**Giới thiệu về Tuple trong Python**

**Tuple** là một container cũng được sử dụng rất nhiều trong các chương trình Python không thua kém gì List. (List đã được giới thiệu trong bài [KIỂU DỮ LIỆU LIST TRONG PYTHON](http://www.howkteam.vn/course/kieu-du-lieu-list-trong-python--phan-2/kieu-du-lieu-list-trong-python--phan-1-1548))

Một Tuple gồm các yếu tố sau:

* Được giới hạn bởi cặp ngoặc **()**, tất cả những gì nằm trong đó là những phần tử của Tuple.
* Các phần tử của Tuple được phân cách nhau ra bởi dấu phẩy (**,**).
* Tuple có khả năng chứa mọi giá trị, đối tượng trong Python.

**Ví dụ:**

>>> (1, 2, 3, 4, 5) # Một Tuple chứa 5 số nguyên

(1, 2, 3, 4, 5)

>>> ('k', 't', 'e', 'r') # Một Tuple chứa 4 chuỗi

('k', 't', 'e', 'r')

>>> ([1, 2], (3, 4)) # Một Tuple chứa 1 List là [1, 2] và 1 Tuple là (3, 4)

([1, 2], (3, 4))

>>> (1, 'kteam', [2, 'k9']) # Tuple chứa số nguyên, chuỗi, và List

(1, 'kteam', [2, 'k9'])

**Cách khởi tạo Tuple**

**Sử dụng cặp  dấu ngoặc () và 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:**

>>> tup = (1, 2, 3, 4)

>>> tup

(1, 2, 3, 4)

>>> empty\_tup = () # khởi tạo tuple rỗng

>>> empty\_tup

()

>>> type(tup) # kiểu dữ liệu Tuple thuộc lớp tuple

<**class** '**tuple**'>

Bạn hãy chú ý khi khởi tạo tuple với một giá trị.

>>> tup = (9) # Tuple có một giá trị là số 9

>>> tup # có kết quả lạ

9

>>> type(tup) # không thuộc lớp Tuple

<**class** '**int**'>

>>> **str\_tup** = ('howkteam') # **th**ử **m**ộ**t** **tr**ườ**ng** **h**ợ**p** **kh**á**c**

>>> **str\_tup**

'**howkteam**'

>>> **type**(tup)

<**class** '**str**'>

Vì sao khi khởi tạo một Tuple với một phần tử thì kiểu kiểu dữ liệu của Tuple đó lại là kiểu dữ liệu của phần tử duy nhất đó?

* Đó là do khi bạn viết một giá trị nào đó đặt trong cặp dấu ngoặc đơn thì nó được xem là một giá trị.

Vì sao lại phải xem là một giá trị?

* Vì khi ta tính toán, hay sử dụng cặp ngoặc () để được ưu tiên.

>>> 1 + 3 \* 2 # 3 \* 2 sau đó + 1 vì nhân trước cộng sau theo như toán học

7

>>> (1 + 3) \* 3 # giờ thi ta sẽ làm phép tính trong ngoặc trước

12

Thế nên, trường hợp đó không thể tính là một Tuple. Do đó, khi muốn khởi tạo một Tuple chỉ duy nhất một phần tử, ta phải thêm dấu ,, vào sau giá trị đó, để báo cho Python biết, đây là Tuple.

>>> tup = (9,)

>>> tup

(9,)

>>> type(tup) # kết quả đã như mong đợi

<**class** '**tuple**'>

**Sử dụng Tuple Comprehension**

Với Tuple thì khái niệm Comprehension này không được áp dụng

>>> tup = (value **for** value **in** range(3))

>>> tup

<generator object <genexpr> at 0x039F5D20>

Mà đó được coi là **Generator Expression** (Kteam sẽ giới thiệu trong tương lai).

Đối tượng được tạo từ Generator Expression cũng là một dạng iterable.

**Sử dụng constructor Tuple**

**Cú pháp:**

**tuple(iterable)**

|  |
| --- |
|  |

**Công dụng:** Giống hoàn toàn với việc bạn sử dụng constructor List. Khác biệt duy nhất là constructor Tuple sẽ tạo ra một Tuple.

>>> tup = tuple([1, 2, 3])

>>> tup

(1, 2, 3)

>>> str\_tup = tuple('KTEAM')

>>> str\_tup

('K', 'T', 'E', 'A', 'M')

>>> generator = (value **for** value **in** range(10) **if** value % 2 == 0)

>>> generator # bạn không cần phải cố gắng hiểu khi chưa rõ comprehension

<generator object <genexpr> at 0x039F5D20>

>>> tuple(generator)

(0, 2, 4, 6, 8)

>>> tuple(123)

Traceback (most recent call last):

File "<stdin>", line 1, **in** <module>

TypeError: 'int' object **is** **not** iterable

**Một số toán tử với Tuple trong Python**

Các toán tử của Tuple giống với toán tử của chuỗi. Nếu bạn đọc kĩ phần này ở bài List thì bạn sẽ thấy Kteam đề cập là toán tử của List chỉ là gần giống với toán tử của chuỗi. Lí do vì sao sẽ được giải thích trong bài sự khác biệt các toán tử của **hash object** (immutable như chuỗi, Tuple) và **unhash object** (mutable như List)

