**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:

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

print(fred)

**Kết quả:**

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)

**Kết quả:**

What is pink and fluffy? Pink fluff!!

Tuy nhiên, nếu bạn cố nhập nhiều dòng văn bản trong chuỗi bằng dấu nháy đơn (') hoặc nháy kép (") hoặc bắt đầu với một loại dấu và kết thúc bằng loại dấu khác, bạn sẽ gặp lỗi.

Ví dụ:

fred = "How do dinosaurs pay their bills?

**Kết quả:**

SyntaxError: EOL while scanning string literal

Lỗi này xảy ra vì Python không tìm thấy dấu ngoặc kép đóng chuỗi.

**Giải pháp:** Để sử dụng nhiều dòng trong chuỗi (được gọi là **chuỗi nhiều dòng**), bạn dùng ba dấu nháy đơn ('''), như sau:

fred = '''How do dinosaurs pay their bills?

With tyrannosaurus checks!'''

print(fred)

**Kết quả:**

How do dinosaurs pay their bills?

With tyrannosaurus checks!

**Xử lý vấn đề với chuỗi**

Xét ví dụ sau đây:

silly\_string = 'He said, "Aren't can't shouldn't wouldn't."'

**Kết quả:**

SyntaxError: invalid syntax

Lỗi này xảy ra vì Python không hiểu cách sử dụng dấu nháy đơn và kép lẫn lộn.

**Cách khắc phục:**

1. Dùng chuỗi nhiều dòng ('''):
2. silly\_string = '''He said, "Aren't can't shouldn't wouldn't."'''
3. Dùng ký tự thoát (\):
4. silly\_string = 'He said, "Aren\'t can\'t shouldn\'t wouldn\'t."'

**Kết quả:**

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

**Nhúng giá trị vào chuỗi**

Bạn có thể chèn giá trị vào chuỗi bằng %s, ví dụ:

myscore = 1000

message = 'I scored %s points'

print(message % myscore)

**Kết quả:**

I scored 1000 points

Bạn cũng có thể nhúng nhiều giá trị:

nums = 'What did the number %s say to the number %s? Nice belt!!'

print(nums % (0, 8))

**Kết quả:**

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

**Nhân chuỗi**

Python cho phép nhân chuỗi với số nguyên:

print(10 \* 'a')

**Kết quả:**

aaaaaaaaaa

**Ví dụ:**

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')

**Chuỗi, Danh sách, Bộ dữ liệu, và Bản đồ**

31  
[www.it-ebooks.info](http://www.it-ebooks.info/)

**Lưu ý**: Từ giờ, khi bạn thấy dòng **Save As: somefilename.py** ở trên đoạn mã, điều đó có nghĩa bạn cần chọn **File → New Window**, nhập mã vào cửa sổ mới hiện ra, rồi lưu nó lại như ví dụ trên.

Trong dòng đầu tiên của ví dụ này, chúng ta tạo biến spaces bằng cách nhân 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 bên phải của shell. Kết quả của các câu lệnh print như sau:

Ngoài việc sử dụng phép nhân để căn chỉnh, chúng ta cũng có thể sử dụng nó để lấp đầy màn hình với các thông điệp gây khó chịu. Thử ví dụ này:

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

**Danh sách mạnh mẽ hơn chuỗi**

“Spider legs, toe of frog, eye of newt, bat wing, slug butter, and snake dandruff” không hoàn toàn là một danh sách mua sắm thông thường (trừ khi bạn là phù thủy), nhưng chúng ta sẽ sử dụng nó như một ví dụ đầu tiên để tìm hiểu sự khác biệt giữa chuỗi và danh sách.  
32 Chapter 3  
[www.it-ebooks.info](http://www.it-ebooks.info/)

Chúng ta có thể lưu danh sách các mục này vào biến wizard\_list bằng 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

Nhưng chúng ta cũng có thể tạo một danh sách, một loại đối tượng Python kỳ diệu có thể được thao tác. Đây là cách các mục 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']

Tạo danh sách tốn nhiều công gõ hơn so với tạo chuỗi, nhưng danh sách hữu ích hơn vì có thể thao tác được. Ví dụ, chúng ta có thể in mục thứ ba trong wizard\_list (eye of newt) bằng cách nhập vị trí của nó trong danh sách (gọi là vị trí chỉ số) trong dấu ngoặc vuông ([]), như sau:

>>> print(wizard\_list[2])

eye of newt

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

Chúng ta cũng có thể thay đổi một mục trong danh sách dễ dàng hơn nhiều so với chuỗi. Ví dụ, thay vì eye of newt, chúng ta cần snail tongue. Đây là cách thực hiện:

>>> 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 mục tại chỉ số 2, trước đây là eye of newt, bằng snail tongue.

Dưới đây là phần tiếp theo của bản dịch đầy đủ:

**Một lựa chọn khác là hiển thị một tập hợp con 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 sau để xem các mục từ thứ ba đến thứ năm trong danh sách (một tập hợp nguyên liệu tuyệt vời cho một chiếc bánh sandwich ngon):

>>> print(wizard\_list[2:5])

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

Việc viết [2:5] giống như nói: "hiển thị các mục từ vị trí chỉ số 2 đến (nhưng không bao gồm) vị trí chỉ số 5" — hoặc nói cách khác, các mục 2, 3 và 4.

