# 3

# GIỚI THIỆU DANH SÁCH

Trong chương này và chương tiếp theo, bạn sẽ tìm hiểu danh sách là gì và cách bắt đầu làm việc với các phần tử trong danh sách. Danh sách cho phép bạn lưu trữ Tập hợp thông tin tại một nơi, dù bạn chỉ có vài mục hay hàng triệu mục. Danh sách là một trong những tính năng mạnh mẽ nhất của Python, dễ dàng tiếp cận với các lập trình viên mới, và chúng liên kết nhiều khái niệm quan trọng trong lập trình.

## Danh sách là gì?

Danh sách là tập hợp các mục theo một thứ tự cụ thể. Bạn có thể tạo một danh sách bao gồm các chữ cái trong bảng chữ cái, các chữ số từ 0 đến 9, hoặc tên của tất cả mọi người trong gia đình. Bạn có thể đưa bất cứ thứ gì bạn muốn vào danh sách, và các mục trong danh sách không nhất thiết phải liên quan đến nhau theo bất kỳ cách cụ thể nào. Bởi vì

một danh sách thường chứa nhiều hơn một phần tử, bạn nên đặt tên danh sách ở dạng số nhiều, chẳng hạn như chữ cái, chữ số hoặc tên.

Trong Python, dấu ngoặc vuông ([]) biểu thị một danh sách, và các phần tử riêng lẻ trong danh sách được phân tách bằng dấu phẩy. Dưới đây là một ví dụ đơn giản về một danh sách chứa một số loại xe đạp:

```
bicycles.py bicycles = ['trek', 'cannondale', 'redline', 'specialized']
in (xe đạp)
```

Nếu bạn yêu cầu Python in một danh sách, Python sẽ trả về biểu diễn của nó danh sách, bao gồm cả dấu ngoặc vuông:

```
['trek', 'cannondale', 'redline', 'chuyên biệt']
```

Vì đây không phải là kết quả đầu ra mà bạn muốn người dùng của mình thấy, chúng ta hãy tìm hiểu cách để truy cập vào từng mục riêng lẻ trong danh sách.

### Truy cập các phần tử trong danh sách

Danh sách là các tập hợp được sắp xếp, vì vậy bạn có thể truy cập bất kỳ phần tử nào trong danh sách bằng cách cho Python biết vị trí hoặc chỉ mục của phần tử mong muốn. Để truy cập một phần tử trong danh sách, hãy viết tên của danh sách theo sau là chỉ mục của phần tử được đặt trong dấu ngoặc vuông.

Ví dụ, hãy lấy chiếc xe đạp đầu tiên trong danh sách xe đạp:

```
xe đạp = ['trek', 'cannondale', 'redline', 'specialized']
in(xe đạp[0])
```

Khi chúng ta yêu cầu một mục duy nhất từ danh sách, Python chỉ trả về phần tử đó mà không có dấu ngoặc vuông:

```
hành trình
```

Đây là kết quả mà bạn muốn người dùng thấy: đầu ra sạch sẽ, được định dạng gọn gàng.

Bạn cũng có thể sử dụng các phương thức chuỗi từ Chương 2 trên bất kỳ phần tử nào trong danh sách này. Ví dụ: bạn có thể định dạng phần tử 'trek' để trông dễ trình bày hơn bằng cách sử dụng phương thức title() :

```
xe đạp = ['trek', 'cannondale', 'redline', 'specialized']
in(xe đạp[0].title())
```

Ví dụ này tạo ra kết quả giống như ví dụ trước, ngoại trừ 'Trek' được viết hoa.

### Vị trí chỉ số bắt đầu từ 0, không phải 1

Python coi phần tử đầu tiên trong danh sách ở vị trí 0 chứ không phải vị trí 1. Điều này đúng với hầu hết các ngôn ngữ lập trình và lý do liên quan đến

cách các thao tác danh sách được triển khai ở cấp độ thấp hơn. Nếu bạn nhận được kết quả không mong muốn, hãy tự hỏi liệu bạn có đang mắc phải lỗi sai lệch đơn giản nhưng phổ biến này không.

Mục thứ hai trong danh sách có chỉ số là 1. Sử dụng hệ thống đếm này, bạn có thể lấy bất kỳ phần tử nào bạn muốn từ danh sách bằng cách trừ đi một phần tử khỏi vị trí của nó trong danh sách. Ví dụ: để truy cập mục thứ tư trong danh sách, bạn yêu cầu mục ở chỉ số 3.