**Toán tử +**

>>> tup = [1, 2]

>>> tup += ('how', 'kteam')

>>> tup

[1, 2, 'how', 'kteam']

**Toán tử \***

>>> tup = tuple('kter') \* 3

>>> tup

('k', 't', 'e', 'r', 'k', 't', 'e', 'r', 'k', 't', 'e', 'r')

>>> (1,) \* 0

()

>>> (1,) \* 3

(1, 1, 1)

**Toán tử in**

>>> 1 **in** (1, 2, 3)

**True**

>>> 4 **in** ('k', 'kteam', 9)

**False**

**Indexing và cắt Tuple trong Python**

Indexing và cắt Tuple hoàn toàn tương tự như với kiểu dữ liệu List. (Nếu chưa biết về List bạn có thể tham khảo qua các bài về [KIỂU DỮ LIỆU LIST TRONG PYTHON](http://www.howkteam.vn/course/kieu-du-lieu-list-trong-python--phan-2/kieu-du-lieu-list-trong-python--phan-1-1548))

>>> tup = (1, 2, 'a', 'b', [3, 4])

>>> len(tup) # lấy số phần tử có trong tuple

5

>>> tup[0]

1

>>> tup[-1]

[3, 4]

>>> tup[3]

'b'

>>> tup[1:3]

(2, 'a')

>>> tup[:2]

(1, 2)

>>> tup[2:]

('a', 'b', [3, 4])

>>> tup[::-1]

([3, 4], 'b', 'a', 2, 1)

**Thay đổi nội dung Tuple trong Python**

Tuple và chuỗi đều là một dạng **hash object**(immutable). Do đó việc bạn muốn thay đổi nội dung của nó trên lí thuyết là không.

>>> tup = ('kter', 'howkteam', 69)

>>> tup[1]

'howkteam'

>>> tup[1] = 'changed'

Traceback (most recent call last):

File "<stdin>", line 1, **in** <module>

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

Vì sao lại nói là trên lí thuyết? Bạn sẽ biết được ngay ở phần sau.

**Ma trận**

Nếu bạn nắm vững khái niệm này ở List. Thì xin chúc mừng bạn vì không phải đau đầu. Nó hoàn toàn tương tự.

>>> tup = ((1, 2, 3), [4, 5])

>>> tup[0]

(1, 2, 3)

>>> tup[0][2]

3

>>> tup[1][-2]

4

**Tuple có phải luôn luôn là một hash object?**

Như đã định nghĩa ở bài chuỗi, một **hash object** là một đối tượng bạn không thể thay đổi nội dung của nó. Và trong phần thay đổi nội dung Tuple, bạn cũng thấy ta không thể thay đổi giá trị ở bên trong Tuple. Tuy nhiên, không phải lúc nào cũng vậy.

>>> tup = ([1, 2],)

>>> tup[0]

[1, 2]

>>> tup[0][0]

1

>>> tup[0][1]

2

Giá trị bên trong tuple đó là một List. Và, List là một **unhash object**. Suy ra, ta có thể thay đổi nội dung của nó.

>>> tup[0][0]

1

>>> tup[0][0] = 'howkteam'

>>> tup

(['howkteam', 2],)

Ta đã thay đổi nội dung của Tuple bằng một cách đó là thay đổi nội dung của một phần tử trong Tuple.

Vì thế, một Tuple sẽ được coi là một hash object khi nó chứa các phần tử đều là hash object.

**Các phương thức của Tuple**

**Phương thức count**

**Cú pháp:**

<Tuple>**.count(value)**

**Công dụng:** Trả về một số nguyên, chính là số lần xuất hiện của value trong Tuple.

>>> tup = (1, 5, 3, 5, 6, 1, 1)

>>> tup.count(1)

3

>>> tup.count(4)

0

**Phương thức index**

**Cú pháp:**

<Tuple>**.index(sub[, start[, end]])**

**Công dụng:**Tương tự phương thức index của kiểu dữ liệu chuỗi.

>>> tup = (1, 2, 3)

>>> tup.index(2)

1

>>> tup.index(4)

Traceback (most recent call last):

File "<stdin>", line 1, **in** <module>

ValueError: 4 **is** **not** **in** list

**Khi nào thì chọn Tuple thay cho List?**

Tuple khác List ở chỗ Tuple không cho phép bạn sửa chữa nội dung, còn List thì có. Vì đặc điểm đó, Tuple mạnh hơn List ở những điểm sau:

* **Tốc độ truy xuất**của Tuple **nhanh hơn** so với List
* **Dung lượng** chiếm trong bộ nhớ của Tuple**nhỏ hơn** so với List
* Bảo vệ dữ liệu của bạn sẽ không bị thay đổi
* Có thể dùng làm key của Dictonary (một kiểu dữ liệu sẽ được giới thiệu). Điều mà List không thể vì List là unhash object.

Những điểm trên là những điều giúp bạn có thể cân nhắc việc chọn Tuple hay List để lưu dữ dữ liệu dưới một mảng.