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

>>> some\_numbers = [1, 2, 5, 10, 20]

Danh sách cũng có thể chứa chuỗi:

>>> some\_strings = ['Which', 'Witch', 'Is', 'Which']

Chúng cũng có thể bao gồm hỗn hợp 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]

Thậm chí, danh sách có thể chứa 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 danh sách-lồng-trong-danh-sách này tạo ra ba biến: numbers chứa bốn số, strings chứa tám chuỗi, và mylist sử dụng numbers và strings. Danh sách thứ ba (mylist) chỉ có hai phần tử vì nó là danh sách của các tên biến, không phải nội dung của các biến.

**Thêm mục vào danh sách**

Để thêm mục vào danh sách, chúng ta sử dụng hàm append. Một hàm là một đoạn mã yêu cầu Python làm điều gì đó. Trong trường hợp này, append thêm một mục vào cuối danh sách.

Ví dụ, để thêm bear burp (chắc chắn là có thứ này) vào danh sách mua sắm của phù thủy, 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 nhiều mục ma thuật vào danh sách của phù thủy theo cách tương tự, như sau:

>>> wizard\_list.append('mandrake')

>>> wizard\_list.append('hemlock')

>>> wizard\_list.append('swamp gas')

Bây giờ danh sách của phù thủy trông như thế này:

>>> print(wizard\_list)

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

Phù thủy này rõ ràng đã sẵn sàng thực hiện một phép thuật nghiêm túc!

**Xóa mục khỏi danh sách**

Để xóa mục khỏi danh sách, sử dụng lệnh del (viết tắt của delete). Ví dụ, để xóa mục thứ sáu trong danh sách của phù thủy, snake dandruff, 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**:  
Hãy nhớ rằng các vị trí bắt đầu từ số 0, vì vậy wizard\_list[5] thực sự chỉ đến mục thứ sáu trong danh sách.

Đây là cách xóa các mục mà chúng ta vừa thêm (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']

**Phép toán với danh sách**

Chúng ta có thể nối các danh sách bằng cách cộng chúng, giống như cộng các số, sử dụng dấu cộng (+). Ví dụ, giả sử chúng ta có hai danh sách: list1, chứa các số từ 1 đến 4, và list2, chứa một vài từ. Chúng ta có thể cộng chúng lại như sau:

>>> 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 cho một biến khác.

python

Sao chép mã

>>> 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']

Và chúng ta có thể nhân một danh sách với một số. Ví dụ, để nhân list1 với 5, ta viết:

python

Sao chép mã

>>> list1 = [1, 2]

>>> print(list1 \* 5)

[1, 2, 1, 2, 1, 2, 1, 2, 1, 2]

Câu lệnh trên thực chất yêu cầu Python lặp lại danh sách list1 năm lần, tạo ra kết quả: 1, 2, 1, 2, 1, 2, 1, 2, 1, 2.

**Lỗi khi sử dụng phép chia (/) và trừ (-)**

Tuy nhiên, phép chia (/) và trừ (-) trên danh sách sẽ tạo ra lỗi, như các ví dụ sau:

python

Sao chép mã

>>> list1 / 20

Traceback (most recent call last):

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

list1 / 20

TypeError: unsupported operand type(s) for /: 'list' and 'int'

>>> list1 - 20

Traceback (most recent call last):

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

list1 - 20

TypeError: unsupported operand type(s) for -: 'list' and 'int'

**Tại sao lại có lỗi này?**

Việc nối danh sách bằng dấu cộng (+) và lặp lại danh sách bằng dấu nhân (\*) là các thao tác đơn giản. Chúng ta cũng có thể hiểu rõ ràng chúng trong thế giới thực. Ví dụ:

* Nếu bạn đưa cho tôi hai danh sách mua sắm và nói "Hãy cộng hai danh sách này", tôi sẽ viết tất cả các mục từ cả hai danh sách lên một tờ giấy, lần lượt từ đầu đến cuối.
* Tương tự, nếu bạn nói "Nhân danh sách này lên 3 lần", tôi có thể tưởng tượng viết danh sách đó ba lần trên một tờ giấy khác.

Nhưng chia một danh sách thì sao? Hãy thử xem: làm thế nào bạn chia một danh sách gồm sáu số (1 đến 6) thành hai phần? Dưới đây là ba cách chia khác nhau:

css

Sao chép mã

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

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

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

Chúng ta sẽ chia ở giữa, sau mục đầu tiên, hay chọn một điểm ngẫu nhiên nào đó để chia?

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 Python phản hồi bằng một lỗi.

