2

BIẾN VÀ
Kiểu dữ liệu đơn giản

Trong chương này, bạn sẽ tìm hiểu về các loại dữ liệu khác nhau mà bạn có thể làm việc trong các chương trình Python của mình. Bạn cũng sẽ học cách sử dụng các biến để biểu diễn dữ liệu trong chương trình của mình. chương trình.

Điều gì thực sự xảy ra khi bạn chạy hello_world.py

Hãy cùng xem xét kỹ hơn những gì Python thực hiện khi bạn chạy hello_world.py. Hóa ra, Python thực hiện khá nhiều công việc, ngay cả khi nó chỉ chạy một chương trình đơn giản:

hello_world.py
```
print("Xin chào thế giới Python!")
```

Khi bạn chạy mã này, bạn sẽ thấy kết quả sau:

```
Xin chào thế giới Python!
```

Khi bạn chạy tệp hello_world.py, phần mở rộng .py cho biết tệp là một chương trình Python. Trình soạn thảo của bạn sau đó chạy tệp thông qua trình thông dịch Python, trình thông dịch này sẽ đọc qua chương trình và xác định ý nghĩa của từng từ trong chương trình. Ví dụ: khi trình thông dịch thấy từ " print" theo sau là dấu ngoặc đơn, nó sẽ in ra màn hình bất cứ nội dung nào nằm trong dấu ngoặc đơn.

Khi bạn viết chương trình, trình soạn thảo sẽ làm nổi bật các phần khác nhau của chương trình theo những cách khác nhau. Ví dụ, nó nhận ra rằng print() là tên của một hàm và hiển thị từ đó bằng một màu. Nó nhận ra rằng "Hello Python world!" không phải là mã Python, và hiển thị cụm từ đó bằng một màu khác. Tính năng này được gọi là tô sáng cú pháp và khá hữu ích khi bạn bắt đầu viết chương trình của riêng mình.

Biến số

Hãy thử sử dụng một biến trong hello_world.py. Thêm một dòng mới vào đầu tệp và sửa đổi dòng thứ hai:

```
hello_world.py message = "Xin chào thế giới Python!"
in(tin nhắn)
```

Chạy chương trình này để xem điều gì xảy ra. Bạn sẽ thấy kết quả tương tự như trước đó:

```
Xin chào thế giới Python!
```

Chúng tôi đã thêm một biến có tên là message. Mỗi biến được kết nối với một giá trị, là thông tin liên quan đến biến đó. Trong trường hợp này, giá trị là văn bản "Hello Python world!" .

Việc thêm biến sẽ khiến trình thông dịch Python phải làm việc nhiều hơn một chút. Khi xử lý dòng đầu tiên, nó sẽ liên kết biến message với văn bản "Hello Python world!" . Khi đến dòng thứ hai, nó sẽ in giá trị được liên kết với message ra màn hình.

Hãy mở rộng chương trình này bằng cách sửa đổi hello_world.py để in ra thông điệp thứ hai. Thêm một dòng trống vào hello_world.py, rồi thêm hai dòng mã mới:

```
message = "Xin chào thế giới Python!"
in(tin nhắn)

message = "Xin chào thế giới Python Crash Course!"
in(tin nhắn)
```

Bây giờ khi bạn chạy hello_world.py, bạn sẽ thấy hai dòng đầu ra:

```
Xin chào thế giới Python!
Xin chào thế giới Python Crash Course!
```

Bạn có thể thay đổi giá trị của biến trong chương trình bất kỳ lúc nào và Python sẽ luôn theo dõi giá trị hiện tại của biến đó.

Đặt tên và sử dụng biến

Khi sử dụng biến trong Python, bạn cần tuân thủ một số quy tắc và hướng dẫn. Việc vi phạm một số quy tắc này sẽ gây ra lỗi; các hướng dẫn khác chỉ giúp bạn viết mã dễ đọc và dễ hiểu hơn. Hãy nhớ những quy tắc sau khi làm việc với biến:

- Tên biến chỉ có thể chứa chữ cái, số và dấu gạch dưới.
  Chúng có thể bắt đầu bằng một chữ cái hoặc dấu gạch dưới, nhưng không phải bằng một con số. Ví dụ, bạn có thể gọi biến message_1 nhưng không thể gọi biến 1_message.
- Không được phép sử dụng khoảng trắng trong tên biến, nhưng có thể sử dụng dấu gạch dưới để phân tách các từ trong tên biến. Ví dụ, greeting_message hoạt động nhưng greeting_message sẽ gây ra lỗi.
- Tránh sử dụng từ khóa Python và tên hàm làm tên biến.
  Ví dụ, không sử dụng từ print làm tên biến; Python đã dành riêng từ này cho một mục đích lập trình cụ thể. (Xem “Từ khóa Python và Hàm tích hợp” ở trang 466.)
- Tên biến nên ngắn gọn nhưng phải mô tả được nội dung. Ví dụ: name tốt hơn n, student_name tốt hơn s_n, và name_length tốt hơn length_of_persons_name.
- Cẩn thận khi sử dụng chữ l thường và chữ O hoa vì chúng có thể bị nhầm lẫn với số 1 và số 0.

Bạn có thể cần phải luyện tập để học cách đặt tên biến hay, đặc biệt là khi chương trình của bạn trở nên thú vị và phức tạp hơn. Khi bạn viết nhiều chương trình hơn và bắt đầu đọc mã của người khác, bạn sẽ giỏi hơn trong việc đưa ra những cái tên có ý nghĩa.

LƯU Ý: Các biến Python bạn đang sử dụng tại thời điểm này phải được viết thường. Bạn sẽ không gặp lỗi nếu sử dụng chữ in hoa, nhưng chữ in hoa trong tên biến có ý nghĩa đặc biệt mà chúng ta sẽ thảo luận trong các chương sau.

