

w06-Lec1

One-Dimensional Lists and Tuples - Part II

for 204111

by Kittipitch Kuptavanich

Tuples

- Tuple เป็นรายการข้อมูลที่มีลำดับ (Sequence Data Type) เช่นเดียวกันกับ List, String, และ Range* ที่มีลักษณะ

Immutable

- เราใช้เครื่องหมาย Comma , คั่นระหว่างแต่ละ Element และ ใช้เครื่องหมายวงเล็บ () ล้อมรอบ เช่น (1, 2, 3)
- Tuple ต้องมี Comma เสมอ – แต่ไม่จำเป็นต้องมีเครื่องหมายวงเล็บ

```
>>> t = 12345, 54321, 'hello!'
>>> t[0]
12345
>>> t
(12345, 54321, 'hello!')
```

*<https://docs.python.org/3/library/stdtypes.html#typeseq>

Tuples

- กรณีนี้น่าจะเป็น Tuple ว่างเราจำเป็นต้องใส่เครื่องหมายวงเล็บ
- หากไม่ใช่ Tuple ว่าง ต้องใส่ Comma ทุกกรณี

```
>>> () # Empty tuple, needs parentheses
>()
>>> 1, # Singleton (One Element)
(1,)

>>> t = 12345, 54321, 'python!' # Tuples may be nested:
>>> u = t, (1, 2, 3, 4, 5)
>>> u ((12345, 54321, 'python!'), (1, 2, 3, 4, 5))
```

Tuples Assignment

```
>>> tuple('hello') # Creating from iterables using tuple()  
('h', 'e', 'l', 'l', 'o')
```

```
>>> a = tuple([2, 3, 5, 7]) # tuple from list  
(2, 3, 5, 7)
```

```
>>> a, b, c = 'cat' # multiple assignment
```

```
>>> a  
'c'
```

```
>>> b  
'a'
```

```
>>> c  
't'
```

```
>>> a, b = [2, 8]
```

```
>>> a  
2
```

```
>>> b  
8
```

Immutability

เปลี่ยนไม่ได้

```
>>> t[0] = 88888                                # Tuples are immutable:
TypeError: 'tuple' object does not support item assignment
>>> # but they can contain mutable objects:
>>> v = ([1, 2, 3], [3, 2, 1])
>>> v
([1, 2, 3], [3, 2, 1])
>>> t = ('a', 'b', 'c', 'd', 'e')
>>> # Cannot modify but can replace one tuple with another:
>>> t = ('A',) + t[1:]
>>> print(t)
('A', 'b', 'c', 'd', 'e')
```

- เราไม่สามารถเปลี่ยน Element ใน Tuple ได้ แต่เราสามารถ Assign ค่าใหม่ได้ (ย้าย Reference ไปชี้ที่ Tuple ใหม่)

Tuple Swap

```
>>> a, b, c = [2, 8, 5]
```

```
>>> a
```

```
2
```

```
>>> b
```

```
8
```

```
>>> c
```

```
5
```

```
>>> # Swapping
```

```
>>> b, c, a = a, b, c
```

```
>>> a
```

```
5
```

```
>>> b
```

```
2
```

```
>>> c
```

```
8
```

Tuples as Return Values



- ฟังก์ชันใด ๆ สามารถคืนค่าได้เพียงค่าเดียว หากต้องการคืนค่ามากกว่าหนึ่งค่า เราสามารถคืนค่าเป็น **Sequence Type** เช่น **Tuple** (หรือ **List**) ได้
 - เช่น ในการหารจำนวนเต็มการคำนวณหาผลหาร (**Quotient**) และเศษ (**Remainder**) ทั้งสองค่าในคราวเดียวกัน เป็นวิธีที่มีประสิทธิภาพมากกว่าโดยการใช้ฟังก์ชัน built-in **divmod()**

```
>>> t = divmod(7, 3)
>>> t
(2, 1)
>>> quot, rem = divmod(7, 3)
>>> quot
2
>>> rem
1
```

Strings, Lists and Tuples

`seq[i]` returns the i^{th} element in the sequence.
`len(seq)` returns the length of the sequence.
`seq1 + seq2` returns the concatenation of the two sequences.
`n * seq` returns a sequence that repeats `seq` n times.
`seq[start:end]` returns a slice of the sequence.
`e in seq` is True if `e` is contained in the sequence and False otherwise.
`e not in seq` is True if `e` is not in the sequence and False otherwise.
`for e in seq` iterates over the elements of the sequence.

Figure 5.6 Common operations on sequence types

| Type | Type of elements | Examples of literals | Mutable |
|-------|------------------|---------------------------------------------------------------|---------|
| str | characters | <code>'', 'a', 'abc'</code> | No |
| tuple | any type | <code>()</code> , <code>(3,)</code> , <code>('abc', 4)</code> | No |
| list | any type | <code>[]</code> , <code>[3]</code> , <code>['abc', 4]</code> | Yes |

Figure 5.7 Comparison of sequence types

STRING METHODS INVOLVING LISTS

Basic String Methods w/ Lists

- `str.split([sep[, maxsplit]])`
 - สร้าง List ของ String ย่อยที่เกิดจากการตัด `str` ด้วย String `sep` (Separator) โดยสามารถใช้ Optional Argument `maxsplit` เพื่อจำกัดจำนวนครั้งที่ทำการตัด

```
>>> '1,2,3'.split(',')
['1', '2', '3']
>>> '1,2,3'.split(',', maxsplit=1)
['1', '2 3']
>>> '1,2,,3,.'.split(',')
['1', '2', '', '3', '']
```

Note: ถ้าไม่ระบุ `sep` หรือ `sep` มีค่า `None` การตัดจะถือว่าอักขระ `whitespace` ที่ติดกันทั้งหมด เป็น `Separator` ตัวเดียว

```
>>> '1 2 3'.split()
['1', '2', '3']
>>> ' 1 2 3 '.split()
['1', '2', '3']
```

Basic String Methods w/ Lists [2]

- `str.splitlines(s(keepend=False))`
 - สร้าง List ของ String ย่อยที่เกิดจากแยกบรรทัด `str`

```

01 s = """
02 This is a sample
03 multi-line
04 string
05 """
06
07 print("Lines with splitlines():")
08 for line in s.splitlines():
09     print(" line:", line)
10
11 print("=====")
12
13 print("Lines with splitLines(True):")
14
15 for line in s.splitlines(True):
16     print(" line:", line)
17

```

```

>>>
Lines with splitlines():
line:
line: This is a sample
line: multi-line
line: string
=====
Lines with splitLines(True):
line:

line: This is a sample

line: multi-line

line: string

```

Reference

- <http://www.kosbie.net/cmu/spring-13/15-112/handouts/notes-1d-lists.html>
- <https://docs.python.org/3/tutorial/introduction.html#lists>
- <https://docs.python.org/3/tutorial/datastructures.html#more-on-lists>
- <https://docs.python.org/3.3/tutorial/datastructures.html#tuples-and-sequences>
- <https://docs.python.org/3/library/stdtypes.html#typeseq-mutable>
- <https://docs.python.org/3/library/stdtypes.html#tuple>
- Gutttag, John V *Introduction to Computation and Programming Using Python, Revised*