

# 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 đều 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:

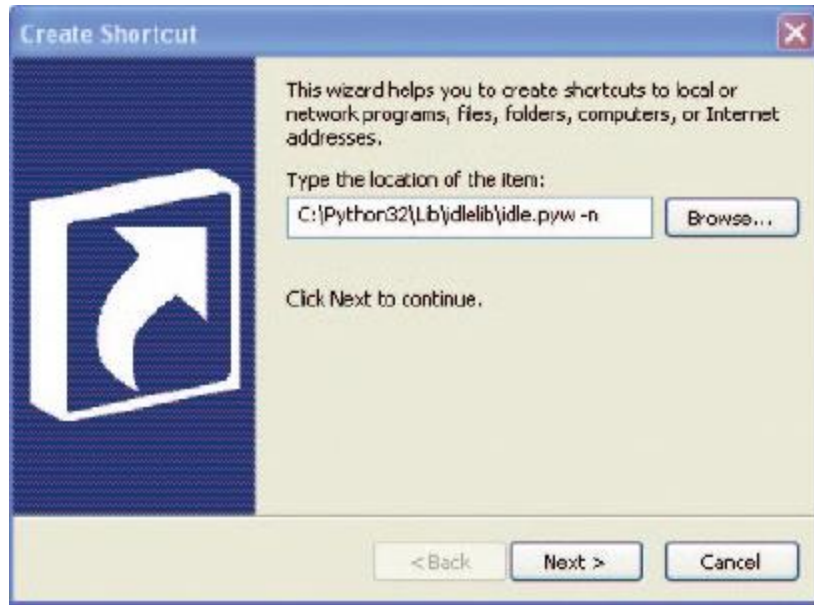
1. Nhấp chuột phải vào màn hình chính của bạn, rồi chọn **New ► Shortcut** từ menu bật lên.
2. Nhập vào ô nơi có dòng **Type the location of the item** (đảm bảo rằng thư mục bạn nhập vào là giống với thư mục bạn đã ghi lại trước đó):

`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.

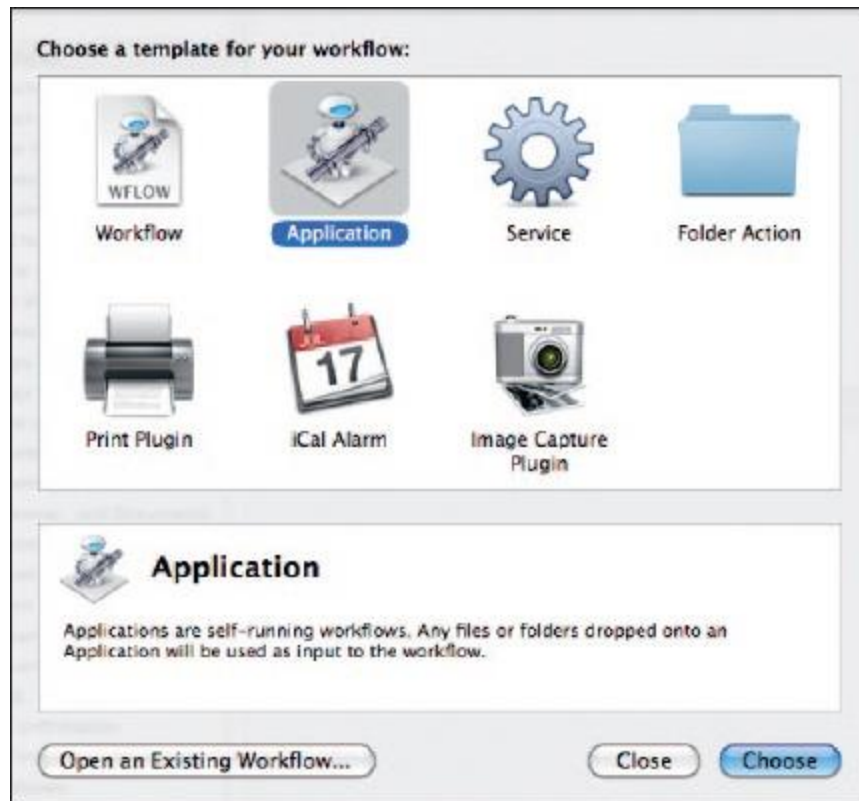
Sau khi tệp đã được tải xuống (nó sẽ có phần mở rộng tên tệp là .dmg), hãy nhấp đúp vào nó. Bạn sẽ thấy một cửa sổ hiển thị nội dung của tệp.



