NumPy 陣列的索引、切片和迭代

簡報閱讀 範例與作業 問題討論 學習心得(完成)

重要知識點

- 知道如何從 NumPy 陣列存取元素
- 了解一維與多維陣列的迴圈用法
- 初步理解陣列與其迭代物件

從 Python 容器中取出元素的方法有索引（index）跟切片（slice），在 NumPy 的陣列中，也一樣保留了這兩種存取資料的用法。

```
L = [0, 1, 8]

print(L)
print(L[-1]) # 8
print(L[0]) # 0
print(L[1:3]) # [1, 8]
```

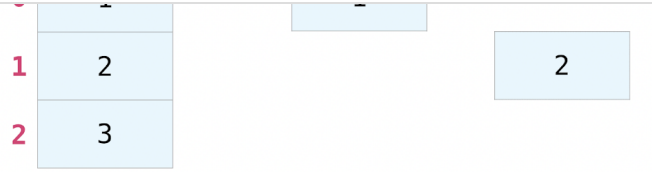
```
import numpy as np

a = np.arange(3) ** 3

print(a)
print(a[-1]) # 8
print(a[0]) # 0
print(a[1:3]) # [1 8]
```

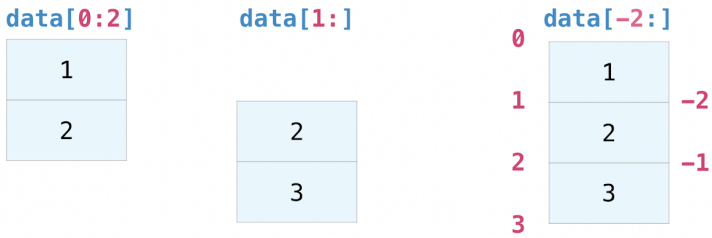
一維陣列的切片與索引

從一維陣列中取出元素的索引跟切片用法：

```
import numpy as np
```

```
2
3  data = np.array([1, 2, 3])
4
5  print(data[0]) # 取出第 0 個
6  print(data[1]) # 取出第 1 個
7  print(data[0:2]) # 第 0 - 1 個
8  print(data[1:]) # 第 1 到最後一個
9  print(data[-2:]) # 倒數第二到最後一個
```



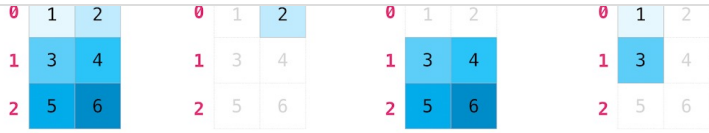
一維陣列的的迭代

如果想要對每個元素作運算的話，我們可以會想到用「迴圈」的方法，對陣列進行迭代：

```
1  import numpy as np
2
3  a = np.arange(3) ** 3
4
5  for i in a:
6      print(i)
7  # 0
8  # 1
9  # 8
```

多維陣列的切片與索引

而在多維陣列的資料存取中可以分層存取，用多維座標的方法進行存取：



多維陣列的多層迴圈

對於多維陣列如果想要迭代的話，可能就必須要一層一層的迴圈逐步取出資料：

```
import numpy as np

a = np.arange(6).reshape(3, 2)

for row in a:
    print(row)
# [0 1]
# [2 3]
# [4 5]
```

```
for row in a:
    for d in row:
        print(d)
# 0
# 1
# 2
# 3
```

```
# 4
# 5
```

攤平後再迭代多維陣列

因此，實務上我們會建議可以先攤平在進行迴圈的操作：


```
a = np.arange(6).reshape(3, 2)

for d in a.flat:
    print(d)
```

0
1
2
3
4
5

np.nditer 迭代物件

正確的用法會建立改成使用 np.nditer 迭代物件，這樣才可以享有向量運算的特性。

```
import numpy as np

a = np.arange(6).reshape(3, 2)

for d in np.nditer(a):
    print(d)
```

```
# 0
# 1
# 2
# 3
# 4
# 5
```

迭代物件的儲存方向

這樣才可以享有向量運算的特性。

```python
import numpy as np

a = np.arange(6).reshape(3, 2)

for d in np.nditer(a, order='C'):
    print(d)
```

```
# 0
# 1
# 2
# 3
# 4
# 5
```

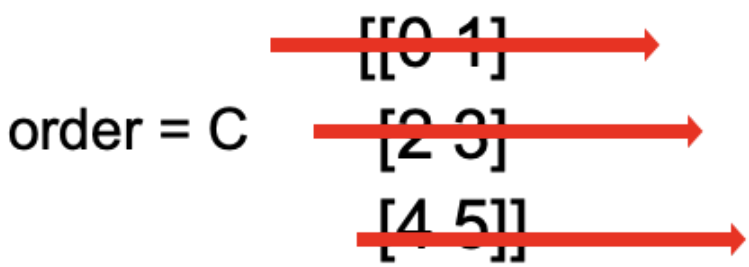
```python
import numpy as np

a = np.arange(6).reshape(3, 2)

for d in np.nditer(a, order='F'):
    print(d)
```