Câu hỏi sau đây yêu cầu tìm xe đạp ở vị trí số 1 và số 3:

```
xe đạp = ['trek', 'cannondale', 'redline', 'specialized']
in(xe đạp[1])
in(xe đạp[3])
```

Đoạn mã này trả về chiếc xe đạp thứ hai và thứ tư trong danh sách:

```
Cannondale
chuyên
```

Python có cú pháp đặc biệt để truy cập phần tử cuối cùng trong danh sách. Nếu bạn yêu cầu mục ở chỉ mục -1, Python luôn trả về mục cuối cùng trong danh sách:

```
xe đạp = ['trek', 'cannondale', 'redline', 'specialized']
in(xe đạp[-1])
```

Đoạn mã này trả về giá trị 'specialized'. Cú pháp này khá hữu ích, vì bạn thường muốn truy cập các mục cuối cùng trong danh sách mà không cần biết chính xác danh sách dài bao nhiêu. Quy ước này cũng áp dụng cho các giá trị chỉ mục âm khác. Chỉ mục -2 trả về mục thứ hai từ cuối danh sách, chỉ mục -3 trả về mục thứ ba từ cuối danh sách, v.v.

### Sử dụng các giá trị riêng lẻ từ danh sách

Bạn có thể sử dụng các giá trị riêng lẻ từ danh sách giống như bất kỳ biến nào khác. Ví dụ: bạn có thể sử dụng chuỗi f để tạo thông báo dựa trên giá trị từ danh sách.

Hãy thử lấy chiếc xe đạp đầu tiên từ danh sách và soạn tin nhắn sử dụng giá trị đó:

```
xe đạp = ['trek', 'cannondale', 'redline', 'specialized']
message = f"Chiếc xe đạp đầu tiên của tôi là {bicycles[0].title()}."

in(tin nhắn)
```

Chúng ta xây dựng một câu sử dụng giá trị tại bicycles[0] và gán nó cho biến message. Đầu ra là một câu đơn giản về chiếc xe đạp đầu tiên trong danh sách:

```
Chiếc xe đạp đầu tiên của tôi là Trek.
```

HÃY TỰ THỬ

Hãy thử các chương trình ngắn này để có được trải nghiệm trực tiếp với danh sách của Python. Bạn có thể muốn tạo một thư mục mới cho các bài tập của mỗi chương để sắp xếp chúng một cách có tổ chức.

3-1. Tên: Lưu trữ tên của một vài người bạn trong danh sách gọi là tên. In tên của từng người bằng cách truy cập từng phần tử trong danh sách, từng phần tử một.

3-2. Lời chào: Bắt đầu với danh sách bạn đã dùng trong Bài tập 3-1, nhưng thay vì chỉ in tên từng người, hãy in một tin nhắn gửi đến họ. Nội dung của mỗi tin nhắn phải giống nhau, nhưng mỗi tin nhắn phải được cá nhân hóa bằng tên của người nhận.

3-3. Danh sách của riêng bạn: Hãy nghĩ về phương tiện di chuyển yêu thích của bạn, chẳng hạn như xe máy hoặc ô tô, và lập một danh sách lưu trữ một số ví dụ. Sử dụng danh sách của bạn để in một loạt các câu nói về những mục này, chẳng hạn như "Tôi muốn sở hữu một chiếc xe máy Honda".

## Sửa đổi, Thêm và Xóa các Phần tử

Hầu hết các danh sách bạn tạo sẽ là danh sách động, nghĩa là bạn sẽ xây dựng một danh sách rồi thêm và bớt các phần tử trong đó khi chương trình chạy. Ví dụ, bạn có thể tạo một trò chơi trong đó người chơi phải bắn hạ người ngoài hành tinh. Bạn có thể lưu trữ tập hợp người ngoài hành tinh ban đầu vào một danh sách và sau đó xóa một người khỏi danh sách mỗi khi có một người bị bắn hạ. Mỗi khi một người ngoài hành tinh mới xuất hiện trên màn hình, bạn sẽ thêm người đó vào danh sách. Danh sách người ngoài hành tinh của bạn sẽ tăng và giảm độ dài trong suốt quá trình chơi.

### Sửa đổi các phần tử trong danh sách

Cú pháp để sửa đổi một phần tử tương tự như cú pháp để truy cập một phần tử trong danh sách. Để thay đổi một phần tử, hãy sử dụng tên của danh sách, theo sau là chỉ mục của phần tử bạn muốn thay đổi, rồi nhập giá trị mới mà bạn muốn phần tử đó có.

Ví dụ, giả sử chúng ta có một danh sách xe máy và mục đầu tiên trong danh sách là 'honda'. Chúng ta có thể thay đổi giá trị của mục đầu tiên này sau khi danh sách được tạo:

motorcycles.py
```
xe máy = ['honda', 'yamaha', 'suzuki']
in (xe máy)

xe máy[0] = 'ducati'
in (xe máy)
```

Ở đây chúng tôi định nghĩa danh sách xe máy, với 'honda' là phần tử đầu tiên.
Sau đó, chúng ta thay đổi giá trị của mục đầu tiên thành 'ducati'. Kết quả trả về cho thấy mục đầu tiên đã được thay đổi, trong khi phần còn lại của danh sách vẫn giữ nguyên:

```
['honda', 'yamaha', 'suzuki']
['ducati', 'yamaha', 'suzuki']
```

Bạn có thể thay đổi giá trị của bất kỳ mục nào trong danh sách, không chỉ mục đầu tiên.

