấm (:) trong dấu ngoặc vuông. Ví dụ, nhập đoạn mã sau để xem các món đồ từ thứ ba đến thứ năm trong danh sách (một bộ nguyên liệu tuyệt vời cho một chiếc sandwich tuyệt hảo):



```
>>> print(wizard_list[2:5])
```

```
['snail tongue', 'bat wing', 'slug butter']
```

---

Viết [2:5] tương đương với việc nói, "Hiển thị các món đồ từ vị trí chỉ số 2 đến vị trí chỉ số 5 (nhưng không bao gồm vị trí 5)"—nói cách khác, là các món đồ ở vị trí chỉ số 2, 3, và 4.

Danh sách có thể được sử dụng để lưu trữ đủ loại mục, chẳng hạn như các số:

---

```
>>> some_numbers = [1, 2, 5, 10, 20]
```

---

Chúng cũng có thể chứa các chuỗi:

---

```
>>> some_strings = ['Which', 'Witch', 'Is', 'Which']
```

---

Danh sách có thể chứa sự pha trộn giữa số và chuỗi:

---

```
>>> numbers_and_strings = ['Why', 'was', 6, 'afraid', 'of', 7, 'because', 7, 8, 9]
```

```
>>> print(numbers_and_strings)
```

```
['Why', 'was', 6, 'afraid', 'of', 7, 'because', 7, 8, 9]
```

---

Danh sách thậm chí có thể lưu trữ các danh sách khác:

---

```
>>> numbers = [1, 2, 3, 4]
```

```
>>> strings = ['I', 'kicked', 'my', 'toe', 'and', 'it', 'is', 'sore']
```

```
>>> mylist = [numbers, strings]
```

```
>>> print(mylist)
```

```
[[1, 2, 3, 4], ['I', 'kicked', 'my', 'toe', 'and', 'it', 'is', 'sore']]
```

---

Ví dụ về danh sách chứa danh sách này tạo ra ba biến: numbers với bốn số, strings với tám chuỗi, và mylist chứa numbers và strings. Danh sách thứ ba (mylist) chỉ có hai phần tử vì đó là một danh sách các tên biến, không phải là nội dung của các biến đó.

## THÊM ITEMS VÀO DANH SÁCH

Để thêm các mục vào danh sách, ta sử dụng hàm append. Hàm là một đoạn mã yêu cầu Python thực hiện một tác vụ nào đó. Trong trường hợp này, append thêm một mục vào cuối danh sách.

Ví dụ, để thêm một "bear burp" (một món chắc chắn tồn tại) vào danh sách mua sắm của phù thủy, ta làm như sau:

---

```
>>> wizard_list.append('bear burp')
>>> print(wizard_list)
['spider legs', 'toe of frog', 'snail tongue', 'bat wing', 'slug butter', 'snake dandruff', 'bear burp']
```

---

Bạn có thể tiếp tục thêm các món đồ kỳ diệu vào danh sách của phù thủy bằng cách sử dụng append:

---

```
>>> wizard_list.append('mandrake')
>>> wizard_list.append('hemlock')
>>> wizard_list.append('swamp gas')
```

---

Kết quả danh sách bây giờ sẽ là:

---

```
>>> print(wizard_list)
```



['spider legs', 'toe of frog', 'snail tongue', 'bat wing', 'slug butter', 'snake dandruff', 'bear burp', 'mandrake', 'hemlock', 'swamp gas']

---

## XOÁ ITEMS KHỎI DANH SÁCH

Để xóa một mục khỏi danh sách, ta sử dụng lệnh `del` (viết tắt của "delete"). Ví dụ, để xóa món đồ thứ sáu trong danh sách của phù thủy, là "snake dandruff", ta làm như sau:

---

```
>>>del wizard_list[5]
```

```
>>>print(wizard_list)
```

```
['spider legs', 'toe of frog', 'snail tongue', 'bat wing', 'slug butter', 'bear burp', 'mandrake',  
'hemlock', 'swamp gas']
```

---

*Lưu ý rằng chỉ số bắt đầu từ 0, vì vậy `wizard_list[5]` thực sự ám chỉ món đồ thứ sáu trong danh sách.*

Và đây là cách để loại bỏ các mục mà chúng ta vừa thêm vào (mandrake, hemlock và swamp gas):

---

```
>>>del wizard_list[8]
```

```
>>>del wizard_list[7]
```

```
>>>del wizard_list[6]
```

```
>>>print(wizard_list)
```

```
['spider legs', 'toe of frog', 'snail tongue', 'bat wing', 'slug butter', 'bear  
burp']
```

---

## DANH SÁCH TOÁN HỌC

Chúng ta có thể nối các danh sách lại với nhau bằng cách sử dụng dấu cộng (+), giống như cộng các số, ví dụ:

---

```
>>> list1 = [1, 2, 3, 4]
>>> list2 = ['I', 'tripped', 'over', 'and', 'hit', 'the', 'floor']
>>> print(list1 + list2)
[1, 2, 3, 4, 'I', 'tripped', 'over', 'and', 'hit', 'the', 'floor']
```

---

Chúng ta cũng có thể cộng hai danh sách và gán kết quả vào một biến khác:

---

```
>>> list1 = [1, 2, 3, 4]
>>> list2 = ['I', 'ate', 'chocolate', 'and', 'I', 'want', 'more']
>>> list3 = list1 + list2
>>> print(list3)
[1, 2, 3, 4, 'I', 'ate', 'chocolate', 'and', 'I', 'want', 'more']
```

---

Chúng ta cũng có thể nhân một danh sách với một số. Ví dụ, để nhân list1 với 5, ta viết list1 \* 5:

---

```
>>> list1 = [1, 2]
>>> print(list1 * 5)
[1, 2, 1, 2, 1, 2, 1, 2, 1, 2]
```

---

Điều này thực sự đang yêu cầu Python lặp lại list1 năm lần, kết quả là 1, 2, 1, 2, 1, 2, 1, 2, 1, 2.

Mặt khác, phép chia (/) và phép trừ (-) chỉ đưa ra lỗi, như trong các ví dụ sau:

---

```

>>> list1 / 20
Traceback (most recent call last):
  File "", line 1, in list1 /
TypeError: unsupported operand type(s) for /: 'list' and 'int'

>>> list1 - 20
Traceback (most recent call last):
  File "", line 1, in list1 -
TypeError: unsupported operand type(s) for -: 'list' and 'int'

```

---

Nhưng tại sao lại như vậy? Việc nối các danh sách với dấu cộng (+) và lặp lại các danh sách với dấu sao (\*) là những thao tác khá đơn giản. Những khái niệm này cũng khá hợp lý trong thực tế. Ví dụ, nếu tôi đưa cho bạn hai danh sách mua sắm giấy và nói, "Hãy cộng hai danh sách này lại," bạn có thể viết tất cả các mục trong một tờ giấy khác theo thứ tự, nối chúng lại với nhau. Điều tương tự cũng xảy ra nếu tôi nói, "Nhân danh sách này với 3." Bạn có thể tưởng tượng việc viết ra một danh sách của tất cả các mục ba lần trên một tờ giấy khác.

Nhưng làm sao bạn có thể chia một danh sách? Ví dụ, hãy xem xét cách bạn chia một danh sách gồm sáu số (từ 1 đến 6) thành hai phần. Dưới đây là ba cách khác nhau:

---

```

[1, 2, 3]      [4, 5, 6]
[1]            [2, 3, 4, 5, 6]
[1, 2, 3, 4]   [5, 6]

```

---

Liệu chúng ta sẽ chia danh sách ở giữa, chia sau phần tử đầu tiên, hay chỉ chọn một vị trí ngẫu nhiên và chia nó ở đó? Không có câu trả lời đơn giản, và khi bạn yêu cầu Python chia một danh sách, nó cũng không biết phải làm gì. Đó là lý do tại sao Python phản hồi với một lỗi.

Điều tương tự cũng xảy ra khi bạn cố gắng thêm bất cứ thứ gì khác ngoài một danh sách vào một danh sách. Bạn không thể làm điều đó. Ví dụ, đây là điều xảy ra khi chúng ta thử thêm số 50 vào list1:



---

```
>>> list1 + 50
Traceback (most recent call last):
  File "in", line 1, in
    list1 + 50
TypeError: can only concatenate list (not "int") to list
```

---

Tại sao lại có lỗi ở đây? Vậy thêm 50 vào một danh sách có nghĩa là gì? Có phải là thêm 50 vào mỗi phần tử? Nhưng nếu các phần tử không phải là số thì sao? Có phải là thêm số 50 vào cuối hoặc đầu danh sách không?