Tránh lỗi tên khi sử dụng biến

Mọi lập trình viên đều mắc lỗi và hầu hết đều mắc lỗi hàng ngày. Mặc dù lập trình viên giỏi có thể mắc lỗi, nhưng họ cũng biết cách xử lý những lỗi đó một cách hiệu quả. Hãy cùng xem xét một lỗi bạn có thể mắc phải ngay từ đầu và tìm hiểu cách khắc phục.

Chúng ta sẽ viết một đoạn mã cố ý tạo ra lỗi. Hãy nhập đoạn mã sau, bao gồm cả từ sai chính tả "mesage", được in đậm:

```
message = "Xin chào độc giả của Python Crash Course!"
in(tin nhắn)
```

Khi xảy ra lỗi trong chương trình, trình thông dịch Python sẽ cố gắng hết sức để giúp bạn tìm ra vấn đề. Trình thông dịch sẽ cung cấp một bản ghi theo dõi khi chương trình không chạy thành công. Bản ghi theo dõi là bản ghi về vị trí trình thông dịch gặp sự cố khi cố gắng thực thi mã của bạn.

Sau đây là một ví dụ về chức năng theo dõi mà Python cung cấp sau khi bạn vô tình viết sai tên biến:

```
Theo dõi (cuộc gọi gần đây nhất cuối cùng):
1 Tệp "hello_world.py", dòng 2, trong <module>
2 bản in (tin nhắn)
              ^^^^^^^

3 NameError: tên 'message' chưa được xác định. Ý bạn là: 'message' phải không?
```

Đầu ra báo cáo rằng có lỗi xảy ra ở dòng 2 của tệp hello_world.py 1. Trình thông dịch hiển thị dòng này 2 để giúp chúng ta phát hiện lỗi nhanh chóng và cho chúng ta biết loại lỗi nào mà nó tìm thấy 3. Trong trường hợp này, nó tìm thấy lỗi tên và báo cáo rằng biến được in, mesage, vẫn chưa được xác định. Python không thể xác định tên biến được cung cấp. Lỗi tên thường có nghĩa là chúng ta quên đặt giá trị cho biến trước khi sử dụng, hoặc nhập sai chính tả khi đặt tên biến. Nếu Python tìm thấy một tên biến tương tự với tên mà nó không nhận ra, nó sẽ hỏi xem đó có phải là tên bạn muốn sử dụng không.

Trong ví dụ này, chúng ta đã bỏ chữ s trong tên biến message ở dòng thứ hai. Trình thông dịch Python không kiểm tra chính tả mã của bạn, nhưng nó đảm bảo tên biến được viết một cách nhất quán. Ví dụ, hãy xem điều gì xảy ra khi chúng ta viết sai chữ message trong dòng định nghĩa biến:

```
message = "Xin chào độc giả của Python Crash Course!"
in(tin nhắn)
```

Trong trường hợp này, chương trình chạy thành công!

```
Xin chào độc giả của Python Crash Course!
```

Tên biến khớp nhau, nên Python không gặp vấn đề gì. Ngôn ngữ lập trình rất nghiêm ngặt, nhưng chúng không phân biệt đúng sai. Do đó, bạn không cần phải cân nhắc các quy tắc chính tả và ngữ pháp tiếng Anh khi cố gắng tạo tên biến và viết mã.

Nhiều lỗi lập trình chỉ là những lỗi đánh máy đơn giản, chỉ một ký tự trong một dòng chương trình. Nếu bạn thấy mình mất nhiều thời gian để tìm kiếm một trong những lỗi này, hãy biết rằng bạn không phải là ngoại lệ. Nhiều lập trình viên giàu kinh nghiệm và tài năng đã dành hàng giờ để săn lùng những lỗi nhỏ nhặt như thế này. Hãy cố gắng mỉm cười và tiếp tục, vì biết rằng nó sẽ xảy ra thường xuyên trong suốt cuộc đời lập trình của bạn.

Biến là nhãn

Biến thường được mô tả như những hộp chứa giá trị. Ý tưởng này có thể hữu ích trong vài lần đầu sử dụng biến, nhưng không phải là cách chính xác để mô tả cách biến được biểu diễn nội bộ trong Python. Tốt hơn hết là nên coi biến như những nhãn mà bạn có thể gán cho các giá trị. Bạn cũng có thể nói rằng một biến tham chiếu đến một giá trị nhất định.

Sự khác biệt này có lẽ sẽ không quan trọng lắm trong các chương trình ban đầu của bạn, nhưng việc học sớm vẫn đáng giá hơn là muộn. Đến một lúc nào đó, bạn sẽ thấy một biến hoạt động bất thường, và việc hiểu rõ cách thức hoạt động của biến sẽ giúp bạn xác định điều gì đang xảy ra trong mã của mình.

LƯU Ý: ách tốt nhất để hiểu các khái niệm lập trình mới là thử áp dụng chúng vào chương trình của bạn. Nếu bạn gặp khó khăn khi làm bài tập trong sách này, hãy thử làm việc khác một lúc. Nếu vẫn còn khó khăn, hãy xem lại phần liên quan của chương đó. Nếu bạn vẫn cần trợ giúp, hãy xem các gợi ý trong Phụ lục C.

HÃY TỰ THỬ

Viết một chương trình riêng để thực hiện từng bài tập này. Lưu mỗi chương trình với tên tệp tuân theo quy ước chuẩn của Python, sử dụng chữ thường và dấu gạch dưới, chẳng hạn như simple_message.py và simple_messages.py.

2-1. Thông báo đơn giản: Gán một thông báo cho một biến, sau đó in thông báo đó tin nhắn.

2-2. Tin nhắn đơn giản: Gán một tin nhắn cho một biến và in tin nhắn đó. Sau đó thay đổi giá trị của biến thành một thông báo mới và in thông báo mới tin nhắn.

