Numpy 陣列中不同的資料型態

簡報閱讀 範例與作業 問題討論 學習心得(完成)

重要知識點

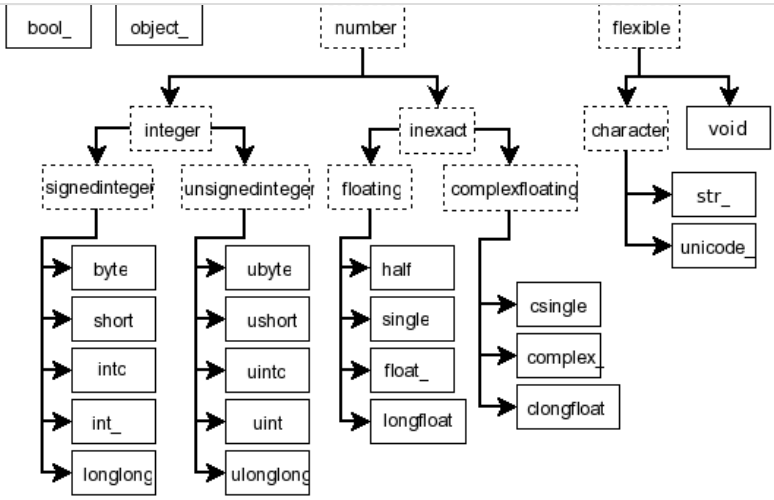
- 認識 NumPy 中提供的資料型態
- 了解各種型態的差異與定義
- 正確的比較型態判斷

NumPy 提供了一個同類型元素的多維容器型態，稱為是數組或陣列。陣列的全名是 N-dimensional Array，習慣簡寫為 NdArray 或 Array。Numpy 的類型系統中，提供以下幾種型態：

number, inexact, floating	float
complexfloating	cfloat
integer, signedinteger	int_
unsignedinteger	uint
character	string
generic, flexible	void

補充：更細緻的 Numpy 中的資料類型

Numpy 的型態其實不只有前面講的那幾個，其實有更多種的型態：



除了更多樣的型態之外，每一種型態也增加了範圍大小的彈性。

bool_	Boolean (True or False) stored as a byte
int_	Default integer type
intc	Identical to C int e.g int32 in64
intp	Integer used for indexing
int8	Byte (-128 to 127)
int16	Integer (-32768 to 32767)
int32	Integer (-2147483648 to 2147483647)
int64	Integer (-9223372036854775808 to 9223372036854775807)
uint8	Unsigned integer (0 to 255)
uint16	Unsigned integer (0 to 65535)
uint32	Unsigned integer (0 to 4294967295)
uint64	Unsigned integer (0 to 18446744073709551615)
float16	Half precision float: sign bit, 5 bits exponent, 10 bits mantissa
float32	Single precision float: sign bit, 8 bits exponent, 23 bits mantissa
float64	Double precision float: sign bit, 11 bits exponent, 52 bits mantissa
complex64	Complex number, represented by two 32-bit floats (real and imaginary components)
complex128	Complex number, represented by two 64-bit floats (real and imaginary components)

Numpy 資料型態與 Python 資料型態的差異

NumPy 的型態與 Python 內建的型態比起來細緻（尤其是數字型態），其目的是為了「更好的操作」與「更佳的儲存」。

Python type	NumPy type
str or mixed	string_, unicode_, mixed types
int	int_, int8, int16, int32, int64, uint8, ui uint32, uint64
float	float_, float16, float32, float64
bool	bool_
NA	datetime64(ns)
NA	NA
NA	NA

dtype 與 itemsize

在陣列的屬性當中，有兩個跟型態有關：

- ndarray.dtype： 陣列中的資料型態
- ndarray.itemsize： 陣列中每個元素佔用的空間

以上次使用的 a 變數的例子來看：

```
1 print(a.dtype) # int64
2 print(a.itemsize) # 8
```

int64 代表的是 64 個位元長度的 int，每個元素佔用 8 Bytes。

numpy 中的型態判斷

```
print(a.dtype == 'int64')
```

回傳 a 陣列的型態是否等於 int64 的布林結果。

numpy 中的屬性判斷

也可能會想用 is 的方式來做比較：

```
print(a.dtype is 'int64')
```

或使用 np.int64 表示 int64 型態：

```
print(a.dtype is np.int64)
```

屬性判斷比你想的還要複雜

嘗試過了前兩頁的比較之後，你可能會發現有一些「不如你預期」的結果出現，那究竟背後的邏輯是什麼呢？

在 NumPy 中有幾種表示型態的方法：

- 'int' 字串
- 'int64' 字串
- np.int64 物件
- np.dtype('float64') 物件

空間」等等的資訊。而「is」跟「==」的差異在於 is 是強比較，必須要所有規格都相同，包含是哪一種物件；而 == 的比較只會考慮自定義的規則，例如只需要物件的表現形式。

NumPy 的型態定義成用型態名稱的字串作為表示，因此在使用 == 比較的時候，會可以接受的字串或物件的形式；但使用 is 的話就必須要跟 np.dtype 物件來相比。

資料型態的正確用法

因此，正確的使用方法如下：

```
print(a.dtype == 'int64') # True
print(a.dtype is 'int64') # False
print(a.dtype is np.dtype('int64')) # True
```

