Lecture 13 Collecting Things Together- List and Dictionaries ที่เก็บสิ่งของ ลิสต์และดิกชั่นนารี่

13 ตุลาคม 2557 14:58

เราสามารถเก็บค่าไว้คำนวณในหน่วยความจำคอมพิวเตอร์ (Memory) โดยการตั้งชื่อ ตัวแปร (variable) เพื่อที่จะเก็บค่าไม่ว่าจะเป็นข้อความหรือค่าตัวเลข และสามารถเก็บได้หนึ่งค่าต่อหนึ่งตัวแปรเท่านั้น ทั้งนี้เราสามารถที่จะเก็บค่าเป็นชุดหรือกลุ่มของค่าต่างๆ อยู่ที่เดียวกันเราเรียกว่า คอลเล็คชั่น (collection) เราสามารถจะอ่านค่าและเก็บค่าเป็นกลุ่มอย่างง่าย ชนิดของข้อมูลที่เป็นคอลเล็คชั่นที่เราจะกำลังเรียนรู้คือ list (ลิสต์)และ dictionary (ดิกชั่นนารี่) เกี่ยวกับการสร้าง การแก้ไขและการใช้ต่างๆ
List มีประโยชน์มากและเราใช้เป็นจำนวนมาก ตัวอย่างเช่นเกมส์สกีที่เราเขียน เราได้ใช้ list ในการเก็บข้อมูลชื่อของรูปภาพในเกม และใช้ในบทต่อไปทั้งการเขียนเกมและกราฟฟิค ซึ่งทุกอย่างจะถูกเก็บไว้ในลิสต์

ลิสต์คืออะไร
ตัวอย่างเช่น เราลองเขียนชื่อคนในครอบครัวว่าประกอบด้วยใคร ซึ่งนี้เป็นลักษณะคล้ายกับลิสต์มาก

```
family = ['Mom', 'Dad', 'Junior', 'Baby']
```

หรือตัวเลขนำโชคต่างๆ

```
luckyNumbers = [2, 7, 14, 26, 30]
```

สิ่งของที่ละอันที่อยู่ในทั้ง family หรือ luckyNumbers เช่น 'Mom' หรือ 2 เราเรียกว่า ไอเทม (Item)
List ในภาษา python จะอยู่ด้านในของ บรักเก็ต (bracket) [] และแต่ข้อมูลหรือไอเทม จะอยู่ด้านในนี้และขั้นกันด้วยคอมม่า

การสร้าง List : Creating a list

family และ luckyNumbers คือชื่อของตัวแปรที่เป็นชนิด list ซึ่งเราได้ลองเก็บค่าต่างๆ ทั้งตัวเลขและข้อความแล้ว แต่บางครั้งเราอย่างต้องการใช้ list โดยที่ไม่รู้ว่าจะเก็บค่าอะไรล่วงหน้า เราสามารถสร้าง list เปล่าๆ ได้ (empty list)

```
newList = []
```

จะเห็นได้ว่าไม่มีสิ่งของอยู่ด้านใน list

การเพิ่มข้อมูล : Adding things to a list

การเพิ่มข้อมูลเราจะใช้คำสั่ง append() ,

```
>>> friends = []
>>> friends.append("Golf")
>>> print(friends)
['Golf']
```

ลองเพิ่มข้อมูล

```
>>> friends.append("Gig")
>>> print(friends)
['Golf', 'Gig']
```

append หมายถึง การนำบางสิ่งต่อท้าย ถ้าเรา append item ลง list สิ่งของนั้นจะมาต่อท้าย

จุดคืออะไร What's the dot?

ทำไมเราใช้จุด (dot) ระหว่าง friends และ append() นี้คือลักษณะการใช้งานเบื้องต้นเชิงวัตถุ (object) ที่เราจะได้เรียนในบทต่อๆ ไป object ประกอบด้วยสองส่วนคือข้อมูล ที่เก็บไว้ในตัวแปรต่างๆ และงานหรือหน้าที่ที่ object สามารถทำได้ เช่น list สามารถที่จะเพิ่มค่าเก็บได้ (append)

```
friends.append(something)
```

List สามารถเก็บสิ่งของได้หลายชนิด

```
myList = [5, 10, 23.76, 'Hello', myTeacher, 7, another_list]
```

เราสามารถเก็บตัวเลขจำนวนเต็ม จำนวนทศนิยม ข้อความ หรือ ตัวแปรไว้ใน list ได้ แม้กระทั่งตัวแปรที่เป็น list ของตัวอื่น

ด้วย ไว้ใน list เดียวกันได้

การอ่านค่า Getting items from a list

เราจะทดสอบจาก list นี้

```
>>> letters = ['a', 'b', 'c', 'd', 'e']
>>> print(letters)
['a', 'b', 'c', 'd', 'e']
```

เราจะอ่านค่าใน list โดยระบุช่องที่เก็บหรืออินเด็กซ์ (index) โดยจะเริ่มจากช่องที่ 0 (ช่องแรก)

```
>>> letters[0]
'a'
```

ค่าที่ได้ คือ 'a' ค่าที่ช่องแรก เมื่อทดสอบ อ่านค่า index = 3 (ค่าที่ช่อง 4)

```
>>> letters[3]
'd'
```

ทำไมอินเดกซ์ (index) เริ่มที่ 0 ไม่ 1?

