Danh sách (List)

Nội dung

- Định nghĩa Danh sách
- Thay đổi, thêm, và xóa các phần tử
- Tổ chức danh sách
- Tránh lỗi chỉ mục
- Lặp qua toàn bộ danh sách
- Lập danh sách số
- Làm việc với một phần của danh sách
- Tuples

Định nghĩa Danh sách

- Trong Python, ngoặc vuông ([ ]) chỉ định một danh sách và các phần tử trong danh sách được phân tách bởi dấu phẩy (,). Ví dụ danh sách chứa các loại xe đạp khác nhau.

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
print(bicycles)
```

```
['trek', 'cannondale', 'redline', 'specialized']
```

Truy cập các phần tử trong danh sách

- Để truy cập một phần tử trong danh sách, hãy viết tên của danh sách theo sau là chỉ mục của mục được đặt trong dấu ngoặc vuông. Ví dụ: hãy lấy chiếc xe đạp đầu tiên trong danh sách bicycles:

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
①   print(bicycles[0])
```

```
trek
```

Python chỉ trả về phần tử đó mà không có dấu ngoặc vuông:

Có thể định dạng phần tử 'trek' gọn gàng hơn bằng cách sử dụng phương thức title():

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
print(bicycles[0].title())
```

```
Trek
```

Đánh chỉ mục

- Chỉ mục danh sách bắt đầu từ 0
- Để truy cập mục thứ tư trong danh sách, ta yêu cầu phần tử chỉ mục 3
- Để truy cập phần tử cuối cùng trong danh sách, dùng chỉ mục -1

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
print(bicycles[1])
print(bicycles[3])
```

```
cannondale
specialized
```

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
print(bicycles[-1])
```

```
specialized
```

Sử dụng giá trị riêng từ danh sách

- Có thể sử dụng các giá trị riêng lẻ từ một danh sách giống như cách ta làm với bất kỳ biến nào khác
- Ví dụ: ta có thể sử dụng f-string để tạo một thông báo dựa trên giá trị từ một danh sách.

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
message = f"My first bicycle was a {bicycles[0].title()}."
print(message)
```

```
My first bicycle was a Trek.
```

Ví dụ

Đoạn mã sau đây lặp lại danh sách tên xe và tìm kiếm giá trị 'bmw'. Bất cứ khi nào giá trị là 'bmw', nó sẽ được in bằng chữ hoa thay vì chữ hoa tiêu đề

```
cars = ['audi', 'bmw', 'subaru', 'toyota']
for car in cars:
    if car=='bmw':
        print(car.upper())
    else:
        print(car.title())
```

```
Audi
BMW
Subaru
Toyota
```

Thay đổi, thêm, và xóa các phần tử

- Hầu hết các danh sách tạo sẽ là động, nghĩa là ta sẽ xây dựng một danh sách và sau đó thêm và xóa các phần tử khỏi nó khi chương trình chạy.
- Ví dụ, ta có thể tạo một trò chơi trong đó người chơi phải bắn quái vật trên bầu trời. Ta có thể lưu trữ nhóm quái vật ban đầu trong một danh sách và sau đó xóa một con quái vật trong danh sách mỗi khi quái vật bị bắn hạ.
- Mỗi lần một quái vật mới xuất hiện trên màn hình, ta thêm nó vào danh sách. Danh sách quái vật sẽ tăng lên và giảm độ dài trong suốt quá trình của trò chơi.

Thay đổi phần tử trong danh sách

- Để thay đổi một phần tử, sử dụng tên của danh sách theo sau bằng chỉ mục của phần tử mong muốn thay đổi, sau đó cung cấp giá trị

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
motorcycles[0] = 'ducati'
print(motorcycles)
```

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

Thêm phần tử vào danh sách

- Thêm phần tử vào cuối danh sách: dùng phương thức append()

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
motorcycles.append('ducati')
print(motorcycles)
```

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

- Dùng append() để tạo danh sách động:

```
motorcycles = []
motorcycles.append('honda')
motorcycles.append('yamaha')
motorcycles.append('suzuki')
print(motorcycles)
```

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

Thêm một phần tử vào danh sách

- Thêm một phần tử mới ở bất kỳ vị trí nào trong danh sách bằng cách sử dụng phương thức insert()

```
motorcycles = ['honda', 'yamaha', 'suzuki']
motorcycles.insert(0, 'ducati')
print(motorcycles)
['ducati', 'honda', 'yamaha', 'suzuki']
```

Phương thức insert() tạo ra một khoảng trống tại giá trị 0 và lưu giá trị ‘ducati’ tại vị trí đó. Câu lệnh này dịch chuyển tất cả các phần tử còn lại một vị trí sang phải.