## Thêm phần tử vào danh sách

Bạn có thể muốn thêm một phần tử mới vào danh sách vì nhiều lý do. Ví dụ: bạn có thể muốn thêm người ngoài hành tinh mới vào trò chơi, thêm dữ liệu mới vào hình ảnh trực quan hoặc thêm người dùng đã đăng ký mới vào trang web bạn đã xây dựng. Python cung cấp nhiều cách để thêm dữ liệu mới vào danh sách hiện có.

### Thêm các phần tử vào cuối danh sách

Cách đơn giản nhất để thêm một phần tử mới vào danh sách là thêm phần tử đó vào danh sách. Khi bạn thêm một phần tử vào danh sách, phần tử mới sẽ được thêm vào cuối danh sách. Sử dụng cùng danh sách đã có trong ví dụ trước, chúng ta sẽ thêm phần tử mới 'ducati' vào cuối danh sách:

```
xe máy = ['honda', 'yamaha', 'suzuki']
in (xe máy)

xe máy.append('ducati')
in (xe máy)
```

Ở đây, phương thức append() thêm 'ducati' vào cuối danh sách mà không ảnh hưởng đến bất kỳ phần tử nào khác trong danh sách:

```
['honda', 'yamaha', 'suzuki']
['honda', 'yamaha', 'suzuki', 'ducati']
```

Phương thức append() giúp dễ dàng xây dựng danh sách động. Ví dụ, bạn có thể bắt đầu với một danh sách rỗng và sau đó thêm các mục vào danh sách bằng một loạt các lệnh append() . Sử dụng một danh sách rỗng, hãy thêm các phần tử 'honda', 'yamaha' và 'suzuki' vào danh sách:

```
xe máy = []

xe máy.append('honda')
xe máy.append('yamaha')
xe máy.append('suzuki')

in (xe máy)
```

Danh sách kết quả trông giống hệt như danh sách trong các ví dụ trước:

```
['honda', 'yamaha', 'suzuki']
```

Việc xây dựng danh sách theo cách này rất phổ biến, vì bạn thường không biết dữ liệu mà người dùng muốn lưu trữ trong chương trình cho đến khi chương trình chạy. Để người dùng nắm quyền kiểm soát, hãy bắt đầu bằng cách định nghĩa một danh sách rỗng chứa các giá trị của người dùng. Sau đó, thêm từng giá trị mới được cung cấp vào danh sách bạn vừa tạo.

### Chèn các phần tử vào danh sách

Bạn có thể thêm một phần tử mới vào bất kỳ vị trí nào trong danh sách của mình bằng cách sử dụng insert() phương pháp. Bạn thực hiện điều này bằng cách chỉ định chỉ mục của phần tử mới và giá trị của mục mới:

```
xe máy = ['honda', 'yamaha', 'suzuki']

xe máy.chèn(0, 'ducati')
in (xe máy)
```

Trong ví dụ này, chúng ta chèn giá trị 'ducati' vào đầu danh sách. Phương thức insert() mở một khoảng trắng ở vị trí 0 và lưu trữ giá trị 'ducati' tại vị trí đó:

```
['ducati', 'honda', 'yamaha', 'suzuki']
```

Hoạt động này dịch chuyển mọi giá trị khác trong danh sách sang vị trí bên phải một vị trí.

## Xóa các phần tử khỏi danh sách

Thông thường, bạn sẽ muốn xóa một vật phẩm hoặc một nhóm vật phẩm khỏi danh sách. Ví dụ: khi một người chơi bắn hạ một người ngoài hành tinh từ trên trời, rất có thể bạn sẽ muốn xóa nó khỏi danh sách người ngoài hành tinh đang hoạt động. Hoặc khi người dùng quyết định hủy tài khoản trên ứng dụng web bạn đã tạo, bạn sẽ muốn xóa người dùng đó khỏi danh sách người dùng đang hoạt động. Bạn có thể xóa một vật phẩm dựa trên vị trí của nó trong danh sách hoặc dựa trên giá trị của nó.