โปรแกรมเมอร์และนักวิทยาการคอมพิวเตอร์ได้ทำการตกลงกัน เนื่องจาก คอมพิวเตอร์เริ่มโดยการเก็บค่าไบนารี่ binary และการคำนวณแต่ละโปรแกรมจะต้องใช้เวลา เนื่องจากค่าที่เครื่องปกติเริ่มที่ 0 จริงไม่จำเป็นจะต้องเพิ่มค่า 1 เข้าไป ทำให้มีประสิทธิภาพที่ดีกว่า และทุกวันนี้การใช้ค่า 0 ในการเริ่มต้นเป็นสิ่งที่เป็นเรื่องปกติในการเขียนโปรแกรม ซึ่งโปรแกรมเมอร์ส่วนมากรู้และเข้าใจ

การตัด slicing a list

เราสามารถที่จะสามารถอ่านค่ามากกว่าหนึ่งช่องหรือหลายตำแหน่ง โดยระบุช่องที่จะเริ่มและหยุดก่อนช่องที่

```
>>> letters[1:4]
['b', 'c', 'd']
```

จะสังเกตุได้ว่า เราระบุ 1 - 4 ไป แต่ทำไมมีแค่ 3 ตัว ยังจำได้หรือไม่เวลาเราใช้ range(5) ค่าที่ได้คือ 0-4 ไม่รวมค่าสุดท้ายเช่นเดียวกัน ในที่นี้จะไม่รวมตัวในค่าสุดท้าย นิยามที่นี้คือการหยุดก่อน (stop before)
การตัดจะเป็นการสร้าง list ใหม่ขึ้นมา โดยไม่แก้ไข list เก่า ถ้าเราสังเกตุดูค่าที่ออกมาจะมี brackets ครอบไว้แสดงว่า ค่าที่ได้ออกมายังเป็น list อยู่ ซึ่งแตกต่างจากการระบุ index ที่จะได้ค่าที่เก็บในช่องออกมา

```
>>> print(letters[1])
b
>>> print(letters[1:2])
['b']
```

เมื่อเราพิมพ์ชนิดข้อมูลดู จะเห็นผลดังนี้

```
>>> type(letters[1])
<class 'str'>
>>> type(letters[1:2])
<class 'list'>
```

ถ้าเราไม่ระบุตัวเริ่ม จะหมายถึงค่าที่ได้จะเริ่มจากช่องแรก

```
>>> letters[:2]
['a', 'b']
```

และถ้าไม่ระบุช่องหยุดก่อน จะหมายถึงทำไปจนหมดทุกตัว

```
>>> letters[2:]
['c', 'd', 'e']
```

ถ้าไม่ระบุทั้งเริ่มและหยุดจะหมายถึงคัดลอก ทั้งหมดจาก list

```
>>> letters[:]
['a', 'b', 'c', 'd', 'e']
```

การแก้ไขข้อมูลใน list : Modifying items

เราสามารถแก้ไขข้อมูลที่เก็บไว้ได้ เช่นต้องการเปลี่ยนตัว 'c' ให้เป็นตัว 'z' ซึ่งเราจะต้องเข้าไปแก้ที่ช่อง index ที่ 2 ให้มีค่า 'z'

```
>>> print(letters)
['a', 'b', 'c', 'd', 'e']
>>> letters[2] = 'z'
>>> print(letters)
['a', 'b', 'z', 'd', 'e']
```

แต่ถ้าเราแก้ไขช่อง index เท่ากับ 5

```
>>> letters[5] = 'f'
Traceback (most recent call last):
  File "<pyshell#28>", line 1, in <module>
    letters[5] = 'f'
IndexError: list assignment index out of range
```

จะเกิด error เนื่องจากช่องเก็บมีแค่ index เท่ากับ 0-4 เท่านั้น
เราลองแก้กลับให้เป็นค่า 'c' เหมือนเดิม

```
>>> letters[2] = 'c'
>>> print(letters)
```

```
>>> letters[2] = 'c'
>>> print(letters)
['a', 'b', 'c', 'd', 'e']
```

