Chapter 10
集合 (Set)

10.1 建立集合

- 集合(set)的基本觀念是無序且元素值是唯一的。
  - 可以看成是字典的 key，每個 key 都是唯一的。
  - 集合內元素是不可變的(immutable)，但是集合本身是可變的(mutable)，可以增加或刪除集合的元素。
  - 常見的集合元素種類有整數、浮點數、字串、元組等。
  - 串列、字典、集合...等不可以是集合元素。

10.1 建立集合

- Python可以使用大括號 "{}" 或 set() 函數建立集合。

```
1   setLang = {"Python", "C", "Java"}
2   setA = {1, 2, 3, 4, 5}
3   setB = {1, 1, 2, 2, 2, 3, 4, 5, 5}
4
5   print(setLang)
6   print(setA)
7   print(setB)
8
9   setLangList = {["Python", "C"], ["Java", "PHP"]}
```

```
{'C', 'Python', 'Java'}
{1, 2, 3, 4, 5}
{1, 2, 3, 4, 5}
Traceback (most recent call last):
  File "e:\PythonTest\test-CH10.py", line 9, in <module>
    setLangList = {["Python", "C"], ["Java", "PHP"]}
TypeError: unhashable type: 'list'
```

```
1   setDL = set("DeepLearning")
2   print(setDL)
3
4   setLang = set(["Python", "C", "Java"])
5   print(setLang)
```

```
{'i', 'a', 'D', 'p', 'e', 'r', 'g', 'n', 'L'}
{'C', 'Python', 'Java'}
```

10.1 建立集合

- 集合的基數(cardinality)：
  - 指集合內元素的個數，可以用 len()函數來取得。

```
1   setDL = set("DeepLearning")
2   print(f"{setDL = }, 基數為：{len(setDL)}")
3
4   setLang = set(["Python", "C", "Java"])
5   print(f"{setLang = }, 基數為：{len(setLang)}")
```

```
setDL = {'r', 'a', 'p', 'e', 'i', 'L', 'D', 'n', 'g'}, 基數為：9
setLang = {'C', 'Python', 'Java'}, 基數為：3
```

- 建立空集合：

```
1   dictEmpty = {}        #建立空字典
2   setEmpty = set()      #建立空集合
3
4   tupleEmpty = ()       #建立空元組
5
6   print(f"{type(dictEmpty) = }")
7   print(f"{type(setEmpty) = }")
8   print(f"{type(tupleEmpty) = }")
```

```
type(dictEmpty) = <class 'dict'>
type(setEmpty) = <class 'set'>
type(tupleEmpty) = <class 'tuple'>
```

10.1 建立集合

- Ex 10-1: 將串列中重複的元素刪除。

```
1   listFruits = ["Apple", "Banana", "Orage",
2                 "Apple", "Orage", "Apple",
3                 "Banana", "Apple", "Watermelon"]
4
5   setFruits = set(listFruits)
6   listFruitsNew = list(setFruits)
7   print(listFruitsNew)
```

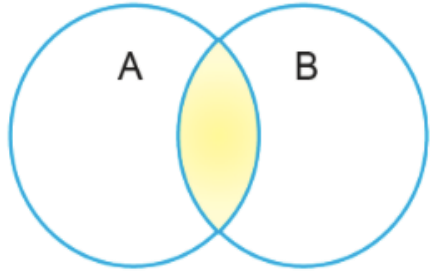
```
['Apple', 'Banana', 'Watermelon', 'Orage']
```

