D06 使用 NumPy 存取各種檔案內容

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 I/O - save()、savez()、load()

NumPy I/O – 檔案格式效能比較

NumPy I/O - save()、

重要知識點

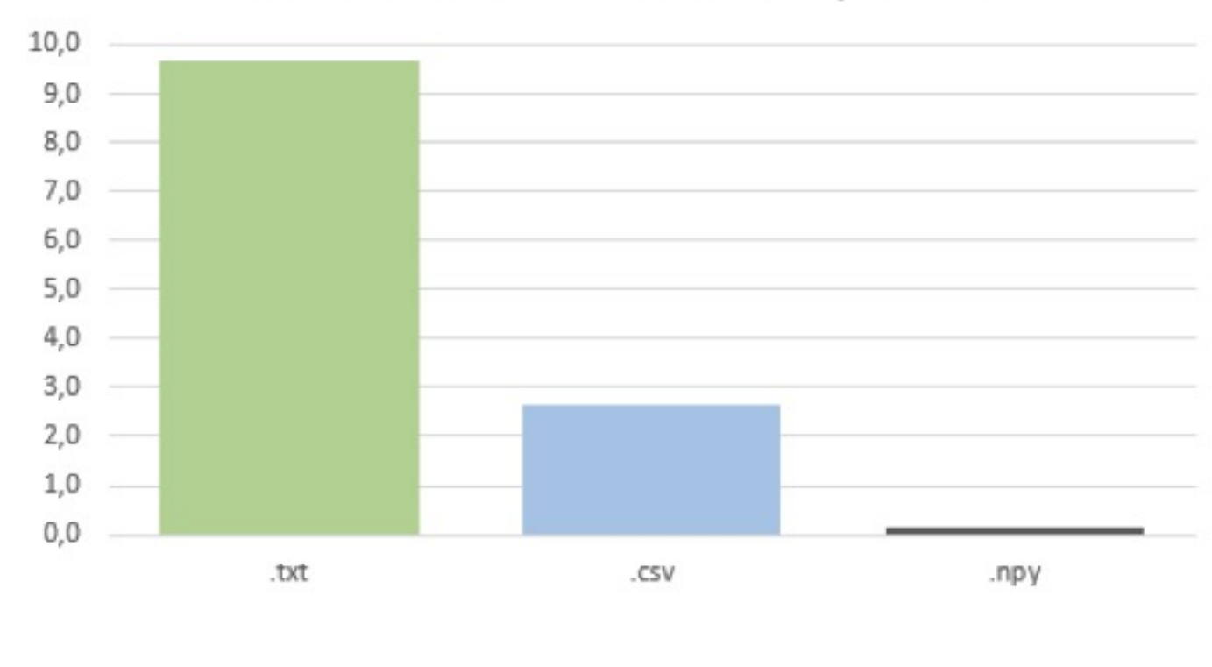
- NumPy 提供自己的高效能檔案格式，可透過 save()、load() 等函式進行儲存或讀取，儲存時也可使用 savez() 將檔案壓縮。
- 從一般的文字檔讀取或儲存，可以使用 savetxt() 與 loadtxt()。
- genfromtxt() 是一個功能強大且彈性的函式，能從文字檔中讀取陣列資料。

NumPy I/O - save()、savez()、load()

- numpy.save() 是將單一陣列儲存到 .npy 格式的函式。
- numpy.savez() 可以將多個陣列儲存到同一個 .npz 格式的檔案中。
- 讀取 .npy / .npz 檔案，使用 numpy.load() 函式來開啟檔案，並回傳檔案中的陣列。

NumPy I/O – 檔案格式效能比較

相較於 CSV 或 TXT 檔案，開啟 NumPy 格式的檔案在效能上快非常多。



參考資料：URL

NumPy I/O - save()、load()

儲存單一陣列到 .npy 檔案，並用 numpy.load() 載入回傳陣列。

```
with open('one_array.npy', 'wb') as f:
    np.save(f, np.array([1, 2]))
```

```
np.load('one_array.npy')
```

```
array([1, 2])
```

NumPy I/O – savez()