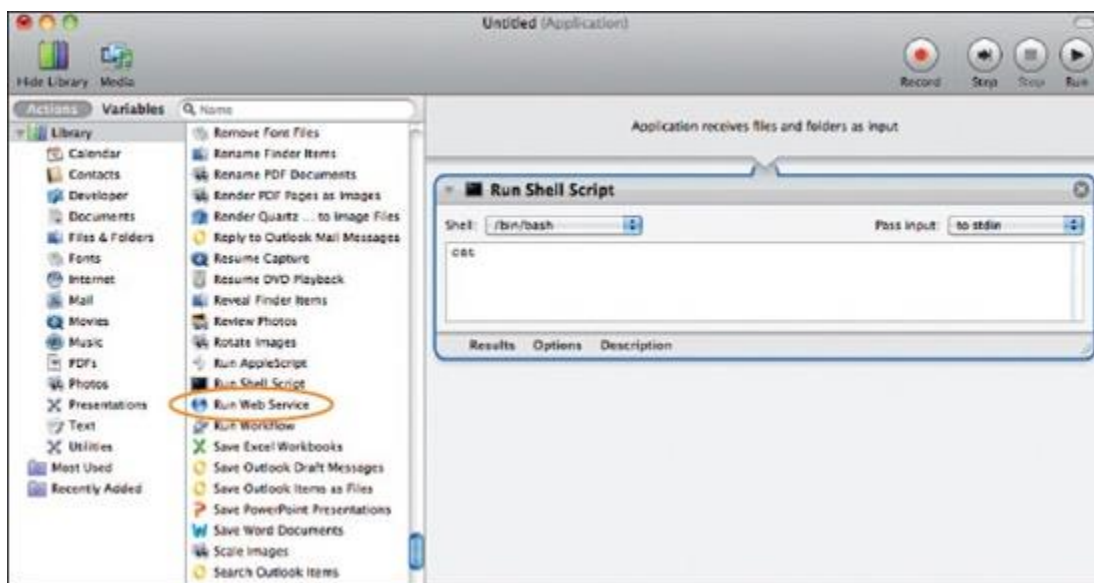
Trong cửa sổ này, hãy nhấp đúp vào tệp Python.mpkg, sau đó làm theo hướng dẫn để cài đặt phần mềm. Bạn sẽ được yêu cầu nhập mật khẩu quản trị viên cho máy Mac của bạn trước khi Python được cài đặt. (Không biết mật khẩu quản trị viên? Cha mẹ của bạn có thể cần nhập nó.)

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.

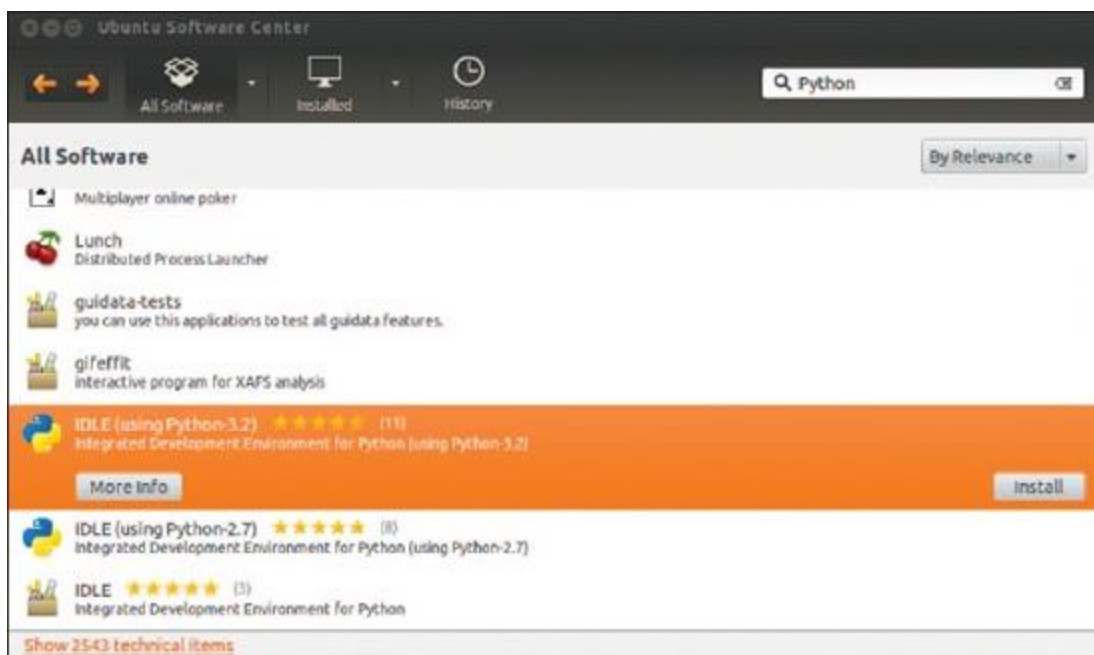
## Cài đặt Python trên Ubuntu

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:

A screenshot of a Windows-style window titled "Python Shell". The window has a menu bar with "File", "Edit", "Debug", "Options", "Windows", and "Help". The main text area shows 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 ====
>>>
```

The status bar at the bottom right of the window shows "Ln: 4 Col: 4".

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")
```

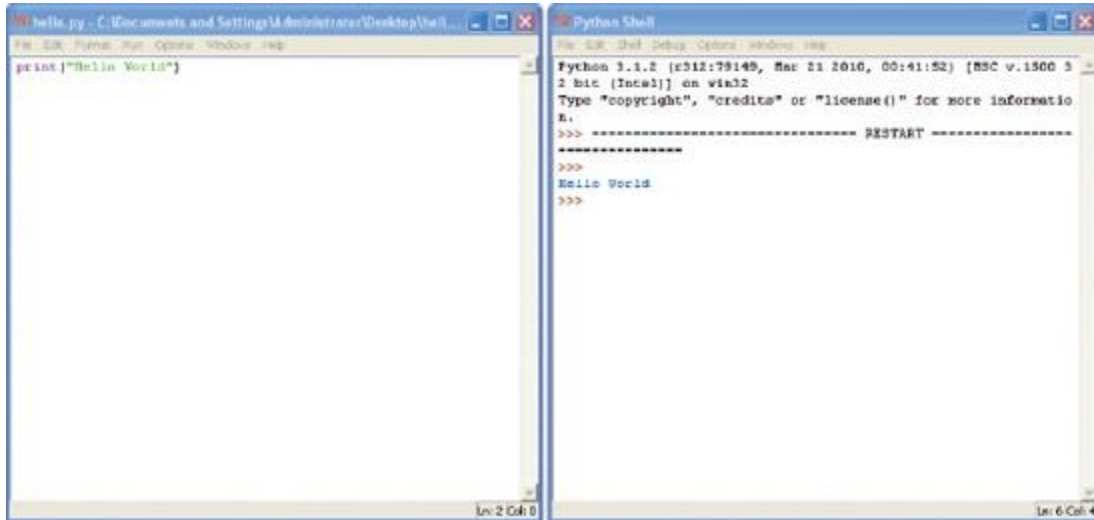
Bây giờ, hãy làm theo các bước sau để lưu và chạy chương trình Python của bạn:

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:



```
hello.py - C:\Documents and Settings\Administrator\Desktop\hell...
File Edit Format Run Options Window Help
print('Hello World')
Ln: 2 Col: 0

Python Shell
File Edit Shell Debug Options Window Help
Python 3.1.2 [x312:79149, Mar 21 2010, 00:41:52] [MSC v.1500 3
2 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more informatio
n.
>>> ===== RESTART =====
>>>
Hello World
>>>
```

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.)