10.2 集合的操作

- 集合的操作有：

Python 符號	說明
&	交集
|	聯集
-	差集
^	對稱差集
==	等於
!=	不等於
in	是成員
not in	不是成員

10.2 集合的操作

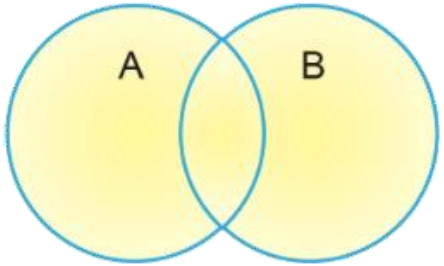


- 交集(intersection)：
  - 有A和B兩個集合，如果想獲得相同的元素，則可以使用交集。
  - 交集可透過符號 & 取得，或是使用 intersection() 方法獲得。
  - Ex 10-2：有數學與物理2個夏令營，請列出同時參加這2個夏令營的成員。

```
1   setMath = {"Strange", "Peter", "Tony"}
2   setPhysics = {"Peter", "Tony", "Benner"}
3   setBoth1 = setMath & setPhysics
4   print(f"同時參加兩個夏令營的人有：{setBoth1}")
5   setBoth2 = setMath.intersection(setPhysics)
6   print(f"同時參加兩個夏令營的人有：{setBoth2}")
```

```
同時參加兩個夏令營的人有：{'Tony', 'Peter'}
同時參加兩個夏令營的人有：{'Tony', 'Peter'}
```

10.2 集合的操作

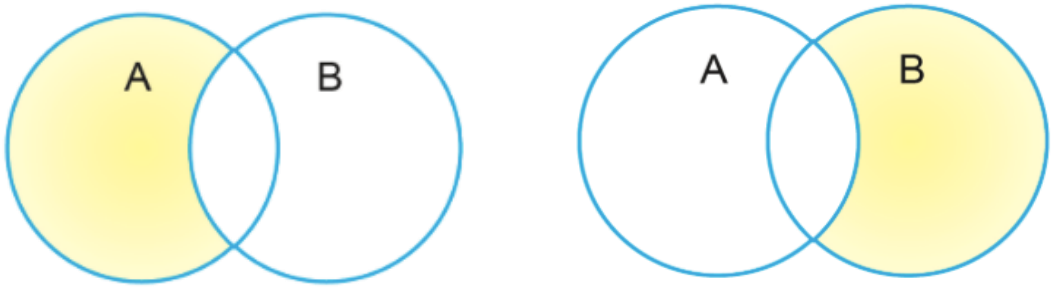


- 聯集(union)：
  - 有A和B兩個集合，如果想獲得所有的元素，則可以使用聯集。
  - 聯集可透過符號 | 取得，或是使用 union() 方法獲得。
  - Ex 10-3：有數學與物理2個夏令營，請列出所有參加這2個夏令營的成員。

```
1   setMath = {"Strange", "Peter", "Tony"}
2   setPhysics = {"Peter", "Tony", "Benner"}
3   setAny1 = setMath | setPhysics
4   print(f"參加任一個夏令營的人有：{setAny1}")
5   setAny2 = setMath.union(setPhysics)
6   print(f"參加任一個夏令營的人有：{setAny2}")
```

```
參加任一個夏令營的人有：{'Peter', 'Tony', 'Benner', 'Strange'}
參加任一個夏令營的人有：{'Peter', 'Tony', 'Benner', 'Strange'}
```

10.2 集合的操作

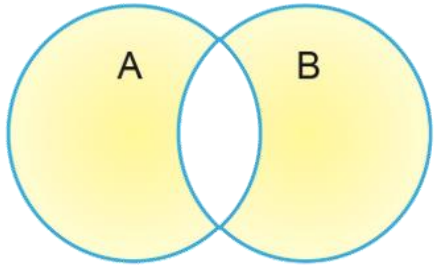


- 差集(difference)：
  - 有A和B兩個集合，如果想獲得屬於A/B集合元素，同時不屬於B/A集合則可以使用差集(A-B) / (B-A) 。(專屬於其中一個集合的元素)
  - 差集可透過符號 - 取得，或是使用 difference() 方法獲得。
  - Ex 10-4：有數學與物理2個夏令營，請列出只參加數學夏令營的成員以及只參加物理夏令營的成員。

```
1   setMath = {"Strange", "Peter", "Tony"}
2   setPhysics = {"Peter", "Tony", "Benner"}
3   setMathOnly1 = setMath - setPhysics
4   print(f"只參加數學夏令營的人有：{setMathOnly1}")
5   setMathOnly2 = setMath.difference(setPhysics)
6   print(f"只參加數學夏令營的人有：{setMathOnly2}")
7
8   setPhyOnly1 = setPhysics - setMath
9   print(f"只參加物理夏令營的人有：{setPhyOnly1}")
10  setPhyOnly2 = setPhysics.difference(setMath)
11  print(f"只參加物理夏令營的人有：{setPhyOnly2}")
```

```
只參加數學夏令營的人有：{'Strange'}
只參加數學夏令營的人有：{'Strange'}
只參加物理夏令營的人有：{'Benner'}
只參加物理夏令營的人有：{'Benner'}
```

