6.2.基礎資料分析: NUMPY與Matplotlib實戰

6 人工智慧與資訊安全

單元6-2.基礎資料分析: NUMPY與Matplotlib實戰

學習目標

- 本單元將講授資料科學與資料分析
- 本單元包括Numpy學習基礎矩陣運算，接著透過Matplotlib實作資料視覺化，將資料以圖形、圖示呈現，讓學生了解資料分析之重要性與應用。
- 本單元以Google Colab平台教學,並提供[神經網路與邏輯運算]趣味的主題

學習主題

1. 資料科學與資料分析
2. Numpy基礎矩陣運算
3. Matplotlib資料視覺化技術
4. 應用: 神經網路與邏輯運算

學習資源

[1]提供上課簡報

本課程適合已具備基本Python程式的學生

前言

各種大量資料的分析問題

手寫識別程式怎麼做？

如何實現人臉識別系統？

如何過濾垃圾郵件？

電子商務網站上猜你喜歡的商品是什麼原理？如何實現？

電影網站如何去推薦符合用戶喜好的電影？

如何利用機器學習對消費者的特性進行細分，從而更好地服務各細分市場的消費者？

銀行如何去檢測用戶的信用卡可能被盜了？

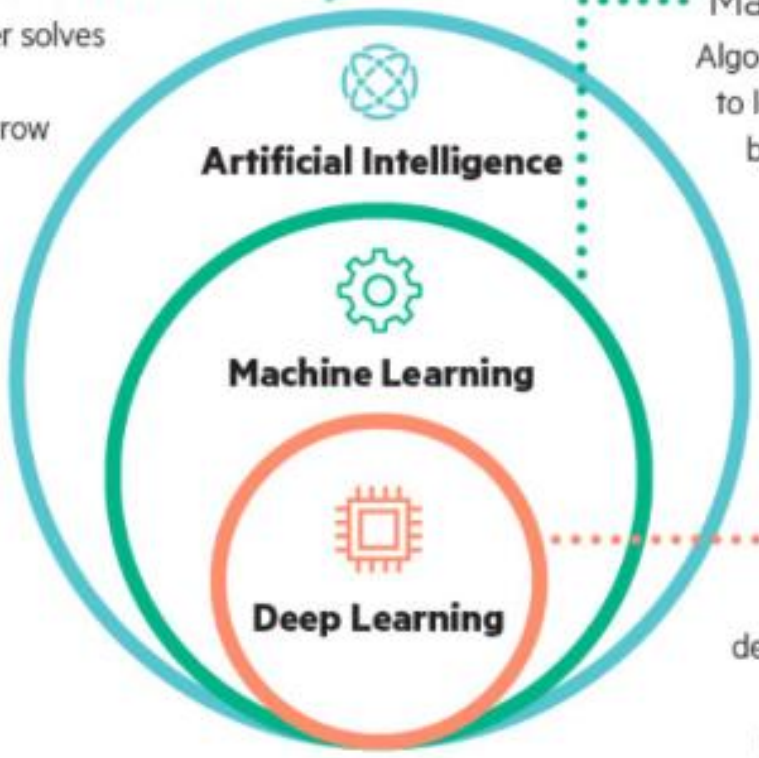
從工人(專家)智慧到人工(機器)智慧

What Makes a Machine Intelligent?

While AI is the headliner, there are actually subsets of the technology which can be applied to solving human problems in different ways.

Artificial Intelligence (AI)

A process where a computer solves a task in a way that mimics human behavior. Today, narrow AI—when a machine is trained to do one particular task—is becoming more widely used, from virtual assistants to self-driving cars to automatic tagging your friends in your photos on Facebook.



Machine Learning (ML)

Algorithms that allow computers to learn from examples without being explicitly programmed.

Deep Learning (DL)

A subset of ML which uses deep artificial neural networks as models and does not require feature engineering.

學習套件

AI Deep Learning

Keras

TensorFlow

PYTORCH

Caffe2

OpenAI

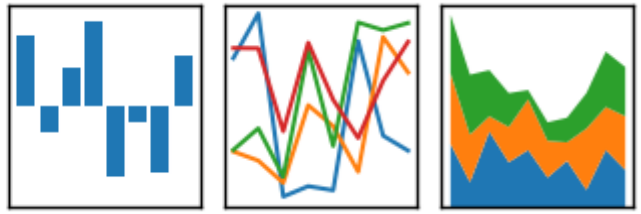
Machine Learning

scikit learn

Data Science

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



Pandas

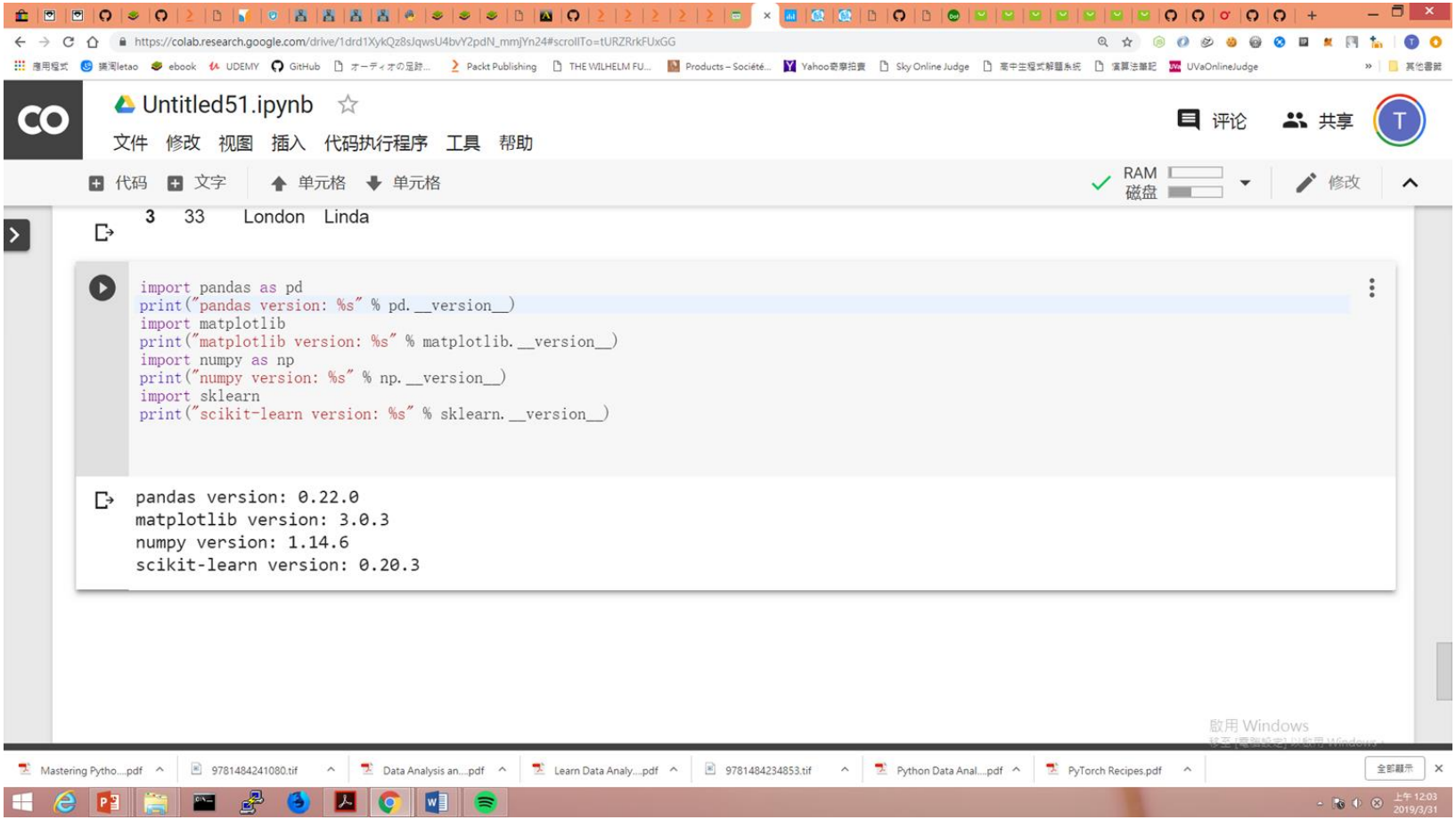
NumPy

matplotlib

學習平台

確認套件版本

```
import pandas as pd
print(“pandas version: %s” % pd.__version__)
import matplotlib
print("matplotlib version: %s" % matplotlib.__version__)
import numpy as np
print("numpy version: %s" % np.__version__)
import sklearn
print("scikit-learn version: %s" % sklearn.__version__)
```


