

D02 NumPy 陣列進階操作

[簡報閱讀](#)[範例與作業](#)[問題討論](#)[學習心得\(完成\)](#)

重要知識點

NumPy 陣列重塑 -
flatten() 與 ravel()

NumPy 陣列重塑 -
reshape()

NumPy 陣列重塑 -
resize()

軸 (axis) 與維度
(dimension)

NumPy 陣列的合併 -
concatenate()

重要知識點



- 介紹陣列重塑
- 介紹軸 (axis) 與維度 (dimension)
- 介紹陣列的合併與分割
- 介紹陣列的迭代
- 介紹陣列的搜尋與排序

- 透過 `flatten()` 與 `ravel()` 均可將多維陣列轉形為一維陣列，`flatten()` 與 `ravel()` 的使用透過下列兩種方法，得到的結果都是完全一樣的。
- 不同的是，`ravel()` 建立的是原來陣列的 `view`，所以在 `ravel()` 回傳物件中做的元素值變更，將會影響原陣列的元素值。
- 預設展開的順序是 C-style，展開時是以 row 為主的順序展開。

NumPy 陣列重塑 - `reshape()`

- 呼叫 `reshape()` 時指定新的形狀 (`shape`)，可將陣列重塑為該形狀，但如果新的總數與原先 `shape` 總數不一致的話，則會產生錯誤。
- Reshape 時，新的形狀可以採用模糊指定為 `-1`，讓 NumPy 自動計算，例如：
`a.reshape((5, -1))`。
- 若 reshape 後的陣列元素值改變，會影響原陣列對應的元素值也跟著改變。

NumPy 陣列重塑 - `resize()`

- 與 `reshape()` 不同的地方在於，如果 `resize` 的大小超過總元素值，則會在後面的元素值的指定為 0。
- 如果 `resize` 的大小小於總元素值，則會依照 C-style 的順序，取得 `resize` 後的陣列元素。

- 軸 (axis) 在 NumPy 多維陣列中是很重要觀念，但是在應用上容易混淆。軸的數目也就是 NumPy 陣列的維度 (dimension) 數，軸的順序編號從 0 開始。
- 如果是要增加軸數的話，可以使用 `np.newaxis` 物件。將 `np.newaxis` 加到要增加的軸的位置即可。
- 一維、二維、三維陣列的軸如下圖示範：

NumPy 陣列的合併 - `concatenate()`

使用 `concatenate()` 進行陣列的合併時，須留意除了指定的軸之外 (預設為 `axis 0`)，其他軸的形狀必須完全相同，合併才不會發生錯誤。

NumPy 陣列的合併 - `stack()`, `hstack()`, `vstack()`

- `stack()`, `hstack()`, `vstack()` 的觀念及用法類似，不同點在於 `stack()` 回傳的陣列維度會是合併前的維度 +1，而 `hstack()` 與 `vstack()` 回傳的陣列維度則是依合併的陣列而定。
- 至於是否可以合併，`stack()` 必須要所有陣列的形狀都一樣；而 `hstack()` 與 `vstack()` 則跟

NumPy 陣列的分割 – `split()`、`hsplit()`、`vsplit()`

- 呼叫 `split()` 時 `indices_or_sections` 引數如果給定單一整數的話，那就會按照軸把陣列等分；如果給定一個 List 的整數值的話，就會按照區段去分割。
- `hsplit()` 與 `vsplit()`，分別是依照水平軸和垂直軸去做分割。

NumPy 陣列迭代

- 一維陣列的迭代，跟 Python 集合型別 (例如 List) 的迭代相同。
- 多維陣列的迭代則以 `axis 0` 為準。

```
1 for row in b:  
2     print(row)
```

合 flat 屬性。

```
1 for i in b.flat:  
2     print(i)
```

NumPy 陣列搜尋與排序 - `amax()`、`amin()`、`max()`、`min()`

- 顯示陣列元素最大值和最小值，可以透過 `amax()`、`amin()`、`max()`、`min()`，也可以依照軸列出各軸的最大/最小元素值。
- 如果是多維陣列的話，用法也是相同，也可以依照軸列出最大或最小值。
- 有 2 種不同的使用方式：

NumPy 陣列搜尋與排序 – `argmax()` 與 `argmin()`

最小值的索引，也可以依照軸找出各軸最大值和最小值的索引。

- 有 2 種不同的使用方式：

NumPy 陣列搜尋與排序 – where()

- 傳入條件式，回傳值為符合條件的元素索引。
 - 若是多維陣列的話，會回傳多個陣列的索引值，要合在一起看。以二維陣列為例：
-
- 上面的回傳值代表 `a[0, 0]`, `a[0, 1]`, `a[1, 3]`, `a[2, 2]` 均為符合條件的元素索引值。

NumPy 陣列搜尋與排序 – nonzero()

- `nonzero` 等同於 `np.where(array != 0)` 的語法，同樣的也是回傳符合非 0 條件的元素索引值。
- 有 2 種不同的使用方式：

NumPy 陣列搜尋與排序 – sort() 與 argsort()

排序後的陣列，而 `argsort()` 回傳的是排序後的陣列索引值。

- 有 2 種不同的使用方式：
- 與 `np.sort()` 不同的是，陣列物件 `.sort()` 的語法會進行 **in-place** 排序，也就是原本的陣列內容會跟著改變。
- 多維陣列在排序時可以指定要依據的軸。
- 排序支援多種不同的排序算法，包括 `quicksort` (預設)、`heapsort`、`mergesort`、`timesort`，在 `kind` 引數指定即可。依照官網文件指出排序速度是以 `quicksort` 最快，`mergesort` / `timesort` 其次，之後是 `heapsort`。

知識點回顧

- 軸 (axis) 與維度 (dimension) 是非常重要的觀念，在實際使用上常因為軸及維度、形狀等問題而造成程式錯誤。
- 在 NumPy 進階操作的部分，介紹了重塑、合併、分割、迭代、搜尋、排序等操作與相關的函式，在處理陣列時提供方便的工具可運用。

