

Basic Python - List Comprehension

Hoàng-Nguyên Vũ

1. Mô tả:

- **List comprehension** là một cú pháp ngắn gọn và mạnh mẽ trong Python để tạo ra một danh sách mới từ một danh sách hoặc tập hợp dữ liệu có sẵn. Cú pháp này giúp bạn tiết kiệm thời gian và viết code ngắn gọn hơn so với việc sử dụng vòng lặp for truyền thống.

– Ưu điểm:

- + **Giảm thiểu số lượng code:** giúp code ngắn gọn và dễ đọc hơn so với sử dụng vòng lặp for truyền thống.
- + **Dễ sử dụng:** có cú pháp đơn giản và dễ học.

– Nhược điểm:

- + **Khó đọc:** có thể khó đọc và khó hiểu hơn so với vòng lặp for truyền thống trong một số trường hợp.
- + **Hạn chế về chức năng:** không thể thực hiện một số thao tác mà vòng lặp for truyền thống có thể làm được

Cấu trúc của List comprehension như sau:



2. **Bài tập:** Trong NLP, chúng ta cần loại bỏ 1 số từ không quan trọng (stopwords) ra khỏi câu để tránh gây nhiễu trong việc xử lý. Hãy loại bỏ các từ có trong `stop_words = ["I", "love", "and", "to"]` câu đầu vào `"I love AI and listen to music"`. Hãy áp dụng **List comprehension** và For truyền thống để thực hiện

```
1 stop_words = ["I", "love", "and", "to"]
2 input = "I love AI and listen to music"
3 # Your code here
```

Output: `['AI', 'listen', 'music']`

Basic Python - List - Tuple

Trung-Trúc Trần và Hoàng-Nguyên Vũ

1. Giới thiệu:

- Tuple là một kiểu dữ liệu cơ bản trong Python, được sử dụng để lưu trữ tập hợp các phần tử có thứ tự. Tuple có thể chứa bất kỳ kiểu dữ liệu nào, bao gồm số nguyên, chuỗi, số thập phân, danh sách con, v.v.

2. Mô tả:

Bảng 1: Sự Giống Nhau và Khác Nhau giữa Tuple và List

Đặc Điểm	Tuple	List
Đặc Điểm Cơ Bản	Tập hợp các phần tử không thay đổi được, đặt trong cặp dấu ngoặc đơn.	Tập hợp các phần tử có thể thay đổi được, đặt trong cặp dấu ngoặc vuông.
Thay Đổi Dữ Liệu	Không thể thay đổi (immutable). Không thể thêm, xóa hoặc thay đổi các phần tử.	Có thể thay đổi (mutable). Có thể thêm, xóa và thay đổi các phần tử.
Sử Dụng	Thích hợp để bảo vệ dữ liệu.	Thích hợp cho các cấu trúc dữ liệu có thể thay đổi.
Hiệu Suất	Trong một số trường hợp, tuple có thể nhanh hơn	List có sự linh hoạt cao hơn.

3. Bài tập: Khởi tạo Tuple và thao tác tìm kiếm, trích xuất thông tin trên Tuple đó.

- Câu 1:** Tạo mới hai Tuple: `my_tuple1 = (2,3)`, `my_tuple2 = (3,6)` mỗi Tuple có 2 phần tử đại diện cho một vector trong không gian 2D.
- Câu 2:** In ra kết quả của tổng và tích 2 vector trên.
- Câu 3:** In ra kết quả của khoảng cách của hai vector trên theo công thức. Biết $distance(P, Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2}$
- Câu 4:** In ra vị trí của phần tử có giá trị là 3
Sử dụng cú pháp: `my_tuple.index(values)` để trích xuất vị trí của giá trị cần tìm.

```
1 my_tuple1 = ()
2 my_tuple2 = ()
3 # Your code here
```

Output:

- Câu 2: `Result_vector1=(5,6)`, `Result_vector2=(9,18)`
- Câu 3: `√10=3.1622776601683795`.
- Câu 4: `index=(1, 0)`.