使用 numpy.savez() 時，可以儲存多個陣列。下面範例在儲存陣列時並指定陣列關鍵字 (array1，array2...)，若未指定的話預設會以 arr_0，arr_1... 關鍵字設定。

```
x = np.arange(10)
y = np.array([1, 2, 3])
z = np.random.rand(10)

with open('multi_array.npz', 'wb') as f:
    np.savez(f, array1=x, array2=y, array3=z)
```

NumPy I/O – savez() 與 load()

當呼叫 numpy.load() 載入 .npz 檔案時，回傳的會是 NpzFile 類別。

```
npzfile = np.load('multi_array.npz')
type(npzfile)
```

```
numpy.lib.npyio.NpzFile
```

透過 files 屬性回傳的 List，可以看到載入的物件裡面包含 3 個陣列，名稱分別為 array1，array2，array3。

```
npzfile.files
```

```
['array1', 'array2', 'array3']
```

顯示每一個陣列的內容。

```
print(npzfile['array1'])
print(npzfile['array2'])
print(npzfile['array3'])
```

```
[0 1 2 3 4 5 6 7 8 9]
[1 2 3]
[0.5788873  0.69082947 0.04922545 0.86517602 0.29889969 0.55285575
 0.69389689 0.49183867 0.8308363  0.06454569]
```

NumPy I/O – savetxt()

- savetxt() 可將一維或是二維陣列儲存到文字檔，並且可以設定元素值的格式、分隔符號、換行字元、檔頭 (header)、檔尾 (footer)、檔案字元編碼... 等引數。
- 需注意，如果儲存的陣列是一維的話，須加上中括號才能正常產生符號分隔檔格式，否則分隔符號會被忽略。範例如下：

```
np.savetxt('test.out', [x], delimiter=',')
```

- 如果檔案副檔名為 .gz 的話，存檔時會存為壓縮的 gzip 檔案。

```
np.savetxt('test.gz', [x], delimiter=',')
```

- 使用 fmt 引數可以指定輸出的格式，下例是指定科學記號的格式來輸出陣列值。
- 在存檔時也可以加入 header / footer 做為檔案註解說明。

```
np.savetxt('test.out', x, fmt='%1.4e', delimiter=',', header='this is,\nheader', footer='this is footer')
```

NumPy I/O – loadtxt()

- loadtxt() 函式與稍後會介紹的 genfromtxt() 函式有一些相同的引數及功能，但是 genfromtxt() 功能更有彈性，所以相關的功能會一併在 genfromtxt() 中介紹。
- loadtxt() 函式定義如下：

```
numpy.loadtxt(fname, dtype=<class 'float'>, comments='#', delimiter=None, converters=None, skiprows=0, usecols=
None, unpack=False, ndmin=0, encoding='bytes', max_rows=None)
```

NumPy I/O – genfromtxt()

跟 loadtxt() 相比，genfromtxt() 提供更 powerful 及更有彈性的功能，用來讀取文字檔格式的陣列。函式定義如下：

```
numpy.genfromtxt(fname, dtype=<class 'float'>, comments='#', delimiter=None, skip_header=0, skip_footer=0, conv
erters=None, missing_values=None, filling_values=None, usecols=None, names=None, excludelist=None, deletechars
=" !#$%&'()*+, -./:;<=>?@[\]^{|}~", replace_space='_', autostrip=False, case_sensitive=True, defaultfmt='f%i',
unpack=None, usemask=False, loose=True, invalid_raise=True, max_rows=None, encoding='bytes')
```

要將文字檔內容讀入並正確分隔 Column，才能獲得預期中的陣列及元素值。常用的分隔符號有逗號、tab...

genfromtxt() 預設的分隔符號為None，所以必須指定正確的分隔符號。

```
np.genfromtxt("test.csv", delimiter=",")
```

```
array([[0., 1., 2., 3., 4.],
       [5., 6., 7., 8., 9.]])
```

```
np.genfromtxt("test.csv", delimiter=",")
```

