

---

---

# Python Set

臺北科技大學資訊工程系

---

# Set 基本操作

- ❑ 集合型別的字面常數使用 **大括弧** { }
- ❑ 屬於**複合**資料型別 (compound data type) ，包含多個元素。
- ❑ 無序、元素**不重複**
- ❑ 使用 in 測試元素**是否在集合中**
- ❑ 使用 is 測試兩集合是否同一個

```
01 numX = {1,1,1,2,2,3,4,5}
02 numY = {5, 4, 2, 1, 3}
03 print(numX)
04 print(numY)
05 print(4 in numY)
06 print(numX == numY)
07 print(numX is numY)
08 print(id(numX))
09 print(id(numY))
```

```
{1, 2, 3, 4, 5}
{1, 2, 3, 4, 5}
True
True
False
2042423848296
2042423847400
```

# Set 基本操作

- ❑ `mySet = set()`：使用 `set()` 造出空集合物件
- ❑ `mySet = {}` 建立字典，非空集合
- ❑ `.add()`：加入一個元素 (`.update()`：加入多個元素)

```
01 def setOp():
02     Language = set()
03     print(Language)
04     mySet = {} #字典
05     print(type(mySet))
06     Language = {"asm", "C", "C++", "Java", "iOS", "Ruby", "perl", "delphi"}
07     Language.add("python")
08     print(Language)
```

`set()`

`<class 'dict' >`

`{'asm', 'python', 'C++', 'perl', 'Java', 'iOS', 'C', 'Ruby', 'delphi'}`

# Set 基本操作

- ❑ `in` 測試元素是否在集合中
  - ❑ `&` 交集運算，
  - ❑ `|` 聯集運算，
  - ❑ `-` 差集運算
  - ❑ `^` 排除相同元素(XOR)運算
  - ❑ `>` 測試左集合是否為右集合的父集
  - ❑ `<` 測試左集合是否為右集合的子集
- ❑ `nums = [1,1,1,2,2,3,4,5]`，若需將重複元素去除怎麼辦

```
nums = [1,1,1,2,2,3,4,5]  
num = set(nums)  
print(num)
```

```
{1, 2, 3, 4, 5}
```

# Set 基本操作

## □ & | - ^ > < 等元素性質

```
01 def setOp():
02     admins = set()
03     users = {'Smile', 'Tony', 'Happy', 'Sherry', 'Allen', 'Andy', 'Mars'}
04     admins.add('ihc')
05     admins.add('Mars')
06
07     print (admins)           {'ihc', 'Mars'}
08     print (users)           {'Sherry', 'Happy', 'Mars', 'Smile', 'Allen', 'Tony', 'Andy'}
09
10     print (admins & users)    {'Mars'}
11     print (admins | users)    {'Mars', 'Happy', 'Smile', 'ihc', 'Tony', 'Andy', 'Sherry', 'Allen'}
12     print (admins ^ users)    {'Happy', 'ihc', 'Tony', 'Smile', 'Andy', 'Sherry', 'Allen'}
13     print (admins - users)    {'ihc'}
14     print (users - admins)    {'Happy', 'Smile', 'Tony', 'Andy', 'Sherry', 'Allen'}
```

# Set 基本操作

01	<code>setx = set(["apple", "mango"])</code>		
02	<code>sety = set(["mango", "orange"])</code>		
03	<code>setz = set(["mango"])</code>		
04			
05	<code>my_issubset = setx &lt;= sety</code>		
06	<code>print(my_issubset)</code>	False	
07	<code>my_issuperset = setx &gt;= sety</code>		
08	<code>print(my_issuperset)</code>	False	
09	<code>my_issubset = setz &lt;= sety</code>		
10	<code>print(my_issubset)</code>	True	
11	<code>my_issuperset = sety &gt;= setz</code>		
12	<code>print(my_issuperset)</code>	True	
13			
14			
15	<code>zz = setx.issubset(sety)</code>		
16	<code>print(zz)</code>	False	
17	<code>yy = setx.isdisjoint(sety)</code>		
18	<code>print(yy)</code>	False	