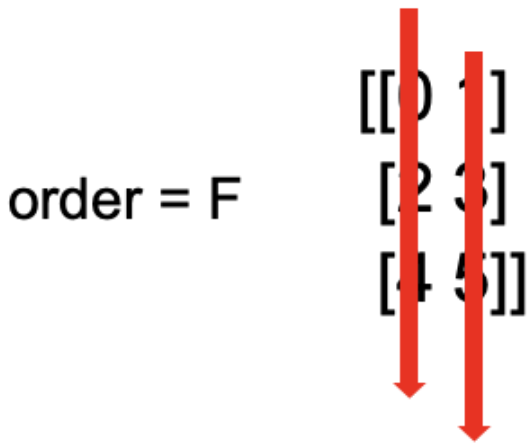
```
# 0
# 1
# 2
```

2
3
4
5

np.nditer 迭代物件


```
1  import numpy as np
2
3  a = np.arange(6).reshape(3, 2)
```





陣列中的矩陣特性與迴圈操作

綜合今天的內容來說，我們可以了解到陣列可以滿足容器中大部分的運算，不過這麼做就沒有享受到矩陣的運算特性。

NumPy陣列的索引和切片 (Slicing)

- 透過索引存取陣列元素或進行切片 (slicing)，可以使用索引值，或是 [start:stop:step] 語法取得範圍內的元素，要留意的是起始-結束範圍仍是 half-open

- 索引 -1 表示取得最後一個元素。切片如果只有給定 step 值為 -1 的話，則代表是反向取出，元素值是從最後一筆開始取出。

- 若沒有給定 start 或 stop 值的話則代表是取出該索引之前或之後的所有元素。若 start 和 stop 值都沒有給定的話，就是取出所有元素值。

知識點回顧

- 知道如何從 NumPy 陣列存取元素
- 了解一維與多維陣列的迴圈用法
- 初步理解陣列與其迭代物件

參考資料

python nditer - 迭代陣列
網站：csdn

透過範例介紹了 nditer 迭代物件的用法與操作。適合第一次學習者深入理解背後的觀念。

單個數組的迭代（Single Array Iteration）

迭代器最基本的任務的可以完成對數組元素的訪問，迭代器接口可以一個接一個地提供的每一個元素。

例如：

```
a = np.arange(6).reshape(2, 3)
for x in np.nditer(a):
    print x, " "
0  1  2  3  4  5
```

對於這種迭代方式需要注意的是：所選擇的順序是和數組內存佈局一致的，而不是使用標準C或者Fortran順序。這是為了
的，這反映了默認情況下只需訪問每個元素，而無需考慮其特定順序。我們可以通過迭代上述數組的轉置來看到這一點
問 **數組轉置的copy**的方式做對比，有：

```
a = np.arange(6).reshape(2, 3)
for x in np.nditer(a.T):
    print x,
print "\n"
for x in np.nditer(a.T.copy(order = 'C')):
    print x,

0 1 2 3 4 5

0 3 1 4 2 5
```


【Python 庫】NumPy 詳細教程（3）：ndarray 的內部機制及高級迭代

網站：cnblogs

本文深入介紹 ndarray 內部結構與迭代的各種用法，適合想要釐清觀念的同學查閱。

1、ndarray 的組成

*ndarray*與數組不同，它不僅僅包含數據信息，還包括其他描述信息。*ndarray*內部由以下內容組成：

- 數據指針：一個指向實際數據的指針。
- 數據類型（dtype）：描述了每個元素所佔字節數。
- 維度（shape）：一個表示數組形狀的元組。
- 跨度（strides）：一個表示從當前維度前進道下一維度的當前位置所需要“跨過”的字節數。

*NumPy*中，數據存儲在一個**均勻連續的內存塊**中，可以這麼理解，*NumPy*將多維數組在內部以一維數組的方式存儲，我們只要知道了每個元素所佔的字節數（dtype）以及每個維度中元素的個數（shape），就可以快速定位到任意維度的任意一個元素。

*dtype*及*shape*前文中已經有詳細描述，這裡我們來講下*strides*。

示例

```
ls = [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]],
      [[13, 14, 15, 16], [17, 18, 19, 20], [21, 22, 23, 24]]]
a = np.array(ls, dtype=int)
print(a)
print(a.strides)
```

輸出：

```
[[[ 1  2  3  4]
  [ 5  6  7  8]
  [ 9 10 11 12]]

 [[13 14 15 16]
  [17 18 19 20]
  [21 22 23 24]]]
(48, 16, 4)
```

下一步：閱讀範例與完成作業