Dây đàn

Vì hầu hết các chương trình đều định nghĩa và thu thập một số loại dữ liệu, sau đó sử dụng chúng để làm những việc hữu ích, nên việc phân loại các loại dữ liệu khác nhau rất hữu ích. Kiểu dữ liệu đầu tiên chúng ta sẽ xem xét là chuỗi. Thoạt nhìn, chuỗi khá đơn giản, nhưng bạn có thể sử dụng chúng theo nhiều cách khác nhau.

Chuỗi là một chuỗi ký tự. Bất kỳ ký tự nào nằm trong dấu ngoặc kép đều được coi là một chuỗi trong Python, và bạn có thể sử dụng dấu ngoặc đơn hoặc dấu ngoặc kép quanh chuỗi như sau:

```
"Đây là một sợi dây."
'Đây cũng là một sợi dây.'
```

Tính linh hoạt này cho phép bạn sử dụng dấu ngoặc kép và dấu nháy đơn trong chuỗi của mình:

```
'Tôi nói với bạn tôi, "Python là ngôn ngữ yêu thích của tôi!"
"Ngôn ngữ 'Python' được đặt theo tên của Monty Python, chứ không phải tên của loài rắn."
"Một trong những điểm mạnh của Python là cộng đồng đa dạng và hỗ trợ lẫn nhau."
```

Hãy cùng khám phá một số cách bạn có thể sử dụng chuỗi.

Thay đổi chữ hoa và chữ thường trong một chuỗi bằng các phương thức

Một trong những thao tác đơn giản nhất bạn có thể thực hiện với chuỗi ký tự là đổi chữ hoa/thường của các từ trong chuỗi. Hãy xem đoạn mã sau và thử xác định điều gì đang xảy ra:

```
name.py name = "ada lovelace"
in(tên.tiêu đề())
```

Lưu tệp này dưới dạng name.py và chạy nó. Bạn sẽ thấy kết quả sau:

```
Ada Lovelace
```

Trong ví dụ này, tên biến tham chiếu đến chuỗi ký tự thường "ada lovelace". Phương thức title() xuất hiện sau biến trong lệnh gọi print() . Một phương thức là một hành động mà Python có thể thực hiện trên một phần dữ liệu. Dấu chấm (.) sau tên trong name.title() cho Python biết phải để phương thức title() tác động lên tên biến . Mỗi phương thức được theo sau bởi một cặp dấu ngoặc đơn, vì các phương thức thường cần thêm thông tin để thực hiện công việc. Thông tin đó được cung cấp bên trong dấu ngoặc đơn. Hàm title() không cần thêm thông tin nào, vì vậy các dấu ngoặc đơn của nó được để trống.

Phương thức title() chuyển đổi mỗi từ thành chữ hoa đầu tiên, trong đó mỗi từ bắt đầu bằng chữ in hoa. Điều này rất hữu ích vì bạn thường muốn coi tên như một thông tin. Ví dụ: bạn có thể muốn chương trình của mình nhận dạng các giá trị đầu vào Ada, ADA và ada là cùng một tên, và hiển thị tất cả chúng dưới dạng Ada.

Ngoài ra còn có một số phương pháp hữu ích khác để giải quyết trường hợp này. Ví dụ, bạn có thể thay đổi một chuỗi thành toàn bộ chữ hoa hoặc toàn bộ chữ thường như thế này:

```
tên = "Ada Lovelace"
in(tên.upper())
in(tên.thấp hơn())
```

Nó sẽ hiển thị nội dung sau:

```
ADA LOVELACE
ada lovelace
```

Phương thức lower() đặc biệt hữu ích cho việc lưu trữ dữ liệu. Thông thường, bạn sẽ không muốn tin tưởng vào cách viết hoa mà người dùng cung cấp, vì vậy bạn sẽ chuyển đổi chuỗi sang chữ thường trước khi lưu trữ. Sau đó, khi muốn hiển thị thông tin, bạn sẽ sử dụng chữ hoa/thường phù hợp nhất cho mỗi chuỗi.

Sử dụng Biến trong Chuỗi

Trong một số trường hợp, bạn sẽ muốn sử dụng giá trị của một biến bên trong một chuỗi. Ví dụ, bạn có thể muốn sử dụng hai biến để biểu diễn tên và

tương ứng là họ, sau đó kết hợp các giá trị đó để hiển thị tên đầy đủ của ai đó:

full_name.py

```
first_name = "ada"
họ = "lovelace"
tên_đầy_đủ = f"{tên} {họ}"
in(tên_đầy_đủ)
```

Để chèn giá trị của một biến vào chuỗi, hãy đặt chữ f ngay trước dấu ngoặc kép mở đầu 1. Đặt dấu ngoặc kép quanh tên hoặc các tên của bất kỳ biến nào bạn muốn sử dụng bên trong chuỗi. Python sẽ thay thế mỗi biến bằng giá trị của nó khi chuỗi được hiển thị.

