HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
Posts and Telecommunications Institute of Technology

CHƯƠNG 3: DANH SÁCH (LIST)

VŨ HOÀI THƯ
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

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 thi các tác vụ giống nhau.
- Khi ta muốn thực hiện cùng một hành động với mọi phần tử của danh sách, ta có thể sử dụng vòng lặp for của Python.

LẶP QUA TOÀN BỘ DANH SÁCH

- Ví dụ: Dùng một vòng for để in ra toàn bộ các phần tử có trong danh sách:

```
# In ra các phần tử có trong danh sách
singers = ['alice', 'david', 'carolina']

for singer in singers:

    print(singer)
```

```
alice
david
carolina
```

CẬN CẢNH VỀ VÒNG LẶP FOR

- Khi sử dụng vòng lặp, các bước được lặp lại một lần cho mỗi phần tử trong danh sách, bất kể có bao nhiêu phần tử trong danh sách.
- Khi viết các vòng lặp, có thể chọn một tên bất kỳ cho biến tạm thời sẽ được liên kết với mỗi giá trị trong danh sách.
- Nên chọn tên biến có ý nghĩa trong vòng for đại diện cho mỗi phần tử trong danh sách.

```
for cat in cats:

for dog in dogs:

for item in list_of_items:
```

BỔ SUNG THÊM VỀ VÒNG LẶP

- Có thể thực hiện bất cứ tác vụ gì với mỗi mục trong vòng lặp for.
- Ví dụ:

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

for singer in singers:
    print(f"{singer.title()}, that was a great singer!")
```

```
Alice, that was a great singer!
David, that was a great singer!
Carolina, that was a great singer!
```

```
numbers = [1, 5, 8, 6, 7, 6, 2, 9]
for num in numbers:
    print(num**3)
```

```
1
125
512
216
343
216
8
729
```

MỘT SỐ CHÚ Ý VỀ VÒNG LẶP FOR

- Bất kỳ dòng mã nào sau vòng lặp for không được thụt lề đều chỉ 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 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 các cấp độ khác nhau.
- Khi mới bắt đầu viết chương trình, cần để ý một vài lỗi thụt lề phổ biến. Ví dụ: quên thụt lề, hoặc thụt lề các dòng mã không cần thụt lề,...

TRÁNH LỖI THỤT LỀ - QUÊN THỤT LỀ

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

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

for singer in singers:

print(singer)
```

```
  File "C:\Users\HoaiT\AppData\Local\Temp/ipykernel_10988/1243018688.py", line 6
    print(singer)
    ^
IndentationError: expected an indented block
```

TRÁNH LỖI THỤT LỀ - 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"

    print(message)
```

```
  File "C:\Users\HoaiT\AppData\Local\Temp/ipykernel_10988/1963583163.py", line 3
    print(message)
    ^
IndentationError: unexpected indent
```

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

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

for singer in singers
    print(singer)
```

```
  File "C:\Users\HoaiT\AppData\Local\Temp/ipykernel_10988/552604324.py", line 3
    for singer in singers
                       ^
SyntaxError: invalid syntax
```

LẬP DANH SÁCH SỐ

- Danh sách là nơi lý tưởng để lưu trữ các tập hợp số và Python cung cấp nhiều công cụ để giúp 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, dùng hàm range(1,6)

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

```
1
2
3
4
5
```

SỬ DỤNG HÀM RANGE() ĐỂ TẠO 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()

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

print(numbers)
```

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

- Khi bọc list() xunh quanh một hàm range(), đầu ra sẽ là một danh sách số
- Nếu truyền đối số thứ ba vào range(), Python 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 HÀM RANGE() ĐỂ TẠO DANH SÁCH SỐ

- Tạo danh sách các số bình phương tới 10

```
squares = []

for value in range(1, 11):
    square = value ** 2
    squares.append(square)

print(squares)
```

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

```
squares = []

for value in range(1,11):
    squares.append(value**2)

print(squares)
```

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

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]
```

```
1 min(digits)
```

```
0
```

```
1 max(digits)
```

```
9
```

```
1 sum(digits)
```

```
45
```

HIỂU VỀ DANH SÁCH

- Ví dụ xây dựng cùng một danh sách các số bình phương sử dụng khả năng hiểu danh sách.

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

print(squares)
```

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

- 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

BÀI TẬP

- Bài tập 1: Cho một danh sách số. Viết chương trình in ra các số có số lần xuất hiện lớn hơn hoặc bằng 3 (lần).

  Ví dụ: list_numbers = [1, 5, 1, 8, 4, 1, 4, 3, 2, 8, 4, 9]

  => Kết quả [1, 4]

- Bài tập 2: Cho một danh sách gồm các chuỗi. Viết chương trình in ra các chuỗi bắt đầu bằng chữ ‘p’ hoặc ‘P’.

  Ví dụ: list_strings = [‘python’, ‘programming’, ‘language’, ‘Python’, ‘most’]

  => Kết quả [‘python’, ‘programming’ ,‘Python’]

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. Có thể xuất bất kỳ lát nào từ cuối danh sách.

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

→ `['michael', 'florence', 'eli']`

LẶP QUA MỘT LÁT CẮT

- Lặp lại ba người chơi và in ra 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 ([:])

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

copy_players = players[:]

print(copy_players)
```

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

SAO CHÉP DANH SÁCH

- Sao chép hai danh sách không sử dụng lát cắt

```
players = ['charles', 'martina', 'michael', 'florence', 'eli']
copy_players = players.copy()
print(copy_players)
```

→ ['charles', 'martina', 'michael', 'florence', 'eli']

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

→ ['charles', 'martina', 'michael', 'florence', 'eli']

TUPLE

- 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.
-

```
dimensions = (200, 50)

print(dimensions[0])

print(dimensions[1])
```

```
200
50
```

LẶP QUA TOÀN BỘ GIÁ TRỊ TRONG TUPLE

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

```
dimensions = (200, 50)

dimensions[0] = 250
```

```
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_10988/4275015717.py in <module>
      1 dimensions = (200, 50)
      2
----> 3 dimensions[0] = 250

TypeError: 'tuple' object does not support item assignment
```

- 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
```

BÀI TẬP

- Bài tập 5: Cho một danh sách gồm các tên trang web sau:

["www.zframez.com", "www.wikipedia.org", "www.asp.net", "www.abcd.in"].

Viết chương trình in ra các hậu tố từ tên các trang web.

⇒ ***Kết quả: [‘com’, ‘org’, ‘net’, ‘in’]***

- Bài tập 6: Cho một danh sách gồm các tuple. Hãy sắp xếp danh sách theo phần tử thứ 2 của tuple.

Ví dụ: list_tuples = [('a', 23),('b', 37),('c', 11), ('d',29)]

Kết quả: [('c', 11), ('a', 23), ('d',29), ('b', 37)]

BÀI TẬP

- Bài tập 3: Cho một danh sách số. Viết chương trình đếm các số nguyên tố có trong danh sách.

- Bài tập 4: Cho một danh sách số.Viết chương trình tính tổng tích lũy của một danh sách, nghĩa là, kết quả là một danh sách mới trong đó phần tử thứ i là tổng của i+1 phần tử đầu tiên từ danh sách ban đầu.

  Ví dụ: numbers = [1, 2, 3] => Kết quả results= [1, 3, 6]

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.