

D08 NumPy 結構化陣列 (Structured Arrays)

[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 資料型別dtype
及對應

NumPy 資料型別dtype

NumPy 結構化陣列
(Structured Arrays)

NumPy 結構化陣列：
RecordArray

知識點回顧



重要知識點



用

- 介紹如何及操作結構化陣列 (Structured Arrays)

NumPy 資料型別dtype及對應

在NumPy中指定資料型別時，可以用字母、Python型別、NumPy通用資料型別來表示，下表列出各種型別及其對應。

資料型別	字母	Python資料型別	NumPy通用資料型別
boolean	?	bool	np.bool_
signed byte	b	bytes	np.bytes_
unsigned byte	B	bytes	np.bytes_
signed integer	i	int	np.int_
unsigned integer	u		np.uint
floating-point	f	float	np.float_
complex-floating point	c	complex	np.cfloat
timedelta	m	datetime.timedelta	np.timedelta64
datetime	M	datetime.datetime	np.datetime64
string	S, a	str	np.str_
Unicode string	U	str	np.str_

NumPy 資料型別dtype

透過 `numpy.dtype` 物件，指定要讀入各 Column 的資料型別，例如 `f8` 代表 `float64` (8 bytes)，`U5` 代表長度 5 以下的 `unicode` 字串。

```
# 使用字母代表的資料型別
dt = np.dtype('U5, i8, i8, U3')
```

在讀取文字檔時，可將定義的資料型別指定給 dtype 引數。

```
a = np.genfromtxt("structured.txt", delimiter=',', dtype=dt)
a
array([( 'Jay', 1, 2, 'Yes'), ('James', 3, 4, 'No'), ('Joe', 5, 6, 'Yes')],
      dtype=[('f0', '<U5'), ('f1', '<i8'), ('f2', '<i8'), ('f3', '<U3')])
```

NumPy 結構化陣列 (Structured Arrays)

建立結構化陣列可透過 dictionary 型別的資料建立 np.dtype 物件，並指定 dtype 給陣列。

資料型別可以使用 Python 的資料型別、NumPy 的資料型別、或是字母代表的型別皆可。在範例中我們混用了 3 種型別的代表方式：

```
dt = np.dtype({'names':('Name', 'num1', 'num2', 'True'), 'formats':((np.str_, 5), np.int32, int, 'U3')})
b = np.genfromtxt("structured.txt", delimiter=',', dtype=dt)
b
array([( 'Jay', 1, 2, 'Yes'), ('James', 3, 4, 'No'), ('Joe', 5, 6, 'Yes')],
      dtype=[('Name', '<U5'), ('num1', '<i4'), ('num2', '<i8'), ('True', '<U3')])
```

建立陣列後，可以用索引的方式存取元素資料。

```
b[0]
```

```
('Jay', 1, 2, 'Yes')
```

也可以用 Column 名稱，取得 Column 所有元素值。

```
b['Name']
```

```
array(['Jay', 'James', 'Joe'], dtype='<U5')
```

```
b[1]['True']
```

```
'No'
```

也可以進行邏輯操作，取得對應的結果。

```
b[b['num2'] >= 3]['Name']
```

```
array(['James', 'Joe'], dtype='<U5')
```

新建立一個結構化陣列，方式跟建立陣列非常類似。

下例使用 `zeros()` 初始化陣列，並指定 `dtype`。

```
c = np.zeros(3, dtype=dt)
c
array([(0, 0, 0), (0, 0, 0), (0, 0, 0)],
      dtype=[('Name', '<U5'), ('num1', '<i4'), ('num2', '<i8'), ('True', '<U3')])
```

將清單資料餵入結構化陣列中。

```
name = ['Chloe', 'Charlotte', 'Clara']
num_1 = [11, 12, 13]
num_2 = [14, 15, 16]
check = ['Y', 'Y', 'N']

c['Name'] = name
c['num1'] = num_1
c['num2'] = num_2
c['True'] = check

print(c)
[(0, 0, 0), (0, 0, 0), (0, 0, 0)]
```