https://colab.research.google.com/drive/1drd1XykQz8sJqwsU4bvY2pdN_mmjYn24#scrollTo=tURZRrkFUxGG
Untitled51.ipynb
文件 修改 视图 插入 代码执行程序 工具 帮助
评论
共享
代码 文字 单元格 单元格
RAM
磁盘
修改
3 33 London Linda
import pandas as pd
print("pandas version: %s" % pd.__version__)
import matplotlib
print("matplotlib version: %s" % matplotlib.__version__)
import numpy as np
print("numpy version: %s" % np.__version__)
import sklearn
print("scikit-learn version: %s" % sklearn.__version__)
pandas version: 0.22.0
matplotlib version: 3.0.3
numpy version: 1.14.6
scikit-learn version: 0.20.3
啟用 Windows
Mastering Pytho....pdf
9781484241080.tif
Data Analysis an....pdf
Learn Data Analy....pdf
9781484234853.tif
Python Data Anal....pdf
PyTorch Recipes.pdf
全部顯示

AI環境建置

- Ubuntu 18.04 LTS 64-bit
- 安裝anaconda
- 安裝Tensorflow
- 安裝PYTORCH
- 安裝open AI gym

Data Science 資料科學

計算機(演算法)如何處理底下資料:

文字(詩/小說/新聞/網路封包/log檔/email)

圖片(人臉辨識/醫療片/晶圓片品管,,)

聲音 對話 音樂

影片

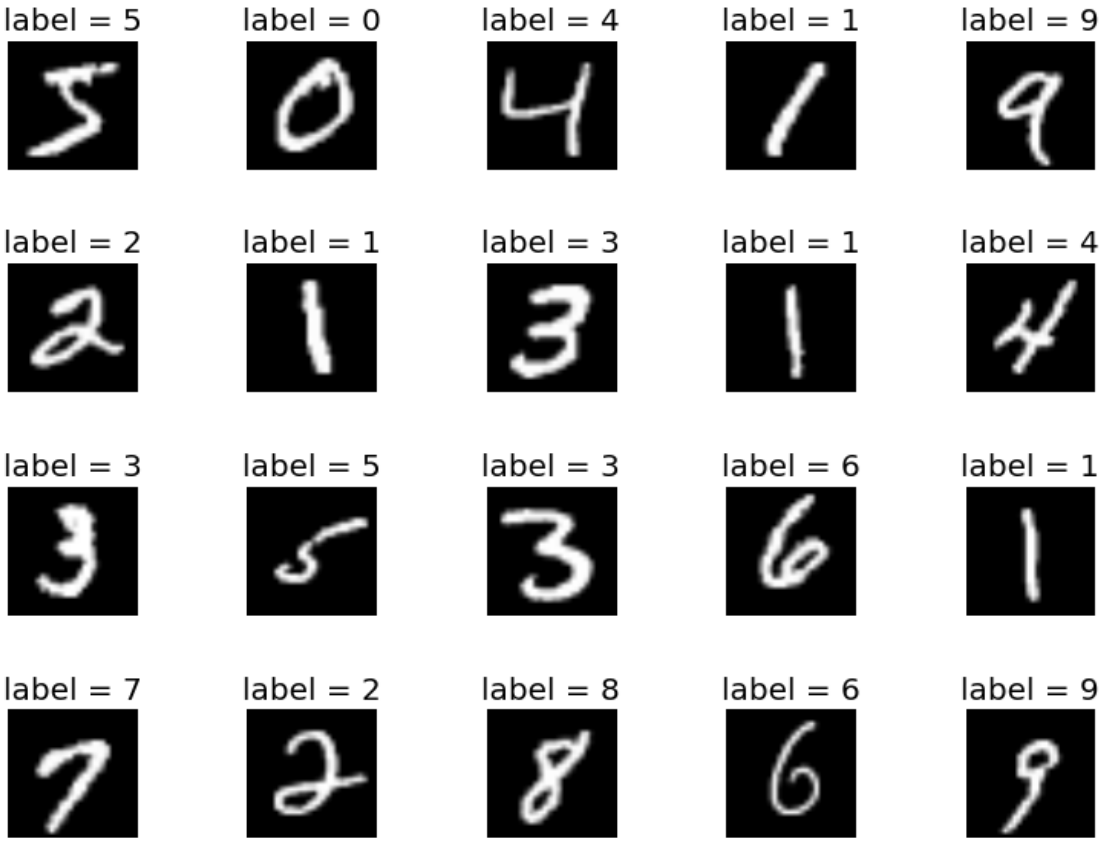
MNIST handwritten digit

MNIST 手寫數字資料集

訓練集為 60,000 張 28x28 圖元灰度圖像，

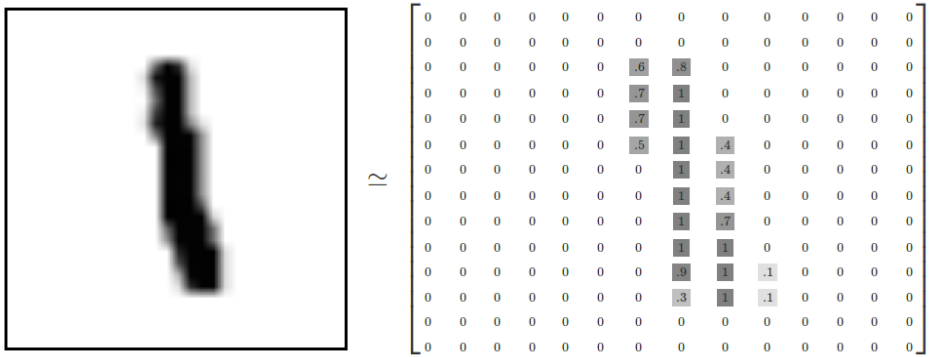
測試集為 10,000 同規格圖像，

總共 10 類數字標籤。

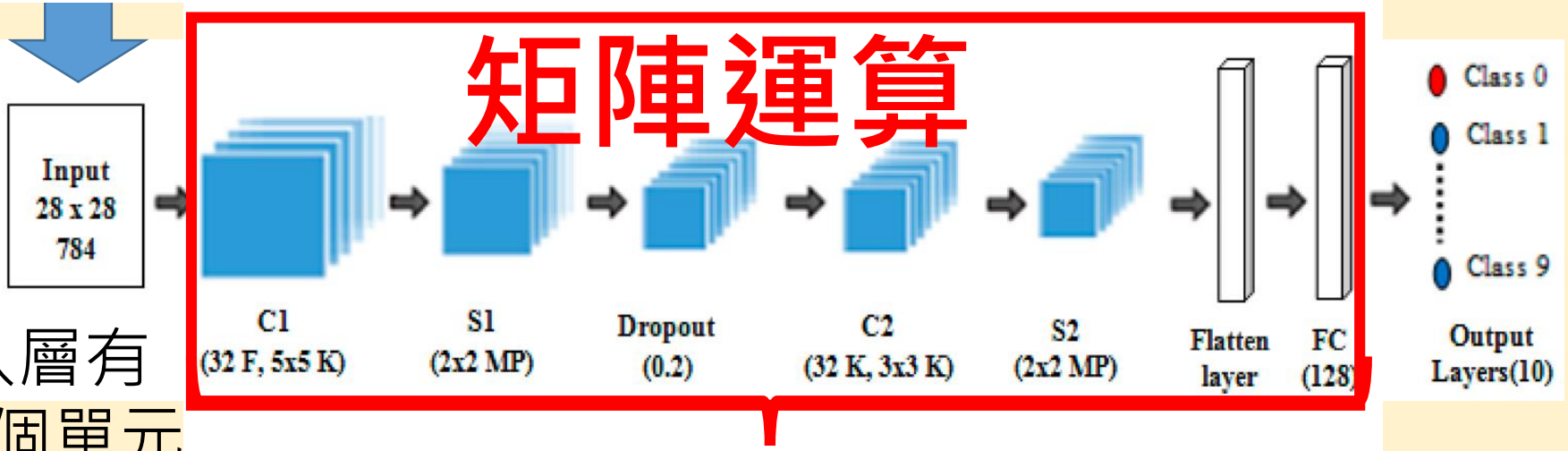


每一張手寫的數字都是
28*28(=784)

用28*28矩陣來代表一張圖



輸入層有
784個單元



網路架構設計:使用那些層
Convolution? Pooling? Dropout?Dense?