การเพิ่มข้อมูลลง list :append()

เราได้เห็นวิธีการเพิ่มข้อมูลลง list ด้วยคำสั่ง append() ไปแล้ว นอกจาก append เรามีวิธีการเพิมข้อมูลอีก 2 รูปแบบ สามารถสรุปได้ดังนี้

- append() การเพิ่มข้อมูลต่อท้ายตัวสุดท้าย
- extend() การเพิ่มข้อมูลได้ครั้งล่ะหลายค่า หรือการต่อท้ายข้อมูลกับ list ตัวอื่น เพิ่มเติม
- insert() การเพิ่มข้อมูลแทรกก่อนตรงช่องหรือตำแหน่ง index ที่ต้องการ

การเพิ่มด้วย append()

เราได้แสดงตัวอย่างไปแล้ว ลองดูตัวอย่างเพิ่มเติม การเพิ่มตัว 'n'

```
>>> letters.append('n')
>>> print(letters)
['a', 'b', 'c', 'd', 'e', 'n']
```

การเพิ่มตัว 'g'

```
>>> letters.append('g')
>>> print(letters)
['a', 'b', 'c', 'd', 'e', 'n', 'g']
```

การเพิ่มข้อมูลด้วยการต่อ list : extend()

เนื่องจากเป็นการเพิ่มหลายตัว ดังนั้นเราจะเก็บค่าไว้ใน list อื่น จะต้องใช้ bracket เข้าช่วย ดังนั้นเราจะเห็นดังโค้ดนี้

```
>>> letters.extend(['1', '2', '3'])
>>> print(letters)
['a', 'b', 'c', 'd', 'e', 'n', 'g', '1', '2', '3']
```

เราจะเห็น มี brackets หุ้ม ['1' , '2', '3'] ซึ่งนี้แสดงให้เห็นว่าเป็นอีก list เมื่อเรา extend เหมือนกับว่าเรารวม list สองอันด้วยกัน

ข้อแตกต่างระหว่าง append() และ extend()

จากตัวอย่างที่แล้ว เราจะเห็นได้ว่า ข้อมูลรวมกัน แต่เมื่อเราใช้คำสั่ง append() นั้น list = ['1', '2', '3']

```
>>> letters = ['a', 'b', 'c', 'd', 'e']
>>> letters.append(['1', '2', '3'])
>>> print(letters)
['a', 'b', 'c', 'd', 'e', ['1', '2', '3']]
```

เราจะสังเกตุได้ว่า ['1', '2', '3'] จะเป็นก้อนข้อมูลหรือ item หนึ่งใน letters ซึ่งจะเห็นมี brackets อยู่ด้านในซ้อนกันอยู่ ข้อมูลจะไม่แยกกัน

การลบข้อมูลจาก list

list จะมีคำสั่งที่สามารถลบข้อมูลได้ดังนี้ remove() , del() , pop()

การลบข้อมูลด้วย remove()

เราจะระบุข้อมูลที่ต้องการลบไปตรงๆ เช่น ต้องการลบ ตัว 'b'

```
>>> letters = ['a', 'b', 'c', 'd', 'e']
>>> letters.remove('b')
>>> print(letters)
['a', 'c', 'd', 'e']
```

แต่ถ้าลบข้อมูลที่ไม่มีอยู่ เช่น 'f' จะเกิด error

```
>>> letters.remove('f')
Traceback (most recent call last):
  File "<pyshell#45>", line 1, in <module>
    letters.remove('f')
ValueError: list.remove(x): x not in list
```

การลบข้อมูลด้วย del()

Del ย่อมาจาก delete หมายถึงลบ เราจะส่งข้อมูลที่ต้องการลบเข้าไป เช่นลบที่ตำแหน่ง index 3

```
>>> letters = ['a', 'b', 'c', 'd', 'e']
>>> del(letters[3])
>>> print(letters)
['a', 'b', 'c', 'e']
```

การลบข้อมูลด้วย pop()

การลบข้อมูลช่องสุดท้าย pop() เป็นคำสั่งถอนข้อมูลล่าสุด หรือ ข้อมูลที่ตำแหน่งสุดท้ายคืนให้ ผู้ใช้

```
>>> letters = ['a', 'b', 'c', 'd', 'e']
>>> lastLetter = letters.pop()
>>> print(letters)
['a', 'b', 'c', 'd']
>>> print(lastLetter)
e
```