Xóa phần tử khỏi danh sách

- Xóa phần tử sử dụng lệnh **del**

(Khi biết vị trí của phần tử trong danh sách)

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
del motorcycles[0]
print(motorcycles)
```

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

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
del motorcycles[1]
print(motorcycles)
```

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

Trong cả hai ví dụ trên, chúng ta không thể truy cập vào được phần tử bị xóa khỏi danh sách sau khi sử dụng lệnh del.

Xóa một phần tử sử dụng phương thức pop()

- Phương thức pop() loại bỏ mục cuối cùng trong danh sách, nhưng nó cho phép ta làm việc với mục đó sau khi loại bỏ nó.

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
popped_motorcycle = motorcycles.pop()
print(motorcycles)
print(popped_motorcycle)
```

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

```
motorcycles = ['honda', 'yamaha', 'suzuki']
last_owned = motorcycles.pop()
print(f"The last motorcycle I owned was a {last_owned.title()}.")
```

```
The last motorcycle I owned was a Suzuki
```

Lấy phần tử từ bất kỳ vị trí nào trong danh sách

- Sử dụng pop() để xóa một mục khỏi bất kỳ vị trí nào trong danh sách bằng cách bao gồm chỉ mục của mục mà ta muốn xóa:

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

first_owned = motorcycles.pop(0)

print(f"The first motorcycle I owned was a {first_owned.title()}.")
```

```
The first motorcycle I owned was a Honda.
```

Khi ta 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 ta muốn sử dụng phần tử khi ta xóa nó, hãy sử dụng phương thức pop().

Xóa phần tử bằng giá trị

- Nếu ta chỉ biết giá trị của phần tử muốn xóa, ta có thể sử dụng phương thức remove().

```
motorcycles = ['honda', 'yamaha', 'suzuki', 'ducati']
print(motorcycles)
motorcycles.remove('ducati')
print(motorcycles)
```

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

Xóa phần tử bằng giá trị (t)

Sử dụng phương thức remove() để làm việc với một giá trị đang bị xóa khỏi danh sách

```
motorcycles = ['honda', 'yamaha', 'suzuki', 'ducati']
print(motorcycles)
too_expensive = 'ducati'
motorcycles.remove(too_expensive)
print(motorcycles)
print(f"\nA {too_expensive.title()} is too expensive for me.")
```

```
['honda', 'yamaha', 'suzuki', 'ducati']
['honda', 'yamaha', 'suzuki']
A Ducati is too expensive for me.
```

Tổ chức danh sách

- Thông thường, danh sách sẽ được tạo theo một thứ tự không thể biết trước, bởi vì không thể kiểm soát thứ tự mà người dùng cung cấp dữ liệu của họ.
- Vấn đề là ta luôn cần trình bày thông tin theo một trật tự nào đó.
- Đôi khi, ta cần giữ nguyên thứ tự ban đầu của dữ liệu, đôi khi lại cần thay đổi thứ tự theo lúc đầu.
- Python cung cấp một số cách khác nhau để thay đổi thứ tự tùy theo tình huống

Sắp xếp danh sách vĩnh viễn với phương thức sort()

Phương thức sort() của Python giúp sắp xếp danh sách:

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
cars.sort()
print(cars)
```

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

Sắp xếp danh sách này theo thứ tự bảng chữ cái ngược lại bằng cách chuyển đối số reverse = True vào phương thức sort()

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
cars.sort(reverse=True)
print(cars)
```

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

Sắp xếp danh sách tạm thời với phương thức sorted()

- Hàm sorted() cho phép hiển thị danh sách theo một thứ tự cụ thể nhưng không ảnh hưởng đến thứ tự thực tế của danh sách.