10.2 集合的操作



- 對稱差集(symmetric difference)：
  - 有A和B兩個集合，如果想獲得屬於A或B集合元素，但是排除同時屬於A和B集合的元素。（只屬於其中一個集合的元素）
  - 對稱差集可透過符號 ^ 取得，或是使用 symmetric_difference() 方法獲得。
  - Ex 10-5：有數學與物理2個夏令營，請列出只參加其中一個夏令營的成員。

```
1   setMath = {"Strange", "Peter", "Tony"}
2   setPhysics = {"Peter", "Tony", "Benner"}
3   setOnlyOne1 = setMath ^ setPhysics
4   print(f"只參加其中一個夏令營的人有：{setOnlyOne1}")
5   setOnlyOne2 = setMath.symmetric_difference(setPhysics)
6   print(f"只參加其中一個夏令營的人有：{setOnlyOne2}")
```

```
只參加其中一個夏令營的人有：{'Strange', 'Benner'}
只參加其中一個夏令營的人有：{'Strange', 'Benner'}
```

10.2 集合的操作

- 等於：
  - 判斷兩集合是否相等，可透過符號 == 取得，如果相等傳回True，否則傳回False。
  - Ex 10-6：測試兩集合是否相等。

```
1   setA = {1, 2, 3, 4, 5}
2   setB = {1, 1, 2, 2, 2, 3, 4, 5, 5}
3   setC = {2, 4, 6, 8, 10}
4
5   print(f"集合 A 與 B 相等: {setA == setB}")
6   print(f"集合 A 與 C 相等: {setA == setC}")
```

```
集合 A 與 B 相等: True
集合 A 與 C 相等: False
```

10.2 集合的操作

- 不等於：
  - 判斷兩集合是否不相等，可透過符號 != 取得，如果不相等傳回True，否則傳回False。
  - Ex 10-7：測試兩集合是否不相等。

```
1   setA = {1, 2, 3, 4, 5}
2   setB = {1, 1, 2, 2, 2, 3, 4, 5, 5}
3   setC = {2, 4, 6, 8, 10}
4
5   print(f"集合 A 與 B 不相等: {setA != setB}")
6   print(f"集合 A 與 C 不相等: {setA != setC}")
```

```
集合 A 與 B 不相等: False
集合 A 與 C 不相等: True
```

10.2 集合的操作

- 是成員：
  - Python的關鍵字 in 可以測試元素是否是集合的元素成員。
  - Ex 10-8：in 的應用。

```
1   setStr = set("Orange")
2   print(f"字元 a 屬於 setStr? {'a' in setStr}")
3   print(f"字元 b 屬於 setStr? {'b' in setStr}")
4
5   setMath = {"Strange", "Peter", "Tony"}
6   print(f"Strange 參加數學夏令營? {'Strange' in setMath}")
7   print(f"Banner 參加數學夏令營? {'Banner' in setMath}")
```

```
字元 a 屬於 setStr? True
字元 b 屬於 setStr? False
Strange 參加數學夏令營? True
Banner 參加數學夏令營? False
```

```
1   setMath = {"Strange", "Peter", "Tony"}
2   for name in setMath:
3       print(name)
```

```
Peter
Tony
Strange
```

10.2 集合的操作

- 不是成員：
  - Python的關鍵字 not in 可以測試元素是否不是集合的元素成員。
  - Ex 10-9：not in 的應用。

```
1   setStr = set("Orange")
2   print(f"字元 a 屬於 setStr? {'a' not in setStr}")
3   print(f"字元 b 屬於 setStr? {'b' not in setStr}")
4
5   setMath = {"Strange", "Peter", "Tony"}
6   print(f"Strange 參加數學夏令營? {'Strange' not in setMath}")
7   print(f"Banner 參加數學夏令營? {'Banner' not in setMath}")
```

```
字元 a 屬於 setStr? False
字元 b 屬於 setStr? True
Strange 參加數學夏令營? False
Banner 參加數學夏令營? True
```