คำสั่ง pop() จะถอนข้อมูลล่าสุดคืนให้ผู้ใช้ เมื่อเราสร้างตัวแปร lastLetter มารับ ข้อมูลใน list ตัวสุดท้ายจะโดนลบไป และ lastLetter จะเท่ากับ 'e' เนื่องจาก เป็นข้อมูลสุดท้ายที่ถอนคืน
เราสามารถระบุตำแหน่ง index ที่ต้องการลบด้วยคำสั่ง pop() ได้ดังนี้ เราต้องการลบค่าที่ช่องที่ 1

```
>>> letters = ['a', 'b', 'c', 'd', 'e']
>>> second = letters.pop(1)
>>> print(second)
b
>>> print(letters)
['a', 'c', 'd', 'e']
```

การค้นหาข้อมูลใน list searching a list

การค้นหามีสองรูปแบบคือ

1. หา item ใน ว่าอยู่มีหรือไม่
2. หาตำแหน่งช่องที่ item นี้มีอยู่

ถ้า 'a' มีอยู่ให้แสดงข้อความบนหน้าจอว่ามี ถ้าไม่มีให้แสดงว่าหาไม่เจอ

การใช้คำว่า in

```
letters = ['a', 'b', 'c', 'd', 'e']
if 'a' in letters:
    print("found 'a' in letters")
else:
    print("didn't find 'a' in letters")
```

ได้ผลดังนี้

```
>>>
found 'a' in letters
```

เราจะตรวจสอบว่าตัว 'a' อยู่ใน list นี้ หรือไม่

```
>>> 'a' in letters
True
>>> 's' in letters
False
```

ถ้าผลลัพท์เป็นจริงTrue แสดงว่า มี 'a' เป็น item ใน letters
ถ้าผลลัพท์ไม่เป็นจริง False เช่น ตัว 's' ไม่ได้เป็น item ใน letters

ประโยชน์ของ in เราอาจใช้ตรวจสอบข้อมูลก่อนลบ ถ้ายังจำได้ถ้าเราสั่งลบข้อมูลที่ไม่อยู่จะเกิน error เพราะฉะนั้นเราควรพิจารณาและตรวจสอบก่อนที่จะลบข้อมูล ตัวอย่างการตรวจสอบเราอาจใช้คำสั่ง if ก่อน เช่น

```
>>> if 'a' in letters:
        letters.remove('a')
```

หาตำแหน่ง index ใน list

เราจะหาตำแหน่งของ item ที่เราสนใจเช่น หาว่าตัวอ 'd' อยู่ตำแหน่งไหน

```
>>> letters = ['a', 'b', 'c', 'd', 'e']
>>> print(letters.index('d'))
3
```

เช่นเดียวกันกับ remove ถ้าหาตำแหน่งไม่เจอ อาจจะเกิด error ได้

```
>>> letters.index('z')
Traceback (most recent call last):
  File "<pyshell#69>", line 1, in <module>
    letters.index('z')
ValueError: 'z' is not in list
```

ดังนั้นเราควรใช้ if ตรวจสอบว่ามี 'z' อยู่ก่อนหรือไม่ เช่น

```
>>> if 'z' in letters:
        print(letters.index('z'))
else:
        print("Don't have 'z' in list")


Don't have 'z' in list
```

การวนค่าใน list :Looping through a list

เราเคยได้เห็นการวนค่าใน list ตอนใช้คำสั่ง for ไปแล้ว ทั้งทำงานร่วมกับคำสั่ง range() หรือตัว list เอง ดูตัวอย่างนี้อีกครั้งกับการวนใน letters ดังนี้

```
>>> letters = ['a', 'b', 'c', 'd', 'e']
>>> for letter in letters:
        print(letter)


a
b
c
d
e
```

การเรียงลำดับใน list : Sorting lists

Sort คือการเรียงลำดับ จะเรียงลำดับจากน้อยไปมาก

```
>>> letters = ['d', 'a', 'e', 'c', 'b']
>>> print(letters)
['d', 'a', 'e', 'c', 'b']
>>> letters.sort()
>>> print(letters)
['a', 'b', 'c', 'd', 'e']
```

เราสามารถกลับการเรียงของ list ได้ ด้วยคำสั่ง reverse()

```
>>> letters = ['d', 'a', 'e', 'c', 'b']
>>> letters.reverse()
>>> print(letters)
```

ดังนั้นถ้าเราต้องการจะเรียงจากมากไปน้อย ได้ดังนี้

```
>>> letters = ['d', 'a', 'e', 'c', 'b']
>>> print(letters)
['d', 'a', 'e', 'c', 'b']
>>> letters.sort()
>>> print(letters)
['a', 'b', 'c', 'd', 'e']
>>> letters.reverse()
>>> print(letters)
['e', 'd', 'c', 'b', 'a']
```