Những chuỗi này được gọi là chuỗi f. Chữ f là viết tắt của format (định dạng), vì Python định dạng chuỗi bằng cách thay thế tên của bất kỳ biến nào trong dấu ngoặc nhọn bằng giá trị của nó. Đầu ra của đoạn mã trên là:

```
ada lovelace
```

Bạn có thể làm được rất nhiều việc với chuỗi f. Ví dụ, bạn có thể sử dụng chuỗi f để soạn thảo các thông điệp hoàn chỉnh bằng cách sử dụng thông tin liên quan đến một biến, như minh họa ở đây:

```
tên_đầu_tiên = "ada"
họ = "lovelace"
tên_đầy_đủ = f"{tên} {họ}"
print(f"Xin chào, {full_name.title()}!")
```

Tên đầy đủ được sử dụng trong câu chào người dùng 1, và phương thức title() sẽ chuyển tên thành chữ hoa đầu dòng. Đoạn mã này trả về một lời chào đơn giản nhưng được định dạng đẹp mắt:

```
Xin chào, Ada Lovelace!
```

Bạn cũng có thể sử dụng chuỗi f để soạn tin nhắn, sau đó gán toàn bộ tin nhắn cho một biến:

```
tên_đầu_tiên = "ada"
họ = "lovelace"
tên_đầy_đủ = f"{tên} {họ}"
tin nhắn = f"Xin chào, {full_name.title()}!"
bản in (tin nhắn)
```

Mã này cũng hiển thị thông báo Xin chào, Ada Lovelace!, nhưng bằng cách gán thông báo cho biến 1, chúng ta làm cho lệnh gọi print() cuối cùng trở nên đơn giản hơn nhiều 2.

Thêm khoảng trắng vào chuỗi bằng Tab hoặc dòng mới

Trong lập trình, khoảng trắng dùng để chỉ bất kỳ ký tự nào không in ra được, chẳng hạn như khoảng trắng, tab và ký hiệu cuối dòng. Bạn có thể sử dụng khoảng trắng để sắp xếp nội dung đầu ra sao cho người dùng dễ đọc hơn.

Để thêm tab vào văn bản của bạn, hãy sử dụng tổ hợp ký tự \t:

```
>>> in("Python")
Trăn
>>> in("\tPython")
    Trăn
```

Để thêm dòng mới vào chuỗi, hãy sử dụng tổ hợp ký tự \n:

```
>>> print("Ngôn ngữ:\nPython\nC\nJavaScript")
Ngôn ngữ:
Trăn
C
JavaScript
```

Bạn cũng có thể kết hợp tab và xuống dòng trong một chuỗi. Chuỗi "\n\t" yêu cầu Python di chuyển đến một dòng mới và bắt đầu dòng tiếp theo bằng phím tab. Ví dụ sau đây cho thấy cách bạn có thể sử dụng chuỗi một dòng để tạo ra bốn dòng đầu ra:

```
>>> print("Ngôn ngữ:\n\tPython\n\tC\n\tJavaScript")
Ngôn ngữ:
    Trăn
    C
    JavaScript
```

Các dòng mới và tab sẽ rất hữu ích trong hai chương tiếp theo, khi bạn bắt đầu tạo ra nhiều dòng đầu ra chỉ từ một vài dòng mã.

Xóa khoảng trắng

Khoảng trắng thừa có thể gây nhầm lẫn trong chương trình của bạn. Đối với lập trình viên, 'python' và 'python' trông khá giống nhau. Nhưng đối với chương trình, chúng là hai chuỗi khác nhau. Python phát hiện khoảng trắng thừa trong 'python' và coi đó là quan trọng trừ khi bạn yêu cầu nó làm khác đi.

Điều quan trọng là phải cân nhắc đến khoảng trắng, vì thường bạn sẽ muốn so sánh hai chuỗi để xác định xem chúng có giống nhau hay không. Ví dụ, một trường hợp quan trọng có thể liên quan đến việc kiểm tra tên người dùng của mọi người khi họ đăng nhập vào một trang web. Khoảng trắng thừa cũng có thể gây nhầm lẫn trong những tình huống đơn giản hơn nhiều. May mắn thay, Python giúp dễ dàng loại bỏ khoảng trắng thừa khỏi dữ liệu mà người dùng nhập vào.

Python có thể tìm kiếm khoảng trắng thừa ở bên phải và bên trái của một chuỗi. Để đảm bảo không có khoảng trắng nào tồn tại ở bên phải của chuỗi, hãy sử dụng phương thức rstrip() :

```
1 >>> ngôn ngữ yêu thích = 'python '
2 >>> ngôn ngữ yêu thích
  'trăn'
3 >>> ngôn ngữ yêu thích.rstrip()
  'trăn'
4 >>> ngôn ngữ yêu thích
  'trăn'
```

Giá trị liên quan đến favorite_language 1 chứa khoảng trắng thừa ở cuối chuỗi. Khi bạn yêu cầu Python nhập giá trị này trong một phiên terminal, bạn có thể thấy khoảng trắng ở cuối giá trị 2. Khi phương thức rstrip() tác động lên biến favorite_language 3, khoảng trắng thừa này sẽ bị xóa. Tuy nhiên, nó chỉ bị xóa tạm thời. Nếu bạn yêu cầu giá trị của favorite_language một lần nữa, chuỗi sẽ trông giống như lúc nhập, bao gồm cả khoảng trắng thừa 4.

Để xóa khoảng trắng khỏi chuỗi vĩnh viễn, bạn phải liên kết giá trị đã xóa với tên biến:

```
>>> ngôn ngữ yêu thích = 'python '
1 >>> ngôn ngữ yêu thích = ngôn ngữ yêu thích.rstrip()
>>> ngôn ngữ yêu thích
'trăn'
```

Để xóa khoảng trắng khỏi chuỗi, bạn xóa khoảng trắng ở bên phải chuỗi và sau đó liên kết giá trị mới này với biến ban đầu 1. Việc thay đổi giá trị của biến thường được thực hiện trong lập trình. Đây là cách giá trị của biến có thể được cập nhật khi chương trình được thực thi hoặc để phản hồi đầu vào của người dùng.

