
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測試兩集和是否一樣

```
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 = {}
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

- ❑ 使用in測試元素是否在集合中
 - ❑ $\&$ 交集運算， $|$ 聯集運算， $-$ 差集運算
 - ❑ \wedge 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     print (admins & users)
07     print (admins | users)
08     print (admins ^ users)
09     print (admins - users)
10     print (users - admins)
```

```
{'Mars'}
{'Mars', 'Happy', 'Smile', 'ihc', 'Tony', 'Andy', 'Sherry', 'Allen'}
{'Happy', 'ihc', 'Tony', 'Smile', 'Andy', 'Sherry', 'Allen'}
{'ihc'}
{'Happy', 'Smile', 'Tony', 'Andy', 'Sherry', 'Allen'}
```

set基本操作

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

```
False
False
True
True

False
False
```

Set 函 式

| Method | Description |
|---|--|
| <code>add()</code> | Adds an element to the set |
| <code>clear()</code> | Removes all the elements from the set |
| <code>copy()</code> | Returns a copy of the set |
| <code>x.difference(y)</code> | Returns x 和 y 不同的元素的 set |
| <code>x.difference_update(y)</code> | 從 x 中移除 x 和 y 共有的 items |
| <code>discard()</code> | Remove the specified item (won't raise error if item doesn't exist) |
| <code>intersection()</code> | Returns a set, that is the intersection of two other sets |
| <code>intersection_update()</code> | Removes the items in this set that are not present in other specified set(s) |
| <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> | Removes an element from the set |
| <code>remove()</code> | Removes the specified element (raise error if item doesn't exist) |
| <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 union of sets |
| <code>update()</code> | Update the set with the union of this set and others |

Set函式

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

```
{'orange', 'mango', 'cherry', 'apple', 'banana'}
5
{'orange', 'mango', 'cherry', 'apple'}
{'orange', 'mango', 'cherry', 'apple'}
{'cherry'}
```


Set函数

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

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

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

Exercise

- 輸入兩個字串，輸出兩個字串均有的部分

```
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

- 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))
```

Exercise

- 使用集合，輸入一個字串，若字串都是單一長度，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"))
```

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。
- ❑ 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。

○ 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。

Exercise

- 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。
- 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。

Exercise

| Sample Input | Sample Output |
|---|---------------------------------------|
| 5 NSYSU NC CT NS NM NTU BC NC CT NS NCCU BC NL HL Providence BC NC NTHU BC NS 2 BC NS + CT HL NM + BC NL + BC NC | NTU NTHU NSYSU NCCU NTU Providence |
| 5 NSYSU NC CT NS NM NTU BC NC CT NS NCCU BC NL HL Providence BC NC NTHU BC NS 1 BC NS NL + CT HL | 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