# Set 函式

Method	Description
<code>add()</code>	<b>Adds</b> an element to the set
<code>clear()</code>	<b>Removes</b> all the elements from the set
<code>copy()</code>	Returns a <b>copy of the set</b>
<code>x.difference(y)</code>	Returns <b>x 和 y 不同的元素</b> (移除 x 中 y 有的)之 set
<code>x.difference_update(y)</code>	從 x 中移除 <b>x 和 y 共有的 items</b>
<code>discard()</code>	Remove the specified item ( <b>won't raise error if item doesn't exist</b> )
<code>intersection()</code>	Returns a set, that is the <b>intersection 交集</b> of two other sets
<code>intersection_update()</code>	Removes the items in this set that <b>are not present in other specified set(s)</b>
<code>isdisjoint()</code>	Returns whether two sets have a intersection or not
<code>issubset()</code>	Returns whether another set contains this set or not
<code>issuperset()</code>	Returns whether this set contains another set or not
<code>pop()</code>	<b>Removes an element</b> from the set
<code>remove()</code>	Removes the specified element ( <b>raise error if item doesn't exist</b> )
<code>x.symmetric_difference(y)</code>	Returns (x 中 y 沒有的) + (y 中 x 沒有的)
<code>x.symmetric_difference_update(y)</code>	inserts (x 中 y 沒有的) + (y 中 x 沒有的)
<code>union()</code>	Return a set containing the <b>union</b> of sets
<code>update()</code>	Update the set with the <b>union</b> of this set and others

# Set 函式

01	w = "kiwi"	
02	x = {"apple", "banana", "cherry"};	
03	y = {"orange", "mango"}	
04	z = "banana";	
05		
06	y.update(x)	
07	print(y)	{'orange', 'mango', 'cherry', 'apple', 'banana'}
08	print(len(y))	5
09	y.remove(z)	
10	print(y)	{'orange', 'mango', 'cherry', 'apple'}
11	#my_set.remove("kiwi") #產生錯誤-無此元素	
12	y.discard(w) #不會產生錯誤-無此元素	
13	print(y)	{'orange', 'mango', 'cherry', 'apple'}
14	a = {"apple", "banana", "cherry"};	
15	b = {"apple", "banana", "watermelon"}	
16	a.difference_update(b)	
17	print(a)	{'cherry'}



# Set 函式

01	setx = set(["apple", "mango"])	
02	sety = set(["mango", "orange"])	
03	setw = setx.difference(sety)	
04	print(setw)	{'apple'}
05	print(setx)	{'mango', 'apple'}
06	setx.difference_update(sety)	
07	print(setx)	{'apple'}
01	setx = set(["apple", "mango"])	
02	sety = set(["mango", "orange"])	
03	setw = setx.intersection(sety)	
04	print(setw)	{'mango'}
05	print(setx)	{'mango', 'apple'}
06	setx.intersection_update(sety)	
07	print(setx)	{'mango'}
01	setx = set(["apple", "mango"])	
02	sety = set(["mango", "orange"])	
03	setw = setx.symmetric_difference(sety)	
04	print(setw)	{'orange', 'apple'}
05	print(setx)	{'mango', 'apple'}
06	setx.symmetric_difference_update(sety)	
07	print(setx)	{'orange', 'apple'}

# Exercise

- 輸入兩個字串，輸出兩個字串有出現過的字母

```
01 s1=input("Enter first string: ")
02 s2=input("Enter second string: ")
03 a=list(set(s1) | set(s2))
04 print("The letters are:")
05 for i in a:
06     print(i)
```

輸入字串1='banana'  
輸入字串2='orange'  
寫下輸出內容

The letters are:

o  
n  
a  
g  
e  
r  
b

- 求出集合中最大與最小值

```
01 seta = set([5, 10, 3, 15, 2, 20])
02 #Find maximum value
03 print(max(seta))
04 #Find minimum value
05 print(min(seta))
```

20

2

# Exercise

- ❑ 使用 **集合**，輸入一個 **字串**，
- ❑ 若 **字串** 都是 沒有重複字母，回傳 **True**，有重複，回傳 **False**.

## sample input

just => True (has unique characters)  
Alexander => False (has duplicates)

輸入 = 'alexander'  
寫下執行編號順序  
寫下輸出內容

```
01 def is_unique(given_string):
02     #creating an empty set
03     chars_set = set()
04     for char in given_string:
05         #if char already in set
06         #it is duplicate
07         if char in chars_set:
08             return False
09     else:
10         #char not in set, add it to chars_set
11         chars_set.add(char)
12     #if no duplicates
13     return True
14 print(is_unique("just"))
```

True

# Exercise

- 使用集合寫程式，
  - 顯示字母沒有出現在第二個字串，
  - 但出現在第一個字串。
  
- 使用集合寫程式，
  - 輸入兩個字串，
  - 輸出均有的內容。

# Exercise

- 輸入兩個集合，檢查是否一個集合是另一個集合的子集，若是則刪除所有子集的元素

```
firstSet = {27, 43, 34}  
secondSet = {34, 93, 22, 27, 43, 53, 48}
```

