

# Chương 3 - Danh sách (List)

---

NGUYỄN HOÀNG ANH

CNTT1. PTIT



# 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

## 3.1. 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])
```

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

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.
```

## 3.2. 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 người ngoài hành tinh ban đầu trong một danh sách và sau đó xóa một người ngoài hành tinh 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. Hành động 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.
```

## 3.3. 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 đổi 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*

## 3.4. 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 xảy ra lỗi trong trường hợp duy nhất đó là khi danh sách trống

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

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

## 3.5. Lặp qua toàn bộ danh sách

- ❑ Một tác vụ quan trọng mà chương trình thường phải làm là chạy qua toàn bộ các phần tử trong danh sách và thực hiện các việc giống nhau.
- ❑ Ví dụ: trong một trò chơi, ta có thể muốn di chuyển mọi phần tử trên màn hình theo cùng một khoảng hoặc trong danh sách các số ta có thể muốn thực hiện cùng một thao tác thống kê trên mọi phần tử, hoặc khi muốn hiển thị các Tiêu đề của các bài báo trong danh sách.
- ❑ Khi ta muốn thực hiện cùng một hành động với mọi mục trong danh sách, ta có thể sử dụng vòng lặp for của Python.

# Lặp qua toàn bộ danh sách(t)

---

❑ Dùng một vòng for để in ra mỗi tên trong danh sách

```
magicians = ['alice', 'david', 'carolina']
```

```
for magician in magicians:
```

```
    print(magician)
```

```
alice
```

```
david
```

```
carolina
```



# Cận cảnh về vòng lặp for

- ❑ Khi chúng ta sử dụng vòng lặp lần đầu tiên, hãy nhớ rằng tập các bước được lặp lại một lần cho mỗi mục trong danh sách, bất kể có bao nhiêu mục có trong danh sách.
- ❑ Khi viết các vòng lặp, có thể chọn bất kỳ tên nào mong muốn cho biến tạm thời sẽ được liên kết với mỗi giá trị trong danh sách.
- ❑ Sẽ hữu ích khi chọn một cái tên có ý trong vòng for nghĩa đại diện cho một mục duy nhất từ danh sách

```
for cat in cats:
```

```
for dog in dogs:
```

```
for item in list_of_items:
```

# Bổ sung thêm về vòng lặp

Có thể làm bất cứ điều gì với mỗi mục trong vòng lặp for

```
magicians = ['alice', 'david', 'carolina']  
for magician in magicians:  
    print(f"{magician.title()}, that was a great trick!")
```

Alice, that was a great trick!

David, that was a great trick!

Carolina, that was a great trick!

# Bổ sung thêm về vòng lặp

---

Thêm dòng thứ hai vào thông điệp, nói với mỗi nhà ảo thuật rằng chúng ta đang mong chờ trò ảo thuật tiếp theo của họ

```
magicians = ['alice', 'david', 'carolina']  
for magician in magicians:  
    print(f"{magician.title()}, that was a great trick!")  
    print(f"I can't wait to see your next trick, {magician.title()}.\n")
```

Alice, that was a great trick!

I can't wait to see your next trick, Alice.

David, that was a great trick!

I can't wait to see your next trick, David.

Carolina, that was a great trick!

I can't wait to see your next trick, Carolina.

# Một số việc sau vòng lặp For

Bất kỳ dòng mã nào sau vòng lặp for không được thực thi một lần mà không lặp lại.

```
magicians = ['alice', 'david', 'carolina']
for magician in magicians:
    print(f"{magician.title()}, that was a
    great trick!")

    print(f"I can't wait to see your next
    trick, {magician.title()}.\n")
print("Thank you, everyone. That was a great
magic show!")
```

Alice, that was a great trick!

I can't wait to see your next trick, Alice.

David, that was a great trick!

I can't wait to see your next trick, David.

Carolina, that was a great trick!

I can't wait to see your next trick, Carolina.

Thank you, everyone. That was a great magic show!

# Tránh lỗi thụt lề

Về cơ bản, python sử dụng khoảng trắng để buộc viết mã có định dạng gọn gàng với một cấu trúc trực quan rõ ràng. Trong các chương trình Python dài hơn, các khối mã được thụt lề ở một vài cấp độ khác nhau. Các mức thụt lề này giúp có được cảm nhận chung về tổ chức chung của chương trình.

Khi bắt đầu viết mã dựa vào thụt lề thích hợp, ta sẽ cần để ý một vài lỗi thụt lề phổ biến. Ví dụ, đôi khi thụt lề dòng mã không cần phải thụt lề hoặc quên để thụt lề các dòng cần thụt lề.

# Quên thụt lề

---

Luôn thụt lề dòng sau câu lệnh for trong một vòng lặp. Nếu quên, Python sẽ nhắc nhở ta

