w14-Lec

Other Collection Types: Sets and Dictionaries

Assembled for 204111

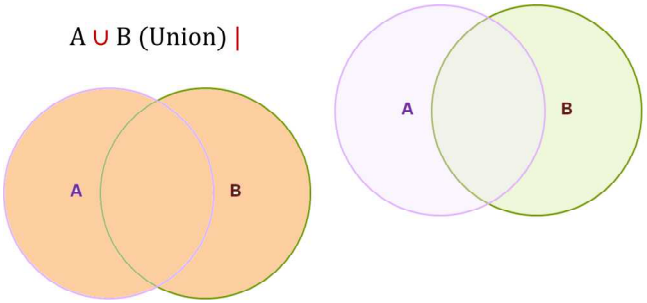
by Kittipitch Kuptavanich

Sets

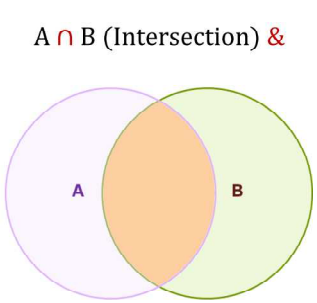
- **Sets เป็นวัตถุประเภท Collection ที่มีสมาชิกไม่ซ้ำกัน (ลักษณะคล้าย Set ทางคณิตศาสตร์) ประกอบด้วย**
  - `set`
  - `frozenset` (Immutable Collection)
- **เราสามารถใช้ Set เพื่อทำ Operation ต่าง ๆ ได้แก่**
  - **การกำจัด Element ที่ซ้ำออก**
  - **การหายูเนียน (Union: ∪)**
  - **การหาอินเตอร์เซคชัน (Intersection: ∩)**
  - **การหาผลต่างของเซต (Difference: \)**
  - **การหาผลต่างสมมาตร (Symmetric Difference: ⊕)**

Common Set Operations

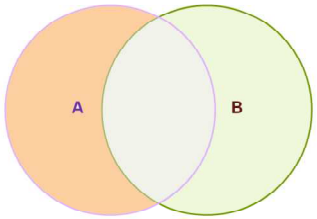
A ∪ B (Union) |



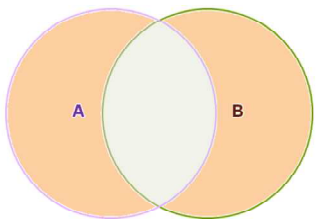
A ∩ B (Intersection) &



A \ B (Difference) -



A ⊕ B (Symmetric Difference) ^



Examples

- **เราใช้เครื่องหมายปีกกา `{}` และ Comma `,` เพื่อสร้าง Set**

```
01 s = {2, 3, 5, 7, 9}
02 print(3 in s)            # prints True
03 print(4 in s)            # prints False
04 for x in range(10):
05     if (x not in s):
06         print(x, end="") # prints 0 1 4 6 8
```

- **เราสามารถใช้ Operation เช่น `in` หรือ Loop ได้เหมือนใน Collection Type อื่น ๆ**

Properties of Sets

- **Set เป็น Collection Type แบบไม่มีลำดับ**

```
09 s = set([2, 4, 8])
10 print(s)              # prints {8, 2, 4}
11 for element in s:
12     print(element, end="")  # prints 8 2 4
```

- **Element แต่ละตัวจะไม่ซ้ำกัน**

```
14 s = set([2, 2, 2])
15 print(s)              # prints {2}
16 print(len(s))         # prints 1
```


Properties of Sets [2]

- **แต่ละ Element ต้องมีคุณสมบัติ Immutable เช่น `str`, `int` หรือ Atomic Type อื่น ๆ**

```
>>> a = ["lists", "are", "mutable"]
>>> s = set([a])
TypeError: unhashable type: 'list'

>>> s1 = set(["sets", "are", "mutable", "too"])
>>> s2 = set([s1])
TypeError: unhashable type: 'set'
```

- **Set ไม่สามารถมี Element เป็นชนิด List หรือ Set ได้**
  - **Set _____ Element เป็นชนิด Tuple ได้**