10.3 適合集合的方法

方法	說明
add()	加一個元素到集合
clear()	刪除集合所有元素
copy()	複製集合
difference_update()	刪除集合內與另一集合重複的元素
discard()	如果是集合成員則刪除
intersection_update()	可以使用交集更新集合內容
isdisjoint()	如果 2 個集合沒有交集則回傳 True
issubset()	如果另一個集合包含這個集合則回傳 True
issuperset()	如果這個集合包含另一個集合則回傳 True
pop()	傳回所刪除的元素，如果是空集合則傳回 False
remove()	刪除指定元素，如果此元素不存在則會造成程式 KeyError 異常中止
symmetric_difference_update()	使用對稱差集更新集合內容
update()	使用聯集更新集合內容

10.3 適合集合的方法

- add()
  - 增加一個元素。

    set.add(新增元素)
  - 將新增元素新增到集合中。

```
1    setMath = {"Strange", "Peter", "Tony"}
2    setMath.add("Banner")
3    print(setMath)
```

```
{'Strange', 'Tony', 'Banner', 'Peter'}
```

10.3 適合集合的方法

- copy()
  - 複製舊集合成為一個新集合（淺拷貝）。 deepcopy 有需要嗎？

setNew = setOld.copy()

```
setNum = {1, 2, 3}
setNum1 = setNum
setNum1.add(4)
print(f"賦值:  {setNum = }")
print(f"賦值: {setNum1 = }")

setNumCopy = setNum.copy()
setNumCopy.add(10)
print(f"淺拷貝:     {setNum = }")
print(f"淺拷貝: {setNumCopy = }")
```

```
賦值:  setNum = {1, 2, 3, 4}
賦值: setNum1 = {1, 2, 3, 4}
淺拷貝:     setNum = {1, 2, 3, 4}
淺拷貝: setNumCopy = {1, 2, 3, 4, 10}
```

```
setList = {(1, 2), (3, 4, 5)}
setListCopy = setList.copy()
setListCopy.add((6, 7))
print(setList)
print(setListCopy)
```

```
{(3, 4, 5), (1, 2)}
{(3, 4, 5), (6, 7), (1, 2)}
```

10.3 適合集合的方法

- remove()
  - 刪除指定元素，若指定元素不在集合內，會產生 KeyError 的程式異常。

set.remove(欲刪除的元素)

```
1   setList = {(1, 2), (3, 4, 5)}
2   setListCopy = setList.copy()
3   setList.remove((1, 2))
4   print(setList)
5   setListCopy.remove((3, 4))
6   print(setListCopy)
```

```
{(3, 4, 5)}
Traceback (most recent call last):
  File "e:\PythonTest\test.py", line 5, in <module>
    setListCopy.remove((3, 4))
KeyError: (3, 4)
```

  - 可用 discard() 避免 KeyError 的程式異常。

10.3 適合集合的方法

- discard()
  - 刪除指定元素，若指定元素不在集合內也不會有錯誤產生。

retVal = set.discard(欲刪除的元素)

```
1   setList = {(1, 2), (3, 4, 5)}
2   setListCopy = setList.copy()
3   print(setList)
4   print(setListCopy)
5   retVal = setList.discard((1, 2))
6   print(f"刪除之後{setList = }, 回傳值為 {retVal}")
7   retVal = setListCopy.discard((1, 4))
8   print(f"刪除之後{setListCopy = }, 回傳值為 {retVal}")
```

```
{(3, 4, 5), (1, 2)}
{(3, 4, 5), (1, 2)}
刪除之後setList = {(3, 4, 5)}, 回傳值為 None
刪除之後setListCopy = {(3, 4, 5), (1, 2)}, 回傳值為 None
```