### Xóa một mục bằng cách sử dụng câu lệnh del

Nếu bạn biết vị trí của mục bạn muốn xóa khỏi danh sách, bạn có thể sử dụng câu lệnh del :

```
xe máy = ['honda', 'yamaha', 'suzuki']
in (xe máy)

del xe máy[0]
in (xe máy)
```

Ở đây chúng ta sử dụng câu lệnh del để xóa mục đầu tiên, 'honda', khỏi danh sách xe máy:

```
['honda', 'yamaha', 'suzuki']
['yamaha', 'suzuki']
```

Bạn có thể xóa một mục khỏi bất kỳ vị trí nào trong danh sách bằng câu lệnh del nếu bạn biết chỉ mục của mục đó. Ví dụ: đây là cách xóa mục thứ hai, 'yamaha', khỏi danh sách:

```
xe máy = ['honda', 'yamaha', 'suzuki']
in (xe máy)

del xe máy[1]
in (xe máy)
```

Chiếc xe máy thứ hai bị xóa khỏi danh sách:

```
['honda', 'yamaha', 'suzuki']
['honda', 'suzuki']
```

Trong cả hai ví dụ, bạn không thể truy cập vào giá trị đã bị xóa khỏi danh sách sau khi sử dụng câu lệnh del .

### Xóa một mục bằng phương thức pop()

Đôi khi bạn sẽ muốn sử dụng giá trị của một mục sau khi xóa nó khỏi danh sách. Ví dụ: bạn có thể muốn lấy vị trí x và y của một người ngoài hành tinh vừa bị bắn hạ, để có thể vẽ một vụ nổ tại vị trí đó. Trong một ứng dụng web, bạn có thể muốn xóa một người dùng khỏi danh sách thành viên đang hoạt động rồi thêm người dùng đó vào danh sách thành viên không hoạt động.

Phương thức pop() xóa phần tử cuối cùng trong danh sách, nhưng cho phép bạn làm việc với phần tử đó sau khi xóa nó. Thuật ngữ pop xuất phát từ việc hình dung danh sách như một chồng các phần tử và việc lấy một phần tử ra khỏi đỉnh chồng là một phép loại bỏ. Trong phép so sánh này, đỉnh của một chồng tương ứng với điểm cuối của danh sách.

Chúng ta hãy loại một chiếc xe máy ra khỏi danh sách xe máy:

```
1 xe máy = ['honda', 'yamaha', 'suzuki']
  in (xe máy)

2 popped_motorcycle = motorcycles.pop()
3 bản in (xe máy)
4 bản in (popped_motorcycle)
```

Chúng ta bắt đầu bằng cách định nghĩa và in danh sách xe máy 1. Sau đó, chúng ta lấy một giá trị ra khỏi danh sách và gán giá trị đó cho biến popped_motorcycle 2.
Chúng tôi in danh sách 3 để cho thấy một giá trị đã bị xóa khỏi danh sách.
Sau đó, chúng ta in giá trị đã bật ra là 4 để chứng minh rằng chúng ta vẫn có thể truy cập vào giá trị đã bị xóa.

Đầu ra cho thấy giá trị 'suzuki' đã bị xóa khỏi phần cuối của
danh sách và hiện được gán cho biến popped_motorcycle:

```
['honda', 'yamaha', 'suzuki']
['honda', 'yamaha']
suzuki
```

Phương thức pop() này có thể hữu ích như thế nào? Hãy tưởng tượng các xe máy trong danh sách được lưu trữ theo thứ tự thời gian, dựa trên thời điểm chúng ta sở hữu chúng. Nếu vậy, chúng ta có thể sử dụng phương thức pop() để in ra một câu lệnh về chiếc xe máy cuối cùng chúng ta đã mua:

```
xe máy = ['honda', 'yamaha', 'suzuki']

last_owned = motorcycles.pop()
print(f"Chiếc xe máy cuối cùng tôi sở hữu là {last_owned.title()}.")
```

Đầu ra là một câu đơn giản về chiếc xe máy gần đây nhất mà chúng tôi sở hữu:

```
Chiếc xe máy cuối cùng tôi sở hữu là một chiếc Suzuki.
```

### Lấy các mục từ bất kỳ vị trí nào trong danh sách

Bạn có thể sử dụng pop() để xóa một mục khỏi bất kỳ vị trí nào trong danh sách bằng cách bao gồm chỉ mục của mục bạn muốn xóa trong dấu ngoặc đơn:

```
xe máy = ['honda', 'yamaha', 'suzuki']

first_owned = motorcycles.pop(0)
print(f"Chiếc xe máy đầu tiên tôi sở hữu là {first_owned.title()}.")
```

Chúng tôi bắt đầu bằng cách đưa chiếc xe máy đầu tiên vào danh sách, sau đó in ra thông báo về chiếc xe máy đó. Kết quả là một câu đơn giản mô tả chiếc xe máy đầu tiên tôi từng sở hữu:

```
Chiếc xe máy đầu tiên tôi sở hữu là một chiếc Honda.
```

Hãy nhớ rằng mỗi lần bạn sử dụng pop(), mục bạn làm việc cùng là no không còn được lưu trữ trong danh sách nữa.