Creating Sets

```
>>> # Create an empty set
>>> s = set()
>>> print(s)     # prints set()
set()

>>> # Create a set from a list
>>> s = set(["cat", "cow", "dog"])
>>> print(s)
{'cow', 'cat', 'dog'}

>>> # Create a set from any iterable object
>>> s = set("wahoo")
>>> print(s)
{'a', 'h', 'w', 'o'}
```


https://docs.python.org/3/library/stdtypes.html#set

Set Operations in Python

Operation	Result	Notes
len(s)	cardinality of set ***s***	
s.copy()	new set with a shallow copy of ***s***	
s.pop()	remove and return an arbitrary element from ***s***; raises KeyError if empty	
s.clear()	remove all elements from set ***s***	
x in s	test ***x*** for membership in ***s***	
x not in s	test ***x*** for non-membership in ***s***	
s.add(x)	add element ***x*** to set ***s***	
s.remove(x)	remove ***x*** from set ***s***; raises KeyError if not present	
s.discard(x)	Remove element ***x*** from the set if it is present.	

https://docs.python.org/3/library/stdtypes.html#set

Set Operations in Python [2]

Operation	≡	Result	Notes
s.issubset(t)	s <= t	test whether every element in ***s*** is in ***t***	
s.issuperset(t)	s >= t	test whether every element in ***t*** is in ***s***	
s.union(t)	s | t	new set with elements from both ***s*** and ***t***	
s.intersection(t)	s & t	new set with elements common to ***s*** and ***t***	
s.symmetric_difference(t)	s ^ t	new set with elements in either ***s*** or ***t*** but not both	
s.update(t)	s |= t	Same as s = s | t	
s.intersection_update(t)	s &= t	Same as s = s & t	
s.difference_update(t)	s -= t	Same as s = s - t	
s.symmetric_difference_update(t)	s ^= t	Same as s = s ^ t	

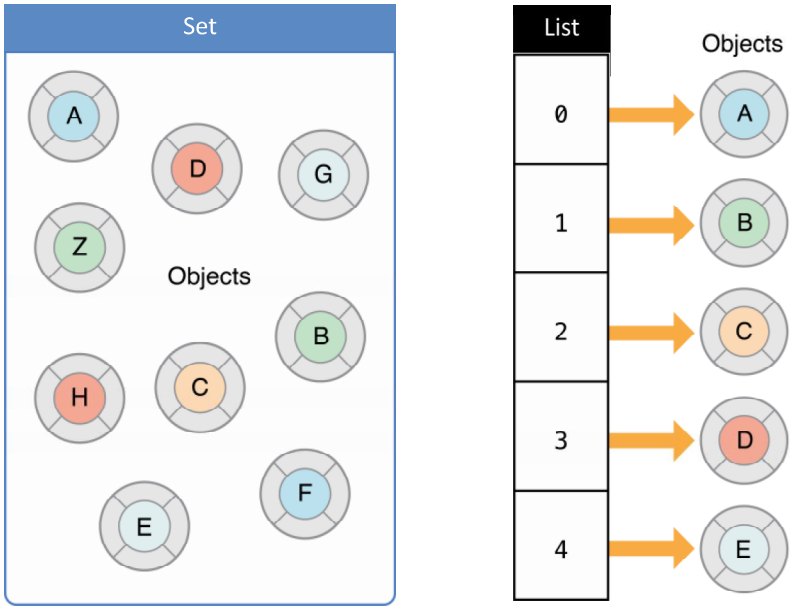
Example Using Sets

```
01 def repeats(a):
02     seen = set()
03     seenAgain = set()
04     for element in a:
05         if (element in seen):
06             seenAgain.add(element)
07         seen.add(element)
08     return sorted(seenAgain)
```

- ฟังก์ชัน `repeats()` ทำหน้าที่_____

```
>>> print(repeats([2, 5, 3, 4, 6, 4, 2]))
[2, 4]
```


Sets vs Lists

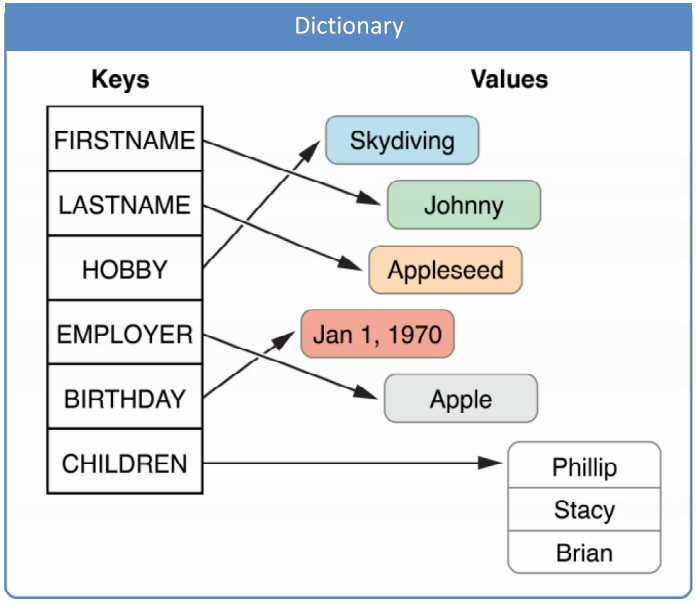


https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/Collections/Collections.html

