Bài 12: KIỂU DỮ LIỆU LIST TRONG PYTHON (Phần 1)

Xem bài học trên website để ủng hộ Kteam: KIỂU DỮ LIỆU LIST TRONG PYTHON (Phần 1)

Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!

Dẫn nhập

Trong những bài trước, Kteam đã giới thiệu đến bạn loạt bài về KIỂU DỮ LIỆU CHUỖI trong Python gồm rất nhiều kiến thức chi tiết và dễ hiểu nhất có thể.

Sang bài này, chúng ta sẽ cùng nhau tìm hiểu một trong những kiểu dữ liệu cực kỳ quan trọng trong Python. Đó chính là **KIỂU DỮ LIỆU LIST**

Nội dung

Để đọc hiểu bài này tốt nhất bạn cần:

- Cài đặt sẵn MÔI TRƯỜNG PHÁT TRIỂN CỦA PYTHON.
- Xem qua bài CÁCH CHẠY CHƯƠNG TRÌNH PYTHON.
- Nắm CÁCH GHI CHÚ và BIẾN TRONG PYTHON.
- KIỂU DỮ LIỆU SỐ và KIỂU DỮ LIỆU CHUỖI trong Python.

Bạn và Kteam sẽ cùng tìm hiểu những nội dun g sau đây

- Container. Đặt vấn đề và cách giải quyết
- Giới thiệu về List trong Python
- Cách khởi tạo List
- Một số toán tử với List trong Python
- Indexing và cắt List trong Python
- Thay đổi nội dung List trong Python
- Ma trận
- Vấn đề cần lưu tâm khi sử dụng List
- Củng cố bài học

Container. Đặt vấn đề và cách giải quyết

Các bạn đã biết đến **BIẾN** (đã giới thiệu trong bài BIẾN TRONG PYTHON), đó là một **container** cho phép ta lưu trữ các dữ liệu và lấy ra khi cần, thay đổi khi ta cần cập nhật giá trị hoặc sửa chữa.

Nhưng, khả năng của biến vẫn bị giới hạn! Đơn giản với một ví dụ, ta cần biến **teo** lưu cho ta giá trị là chuỗi `"Teo"` là tên của Tèo, và tuổi của Tèo là số 17.

```
>>> teo = "Teo"
>>> teo
'Teo'
>>> teo = 17
>>> teo
17
```

Biến **teo** của chúng ta không thể lưu hai giá trị một lúc. Không chỉ tên và tuổi, Tèo còn rất nhiều thông tin muốn lưu vào biến **teo** nữa như ngày sinh của gấu, số lần fix bug trong một tháng, khóa học mới coi gần nhất, số lần tè dầm ở tuổi 17,...

Với năng lực của một người mới học lập trình, sáng kiến tối ưu nhất họ đưa ra là mỗi giá trị ta có một biến riêng biệt. Và đây được coi là một giải pháp hay!

Tuy nhiên vẫn ở tầm vi mô. Tèo nó tham, muốn lưu cả mấy thứ linh tinh về cô gấu dễ thương của hắn. Lúc đó, việc bạn kêu Tèo tạo ra số lượng biến để lưu trữ cũng là một điều gian nan rồi.

Đó là vì sao ta cần một thứ cũng như biến, nhưng nội công lại thâm hậu hơn biến, có khả năng lưu trữ nhiều giá trị cùng một lúc. Vì thế, Python có rất nhiều các container cho phép ta lưu trữ nhiều các giá trị, đối tượng cùng một lúc, hỗ trợ cho chúng ta trong việc truy xuất, tính toán, thay đổi (một số container trong Python không hỗ trợ việc thay đổi),...

Trong các ngôn ngữ lập trình khác, những container chứa được nhiều giá trị cùng một lúc thường được gọi là **ARRAY** (mảng).