หรืออาจทำให้สั้นลงด้วยคำสั่ง sort() แต่ระบุวิธีการเรียงแบบ reverse ดังนี้

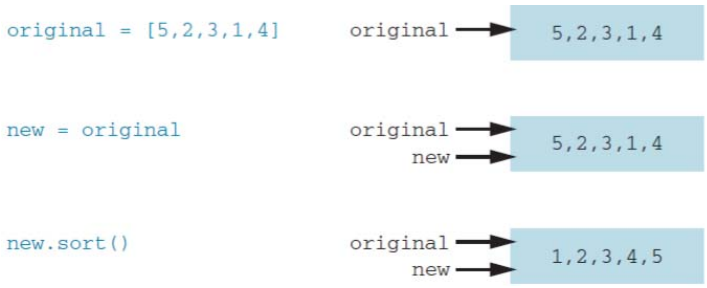
```
>>> letters = ['d', 'a', 'e', 'c', 'b']
>>> letters.sort(reverse = True)
>>> print(letters)
['e', 'd', 'c', 'b', 'a']
```

จากตัวอย่างการเรียงเราจะเห็นได้ว่า ข้อมูลใน list ของเราได้ถูกจัดเรียงใหม่ แล้วถ้าเราไม่ต้องการเปลี่ยนแปลงข้อมูลเดิมแต่อยากแสดงผลเพียงการเรียงเท่านั้น จะทำอย่างไร เราจำเป็นจะต้องคัดลอกข้อมูลไว้ก่อนเรียง เช่น

```
>>> old_list = ['Tom', 'James', 'Sarah', 'Fred']
>>> new_list = old_list[:]
>>> new_list.sort()
>>> print(old_list)
['Tom', 'James', 'Sarah', 'Fred']
>>> print(new_list)
['Fred', 'James', 'Sarah', 'Tom']
```

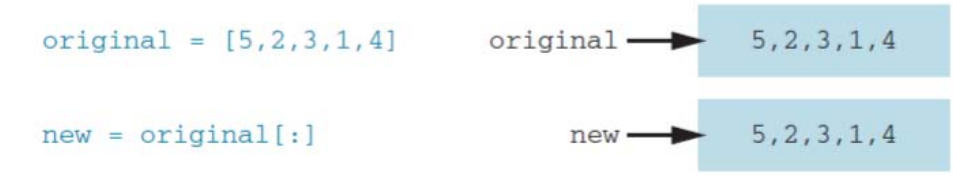
ถ้าเราสั่งเพียงแค่ new_list = old_list จะเป็นเพียงการเพิ่มชื่อให้กับค่า list ข้อมูลมีก้อนเดียวแต่มีตัวชี้ที่เดียวกันสองตัวแปร เมื่อแก้ไขข้อมูลใดข้อมูลหนึ่ง จะเหมือนกับการเปลี่ยนอีกตัวแปรไปด้วยเช่น

```
>>> old_list = ['Tom', 'James', 'Sarah', 'Fred']
>>> new_list = old_list
>>> new_list.sort()
>>> print(old_list)
['Fred', 'James', 'Sarah', 'Tom']
```



สร้างเพิ่มใหม่

คัดลอกใหม่



การเรียงแบบสร้างใหม่ โดย sorted()

```
>>> original = [5, 2, 3, 1, 4]
>>> newer = sorted(original)
>>> print original
[5, 2, 3, 1, 4]
>>> print newer
[1, 2, 3, 4, 5]
```

การเปลี่ยนแปลงไม่ได้ (immutable)

List เป็น mutable คือสามารถเปลี่ยนแปลงค่าได้
ชนิดข้อมูล Tuple คือ list ที่ไม่สามารถเปลี่ยนแปลงได้ tuple จะให้วงเล็บ แทน brackets ดังนี้

```
my_tuple = ("red", "green", "blue")
```

List ของ list : table ตาราง

ตัวแปรคือข้อมูลหนึ่งช่อง

List คือตารางของหนึ่งแถว มีได้หลายช่อง

และเราจะสร้างตารางของข้อมูลที่ช่องแนวตั้งและแนวนอนได้อย่างไร เช่น ตารางคะแนนสอบ แต่ละวิชา

classMarks →	Math	Science	Reading	Spelling
Joe	55	63	77	81
Tom	65	61	67	72
Beth	97	95	92	88

เราสามารถสร้างข้อมูลคะแนนสอบของแต่ละคนของทุกวิชาได้ เช่น

```
>>> joeMarks = [55, 63, 77, 81]
>>> tomMarks = [65, 61, 67, 72]
>>> bethMarks = [97, 95, 92, 88]
```