Python list ?

Python 資料分析
NUMPY套件

https://zh.wikipedia.org/wiki/NumPy

NumPy是Python語言的一個擴充程式庫。支援高階大量的維度陣列與矩陣運算，此外也針對陣列運算提供大量的數學函式函式庫。

NumPy的前身Numeric最早是由Jim Hugunin與其它協作者共同開發，2005年，Travis Oliphant在Numeric中結合了另一個同性質的程式庫Numarray的特色，並加入了其它擴充功能而開發了NumPy。NumPy為開放原始碼並且由許多協作者共同維護開發。

NumPy的核心功能是**"ndarray"(即n-dimensional array，**多維陣列)資料結構。這是一個表示多維度、同質並且固定大小的陣列物件。而由一個與此陣列相關聯的資料型態物件來描述其陣列元素的資料格式(例如其字元組順序、在記憶體中佔用的字元組數量、整數或者浮點數等等)。

https://en.wikipedia.org/wiki/Travis_Oliphant

Travis Oliphant is an American data scientist and businessman. He is founder of technology startup **Anaconda** (previously Continuum Analytics).
In addition, he is the primary creator of **NumPy** and founding contributor to the **SciPy** packages in the Python programming languages

https://www.slideshare.net/teoliphant/scale-up-and-scale-out-anaconda-and-pydata

CONTINUUM ANALYTICS

Scale Up and Scale Out with the Anaconda Platform

Travis Oliphant
CEO

ANACONDA

Scale up and Scale Out Anaconda and PyData

Travis Oliphant, PhD — About me

CONTINUUM ANALYTICS

- PhD 2001 from Mayo Clinic in Biomedical Engineering
- MS/BS degrees in Elec. Comp. Engineering from BYU
- Created **SciPy** (1999-2009)
- Professor at BYU (2001-2007)
- Author and Principal Dev of **NumPy** (2005-2012)
- Started **Numba** (2012)
- Founding Chair of **NumFocus** / **PyData**
- Former PSF Director (2015)
- Founder of Continuum Analytics in 2012.

NumPy and SciPy: History and Ideas for the Future

Travis E. Oliphant

SciPy Tokyo. March 18, 2012

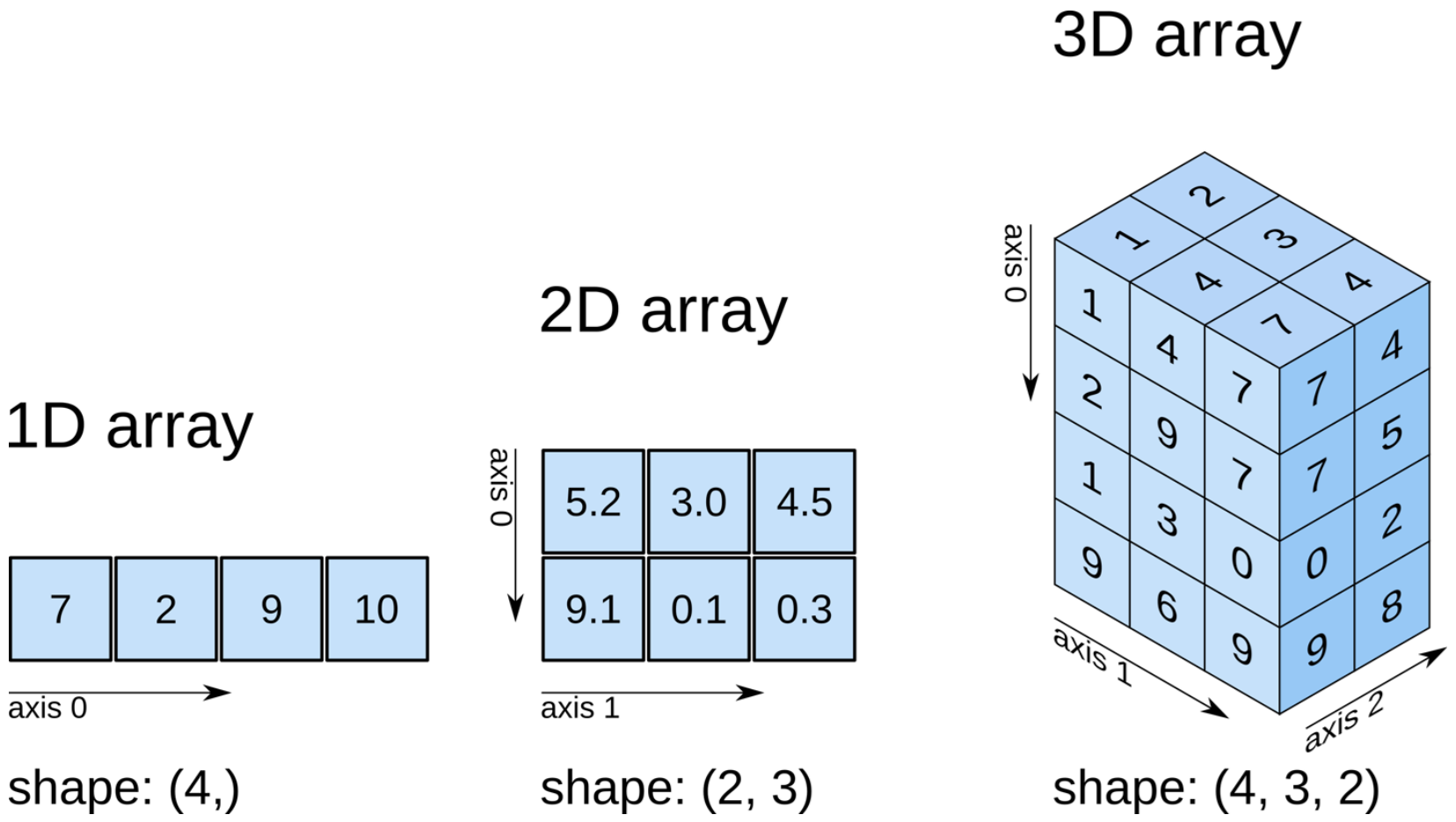
https://www.slideshare.net/shoheihido/sci-pyhistory

NUMPY ndarray

N-Dimensional Arrays

NumPy N-Dimensional Arrays

ndarray, or just array



NUMPY ndarray

N-Dimensional Arrays

重要屬性

import numpy as np

```
ar2=np.array([[0,3,5],[2,8,7]]) # 2D array
```

Shape of the array

ar2.shape

Number of dimensions

ar2.**ndim**

ndarray.ndim 秩，即軸的數量或維度的數量

ndarray.shape 陣列的維度，對於矩陣，n 行 m 列

NUMPY ndarray

N-Dimensional Arrays

資料型態(dtype)與型態轉換(astype)

```
import numpy as np

    ar=np.array([2,4,6,8]);
    ar.dtype

    ar=np.array([2,-1,6,3], dtype='float' );
    ar
    ar.dtype

   ar=np.array([2.,4,6,8]);
    ar.dtype
```

型態轉換(astype)

```
f_ar = np.array([13,-3,8.88])
f_ar

intf_ar=f_ar.astype(int
)
intf_ar
```

Table 4-2. NumPy data types

Type	Type code	Description
`int8, uint8`	`i1, u1`	Signed and unsigned 8-bit (1 byte) integer types
`int16, uint16`	`i2, u2`	Signed and unsigned 16-bit integer types
`int32, uint32`	`i4, u4`	Signed and unsigned 32-bit integer types
`int64, uint64`	`i8, u8`	Signed and unsigned 64-bit integer types
`float16`	`f2`	Half-precision floating point
`float32`	`f4 or f`	Standard single-precision floating point; compatible with C float
`float64`	`f8 or d`	Standard double-precision floating point; compatible with C double and Python `float` object
`float128`	`f16 or g`	Extended-precision floating point
`complex64, complex128, complex256`	`c8, c16, c32`	Complex numbers represented by two 32, 64, or 128 floats, respectively
`bool`	`?`	Boolean type storing `True` and `False` values
`object`	`O`	Python object type; a value can be any Python object
`string_`	`S`	Fixed-length ASCII string type (1 byte per character); for example, to create a string dtype with length 10, use `'S10'`
`unicode_`	`U`	Fixed-length Unicode type (number of bytes platform specific); same specification semantics as `string_` (e.g., `'U10'`)