當 delimiter 給定的是一個整數、或是整數的序列時，可以用來將固定寬度的字串讀入，在下面的範例中，固定寬度包含了空格。

```
from io import StringIO

data = u"  1  2  3\n  4  5 67\n890123  4"
np.genfromtxt(StringIO(data), delimiter=3)
```

```
array([[  1.,   2.,   3.],
       [  4.,   5.,  67.],
       [890., 123.,   4.]])
```

```
from io import StringIO

data = u"  1  2  3\n  4  5 67\n890123  4"
np.genfromtxt(StringIO(data), delimiter=3)
```

- 如果給定的是單一整數代表所有陣列元素都是同一寬度；如有不同寬度時，可以使用整數序列來定義。

autostrip 引數如果設為 True，在讀取時會自動將元素值的空格去除。

```
data = u"1, 2 , 4\n 4, 5, 6"
np.genfromtxt(StringIO(data), delimiter=",", autostrip=True)
```

```
array([[1., 2., 4.],
       [4., 5., 6.]])
```

```
data = u"1, 2 , 4\n 4, 5, 6"
np.genfromtxt(StringIO(data), delimiter=",", autostrip=True)
```

與 loadtxt() 相同，讀取時可以略過註解文字，或是 header / footer。comments 引數值代表註解是由 # 起始的 row。

```
np.genfromtxt("test.out", comments="#")
```

```
array([0., 1., 2., 3., 4.])
```

```
np.genfromtxt("test.out", comments="#")
```

header 有 2 行而 footer 有 1 行，設定要略過的行數就可以正確讀入欲讀取的陣列元素。例如：

```
np.genfromtxt("test.out", comments=None, skip_footer=1, skip_header=2)
```

```
array([0., 1., 2., 3., 4.])
```

```
np.genfromtxt("test.out", comments=None, skip_footer=1, skip_header=2)
```

names 引數是用來指明是否檔案內容中有 Column 名稱，或是如果原來內容沒有的話，可以給定 Column名稱。

names=True 代表這個讀入的內容中有 Column 名稱。

```
np.genfromtxt("names.txt", delimiter=",", names=True)
```

```
array([(1., 2., 3.), (4., 5., 6.), (7., 8., 9.)],
      dtype=[('a', '<f8'), ('b', '<f8'), ('c', '<f8')])
```

```
np.genfromtxt("names.txt", delimiter=",", names=True)
```

若是原始資料中沒有名稱，可以透過 names 指定。

```
data = StringIO("1 2 3\n 4 5 6")
np.genfromtxt(data, names="a, b, c")
```

```
array([(1., 2., 3.), (4., 5., 6.)],
      dtype=[('a', '<f8'), ('b', '<f8'), ('c', '<f8')])
```

```
data = StringIO("1 2 3\n 4 5 6")
np.genfromtxt(data, names="a, b, c")
```

透過 usecols 引數可以選擇要讀入的 Column，下面的例子是指定要讀入的 Column 名稱。

```
a = u"1,2,3,4,5\n6,7,8,9,10"
np.genfromtxt(StringIO(a), delimiter=",", names="a, b, c", usecols=("a", "c"))
```

```
array([(1., 3.), (6., 8.)], dtype=[('a', '<f8'), ('c', '<f8')])
```

```
a = u"1,2,3,4,5\n6,7,8,9,10"
np.genfromtxt(StringIO(a), delimiter=",", names="a, b, c", usecols=("a", "c"))
```

如果沒有 Column 名稱的話，可以使用整數指定要讀取的 Column 索引值。

```
a = u"1 2 3 4 5\n6 7 8 9 10"
np.genfromtxt(StringIO(a), usecols=(1, -1))
```

```
array([[ 2.,  5.],
       [ 7., 10.]])
```

```
a = u"1 2 3 4 5\n6 7 8 9 10"
np.genfromtxt(StringIO(a), usecols=(1., -1))
```

- 如果沒有給定 names 或是給的數目少於 Column，那麼在回傳結構化陣列時，會自動以 f%i 的命名規則產生 names。
- 但是若已有 names 的話，使用索引值會產生錯誤訊息。