Dictionaries

- **Dictionary (หรือ Hash Map) มีลักษณะคล้าย List**
  - **Index ใน List ต้องเป็นเลขจำนวนเต็ม (≥ 0)**
  - **แต่ Index ใน Dictionary สามารถเป็นข้อมูลได้ (เกือบ) ทุกชนิด**
- **เราเรียก Index ใน Dictionary ว่า `key`**
- **และเรียกค่าที่เก็บไว้ใน Index นั้น ๆ ว่า `value`**
- **Dictionary เป็นข้อมูลประเภท Mapping Type คือ เป็นการ Map `key` → `value`**

Dictionaries [2]



https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/Collections/Collections.html

Dictionaries [3]

- เราใช้ฟังก์ชัน **Built-in dict()** ในการสร้าง **Dictionary** เปล่า
  - เช่นหากต้องการสร้าง **Dictionary** สำหรับเก็บหมายเลขโทรศัพท์

```
>>> d = dict()
>>> d['fred'] = '555-1212'
>>> d['wilma'] = '555-3456'
>>> print(d['fred'])
'555-1212'

>>> # now fred and wilma get married, so...
>>> d['fred'] = d['wilma']
>>> print(d['fred'])
'555-3456'
```


Think Python: How to Think Like a Computer Scientist

Properties of Dictionaries

- **Dictionary** เป็นการผูกค่าระหว่าง `key` และ `value`
  - `key` เป็น **Set** ดังนั้น `key` แต่ละตัวต้องมีลักษณะ **Immutable, ไม่ซ้ำกัน และไม่มีลำดับ (Unordered)**
    - **Strings, number**
    - **Tuples (ที่มี Element เป็น Immutable)**
      - ดังนั้น ~~`([2], 3)`~~ เป็น **key** ไม่ได้ เนื่องจากเป็น **Tuple** ที่ประกอบด้วย **List (Mutable)**
  - `value` สามารถเป็นข้อมูลประเภทใดก็ได้ ไม่มีข้อจำกัด

Think Python: How to Think Like a Computer Scientist

Creating a Dictionary

- ใช้เครื่องหมายปีกกา `{}`

```
>>> # Empty dictionary
>>> d = {}
>>> print(d)
{}

>>> # key:value
>>> d = {"cow": 5, "dog": 98, "cat": 1}
>>> print(d)
{'dog': 98, 'cow': 5, 'cat': 1}
```

- เนื่องจาก **Dictionary** ไม่มีลำดับที่ตายตัว ผลลัพธ์ที่ได้จากฟังก์ชัน `print()` ของแต่ละคนอาจมีลำดับต่างกัน

Think Python: How to Think Like a Computer Scientist

Creating a Dictionary [2]

- ใช้ฟังก์ชัน `dict()`

```
>>> d = dict()         # Empty dictionary
>>> print(d)
{}

>>> # From a Collection of Tuple (key, value) i.e. a List
>>> pairs = [("one", 1), ("two", 2), ("three", 3)]
>>> d = dict(pairs)
>>> print(d)
{'one': 1, 'three': 3, 'two': 2}

>>> # function parameters
>>> d = dict(one=1, two=2, three=3)
>>> print(d)
{'one': 1, 'three': 3, 'two': 2}
```


https://docs.python.org/3/library/stdtypes.html#mapping-types-dict

Dictionary Operations

Operation	Result	Notes
len(d)	Return the number of items (key-value pairs) in the dictionary ***d***.	
d.clear()	Remove all items from the dictionary ***d***.	
d.copy()	Return a shallow copy of the dictionary ***d***.	
d.keys()	Return a new view of the dictionary's keys.	
d.popitem()	Remove and return an arbitrary (key, value) pair from the dictionary. If the dictionary is empty, calling popitem() raises a KeyError.	

for key in d	Iterate over all keys in ***d*** For example:
	`d = {"cow": 5, "dog": 98, "cat": 1}`
	`for key in d:`
	`  print(key, d[key])`	

https://docs.python.org/3/library/stdtypes.html#mapping-types-dict

Dictionary Operations [2]