Trong lập trình máy tính, các lệnh phải hoạt động một cách nhất quán mỗi khi bạn nhập chúng. Máy tính, vì là một cỗ máy đơn giản, chỉ nhìn nhận mọi thứ dưới dạng trắng đen. Yêu cầu nó đưa ra một quyết định phức tạp, và nó sẽ không thể hiểu được, dẫn đến lỗi.

## TUPLES

Một tuple giống như một danh sách nhưng sử dụng dấu ngoặc đơn, như trong ví dụ sau:

---

```
>>> fibs = (0, 1, 1, 2, 3)
```

```
>>> print(fibs[3])
```

```
2
```

---

Ở đây, chúng ta định nghĩa biến `fibs` là các số 0, 1, 1, 2 và 3. Sau đó, giống như một danh sách, chúng ta in ra phần tử tại vị trí chỉ mục 3 trong tuple bằng cách sử dụng `print(fibs[3])`.

Điểm khác biệt chính giữa tuple và danh sách là một tuple không thể thay đổi sau khi bạn đã tạo ra nó. Ví dụ, nếu chúng ta cố gắng thay thế giá trị đầu tiên trong tuple `fibs` bằng số 4 (giống như khi thay đổi giá trị trong `wizard_list`), chúng ta sẽ nhận được một thông báo lỗi:

---

```
>>> fibs[0] = 4
```

Traceback (most recent call last):

File "", line 1, in

```
fibs[0] = 4
```

TypeError: 'tuple' object does not support item assignment

---

Bạn sẽ sử dụng một tuple thay vì một danh sách khi bạn cần một cấu trúc dữ liệu không thay đổi. Nếu bạn tạo một tuple với hai phần tử bên trong, nó sẽ luôn giữ nguyên hai phần tử đó, không thay đổi. Điều này đặc biệt hữu ích khi bạn muốn đảm bảo rằng dữ liệu của mình không bị thay đổi trong suốt quá trình sử dụng.

## CÁC BẢN ĐỒ TRONG PYTHON SẼ KHÔNG GIÚP BẠN TÌM ĐƯỜNG

Trong Python, một map (hay còn gọi là dict, viết tắt của từ dictionary) là một tập hợp các đối tượng, giống như danh sách và tuple. Sự khác biệt giữa maps và danh sách hoặc tuple là mỗi mục trong map có một key (khóa) và một value (giá trị) tương ứng.

Ví dụ, giả sử chúng ta có một danh sách những người và môn thể thao yêu thích của họ. Chúng ta có thể lưu trữ thông tin này trong một danh sách Python, với tên của người đó và môn thể thao yêu thích của họ, như sau:

---

```
favorite_sports = ['Ralph Williams, Football',  
                  'Michael Tippett, Basketball',  
                  'Edward Elgar, Baseball',
```

```
'Rebecca Clarke, Netball',  
'Ethel Smyth,  
Badminton', 'Frank  
Bridge, Rugby']
```

---

Nếu tôi hỏi bạn môn thể thao yêu thích của Rebecca Clarke, bạn có thể lướt qua danh sách và tìm thấy câu trả lời là netball. Nhưng nếu danh sách chứa 100 người (hoặc nhiều hơn), việc tìm kiếm sẽ trở nên khó khăn và mất thời gian.

Bây giờ, nếu chúng ta lưu trữ cùng thông tin này trong một map, với tên người làm khóa và môn thể thao yêu thích làm giá trị, mã Python sẽ trông như thế này:



```
>>> favorite_sports = {'Ralph Williams': 'Football',  
                        'Michael Tippett': 'Basketball',  
                        'Edward Elgar': 'Baseball',  
                        'Rebecca Clarke': 'Netball',  
                        'Ethel Smyth': 'Badminton',  
                        'Frank Bridge': 'Rugby'}
```

---

Ở đây, chúng ta sử dụng dấu hai chấm (:) để tách mỗi khóa khỏi giá trị của nó, và mỗi khóa và giá trị được bao quanh bởi dấu nháy đơn. Lưu ý rằng các mục trong map được bao quanh bởi dấu ngoặc nhọn {}, thay vì dấu ngoặc đơn hoặc ngoặc vuông.

Kết quả là một bản đồ (mỗi khóa trỏ đến một giá trị cụ thể), như trong bảng dưới đây:

Bảng 3-1: Các khóa trỏ đến các giá trị trong bản đồ các môn thể thao yêu thích

Key	Value
Ralph Williams	Football
Michael Tippett	Basketball
Edward Elgar	Baseball
Rebecca Clarke	Netball
Ethel Smyth	Badminton
Frank Bridge	Rugby

Bây giờ, để lấy thông tin môn thể thao yêu thích của Rebecca Clarke, chúng ta truy cập vào bản đồ `favorite_sports` bằng tên của cô ấy làm khóa, như sau:

---

```
>>> print(favorite_sports['Rebecca Clarke'])
```

```
Netball
```

---

Và kết quả là môn thể thao yêu thích của cô ấy là "Netball".

Để xóa một giá trị trong bản đồ, ta sử dụng khóa của nó. Ví dụ, đây là cách xóa Ethel Smyth:

---

```
>>> del favorite_sports['Ethel Smyth']
```

```
>>> print(favorite_sports)
```

```
{'Rebecca Clarke': 'Netball', 'Michael Tippett': 'Basketball', 'Ralph Williams': 'Football',  
'Edward Elgar': 'Baseball', 'Frank Bridge': 'Rugby'}
```

---

Để thay thế một giá trị trong bản đồ, ta cũng sử dụng khóa của nó:

---

```
>>> favorite_sports['Ralph Williams'] = 'Ice Hockey'
```

```
>>> print(favorite_sports)
```

```
{'Rebecca Clarke': 'Netball', 'Michael Tippett': 'Basketball', 'Ralph Williams': 'Ice Hockey',  
'Edward Elgar': 'Baseball', 'Frank Bridge': 'Rugby'}
```

---

Chúng ta thay thế môn thể thao yêu thích "Football" bằng "Ice Hockey" bằng cách sử dụng khóa "Ralph Williams".

Như bạn có thể thấy, làm việc với bản đồ (maps) khá giống với làm việc với danh sách (lists) và bộ dữ liệu (tuples), ngoại trừ việc bạn không thể kết hợp các bản đồ với toán tử cộng (+). Nếu bạn thử làm vậy, bạn sẽ nhận được thông báo lỗi:

---

```
>>> favorite_sports = {'Rebecca Clarke': 'Netball',  
                        'Michael Tippett': 'Basketball',  
                        'Ralph Williams': 'Ice Hockey',  
                        'Edward Elgar': 'Baseball',  
                        'Frank Bridge': 'Rugby'}  
  
>>> favorite_colors = {'Malcolm Warner' : 'Pink polka dots',  
                        'James Baxter' : 'Orange stripes',  
                        'Sue Lee' : 'Purple paisley'}  
  
>>> favorite_sports + favorite_colors
```

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: unsupported operand type(s) for +: 'dict' and 'dict'

---

Việc kết hợp các bản đồ không hợp lý với Python, vì vậy nó sẽ ném ra lỗi.

**NHỮNG GÌ BẠN ĐÃ HỌC ĐƯỢC**



Trong chương này, bạn đã học cách Python sử dụng chuỗi (strings) để lưu trữ văn bản, và cách sử dụng danh sách (lists) và bộ dữ liệu (tuples) để xử lý nhiều mục. Bạn đã thấy rằng các mục trong danh sách có thể thay đổi, và bạn có thể kết hợp một danh sách với danh sách khác, nhưng các giá trị trong bộ dữ liệu không thể thay đổi. Bạn cũng đã học cách sử dụng bản đồ (maps) để lưu trữ các giá trị với các khóa (keys) xác định chúng.

## CÂU ĐÓ LẬP TRÌNH

Dưới đây là một số thí nghiệm bạn có thể thử. Các câu trả lời có thể được tìm thấy tại <http://python-for-kids.com/>.

**#1:**                      **Những**                      **sở**                      **thích**                      **yêu**

**thích** Hãy tạo một danh sách về sở thích yêu thích của bạn và gán cho biến tên games. Sau đó, hãy tạo một danh sách về những món ăn yêu thích của bạn và gán cho biến tên foods. Kết hợp hai danh sách này và đặt kết quả vào biến favorites. Cuối cùng, hãy in biến favorites.