- 輸出

```
First set is subset of Second set - True  
Second set is subset of First set - False  
After deleting the First set, the Second set is {93, 22 , 53, 48}
```

# Exercise

## □ 理想大學環境

○ 每一大學可用下列七種屬性表示：

- BC (Big Campus) : 代表有大校園。
- NC (Next to City) : 代表鄰近有大城市。
- CT (Convenient Transportation) : 代表交通方便。
- NS (Next to Sea) : 代表靠海。
- NM (Next to Mountain) : 代表依山。
- HL (Has Lake) : 代表校園有湖。
- NL (Near Landscape) : 代表附近有風景區。

# Exercise

## □ 理想大學環境

### ○ 大學簡稱

- |         |        |
|---------|--------|
| ➤ NSYSU | 國立中山大學 |
| ➤ NTU   | 國立臺灣大學 |
| ➤ NCCU  | 國立政治大學 |
| ➤ PPU   | 私立靜宜大學 |
| ➤ NTHU  | 國立清華大學 |

# Exercise

## □ 理想大學環境

Sample Input	Sample Output
5 NSYSU NC CT NS NM NTU BC NC CT NS NCCU BC NL HL PPU BC NC NTHU BC NS	NTU NTHU NSYSU NCCU NTU PPU
2 BC NS + CT HL NM + BC NL + BC NC	

### ○ 輸入說明：

- 1. 輸入理想大學條件，用 + 號區格條件是“或”的關係，沒有 + 區隔是“且”的關係，屬性間和 + 間以空白間隔。
- 例如：BC NS + CT HL 是找有 **大校園** 且 **靠海**，或 **交通方便** 且 **校園有湖** 的所有大學。
- 2. 第一行一正整數，代表 **大學個數 n** ( $n \leq 10$ )。
- 3. 其後 n 行，每一行為大學名稱，接著大學 **具備屬性**。大學名稱最多 10 個字母，屬性 2 個字母，均為英文字母，大學名稱及屬性間以一個空白分隔。
- 4. 接下來一行 **正整數 m**，為查詢個數， $m \leq 10$ 。
- 5. 其後 m 行，每一行有一個 查詢。條件為 **校園屬性**。



# Exercise

## □ 理想大學環境

○ 輸出說明：(1, 2 輸出各得 1 / 2 分數)

- 1. m 行，第 i 列印出第 i 個查詢中，所有符合之**大學名稱**。
  - (1) 若有多個大學符合一個查詢，大學間以空白分隔。
  - (2) 每行查詢輸出順序，根據先後**查詢條件符合**的大學順序。
- 2. 如果都沒有**完全符合**，則輸出**最多符合**的大學。

Sample Input	Sample Output
5 NSYSU NC CT NS NM NTU BC NC CT NS NCCU BC NL HL PPU BC NC NTHU BC NS 2 BC NS + CT HL NM + BC NL + BC NC	NTU NTHU NSYSU NCCU NTU PPU

# Exercise

Sample Input	Sample Output
<b>5</b> <b>NSYSU NC CT NS NM</b> <b>NTU BC NC CT NS</b> <b>NCCU BC NL HL</b> <b>PPU BC NC</b> <b>NTHU BC NS</b> <b>2</b> <b>BC NS + CT HL</b> <b>NM + BC NL + BC NC</b>	NTU NTHU NSYSU NCCU NTU PPU
<b>5</b> <b>NSYSU NC CT NS NM</b> <b>NTU BC NC CT NS</b> <b>NCCU BC NL HL</b> <b>PPU BC NC</b> <b>NTHU BC NS</b> <b>1</b> <b>BC NS NL + CT HL</b>	NTU NTHU

# Exercise

```
def getData():
    university = {}
    n = int(input())
    for i in range(n):
        item = input().split()
        university[item[0]] = set(item[1:])
    return university

def match(con, feature):
    conds = con.split(' + ')
    maxNum = 0
    for i in range(len(conds)):
        if feature.issuperset(set(conds[i].split())):
            return -1
        k = len(feature & set(conds[i].split()))
        if k > maxNum: maxNum = k
    return maxNum
```

```
def compute(con, university):
    data = {}
    for name, feature in university.items():
        data[name] = match(con, feature)
    if data[name] == -1:
        print(name, end=' ')
    if -1 not in data.values():
        value = max(data.values())
        for key, v in data.items():
            if v == value: print(key, end=' ')
    print()

def main():
    university = getData()
    n = int(input())
    for i in range(n):
        con = input()
        compute(con, university)
```

---

# END

---

