numpy筆記

Numpy介紹

NumPy（Numerical Python）是 Python 中進行科學計算的基礎套件，它提供：

- 高效能的多維陣列（ndarray）物件
- 各種數學運算（如線性代數、傅立葉變換、隨機數生成）
- 廣泛的向量化操作與廣播功能（Broadcasting）

NumPy 的核心是 `ndarray`，它比 Python 原生的 list 在資料儲存與運算上更高效，適合用於大規模資料處理與矩陣操作。

---

ndarray

📎 numpy.ndarray

`ndarray` 是 NumPy 中的核心資料結構，用於儲存同質資料的多維陣列。

建立 ndarray

```
import numpy as np

# 一維陣列
arr1 = np.array([1, 2, 3])

# 二維陣列
arr2 = np.array([1, 2](1,%202))
```

屬性（Properties）

- `ndarray.ndim`：陣列的維度
- `ndarray.shape`：每個維度的大小（row, column, ...）
- `ndarray.size`：元素總數
- `ndarray.dtype`：陣列中元素的資料型別

```
arr = np.array([1, 2, 3](1,%202,%203))
print(arr.ndim)  # 2
print(arr.shape)  # (2, 3)
print(arr.size)  # 6
print(arr.dtype)  # int64 (依平臺不同略有不同)
```

索引與切片

```
print(arr[0, 1])   # 2
print(arr[:, 1])   # [2 5]
print(arr[1, :2])  # [4 5]
```

---

ndarray的特性

多維性

```
import numpy as np
```

```
arr = np.array(5)  # 建立0維的ndarray
print(arr)
print('arr的維度：', arr.ndim)
```

```
5
arr的維度： 0
```

```
arr = np.array([1, 2, 3])  # 建立1維ndarray
print(arr)
print('arr的維度：', arr.ndim)
```

```
[1 2 3]
arr的維度： 1
```

```
arr = np.array([1, 2, 3](1,%202,%203)) # 建立2維ndarray
print(arr)
print('arr的維度：', arr.ndim)
```

```
[[1 2 3]
 [4 5 6]]
arr的維度： 2
```

同質性

```
arr = np.array([1, 'hello']) # 會自動轉型為相同型別
print(arr)
```

```
['1' 'hello']
```

```
arr = np.array([1, 2.5]) # 混合整數與浮點數會轉為float
print(arr)
```

```
[1.  2.5]
```

---

ndarray的屬性

```
arr = np.array(1)
print(arr)
print('形狀：', arr.shape)
print('維度：', arr.ndim)
print('元素個數：', arr.size)
print('資料型別：', arr.dtype)
print('轉置：', arr.T)
```

```
1
陣列的形狀： ()
```

```
陣列的維度： 0
元素的個數： 1
元素的資料型別： int64
元素的轉置 1
```

```
arr = np.array([1, 2.5, 3])
print(arr)
print('形狀：', arr.shape)
print('維度：', arr.ndim)
print('元素個數：', arr.size)
print('資料型別：', arr.dtype)
print('轉置：', arr.T)
```

```
[1.  2.5 3. ]
陣列的形狀： (3,)
陣列的維度： 1
元素的個數： 3
元素的資料型別： float64
元素的轉置 [1.  2.5 3. ]
```

```
arr = np.array([1, 2, 3](1,%202,%203))
print(arr)
print('形狀：', arr.shape)
print('維度：', arr.ndim)
print('元素個數：', arr.size)
print('轉置：', arr.T)
```

```
[[1 2 3]
 [4 5 6]]
陣列的形狀： (2, 3)
陣列的維度： 2
元素的個數： 6
元素的轉置 [[1 4]
 [2 5]
 [3 6]]
```

ndarray的建立

基本方法

```
list1 = [4, 5, 6]
arr = np.array(list1, dtype=np.float64)
print(arr.ndim)
print(arr)
```

```
1
[4. 5. 6.]
```

複製

```
arr1 = np.copy(arr)
print(arr1)
arr1[0] = 8
print(arr1)
print(arr)  # 原始資料未變
```

```
[4. 5. 6.]
[8. 5. 6.]
[4. 5. 6.]
```