**#2:**                      **Đếm**                      **số**                      **người**                      **tham**

**chiến** Nếu có 3 tòa nhà với 25 ninja ẩn nấp trên mỗi mái nhà và 2 đường hầm với 40 samurai ẩn nấp trong mỗi đường hầm, thì tổng số ninja và samurai chuẩn bị chiến đấu là bao nhiêu? (Bạn có thể làm điều này với một phương trình trong Python shell.)

**#3:**    **Lời**    **chào!**

Tạo hai biến: một biến chứa tên của bạn và một biến chứa họ của bạn. Sau đó, tạo một chuỗi và sử dụng các dấu chèn (placeholders) để in tên của bạn kèm theo một thông điệp, ví dụ như “Chào bạn, Brando Ickett!”

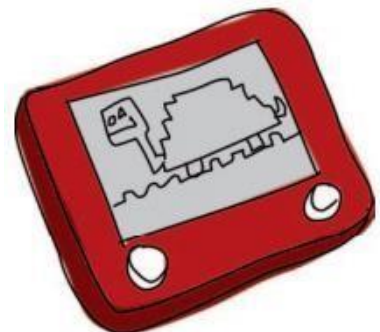


Trong Python, con *turtle* giống như một con rùa trong thế giới thực. Chúng ta biết một con rùa là loài bò sát di chuyển rất chậm và mang trên lưng ngôi nhà của mình. Trong thế giới Python, một con rùa là một mũi tên màu đen di chuyển chậm trên màn hình. Thực ra, vì con rùa Python để lại dấu vết khi di chuyển trên màn hình, nó thực tế giống một con ốc sên hoặc một con sên hơn là một con rùa.

Turtle là một công cụ tuyệt vời để học một số kiến thức cơ bản về đồ họa máy tính. Vì vậy, trong chương này, chúng ta sẽ sử dụng một Python turtle để vẽ một số hình dạng và đường đơn giản.

## SỬ DỤNG MÔ-ĐUN TURTLE TRONG PYTHON

Một mô-đun trong Python là một cách cung cấp mã nguồn hữu ích để được sử dụng bởi chương trình khác (mô-đun có thể chứa các hàm mà chúng ta có thể sử dụng). Chúng ta sẽ tìm hiểu thêm về mô-đun trong Chương 7. Python có một mô-đun đặc biệt gọi là **turtle** mà chúng ta có thể sử dụng để học cách máy tính vẽ hình ảnh trên màn hình. Mô-đun turtle là



một cách lập trình đồ họa vector, cơ bản là vẽ bằng các đường thẳng, chấm và đường cong đơn giản.

Hãy xem cách mô-đun turtle hoạt động. Đầu tiên, mở Python shell bằng cách nhấp vào biểu tượng trên màn hình (hoặc nếu bạn đang sử dụng Ubuntu, chọn **Applications** → **Programming** → **IDLE**). Tiếp theo, yêu cầu Python sử dụng mô-đun turtle bằng cách nhập lệnh sau:

---

```
>>> import turtle
```

---

Việc nhập lệnh này thông báo cho Python rằng bạn muốn sử dụng mô-đun turtle.

*Lưu ý: Nếu bạn đang sử dụng Ubuntu và gặp lỗi ở bước này, bạn có thể cần cài đặt tkinter. Để làm điều đó, hãy mở Ubuntu Software Center và nhập python-tk vào ô tìm kiếm. “Tkinter – Writing Tk Applications with Python” sẽ xuất hiện trong cửa sổ. Hãy nhấp Install để cài đặt gói này.*

## TẠO MỘT CANVAS

Bây giờ, sau khi đã nhập mô-đun turtle, chúng ta cần tạo một canvas—một không gian trắng để vẽ, giống như một bức tranh của họa sĩ. Để làm điều này, chúng ta gọi hàm Pen từ mô-đun turtle, hàm này tự động tạo ra một canvas. Nhập lệnh này vào Python shell:

---

```
>>> t = turtle.Pen()
```

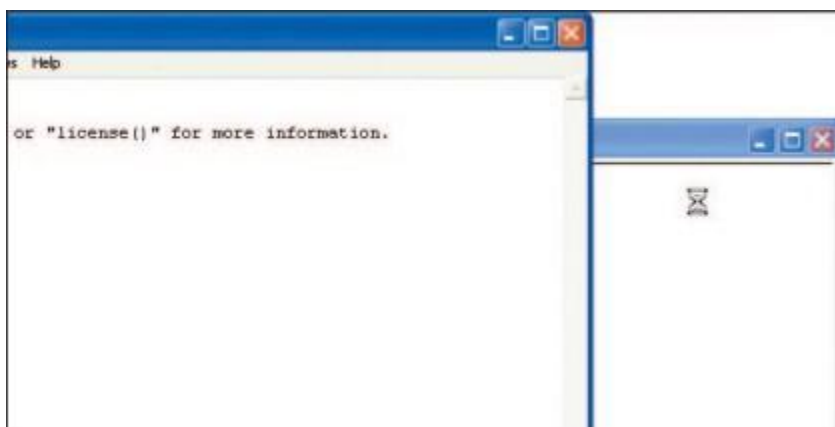
---

Bạn sẽ thấy một hộp trống (canvas), với một mũi tên ở giữa, trông như thế này:



Mũi tên ở giữa màn hình chính là con turtle, và đúng là nó không giống như một con rùa chút nào.

Nếu cửa sổ Turtle xuất hiện phía sau cửa sổ Python Shell, bạn có thể thấy rằng nó không hoạt động đúng cách. Khi di chuyển chuột vào cửa sổ Turtle, con trỏ sẽ chuyển thành hình đồng hồ cát, như thế này:



Điều này có thể xảy ra vì một số lý do: bạn chưa mở shell từ biểu tượng trên màn hình (nếu bạn đang sử dụng Windows hoặc Mac), bạn đã nhấp vào IDLE (Python GUI) trong menu Start của Windows, hoặc IDLE không được cài đặt đúng cách. Hãy thử thoát và khởi động lại shell từ biểu tượng trên màn hình. Nếu điều đó không thành công, hãy thử sử dụng Python console thay vì shell, như sau:

- Trong Windows, chọn **Start** → **All Programs**, sau đó trong nhóm **Python 3.2**, nhấp vào **Python (command line)**.
- Trong Mac OS X, nhấp vào biểu tượng **Spotlight** ở góc trên bên phải màn hình và nhập **Terminal** vào ô tìm kiếm. Sau đó nhập **python** khi terminal mở ra.
- Trong Ubuntu, mở terminal từ menu **Applications** và nhập **python**.

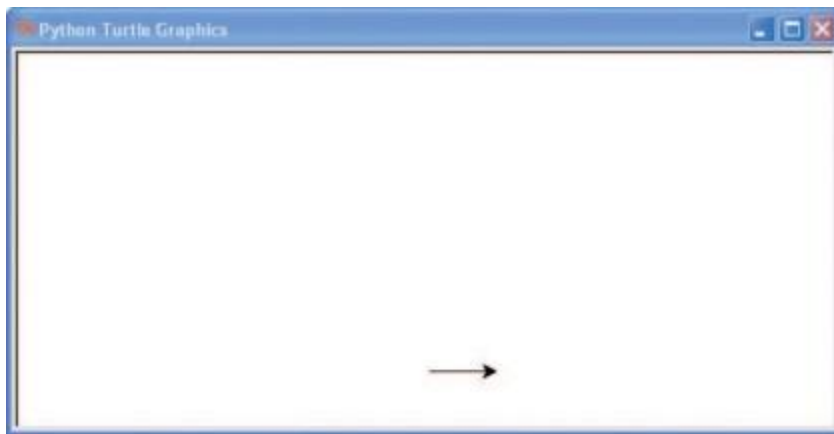
## DI CHUYỂN TURTLE

Bạn gửi các lệnh cho turtle bằng cách sử dụng các hàm có sẵn trên biến t mà chúng ta vừa tạo, tương tự như việc sử dụng hàm Pen trong mô-đun turtle. Ví dụ, lệnh forward yêu cầu turtle di chuyển về phía trước. Để yêu cầu turtle tiến 50 pixel, hãy nhập lệnh sau:

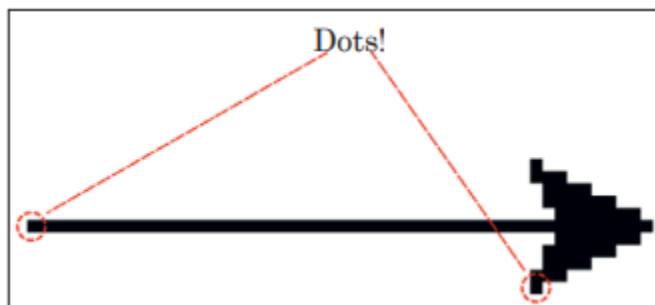


```
>>> t.forward(50)
```

Bạn sẽ thấy một chuyển động như sau:



Con turtle đã di chuyển về phía trước 50 pixel. Một pixel là một điểm duy nhất trên màn hình - yếu tố nhỏ nhất có thể được biểu diễn. Mọi thứ bạn nhìn thấy trên màn hình máy tính của mình đều được tạo thành từ các pixel, những chấm nhỏ hình vuông. Nếu bạn có thể phóng to lên canvas và đường đi mà turtle vẽ ra, bạn sẽ thấy mũi tên đại diện cho con đường của turtle chỉ là một loạt các pixel. Đó chính là đồ họa máy tính đơn giản.