```
magicians = ['alice', 'david', 'carolina']  
for magician in magicians:  
    print(magician)
```

File "magicians.py", line 3

```
print(magician)
```

^

IndentationError: expected an indented block

# Quên thụt lề các dòng bổ sung

---

❑ Vòng lặp chạy mà không có bất kỳ lỗi nào nhưng sẽ không tạo ra kết quả mong đợi. Điều này có thể xảy ra khi ta đang cố gắng thực hiện một số tác vụ trong một vòng lặp và quên thụt lề một số dòng của nó.

```
magicians = ['alice', 'david', 'carolina']  
for magician in magicians:  
    print(f"{magician.title()}, that was a great trick!")  
print(f"I can't wait to see your next trick, {magician.title()}.\n")
```

Alice, that was a great trick!

David, that was a great trick!

Carolina, that was a great trick!

I can't wait to see your next trick, Carolina.

# Thụt lề không cần thiết

---

Nếu vô tình thụt lề một dòng không cần phải thụt lề, Python sẽ thông báo về sự thụt lề không mong muốn:

```
message = "Hello Python world!"  
    print(message)
```

File "hello\_world.py", line 2

```
print(message)
```

^

IndentationError: unexpected indent



# Thụt lề không cần thiết sau vòng lặp for

---

Nếu vô tình thụt lề mã sẽ chạy sau khi một vòng lặp kết thúc, mã sẽ được lặp lại một lần cho mỗi mục trong danh sách.

```
magicians = ['alice', 'david', 'carolina']
for magician in magicians:
    print(f"{magician.title()}, that was a great
    trick!")
    print(f"I can't wait to see your next trick,
    {magician.title()}.\\n")
    print("Thank you everyone, that was a great
    magic show!")
```

Alice, that was a great trick!

I can't wait to see your next trick, Alice.

Thank you everyone, that was a great magic show!

David, that was a great trick!

I can't wait to see your next trick, David.

Thank you everyone, that was a great magic show!

Carolina, that was a great trick!

I can't wait to see your next trick, Carolina.

Thank you everyone, that was a great magic show!

# Quên dấu hai chấm ':'

---

Dấu hai chấm ở cuối câu lệnh for yêu cầu Python giải thích câu lệnh tiếp theo dòng là điểm bắt đầu của một vòng lặp

```
magicians = ['alice', 'david', 'carolina']  
for magician in magicians  
    print(magician)
```

```
for magician in magicians
```

^

```
SyntaxError: invalid syntax
```

## 3.6. 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.

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

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

[1, 2, 3, 4, 5]

```
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]
```



## 3.7. 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']    ['charles', 'martina', 'michael']  
print(players[0:3])
```

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

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']

## 3.8. 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

---

## 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])
```

# 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

<code>dimensions = (200, 50)</code>	<code>Original dimensions:</code>
<code>print("Original dimensions:")</code>	<code>200</code>
<code>for dimension in dimensions:</code>	<code>50</code>
<code>    print(dimension)</code>	
<code>    dimensions = (400, 100)</code>	<code>Modified dimensions:</code>
<code>print("\nModified dimensions:")</code>	<code>400</code>
<code>for dimension in dimensions:</code>	<code>100</code>
<code>    print(dimension)</code>	

# Kết chương

- ❑ Trong chương này, chúng ta đã học cách làm việc hiệu quả với các phần tử trong danh sách:
  - Cách làm việc thông qua danh sách bằng vòng lặp for
  - Cách Python sử dụng thụt lề để cấu trúc một chương trình và cách tránh một số lỗi thụt lề.
  - Cách tạo danh sách số đơn giản, cũng như một số thao tác có thể thực hiện trên danh sách số.
  - Cách cắt lát danh sách để làm việc với một tập hợp con các phần tử và cách sao chép danh sách đúng cách bằng cách sử dụng lát cắt.
  - Tìm hiểu về tuple, cung cấp mức độ bảo vệ đến một tập hợp các giá trị không được thay đổi.
- ❑ Chương tiếp theo, chúng ta học cách phản ứng thích hợp với các điều kiện khác nhau bằng cách sử dụng câu lệnh if.

# Bài tập