**Tương tự, bạn cũng không thể cộng bất cứ thứ gì ngoài danh sách vào một danh sách.**

Ví dụ, dưới đây là những gì xảy ra khi chúng ta thử cộng số 50 vào danh sách list1:

python

Sao chép mã

>>> list1 + 50

Traceback (most recent call last):

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

list1 + 50

TypeError: can only concatenate list (not "int") to list

**Tại sao chúng ta gặp lỗi này?**

Hãy nghĩ xem: việc cộng số 50 vào một danh sách có nghĩa là gì?

* Có phải là thêm 50 vào mỗi mục trong danh sách? Nhưng nếu các mục không phải là số thì sao?
* Hay là thêm số 50 vào cuối hoặc đầu danh sách?

Trong lập trình máy tính, các lệnh phải hoạt động chính xác theo cùng một cách mỗi khi bạn nhập chúng. Máy tính không thông minh và chỉ nhìn mọi thứ theo cách "đúng" hoặc "sai". Yêu cầu nó thực hiện một quyết định phức tạp và nó sẽ tạo ra lỗi.

**Vẽ Với Turtle**

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

Turtle là một cách thú vị để học một số điều cơ bản về đồ họa máy tính. Vì vậy, trong chương này, chúng ta sẽ sử dụng turtle trong Python để vẽ một số hình dạng và đường thẳng đơn giản.

**Sử Dụng Module turtle Của Python**

**Module** trong Python là cách cung cấp mã hữu ích để sử dụng trong một chương trình khác (ngoài ra, module còn có thể chứa các hàm mà chúng ta có thể sử dụng). Chúng ta sẽ học thêm về module trong Chương 7. Python có một module đặc biệt gọi là turtle, được sử dụng để học cách máy tính vẽ hình trên màn hình.

Module turtle là một cách lập trình đồ họa vector, tức là vẽ với các đường thẳng, điểm, và đường cong đơn giản.

Hãy xem cách turtle hoạt động. Đầu tiên, khởi chạy Python Shell bằng cách nhấp vào biểu tượng trên màn hình (hoặc nếu bạn đang dùng Ubuntu, chọn **Applications > Programming > IDLE**). Tiếp theo, yêu cầu Python sử dụng module turtle bằng cách nhập lệnh sau:

>>> import turtle

Việc nhập một module cho Python biết rằng bạn muốn sử dụng nó.

**Lưu ý:** Nếu bạn sử dụng Ubuntu và gặp lỗi tại bước này, có thể bạn cần cài đặt tkinter. Mở **Ubuntu Software Center**, nhập python-tk vào ô tìm kiếm. Kết quả "Tkinter – Writing Tk Applications with Python" sẽ xuất hiện. Nhấp **Install** để cài đặt gói này.

**Tạo Một Canvas**

Sau khi nhập module turtle, chúng ta cần tạo một canvas—một không gian trống để vẽ, giống như một tấm vải của nghệ sĩ. Để làm điều này, chúng ta gọi hàm Pen từ module turtle, tự động tạo một canvas. Nhập lệnh sau vào Python Shell:

>>> t = turtle.Pen()

Bạn sẽ thấy một hộp trống (canvas) với một mũi tên ở giữa, giống như sau:

Mũi tên ở giữa màn hình chính là turtle, và bạn nói đúng—nó không giống một con rùa lắm.

**Di Chuyển Turtle**

Bạn gửi các lệnh đến turtle bằng cách sử dụng các hàm có sẵn trên biến t mà chúng ta vừa tạo, giống như sử dụng hàm Pen trong module turtle. Ví dụ, lệnh forward sẽ yêu cầu turtle di chuyển về phía trước.

Để yêu cầu turtle di chuyển 50 pixel, hãy nhập lệnh sau:

>>> t.forward(50)

Bạn sẽ thấy turtle di chuyển 50 pixel về phía trước, để lại một đường thẳng trên canvas.

**Ghi chú:** Một pixel là một điểm nhỏ nhất có thể hiển thị trên màn hình. Tất cả những gì bạn thấy trên màn hình máy tính đều được tạo thành từ các pixel nhỏ, hình vuông.

**Các Lệnh Điều Hướng Khác**

Bạn có thể sử dụng lệnh left để yêu cầu turtle xoay trái 90 độ:

>>> t.left(90)

Ngoài ra, có thể sử dụng các lệnh như right, backward, up (dừng vẽ), và down (bắt đầu vẽ lại).

**Học Qua Ví Dụ**

**Vẽ Một Hình Vuông**

Nhập các lệnh sau để vẽ một hình vuông:

>>> t.forward(50)

>>> t.left(90)

>>> t.forward(50)

>>> t.left(90)

>>> t.forward(50)

>>> t.left(90)

>>> t.forward(50)

**Xóa Canvas**

Để xóa canvas, sử dụng lệnh:

>>> t.reset()

Nếu chỉ muốn xóa màn hình mà không thay đổi vị trí của turtle:

>>> t.clear()

**Thực Hành**

1. **Vẽ Hình Chữ Nhật**: Tạo một canvas mới và vẽ một hình chữ nhật.
2. **Vẽ Tam Giác**: Tạo một canvas khác và vẽ một tam giác.
3. **Vẽ Hình Hộp Không Góc**: Vẽ bốn đường thẳng như hình minh họa.