NumPy 結構化陣列：RecordArray

RecordArray 與 Structured Array 非常類似，但是提供更多的屬性可以用來存取結構化陣列。不過 RecordArray 雖然方便但是在效能上會比原來的陣列差。使用方法如下：

```
rec.array([('Chloe', 11, 14, 'Y'), ('Charl', 12, 15, 'Y'),  
         ('Clara', 13, 16, 'N')],  
        dtype=[('Name', '<U5'), ('num1', '<i4'), ('num2', '<i8'), ('True', '<U3')])
```

原先我們是透過索引或是名稱存取元素值，但是 RecordArray 可以使用屬性的方式來取得。

```
c_rec.Name
```

```
array(['Chloe', 'Charl', 'Clara'], dtype='<U5')
```

知識點回顧

- 資料型別常在陣列中用到，NumPy 的 dtype 使用彈性很大，並且可以與 Python 資料型別交互使用，建議可以參照內容中提供的對照表。
- 除了數值資料之外，NumPy 陣列也可以儲存複合式資料，也就是包含不同資料型別的元素。這就是結構化陣列 (Structured Arrays) 的功能，進行後續的資料存取及處理。

延伸閱讀

NumPy Structured arrays 官方文件

網站：[numpy](https://numpy.org/doc/stable/reference/arrays-structured.html)

- Structured Datatype Creation
- Manipulating and Displaying Structured Datasets
- Automatic Byte Offsets and Alignment
- Field Titles
- Union types
- Indexing and Assignment to Structured arrays
 - Assigning data to a Structured Array
 - Assignment from Python Native Types (Cython)
 - Assignment from Scalars
 - Assignment from other Structured Arrays
 - Assignment involving subarrays
 - Indexing Structured Arrays
 - Accessing Individual Fields
 - Accessing Multiple Fields
 - Indexing with an integer to get a Structured Scalar
 - Viewing Structured Arrays Containing Objects
 - Structure Comparison
 - Record Arrays
 - Recarray Helper Functions

Previous topic

Byte-swapping

Next topic

Writing custom array containers

Full's search

Structured arrays are ndarrays whose datatype is a composition of simpler datatypes organized as a sequence of named **fields**. For example,

```

>>> x = np.array([(1, 'foo', 9, 81.0), ('fido', 3, 27.0)],
...              dtype=[('name', 'U10'), ('age', '<i>int32'), ('weight', '<i>float64')])
>>> x
array([(1, 'foo', 9, 81.0), ('fido', 3, 27.0)],
      dtype=[('name', '<i>U10'), ('age', '<i>int32'), ('weight', '<i>float64')])

```

Here **x** is a one-dimensional array of length two whose datatype is a structure with three fields: 1. A string of length 10 or less named 'name', 2. a 32-bit integer named 'age', and 3. a 64-bit float named 'weight'.

If you index **x** at position 1 you get a structure:

```

>>> x[1]
('fido', 3, 27.0)

```

You can access and modify individual fields of a structured array by indexing with the field name:

```

>>> x['age']
array([9, 3], dtype=int32)
>>> x['age'] = 5
>>> x
array([(1, 'foo', 5, 81.0), ('fido', 5, 27.0)],
      dtype=[('name', 'U10'), ('age', '<i>int32'), ('weight', '<i>float64')])

```

Structured datatypes are designed to be able to mimic 'structs' in the C language, and share a similar memory layout. They are meant for interfacing with C code and for low-level manipulation of structured buffers, for example for interpreting binary blobs. For these purposes they support specialized features such as subarrays, nested datatypes, and unions, and allow control over the memory layout of the structure.

Users looking to manipulate tabular data, such as stored in csv files, may find other pandas objects more suitable, such as `xarray`.

下一步：閱讀範例與完成作業