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
print("Here is the original list:")
print(cars)
print("\nHere is the sorted list:")
print(sorted(cars))
print("\nHere is the original list again:")
print(cars)
```

```
Here is the original list:
['bmw', 'audi', 'toyota', 'subaru']
Here is the sorted list:
['audi', 'bmw', 'subaru', 'toyota']
Here is the original list again:
['bmw', 'audi', 'toyota', 'subaru']
```

Hàm sorted() cũng có thể chấp nhận đối số reverse = True nếu ta muốn hiển thị danh sách theo thứ tự bảng chữ cái ngược lại.

In danh sách theo thứ tự ngược

- Để đảo ngược thứ tự ban đầu của danh sách, ta có thể sử dụng phương thức reverse()

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
print(cars)
cars.reverse()
print(cars)
```

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

Lưu ý rằng phương thức reverse() không sắp xếp theo thứ tự alphabet, nó chỉ đơn thuần là đảo ngược thứ tự của các phần tử trong danh sách hiện tại.

Tìm độ dài của danh sách

- Tìm độ dài của danh sách bằng cách sử dụng hàm len().

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

Chú ý: Python đếm các mục trong danh sách bắt đầu bằng một, vì vậy ta sẽ không gặp phải bất kỳ lỗi nào khi xác định độ dài của danh sách

Tránh lỗi chỉ mục

- Giả sử danh sách có 3 phần tử, ta yêu cầu in ra phần tử thứ tư, Python sẽ thông báo lỗi chỉ mục

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles[3])
```

```
Traceback (most recent call last):
File "motorcycles.py", line 2, in <module>
print(motorcycles[3])
IndexError: list index out of range
```

Lỗi chỉ mục có nghĩa là Python không thể kết xuất một mục tại chỉ mục được yêu cầu. Nếu lỗi chỉ mục xảy ra trong chương trình, hãy thử điều chỉnh chỉ mục đang được yêu cầu một đơn vị.

Tránh lỗi chỉ mục (t)

- Bất cứ khi nào ta muốn truy cập phần tử cuối cùng trong danh sách, hãy sử dụng chỉ mục (-1).

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles[-1])
```

```
'suzuki'
```

Tiếp cận này chỉ xẩy ra lỗi trong trường hợp duy nhất đó là khi danh sách trống

```
motorcycles = []
print(motorcycles[-1])
```

```
Traceback (most recent call last):
File "motorcyles.py", line 3, in <module>
print(motorcycles[-1])
IndexError: list index out of range
```

Bài tập

3-6. More Guests: You just found a bigger dinner table, so now more space is available. Think of three more guests to invite to dinner.

- Start with your program from Exercise 3-4 or Exercise 3-5. Add a `print()` call to the end of your program informing people that you found a bigger dinner table.
- Use `insert()` to add one new guest to the beginning of your list.
- Use `insert()` to add one new guest to the middle of your list.
- Use `append()` to add one new guest to the end of your list.
- Print a new set of invitation messages, one for each person in your list.

Bài tập

3-7. Shrinking Guest List: You just found out that your new dinner table won't arrive in time for the dinner, and you have space for only two guests.

- Start with your program from Exercise 3-6. Add a new line that prints a message saying that you can invite only two people for dinner.
- Use `pop()` to remove guests from your list one at a time until only two names remain in your list. Each time you pop a name from your list, print a message to that person letting them know you're sorry you can't invite them to dinner.
- Print a message to each of the two people still on your list, letting them know they're still invited.
- Use `del` to remove the last two names from your list, so you have an empty list. Print your list to make sure you actually have an empty list at the end of your program.

Bài tập

3-8. Seeing the World: Think of at least five places in the world you'd like to visit.

- Store the locations in a list. Make sure the list is not in alphabetical order.
- Print your list in its original order. Don't worry about printing the list neatly, just print it as a raw Python list.
- Use `sorted()` to print your list in alphabetical order without modifying the actual list.
- Show that your list is still in its original order by printing it.
- Use `sorted()` to print your list in reverse alphabetical order without changing the order of the original list.
- Show that your list is still in its original order by printing it again.
- Use `reverse()` to change the order of your list. Print the list to show that its order has changed.
- Use `reverse()` to change the order of your list again. Print the list to show it's back to its original order.
- Use `sort()` to change your list so it's stored in alphabetical order. Print the list to show that its order has been changed.
- Use `sort()` to change your list so it's stored in reverse alphabetical order. Print the list to show that its order has changed.

Bài tập

3-9. Dinner Guests: Working with one of the programs from Exercises 3-4 through 3-7 (page 42), use `len()` to print a message indicating the number of people you are inviting to dinner.

3-10. Every Function: Think of something you could store in a list. For example, you could make a list of mountains, rivers, countries, cities, languages, or anything else you'd like. Write a program that creates a list containing these items and then uses each function introduced in this chapter at least once.