Operation	Result	Notes
key in d	Return True if ***d*** has a key ***key***, else False.	
key not in d	Equivalent to not ***key*** in ***d***.	
d[key]	Return the item of ***d*** with key ***key***. Raises a KeyError if ***key*** is not in the map.	
get(key[,default])	Return the value for ***key*** if ***key*** is in the dictionary, else ***default***. If ***default*** is not given, it defaults to None, so that this method never raises a KeyError.	
d[key] = value	Set ***d***[***key***] to ***value***.	
del d[key]	Remove ***d***[***key***] from ***d***. Raises a KeyError if ***key*** is not in the map.	
update([other])	Update the dictionary with the ***key***/***value*** pairs from ***other***, overwriting existing keys. Return None.	

https://docs.python.org/3/tutorial/datastructures.html#dictionaries

Example Operations

```
>>> tel = {'jack': 4098, 'sape': 4139}
>>> tel['guido'] = 4127
>>> tel
{'sape': 4139, 'guido': 4127, 'jack': 4098}
>>> tel['jack']
4098
>>> del tel['sape']
>>> tel['irv'] = 4127
>>> tel
{'guido': 4127, 'irv': 4127, 'jack': 4098}

>>> list(tel.keys())
['irv', 'guido', 'jack']
>>> sorted(tel.keys())
['guido', 'irv', 'jack']

>>> 'guido' in tel
True
>>> 'jack' not in tel
False
```

Example Using Dictionaries

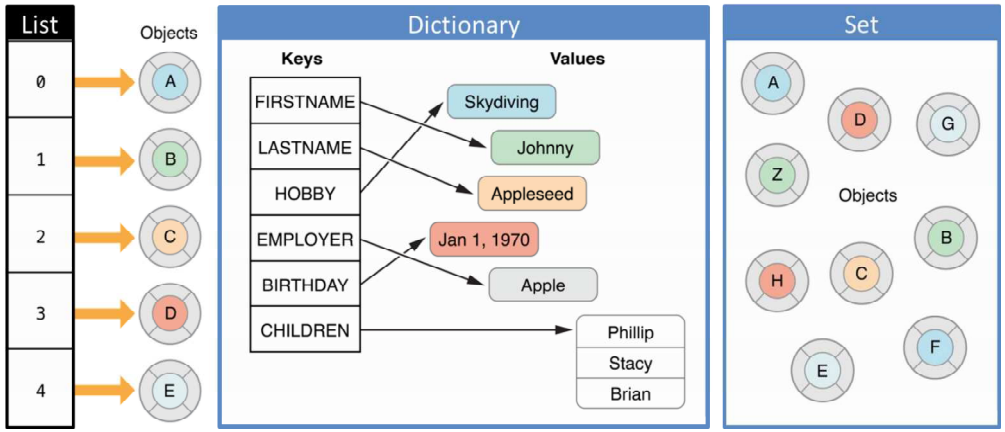
```
04 def mostFrequent(a):
05     maxValue = None
06     maxCount = 0
07     frequency = dict()
08     for element in a:
09         if element in frequency:
10             count = frequency[element]
11         else:
12             count = 0
13         count += 1
14         frequency[element] = count
15         if (count > maxCount):
16             maxCount = count
17             maxValue = element
18     return maxValue
19
20 print(mostFrequent([2, 5, 3, 4, 6, 4, 2, 4, 5])) # prints 4
```


Example Using Dictionaries [2]

```
24 def mostFrequent(a):
25     maxValue = None
26     maxCount = 0
27     frequency = dict()
28     for element in a:
29         count = 1 + frequency.get(element, 0)
30         frequency[element] = count
31         if (count > maxCount):
32             maxCount = count
33             maxValue = element
34     return maxValue
35
36 print(mostFrequent([2, 5, 3, 4, 6, 4, 2, 4, 5])) # prints 4
```

- สังเกตการใช้ **Method** `dict.get()` โดยมี **0** เป็นค่า **Default** ที่บรรทัดที่ **29**

Collections Recap



https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/Collections/Collections.html

References

- https://docs.python.org/3/library/stdtypes.html#set
- https://docs.python.org/3/tutorial/datastructures.html#sets
- http://www.cs.cmu.edu/~112/notes/notes-sets.html
- https://docs.python.org/3/library/stdtypes.html#mapping-types-dict
- https://docs.python.org/3/tutorial/datastructures.html#dictionaries
- http://www.cs.cmu.edu/~112/notes/notes-maps.html