Giới thiệu về List trong Python

LIST là một container được sử dụng rất nhiều trong các chương trình Python. Một List gồm các yếu tố sau:

- Được giới hạn bởi cặp ngoặc **[ ]**, tất cả những gì nằm trong đó là những phần tử của List.
- Các phần tử của List được phân cách nhau ra bởi dấu phẩy (**,**).
- List có khả năng chứa mọi giá trị, đối tượng trong Python. Và bao gồm chứa chính nó! (một trường hợp hay ho Kteam sẽ giới thiệu ở phần khác).

Ví dụ:

```
>>> [1, 2, 3, 4, 5] # Một List chứa 5 số nguyên
[1, 2, 3, 4, 5]
>>> ['a', 'b', 'c', 'd'] # Một List chứa 4 chuỗi
['a', 'b', 'c', 'd']
>>> [[1, 2], [3, 4]] # Một List chứa 2 List là [1, 2] và [3, 4]
[[1, 2], [3, 4]]
>>> [1, 'one', [2, 'two']] # List chứa số nguyên, chuỗi, và List
[1, 'one', [2, 'two']]
```

Cách khởi tạo List

Sử dụng cặp dấu ngoặc [] đặt giá trị bên trong

Cú pháp:

> **[<giá trị thứ nhất>, <giá trị thứ hai>, .., <giá trị thứ n – 1>, <giá trị thứ n>]**

Ví dụ:

```
>>> lst = [1,2,5,"kteam"]
>>> lst
[1, 2, 5, 'kteam']
>>> empty_list = []  # khởi tạo list rỗng
>>> empty_list
[]
```

Sử dụng List Comprehension

Cú pháp

> **[Comprehension]**

Ví dụ:

```
>>> a = [kteam for kteam in range(3)]
>>> a
[0, 1, 2]
>>> another_lst = [[n, n * 1, n * 2] for n in range(1, 4)]
>>> another_lst
[[1, 1, 2], [2, 2, 4], [3, 3, 6]]
```

List **comprehension** là một cách khởi tạo một List rất thú vị trong Python. Do đó, rất khó để có thể nói hết các trường hợp. Vì vậy, hãy tạm gác lại kiến thức này, bạn không cần phải cố gắng hiểu nó khi chúng ta chưa gặp gỡ các vòng lặp.

Sử dụng constructor List

Cú pháp:

> **list (iterable)**

Lưu ý: iterable là một đối tượng nói chung của các container. Khái niệm này sẽ được Kteam giới thiệu ở bài sau. Đối với bạn khi theo dõi khóa học này của Kteam, bạn đã được biết hai iterable đó chính là chuỗi, và List.

Ví dụ:

```
>>> lst = list([1, 2, 3])
>>> lst
[1, 2, 3]
>>> str_lst = list('HOWKTEAM')
>>> str_lst
['H', 'O', 'W', 'K', 'T', 'E', 'A', 'M']
>>> list(1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'int' object is not iterable
```

Một số toán tử với List trong Python

Các toán tử của List gần giống và tương tự với chuỗi (bạn có thể tham khảo toán tử của chuỗi ở bài KIỂU DỮ LIỆU CHUỖI – phần 2).

Toán tử +

```
>>> lst = [1, 2]
>>> lst += ['one', 'two']
>>> lst
[1, 2, 'one', 'two']
>>> lst += 'abc'  # cộng List và chuỗi
>>> lst
[1, 2, 'one', 'two', 'a', 'b', 'c']
>>> 'abc' + [1, 2]  # List cộng chuỗi cho phép, chuỗi cộng List thì không.
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: must be str, not list
```

Toán tử *

```
>>> lst = list('KTER') * 2
>>> lst
['K', 'T', 'E', 'R', 'K', 'T', 'E', 'R']
>>> [1, 2] * 3
[1, 2, 1, 2, 1, 2]
```

Toán tử in

```
>>> 'a' in [1, 2, 3]
False
>>> 'a' in ['a', 2, 3]
True
>>> 'a' in [['a'], 'b', 'c'] # chỉ có ['a'] thôi, không có 'a'
False
```

