

w06-Lec

One-Dimensional Lists and Tuples - Part I

for 204111

by Kittipitch Kuptavanich

Lists

- พิจารณาการคำนวณค่าเบี่ยงเบนมาตรฐานของจำนวนจริง N จำนวน

$$s_N = \sqrt{\frac{1}{N} \sum_{i=1}^N (x_i - \bar{x})^2}$$

- เริ่มจากการหาค่าเฉลี่ย \bar{x}
 - และหาผลรวม (Σ) ของกำลังสอง ของผลต่างจากค่าเฉลี่ย $(x_i - \bar{x})^2$ ของแต่ละจำนวน
- จำเป็นต้องเก็บข้อมูล N จำนวน (อ่านข้อมูล 2 ครั้งต่อจำนวน) - ตัวแปร N ตัว? \rightarrow ไม่สะดวกในการเรียกใช้
- List เป็นหนึ่งในชนิดข้อมูลที่สามารถใช้เก็บข้อมูลหลาย ๆ ค่าในตัวแปรหนึ่งตัว

Lists [2]

- List เป็นชนิดข้อมูลแบบประกอบ (Compound Data Type) ที่มีลักษณะเป็นรายการข้อมูลที่มีลำดับ (Sequence Type, e.g. **list**, **tuple**, and **range**) เช่นเดียวกับกับ String (Text Sequence Type)
 - String เป็นรายการอักขระ
 - List เป็นรายการข้อมูลประเภทใดก็ได้
- เราเรียกข้อมูลแต่ละตัวที่อยู่ใน List ว่า Element หรือ Item
- เราใช้เครื่องหมาย Bracket **[]** เพื่อแสดง List และคั่นระหว่างแต่ละ Element ด้วยเครื่องหมาย Comma **,** เช่น

```
>>> squares = [1, 4, 9, 16, 25]
>>> squares
[1, 4, 9, 16, 25]
```

Lists [3]

- โดยมากแต่ละ **Element** ใน **List** จะมีชนิดข้อมูล (Data Type) เป็นชนิดเดียวกันทั้งหมด แต่ **List** สามารถประกอบด้วย **Element** ที่มีชนิดข้อมูล ต่างกันก็ได้

```
>>> mixed_list = ['spam', 2.0, 5, [10, 20]]  
>>> empty = []
```

- **List** ด้านบนประกอบด้วย **Element** ชนิด **String**, **Float**, **Integer** และ อีก **List** ซ่อนอยู่ข้างใน (**Nested List**)
- เราเรียก **List** ที่ไม่มี **Element** เรียกว่า **Empty List** (ลิสต์ว่าง) **[]**

Lists and Strings

- ฟังก์ชัน `list()` ใช้สร้าง List จาก Iterables อื่น ๆ เช่น String

```
>>> a = list("wahoo!")           # from a string
>>> a
['w', 'a', 'h', 'o', 'o', '!']

>>> b = list(range(5))           # or from a range
>>> b
[0, 1, 2, 3, 4]

>>> e = list()
>>> e
[]                               # same as list("")

>>> s = "".join(a)               # use "".join(a) to convert a list
>>> s                             # of character to a string
'wahoo!'

                                   # also works with a list of string
>>> "--".join(['parsley', '', 'is', '', 'gharsley'])
'parsley----is----gharsley'
```

Lists and Ranges

- ตัวอย่างการสร้าง List จาก `range()`

```
>>> list(range(1, 11))           # start:stop  
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
>>> list(range(0, 30, 5))        # start:stop:step  
[0, 5, 10, 15, 20, 25]
```



```
>>> list(range(0, 10, 3))  
[0, 3, 6, 9]
```

```
>>> list(range(0, -10, -1))  
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
```