Bây giờ, chúng ta sẽ yêu cầu turtle quay sang trái 90 độ với lệnh sau:

---

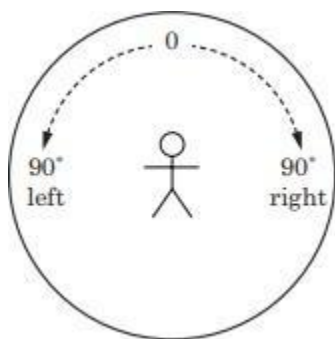
```
>>> t.left(90)
```

---

Nếu bạn chưa học về độ (degree), đây là cách bạn có thể hiểu về chúng. Hãy tưởng tượng bạn đang đứng ở trung tâm của một vòng tròn:

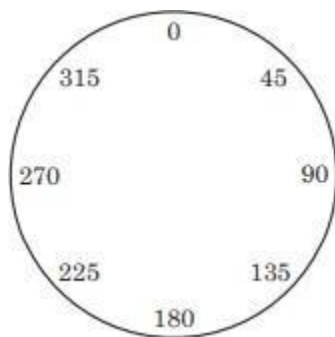
- Hướng bạn đang đối diện là 0 độ.
- Nếu bạn giơ tay trái ra, đó là 90 độ về bên trái.
- Nếu bạn giơ tay phải ra, đó là 90 độ về bên phải.

Bạn có thể thấy sự quay 90 độ sang trái hoặc phải ở



đây:

Nếu bạn tiếp tục di chuyển quanh vòng tròn theo hướng bên phải từ nơi tay phải của bạn đang chỉ, 180 độ sẽ là phía sau bạn, 270 độ sẽ là hướng mà tay trái của bạn đang chỉ, và 360 độ sẽ là vị trí bạn bắt đầu; các độ từ 0 đến 360. Các độ trong một vòng tròn đầy đủ khi quay sang phải có thể thấy như sau, mỗi lần quay là 45 độ:



Khi turtle của Python quay sang trái, nó xoay để hướng về phía mới (giống như bạn xoay cơ thể để đối diện với hướng mà tay bạn đang chỉ 90 độ về phía trái).

Lệnh `t.left(90)` sẽ làm mũi tên quay lên (vì nó ban đầu hướng sang phải):



*Lưu ý: Khi bạn gọi lệnh `t.left(90)`, điều đó tương đương với việc gọi `t.right(270)`. Điều này cũng đúng với lệnh `t.right(90)`, tương đương với việc gọi `t.left(270)`. Hãy tưởng tượng vòng tròn và làm theo các độ.*

Bây giờ, chúng ta sẽ vẽ một hình vuông. Thêm mã sau vào các dòng mã bạn đã nhập trước đó:

---

```
>>> t.forward(50)
```

```
>>> t.left(90)
```

```
>>> t.forward(50)
```

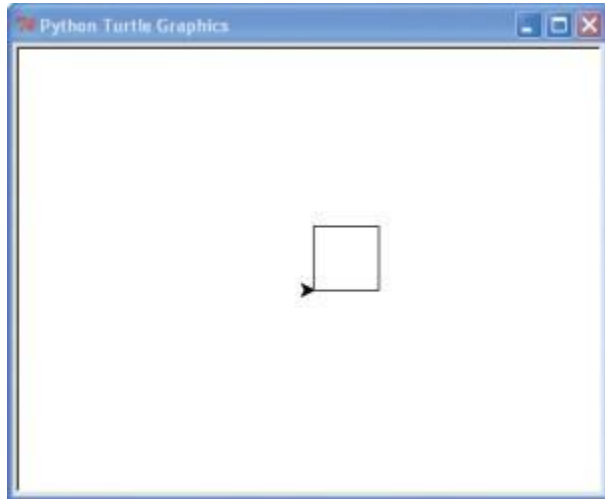
```
>>> t.left(90)
```

```
>>> t.forward(50)
```

```
>>> t.left(90)
```

---

Turtle của bạn nên đã vẽ một hình vuông và hiện tại nó đang hướng theo cùng một hướng mà nó bắt đầu.



Để xóa canvas, bạn nhập lệnh reset. Lệnh này sẽ xóa canvas và đưa turtle trở lại vị trí ban đầu.

---

```
>>> t.reset()
```

---

Bạn cũng có thể sử dụng lệnh clear, lệnh này chỉ xóa màn hình mà không làm thay đổi vị trí của turtle.

---

```
>>> t.clear()
```

---

Chúng ta cũng có thể xoay turtle sang phải hoặc di chuyển nó lùi lại. Lệnh up giúp nâng bút lên khỏi trang (nói cách khác, yêu cầu turtle ngừng vẽ), còn lệnh down giúp bắt đầu vẽ lại. Các lệnh này được viết tương tự như các lệnh khác mà chúng ta đã sử dụng.

Hãy thử vẽ thêm một hình bằng một số lệnh này. Lần này, chúng ta sẽ yêu cầu turtle vẽ hai đường thẳng. Nhập mã sau:

---

```
>>> t.reset()
```

```
>>> t.backward(100)
```



```
>>> t.up()

>>> t.right(90)

>>> t.forward(20)

>>> t.left(90)

>>> t.down()

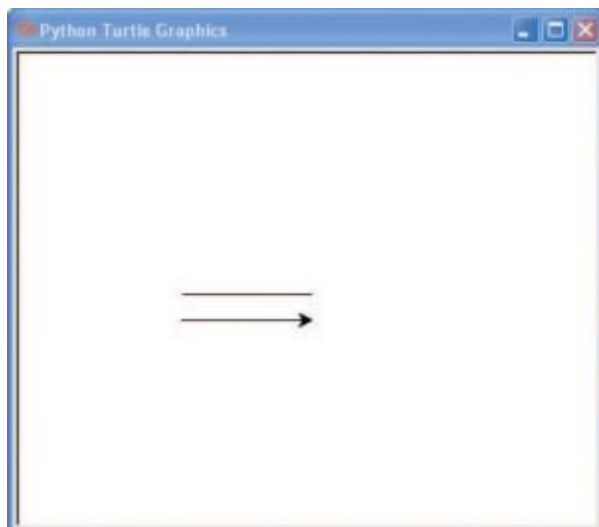
>>> t.forward(100)
```

---

Đầu tiên, chúng ta gọi `t.reset()` để làm mới canvas và di chuyển turtle về lại vị trí bắt đầu. Tiếp theo, chúng ta di chuyển turtle lùi lại 100 pixel với lệnh `t.backward(100)`, và sau đó sử dụng `t.up()` để nhấc bút lên và ngừng vẽ.

Sau đó, với lệnh `t.right(90)`, chúng ta xoay turtle sang phải 90 độ để nó hướng xuống, về phía dưới màn hình, và với lệnh `t.forward(20)`, chúng ta di chuyển về phía trước 20 pixel. Không có gì được vẽ ra vì chúng ta đã dùng lệnh `up` ở dòng thứ ba. Chúng ta xoay turtle sang trái 90 độ với lệnh `t.left(90)` để nó hướng sang phải, rồi sau đó với lệnh `down`, chúng ta yêu cầu turtle đặt bút xuống và bắt đầu vẽ lại. Cuối cùng,

chúng ta vẽ một đường thẳng về phía trước, song song với đường thẳng đầu tiên đã vẽ, với lệnh `t.forward(100)`. Hai đường thẳng song song mà chúng ta vẽ sẽ trông như sau:



## Điều bạn đã học được

Trong chương này, bạn đã học cách sử dụng mô-đun turtle của Python. Chúng ta đã vẽ một số đường thẳng đơn giản, sử dụng các lệnh xoay trái và phải cùng với các lệnh di chuyển về phía trước và lùi lại. Bạn đã tìm hiểu cách ngừng vẽ của turtle bằng lệnh up, và bắt đầu vẽ lại với lệnh down. Bạn cũng đã khám phá cách turtle xoay theo độ.