Nếu bạn không chắc chắn nên sử dụng câu lệnh del hay phương thức pop() , đây là một cách đơn giản để quyết định: khi bạn muốn xóa một mục khỏi danh sách và không sử dụng mục đó theo bất kỳ cách nào, hãy sử dụng câu lệnh del ; nếu bạn muốn sử dụng một mục khi xóa nó, hãy sử dụng phương thức pop() .

### Xóa một mục theo giá trị

Đôi khi bạn không biết vị trí của giá trị bạn muốn xóa khỏi danh sách. Nếu bạn chỉ biết giá trị của phần tử bạn muốn xóa, bạn có thể sử dụng phương thức remove() .

Ví dụ, giả sử chúng ta muốn xóa giá trị 'ducati' khỏi danh sách xe máy:

```
xe máy = ['honda', 'yamaha', 'suzuki', 'ducati']
in (xe máy)

xe máy.xóa('ducati')
in (xe máy)
```

Ở đây phương thức remove() yêu cầu Python tìm ra vị trí của 'ducati'
xuất hiện trong danh sách và xóa phần tử đó:

```
['honda', 'yamaha', 'suzuki', 'ducati']
['honda', 'yamaha', 'suzuki']
```

Bạn cũng có thể sử dụng phương thức remove() để xử lý giá trị đang bị xóa khỏi danh sách.
Hãy xóa giá trị 'ducati' và in ra lý do xóa nó khỏi danh sách:

```
1 xe máy = ['honda', 'yamaha', 'suzuki', 'ducati']
  in (xe máy)

2 too_expensive = 'ducati'
3 xe máy.xóa(quá_đắt)
  in (xe máy)
4 print(f"\nA {too_expensive.title()} quá đắt đối với tôi.")
```

Sau khi xác định danh sách 1, chúng ta gán giá trị 'ducati' cho một biến có tên là
too_expensive 2. Sau đó, chúng ta sử dụng biến này để cho Python biết giá trị nào cần xóa khỏi
danh sách 3. Giá trị 'ducati' đã bị xóa khỏi danh sách 4 nhưng vẫn có thể truy cập được
thông qua biến too_expensive, cho phép chúng ta in ra câu lệnh về lý do tại sao chúng ta
xóa 'ducati' khỏi danh sách xe máy:

```
['honda', 'yamaha', 'suzuki', 'ducati']
['honda', 'yamaha', 'suzuki']

Một chiếc Ducati quá đắt đối với tôi.
```

LƯU Ý: hương thức remove() chỉ xóa lần xuất hiện đầu tiên của giá trị bạn chỉ định. Nếu giá trị đó có khả năng xuất hiện nhiều hơn một lần trong danh sách, bạn sẽ cần sử dụng vòng lặp để đảm bảo tất cả các lần xuất hiện của giá trị đó đều bị xóa. Bạn sẽ tìm hiểu cách thực hiện việc này trong Chương 7.

HÃY TỰ THỬ

Các bài tập sau đây phức tạp hơn một chút so với các bài tập trong Chương 2, nhưng chúng cho bạn cơ hội sử dụng danh sách theo mọi cách được mô tả.

3-4. Danh sách khách mời: Nếu bạn có thể mời bất kỳ ai, dù còn sống hay đã khuất, đến dự tiệc tối, bạn sẽ mời ai? Hãy lập một danh sách bao gồm ít nhất ba người bạn muốn mời đến dự tiệc tối. Sau đó, hãy dùng danh sách đó để in lời mời đến từng người.

(tiếp theo)

3-5. Thay đổi danh sách khách mời: Bạn vừa nghe tin một vị khách không thể tham dự bữa tối, nên bạn cần gửi một bộ thiệp mời mới. Bạn sẽ phải nghĩ đến việc mời thêm người khác.

- Bắt đầu với chương trình từ Bài tập 3-4. Thêm lệnh gọi print() vào cuối chương trình, nêu rõ tên của khách không thể tham dự.
- Sửa đổi danh sách của bạn, thay thế tên của khách không thể tham dự bằng tên của người mới mà bạn đang mời.
- In một bộ tin nhắn mời thứ hai, một tin nhắn cho mỗi người vẫn còn trong danh sách của bạn.

3-6. Thêm khách: Bạn vừa tìm được một chiếc bàn ăn lớn hơn, nên giờ có thêm không gian. Hãy nghĩ đến việc mời thêm ba vị khách nữa đến ăn tối.

- Bắt đầu với chương trình từ Bài tập 3-4 hoặc 3-5. Thêm lệnh gọi print() vào cuối chương trình, thông báo cho mọi người rằng bạn đã tìm thấy một bảng lớn hơn.
- Sử dụng insert() để thêm một khách mới vào đầu danh sách của bạn.
- Sử dụng insert() để thêm một khách mới vào giữa danh sách của bạn.
- Sử dụng append() để thêm một khách mới vào cuối danh sách của bạn.
- In một bộ tin nhắn mời mới, mỗi tin nhắn cho một người trong danh sách của bạn.