預定形狀陣列

```
arr = np.zeros((2, 3), dtype=int)
print(arr)
```

```
[[0 0 0]
 [0 0 0]]
int64
```

```
arr = np.ones((5, 8), dtype=int)
print(arr)
```

```
[[1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1]
 [1 1 1 1 1 1 1 1]]
```

```
arr = np.empty((2, 3))
print(arr)
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
```

```
arr = np.empty((4, 2))
print(arr)
```

```
[[4.29988378e-315 0.00000000e+000]
 [0.00000000e+000 0.00000000e+000]
 [0.00000000e+000 0.00000000e+000]
 [0.00000000e+000 0.00000000e+000]]
```

```
arr = np.full((3, 4), 2025)
print(arr)
```

```
[[2025 2025 2025 2025]
 [2025 2025 2025 2025]
 [2025 2025 2025 2025]]
```

依現有陣列建立

```
arr1 = np.zeros_like(arr)
arr1 = np.empty_like(arr)
```

```
arr1 = np.ones_like(arr)
arr1 = np.full_like(arr, 2026)
print(arr1)
```

```
[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]

[[872866261 0 0 0]
 [0 4051328937705681766 3846976120238454374
  7364855856089883187]

 [7234241398704530226 7233398073269040434
 3617345295167598694 7305230452898685490]]

[[1 1 1 1]
 [1 1 1 1]
 [1 1 1 1]]

[[2026 2026 2026 2026]
 [2026 2026 2026 2026]
 [2026 2026 2026 2026]]
```

數列建立

```
arr = np.arange(1, 51, 1)
print(arr)
```

```
 [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50]
```

```
arr = np.linspace(0, 100, 5)
print(arr)
```

```
[  0.  25.  50.  75. 100.]
```

```
arr = np.linspace(0, 100, 5, dtype=int)
print(arr)
arr = np.arange(0, 101, 25)
print(arr)
```

```
[  0  25  50  75 100]
[  0  25  50  75 100]
```

```
arr = np.logspace(0, 4, 3)
print(arr)
```

```
[1.e+00 1.e+02 1.e+04]
```

```
arr = np.logspace(0, 4, 3, base=2)
print(arr)
```

```
[ 1.  4. 16.]
```

特殊矩陣

```
arr = np.eye(3, 4, dtype=int)
print(arr)
```

```
[[1 0 0 0]
 [0 1 0 0]
 [0 0 1 0]]
```

```
arr = np.diag([5, 1, 2, 3])
print(arr)
```

```
[[5 0 0 0]
 [0 1 0 0]
```

```
[0 0 2 0]
[0 0 0 3]]
```

隨機陣列

```
arr = np.random.rand(2, 3)
print(arr)
```

```
[[0.67900845 0.23917037 0.00414335]
 [0.72518682 0.69888024 0.15564678]]
```

```
arr = np.random.uniform(3, 6, (2, 3))
print(arr)
```

```
[[3.16254705 4.05323667 3.38568092]
 [4.99436618 3.48279181 3.9247733 ]]
```

```
arr = np.random.randint(3, 30, (2, 3))
print(arr)
```

```
[[19 13 10]
 [11 27  8]]
```

```
# 生成隨機數列（正態分佈）（-3到3之間）
arr = np.random.randn(2, 3)
print(arr)
```

```
[[ 1.06154956  0.48960478 -1.00781474]
 [ 0.52528568  0.53723629 -2.07150278]]
```

```
# 設定隨機種子
np.random.seed(20)
```

```
arr = np.random.randint(1, 10, (2, 5))
print(arr)
```

```
[[4 5 7 8 3]
 [1 7 9 6 4]]
```

資料型別與轉換

```
arr = np.array([1, 2, 3], dtype='i8')
print(arr)
```

```
[1 2 3]
```

```
arr = np.array([1, 0, 127, 0], dtype=np.int8)
print(arr)
```

```
[  1   0 127   0]
```

索引與切片

一維

```
arr = np.random.randint(1, 100, 20)
print(arr)
print(arr[10])
print(arr[:])
print(arr[2:5])
print(arr[slice(2, 15, 3)])
print(arr[(arr > 10) & (arr < 70)])
```

```
[71 29 73 86  8 72 49 69 52 73 95 21  9 48 20 61  9  1 33 92]
95
[71 29 73 86  8 72 49 69 52 73 95 21  9 48 20 61  9  1 33 92]
[73 86  8]
```

```
[73 72 52 21 20]
[29 49 69 52 21 48 20 61 33]
```