---

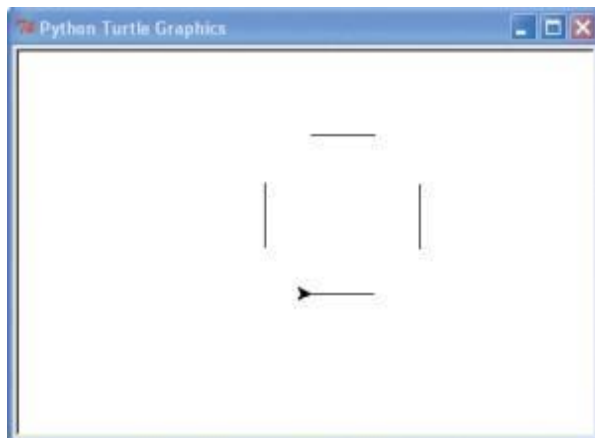
## Các câu đố lập trình

Hãy thử vẽ một số hình dưới đây bằng cách sử dụng turtle. Các đáp án có thể được tìm thấy tại <http://python-for-kids.com/>.

**#1: Một Hình Chữ Nhật** Tạo một canvas mới bằng cách sử dụng hàm Pen của mô-đun turtle và sau đó vẽ một hình chữ nhật.

**#2: Một Hình Tam Giác** Tạo một canvas khác, và lần này vẽ một hình tam giác. Hãy tham khảo lại biểu đồ của vòng tròn với các độ ("Di chuyển Turtle" trên trang 46) để nhắc lại hướng xoay của turtle sử dụng độ.

**#3: Một Hình Hộp Không Có Góc** Viết một chương trình để vẽ bốn đường thẳng như hình dưới đây (kích thước không quan trọng, chỉ cần đúng hình dạng):



# Firewalls and Internet Security Second Edition

Repelling the Wily Hacker

William R. Cheswick  
Steven M. Bellovin  
Aviel D. Rubin



DRAFT COVER  
as of 12/02



eastman  
bring to you



ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

## Chương 15: Phát hiện xâm nhập

“Kẻ ngốc nói rằng: ‘Đừng để tất cả trứng vào cùng một giỏ’ - đây chỉ là cách nói. Hãy ‘phân tán tiền và sự chú ý của bạn’. Nhưng người khôn ngoan nói rằng: ‘Hãy để tất cả trứng vào cùng một giỏ và... trông chừng cái giỏ đó!’”

### — Puddin' Head Wilson's Calendar

Điều quan trọng là phải bố trí người canh gác gần những thứ bạn muốn bảo vệ, và **hệ thống phát hiện xâm nhập (Intrusion Detection System - IDS)** giúp thực hiện chức năng này. Là một sản phẩm thương mại, các công cụ bảo mật này đã được quảng bá như một giải pháp tối ưu cho các hành vi xâm nhập mạng, và nhiều nhà quản lý IT đã tuyên bố rằng mạng của họ an toàn vì họ đã cài đặt tường lửa và IDS mới nhất. Những điều này có thể hữu ích, nhưng chúng còn xa mới đạt đến mức lý tưởng.

Có một số loại hệ thống phát hiện xâm nhập. **IDS mạng (Network IDS - NIDS)** giám sát lưu lượng mạng để tìm các dấu hiệu của sự xâm nhập. Một số hệ thống dựa trên máy chủ quét các tệp hoặc lưu lượng truy cập để tìm virus; một số khác phân tích các mẫu cuộc gọi hệ thống hoặc kiểm tra các tệp bị thay đổi.

IDS phải đối mặt với một số hạn chế cố hữu. **Dương tính giả (False positive)** xảy ra khi IDS kết luận sai rằng một cuộc xâm nhập đã xảy ra. **Âm tính giả (False negative)** là các xâm nhập thực sự mà IDS bỏ sót. Với hầu hết các hệ thống phát hiện xâm nhập, cả hai vấn đề này đều không thể tránh khỏi và xảy ra với tần suất cao đến mức làm giảm đáng kể giá trị của chúng. Thông thường, cần có sự can thiệp của con người để xác định vấn đề hoặc thiếu sót, và một số nguồn gây ra lỗi trong các gói tin bị sai có thể quá khó để khắc phục.

Cuối cùng, các hệ thống IDS mạng thường hoạt động bằng cách “ngửi” lưu lượng mạng và ghép các gói tin lại thành dòng dữ liệu. Làm điều này một cách hợp lý nghe có vẻ đơn giản. Hầu hết các chương trình dò tìm chỉ làm điều đó, nhưng một số bài báo (như của Ptacek và Newsham [1998], Paxson [1998], và đặc biệt là Handley và cộng sự [2001]) chỉ ra rằng công việc này gần như không thể thực hiện chính xác. Vấn đề là một chương trình dò tìm cần biết trạng thái của các ngăn xếp TCP/IP ở cả hai đầu của liên lạc, cộng với các đặc điểm riêng của các chi tiết triển khai. Ví dụ, giả sử hai gói tin đến chứa dữ liệu chồng chéo nhau.

Vấn đề dữ liệu trùng lặp có vẻ như không phổ biến, nhưng các chương trình như **fragrouter** [Song *et al.*, 1999] được thiết kế để tạo luồng TCP/IP không bình thường nhằm gây nhầm lẫn cho các hệ thống phát hiện xâm nhập. Fragrouter sử dụng các kịch bản viết bằng một ngôn ngữ nhỏ để định nghĩa các vấn đề mong muốn trên luồng gói tin. Dữ liệu gửi đi có thể bị biến dạng đến mức hệ thống giám sát không thể giải mã được.

**Dữ liệu chồng lấn có thể xuất hiện khi các gói tin được tái hợp lại và có hai phiên bản dữ liệu trùng lặp cho một vùng. Hệ thống sẽ sử dụng phiên bản nào? Các tiêu chuẩn RFC không đề cập đến vấn đề này, và các cách triển khai có thể khác nhau. Nếu dữ liệu trong hai gói tin không khớp, phiên bản nào mà Hệ thống phát hiện xâm nhập mạng (NIDS) sẽ cho là đúng?**

**Vấn đề dữ liệu chồng lấn nghe có vẻ hiếm, nhưng nó có thể xảy ra trong thực tế. Ví dụ, các chương trình như fragrouter [Song et al., 1999] cố tình thay đổi luồng TCP/IP để gây nhầm lẫn cho các hệ thống giám sát. Fragrouter sử dụng các đoạn mã nhỏ nhằm tạo ra**

các lỗi cụ thể trong dữ liệu gói tin. Kết quả là luồng gói tin đi ra có thể bị bóp méo đến mức hệ thống giám sát không thể giải mã luồng dữ liệu.

Có bốn nơi mà các luồng TCP/IP bất thường cần được xử lý đúng cách: khách hàng, máy chủ, tường lửa, và NIDS. [Handley et al., 2001] đề xuất sử dụng một thiết bị trung gian để chuẩn hóa luồng gói tin. Một cách tiếp cận khác là bổ sung chức năng này vào tường lửa, làm cho nó hoạt động giống như một cổng mạch (circuit-level gateway). Một số tường lửa đã thực hiện điều này; chúng tái hợp các gói tin bị phân mảnh để bảo vệ khỏi các cuộc tấn công bằng gói tin ngắn. Các cổng mạch thực sự cũng có thể làm sạch luồng IP.

Vấn đề này là cơ sở cho khuyến nghị của chúng tôi trong lần xuất bản đầu tiên rằng các tổ chức nên tránh kết nối IP trực tiếp giữa mạng nội bộ và Internet, và thay vào đó sử dụng tường lửa ứng dụng hoặc tường lửa cấp độ mạch.