3-7. Danh sách khách mời đang giảm dần: Bạn vừa phát hiện ra rằng bàn ăn mới của bạn sẽ không được chuyển đến kịp giờ ăn tối và giờ bạn chỉ còn chỗ cho hai khách.

- Bắt đầu với chương trình từ Bài tập 3-6. Thêm một dòng mới in ra thông báo cho biết bạn chỉ có thể mời hai người đi ăn tối.
- Sử dụng hàm pop() để xóa từng khách khỏi danh sách cho đến khi chỉ còn lại hai tên. Mỗi lần xóa tên khỏi danh sách, hãy in một tin nhắn cho người đó để thông báo rằng bạn rất tiếc vì không thể mời họ đi ăn tối.
- In tin nhắn cho mỗi người trong số hai người vẫn còn trong danh sách của bạn, cho họ biết rằng họ vẫn được mời.
- Sử dụng lệnh del để xóa hai tên cuối cùng khỏi danh sách, để bạn có một danh sách trống. In danh sách để đảm bảo bạn thực sự có một danh sách trống ở cuối chương trình.

## Tổ chức danh sách

Thông thường, danh sách của bạn sẽ được tạo theo thứ tự không thể đoán trước vì bạn không thể luôn kiểm soát được thứ tự người dùng cung cấp dữ liệu của họ. Mặc dù điều này là không thể tránh khỏi trong hầu hết các trường hợp, nhưng bạn thường muốn trình bày thông tin theo một thứ tự cụ thể. Đôi khi bạn sẽ muốn

để giữ nguyên thứ tự ban đầu của danh sách, và đôi khi bạn muốn thay đổi thứ tự ban đầu. Python cung cấp nhiều cách khác nhau để sắp xếp danh sách, tùy thuộc vào tình huống.

## Sắp xếp danh sách vĩnh viễn bằng phương thức sort()

Phương thức sort() của Python giúp việc sắp xếp danh sách trở nên tương đối dễ dàng. Hãy tưởng tượng chúng ta có một danh sách các xe hơi và muốn thay đổi thứ tự của danh sách để lưu trữ chúng theo thứ tự bảng chữ cái. Để đơn giản, hãy giả sử tất cả các giá trị trong danh sách đều là chữ thường:

```
cars.py cars = ['bmw', 'audi', 'toyota', 'subaru']
        cars.sort()
        in(ô tô)
```

Phương thức sort() thay đổi thứ tự danh sách vĩnh viễn. Các xe bây giờ được sắp xếp theo thứ tự bảng chữ cái và chúng ta không bao giờ có thể quay lại thứ tự ban đầu:

```
['audi', 'bmw', 'subaru', 'toyota']
```

Bạn cũng có thể sắp xếp danh sách này theo thứ tự ngược bảng chữ cái bằng cách truyền tham số reverse=True vào phương thức sort() . Ví dụ sau đây sắp xếp danh sách xe theo thứ tự ngược bảng chữ cái:

```
xe ô tô = ['bmw', 'audi', 'toyota', 'subaru']
cars.sort(reverse=True)
in(ô tô)
```

Một lần nữa, thứ tự của danh sách được thay đổi vĩnh viễn:

```
['toyota', 'subaru', 'bmw', 'audi']
```

## Sắp xếp danh sách tạm thời bằng hàm sorted()

Để duy trì thứ tự ban đầu của danh sách nhưng vẫn hiển thị theo thứ tự đã được sắp xếp, bạn có thể sử dụng hàm sorted() . Hàm sorted() cho phép bạn hiển thị danh sách theo một thứ tự cụ thể, nhưng không ảnh hưởng đến thứ tự thực tế của danh sách.

Chúng ta hãy thử chức năng này trên danh sách xe ô tô.

```
xe ô tô = ['bmw', 'audi', 'toyota', 'subaru']

1 bản in("Đây là danh sách gốc:")
  in(ô tô)

2 print("\nĐây là danh sách đã được sắp xếp:")
  in(đã sắp xếp(xe hơi))

3 print("\nĐây là danh sách gốc một lần nữa:")
  in(ô tô)
```

Đầu tiên, chúng ta in danh sách theo thứ tự ban đầu là 1, sau đó theo thứ tự bảng chữ cái là 2. Sau khi danh sách được hiển thị theo thứ tự mới, chúng ta thấy rằng danh sách vẫn được lưu trữ theo thứ tự ban đầu là 3:

```
Sau đây là danh sách gốc:
['bmw', 'audi', 'toyota', 'subaru']

Sau đây là danh sách đã được sắp xếp:
['audi', 'bmw', 'subaru', 'toyota']

1 Dưới đây là danh sách gốc một lần nữa:
['bmw', 'audi', 'toyota', 'subaru']
```

Lưu ý rằng danh sách vẫn tồn tại theo thứ tự ban đầu là 1 sau sorted() hàm đã được sử dụng. Hàm sorted() cũng có thể chấp nhận reverse=True tham số nếu bạn muốn hiển thị danh sách theo thứ tự bảng chữ cái ngược lại.