預設空值都被視為缺值 (missing value)，用 filling_values 可以指定要填值 (filling value)。
例如：在讀取檔案時，將缺值都設為 np.nan。

```
a = u", 2, 3\n4, ,"
np.genfromtxt(StringIO(a), delimiter=",", filling_values=np.nan)
```

```
array([[nan,  2.,  3.],
       [ 4., nan, nan]])
```

```
a = u", 2, 3\n4, ,"
np.genfromtxt(StringIO(a), delimiter=",", filling_values=np.nan)
```

除了空值之外，若有特定字串應被視為缺值的話，使用 missing_values 引數可以指定，而且可以使用序列來指定缺值與填值。要留意的是，使用字串序列的話，要每個Column依序指定。

```
a = u"N/A, 2, 3, ???"
np.genfromtxt(StringIO(a), delimiter=",",
              missing_values=["N/A", "N/A", "N/A", "???"],
              filling_values=[0, 0, 0, -999])
```

```
array([   0.,    2.,    3., -999.])
```

```
a = u"N/A, 2, 3, ???"
np.genfromtxt(StringIO(a), delimiter=",",
              missing_values=["N/A", "N/A", "N/A", "???"],
              filling_values=[0, 0, 0, -999])
```

在讀取檔案時使用 converters 引數可以同時轉換資料。例如在檔案中，資料包含 Yes/No 與百分比，在讀取時呼叫自訂的 trans() 與 conversion() 函式進行轉換。

```
np.genfromtxt("transform.txt", delimiter=',', converters={2:trans, 3:conversion})
```

```
array([(1., 2., 1, 0.87), (3., 4., 0, 0.03), (5., 6., 1, 0.55)],
      dtype=[('f0', '<f8'), ('f1', '<f8'), ('f2', '<i8'), ('f3', '<f8')])
```

```
np.genfromtxt("transform.txt", delimiter=',', converters={2:trans, 3:conversion})
```

複製程式碼

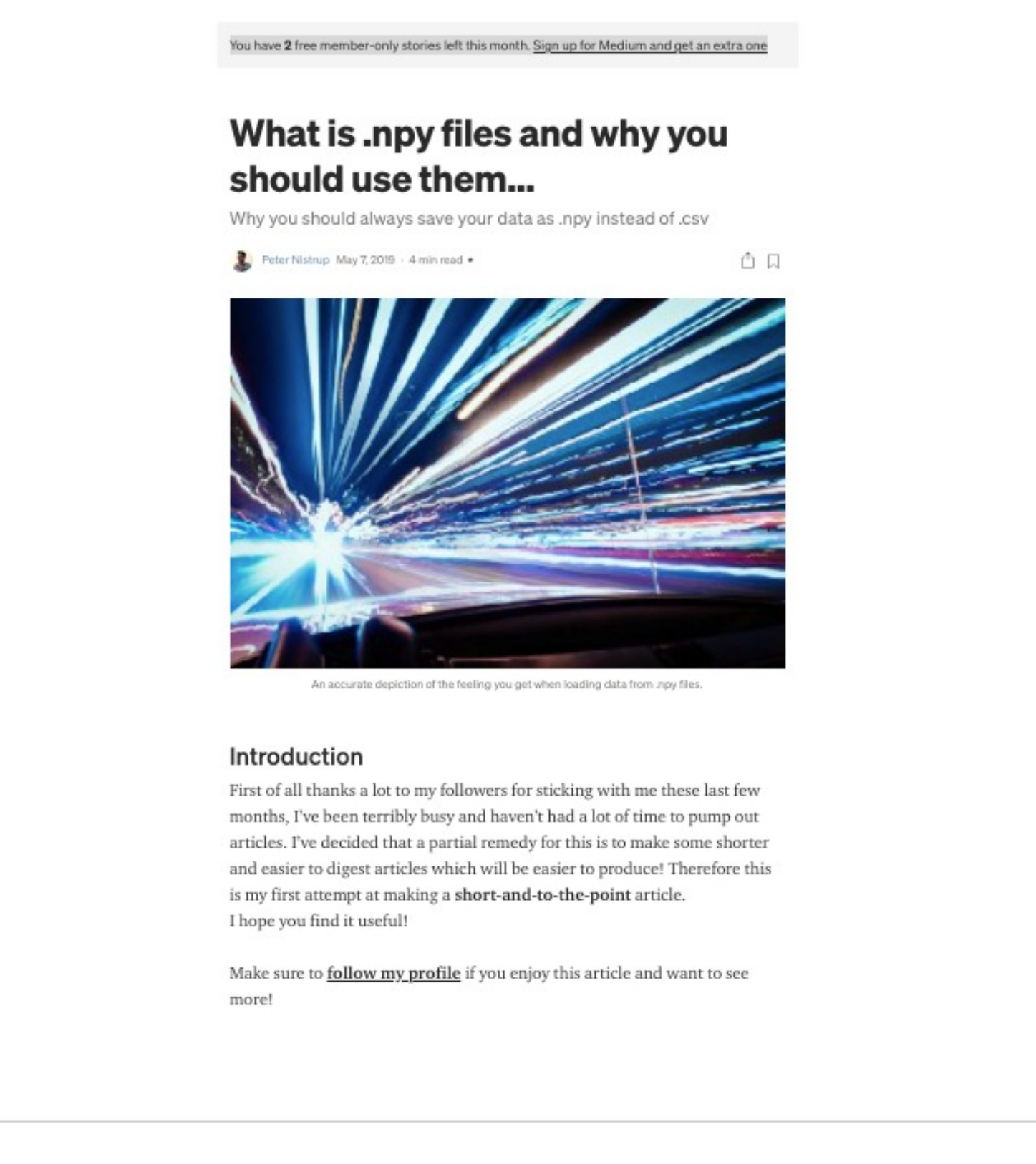
- converters 引數接收的是字典型別 (dictionary)，key 代表的是 Column，可以使用索引或是 names定義的 Column 名稱。