หรือสร้างคะแนนแต่ละวิชาของทุกคนได้เช่น

```
>>> mathMarks = [55, 65, 97]
>>> scienceMarks = [63, 61, 95]
>>> readingMarks = [77, 67, 92]
>>> spellingMarks = [81, 72, 88]
```

เราจะได้เรียนวิชาเกี่ยวกับการเก็บข้อมูลหลายๆ ค่าเช่นนี้ ในวิชา data structure
เราจะเห็นได้ว่า เราจำเป็นต้องอ้างอิงหลายตัวแปร มีวิธีที่จะใช้ตัวแปรเดียวหรือไม่ นั้นคือ list ของ list เช่น

```
>>> classMarks = [joeMarks, tomMarks, bethMarks]
>>> print(classMarks)
[[55, 63, 77, 81], [65, 61, 67, 72], [97, 95, 92, 88]]
```

หรือจะสร้างจากข้อมูลโดยตรงดังนี้

```
>>> classMarks = [[55, 63, 77, 81], [65, 61, 67, 72], [97, 95, 92, 88]]
>>> print(classMarks)
[[55, 63, 77, 81], [65, 61, 67, 72], [97, 95, 92, 88]]
```

การพิมพ์ข้อมูลดู

```
>>> for studentMark in classMarks:
        print(studentMark)


[55, 63, 77, 81]
[65, 61, 67, 72]
[97, 95, 92, 88]
```

จะเห็นได้ว่าเราแสดงข้อมูลค่อนข้างเหมือนกับ ตารางด้านบน

classMarks →	Math	Science	Reading	Spelling
Joe	55	63	77	81
Tom	65	61	67	72
Beth	97	95	92	88

แล้วเราจะอ่านข้อมูล หนึ่งช่องอย่างไร เช่น คะเนนสอบคะแนน Reading ของ joe
เรารู้ว่า joe คือคนแรก เราสามารถอ่านค่าคะแนน

```
>>> print(classMarks[0])
[55, 63, 77, 81]
```

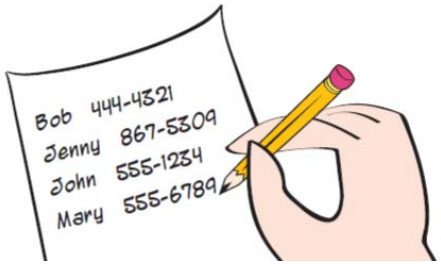
และคะแนน reading ได้ที่ช่อง 2

```
>>> print(classMarks[0][2])
77
```

ดิกชั่นนารี่ Dictionary

ถ้าเราต้องการชนิดข้อมูล ที่คล้ายๆ สมุดโทรศัพท์ ที่เราจะบันทึกชื่อเพื่อเก็บเบอร์โทรไว้ ถ้าเรารู้ชื่อก็รู้เบอร์โทร เช่นเดียวกับดิกชั่นนารี่แปลภาษารู้คำ จะรู้ความหมายที่เก็บไว้

Dictionary ในภาษา python จะมีการเชื่อมโยงสองสิ่งด้วยกัน ได้แก่ คีย์ (key) และ ค่า (value) ซึ่งแต่ item ของ Dictionary จะประกอบด้วยสองสิ่งนี้เป็นคู่กัน เราเรียกว่า key-value pair ตัวอย่างสมุดโทรศัพท์ ชื่อจะทำหน้าที่เป็น key เบอร์โทรคือ value



เราสามารถสร้าง Dictionary โดยใช้ ปีกกา (curly braces) แทน brackets ของ list

```
>>> phoneNumbers = {}
```

เราสามารถสร้าง Dictionary โดยใช้ ปีกกา (curly braces) แทน brackets ของ list


```
>>> phoneNumbers = {}
```

เราสามารถเพิ่มข้อมูลโดยระบุ key และ value ดังนี้

```
>>> phoneNumbers["John"] = "555-1234"
```

ดูผลลัพท์

```
print(phoneNumbers)
{'John': '555-1234'}
```

เราจะเพิ่มข้อมูลอื่น ถ้าเป็น list เราจะใช้ append() แต่นี้คือ Dictionary เราจะเพิ่มชื่อและเบอร์โทรคนอื่นด้วย

```
>>> phoneNumbers["Mary"] = "555-6789"
>>> phoneNumbers["Bob"] = "444-4321"
>>> phoneNumbers["Jenny"] = "867-5309"
```

แสดงผลได้ดังนี้

```
>>> print(phoneNumbers)
{'John': '555-1234', 'Jenny': '867-5309', 'Mary': '555-6789', 'Bob': '444-4321'}
```