  - 無論刪除成功或失敗，回傳值都是 None

10.3 適合集合的方法

- pop()
  - 用隨機的方式刪除集合元素，被刪除的元素將被傳回，如果集合是空集合，則會KeyError的程式異常。

retVal = set.pop()

```
1   setList = {(1, 2), (3, 4, 5), (6, 7)}
2   retVal = setList.pop()
3   print(f"pop之後{setList = }, 回傳值為 {retVal}")
4   retVal = setList.pop()
5   print(f"pop之後{setList = }, 回傳值為 {retVal}")
6   retVal = setList.pop()
7   print(f"pop之後{setList = }, 回傳值為 {retVal}")
8   retVal = setList.pop()
9   print(f"pop之後{setList = }, 回傳值為 {retVal}")
```

```
pop之後setList = {(1, 2), (3, 4, 5)}, 回傳值為 (6, 7)
pop之後setList = {(3, 4, 5)}, 回傳值為 (1, 2)
pop之後setList = set(), 回傳值為 (3, 4, 5)
Traceback (most recent call last):
  File "e:\PythonTest\test.py", line 8, in <module>
    retVal = setList.pop()
KeyError: 'pop from an empty set'
```

10.3 適合集合的方法

- clear()
  - 刪除集合內全部元素。

retVal = set.clear()

```
1   setList = {(1, 2), (3, 4, 5)}
2   retVal = setList.clear()
3   print(f"清除之後{setList = }, 回傳值為 {retVal}")
4   setEmpty = set()
5   retVal = setEmpty.clear()
6   print(f"清除之後{setEmpty = }, 回傳值為 {retVal}")
```

```
清除之後setList = set(), 回傳值為 None
清除之後setEmpty = set(), 回傳值為 None
```

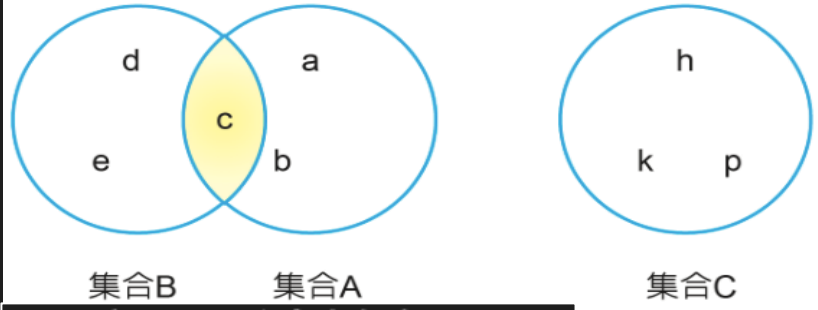
  - 回傳值永遠是 None

10.3 適合集合的方法

- isdisjoint()
  - 如果兩個集合沒有交集，傳回 True，否則傳回 False。

retBoolean = setA.isdisjoint(setB)

```
1   setA = {"a", "b", "c"}
2   setB = {"c", "d", "e"}
3   setC = {"h", "k", "p"}
4
5   retBoolean = setA.isdisjoint(setB)
6   print(f"setA 與 setB 沒有交集為: {retBoolean}")
7
8   retBoolean = setA.isdisjoint(setC)
9   print(f"setA 與 setC 沒有交集為: {retBoolean}")
```

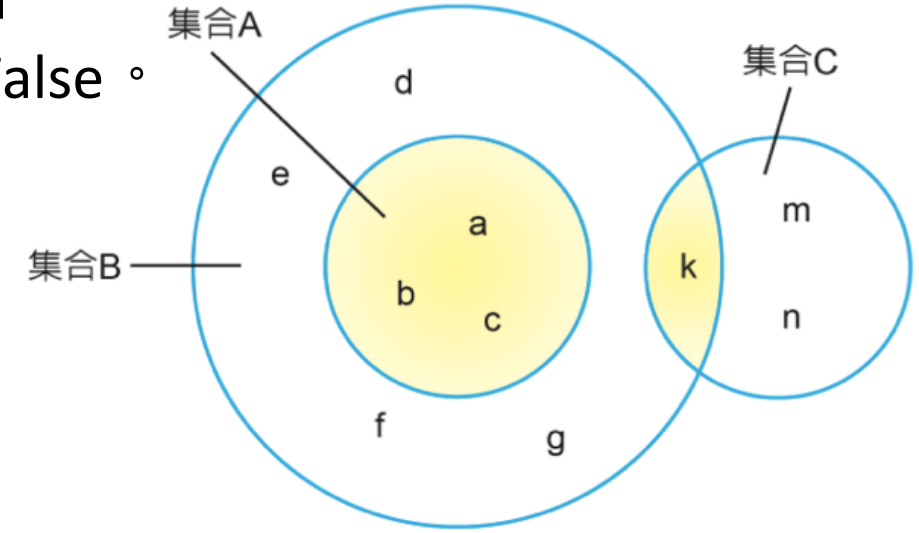



```
setA 與 setB 沒有交集為: False
setA 與 setC 沒有交集為: True
```