二維

```
arr = np.random.randint(1, 100, 20) # 建立一個含有 20 個整數的 ndarray，範圍介於 1 到 99（不包含 100）
print(arr) # 印出整個陣列
print(arr[10]) # 印出第 11 個元素（index 為 10）
print(arr[:]) # 印出整個陣列（等同於 arr）
print(arr[2:5]) # 印出第 3 到第 5 個元素（index 2, 3, 4）
print(arr[slice(2, 15, 3)]) # 使用 slice 對象，每 3 步印出 index 從 2 到 14 的元素（含 2 不含 15）
print(arr[(arr > 10) & (arr < 70)]) # 印出大於 10 且小於 70 的元素（布林遮罩篩選）
print(arr[1, 3]) # 索引
print(arr[1, 2:5])
print(arr[2][arr[2] > 50])
print(arr[:, 3])
```

```
[71 29 73 86  8 72 49 69 52 73 95 21  9 48 20 61  9  1 33 92]
95
[71 29 73 86  8 72 49 69 52 73 95 21  9 48 20 61  9  1 33 92]
[73 86  8]
[73 72 52 21 20]
[29 49 69 52 21 48 20 61 33]
 91
[80 91 60]
[64 79 94]
[19 91 79  4]
```

運算

元素算術

```
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
print(a + b)
```

```
print(a - b)
print(a * b)
print(a / b)
print(a ** 2)
for i in range(len(a)):
   a[i] = a[i] + b[i]
print(a)
```

```
[5 7 9]
[-3 -3 -3]
[ 4 10 18]
[0.25 0.4  0.5 ]
[1 4 9]
[5, 7, 9]
```

與純量運算

```
a = np.array([1, 2, 3](1,%202,%203))
print(a + 3)
print(a * 3)
```

```
[[ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
[[ 3  6  9]
 [12 15 18]
 [21 24 27]]
```

💡廣播機制

廣播機制（Broadcasting）

NumPy 的廣播（Broadcasting）是一種強大的機制，允許不同形狀的 array 之間進行算術運算，而無需複製資料。

定義：

當 NumPy 在處理兩個形狀不同的 array 時，它會嘗試「廣播」較小的 array，使其在維度上與較大的 array 一致，以便能進行元素對應的運算。

廣播規則：

NumPy 在進行運算前會依據以下兩條規則來調整形狀：

1. 如果兩個 array 的 rank（維度數量）不同，則會在維度較少的前面補 1。
2. 兩個 array 在某一維的大小若不相等，則其中之一必須為 1，NumPy 才會將其『拉伸』成與另一個相同。

若這兩條規則都無法滿足，則會拋出 `ValueError`。

範例 1：

```
import numpy as np
A = np.array([1, 2, 3])         # shape (3,)
B = np.array([10](10))  # shape (3,1)
print(A + B)                    # shape (3,3)
```

說明：

- `A` 形狀是 (3,) → 視為 (1,3)
- `B` 形狀是 (3,1)
- 廣播後 → `A` 擴展成 (3,3)，`B` 也成 (3,3)

輸出：

```
[[11 12 13]
 [21 22 23]
 [31 32 33]]
```

範例 2：

```
X = np.array([1, 2, 3](1,%202,%203))  # shape (2,3)
Y = np.array([10, 20, 30])        # shape (3,)
print(X + Y)                      # shape (2,3)
```

說明：

- `Y` 廣播成 shape (2,3)，與 X 相加。

輸出：

```
[[11 22 33]
 [14 25 36]]
```

常見錯誤：

```
A = np.array([1](1))  # shape (3,1)
B = np.array([1, 2])        # shape (2,)
A + B  # ValueError
```

錯誤原因：

- `A` shape (3,1)，`B` shape (2,)
- 廣播後是 (3,2) vs (3,1)，不相容 → 拋出錯誤。

實用技巧：

- 使用 `np.newaxis` 或 `reshape()` 可以手動調整 shape 以符合廣播規則。

```
A = np.array([1, 2, 3])        # shape (3,)
B = np.array([10, 20, 30]).reshape((3,1))  # shape (3,1)
print(A + B)                # 成功廣播，shape (3,3)
```

矩陣乘法

```
a = np.array([1, 2, 3](1,%202,%203))
b = np.array([4, 5, 6](4,%205,%206))
print(a @ b)
```

```
[[ 21  27  33]
 [ 57  72  87]
 [ 93 117 141]]
```