Lập danh sách số

Danh sách là lý tưởng để lưu trữ các tập hợp số và Python cung cấp nhiều công cụ để giúp ta làm việc hiệu quả với danh sách các con số.

Một khi đã hiểu cách sử dụng các công cụ này một cách hiệu quả, mã nguồn sẽ hoạt động tốt ngay cả khi danh sách chứa hàng triệu mục tin.

Sử dụng hàm range()

Hàm range() của Python giúp dễ dàng tạo một chuỗi số

```
for value in range(1, 5):
    print(value)
```

```
1
2
3
4
```

Để in ra số 5 trong trường hợp này, chúng ta dùng range(1,6)

```
for value in range(1, 6):
    print(value)
```

```
1
2
3
4
5
```

Sử dụng range() để tạo ra danh sách số

Nếu muốn tạo một danh sách các số, ta có thể chuyển đổi kết quả của hàm range() trực tiếp vào một danh sách bằng cách sử dụng hàm list().

Khi chúng ta bọc list() xung quanh một hàm range(), đầu ra sẽ là một danh sách các số.

```
numbers = list(range(1, 6))
print(numbers)
```

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

Nếu ta truyền đối số thứ ba vào range(), Python sẽ sử dụng giá trị đó như một kích thước bước khi tạo số

```
even_numbers = list(range(2, 11, 2))
print(even_numbers)
```

```
[2, 4, 6, 8, 10]
```

Sử dụng range() để tạo ra danh sách số

cách ta có thể tạo danh sách các số bình phương tới 10

```
①   squares = []
②   for value in range(1, 11):
③       square = value ** 2
④       squares.append(square)
⑤   print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
squares = []
for value in range(1,11):
①       squares.append(value**2)
print(squares)
```

Thống kê đơn giản với danh sách số

Dễ dàng tổng hợp các giá trị tối thiểu, tối đa và tổng của một danh sách số:

```
>>> digits = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
>>> min(digits)
0
>>> max(digits)
9
>>> sum(digits)
45
```

Hiểu về danh sách

Ví dụ xây dựng cùng một danh sách các số bình phương mà ta đã thấy ở trên nhưng sử dụng khả năng hiểu danh sách:

```
squares = [value**2 for value in range(1, 11)]

print(squares)
```

Vòng for cho giá trị trong range(1,11) để đưa giá trị từ 1 tới 10 vào trong biểu thức. Ghi nhớ là không sử dụng dấu hai chấm ở cuối câu lệnh for.

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Làm việc với một phần của danh sách

Trong các phần trước, ta đã học cách truy cập vào các phần tử đơn trong danh sách và cách duyệt toàn bộ các phần tử trong danh sách.

Ở phần này, ta sẽ học cách làm việc với một nhóm cụ thể trong danh sách, được Python gọi là một lát cắt (slice)

Cắt lát một danh sách

Để in ra 3 phần tử đầu tiên trong danh sách, ta cần chỉ định từ 0 tới 3, Python sẽ trả về 3 phần tử là 0,1,2.

```
players = ['charles', 'martina', 'michael', 'florence', 'eli']
print(players[0:3])
```

```
['charles', 'martina', 'michael']
```

```
players = ['charles', 'martina', 'michael', 'florence', 'eli']
print(players[1:4])
```

```
['martina', 'michael', 'florence']
```

Nếu ta bỏ qua chỉ mục đầu tiên trong một slice, Python sẽ tự động bắt đầu slice ở đầu danh sách:

```
players = ['charles', 'martina', 'michael', 'florence', 'eli']
print(players[:4])
```

```
['charles', 'martina', 'michael', 'florence']
```

Cắt lát một danh sách

Nếu muốn tất cả các phần tử từ thứ ba đến cuối cùng, ta có thể bắt đầu với 2 và bỏ qua chỉ mục thứ hai.

```
players = ['charles', 'martina', 'michael', 'florence', 'eli']

print(players[2:])
```

```
['michael', 'florence', 'eli']
```

Cần nhớ rằng một chỉ mục âm trả về một phần tử cách cuối danh sách một khoảng cách nhất định; Ta có thể xuất bất kỳ lát nào từ cuối danh sách. Ví dụ, nếu ta muốn xuất ra ba cầu thủ cuối cùng trong danh sách, có thể sử dụng: `players[-3:]`

```
players = ['charles', 'martina', 'michael', 'florence', 'eli']

print(players[-3:])
```

```
['michael', 'florence', 'eli']
```