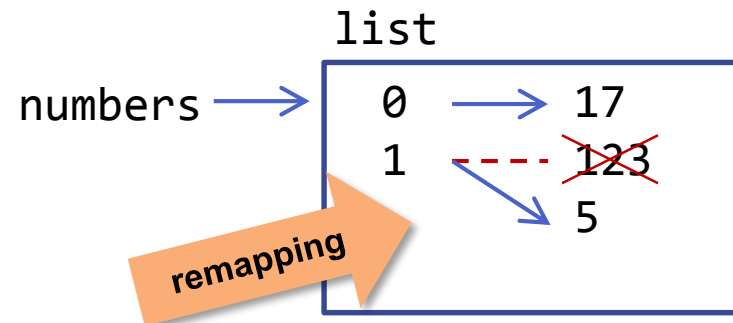
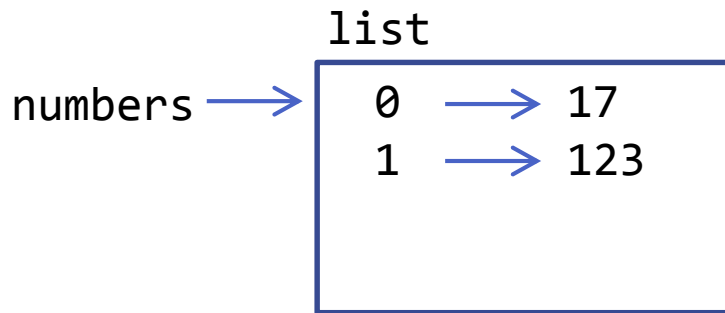
```
>>> list(range(0))  
[]
```

```
>>> list(range(1, 0))  
[]
```

Lists are Mutable

- List มีคุณสมบัติ **Mutable** กล่าวคือสามารถเปลี่ยนแปลงข้อมูลที่เกิดขึ้นใน List ได้ (ต่างจาก String ซึ่งมีคุณสมบัติ **Immutable**)

```
>>> numbers = [17, 123]
>>> numbers[1] = 5
>>> print(numbers)
[17, 5]
```



- เราสามารถพิจารณา List ในลักษณะความสัมพันธ์ระหว่าง Index และ Element เช่น Index 0 สัมพันธ์กับ (Maps to) ค่า 17

Indexing and Slicing

- Indexing และ Slicing ใน List มีลักษณะเดียวกันกับใน String

```
>>> squares = [1, 4, 9, 16, 25, 36, 49]
>>> squares[0]           # indexing returns the item
1

>>> squares[-1]
49

>>> squares[-3:]         # slicing returns a new list
[16, 25, 36]

>>> squares[1:5:2]       # slicing with [start:stop:step]
[4, 16]

>>> squares[:]           # create a (shallow) copy of a list
[1, 4, 9, 16, 25, 36, 49]
```

Note: Indexing และ Slicing
ใช้ได้กับทุก Sequence Type
e.g. List, Tuple, Range

Indexing and Slicing [2]

- เราสามารถ **Assign** ค่าให้กับ **Slice** ของ **List** ได้

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
```

```
>>> # replace some values
```

```
>>> letters[2:5] = ['C', 'D', 'E']
```

```
>>> letters
```

```
['a', 'b', 'C', 'D', 'E', 'f', 'g']
```

```
>>> # now remove them
```

```
>>> letters[2:5] = []
```

```
>>> letters
```

```
['a', 'b', 'f', 'g']
```

```
>>> # replacing all the elements with an empty list
```

```
>>> letters[:] = []
```

```
>>> letters
```

```
[]
```

Indexing and Slicing [3]

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> letters[2:6:3] = [1, 4]      # assigning w/ [start:stop:step]
>>> letters
['a', 'b', 1, 'd', 'e', 4, 'g']

>>> letters[2:6:3] = [1, 4, 5]   # same length only
...
ValueError: attempt to assign sequence of size 3 to extended slice
of size 2

>>> # assigning with values from other iterables
>>> nums = [0, 1, 2, 3, 4, 5, 6, 7]
>>> nums[2:5] = 'hello'
>>> nums
[0, 1, 'h', 'e', 'l', 'l', 'o', 5, 6, 7]

>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> letters[2:5] = range(3)
>>> letters
['a', 'b', 0, 1, 2, 'f', 'g']
```

Displaying Lists with `print()`

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']

>>> print(' '.join(letters)) # using str.join()
a b c d e f g
>>> print(', '.join(letters))
a, b, c, d, e, f, g

>>> print(*letters) # using *
a b c d e f g

>>> nums = [1, 2, 3, 4]
>>> print(', '.join(nums)) # each elem must be a string
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: sequence item 0: expected str instance, int found

>>> print(*nums) # using *
1 2 3 4
```

List Properties (len, min, max, sum)

```
# List Property Built-in Functions
```

```
>>> a = [2, 3, 5, 2]
```

```
>>> a  
[2, 3, 5, 2]
```

```
>>> len(a)  
4
```

```
>>> min(a)  
2
```

```
>>> max(a)  
5
```

```
>>> sum(a)  
12
```

Int

List Operations

- The **+** operator concatenates lists:

```
>>> a = [5, 3]
>>> b = [2, 1] + a
>>> print(b)
[2, 1, 5, 3]
```

- Similarly, the ***** operator repeats a list:

```
>>> [0] * 4
[0, 0, 0, 0]
>>> [1, 2, 3] * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

The `del` Statement

- เราใช้คำสั่ง `del` ประกอบกับ Slicing เพื่อลบสมาชิกบางตัว หรือทุกตัวได้

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> del letters[2:5] # remove some elements
>>> letters
['a', 'b', 'f', 'g']

>>> del letters[:] # remove all elements
>>> letters
[] # empty list

>>> del letters
>>> letters # no reference to the list
...
NameError: name 'letters' is not defined
```

Adding Elements

- List เป็น Data Type ประเภท **Mutable Sequence Type**
 - สามารถใช้ **Method** ของ **Mutable Sequence Type** ได้

เพิ่ม Elements – List เดิม (Destructively)

```
>>> a = [2, 3] # Create a list
```

```
>>> a.append(7) # Add an item with method list.append(item)
```

```
>>> a
```

```
[2, 3, 7]
```

```
>>> a.append([4, 5])
```

```
>>> a
```

```
[2, 3, 7, [4, 5]]
```

list.append() → เพิ่มลงไปในลิสต์รวมของลิสต์ด้วย []

list + list

→ เพิ่มแต่ละองค์ไม่ใส่ในลิสต์

```
>>> a += [11, 13] # Add a list of items with list += list2
```

```
>>> a
```

```
[2, 3, 7, [4, 5], 11, 13]
```

Adding Elements [2]

เพิ่ม Elements – List เดิม (Destructively) [2]

```

>>> a
[2, 3, 7, 11, 13]
# Add a list of items with method list.extend(list2)
>>> a.extend([17, 19])
>>> a
[2, 3, 7, 11, 13, 17, 19]

# Insert an item with method list.insert(index, element)
>>> a.insert(2, 5)
>>> a
[2, 3, 5, 7, 11, 13, 17, 19]

```

1 แล้วย + 1 เดิม ()

แทรกที่ช่อง
ถ้าช่องอยู่แล้วจะทับลงไป

Adding Elements [3]

เพิ่ม Elements – สร้าง List ใหม่ (Non-destructively)

```
>>> a
[2, 3, 7, 11]

# Add an item with list1 + list2
b = a + [13, 17]
>>> a
[2, 3, 7, 11]
>>> b
[2, 3, 7, 11, 13, 17]

# Insert an item at a given index (with list slices)
>>> b = a[:2] + [5] + a[2:]
>>> a
[2, 3, 7, 11]
>>> b
[2, 3, 5, 7, 11]
```

Finding Elements

ค้นหา Elements

```
>>> a = [2, 3, 5, 2, 6, 2, 2, 7]

>>> 2 in a          # Check for list membership: in
True
>>> 4 in a
False

>>> 4 not in a      # Check for list non-membership: not in
True

# Count occurrences with method list.count(item)
>>> a.count(1)
0
>>> a.count(2)
4
```

Finding Elements [2]

ค้นหา Elements [2]

```
>>> a = [2, 3, 5, 2, 6, 2, 2, 7]

# Find index of item with method list.index(item)
>>> a.index(6)
4
>>> a.index(2)
0                                     # Index of the first item found

# list.index(item, start)
>>> a.index(2, 1)
3                                     # Start looking at index 1
>>> a.index(2, 4)
5
>>> a.index(8)
...
ValueError: 8 is not in list
```

Removing Elements

ลบ Elements – List เเดิม (Destructively)

```
>>> a = [2, 3, 5, 3, 7, 5, 11, 13]

# Remove an item with method List.remove(item)
>>> a.remove(5)
>>> a
[2, 3, 3, 7, 5, 11, 13] # Remove only the first occurrence
```

1 อย่างเดียวที่เจอ

```
>>> a.remove(5)
>>> a
[2, 3, 3, 7, 11, 13]

>>> a.remove(5)
...
ValueError: list.remove(x): x not in list
```

Removing Elements [2]

ลบ Elements – List เดิม (Destructively)

```
>>> a = [2, 3, 5, 8, 7, 5, 11, 13]
```

```
# Remove an item at a given index with method List.pop(index)
```

```
>>> item = a.pop(3)
```

```
>>> item
```

```
8
```

```
>>> a
```

```
[2, 3, 5, 7, 5, 11, 13]
```

ลบจากลำดับ ลบตัวที่ 4 ของ

```
# Remove Last item with List.pop()
```

```
>>> item = a.pop()
```

```
>>> item
```

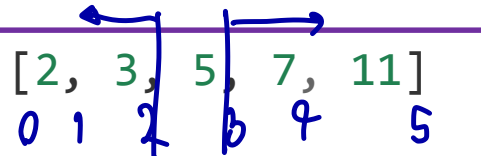
```
13
```

```
>>> a
```

```
[2, 3, 5, 7, 5, 11]
```

Removing Elements [3]

ลบ Elements – สร้าง List ใหม่ (Non-destructively)