### 15.1 Nơi cần giám sát

Điều quan trọng là phải hiểu những hạn chế của Hệ thống Phát hiện Xâm nhập (IDS) trước khi triển khai. Một câu hỏi cần đặt ra là: "Mục đích của IDS là gì?" Một lý do chính đáng để cài đặt IDS ngoài tường lửa là để biện minh cho việc cấp ngân sách (vì đây là một mô hình môi đe dọa trong đó quản lý là "kẻ thù"). Không cần thiết phải giám sát lưu lượng ngoài mạng của bạn để xem liệu bạn có đang bị tấn công hay không — bạn biết mình là mục tiêu. Điều đó không có nghĩa là bạn nên bỏ qua lưu lượng bên ngoài, nhưng tốt hơn là ghi lại và lưu trữ lưu lượng bên ngoài để phân tích sau, thay vì cố gắng phát hiện xâm nhập trong thời gian thực.

Nếu bạn là nhà nghiên cứu học hỏi về các cuộc tấn công mới, thông tin như vậy là vô giá. Tuy nhiên, có quá nhiều lưu lượng xảy ra, và IDS là công cụ quá yếu để thực hiện phân tích thời gian thực. Đây là một công cụ tốt để huấn luyện những người mới làm quen với mạng và thiết bị IDS.

Thiết bị IDS trở nên hữu ích hơn khi được triển khai gần các tài sản quan trọng, bên trong các lớp bảo mật khác nhau. Chúng giống như một camera an ninh được lắp đặt trong kết sắt của ngân hàng, cung cấp một lớp bảo đảm cuối cùng rằng mọi thứ đều ổn. Khi quyền truy cập thông thường vào mạng hoặc máy tính bị giới hạn nghiêm ngặt hơn, các quy tắc dành cho thiết bị phát hiện cũng cần phải nhạy cảm hơn. Con người không nên thực hiện các truy vấn web từ máy tính bằng lương.

### 15.2 Các loại IDS

Các loại IDS khác nhau có điểm mạnh và điểm yếu khác nhau.

#### **IDS dựa trên chữ ký (Signature-based IDS):**

Loại này có một cơ sở dữ liệu các cuộc tấn công đã biết; bất kỳ điều gì khớp với mục

trong cơ sở dữ liệu sẽ bị gấn cò. Bạn sẽ không gặp nhiều cảnh báo sai nếu hệ thống được điều chỉnh đúng cách, nhưng khả năng cao sẽ bỏ sót các cảnh báo đúng vì hệ thống chỉ nhận diện những gì nằm trong cơ sở dữ liệu. Thật không may, việc tìm được sự cân bằng giữa chữ ký rộng (khớp với lưu lượng bình thường) và chữ ký hẹp (để bị mã độc qua mặt) là rất khó. Tốt nhất, các hệ thống dựa trên chữ ký nên tích hợp ngữ cảnh thay vì chỉ dựa vào việc so khớp chuỗi.

### **IDS dựa trên bất thường (Anomaly-based IDS):**

Những hệ thống này tìm kiếm các hành vi **không bình thường** và có khả năng đưa ra các cảnh báo sai hoặc bỏ sót cảnh báo đúng. Chúng hoạt động tốt nhất trong môi trường với phiên bản định nghĩa chặt chẽ về "bình thường," nơi dễ xác định khi một điều gì đó không nên xảy ra. Mục đích càng đặc thù của một máy, thì hành vi "bình thường" càng bị giới hạn, và khả năng cảnh báo sai càng giảm.

Phát hiện bất thường là một lĩnh vực nghiên cứu thú vị, nhưng đến nay vẫn chưa tạo ra nhiều công cụ thực tiễn. [Forrest et al., 1996] và [Ko et al., 2000] đã cho thấy một số kết quả đáng chú ý. Forrest phát triển một công cụ giám sát các quy trình chạy trên máy tính và kiểm tra các lệnh hệ thống. Công cụ này có khái niệm về mẫu lệnh "bình thường" và nhận ra khi điều gì đó không đúng xảy ra. Nó sử dụng **n-grams** của các lệnh hệ thống và phân tích thứ tự thực thi để phát hiện bất thường.

### **IDS dựa trên máy chủ hoặc mạng (Host-based & Network-based IDS):**

**Hai loại này bổ sung cho nhau, không loại trừ nhau. Hệ thống dựa trên máy chủ thường biết trạng thái của chính máy đó, giúp xử lý dữ liệu dễ hơn, nhưng phần mềm có thể bị xâm phạm nếu máy chủ bị tấn công. Hệ thống dựa trên mạng là các thiết bị độc lập và thường ít bị tấn công hơn.**

**Một số môi trường như DMZ (vùng phi quân sự hóa) yêu cầu một loại IDS đặc biệt gọi là honeypot** — một máy tính mà không ai được phép truy cập. Bất kỳ lưu lượng nào đến máy này có thể được xem là hành vi đáng ngờ. Honeypot không phù hợp trong môi trường sản xuất mở, nhưng nó phù hợp để phát hiện các cuộc tấn công mà không ảnh hưởng đến các tài nguyên dành riêng.

Một hệ thống honeypot (bẫy mật) trên mạng Internet công cộng có thể hữu ích để nghiên cứu hành vi của hacker, mặc dù một số hacker đã học cách né tránh chúng. Một trong những ví dụ đẹp nhất là *honeyd* của Niels Provos [Spitzner, 2002, Chương 8]. Nó mô phỏng một mạng lưới hoàn chỉnh, bao gồm nhiều loại máy móc khác nhau. Tuy nhiên, bạn không thể dựa vào nó để xác định liệu ai đó đã xâm nhập vào một máy đơn lẻ hay chưa; tối đa, nó chỉ có thể phát hiện các lượt quét. Để xử lý các cảnh báo dương tính giả và âm tính giả, một số người sử dụng nhiều hệ thống IDS (Hệ thống Phát hiện Xâm nhập) có đầu ra được đồng bộ hóa. Sự tương quan thời gian có thể được sử dụng để phát hiện các cuộc tấn công "chậm và âm thầm."

### 15.3 Quản trị một hệ thống IDS

Một hệ thống phát hiện xâm nhập (IDS) yêu cầu một lượng tài nguyên đáng kể. IDS phải được cài đặt ở các vị trí chiến lược, cấu hình đúng cách và giám sát thường xuyên. Trong hầu hết các môi trường, chúng sẽ phải xử lý một lượng lớn lưu lượng mạng không hợp lệ. Ví dụ, một trình điều khiển máy in HP mà chúng tôi đã sử dụng đã cố tìm mọi thứ trên mạng con mà không biết về mặt nạ mạng, dẫn đến quét toàn bộ mạng /16 để tìm một máy in HP. Phần mềm quản lý mạng đôi khi cũng làm điều tương tự. Người vận hành IDS phải biết cách xử lý loại lưu lượng này và cần có khả năng chịu đựng một lượng lớn "tiếng ồn" [Bellovin, 1993]. Họ cũng cần đảm bảo rằng họ không trở nên quá tự mãn vì IDS có xu hướng "báo động giả."

### 15.4 Công cụ IDS

Có rất nhiều công cụ IDS có sẵn, cả miễn phí lẫn thương mại. Các công cụ như **snort** (xem phần sau), **ethereal**, và **bro** [Paxson, 1998] rất hữu ích. Ethernal cung cấp một giao diện đồ họa (GUI) đẹp, cho phép bạn tái hiện các luồng TCP để xem dữ liệu ở cấp độ ứng dụng. Nó cũng có thể ghi lại lưu lượng mạng để phân tích sau.

Các sản phẩm thương mại dao động từ kém chất lượng đến khá hữu ích. Một số sản phẩm cố gắng áp dụng kỹ thuật trí tuệ nhân tạo để giải quyết vấn đề. Một số khác thu thập thông tin phân tán và cố gắng lắp ráp một cái nhìn tổng thể về cuộc tấn công.

#### 15.4.1 Snort

Có lẽ chương trình phát hiện xâm nhập miễn phí phổ biến nhất là **snort**, được phát triển bởi Martin Roesch. Snort là mã nguồn mở, và có một cộng đồng lớn các người dùng và người đóng góp tích cực. Xem thêm tại:

<http://www.snort.org/>

Chương trình có sẵn trên nhiều nền tảng — nó hoạt động ở bất kỳ đâu mà **libpcap** chạy được.

Snort có thể được sử dụng theo nhiều cách khác nhau. Nó có thể "ngửi" mạng và tạo đầu ra theo định dạng **tcpdump**. Nó cũng có thể được sử dụng để ghi lại các gói tin, để các công cụ khai thác dữ liệu hoặc chương trình của bên thứ ba phân tích lưu lượng mạng sau đó. Tính năng thú vị nhất của snort là khả năng thiết kế một bộ quy tắc nhận diện các mẫu lưu lượng cụ thể. Nhiều quy tắc đã có sẵn cho snort và chúng thường được chia sẻ giữa người dùng và đăng tải trên Internet. Snort có thể được cấu hình để nhận diện các khảo sát bằng **nmap**, các cuộc tấn công tràn bộ đệm đã biết, các khai thác CGI đã biết, và các lưu lượng thăm dò như quét cổng.