Indexing và cắt List trong Python

Như đã đề cập, List với chuỗi giống nhau rất nhiều điểm, và phần Indexing và cắt List này hoàn toàn giống với Indexing và cắt chuỗi. (Nếu chưa biết về chuỗi bạn có thể tham khảo qua các bài về KIỂU DỮ LIỆU CHUỖI TRONG PYTHON – Phần 1)

```
>>> lst = [1, 2, 'a', 'b', [3, 4]]
```

```
>>> lst[0]
1
>>> lst[-1]
[3, 4]
>>> lst[3]
'b'
>>> lst[1:3]
[2, 'a']
>>> lst[:2]
[1, 2]
>>> lst[2:]
['a', 'b', [3, 4]]
>>> lst[::-1]
[[3, 4], 'b', 'a', 2, 1]
```

Thay đổi nội dung List trong Python

Như bạn đã biết, ta không thể thay đổi nội dung của chuỗi như ví dụ bên dưới

```
>>> s = 'math'
>>> s[1]
'a'
>>> s[1] = 'i'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

Còn về phần List, ta có thể thay đổi nội dung của nó

```
>>> lst = [1, 'two', 3]
>>> lst[1]
'two'
>>> lst[1] = 2
>>> lst
[1, 2, 3]
```

Ma trận

Nghe ma trận hoàn tráng thế thôi, bạn đã thấy nó rồi. Ví dụ một List chứa một List khác đấy.

```
>>> lst = [[1, 2, 3], [4, 5, 6]]
>>> lst
[[1, 2, 3], [4, 5, 6]]
```

Ta dễ dàng truy cập hai phần tử của List vừa mới khởi tạo

```
>>> lst[0]
[1, 2, 3]
>>> lst[-1]
[4, 5, 6]
```

Hai giá trị đó cũng là một List. Và lẽ dĩ nhiên, bạn có quyền truy cập đến các phần tử con của phần tử nằm trong List bạn vừa khởi tạo. Thậm chí là cắt List!

```
>>> lst[0][0]
1
>>> lst[0][-1]
3
>>> lst[1][1]
5
>>> lst[0][:2]
[1, 2]
>>> lst[1][:]
[4, 5, 6]
```

Vấn đề cần lưu tâm khi sử dụng List

Những lưu ý này nếu bạn không biết, chương trình của bạn có thể có output khác với bạn mong muốn.

> **Không được phép gán List này qua List kia nếu không có chủ đích**

Hãy xem xét đoạn code sau đây

```
>>> lst = [1, 2, 3]
>>> another_lst = lst
>>> lst
[1, 2, 3]
>>> lst = [1, 2, 3]
>>> lst
[1, 2, 3]
>>> another_lst = lst
>>> another_lst
[1, 2, 3]
>>> lst
[1, 2, 3]
```

Mọi thứ ổn, không có gì xảy ra, cho tới khi bạn thay đổi giá trị bất kì của một trong hai List đó.

```
>>> another_lst[1]
2
>>> another_lst[1] = 'Two'
>>> another_lst
[1, 'Two', 3]
>>> lst
[1, 'Two', 3]
```

Chỉnh một, nhưng đổi tới hai. Lí do là vì khi bạn gán giá trị List trực tiếp như thế, bạn đang đưa hai List đó trỏ cùng vào một nơi.

Hãy tưởng tượng Tèo có 50 nghìn. Sau đó bạn sử dụng phép thuật của mình gán số tiền cô gấu của Tèo bằng số tiền của Tèo. Khi đó, cô gấu dễ thương của Tèo không tự nhiên mà có 50 nghìn, mà thay vào đó, bạn đã gián tiếp cho phép gấu của Tèo sử dụng số tiền của Tèo nhịn ăn mì tôm bấy lâu nay. Và vào một ngày trơi mưa không rơi, cô ấy chạy đi mua một gói Snack mất 5 nghìn và sử dụng số tiền 50 nghìn bạn vừa mới gán cho cô ấy. Hậu quả là Tèo về thấy mất đâu 5 nghìn.

Do đó, trước khi gán, bạn phải copy giá trị của List

