**Python程式設計：集合**

**目錄**

[1. 基本概念 1](#_Toc206962152)

[2. 建立和定義集合 1](#_Toc206962153)

[3. 集合的常用方法 2](#_Toc206962154)

[3.1 add()：新增元素 2](#_Toc206962155)

[3.2 remove()：刪除元素 2](#_Toc206962156)

[3.3 pop()：隨機刪除元素 2](#_Toc206962157)

[3.4 update()：合併多個集合 2](#_Toc206962158)

[4. 集合的數學運算 3](#_Toc206962159)

[4.1 union() 或 |：聯集 3](#_Toc206962160)

[4.2 intersection() 或 &：交集 3](#_Toc206962161)

[4.3 difference() 或 -：差集 3](#_Toc206962162)

[5. 練習 4](#_Toc206962163)

[5.1 Problem: 集合的基礎運算 4](#_Toc206962164)

[5.2 Problem: 夏令營報名統計 5](#_Toc206962165)

# 基本概念

集合是一種無序的、不重複的元素集合。想像一下，它就像一個數學上的集合，每個元素都是唯一的，且沒有固定的順序。

* 無序：集合中的元素沒有索引，因此你不能用 set[0] 這種方式來存取元素。
* 不重複：集合會自動過濾掉重複的元素。如果你試圖新增一個已經存在的元素，它會被忽略。
* 可變（mutable）：集合本身可以新增或刪除元素。
* 不可變的元素：集合中的元素必須是不可變的資料型態，例如：數字、字串、元組（tuple）等。因此，列表（list）和字典（dict）不能作為集合的元素。

# 建立和定義集合

你可以用 {} 或 set() 函數來建立集合。

# 1. 直接用大括號建立集合，元素重複時會自動移除

A = {'Python', 'Java', 'C', 'Python'}

print(A)

# 2. 用 set() 函數將列表轉換為集合，同樣會移除重複元素

my\_list = [1, 2, 3, 4, 3, 2, 1]

B = set(my\_list)

print(B)

# 3. 建立一個空集合，必須使用 set()

# 如果用 {} 建立，會變成一個空字典！

empty\_set = set()

empty\_dict = {}

print(f"empty\_set 的型態是: {type(empty\_set)}")

print(f"empty\_dict 的型態是: {type(empty\_dict)}")

# 集合的常用方法

## add()：新增元素

add() 方法用於向集合中新增一個元素。

cities = {'Taipei', 'Beijing'}

cities.add('Tokyo')

print(cities)

## remove()：刪除元素

remove() 方法用於刪除集合中指定的元素。如果元素不存在，會引發 KeyError 錯誤。

animals = {'dog', 'cat', 'bird'}

animals.remove('cat')

print(animals)

# 如果嘗試刪除不存在的元素，程式會報錯

try:

    animals.remove('fish')

except KeyError:

    print("無法刪除 'fish'，因為它不存在。")

## pop()：隨機刪除元素

pop() 方法會隨機刪除並返回集合中的一個元素。由於集合是無序的，你無法預測會刪除哪一個。

animals = {'dog', 'cat', 'bird'}

ret\_element = animals.pop()

print(f"被移除的元素是: {ret\_element}")

print(f"移除後的集合: {animals}")

## update()：合併多個集合

update() 方法可以將一個或多個集合的元素新增到當前集合中。

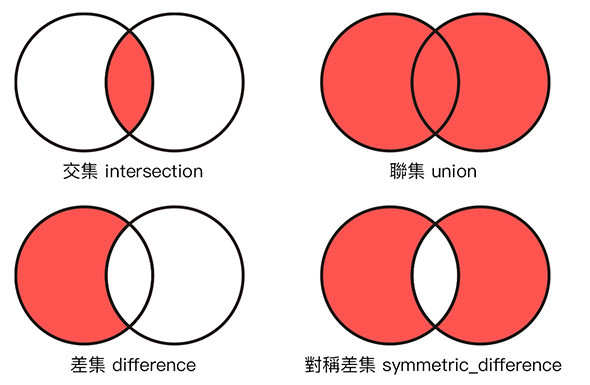
cars1 = {'Nissan', 'Toyota'}

cars2 = {'Audi', 'Ford'}

cars1.update(cars2)

print(cars1)

# 集合的數學運算



## union() 或 |：聯集

聯集會返回包含兩個集合中所有元素的新集合。

math = {'Kevin', 'Peter'}

physics = {'Eric', 'Tim'}

union\_set = math.union(physics)

print(f"聯集: {union\_set}")

# 你也可以使用 | 運算符

union\_set\_op = math | physics

print(f"聯集 (運算符): {union\_set\_op}")

## intersection() 或 &：交集

交集會返回包含兩個集合中共同元素的新集合。

math = {'Kevin', 'Peter', 'Eric'}

physics = {'Kevin', 'Eric', 'Tim'}

intersection\_set = math.intersection(physics)

print(f"交集: {intersection\_set}")

# 你也可以使用 & 運算符

intersection\_set\_op = math & physics

print(f"交集 (運算符): {intersection\_set\_op}")

## difference() 或 -：差集

差集會返回在第一個集合中但不在第二個集合中的元素。

math = {'Kevin', 'Peter', 'Eric'}

physics = {'Kevin', 'Eric', 'Tim'}

difference\_set = math.difference(physics)

print(f"差集 (math - physics): {difference\_set}")

# 你也可以使用 - 運算符

difference\_set\_op = math - physics

print(f"差集 (運算符): {difference\_set\_op}")

# 練習

## Problem: 集合的基礎運算

|  |  |
| --- | --- |
| Problem Description:  給定兩個集合 math 和 physics，分別代表參加數學夏令營和物理夏令營的學生名單。請計算並印出：   * 兩個夏令營都參加的學生名單（交集）。 * 只參加數學夏令營的學生名單（差集）。 | |
| Input: | Output: |
| Sample Input:  math = {'Kevin', 'Peter', 'Eric'}  physics = {'Kevin', 'Eric', 'Tim'} | Sample Output:  都參加的學生名單: {'Kevin', 'Eric'} 只參加數學夏令營的學生名單: {'Peter'} |
| Sample Input:  math = {'A', 'B', 'C'}  physics = {'B', 'D', 'E'} | Sample Output:  都參加的學生名單: {'B'}  只參加數學夏令營的學生名單: {'A', 'C'} |
| Answer: | |

## Problem: 夏令營報名統計

|  |  |
| --- | --- |
| Problem Description:  某班級有 10 個學生，其中 3 個參加了數學夏令營，另外 3 個參加了物理夏令營。請用集合來管理這些名單，並回答以下問題：   * 參加數學或物理夏令營的學生總數。 * 兩者都沒參加的學生名單。 | |
| Input: | Output: |
| Sample Input:  students = {'Peter', 'Norton', 'Kevin', 'Mary', 'John', 'Ford', 'Nelson', 'Damon', 'Ivan', 'Tom'}  math = {'Peter', 'Kevin', 'Damon'}  physics = {'Nelson', 'Damon', 'Tom'} | Sample Output:  有 5 人參加數學或物理夏令營 沒有參加任何夏令營的有 5 人: {'Mary', 'Ford', 'John', 'Norton', 'Ivan'} |
| Sample Input:  students = {'A', 'B', 'C', 'D', 'E', 'F'}  math = {'A', 'B'}  physics = {'C', 'D'} | Sample Output:  有 4 人參加數學或物理夏令營 沒有參加任何夏令營的有 2 人: {'E', 'F'} |
| Answer: | |