10.3 適合集合的方法

- issubset()
  - 測試一個集合是否是另一個集合的子集合，例如，A 集合所有元素均可在 B 集合內發現，則 A 集合是 B 集合的子集合。
  - A集合是B集合的子集合傳回 True，否則傳回 False。

retBoolean = setA.issubset(setB)




```
1   setA = {"a", "b", "c"}
2   setB = {"a", "b", "c", "d", "e", "f", "g", "k"}
3   setC = {"k", "m", "n"}
4
5   retBoolean = setA.issubset(setB)
6   print(f"setA 是 setB 的子集合為: {retBoolean}")
7
8   retBoolean = setC.issubset(setB)
9   print(f"setC 是 setB 的子集合為: {retBoolean}")
```

```
setA 是 setB 的子集合為: True
setc 是 setB 的子集合為: False
```

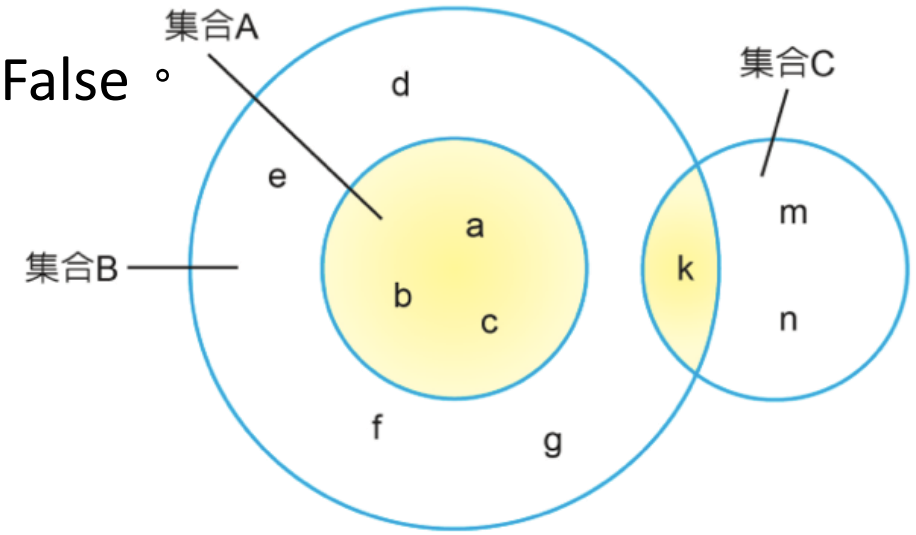
10.3 適合集合的方法

- issuperset()
  - 測試一個集合是否是另一個集合的父集合，例如，A 集合所有元素均可在 B 集合內發現，則 B 集合是 A 集合的父集合。
  - B集合是 A 集合的父集合傳回 True，否則傳回 False。

retBoolean = setB.issuperset(setA)

```
1   setA = {"a", "b", "c"}
2   setB = {"a", "b", "c", "d", "e", "f", "g", "k"}
3   setC = {"k", "m", "n"}
4
5   retBoolean = setB.issuperset(setA)
6   print(f"setB 是 setA 的父集合為: {retBoolean}")
7
8   retBoolean = setB.issuperset(setC)
9   print(f"setB 是 setC 的父集合為: {retBoolean}")
```

```
setB 是 setA 的父集合為: True
setB 是 setC 的父集合為: False
```