numpy中的常用函式

基本數學函式

```
print(np.sqrt([1, 4, 9])) # 計算每個元素的平方根
print(np.exp(1)) # 計算 e 的一次方（自然常數 e 約等於 2.718）
print(np.log(2.71)) # 計算自然對數（以 e 為底）
print(np.sin(np.pi / 2)) # 計算正弦值，π/2 的正弦為 1
print(np.cos(np.pi)) # 計算餘弦值，π 的餘弦為 -1

arr = np.array([-1, 1, 2, -3])
print(np.abs(arr)) # 計算絕對值（忽略負號）
print(np.power(arr, 3)) # 每個元素取三次方

print(np.round([3.2, 4.5, 8.1, 9.6])) # 四捨五入到最接近的整數
arr = np.array([1.6, 25.1, 81.7])
print(np.ceil(arr)) # 將數值無條件進位至最小整數
print(np.floor(arr)) # 將數值無條件捨去至最大整數
print(np.isnan([1, 2, np.nan, 3])) # 判斷哪些元素是 NaN（不是數值）
```

```
[1. 2. 3.]
2.718281828459045
0.9969486348916096
1.0
-1.0
[1 1 2 3]
[ -1   1   8 -27]
[ 3.  4.  8. 10.]
[ 2. 26. 82.]
[ 1. 25. 81.]
[False False  True False]
```

統計函式

```
# 計算標準差、方差
# 1,2,3 的平均值 2
# ((1-2)^2 + (2-2)^2 + (3-2)^2 )/ 3 = 0.666
print(np.sqrt([1, 4, 9])) # 計算每個元素的平方根
print(np.exp(1)) # 計算 e 的一次方（自然常數 e 約等於 2.718）
```

```
print(np.log(2.71)) # 計算自然對數（以 e 為底）
print(np.sin(np.pi / 2)) # 計算正弦值，π/2 的正弦為 1
print(np.cos(np.pi)) # 計算餘弦值，π 的餘弦為 -1

arr = np.array([-1, 1, 2, -3])
print(np.abs(arr)) # 計算絕對值（忽略負號）
print(np.power(arr, 3)) # 每個元素取三次方

print(np.round([3.2, 4.5, 8.1, 9.6])) # 四捨五入到最接近的整數

arr = np.array([1.6, 25.1, 81.7])
print(np.ceil(arr)) # 將數值無條件進位至最小整數
print(np.floor(arr)) # 將數值無條件捨去至最大整數
print(np.isnan([1, 2, np.nan, 3])) # 判斷哪些元素是 NaN（不是數值）
arr = np.random.randint(1, 200, 8) # 產生 8 個 1~199 之間的隨機整數
print(arr)
print(np.sum([1, 2, 3])) # np.sum: 將元素總和
print(np.mean(arr))      # np.mean: 平均值
print(np.median([4, 1, 2])) # np.median: 中位數
print(np.var(arr))       # np.var: 變異數（衡量資料離散程度）
print(np.max(arr), np.argmax(arr)) # np.max: 最大值, np.argmax: 最大值的 index
print(np.min(arr), np.argmin(arr)) # np.min: 最小值, np.argmin: 最小值的 index
print(np.percentile(arr, 80)) # np.percentile: 第 80 百分位數
arr = np.array([1, 2, 3])
print(np.cumsum(arr)) # np.cumsum: 累加和 [1, 1+2=3, 1+2+3=6] => [1 3 6]
print(np.cumprod(arr)) # np.cumprod: 累積乘積 [1, 1*2=2, 1*2*3=6] => [1 2 6]
```

```
[154 172  65 198  14 169  15 183]
6
121.25
2.0
5198.4375
198 3
14 4
178.6
[1 3 6]
[1 2 6]
```

✅ np.max()

回傳陣列中的最大值。

```
arr = np.array([1, 3, 7, 2])
print(np.max(arr))  # 7
```

✅ np.argmax()

回傳最大值所在的 index。

```
print(np.argmax(arr))  # 2
```

✅ np.where(condition)

回傳符合條件的 index。

```
arr = np.array([10, 20, 30, 40])
print(np.where(arr > 25))  # (array([2, 3]),)
```

✅ np.count_nonzero()

計算非零元素數量。

```
arr = np.array([0, 1, 2, 0, 3])
print(np.count_nonzero(arr))  # 3
```

✅ np.unique()

取得唯一值，並可加上 `return_counts=True` 得到每個值的次數。

```
arr = np.array([1, 2, 2, 3, 3, 3])
values, counts = np.unique(arr, return_counts=True)
print(values)  # [1 2 3]
print(counts)  # [1 2 3]
```

✅ np.cumsum()

累加總和。

```
arr = np.array([1, 2, 3])
print(np.cumsum(arr))  # [1 3 6]
```

✅ np.sort()

排序陣列（不改變原陣列）。

```
arr = np.array([3, 1, 2])
print(np.sort(arr))  # [1 2 3]
```

✅ np.argsort()

回傳排序後的 index。

```
arr = np.array([30, 10, 20])
print(np.argsort(arr))  # [1 2 0]
```