NUMPY ndarray

N-Dimensional Arrays

建立ndarray

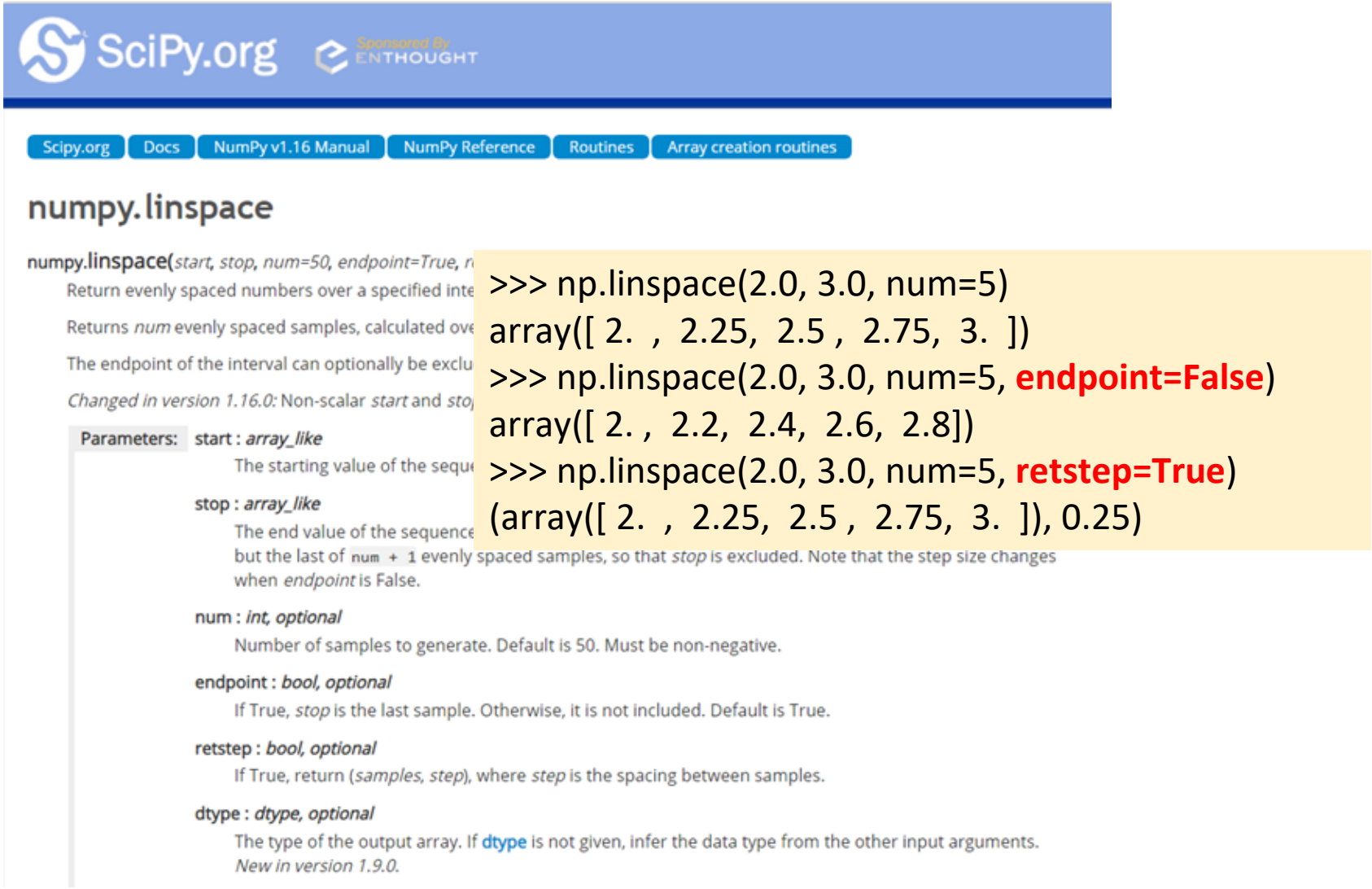
建立array(陣列)

方法	範例
使用Python內建的array()	x = np.**array**([[1,2.0],[0,0],(1+1j,3.)])
使用numpy提供的**創建函數** 直接生成	np.**zeros**((2, 3)) np.**arange**(2, 3, 0.1) # start, end, step np.**linspace**(1., 4., 6) # start, end, num np.**indices**((3, 3))
使用genfromtxt()方法生成	ndtype=[('a',int), ('b', float), ('c', int)] names = ["A", "B", "C"] np.**genfromtxt**("file_name.txt", delimiter=",", names=names, dtype=ndtype, autostrip=True, comments="#", skip_header=3, skip_footer=5, usecols=(0, -1))

https://danzhuibing.github.io/py_numpy_ndarray.html

使用numpy.linspace產生陣列

https://docs.scipy.org/doc/numpy/reference/generated/numpy.linspace.html



create identity matrix

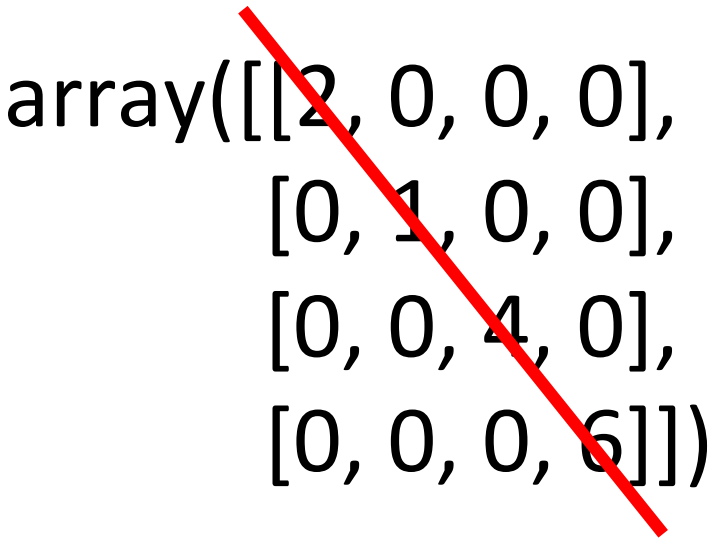
```
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
```

create identity matrix

```
array([[ 1.,  0.,  0.],
       [ 0.,  1.,  0.],
       [ 0.,  0.,  1.]])
```

```
ar9 = np.eye(3);
ar9
```