ดูเฉพาะเบอร์ 'Mary'

```
>>> print(phoneNumbers["Mary"])
555-6789
```

ข้อเปรียบเทียบระหว่าง list และ Dictionary เหมือนกันคือ

- ทั้งสองสามารถเก็บข้อมูลชนิดไหนก็ได้
- มีวิธีที่สามารถหาดูว่าค่าเก็บอะไรได้

สิ่งที่ต่างกันคือ

- List เมื่อเราเพิ่มของตามลำดับ และเราวนดูค่า ทุกอย่างที่อยู่ใน list จะเป็นไปตามลำดับ คือมีลำดับ (ordered) ขณะที่ Dictionary ไม่มีลำดับ เราเพิ่มของลง Dictionary ตามลำดับ แต่เมื่อแสดงผลจะไม่ตามลำดับ
- List จะใช้ตำแหน่งในการเข้าถึงข้อมูล ขณะที่ Dictionary จะใช้ key ในการเข้าถึงข้อมูล

```
>>> print myList[3]
'eggs'
>>> print myDictionary["John"]
'555-1234'
```

เราสามารถดูค่าเฉพาะ key ได้ โดยคำสั่ง keys() หรือที่เรียกว่า method เมธอด

```
>>> phoneNumbers.keys()
dict_keys(['John', 'Jenny', 'Mary', 'Bob'])
```

และดูเฉพาะค่าที่เก็บไว้ด้วย เมธอด values()

```
>>> phoneNumbers.values()
dict_values(['555-1234', '867-5309', '555-6789', '444-4321'])
```

Key ของ Dictionary เป็นได้เฉพาะชนิดข้อมูลที่เป็น imutable คือไม่สามารถเปลี่ยนแปลงค่าได้ เช่น
(Booleans, integers, floats, strings, tuples)
แต่ไม่ใช้พวกที่สามารถเปลี่ยนค่าได้ (mutables) ส่วนชนิดของ value เป็นชนิดอะไรก็ได้

จากที่กล่าวไว้ dictionary ไม่มีลำดับ แล้วเราสามาถทีจะแสดงผลแบบเป็นลำดับได้หรือไม่ ดูการวนค่าพิมพ์แสดงผลทั่วไป ดังนี้

```
>>> for key in phoneNumbers:
        print(key, phoneNumbers[key])


John 555-1234
Jenny 867-5309
Mary 555-6789
Bob 444-4321
```

จะเห็นได้ว่า ลำดับ ไม่เป็นไปตามที่เรา เพิ่มข้อมูล และ ตามลำดับตัวอักษร ถ้าเราต้องการลำดับตามตัวอักษร เราจะใช้ sorted เข้าช่วยดังนี้

```
>>> for key in sorted(phoneNumbers.keys()):
        print(key, phoneNumbers[key])


Bob 444-4321
Jenny 867-5309
John 555-1234
Mary 555-6789
```

เราจะเห็นได้ว่าข้อมุลเรียงตามลำดับ key แต่ถ้าเราอยากเรียงตามลำดับค่า value ที่เก็บหรือเบอร์โทร

```
>>> for value in sorted(phoneNumbers.values()):
        for key in phoneNumbers.keys():
            if phoneNumbers[key] == value:
                print(key, value)


Bob 444-4321
John 555-1234
Mary 555-6789
Jenny 867-5309
```

จะเห็นได้ว่าวิธีการ ค่อนข้างซับซ้อน ถ้าเรารู้เพียงค่า เราอยากจะรู้ว่า key คืออะไร เราจำเป็นต้องวน key ทุกตัว แล้วใช้ค่า key ดูค่าที่เก็บตรงตรงกับค่าที่เราต้องการหา key หรือไม่ ถ้าตรงแสดงว่านี้คือ key
ในทางตรงกันข้ามเรารู้ค่า key เราทราบ value ได้ทันที

การลบค่าใน Dectionary เราจำเป็นจะต้องระบุ key และลบด้วยคำสั่ง del

```
>>> del(phoneNumbers['Mary'])
>>> print(phoneNumbers)
{'John': '555-1234', 'Jenny': '867-5309', 'Bob': '444-4321'}
```

เราสามารถล้างทุกค่า ด้วยคำสั่ง clear()

```
>>> phoneNumbers.clear()
>>> print(phoneNumbers)
{}
```

เช่นเดียวกันกับ list เราสามารถตรวจสอบค่า ว่าอยู่ใน Dictionary หรือไม่ด้วยคำสั่ง in แต่จะตรวจได้ เฉพาะ key เท่านั้น

```
>>> 'Johe' in phoneNumbers
False
>>> 'John' in phoneNumbers
True
>>> '555-1234' in phoneNumbers
False
```