10.3 適合集合的方法

- intersection_update()
  - 將集合更新為與其他集合的交集。

retVal = setA.intersection_update(*set) #回傳值永遠是 None

  - 其中，*set 可以為單一集合，也可以為多個用逗號隔開的集合。

```
1   setA = {"a", "b", "c", "d"}
2   setB = {"a", "k", "c"}
3   setC = {"k", "f", "c"}
4
5   retBoolean = setA.intersection_update(setB)
6   print(f"{retBoolean = }")
7   print(f"{setA = }")
8   print(f"{setB = }")
9
10  retBoolean = setA.intersection_update(setB, setC)
11  print(f"{setA = }")
12  print(f"{setB = }")
13  print(f"{setC = }")
```

```
retBoolean = None
setA = {'a', 'c'}
setB = {'k', 'a', 'c'}
setA = {'c'}
setB = {'k', 'a', 'c'}
setC = {'k', 'f', 'c'}
```

10.3 適合集合的方法

- update()
  - 將集合更新為與其他集合的聯集。

retVal = setA. update(*set)　　#回傳值永遠是 None

  - 其中，*set 可以為單一集合，也可以為多個用逗號隔開的集合。

```
1   setCarsBrand1 = {"Infinity", "Toyota", "Lexus"}
2   setCarsBrand2 = {"Infinity", "Mazda", "Honda"}
3   setCarsBrand3 = {"Mazda", "Honda", "Nissan"}
4
5   retBoolean = setCarsBrand1.update(setCarsBrand2)
6   print(f"{setCarsBrand1 = }")
7   retBoolean = setCarsBrand1.update(setCarsBrand2, setCarsBrand3)
8   print(f"{setCarsBrand1 = }")
```

```
setCarsBrand1 = {'Mazda', 'Infinity', 'Lexus', 'Toyota', 'Honda'}
setCarsBrand1 = {'Mazda', 'Infinity', 'Lexus', 'Toyota', 'Nissan', 'Honda'}
```

10.3 適合集合的方法

- difference_update()
  - 將集合更新為與其他集合的差集。

retVal = setA.difference_ update(*set) #回傳值永遠是 None

  - 其中，*set 可以為單一集合，也可以為多個用逗號隔開的集合。

```
1   setCarsBrand1 = {"Infinity", "Mazda", "Honda", "Subaru"}
2   setCarsBrand2 = {"Infinity", "Toyota", "Lexus"}
3   setCarsBrand3 = {"Mazda", "Honda", "Nissan"}
4
5   retBoolean = setCarsBrand1.difference_update(setCarsBrand2)
6   print(f"{setCarsBrand1 = }")
7   retBoolean = setCarsBrand1.difference_update(setCarsBrand2, setCarsBrand3)
8   print(f"{setCarsBrand1 = }")
```

```
setCarsBrand1 = {'Subaru', 'Honda', 'Mazda'}
setCarsBrand1 = {'Subaru'}
```

10.3 適合集合的方法

- symmetric_difference_update()
  - 將集合更新為與其他集合的對稱差集。

retVal = setA. symmetric_difference_ update(setB)　　#回傳值永遠是 None

```
1   setCarsBrand1 = {"Infinity", "Mazda", "Honda", "Subaru"}
2   setCarsBrand2 = {"Infinity", "Toyota", "Lexus"}
3   setCarsBrand3 = {"Mazda", "Honda", "Nissan"}
4
5   retBoolean = setCarsBrand1.symmetric_difference_update(setCarsBrand2)
6   print(f"{setCarsBrand1 = }")
7   retBoolean = setCarsBrand1.symmetric_difference_update(setCarsBrand3)
8   print(f"{setCarsBrand1 = }")
```

```
setCarsBrand1 = {'Subaru', 'Toyota', 'Honda', 'Lexus', 'Mazda'}
setCarsBrand1 = {'Subaru', 'Nissan', 'Toyota', 'Lexus'}
```