Những nỗ lực để nhận dạng hệ điều hành dựa trên các đặc điểm của ngăn xếp mạng, và nhiều kiểu tấn công khác mà quản trị viên muốn cấu hình quy tắc. Dưới đây là một ví dụ quy tắc được lấy từ [Roesch, 1999]:

**bash**

**Copy**

**code**

```
activate tcp !$HOME_NET any -> $HOME_EIET 143 (flags: PA; content:
"|E8C0FFFFFF|bin|; activates: 1; msg: "(buffer overflow!");
```

```
dynamic tcp !$HOME_NET any -> $HOME_NET 143 (activated_by: 1; count: 50);
```

Quy tắc trên chỉ ra rằng một cảnh báo sẽ được gửi khi một lỗi tràn bộ đệm IMAP được phát hiện. Tại thời điểm đó, 50 gói tin đến tiếp theo hướng đến cổng 143 sẽ được ghi lại. Một số trong những gói tin này có thể chứa thông tin quan trọng về cuộc tấn công mà một nhà phân tích hoặc quản trị viên mạng quan tâm.

Tuy nhiên, có một lỗ hổng: Quy định về cờ "PA" có nghĩa là cả hai bit PUSH và ACK phải được đặt trên gói tin để nó phù hợp với quy tắc này. Điều này khá dễ để kẻ tấn công né tránh bằng cách đảm bảo rằng PUSH không được đặt.

Như mong đợi từ các công cụ phát hiện xâm nhập hữu ích, Snort cung cấp các cơ chế cảnh báo linh hoạt, từ cửa sổ bật lên trên màn hình đến email và thông báo qua thiết bị nhắn tin. Có các nhóm người dùng Snort thường tụ họp để so sánh dữ liệu, chia sẻ tập luật, thảo luận về kết quả dương tính giả, và đề xuất các cải tiến có thể cho chương trình. Ngoài ra, còn có một diễn đàn trực tuyến với rất nhiều thông tin hữu ích tại <http://snort.rapidnet.com/>.

Và đúng vậy, luôn có một "cuộc chạy đua vũ trang" giữa những kẻ tấn công và các nhà phát triển script Snort.



## Chương 19: Chúng ta sẽ đi đâu từ đây?

Chúng tôi hy vọng rằng đến thời điểm này, hai điểm sau đã được làm rõ: rằng thực sự có một mối đe dọa, nhưng mối đe dọa này có thể được kiểm soát thông qua các kỹ thuật phù hợp, bao gồm việc sử dụng tường lửa. Tuy nhiên, tường lửa không phải là giải pháp toàn diện cho bảo mật. Vẫn còn nhiều việc phải làm.

Đây là dự đoán của chúng tôi về tương lai. Chúng tôi đã từng sai trước đây và có lẽ sẽ sai nữa. (Một trong số chúng tôi, Steve, là một trong những nhà phát triển của NetNews. Ông từng dự đoán rằng lưu lượng của NetNews cuối cùng chỉ là một hoặc hai thông điệp mỗi ngày trong 50 đến 100 nhóm thảo luận.)

*“Thật khó để đưa ra dự đoán, đặc biệt là về tương lai.”*

— YOGI BERRA

### 19.1 IPv6

Khi nào IPv6 sẽ được triển khai rộng rãi? IPv6 nên sớm được triển khai trong thế hệ điện thoại di động mới; hiện tại nó cũng đang được áp dụng tại Trung Quốc và Nhật Bản. Các bộ định tuyến xương sống hiện nay không hỗ trợ chuyển tiếp IPv6 ở mức phần cứng, và các phiên bản phần mềm không đủ hiệu quả để xử lý lưu lượng lớn. Vào cuối những năm 1990, các nhà cung cấp dịch vụ Internet (ISP) thường thay đổi bộ định tuyến trong vòng 18 tháng, di chuyển các bộ định tuyến lõi ra vùng biên. Xu hướng này đã chậm lại gần đây do sự suy thoái kinh tế.

Phần lớn các máy khách UNIX và Linux đã hỗ trợ IPv6. Windows XP có hỗ trợ dành cho nhà phát triển đối với IPv6; Microsoft đã công bố rằng hỗ trợ đầy đủ cho người dùng sẽ có trong bản phát hành chính tiếp theo của Windows, dự kiến vào khoảng năm 2004 nếu họ giữ đúng lịch trình. Trong vòng bốn năm sau đó, IPv6 có thể được triển khai rộng rãi. Nhưng liệu nó có được sử dụng hay không?

Chưa rõ các động lực kinh tế nào sẽ khiến các công ty bỏ thời gian và công sức để chuyển sang IPv6. Đúng là vấn đề thiếu không gian địa chỉ sẽ được giải quyết, nhưng hầu hết các mạng nội bộ lớn sử dụng không gian địa chỉ riêng và NAT để xử lý vấn đề này.

Các công ty có thể muốn cải thiện khả năng kết nối với các điện thoại di động được đề cập trước đó (thoại qua IP?), mà không cần thông qua bộ dịch.

Một động lực mạnh mẽ là sự xuất hiện của các dịch vụ Internet trên IPv6 không khả dụng trên IPv4. Nhưng khó có thể tưởng tượng một trang web sẽ giới hạn mình chỉ ở giao thức mới. Thêm nữa, hầu hết các dịch vụ trên v6 đều có thể được thực hiện trên v4, miễn là có đủ không gian địa chỉ. Một ứng dụng tiềm năng là mạng ngang hàng (peer-to-peer) — nếu các ứng dụng hợp pháp trở nên phổ biến đủ mức.

Lý do rõ ràng nhất để chuyển sang IPv6 là vấn đề không gian địa chỉ, cũng là động lực ban đầu để thiết kế IPv6. Không gian IPv4 khan hiếm và được cho là có giá trị cao. Nếu những địa chỉ này được bán đấu giá và hình thành một thị trường không gian địa chỉ (chắc chắn không phải là truyền thống của Internet), thì sẽ có động lực kinh tế mạnh mẽ để chuyển đổi. Xem thêm [Rekhter et al., 1997] để biết thêm chi tiết.

Ba chúng tôi có ý kiến khác nhau về ngày triển khai IPv6 rộng rãi, nhưng đồng ý rằng năm 2008 là thời điểm sớm nhất chúng tôi có thể thấy việc sử dụng phổ biến.

## 19.2 DNSsec

Việc thiếu xác thực trong các phản hồi DNS là một trong những điểm yếu nhất của Internet. Trong bối cảnh Web, vấn đề này càng nghiêm trọng. Chúng ta cần một thứ như DNSsec, và khi các công cụ tấn công giả mạo DNS trở nên phổ biến hơn, việc sử dụng Web như hiện nay có thể bị đình trệ nếu không có biện pháp. Do đó, chúng tôi dự đoán rằng, mặc dù có những vấn đề liên quan đến PKI (ai là gốc?), DNSsec sẽ được triển khai. Bảo mật mà nó mang lại quá quan trọng, và các vấn đề nó giải quyết không thể được khắc phục bằng cách khác. Cuối cùng, một số khóa công khai sẽ được tích hợp trong các bản phân phối khách hàng DNS, và các phản hồi DNS sẽ được ký.

Tuy nhiên, việc triển khai rộng rãi DNSsec không phải là không có thách thức. Chúng ta có thể đủ khả năng để ký miền .COM không? Dấu ấn bộ nhớ của một miền cấp cao được ký sẽ rất lớn. Tuy nhiên, chúng tôi tin rằng những vấn đề này có thể được khắc phục.

Nghiêm trọng hơn, quá nhiều trang web không quan tâm đến bảo mật cho đến khi họ thực sự bị tổn thất. Chúng ta chỉ có thể tiến xa bằng cách đưa các biện pháp bảo vệ vào cơ sở hạ tầng.

## 19.3 Microsoft và Bảo mật

Gần đây, truyền thông đưa tin rằng Microsoft sẽ tập trung vào bảo mật. Điều này dường như là thật, không chỉ là tuyên truyền. Họ đang cung cấp các khóa học nâng cao nhận thức và đào tạo bảo mật rộng rãi, phát triển các công cụ kiểm toán bảo mật mới, và văn hóa doanh nghiệp của họ đã bắt đầu thay đổi. Chúng tôi hoan nghênh nỗ lực này và hy vọng rằng các ngành khác sẽ noi gương.