Create diagonal array



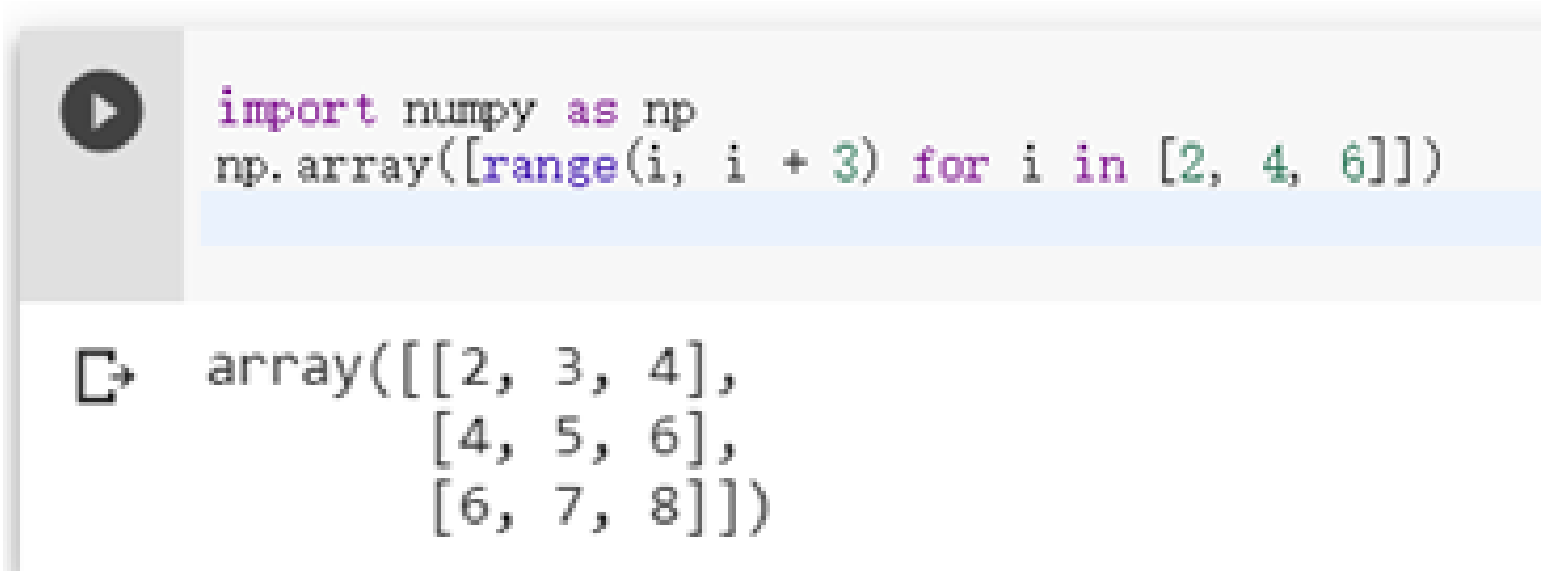
Create diagonal array

```
array([[2, 0, 0, 0],
       [0, 1, 0, 0],
       [0, 0, 4, 0],
       [0, 0, 0, 6]])
```

```
ar10=np.diag((2,1,4,6));
ar10
```

```
import numpy as np
np.array([range(i, i + 3) for i in [2, 4, 6]])
```

import numpy as np
np.**array**([range(i, i + 3) for i in [2, 4, 6]])



SciPy.org Sponsored By ENTHOUGHT

Scipy.org | Docs | NumPy v1.16 Manual | NumPy Reference | Routines | Array manipulation routines

numpy.tile

https://docs.scipy.org/doc/numpy/reference/generated/numpy.tile.html

numpy.tile(*A*, *reps*) [source]

Construct an array by repeating A the number of times given by reps.

If *reps* has length `d`, the result will have dimension of `max(d, A.ndim)`.

If `A.ndim < d`, *A* is promoted to be d-dimensional by prepending new axes. So a shape (3,) array is promoted to (1, 3) for 2-D replication, or shape (1, 1, 3) for 3-D replication. If this is not the desired behavior, promote *A* to d-dimensions manually before calling this function.

If `A.ndim > d`, *reps* is promoted to *A*.ndim by pre-pending 1's to it. Thus for an *A* of shape (2, 3, 4, 5), a *reps* of (2, 2) is treated as (1, 1, 2, 2).

Note : Although tile may be used for broadcasting, i
functions.

Parameters: A : *array_like*
The input array.

reps : *array_like*
The number of repetitions of *A*

Returns: c : *ndarray*
The tiled output array.

```
>>> a = np.array([0, 1, 2])
>>> np.tile(a, 2)
array([0, 1, 2, 0, 1, 2])
>>> np.tile(a, (2, 2))
array([[0, 1, 2, 0, 1, 2],
       [0, 1, 2, 0, 1, 2]])
>>> np.tile(a, (2, 1, 2))
array([[[0, 1, 2, 0, 1, 2]],
       [[0, 1, 2, 0, 1, 2]]])
```

作業:下列答案為何?

```
np.tile(np.array([[1,2],[6,7]]),3)
```

```
np.tile(np.array([[1,2],[6,7]]),(2,2))
```

```
np.tile( np.array([[1,2],[6,7]]),3)
```

```
array([[1, 2, 1, 2, 1, 2],
       [6, 7, 6, 7, 6, 7]])
```

```
np.tile(np.array([[1,2],[6,7]]),(2,2))
```

```
array([[1, 2, 1, 2],
       [6, 7, 6, 7],
       [1, 2, 1, 2],
       [6, 7, 6, 7]])
```

NUMPY ndarray 運算

Array shape manipulation

2*3--->3*2--->6*1

Array shape manipulation::reshape()

```
import numpy as np

x = np.arange(2,10).reshape(2,4)
x

y = np.arange(2,10).reshape(4,2)
y
```

[1]

```
1 x = np.arange(2, 10).reshape(2, 4)
2 x
```

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

代码 文字

[2]

```
1 y = np.arange(2, 10).reshape(4, 2)
2 y
```

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

Array shape manipulation
Flattening and Transpose

```
ar=np.array([np.arange(1,6),np.arange(10,15)]);
ar
```

```
ar.ravel(
)
```

```
ar.T
```

```
ar.T.ravel(
)
```

Adding a dimension

(3,)

```
ar=np.array([14,15,16]);
ar.shape
ar
```

```
array([14, 15, 16])
```

(3, 1)

```
ar=ar[:, np.newaxis];
ar.shape
ar
```

```
array([[14],
       [15],
       [16]])
```

https://github.com/femibyte/mastering_pandas/blob/master/MasteringPandas-chap3_DataStructures.ipynb

NUMPY ndarray

N-Dimensional Arrays

索引()與切片運算(slice)

Array Indexing: Accessing Single Elements

```
Import numpy as np
x = np.arange(2,10)
x
```

x[0]

x[-2]

x[-1]

Array Indexing: Accessing Single Elements

```
ar = np.array([[2,3,4],[9,8,7],[11,12,13]]);
ar
```

ar[1,2] =?

ar[2,:] =?

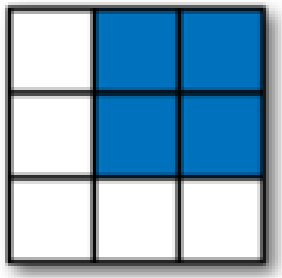
ar[:,1] =?

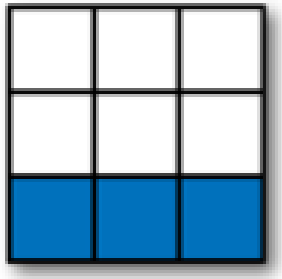
ar[2,**-1**] =?

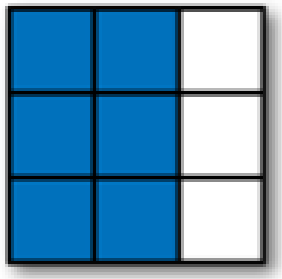
Array slicing陣列的切片運算

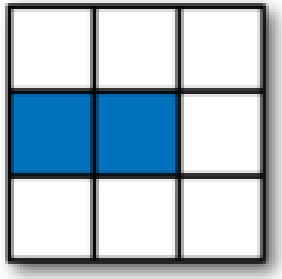
```
ar=2*np.arange(6);
ar
ar[1:5:2]=?
ar[1:6:2]=?
ar[:4]=?
ar[4:] =?
ar[::3]=?
```

```
ar[:3]=1;
ar
ar[2:]=np.ones(4);
ar
```







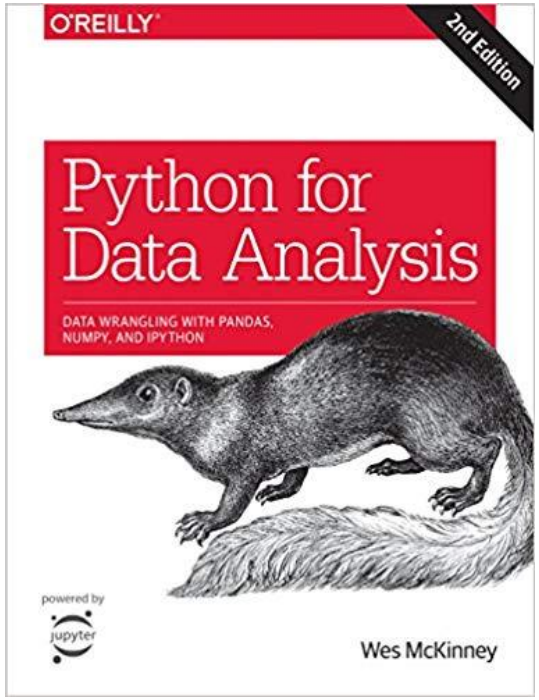


Expression	Shape
`arr[:2, 1:]`	`(2, 2)`
`arr[2]`	`(3,)`
`arr[2, :]`	`(3,)`
`arr[2:, :]`	`(1, 3)`
`arr[:, :2]`	`(3, 2)`
`arr[1, :2]`	`(2,)`
`arr[1:2, :2]`	`(1, 2)`