10.4 適合集合的基本函數操作

函數名稱	說明
enumerate()	傳回連續整數配對的 enumerate 物件
len()	傳回元素數量
max()	傳回最大值
min()	傳回最小值
sorted()	傳回排序完的串列，集合本身不變。
sum()	傳回總和

- 用法與串列與元組相同，不再解說。

10.5 凍結集合 frozenset

- set是可變集合，frozenset是不可變集合也可直譯為凍結集合，這是一個新的類別(class)，只要設定元素後，這個凍結集合就不能再更改了。如果將元組(tuple)想成不可變串列(immutable list)，凍結集合就是不可變集合(immutable set)。
- 集合內元素是不可變的(immutable)，所以集合內的元素不能是集合，但是可以是凍結集合。
- frozenset 主要用作字典中的 key 或是集合中的元素。
- frozenset 不可使用 add() 或 remove() 更動內容，但可執行前面介紹 set 的其他所有方法。

10.5 凍結集合 frozenset

```
X = frozenset([1, 3, 5])
Y = frozenset([5, 7, 9])
print(f"{X = }")
print(f"{Y = }")
print(f"交集 = {X & Y}")
print(f"聯集 = {X.union(Y)}")
```

```
X = frozenset({1, 3, 5})
Y = frozenset({9, 5, 7})
交集 = frozenset({5})
聯集 = frozenset({1, 3, 5, 7, 9})
```

```
fsetX = frozenset([1, 3, 5])

setY = {"a", "b", fsetX}
print(f"{setY = }")

setZ = {fsetX, setY}
print(f"{setZ = }")
```

```
setY = {'a', frozenset({1, 3, 5}), 'b'}
Traceback (most recent call last):
  File "e:\PythonTest\test.py", line 6, in <module>
    setZ = {fsetX, setY}
TypeError: unhashable type: 'set'
```

```
fsetX = frozenset([1, 3, 5])
setVal = {"a", "b", "c"}
listVal =["ab", "cd", 100]

dictTest1 = dict(zip(fsetX, setVal))
print(f"{dictTest1 = }")

dictTest2 = dict(zip(setVal, listVal))
print(f"{dictTest2 = }")
```

```
dictTest1 = {1: 'c', 3: 'a', 5: 'b'}
dictTest2 = {'c': 'ab', 'a': 'cd', 'b': 100}
```

10.6 生成式 (generator)

- 集合的生成式
  - 新集合 = (運算式 for 運算式 in 可迭代項目 (if 條件))

```
1   setA = {n for n in range(1, 100, 2)}
2   print(f"{setA = }")
3
4   gen = (n for n in range(1, 100, 2))
5   setB = set(gen)
6   print(f"{setB = }")
```

```
setA = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71,
 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99}
setB = {1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71,
 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99}
```

```
1   setA = {n for n in range(1, 100, 2) if n % 11 == 0}
2   print(f"{setA = }")
3
4   gen = (n for n in range(1, 100, 2) if n % 11 == 0)
5   setB = set(gen)
6   print(f"{setB = }")
```

```
setA = {33, 99, 11, 77, 55}
setB = {33, 99, 11, 77, 55}
```

10.7 動手練習

- 雞尾酒的實例
  - 藍色夏威夷佬(Blue Hawaiian)：蘭姆酒(rum)、甜酒(sweet wine)、椰奶(coconut cream)、鳳梨汁(pineapple juice)、檸檬汁(lemon juice)。
  - 薑味莫西多(Ginger Mojito)：蘭姆酒(rum)、薑(ginger)、薄荷葉(mint leaves)、萊姆汁(lime juice)、薑汁汽水(ginger soda)。
  - 紐約客(New Yorker)：威士忌(whiskey)、紅酒(red wine)、檸檬汁(lemon juice)、糖水(sugar syrup)。
  - 血腥瑪莉(Bloody Mary)：伏特加(vodka)、檸檬汁(lemon juice)、番茄汁(tomato juice)、酸辣醬(tabasco)、少量鹽(little salt)。

- 為上述雞尾酒建立字典，用雞尾酒的名稱當 key，材料配方為 value。
- 列出含有伏特加的酒、含有檸檬汁的酒、含有蘭姆酒但沒有薑的酒。