LƯU Ý: o xếp danh sách theo thứ tự bảng chữ cái sẽ phức tạp hơn một chút khi tất cả các giá trị không được viết thường. Có một số cách để diễn giải chữ in hoa khi xác định thứ tự sắp xếp, và việc chỉ định thứ tự chính xác có thể phức tạp hơn những gì chúng ta muốn xử lý tại thời điểm này. Tuy nhiên, hầu hết các phương pháp sắp xếp đều được xây dựng trực tiếp dựa trên những gì bạn đã học trong phần này.

## In danh sách theo thứ tự ngược lại

Để đảo ngược thứ tự ban đầu của một danh sách, bạn có thể sử dụng phương thức reverse() . Nếu ban đầu chúng ta lưu trữ danh sách xe theo thứ tự thời gian sở hữu, chúng ta có thể dễ dàng sắp xếp lại danh sách theo thứ tự thời gian đảo ngược:

```
xe ô tô = ['bmw', 'audi', 'toyota', 'subaru']
in(ô tô)

cars.reverse()
in(ô tô)
```

Lưu ý rằng reverse() không sắp xếp ngược theo thứ tự bảng chữ cái; nó chỉ đảo ngược thứ tự của danh sách:

```
['bmw', 'audi', 'toyota', 'subaru']
['subaru', 'toyota', 'audi', 'bmw']
```

Phương thức reverse() thay đổi thứ tự của danh sách vĩnh viễn, nhưng bạn có thể quay lại thứ tự ban đầu bất cứ lúc nào bằng cách áp dụng reverse() cho cùng một danh sách lần thứ hai.

## Tìm độ dài của một danh sách

Bạn có thể nhanh chóng tìm độ dài của một danh sách bằng cách sử dụng hàm len() . Danh sách trong ví dụ này có bốn phần tử, do đó độ dài của nó là 4:

```
>>> xe hơi = ['bmw', 'audi', 'toyota', 'subaru']
>>> len(xe hơi)
4
```

Bạn sẽ thấy len() hữu ích khi bạn cần xác định số lượng người ngoài hành tinh vẫn cần phải bắn hạ trong trò chơi, xác định lượng dữ liệu bạn phải quản lý trong hình ảnh trực quan hoặc tính toán số lượng người dùng đã đăng ký trên một trang web, cùng nhiều nhiệm vụ khác.

**LƯU Ý** ython đếm các mục trong danh sách bắt đầu từ số 1, do đó bạn sẽ không gặp phải bất kỳ lỗi nào khi xác định độ dài của danh sách.

### HÃY TỰ THỬ

3-8. Khám phá thế giới: Hãy nghĩ đến ít nhất năm địa điểm trên thế giới mà bạn muốn đến thăm.

- Lưu trữ các địa điểm trong một danh sách. Đảm bảo danh sách không được sắp xếp theo thứ tự bảng chữ cái.
- In danh sách theo thứ tự ban đầu. Đừng lo lắng về việc in danh sách một cách gọn gàng; chỉ cần in nó dưới dạng danh sách Python thô.
- Sử dụng sorted() để in danh sách của bạn theo thứ tự bảng chữ cái mà không sửa đổi danh sách thực tế.
- Thể hiện danh sách của bạn vẫn theo thứ tự ban đầu bằng cách in nó ra.
- Sử dụng sorted() để in danh sách của bạn theo thứ tự bảng chữ cái ngược mà không thay đổi thứ tự của danh sách ban đầu.
- Chứng minh danh sách của bạn vẫn theo thứ tự ban đầu bằng cách in lại.
- Sử dụng reverse() để thay đổi thứ tự danh sách. In danh sách để cho thấy thứ tự đã thay đổi.
- Sử dụng reverse() để thay đổi thứ tự danh sách một lần nữa. In danh sách để hiển thị nó đã trở lại thứ tự ban đầu.
- Sử dụng sort() để sắp xếp lại danh sách theo thứ tự bảng chữ cái. In danh sách để hiển thị thứ tự đã được thay đổi.
- Sử dụng sort() để thay đổi danh sách của bạn sao cho nó được lưu trữ theo thứ tự bảng chữ cái ngược lại. In danh sách để cho thấy thứ tự của danh sách đã thay đổi.

3-9. Khách ăn tối: Làm việc với một trong các chương trình từ Bài tập 3-4 đến 3-7 (trang 41–42), sử dụng len() để in thông báo cho biết số lượng người bạn mời ăn tối.