Fancy Indexing(花式索引)

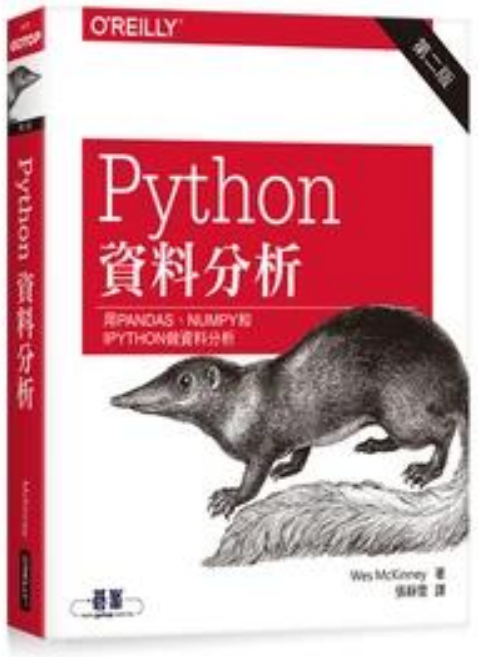
通過整數陣列來索引

Chapter 4: NumPy Basics: Arrays and Vectorized Computation




```
arr = np.empty((8, 4))
for i in range(8):
    arr[i] = i
arr
```

好書推薦



Python for Data Analysis
Wes McKinney
O'Reilly
https://github.com/wesm/pydata-book

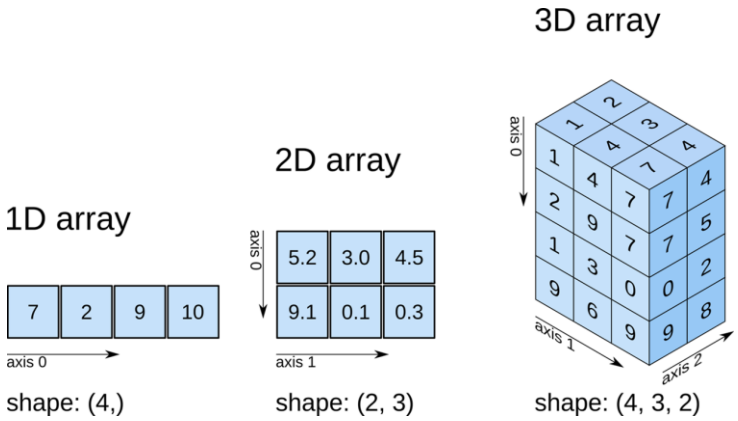
NUMPY ndarray

N-Dimensional Arrays

Reduction Operations

Reduction Operations

```
ar=np.arange(1,5)
ar.prod()
```




```
ar=np.array([np.arange(1,6),np.arange(1,6)]);
ar
```

np.prod(ar,axis=0)=?

np.prod(ar,axis=1)=?

Reduction Operations

```
ar=np.array([[2,3,4],[5,6,7],[8,9,10]]);
ar.sum()
ar.mean()
np.median(ar)
```

NUMPY ndarray 運算

Universal Functions:

Fast Element-Wise Array Functions

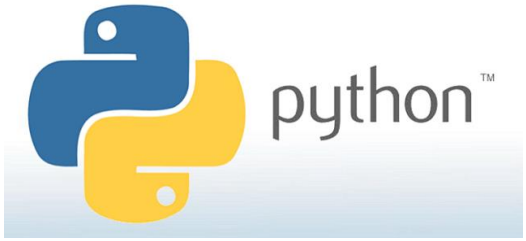
使用加快速度運算的 Universal Functions

計算0的三次方到999的三次方

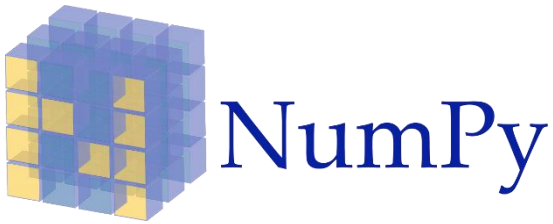
For loop VS **Vectorization** 向量化計算

NumPy vs Python:看看誰比較快

For loop




```
ar=range(1000)
%timeit [ar[i]**3 for i in ar]
```




```
ar=np.arange(1000)
%timeit ar**3
```

Vectorization

向量化計算

更多範例

```
import numpy as np
arr = np.arange(10)
arr
```

np.sqrt(arr)

np.exp(arr)

Table 4-3. Unary ufuncs

Function	Description
`abs, fabs`	Compute the absolute value element-wise for integer, floating-point, or complex values
`sqrt`	Compute the square root of each element (equivalent to `arr ** 0.5`)
`square`	Compute the square of each element (equivalent to `arr ** 2`)
`exp`	Compute the exponent e^x of each element
`log, log10, log2, log1p`	Natural logarithm (base *e*), log base 10, log base 2, and log(1 + x), respectively
`sign`	Compute the sign of each element: 1 (positive), 0 (zero), or –1 (negative)
`ceil`	Compute the ceiling of each element (i.e., the smallest integer greater than or equal to that number)
`floor`	Compute the floor of each element (i.e., the largest integer less than or equal to each element)
`rint`	Round elements to the nearest integer, preserving the `dtype`
`modf`	Return fractional and integral parts of array as a separate array
`isnan`	Return boolean array indicating whether each value is `NaN` (Not a Number)
`isfinite, isinf`	Return boolean array indicating whether each element is finite (non-`inf`, non-`NaN`) or infinite, respectively
`cos, cosh, sin, sinh, tan, tanh`	Regular and hyperbolic trigonometric functions
`arccos, arccosh, arcsin, arcsinh, arctan, arctanh`	Inverse trigonometric functions
`logical_not`	Compute truth value of `not x` element-wise (equivalent to `~arr`).

Table 4-4. Binary universal functions

Function	Description
`add`	Add corresponding elements in arrays
`subtract`	Subtract elements in second array from first array
`multiply`	Multiply array elements
`divide, floor_divide`	Divide or floor divide (truncating the remainder)
`power`	Raise elements in first array to powers indicated in second array
`maximum, fmax`	Element-wise maximum; `fmax` ignores `NaN`
`minimum, fmin`	Element-wise minimum; `fmin` ignores `NaN`
`mod`	Element-wise modulus (remainder of division)
`copysign`	Copy sign of values in second argument to values in first argument

Function	Description
`greater, greater_equal, less, less_equal, equal, not_equal`	Perform element-wise comparison, yielding boolean array (equivalent to infix operators `>, >=, <, <=, ==, !=`)
`logical_and, logical_or, logical_xor`	Compute element-wise truth value of logical operation (equivalent to infix operators `& | ^`)

NUMPY ndarray 運算

A矩陣與B矩陣間的運算

任何兩個大小相等的陣列之間的運算，都是element-wise(元素對元素)

```
arr = np.array([[1., 2., 3.], [4., 5., 6.]])
arr
```

```
arr+arr
arr-arr
arr*arr
1/arr
arr ** 0.5
```

Array multiplication is **element wise**

```
ar=np.array([[1,1],[1,1]]);
ar2=np.array([[2,2],[2,2]]);
ar*ar2

ar.dot(ar2
)
```

Expressing Conditional Logic as Array Operations

```
xarr = np.array([1.1, 1.2, 1.3, 1.4, 1.5])
yarr = np.array([2.1, 2.2, 2.3, 2.4, 2.5])
cond = np.array([True, False, True, True,
False])
  result = [(x if c else y)
            for x, y, c in zip(xarr, yarr, cond)]

  result
```