Bạn cũng có thể xóa khoảng trắng ở phía bên trái của chuỗi bằng cách sử dụng phương thức lstrip() hoặc từ cả hai phía cùng lúc bằng cách sử dụng strip():

```
1 >>> ngôn ngữ yêu thích = python 2
>>> ngôn ngữ yêu thích.rstrip()
'trăn'
3 >>> favorite_language.lstrip()
'trăn'
4 >>> ngôn ngữ yêu thích.strip()
'trăn'
```

Trong ví dụ này, chúng ta bắt đầu với một giá trị có khoảng trắng ở đầu và cuối là 1. Sau đó, chúng ta xóa khoảng trắng thừa ở vế phải 2, vế trái 3 và cả hai vế 4. Việc thử nghiệm với các hàm strip-ping này có thể giúp bạn làm quen với việc thao tác với chuỗi. Trong thực tế, các hàm strip-ping này thường được sử dụng để dọn dẹp dữ liệu đầu vào của người dùng trước khi lưu trữ trong chương trình.

Xóa tiền tố

Khi làm việc với chuỗi, một nhiệm vụ phổ biến khác là xóa tiền tố.

Hãy xem xét một URL có tiền tố chung là https://. Chúng ta muốn xóa tiền tố này để có thể tập trung vào phần URL mà người dùng cần nhập vào thanh địa chỉ. Cách thực hiện như sau:

```
>>> nostarch_url = 'https://nostarch.com'
>>> nostarch_url.removeprefix('https://')
'nostarch.com'
```

Nhập tên biến, theo sau là dấu chấm, rồi nhập phương thức removeprefix(). Bên trong dấu ngoặc đơn, nhập tiền tố bạn muốn xóa khỏi chuỗi gốc.

Giống như các phương thức xóa khoảng trắng, removeprefix() giữ nguyên chuỗi gốc. Nếu bạn muốn giữ nguyên giá trị mới với phần tiền tố đã được xóa, hãy gán lại giá trị đó cho biến gốc hoặc gán nó cho một biến mới:

```
>>> simple_url = nostarch_url.removeprefix('https://')
```

Khi bạn nhìn thấy một URL trong thanh địa chỉ và phần https:// không được hiển thị, trình duyệt có thể đang sử dụng một phương pháp như removeprefix() đằng sau cảnh.

Tránh lỗi cú pháp với chuỗi

Một loại lỗi mà bạn có thể thường xuyên thấy là lỗi cú pháp.
Lỗi cú pháp xảy ra khi Python không nhận dạng được một phần trong chương trình của bạn là mã Python hợp lệ. Ví dụ: nếu bạn sử dụng dấu nháy đơn trong dấu nháy đơn, bạn sẽ gặp lỗi. Điều này xảy ra vì Python diễn giải mọi thứ giữa dấu nháy đơn đầu tiên và dấu nháy đơn thành một chuỗi. Sau đó, nó cố gắng diễn giải phần còn lại của văn bản thành mã Python, mà

gây ra lỗi.

Sau đây là cách sử dụng dấu ngoặc đơn và dấu ngoặc kép đúng cách. Lưu chương trình này dưới dạng apostrophe.py và chạy nó:

```
apostrophe.py message = "Một trong những điểm mạnh của Python là cộng đồng đa dạng của nó."
in(tin nhắn)
```

Dấu nháy đơn xuất hiện bên trong một cặp dấu ngoặc kép, do đó trình thông dịch Python không gặp khó khăn khi đọc chuỗi chính xác:

```
Một trong những điểm mạnh của Python là cộng đồng đa dạng.
```

Tuy nhiên, nếu bạn sử dụng dấu ngoặc đơn, Python không thể xác định được vị trí của chuỗi nên kết thúc:

```
message = 'Một trong những điểm mạnh của Python là cộng đồng đa dạng của nó.'
in(tin nhắn)
```

Bạn sẽ thấy kết quả sau:

```
  Tệp "apostrophe.py", dòng 1
    message = 'Một trong những điểm mạnh của Python là cộng đồng đa dạng của nó.'
                                                                              1 ^
SyntaxError: chuỗi ký tự chưa kết thúc (được phát hiện ở dòng 1)
```

Trong kết quả đầu ra, bạn có thể thấy lỗi xảy ra ngay sau lần phát đơn cuối cùng trích dẫn 1. Lỗi cú pháp này chỉ ra rằng trình thông dịch không nhận ra

Một phần nào đó trong mã là mã Python hợp lệ, và nó cho rằng vấn đề có thể nằm ở một chuỗi không được trích dẫn chính xác. Lỗi có thể đến từ nhiều nguồn khác nhau, và tôi sẽ chỉ ra một số lỗi phổ biến khi chúng phát sinh. Bạn có thể thường xuyên gặp lỗi cú pháp khi học cách viết mã Python đúng. Lỗi cú pháp cũng là loại lỗi ít cụ thể nhất, vì vậy việc xác định và sửa chúng có thể khó khăn và gây bực bội. Nếu bạn gặp phải một lỗi đặc biệt khó sửa, hãy xem các gợi ý trong Phụ lục C.

LƯU Ý: Tính năng tô sáng cú pháp của trình soạn thảo sẽ giúp bạn nhanh chóng phát hiện một số lỗi cú pháp khi viết chương trình. Nếu bạn thấy mã Python được tô sáng như thể đó là tiếng Anh hoặc tiếng Anh được tô sáng như thể đó là mã Python, có thể bạn đã đặt dấu ngoặc kép không khớp ở đâu đó trong tệp.

HÃY TỰ THỬ

Lưu từng bài tập sau đây dưới dạng một tệp riêng biệt, với tên như tên _cases.py. Nếu bạn gặp khó khăn, hãy nghỉ ngơi hoặc xem các gợi ý trong Phụ lục C.

2-3. Tin nhắn cá nhân: Sử dụng một biến để biểu diễn tên của một người và in ra tin nhắn cho người đó. Tin nhắn của bạn nên đơn giản, chẳng hạn như: "Chào Eric, hôm nay bạn có muốn học Python không?"