Lặp qua một lát cắt

Lặp lại ba người chơi và in tên của họ như một phần của danh sách:

```
players = ['charles', 'martina', 'michael', 'florence', 'eli']
print("Here are the first three players on my team:")
for player in players[:3]:
    print(player.title())
```

```
Here are the first three players on my team:
Charles
Martina
Michael
```

Sao chép danh sách

Để sao chép một danh sách, chúng ta có thể tạo một phần bao gồm toàn bộ danh sách gốc bằng cách bỏ qua chỉ mục đầu tiên và chỉ mục thứ hai ([:]).

```
my_foods = ['pizza', 'falafel', 'carrot cake']
friend_foods = my_foods[:]
print("My favorite foods are:")
print(my_foods)
print("\nMy friend's favorite foods are:")
print(friend_foods)
```

```
My favorite foods are:
['pizza', 'falafel', 'carrot cake']

My friend's favorite foods are:
['pizza', 'falafel', 'carrot cake']
```

Sao chép danh sách

Xác thực hai danh sách riêng biệt

```
my_foods = ['pizza', 'falafel', 'carrot cake']
friend_foods = my_foods[:]
my_foods.append('cannoli')
friend_foods.append('ice cream')
print("My favorite foods are:")
print(my_foods)
print("\nMy friend's favorite foods are:")
print(friend_foods)
```

```
My favorite foods are:
['pizza', 'falafel', 'carrot cake', 'cannoli']
My friend's favorite foods are:
['pizza', 'falafel', 'carrot cake', 'ice cream']
```

Sao chép danh sách

Sao chép hai danh sách không sử dụng lát cắt

```
my_foods = ['pizza', 'falafel', 'carrot cake']
# This doesn't work:
friend_foods = my_foods
my_foods.append('cannoli')
friend_foods.append('ice cream')
print("My favorite foods are:")
print(my_foods)
print("\nMy friend's favorite foods are:")
print(friend_foods)
```

```
My favorite foods are:
['pizza', 'falafel', 'carrot cake', 'cannoli', 'ice cream']
My friend's favorite foods are:
['pizza', 'falafel', 'carrot cake', 'cannoli', 'ice cream']
```

Tuples

Danh sách hoạt động tốt để lưu trữ các bộ sưu tập các tin mục có thể thay đổi xuyên suốt vòng đời của một chương trình. Khả năng sửa đổi danh sách là đặc biệt quan trọng khi ta đang làm việc với danh sách người dùng trên trang web hoặc danh sách các ký tự trong một trò chơi.

Tuy nhiên, đôi khi chúng ta muốn tạo một danh sách các mục không thể thay đổi. Tuples cho phép ta làm điều đó. Python đề cập đến các giá trị không thể thay đổi dưới dạng bất biến, và một danh sách không thay đổi được gọi là một tuple.

Định nghĩa một Tuple

Một bộ tuple trông giống như một danh sách ngoại trừ việc ta sử dụng dấu ngoặc đơn thay vì dấu ngoặc vuông.

Khi định nghĩa một bộ tuple, ta có thể truy cập các phần tử riêng lẻ bằng cách sử dụng chỉ mục của từng mục, giống như cách làm đối với danh sách.

```
①   dimensions = (200, 50)
②   print(dimensions[0])
print(dimensions[1])
```

```
200
50
```

Định nghĩa 1 tuple

Hãy thử thay đổi giá trị của một phần tử trong Tuple

```
dimensions = (200, 50)
dimensions[0] = 250
```

```
Traceback (most recent call last):
  File "dimensions.py", line 2, in <module>
    dimensions[0] = 250
TypeError: 'tuple' object does not support item assignment
```

Nếu ta muốn xác định một tuple với một phần tử, ta cần bao gồm một dấu phẩy ở cuối:

```
my_t = (3,)
print(my_t[0])
```

3

Lặp qua toàn bộ giá trị trong Tuple

Lặp lại tất cả các giá trị trong một bộ bằng cách sử dụng vòng lặp for, giống như đã làm với danh sách:

```
dimensions = (200, 50)
for dimension in dimensions:
    print(dimension)
```

```
200
50
```

Ghi lên Tuple

chúng ta không thể sửa đổi tuple, nhưng ta có thể chỉ định một giá trị mới cho một biến đại diện cho một tuple.