3-10. Mọi hàm: Hãy nghĩ về những thứ bạn có thể lưu trữ trong một danh sách. Ví dụ, bạn có thể tạo một danh sách các ngọn núi, dòng sông, quốc gia, thành phố, ngôn ngữ hoặc bất kỳ thứ gì bạn muốn. Hãy viết một chương trình tạo một danh sách chứa các mục này và sau đó sử dụng mỗi hàm được giới thiệu trong chương này ít nhất một lần.

## Tránh lỗi chỉ mục khi làm việc với danh sách

Có một loại lỗi thường gặp khi bạn làm việc với danh sách lần đầu. Giả sử bạn có một danh sách gồm ba mục và bạn yêu cầu mục thứ tư:

motorcycles.py
```
xe máy = ['honda', 'yamaha', 'suzuki']
in(xe máy[3])
```

Ví dụ này dẫn đến lỗi chỉ mục:

```
Theo dõi (cuộc gọi gần đây nhất cuối cùng):
  Tệp "motorcycles.py", dòng 2, trong <module>
    in(xe máy[3])
       ~~~~~~~~~~~^^^

IndexError: danh sách chỉ mục nằm ngoài phạm vi
```

Python cố gắng trả về cho bạn phần tử ở vị trí số 3. Nhưng khi tìm kiếm trong danh sách, không có phần tử nào trong danh sách "motorcycles" có vị trí số 3. Do tính chất lệch nhau của việc lập chỉ mục trong danh sách, lỗi này khá phổ biến. Mọi người thường nghĩ phần tử thứ ba là phần tử số 3, vì chúng bắt đầu đếm từ 1. Nhưng trong Python, phần tử thứ ba là số 2, vì nó bắt đầu lập chỉ mục từ 0.

Lỗi chỉ mục nghĩa là Python không thể tìm thấy mục nào tại chỉ mục bạn yêu cầu. Nếu lỗi chỉ mục xảy ra trong chương trình, hãy thử điều chỉnh chỉ mục bạn đang yêu cầu thêm một đơn vị. Sau đó, chạy lại chương trình để xem kết quả có chính xác không.

Hãy nhớ rằng bất cứ khi nào bạn muốn truy cập vào mục cuối cùng trong danh sách, bạn nên sử dụng chỉ mục -1. Cách này luôn hiệu quả, ngay cả khi danh sách của bạn đã thay đổi kích thước kể từ lần truy cập cuối cùng:

```
xe máy = ['honda', 'yamaha', 'suzuki']
in(xe máy[-1])
```

Chỉ số -1 luôn trả về mục cuối cùng trong danh sách, trong trường hợp này là giá trị 'Suzuki':

```
suzuki
```

Cách tiếp cận này chỉ gây ra lỗi khi bạn yêu cầu mục cuối cùng từ một danh sách trống:

```
xe máy = []
in(xe máy[-1])
```

Không có mục nào trong xe máy, do đó Python trả về một lỗi chỉ mục khác:

```
Theo dõi (cuộc gọi gần đây nhất cuối cùng):
  Tệp "motorcyles.py", dòng 3, trong <module>
    in(xe máy[-1])
       ~~~~~~~~~~~^^^^

IndexError: danh sách chỉ mục nằm ngoài phạm vi
```

Nếu xảy ra lỗi chỉ mục và bạn không thể tìm ra cách giải quyết, hãy thử in danh sách hoặc chỉ in độ dài danh sách. Danh sách của bạn có thể trông khác rất nhiều so với bạn nghĩ, đặc biệt nếu nó được quản lý động bởi chương trình của bạn. Việc xem danh sách thực tế, hoặc số lượng mục chính xác trong danh sách, có thể giúp bạn phân loại các lỗi logic như vậy.

HÃY TỰ THỬ

3-11. Lỗi cố ý: Nếu bạn chưa gặp lỗi chỉ mục trong bất kỳ chương trình nào, hãy thử tạo ra lỗi. Thay đổi chỉ mục trong một chương trình để tạo ra lỗi chỉ mục. Đảm bảo bạn đã sửa lỗi trước khi đóng chương trình.

chương trình.

## Bản tóm tắt

Trong chương này, bạn đã tìm hiểu danh sách là gì và cách làm việc với từng mục riêng lẻ trong danh sách. Bạn đã học cách định nghĩa danh sách và cách thêm và xóa phần tử. Bạn đã học cách sắp xếp danh sách vĩnh viễn và tạm thời cho mục đích hiển thị. Bạn cũng đã học cách tìm độ dài của danh sách và cách tránh lỗi chỉ mục khi làm việc với danh sách.

Trong Chương 4, bạn sẽ học cách làm việc với các mục trong danh sách hiệu quả hơn. Bằng cách lặp qua từng mục trong danh sách chỉ bằng một vài dòng mã, bạn có thể làm việc hiệu quả, ngay cả khi danh sách của bạn chứa hàng nghìn hoặc hàng triệu mục.