```

>>> a = [2, 3, 5, 7, 11]
          0 1 2 3 4
# Remove an item at a given index (with list slices)
>>> b = a[:2] + a[3:]
>>> a
[2, 3, 5, 7, 11]
>>> b
[2, 3, 7, 11]

```

List Alias

```
# Create a list
```

```
>>> a = [2, 3, 5, 7]
```

```
# Create an alias to the list
```

```
>>> b = a
```

```
# We now have two references (aliases) to the SAME list
```

```
>>> a[0] = 42
```

```
>>> b[1] = 99
```

```
>>> print(a)
```

```
[42, 99, 5, 7]
```

```
>>> print(b)
```

```
[42, 99, 5, 7]
```

```
>>> a is b
```

```
True
```

List Alias [2]

```
>>> a = [2, 3, 5, 7]
>>> b = a                # Create an alias to the list

# Create a different list with the same elements
>>> c = [2, 3, 5, 7]

# a and b are references (aliases) to the SAME list
# c is a reference to a different but EQUAL list

>>> a == b
True
>>> a is b
True
>>> a == c
True
>>> a is c
False
```


List Alias [3]

```
# Function parameters are aliases, too!
>>> def f(a):
...     a[0] = 42

>>> a = [2, 3, 5, 7]

>>> f(a)
>>> print(a)
[42, 3, 5, 7]
```

Using Lists with Functions

- List Parameters

- Example: `repeat_last(list)`

```
05 def repeat_last(list_a):  
06     list_a.append(list_a[-1])  
07  
08  
09 print(repeat_last([1, 2, 3]))  
10 # [1, 2, 3, 3]
```

- Modifying list elements is visible to caller
 - If the operation is destructive

Using Lists with Functions [3]

```
>>> def repeat_last_dest(x):  
...     x.append(x[-1])  
...  
>>> a = [1, 2, 3]  
>>> b = repeat_last_dest(a)  
  
>>> print(b)  
  
_____  
  
>>> print(a)  
  
_____
```

```
>>> def repeat_last_n_dest(x):  
...     return x + [x[-1]]  
...  
>>> a = [1, 2, 3]  
>>> b = repeat_last_n_dest(a)  
  
>>> print(b)  
  
_____  
  
>>> print(a)  
  
_____
```

Mutable Object as an Argument

- พิจารณาฟังก์ชันในการหาค่าเฉลี่ยที่รับ Input เป็น List แล้วคืนค่าเฉลี่ยของทุก Element

```
05 def find_average(list_a):
```

- หลังจากการเรียกใช้ฟังก์ชัน `find_average()`
 - `list_a` ควรเปลี่ยนแปลงหรือไม่?
- แนวคิดพื้นฐานเวลาเมื่อทำ Operation ใด ๆ กับ Data คือไม่ควรเปลี่ยนแปลงข้อมูลที่ได้มาแล้วเขียนทับที่เดิม
- ยกเว้นมีความจำเป็นจะต้องแก้ไขข้อมูลเหล่านั้นจริง ๆ

Mutable Objects as Default Arguments

- การใช้ Mutable Object (**list**, **set**, **dict**, etc.) เป็น Default Argument ใน Python ต้องทำความระมัดระวัง

```

05 def make_singleton(x, target=[]):
06     target.append(x)
07     return target
08
09 print(make_singleton('apple'))
10 print(make_singleton('orange'))

```

make_singleton.py

- สิ่งที่คิด

```

$ ./make_singleton.py
['apple']
['orange']

```

- สิ่งที่ได้

```

$ ./make_singleton.py
['apple']
['apple', 'orange']

```

Mutables as Default Arguments [2]

- **Default Argument** ใน Python ถูก **evaluate** เพียงหนึ่งครั้งผ่าน **Function Definition** ไม่ใช่ทุกครั้งที่ถูกเรียกใช้
- กรณีนี้คือได้ List เดิมทุกครั้งที่ถูกใช้ฟังก์ชัน
- หากใช้ **Mutable Object** อื่น ๆ เป็น **Default Argument** ก็เช่นกัน
- **string, int, float, tuple** ไม่เป็น **Mutables**