Vì vậy, nếu chúng ta muốn thay đổi thứ kích cỡ của hình chữ nhật, chúng ta sẽ định nghĩa lại tuple

```
dimensions = (200, 50)
print("Original dimensions:")
for dimension in dimensions:
    print(dimension)
dimensions = (400, 100)
print("\nModified dimensions:")
for dimension in dimensions:
    print(dimension)
```

```
Original dimensions:
200
50

Modified dimensions:
400
100
```

Lặp trên index

```
>>> supplies = ['pens', 'staplers', 'flamethrowers', 'binders']
>>> for i in range(len(supplies)):
...     print('Index ' + str(i) + ' in supplies is: ' + supplies[i])

Index 0 in supplies is: pens
Index 1 in supplies is: staplers
Index 2 in supplies is: flamethrowers
Index 3 in supplies is: binders
```

Using the enumerate() Function with Lists

Instead of using the `range(len(someList))` technique with a `for` loop to obtain the integer index of the items in the list, you can call the `enumerate()` function instead. On each iteration of the loop, `enumerate()` will return two values: the index of the item in the list, and the item in the list itself. For example, this code is equivalent to the code in the "Using for Loops with Lists" on page 84:

```
>>> supplies = ['pens', 'staplers', 'flamethrowers', 'binders']
>>> for index, item in enumerate(supplies):
...     print('Index ' + str(index) + ' in supplies is: ' + item)

Index 0 in supplies is: pens
Index 1 in supplies is: staplers
Index 2 in supplies is: flamethrowers
Index 3 in supplies is: binders
```

Using the random.choice() and random.shuffle() Functions with Lists

The `random` module has a couple functions that accept lists for arguments. The `random.choice()` function will return a randomly selected item from the list. Enter the following into the interactive shell:

```
>>> import random
>>> pets = ['Dog', 'Cat', 'Moose']
>>> random.choice(pets)
'Dog'
>>> random.choice(pets)
'Cat'
>>> random.choice(pets)
'Cat'
```

Shuffle() - một cách ngẫu nhiên

The `random.shuffle()` function will reorder the items in a list. This function modifies the list in place, rather than returning a new list. Enter the following into the interactive shell:

```
>>> import random
>>> people = ['Alice', 'Bob', 'Carol', 'David']
>>> random.shuffle(people)
>>> people
['Carol', 'David', 'Alice', 'Bob']
>>> random.shuffle(people)
>>> people
['Alice', 'David', 'Bob', 'Carol']
```

Bài tập

4-1. Pizzas: Think of at least three kinds of your favorite pizza. Store these pizza names in a list, and then use a `for` loop to print the name of each pizza.

- Modify your `for` loop to print a sentence using the name of the pizza instead of printing just the name of the pizza. For each pizza you should have one line of output containing a simple statement like *I like pepperoni pizza*.
- Add a line at the end of your program, outside the `for` loop, that states how much you like pizza. The output should consist of three or more lines about the kinds of pizza you like and then an additional sentence, such as *I really love pizza!*

4-2. Animals: Think of at least three different animals that have a common characteristic. Store the names of these animals in a list, and then use a `for` loop to print out the name of each animal.

- Modify your program to print a statement about each animal, such as *A dog would make a great pet.*
- Add a line at the end of your program stating what these animals have in common. You could print a sentence such as *Any of these animals would make a great pet!*

Bài tập

4-3. Counting to Twenty: Use a `for` loop to print the numbers from 1 to 20, inclusive.

4-4. One Million: Make a list of the numbers from one to one million, and then use a `for` loop to print the numbers. (If the output is taking too long, stop it by pressing CTRL-C or by closing the output window.)

4-5. Summing a Million: Make a list of the numbers from one to one million, and then use `min()` and `max()` to make sure your list actually starts at one and ends at one million. Also, use the `sum()` function to see how quickly Python can add a million numbers.

4-6. Odd Numbers: Use the third argument of the `range()` function to make a list of the odd numbers from 1 to 20. Use a `for` loop to print each number.

4-7. Threes: Make a list of the multiples of 3 from 3 to 30. Use a `for` loop to print the numbers in your list.

4-8. Cubes: A number raised to the third power is called a *cube*. For example, the cube of 2 is written as `2**3` in Python. Make a list of the first 10 cubes (that is, the cube of each integer from 1 through 10), and use a `for` loop to print out the value of each cube.

4-9. Cube Comprehension: Use a list comprehension to generate a list of the first 10 cubes.

XIN CẢM ƠN!