2-4. Tên viết hoa: Sử dụng một biến để biểu diễn tên của một người, sau đó in tên của người đó bằng chữ thường, chữ hoa và chữ thường chức danh.

2-5. Câu nói nổi tiếng: Tìm một câu nói của một người nổi tiếng mà bạn ngưỡng mộ. In câu nói đó và tên tác giả. Kết quả đầu ra sẽ trông giống như sau, bao gồm cả dấu ngoặc kép:

Albert Einstein đã từng nói: “Người chưa bao giờ mắc sai lầm thì chưa bao giờ thử bất cứ điều gì mới mẻ”.

2-6. Câu nói nổi tiếng 2: Lặp lại Bài tập 2-5, nhưng lần này, hãy biểu diễn tên của người nổi tiếng bằng một biến có tên là famous_person. Sau đó, soạn tin nhắn của bạn và biểu diễn nó bằng một biến mới có tên là message. In tin nhắn của bạn.

2-7. Tách tên: Sử dụng một biến để biểu diễn tên của một người và thêm một số ký tự khoảng trắng vào đầu và cuối tên.
Hãy đảm bảo bạn sử dụng mỗi tổ hợp ký tự "\t" và "\n" ít nhất một lần.
In tên một lần để hiển thị khoảng trắng xung quanh tên.
Sau đó in tên bằng cách sử dụng một trong ba hàm tách tên: lstrip(), rstrip() và strip().

2-8. Phần mở rộng tệp: Python có phương thức removesuffix() hoạt động chính xác như removeprefix(). Gán giá trị 'python_notes.txt' cho một biến có tên là filename. Sau đó, sử dụng phương thức removesuffix() để hiển thị tên tệp không có phần mở rộng, giống như một số trình duyệt tệp.

Số

Số được sử dụng khá thường xuyên trong lập trình để tính điểm trong trò chơi, biểu diễn dữ liệu trong hình ảnh trực quan, lưu trữ thông tin trong ứng dụng web, v.v. Python xử lý số theo nhiều cách khác nhau, tùy thuộc vào cách chúng được sử dụng. Trước tiên, hãy xem cách Python quản lý số nguyên, vì chúng là loại số nguyên dễ sử dụng nhất.

Số nguyên

Bạn có thể cộng (+), trừ (-), nhân (*) và chia (/) số nguyên trong Python.

```
>>> 2 + 3
5
>>> 3 - 2
1
>>> 2 *  3
6
>>> 3 / 2
1,5
```

Trong phiên làm việc đầu cuối, Python chỉ trả về kết quả của phép toán. Python sử dụng hai ký hiệu nhân để biểu diễn số mũ:

```
>>> 3  ** 2
9
>>> 3   **  3
27
>>> 10 ** 6
1000000
```

Python cũng hỗ trợ thứ tự các phép toán, vì vậy bạn có thể sử dụng nhiều phép toán trong một biểu thức. Bạn cũng có thể sử dụng dấu ngoặc đơn để sửa đổi thứ tự các phép toán để Python có thể đánh giá biểu thức theo thứ tự bạn chỉ định. Ví dụ:

```
>>> 2 + 3*4
14
>>> (2 + 3) *    4
20
```

Khoảng cách trong các ví dụ này không ảnh hưởng đến cách Python đánh giá các biểu thức; nó chỉ giúp bạn nhanh chóng phát hiện các thao tác có mức độ ưu tiên khi bạn đọc qua mã.

Phao

Python gọi bất kỳ số nào có dấu thập phân là float. Thuật ngữ này được sử dụng trong hầu hết các ngôn ngữ lập trình và nó ám chỉ thực tế là dấu thập phân

có thể xuất hiện ở bất kỳ vị trí nào trong một số. Mọi ngôn ngữ lập trình đều phải được thiết kế cẩn thận để quản lý số thập phân một cách chính xác, sao cho các con số hoạt động phù hợp, bất kể dấu thập phân xuất hiện ở đâu.

Phần lớn, bạn có thể sử dụng float mà không cần lo lắng về cách chúng hoạt động. Chỉ cần nhập số bạn muốn sử dụng, và Python sẽ tự động thực hiện những gì bạn mong đợi:

```
>>> 0,1 + 0,1
0,2
>>> 0,2 + 0,2
0,4
>>> 2 * 0,1
0,2
>>> 2 * 0,2
0,4
```

Tuy nhiên, hãy lưu ý rằng đôi khi bạn có thể nhận được số lượng chữ số thập phân tùy ý trong câu trả lời của mình:

```
>>> 0,2 + 0,1
0,30000000000000004
>>> 3  * 0,1
0.30000000000000004
```

Điều này xảy ra ở mọi ngôn ngữ và không đáng lo ngại. Python cố gắng tìm cách biểu diễn kết quả chính xác nhất có thể, điều này đôi khi rất khó khăn vì máy tính phải biểu diễn số nội bộ. Tạm thời, hãy bỏ qua các chữ số thập phân thừa; bạn sẽ học cách xử lý các chữ số thừa khi cần trong các dự án ở Phần II.

Số nguyên và số thực

Khi bạn chia bất kỳ hai số nào, ngay cả khi chúng là số nguyên có kết quả là một số nguyên, bạn sẽ luôn nhận được một số thực:

```
>>> 4/2
2.0
```

Nếu bạn kết hợp một số nguyên và một số thực trong bất kỳ phép toán nào khác, bạn cũng sẽ nhận được một số thực:

```
>>> 1 + 2.0
3.0
>>> 2 *  3.0
6.0
>>> 3.0  ** 2
9.0
```

Python mặc định sử dụng float trong bất kỳ hoạt động nào sử dụng float, ngay cả khi đầu ra là một số nguyên.