[1.1, 2.2, 1.3, 1.4, 2.5]

```
result = np.where(cond, xarr, yarr)
result
```

NUMPY ndarray 運算

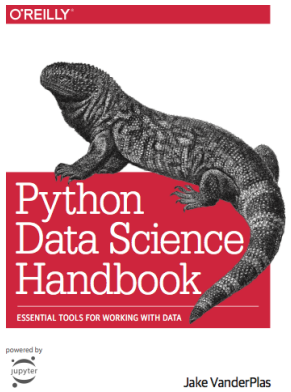
A矩陣與B矩陣間的運算

Broadcasting(廣播機制)

```
x1 = np.arange(9.0).reshape((3,
3))
x2 = np.arange(3.0)
np.multiply(x1, x2)
```

Broadcasting

https://jakevdp.github.io/PythonDataScienceHandbook/02.05-computation-on-arrays-broadcasting.html



$\texttt{np.arange(3)} + 5$

0	1	2	+	5	5	5	=	5	6	7	

$\texttt{np.ones((3, 3))} + \texttt{np.arange(3)}$

1	1	1	+	0	1	2	=	1	2	3	
1	1	1		0	1	2		1	2	3	
1	1	1		0	1	2		1	2	3	

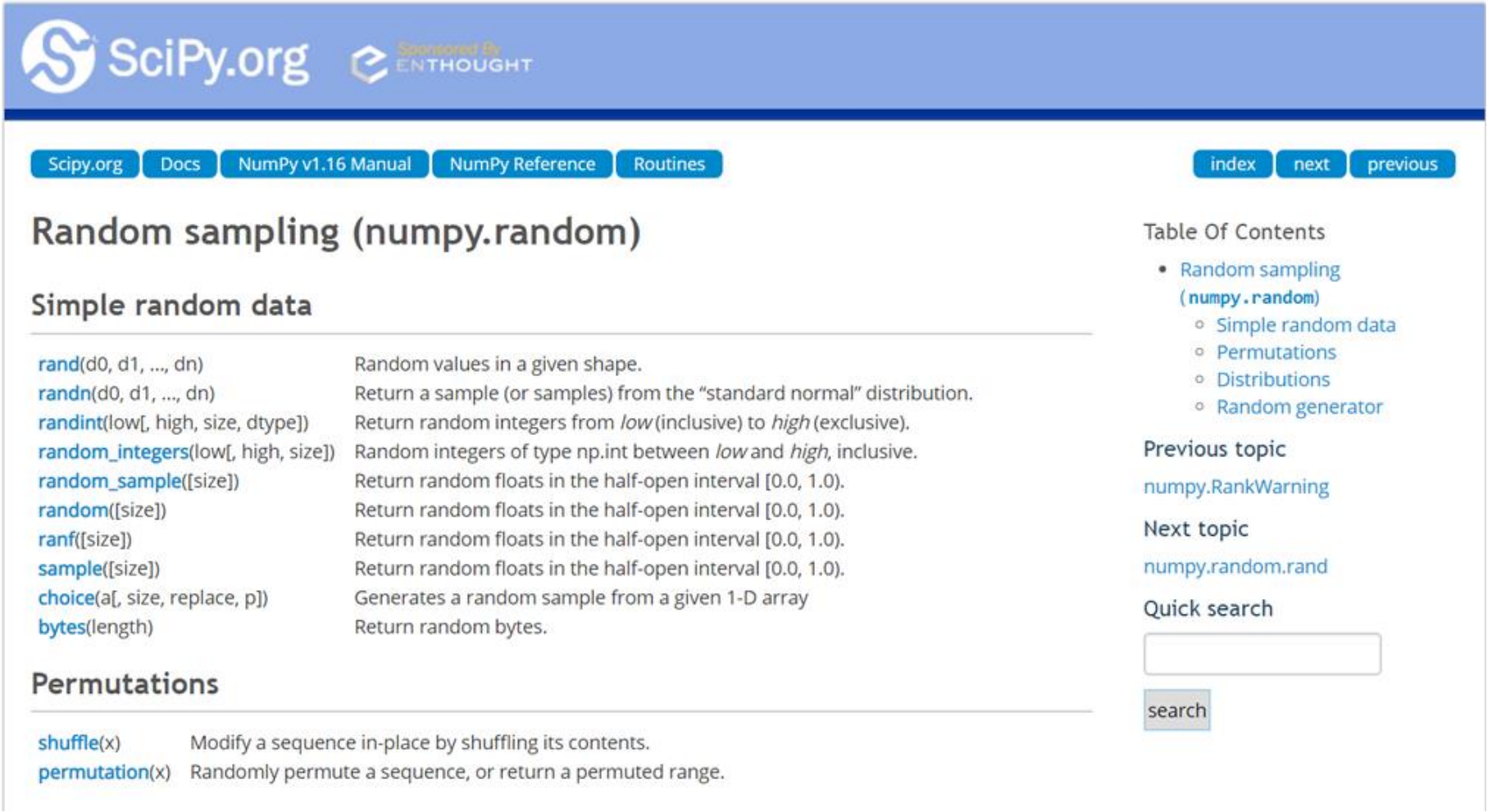
$\texttt{np.arange(3).reshape((3, 1))} + \texttt{np.arange(3)}$

0	0	0	+	0	1	2	=	0	1	2	
1	1	1		0	1	2		1	2	3	
2	2	2		0	1	2		2	3	4	

使用
Numpy random模組
產生隨機資料

https://docs.scipy.org/doc/numpy/reference/routines.random.html

Numpy random模組有許多函數,本課程介紹基礎且最常用的函數



建立array(陣列)

```
import numpy as np
np.random.seed(0)

x1 = np.random.randint(10, size=6)
x2 = np.random.randint(10, size=(3, 4))
x3 = np.random.randint(10, size=(3, 4, 5))
```

One-dimensional array

Two-dimensional array

Three-dimensional array

```
x1 = np.random.randint(10, size=6)

        array([5, 0, 3, 3, 7, 9])

x2 = np.random.randint(10, size=(3, 4))


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

numpy.random.randint()

numpy.random.randint(low, high=None, size=None, dtype='l')

- 返回隨機**整數**，範圍區間為[low,high），包含low，不包含high參數
- low為最小值，high為最大值，size為陣列維度大小，dtype為資料類型，預設的資料類型是np.int
- high沒有填寫時，預設生成亂數的範圍是[0，low)

```
np.random.randint(1,size=5)   # 返回[0,1)之間的整數，所以只有0
                              #  array([0, 0, 0, 0, 0])

np.random.randint(1,5)        # 返回1個[1,5)間的隨機整數

np.random.randint(-5,5,size=(2,2))
                              # array([[ 2, -1],
                                      [ 2,  0]])
```

numpy.random.rand()

numpy.random.rand(d0,d1,...,dn)

- rand函數根據給定維度生成[0,1)之間的資料，包含0，不包含1
- dn表格每個維度
- 返回值為指定維度的array

np.random.rand(4,2)