แล้วถ้าอยากรู้ว่ามีค่าหรือเบอร์โทรเราจำเป็นจะต้องคนหาโดยระบุเฉพาะส่วนของ values เท่านั้น ดังนี้

```
>>> '555-1234' in phoneNumbers.values()
True
```

ทบทวน

- List คืออะไร
- เราจะเพิ่มข้อมูลอย่างไรลงใน list
- ลบข้อมูลใน list
- ค้นหาข้อมูลใน list
- เรียงข้อมูลใน list
- การคัดลอกข้อมูลใน list
- เกี่ยวกับ tuples
- List ของ list หรือตาราง
- Dictionary ในภาษา Python

ทดสอบความรู้

1. จงแสดงโค้ดการเพิ่มค่าลง list มาสองวิธี
2. จงแสดงโค้ดการลบค่าใน list มาสองวิธี
3. จงแสดงโค้ดการเรียงลำดับสิ่งของใน list มาสองวิธี โดยไม่มีการเปลี่ยนแปลงค่าใน list
4. จงแสดงโค้ดการตรวจสอบว่ามี item นี้ใน list หรือไม่
5. จงแสดงการหาตำแหน่งช่อง index ใน list
6. Tuple คืออะไร
7. เราจะสร้างตารางอย่างไร
8. เราจะอ่านค่าหนึ่งช่องในตารางอย่างไร
9. Dictionary คืออะไร
10. เราเพิ่มค่าลงใน Dictionary อย่างไร
11. เราดูค่าใน Dictionary ด้วย key อย่างไร

จงเขียนโปรแกรมต่อไปนี้

1. จงเขียนโปรแกรมป้อนชื่อมา 5 ชื่อ

```
Enter 5 names:
```

1. จงเขียนโปรแกรมป้อนชื่อมา 5 ชื่อ

```
Enter 5 names:
Tony
Paul
Nick
Michel
Kevin
The names are Tony Paul Nick Michel Kevin
```

2. จากข้อหนึ่ง แก้ไขโปรแกรม เรียงลำดับชื่อตามตัวอักษร จากมากไปน้อย (Z ไปยัง A) ได้และให้พิมพ์ค่า list เดิมออกมาด้วย

```
Enter 5 name
Arm
Brid
Game
Golf
Beer
Original: The names are Arm Brid Game Golf Beer
SortedL The names are Golf Game Brid Beer Arm
```

3. แก้ไขโปรแกรมให้ระบุว่า ให้แสดงเพิ่ม ค่าที่เราพิมพ์คนที่สามคือใคร

```
Enter 5 name
Arm
Brid
Game
Golf
Beer
Original: The names are Arm Brid Game Golf Beer
SortedL The names are Golf Game Brid Beer Arm
The third name you entered is: Game
```

4. โปรแกรมสามารถเลือกตำแหน่ง และแทนที่ชื่อที่ตำแหน่งนั้นได้

```
Enter 5 names:
Tony
Paul
Nick
Michel
Kevin
The names are Tony Paul Nick Michel Kevin
Replace one name.  Which one? (1-5): 4
New name: Peter
The names are Tony Paul Nick Peter Kevin
```

5. เขียนโปรแกรมทำหน้าที่เหมือนเป็น Dictionary ของคุณ โดยให้สามารถเลือกได้ว่า จะเพิ่มค่า (a) หรือดูค่า (l) ถ้าเพิ่มค่าเสร็จให้แสดงคำว่า
   Word added! (เพิ่มคำแล้ว)
   และถ้าดูค่าแล้วไม่มีคำนี้ใน Dictionary ของคุณ
   The word isn't in the dictionary yet. (คำนี้ยังไม่มีใน dictionary ของคุณ)
   โปรแกรมจะวนถามไปเรื่อยๆ

```
Add or look up a word (a/l)? a
Type the word: computer
Type the definition: A machine that does very fast math
Word added!
Add or look up a word (a/l)? l
Type the word: computer
A machine that does very fast math
Add or look up a word (a/l)? l
Type the word: qwerty
That word isn't in the dictionary yet.
```

```
>>>
Add or look up a word(a/l)? a
Type the word: cat
Type the defination: แมว
Word added!
Add or look up a word(a/l)? a
Type the word: bat
Type the defination: ค้างคาว
Word added!
Add or look up a word(a/l)? l
Type the word: cat
แมว
Add or look up a word(a/l)? l
Type the word: rat
That word isn't in the dictionary yet.
Add or look up a word(a/l)? a
Type the word: rat
Type the defination: หนู
Word added!
Add or look up a word(a/l)? l
Type the word: rat
หนู
Add or look up a word(a/l)? l
Type the word: bat
ค้างคาว
Add or look up a word(a/l)?
```