Dấu gạch dưới trong số

Khi viết các số dài, bạn có thể nhóm các chữ số bằng dấu gạch dưới để làm cho các số lớn dễ đọc hơn:

```
>>> tuổi_vũ_trụ = 14_000_000_000
```

Khi bạn in một số được xác định bằng dấu gạch dưới, Python chỉ in ra các chữ số:

```
>>> in(tuổi_vũ_trụ)
14000000000
```

Python bỏ qua dấu gạch dưới khi lưu trữ những giá trị như thế này. Ngay cả khi bạn không nhóm các chữ số thành ba, giá trị vẫn không bị ảnh hưởng. Đối với Python, `1000` giống với `1_000`, và `1_000` cũng giống với `10_00`. Tính năng này hoạt động với cả số nguyên và số thực.

Nhiều nhiệm vụ

Bạn có thể gán giá trị cho nhiều biến chỉ bằng một dòng mã. Điều này có thể giúp rút gọn chương trình và giúp chúng dễ đọc hơn; bạn sẽ sử dụng kỹ thuật này thường xuyên nhất khi khởi tạo một tập hợp số.

Ví dụ, đây là cách bạn có thể khởi tạo các biến x, y và z thành 0:

```
>>> x, y, z = 0, 0, 0
```

Bạn cần phân tách tên biến bằng dấu phẩy và làm tương tự với các giá trị, Python sẽ gán từng giá trị cho biến tương ứng.
Miễn là số lượng giá trị khớp với số lượng biến, Python sẽ khớp chúng một cách chính xác.

Hằng số

Hằng số là một biến có giá trị không đổi trong suốt vòng đời của một chương trình. Python không có sẵn các kiểu hằng số, nhưng các lập trình viên Python sử dụng toàn bộ chữ cái viết hoa để chỉ ra rằng một biến nên được coi là một hằng số và không bao giờ bị thay đổi:

```
MAX_CONNECTIONS = 5000
```

Khi bạn muốn coi một biến là hằng số trong mã của mình, hãy viết tên biến bằng chữ in hoa.

HÃY TỰ THỬ

2-9. Số tám: Viết các phép tính cộng, trừ, nhân và chia sao cho mỗi phép tính đều cho kết quả là số 8. Hãy đảm bảo đặt các phép tính của bạn trong các lệnh gọi print() để xem kết quả. Bạn nên tạo bốn dòng trông như thế này:

```
in(5+3)
```

Đầu ra của bạn phải có bốn dòng, với số 8 xuất hiện một lần trên mỗi dòng.

2-10. Con số yêu thích: Sử dụng một biến để biểu diễn con số yêu thích của bạn. Sau đó, sử dụng biến đó, tạo một thông báo hiển thị con số yêu thích của bạn. In thông báo đó.

Bình luận

Chú thích là một tính năng cực kỳ hữu ích trong hầu hết các ngôn ngữ lập trình. Mọi thứ bạn đã viết trong chương trình cho đến nay đều là mã Python. Khi chương trình của bạn trở nên dài hơn và phức tạp hơn, bạn nên thêm ghi chú vào chương trình để mô tả cách tiếp cận tổng thể của bạn đối với vấn đề bạn đang giải quyết. Ghi chú cho phép bạn viết ghi chú bằng ngôn ngữ nói của mình trong chương trình.

Bạn viết bình luận như thế nào?

Trong Python, dấu thăng (#) biểu thị một chú thích. Bất kỳ ký tự nào theo sau dấu thăng trong mã của bạn đều bị trình thông dịch Python bỏ qua. Ví dụ:

comment.py

```
# Xin chào mọi người.
print("Xin chào mọi người Python!")
```

Python bỏ qua dòng đầu tiên và thực thi dòng thứ hai.

```
Xin chào mọi người Python!
```

Bạn nên viết loại bình luận nào?

Lý do chính để viết chú thích là để giải thích mã của bạn được cho là sẽ làm gì và cách bạn làm cho nó hoạt động. Khi bạn đang trong quá trình thực hiện một dự án, bạn sẽ hiểu được cách tất cả các phần khớp với nhau.
Nhưng khi bạn quay lại một dự án sau một thời gian vắng bóng, bạn có thể sẽ có

Bạn có thể quên một số chi tiết. Bạn luôn có thể nghiên cứu mã của mình một lúc và tìm ra cách các phần đoạn hoạt động, nhưng việc viết chú thích tốt có thể giúp bạn tiết kiệm thời gian bằng cách tóm tắt rõ ràng phương pháp tiếp cận tổng thể của bạn.

Nếu bạn muốn trở thành một lập trình viên chuyên nghiệp hoặc cộng tác với các lập trình viên khác, bạn nên viết những bình luận có ý nghĩa. Ngày nay, hầu hết các phần mềm đều được viết theo hình thức cộng tác, dù là bởi một nhóm nhân viên trong cùng một công ty hay một nhóm người cùng làm việc trong một dự án nguồn mở. Các lập trình viên lành nghề mong muốn thấy chú thích trong mã, vì vậy tốt nhất là bạn nên bắt đầu thêm chú thích mô tả vào chương trình ngay bây giờ. Viết chú thích rõ ràng, súc tích trong mã là một trong những thói quen có lợi nhất mà bạn có thể hình thành khi mới vào nghề lập trình.

Khi bạn quyết định có nên viết chú thích hay không, hãy tự hỏi liệu bạn có phải cân nhắc nhiều phương án trước khi tìm ra một cách hợp lý để làm cho chương trình hoạt động hay không; nếu có, hãy viết chú thích về giải pháp của bạn. Việc xóa các chú thích thừa sau này dễ hơn nhiều so với việc quay lại và viết chú thích cho một chương trình có ít chú thích. Từ giờ trở đi, tôi sẽ sử dụng chú thích trong các ví dụ xuyên suốt cuốn sách này để giúp giải thích các đoạn mã.