知識點回顧

- .npy 與 .npz 格式是NumPy的檔案格式，透過 save()、savez()、load() 函式進行儲存與讀取。
- 針對文字檔，可以使用 savetxt()、loadtxt() 來儲存與讀取。功能更強大的 genfromtxt() 則是提供更多選項在讀取檔案時進行操作。

延伸閱讀

What is .npy files and why you should use them...
網站：towardsdatascience



You have 2 free member-only stories left this month. Sign up for Medium and get an extra one

What is .npy files and why you should use them...

Why you should always save your data as .npy instead of .csv

Peter Nistrup May 7, 2019 · 4 min read

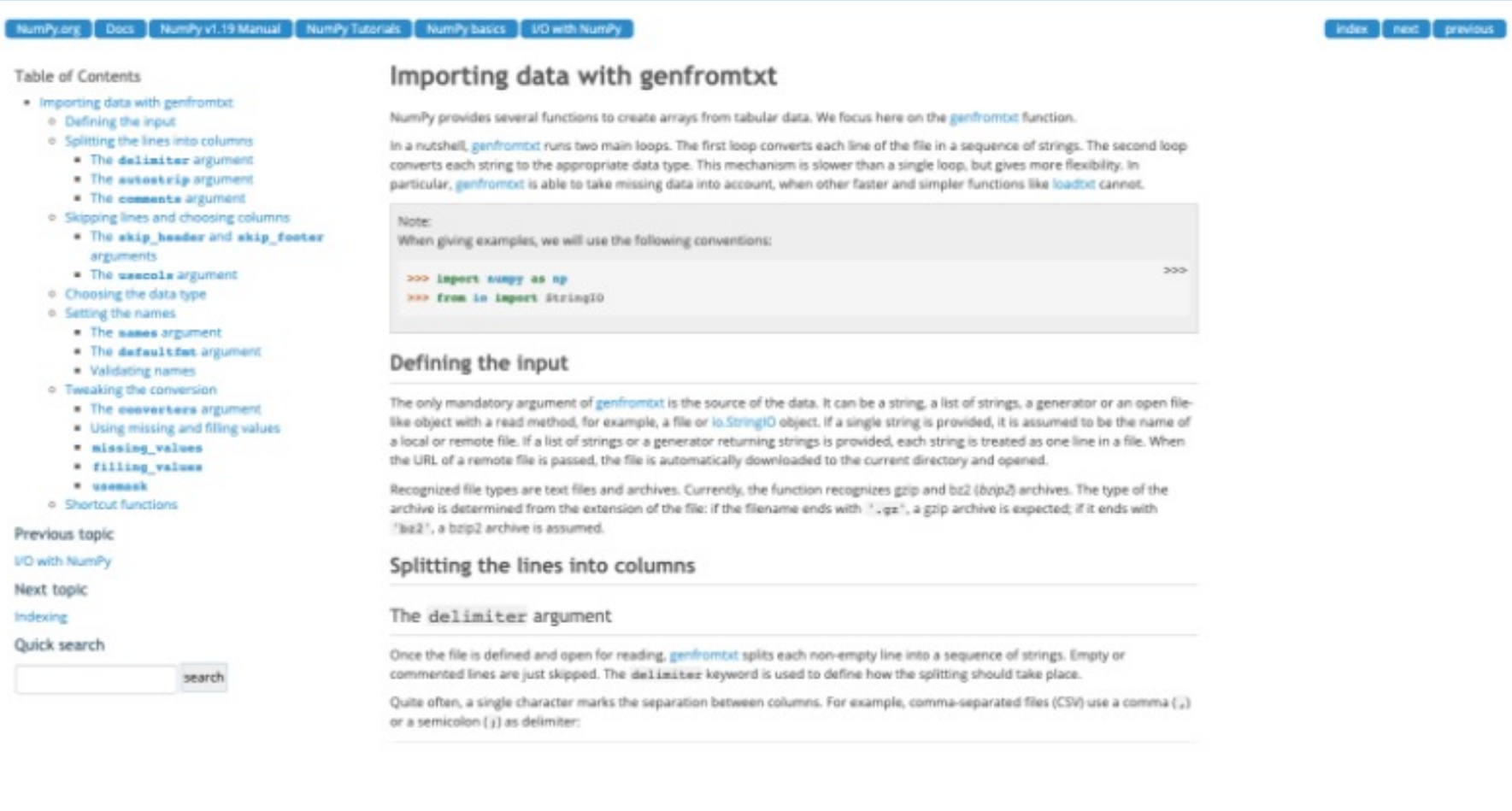
An accurate depiction of the feeling you get when loading data from .npy files.

Introduction

First of all thanks a lot to my followers for sticking with me these last few months, I've been terribly busy and haven't had a lot of time to pump out articles. I've decided that a partial remedy for this is to make some shorter and easier to digest articles which will be easier to produce! Therefore this is my first attempt at making a **short-and-to-the-point** article.
I hope you find it useful!

Make sure to **follow my profile** if you enjoy this article and want to see more!

Importing data with genfromtxt
網站：numpy



Importing data with genfromtxt

NumPy provides several functions to create arrays from tabular data. We focus here on the genfromtxt function.

In a nutshell, genfromtxt runs two main loops. The first loop converts each line of the file in a sequence of strings. The second loop converts each string to the appropriate data type. This mechanism is slower than a single loop, but gives more flexibility. In particular, genfromtxt is able to take missing data into account, when other faster and simpler functions like loadtxt cannot.

Note:
When giving examples, we will use the following conventions:

```
>>> import numpy as np
>>> from io import StringIO
```

Defining the input

The only mandatory argument of genfromtxt is the source of the data. It can be a string, a list of strings, a generator or an open file-like object with a read method, for example, a file or io.StringIO object. If a single string is provided, it is assumed to be the name of a local or remote file. If a list of strings or a generator returning strings is provided, each string is treated as one line in a file. When the URL of a remote file is passed, the file is automatically downloaded to the current directory and opened.

Recognized file types are text files and archives. Currently, the function recognizes gzip and bz2 (bzip2) archives. The type of the archive is determined from the extension of the file: if the filename ends with '.gz', a gzip archive is expected; if it ends with 'bz2', a bzip2 archive is assumed.

Splitting the lines into columns

The `delimiter` argument

Once the file is defined and open for reading, genfromtxt splits each non-empty line into a sequence of strings. Empty or commented lines are just skipped. The `delimiter` keyword is used to define how the splitting should take place.

Quite often, a single character marks the separation between columns. For example, comma-separated files (CSV) use a comma (,) or a semicolon (;) as delimiter:

下一步：閱讀範例與完成作業