```
05 def make_singleton(x, target=None):  
06     if target is None:  
07         target = []  
08  
09     target.append(x)  
10     return target
```

make_singleton.py

Sorting Elements

เรียง Elements – List เดิม (Destructively)

```
>>> a = [7, 2, 5, 3, 5, 11, 7]

# Sort item destructively with method list.sort()
>>> a.sort()
>>> a
[2, 3, 5, 5, 7, 7, 11]
```

เรียง Elements – สร้าง List ใหม่ (Non-destructively)

```
>>> a = [7, 2, 5, 3, 5, 11, 7]
>>> b = sorted(a) # non-destructively with sorted(list)
>>> a
[7, 2, 5, 3, 5, 11, 7]
>>> b
[2, 3, 5, 5, 7, 7, 11]
```

Reversing Elements

กลับลำดับ Elements – List เดิม (Destructively)

```
>>> a = [1, 2, 3, 4, 5, 6, 7]

>>> # Sort item destructively with method list.reverse()
>>> a.reverse()
>>> a
[7, 6, 5, 4, 3, 2, 1]
```

กลับลำดับ Elements – สร้าง List ใหม่ (Non-destructively)

```
>>> a = [1, 2, 3, 4, 5, 6, 7]
>>> b = list(reversed(a)) # non-destructively with reversed(list)
>>> a
[1, 2, 3, 4, 5, 6, 7]
>>> b
[7, 6, 5, 4, 3, 2, 1]
```


Swapping Elements

```
>>> a = [2, 3, 5, 7]
```

```
>>> a[0] = a[1] # Failed swap
```

```
>>> a[1] = a[0]
```

```
>>> a
[3, 3, 5, 7]
```

```
>>> a = [2, 3, 5, 7] # Swap with a temp variable
```

```
>>> temp = a[0]
```

```
>>> a[0] = a[1]
```

```
>>> a[1] = temp
```

```
>>> a
[3, 2, 5, 7]
```

Tuple swap

x, y = y, x

```
>>> a = [2, 3, 5, 7]
```

```
>>> a[0], a[1] = a[1], a[0] # Swap with tuple assignment
```

```
>>> a
[3, 2, 5, 7]
```

Comparing Lists

```
>>> a = [2, 3, 5, 3, 7]
>>> b = [2, 3, 5, 3, 7]
>>> c = [2, 3, 5, 3, 8]
>>> d = [2, 3, 5]
```

same as a

differs in last element

prefix of a

```
>>> a == b
```

```
True
```

```
>>> a == c
```

```
False
```

```
>>> a != b
```

```
False
```

```
>>> a != c
```

```
True
```

```
>>> a < c
```

```
True
```

```
>>> a < d
```

```
False
```

เทียบทีละ Element

ในลักษณะเดียวกับการเทียบคำ

ในภาษาอังกฤษ ทีละตัวอักษร

bat < cat #True

cat < caterpillar #True

List Operation Summary

Operation	Result	Notes
<code>s[i] = x</code>	item <i>i</i> of <i>s</i> is replaced by <i>x</i>	
<code>s[i:j] = t</code>	slice of <i>s</i> from <i>i</i> to <i>j</i> is replaced by the contents of the iterable <i>t</i>	
<code>del s[i:j]</code>	same as <code>s[i:j] = []</code>	
<code>s[i:j:k] = t</code>	the elements of <code>s[i:j:k]</code> are replaced by those of <i>t</i>	
<code>del s[i:j:k]</code>	removes the elements of <code>s[i:j:k]</code> from the list	
<code>s.append(x)</code>	appends <i>x</i> to the end of the sequence (same as <code>s[len(s):len(s)] = [x]</code>)	
<code>s.clear()</code>	removes all items from <i>s</i> (same as <code>del s[:]</code>)	
<code>s.copy()</code>	creates a shallow copy of <i>s</i> (same as <code>s[:]</code>)	
<code>s.extend(t)</code>	extends <i>s</i> with the contents of <i>t</i> (same as <code>s[len(s):len(s)] = t</code>)	
<code>s.insert(i, x)</code>	inserts <i>x</i> into <i>s</i> at the index given by <i>i</i> (same as <code>s[i:i] = [x]</code>)	
<code>s.pop([i])</code>	retrieves the item at <i>i</i> and also removes it from <i>s</i>	
<code>s.remove(x)</code>	remove the first item from <i>s</i> where <code>s[i] == x</code>	
<code>s.reverse()</code>	reverses the items of <i>s</i> in place	

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>
- Guttag, John V *Introduction to Computation and Programming Using Python, Revised*