✅ np.amin() / np.min()

回傳最小值。

```
arr = np.array([5, 3, 8])
print(np.min(arr))  # 3
```

✅ np.amax() / np.max()

回傳最大值。

```
print(np.max(arr))  # 8
```

✅ np.unique(..., return_index=True)

找出唯一值及其第一次出現的 index。

```
arr = np.array([3, 1, 3, 2, 2])
values, index = np.unique(arr, return_index=True)
```

```
print(values)  # [1 2 3]
print(index)   # [1 3 0]
```

建立巢狀JSON資料

```
products = {
  "products": [
    {
      "id": f"P{i}",
      "name": f"Product {i}",
      "category": np.random.choice(["Electronics", "Clothing", "Home", "Food"]),
      "price": round(np.random.uniform(5, 200), 2),
      "in_stock": np.random.choice([True, False]),
      "specs": {
        "weight": round(np.random.uniform(0.1, 5), 2),
        "dimensions": {
          "length": np.random.randint(5, 50),
          "width": np.random.randint(5, 30),
          "height": np.random.randint(5, 20)
        }
      }
    } for i in range(1, 21)
  ]
}

import json
with open('data/products.json', 'w') as f:
  json.dump(products, f, indent=2)
```

📜 NumPy 練習題

🏋️ 1. Weather

題目描述：有一週的氣溫資料 `[28, 30, 29, 31, 32, 30, 29]`

- 計算平均、最高、最低氣溫
- 計算超過 30 度的天數
- 使用布林運算找出符合條件的資料

```
import numpy as np

temps = np.array([28, 30, 29, 31, 32, 30, 29])
print("Average Temperature:", np.mean(temps))
print("Max Temperature:", np.max(temps))
print("Min Temperature:", np.min(temps))
print("Days above 30 degrees:", (temps > 30).sum())

# 等價寫法
above_30 = np.where(temps > 30, 1, 0)
print(np.cumsum(above_30)[-1])
print(np.count_nonzero(temps > 30))
```

📚 2. Student Score Statistics

題目描述：有五位學生數學成績 `[85, 90, 78, 92, 88]`

- 計算平均、標準差與中位數
- 轉換成 10 分制百分等級

```
score = np.array([85, 90, 78, 92, 88])
print("Mean:", np.mean(score))
print("Median:", np.median(score))
print("Standard Deviation:", np.std(score))

# 10 分制
percentile_scores = score / 10
print("Percentile scores:", percentile_scores)
```

🧲 3. Matrix Calculation

題目描述：給定兩個 2x2 矩陸 A, B：

- A = [[1,2],[3,4]]
- B = [[5,6],[7,8]]

操作：

- A + B（元素相加）

- A * B（元素相乘）
- A @ B（矩陸乘法）

```
A = np.array([1, 2](1,%202))
B = np.array([5, 6](5,%206))
print("A + B =\n", A + B)
print("A * B =\n", A * B)
print("A @ B =\n", A @ B)
```

🎲 4. Random Matrix Operations

題目描述：

- 產生一個 3x4 隨機正數矩陸，範圍 [0,10)
- 每個 column 最大值
- 每個 row 最小值
- 替換所有奇數為 -1

```
matrix = np.random.randint(0, 10, (3, 4))
print("Matrix:\n", matrix)
print("Column-wise max:", np.max(matrix, axis=0))
print("Row-wise min:", np.min(matrix, axis=1))
print(np.where(matrix % 2 == 1, -1, matrix))
print(matrix[matrix % 2 == 1])
matrix[matrix % 2 == 1] = -1
print("Modified matrix:\n", matrix)
```

🔄 5. Transform Matrix

題目描述：

- 建立一個 1D array `[1~12]`
- 轉成 3x4 矩陸
- 計算每 row 平均值 & column 總和
- 再轉回 1D

```
arr = np.arange(1, 13)
arr = np.reshape(arr, (3, 4))
```

```
print("Row averages:", np.mean(arr, axis=1))
print("Column sums:", np.sum(arr, axis=0))
arr = np.reshape(arr, 12)
print("Flattened array:", arr)
```

🔍 6. Boolean Indexing

題目描述：

- 產生 5x5 的隨機數矩陸，範圍 [0, 20)
- 找出 >10 的值
- 將其替換成 0

```
np.random.seed(0)
arr = np.random.randint(0, 20, (5, 5))
print("Original matrix:\n", arr)
print("Elements > 10:", arr[arr > 10])
arr[arr > 10] = 0
print("Replaced matrix:\n", arr)
```