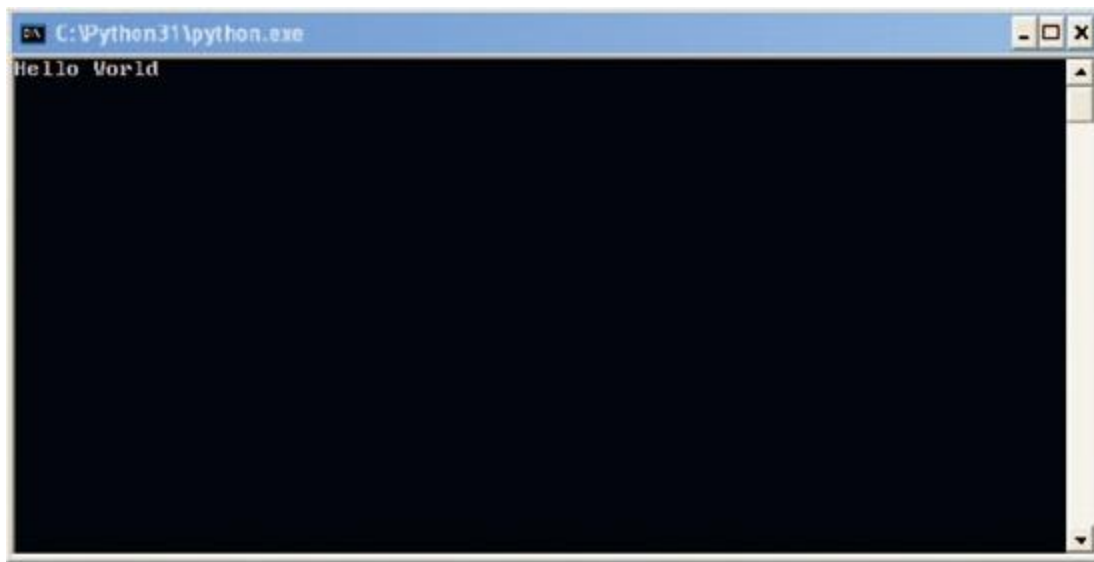




Sau khi chạy mã, bạn sẽ thấy một biểu tượng mới trên màn hình máy tính với nhãn `hello.py`. Nếu bạn nhấp đúp vào biểu tượng này, một cửa sổ màu đen sẽ xuất hiện trong chốc lát rồi biến mất. Chuyện gì đã xảy ra?

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 lên 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
```

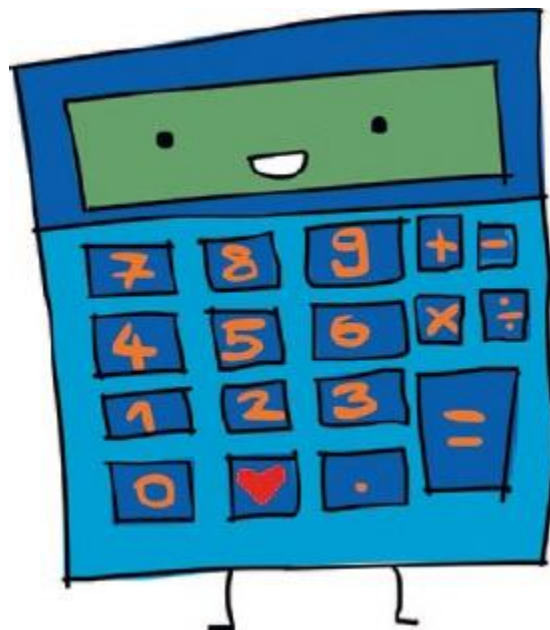
```
605
```

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 nhãn (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.) Thỉnh 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:

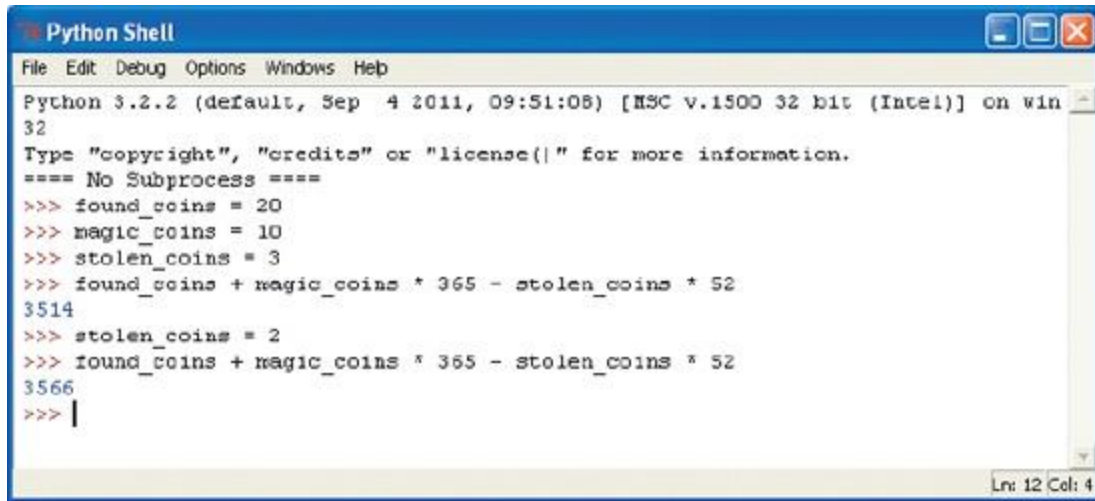
```
Python Shell
File Edit Debug Options Windows Help
Python 3.2.2 (default, Sep  4 2011, 09:51:05) [MSC v.1500 32 bit (Intel)] on win
32
Type "copyright", "credits" or "license()" for more information.
==== No Subprocess ====
>>> found_coins = 20
>>> magic_coins = 10
>>> stolen_coins = 3
>>> found_coins + 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.



```
Python Shell
File Edit Debug Options Windows Help
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.