Tuy nhiên, sẽ mất rất lâu để thấy kết quả thực sự. Ngoài cơ sở đã được cài đặt và yêu cầu tương thích ngược, khối lượng mã không lỗi cần được xem xét và độ phức tạp của nó tạo ra nhiều cơ hội cho các hành vi bất thường.

## 19.4 Tính phổ biến của Internet

Rõ ràng, ngày càng nhiều thiết bị sẽ được kết nối với mạng nội bộ, nếu không phải là Internet. Các khóa cửa khách sạn, tủ lạnh, bộ điều nhiệt, lò sưởi, hệ thống liên lạc nội bộ trong nhà và thậm chí cả hộp thư đã được kết nối mạng. Nhưng làm thế nào để một công tắc đèn trong một ngôi nhà thông minh biết được ai là người đáng tin cậy?

Một trong số chúng tôi đã thực hiện nhiều thí nghiệm với một ngôi nhà có dây kết nối. Phần khó khăn không nằm ở thiết bị điện tử, thiết bị hoặc thậm chí là nghĩ ra những điều hữu ích để làm; mà chính là các nhiệm vụ quản trị hệ thống – những công việc mà bạn phải thêm vào danh sách việc nhà vào mỗi thứ Bảy. Liệu những hệ thống này có thể được triển khai ở quy mô lớn cho công chúng không? Nếu có, liệu ngôi nhà của chúng ta sẽ trở nên hữu ích hơn nhưng kém an toàn hơn?

Bên cạnh các ứng dụng thông thường của kết nối Internet liên tục đến nhà, còn có nhiều khả năng thú vị cho các dịch vụ mới. Các chương trình tự động có thể thông báo các cảnh báo thời tiết và các tình huống khẩn cấp khác. Chúng tôi từng nghe thấy các thông báo giọng nói về quỹ đạo vệ tinh và các sự kiện thiên văn khác, lời nhắc nhở mang rác đi đổ và tái chế, cùng nhiều thông báo khác. Nhiều dịch vụ trong số này có tính thời gian nhạy cảm và có thể được tiếp thị như một dịch vụ nếu có đủ nhu cầu.

Các dịch vụ như TiVo có thể giúp tích hợp giải trí gia đình với lập lịch động. Mạng ngang hàng (peer-to-peer) hiện đã cung cấp một lượng lớn nội dung âm nhạc, mặc dù theo cách ngẫu nhiên và có lẽ là bất hợp pháp. Một cách nào đó, việc tiếp cận giải trí sẽ ngày càng phát triển.

## 19.5 Bảo mật Internet

Bảo mật trên Internet đã suy giảm trong suốt 20 năm qua, và cuộc sống số sẽ trở nên nguy hiểm hơn trong tương lai. Những kẻ viết virus cho máy tính cá nhân có thể chiến thắng trong cuộc chiến với các phần mềm phòng chống virus. Hãy tưởng tượng một thế giới mà phần mềm kiểm tra virus hoàn toàn không hoạt động. Cuối cùng, vấn đề **halting problem** (vấn đề ngừng máy) không đứng về phía chúng ta. Ít nhất, các phần mềm kiểm tra virus sẽ phải tiêu tốn ngày càng nhiều thời gian CPU để xác định xem một tệp có bị nhiễm hay không. Nếu chúng ta không thể tin tưởng vào phần mềm kiểm tra virus, chúng ta sẽ phải quay lại sử dụng các biện pháp vệ sinh mạng tốt hơn, các tệp nhị phân đã được ký, và một cơ sở tính toán đáng tin cậy hơn (**Trusted Computing Base - TCB**).

Cơ sở hạ tầng Internet sẽ đối mặt với các cuộc tấn công ngày càng nhiều. Các điểm dễ bị tổn thương nhất là các máy chủ tên DNS, giao thức BGP, và các chế độ lỗi phổ biến của bộ định tuyến [Schneider, 1999].

Có một phong trào mạnh mẽ nhằm bảo mật quá trình khởi động và xác minh hệ điều hành cùng tất cả các ứng dụng trên hệ thống. Các nhà sản xuất phần cứng lớn, bao gồm

Compaq, HP, IBM và Intel, đã thành lập **Liên minh Nền tảng Tính toán Tin cậy** (Trusted Computing Platform Alliance - TCPA). Ý tưởng này nhằm làm cho máy tính ít bị tổn thương hơn trước các mã độc như Trojan. Microsoft cũng là một phần của TCPA và đang tích cực phát triển **Palladium**, một nền tảng phần mềm được thiết kế để hỗ trợ TCPA. Các ứng dụng của nền tảng này bao gồm quản lý quyền kỹ thuật số (digital rights management) và đảm bảo an ninh toàn bộ hệ thống.

Nhiều kế hoạch như TCPA/Palladium và các nỗ lực bảo mật khác đặt ra nguy cơ tiềm tàng đối với quyền riêng tư, tính mở của các nền tảng, và khả năng các bên thứ ba phát triển phần mềm. Mặc dù những vấn đề này không phải là trọng tâm của cuốn sách, chúng là những yếu tố quan trọng phát sinh từ các nỗ lực đối phó với các mối đe dọa ngày càng tăng trên Internet. Có đáng để mua một máy tính an toàn hơn nhưng lại ít quyền riêng tư hơn và ít lựa chọn nhà cung cấp phần mềm hơn?

Ngoài ra, còn có các câu hỏi khác cần xem xét. Phiên bản Red Hat Linux tiếp theo có chứa khóa công khai trong ROM của máy tính xách tay IBM Thinkpad mới không? Điều này hoàn toàn có thể xảy ra. Nếu bạn mua một đầu DVD hỗ trợ Internet trên eBay, làm thế nào để nó được cài đặt lại để nhận diện bạn là chủ sở hữu mới, đồng thời thu hồi quyền truy cập của chủ sở hữu cũ? Làm thế nào để bảo mật một ngôi nhà được kết nối mạng? Nếu máy giặt muốn gửi dữ liệu về nhà sản xuất, các gói tin đó sẽ đi qua tường lửa của bạn bằng cách nào? Bạn có muốn cho phép không? (Liệu bảo hành của máy giặt có giới hạn số lần bạn được phép sử dụng? Máy giặt có thông báo cho nhà sản xuất rằng bạn đã vận hành nó không đúng cách không? Ai sở hữu dữ liệu của máy giặt đó và làm thế nào để chủ sở hữu kiểm soát việc sử dụng dữ liệu đó?)

## 19.6 Kết luận

Trong cuốn sách này, chúng tôi đã đề cập đến bảo mật Internet trong bối cảnh thế giới ngày nay. Mặc dù chúng tôi không biết những vấn đề tương tự sẽ được giải quyết ra sao trong tương lai, nhưng chúng tôi chắc chắn rằng những nguyên tắc bảo mật đã hướng dẫn con người trong ba thập kỷ qua, và có lẽ trong suốt 5000 năm qua, sẽ tiếp tục đúng.

Như Karger và Schell đã chỉ ra, chúng ta đang đi thụt lùi, không phải tiến lên; các hệ thống ngày nay thậm chí không đạt được mức bảo mật mà Multics đã có vào những năm 1970 [Karger và Schell, 2002]. Chúng ta đang mất lợi thế. Chúng ta không thể để điều này tiếp diễn, và cần làm tốt hơn.

“Tôi đã quyết định rồi. Tôi muốn thấy núi non một lần nữa, Gandalf – những ngọn núi, và sau đó tìm một nơi mà tôi có thể nghỉ ngơi. Trong yên bình và tĩnh lặng, không bị họ hàng tò mò quấy rầy, không có những người khách không mời bấm chuông liên tục. Có lẽ tôi sẽ tìm một nơi để hoàn thành cuốn

sách của mình. Tôi đã nghĩ đến một kết thúc đẹp cho nó: *và ông đã sống hạnh phúc mãi mãi đến cuối đời mình.*”

Gandalf cười lớn. “Tôi hy vọng ông sẽ như vậy. Nhưng không ai đọc cuốn sách đó, dù nó kết thúc ra sao.”

“Ồ, có thể họ sẽ đọc, vào một ngày nào đó.”

**Bilbo Baggins in Lord of the Rings - J.R.R.TOLKIEN**