```
>>> lst = [1, 2, 3]
>>> lst
```

```
[1, 2, 3]
>>> another_list = list(lst) # cách 1
>>> another_list
[1, 2, 3]
>>> last_list = lst[:] # cách 2
>>> last_list
[1, 2, 3]
>>> another_list[1] = 'Two'
>>> last_list[1] = 'Chu'
>>> lst
[1, 2, 3]
>>> another_list
[1, 'Two', 3]
>>> last_list
[1, 'Chu', 3]
```

Thêm một trường hợp nữa bạn cần phải lưu ý, đó là lúc bạn cần copy giá trị của một ma trận

```
>>> lst = [[1, 2, 3], [4, 5, 6]]
>>> lst
[[1, 2, 3], [4, 5, 6]]
>>> another_lst = lst[:]
>>> another_lst[0] = 'ok'
>>> lst
[[1, 2, 3], [4, 5, 6]]
>>> another_lst
['ok', [4, 5, 6]]
```

Đúng như chúng ta mong đợi. Thế nhưng...

```
>>> another_lst[1]
[4, 5, 6]
>>> another_lst[1][0] = 'changed'
>>> another_lst
['ok', ['changed', 5, 6]]
>>> lst
[[1, 2, 3], ['changed', 5, 6]]
```

Lưu ý: nó chỉ sao chép các phần tử của List. Không hề sao chép các phần tử con của các phần tử nằm trong List. Do đó, nếu bạn thay đổi các phần tử trong List thì không sao, tuy nhiên nếu thay đổi phần tử con của các phần tử trong List, thì vấn đề lại xuất hiện.

Đương nhiên, bạn vẫn giải quyết được, nhưng vì rườm rà khi không có vòng lặp. Do đó, chúng ta tạm dừng ở việc nhận biết. Còn phần giải quyết sẽ đợi nay mai. Khi võ công Xà Ngữ của chúng ta được nâng cao và tiếp cận với tuyệt kĩ vòng lặp.

Củng cố bài học

Đáp án bài trước

Bạn có thể tìm thấy câu hỏi của phần này tại CÂU HỎI CỦNG CỐ trong bài KIỂU DỮ LIỆU CHUỖI TRONG PYTHON – Phần 5

- Cách đơn giản

```
>>> s = s.lower()
>>> s = s.strip('a')
>>> s = s.lstrip('ao')
>>> s = s.title()
>>> s
'Neu Mot Ngay Nao Do'
```

- Cách ngắn

```
>>> s = s.lower().strip('a').lstrip('ao').title()
>>> s
'Neu Mot Ngay Nao Do'
```

Câu hỏi củng cố

1. Tìm các cách khởi tạo List hợp lệ dưới đấy
    a. list(list(list('abc'))
    b. [1, 2, 3] + list(4)
    c. list()
    d. [0] * 3
2. List có phải là một hashable object (immutable object)?
3. Với chuỗi s dưới đây

```
s = 'aaaaaaaAAAAAaaa//123123//000000//&&TTT%%abcxyznontqfadf'
```

Hãy lấy mật mã trong chuỗi s, biết mật mã nằm giữa && và %%. Cố gắng tối thiểu dòng code

Đáp án của phần này sẽ được trình bày ở bài tiếp theo. Tuy nhiên, Kteam khuyến khích bạn tự trả lời các câu hỏi để củng cố kiến thức cũng như thực hành một cách tốt nhất!

Kết luận

Bài viết này đã sơ lược cho các bạn KIỂU DỮ LIỆU LIST TRONG PYTHON.

Ở bài sau, Kteam sẽ tiếp tục nói về KIỂU DỮ LIỆU LIST TRONG PYTHON - Phần 2. Cụ thể là một số phương thức của List trong Python

Cảm ơn bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".