HÃY TỰ THỬ

2-11. Thêm chú thích: Chọn hai chương trình bạn đã viết và thêm ít nhất một chú thích cho mỗi chương trình. Nếu bạn không có gì cụ thể để viết vì chương trình của bạn quá đơn giản, chỉ cần thêm tên và ngày hiện tại vào đầu mỗi tệp chương trình. Sau đó, hãy viết một câu mô tả chức năng của chương trình.

Thiền của Python

Các lập trình viên Python giàu kinh nghiệm sẽ khuyến khích bạn tránh sự phức tạp và hướng đến sự đơn giản bất cứ khi nào có thể. Triết lý của cộng đồng Python được thể hiện trong cuốn "The Zen of Python" của Tim Peters. Bạn có thể truy cập bộ nguyên tắc ngắn gọn này để viết mã Python tốt bằng cách nhập import this vào trình thông dịch của bạn. Tôi sẽ không trình bày lại toàn bộ "Thiền Python" ở đây, nhưng tôi sẽ chia sẻ một vài dòng để giúp bạn hiểu tại sao chúng lại quan trọng với bạn, một lập trình viên Python mới bắt đầu.

```
>>> nhập cái này
Thiền của Python, của Tim Peters
Đẹp thì tốt hơn xấu.
```

Các lập trình viên Python chấp nhận quan niệm rằng mã nguồn có thể đẹp và tinh tế. Trong lập trình, con người giải quyết vấn đề. Các lập trình viên luôn coi trọng những giải pháp được thiết kế tốt, hiệu quả và thậm chí đẹp mắt cho các vấn đề. Khi bạn tìm hiểu thêm về Python và sử dụng nó để viết nhiều mã nguồn hơn,

Một ngày nào đó, có thể ai đó nhìn qua vai bạn và nói, "Ồ, đó là một đoạn mã đẹp!"

Đơn giản thì tốt hơn phức tạp.

Nếu bạn có thể lựa chọn giữa một giải pháp đơn giản và một giải pháp phức tạp, và cả hai đều hiệu quả, hãy chọn giải pháp đơn giản. Mã của bạn sẽ dễ bảo trì hơn, và bạn cũng như những người khác sẽ dễ dàng xây dựng dựa trên mã đó hơn sau này.

Phức tạp thì tốt hơn là phức tạp.

Cuộc sống thực tế rất phức tạp, và đôi khi một giải pháp đơn giản cho một vấn đề lại không thể đạt được. Trong trường hợp đó, hãy áp dụng giải pháp đơn giản nhất và hiệu quả nhất.

Tính dễ đọc rất quan trọng.

Ngay cả khi mã của bạn phức tạp, hãy cố gắng làm cho nó dễ đọc. Khi bạn
Khi làm việc trên một dự án liên quan đến mã hóa phức tạp, hãy tập trung vào việc viết các bình luận có thông tin cho mã đó.

Phải có một cách rõ ràng và tốt nhất là chỉ có một cách để thực hiện.

Nếu hai lập trình viên Python được yêu cầu giải quyết cùng một vấn đề, họ nên đưa ra những giải pháp tương thích. Điều này không có nghĩa là không có chỗ cho sự sáng tạo trong lập trình. Ngược lại, vẫn còn rất nhiều chỗ cho sự sáng tạo! Tuy nhiên, phần lớn lập trình bao gồm việc sử dụng các phương pháp tiếp cận nhỏ, phổ biến cho các tình huống đơn giản trong một dự án lớn hơn, sáng tạo hơn. Các nguyên lý cơ bản trong chương trình của bạn phải dễ hiểu đối với các lập trình viên Python khác.

Bây giờ còn hơn không.

Bạn có thể dành cả đời để học tất cả những điều phức tạp của Python và lập trình nói chung, nhưng rồi bạn sẽ chẳng bao giờ hoàn thành được dự án nào. Đừng cố gắng viết code hoàn hảo; hãy viết code chạy được, rồi sau đó quyết định xem có nên cải thiện code cho dự án đó hay chuyển sang một dự án mới.

Khi bạn tiếp tục chương tiếp theo và bắt đầu tìm hiểu sâu hơn về các chủ đề, hãy cố gắng ghi nhớ triết lý về sự đơn giản và rõ ràng này.
Các lập trình viên giàu kinh nghiệm sẽ tôn trọng mã của bạn hơn và sẽ vui lòng phản hồi và hợp tác với bạn trong các dự án thú vị.

HÃY TỰ THỬ

2-12. Thiền của Python: Nhập lệnh này vào phiên bản Python terminal và lướt qua các nguyên tắc bổ sung.

Bản tóm tắt

Trong chương này, bạn đã học cách làm việc với các biến. Bạn đã học cách sử dụng tên biến mang tính mô tả và giải quyết lỗi tên cũng như lỗi cú pháp khi chúng phát sinh. Bạn đã học về chuỗi là gì và cách hiển thị chúng bằng cách sử dụng chữ thường, chữ hoa và chữ thường. Bạn đã bắt đầu sử dụng khoảng trắng để sắp xếp đầu ra gọn gàng, và bạn đã học cách loại bỏ các phần tử không cần thiết khỏi một chuỗi. Bạn đã bắt đầu làm việc với số nguyên và số thực, và bạn đã học một số cách để làm việc với dữ liệu số. Bạn cũng đã học cách viết chú thích giải thích để giúp bạn và những người khác dễ đọc mã hơn.
Cuối cùng, bạn đọc về triết lý giữ cho mã của mình càng đơn giản càng tốt, bất cứ khi nào có thể.

Trong Chương 3, bạn sẽ học cách lưu trữ các tập hợp thông tin trong các cấu trúc dữ liệu gọi là danh sách. Bạn cũng sẽ học cách xử lý danh sách, thao tác với bất kỳ thông tin nào trong danh sách đó.