另外， int 代表「執行電腦中最大可表示的範圍」，會取決於電腦不同而產生不同的結果：

```
print(a.dtype == 'int')
print(a.dtype == np.int)
print(a.dtype == np.dtype('int'))
```

補充：型態縮寫

NumPy 的型態有些可以縮寫寫成這樣：

```
dt = np.dtype('i4') # int4
```

`'?'`	boolean
`'b'`	(signed) byte
`'B'`	unsigned byte
`'i'`	(signed) integer
`'u'`	unsigned integer
`'f'`	floating-point
`'c'`	complex-floating point
`'m'`	timedelta
`'M'`	datetime
`'O'`	(Python) objects
`'S'`, `'a'`	zero-terminated bytes (not recommended)
`'U'`	Unicode string
`'V'`	raw data (**`void`**)

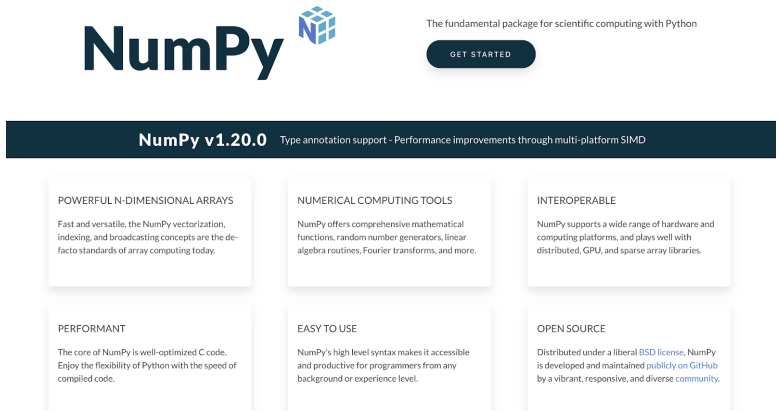
知識點回顧

- 認識 NumPy 中提供的資料型態
- 了解各種型態的差異與定義
- 正確的比較型態判斷

參考資料

Data type objects (dtype)
網站：NumPy

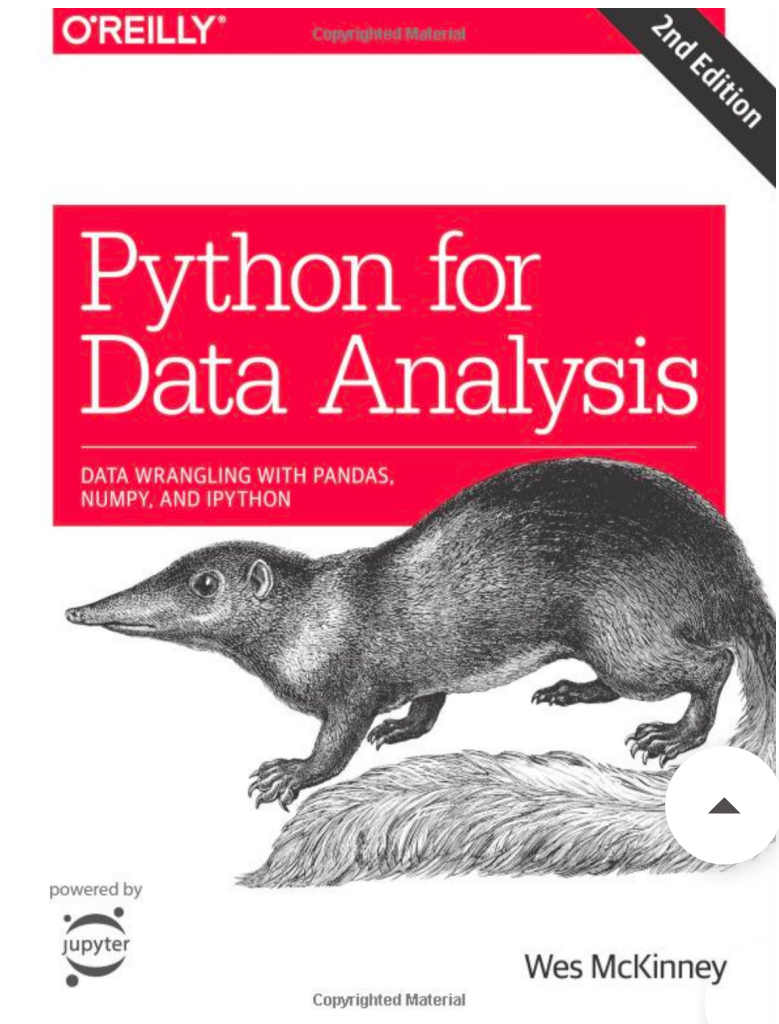
NumPy 的官方網站中對於型態的說明與介紹，這一篇就已經介紹完大部分的基礎內容了！



Chapter 12. Advanced NumPy

網站：[oreilly media]

來自經典書籍「Python for Data Analysis」的章節，當中以大量的實作範例說明的各種型態的差異。



重要知識點

Numpy 中的資料類型

補充：更細緻的 Numpy 中的資料類型

Numpy 資料型態與 Python 資料型態的差異

下一步：閱讀範例與完成作業