```
array([[ 0.02173903,  0.44376568],
       [ 0.25309942,  0.85259262],
       [ 0.56465709,  0.95135013],
       [ 0.14145746,  0.55389458]])
```

numpy.random.seed()

作用：使得亂數據可預測。
當我們設置相同的seed，每次生成的亂數相同。
如果不設置seed，則每次會生成不同的亂數

np.random.seed(0)
np.random.rand(5)

→ array([0.5488135 , 0.71518937, 0.60276338, 0.54488318, 0.4236548])

np.random.seed(1676)
np.random.rand(5)

→ array([0.39983389, 0.29426895, 0.89541728, 0.71807369, 0.3531823])

np.random.seed(1676)
np.random.rand(5) →

$$(f * g)(x) = \int_{\mathbf{R}^d} f(y)g(x-y)\,dy = \int_{\mathbf{R}^d} f(x-y)g(y)\,dy,$$

NUMPY ndarray 運算

A矩陣與B矩陣間的convolute運算

https://en.wikipedia.org/wiki/Convolution

https://www.numpy.org/devdocs/reference/generated/numpy.convolve.html

numpy函數中的卷積函式程式庫

numpy.convolve(a, v, mode='full')

a:(N,)輸入的一維陣列
b:(M,)輸入的第二個一維陣列
mode:{'full', 'valid', 'same'}

參數可選　　'

full'　預設值，返回每一個卷積值，長度是N+M-1,在卷積的邊緣處，信號不重疊，存在邊際效應。

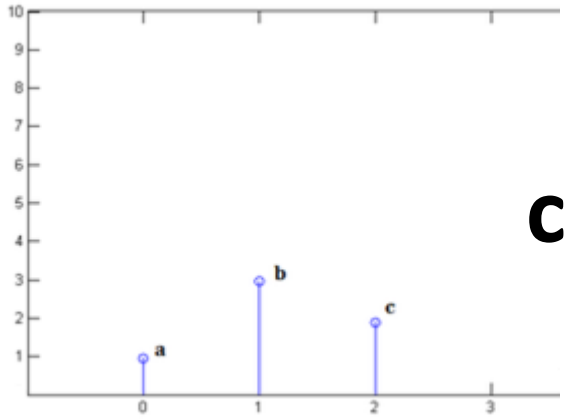
'same'　返回的陣列長度為max(M, N),邊際效應依舊存在。

'valid'　返回的陣列長度為max(M,N)-min(M,N)+1,此時返回的是完全重疊的點。邊緣的點無效。

https://blog.csdn.net/u011599639/article/details/76254442

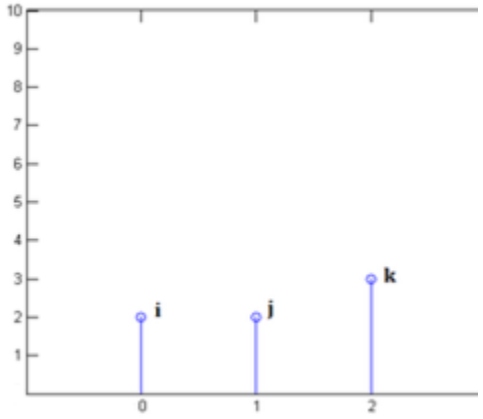
numpy.convolve(x, y, mode='full')

x[0] = a,x[1]=b,x[2]=c

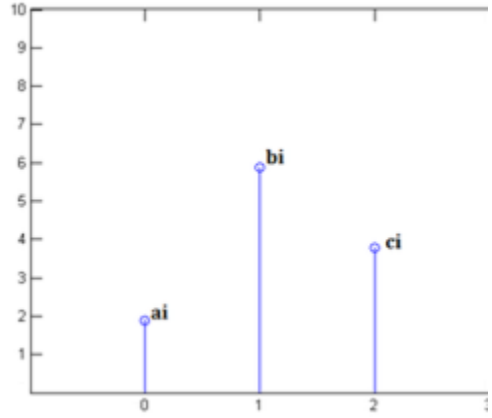


convolve

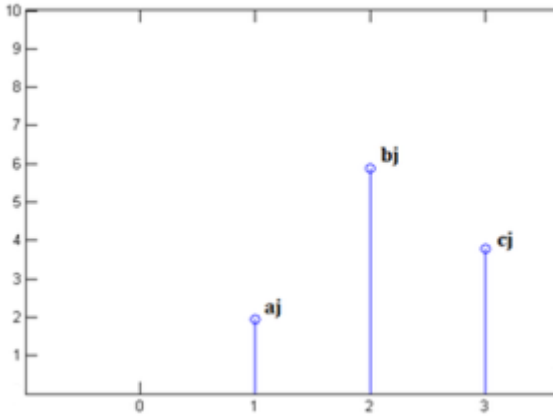
y[0]=i,y[1]=j,y[2]=k



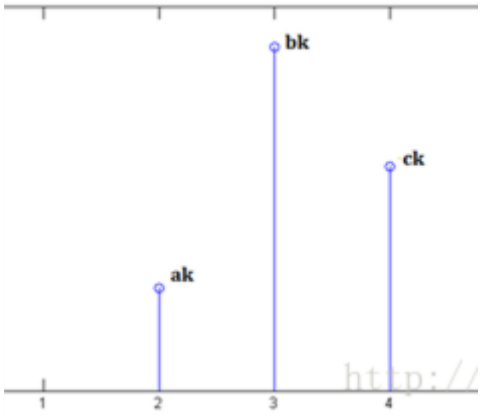
x[n]乘以y[0]并平移到位置0



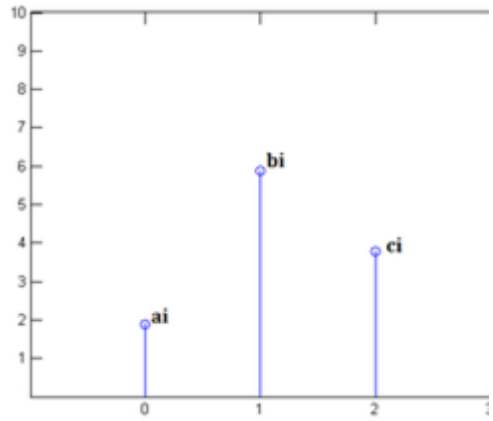
x[n]乘以y[1]并平移到位置1



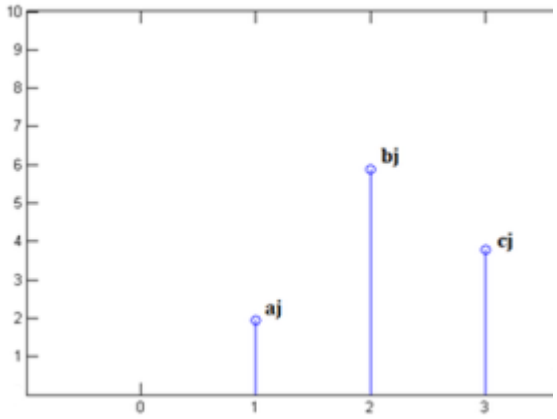
x[n]乘以y[2]并平移到位置2



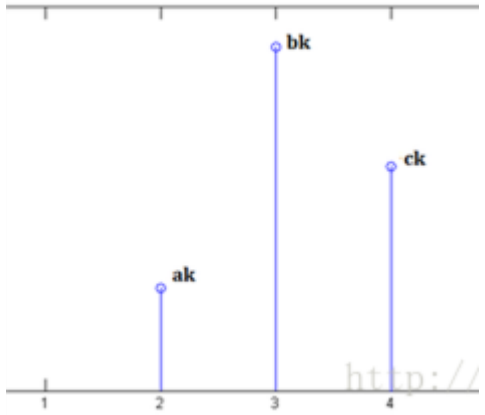
x[n]乘以y[0]并平移到位置0



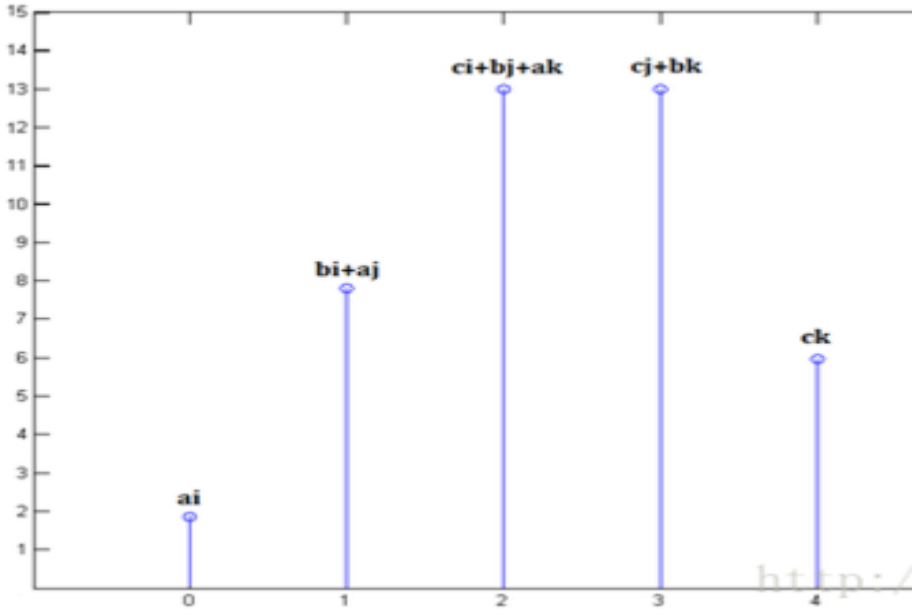
x[n]乘以y[1]并平移到位置1



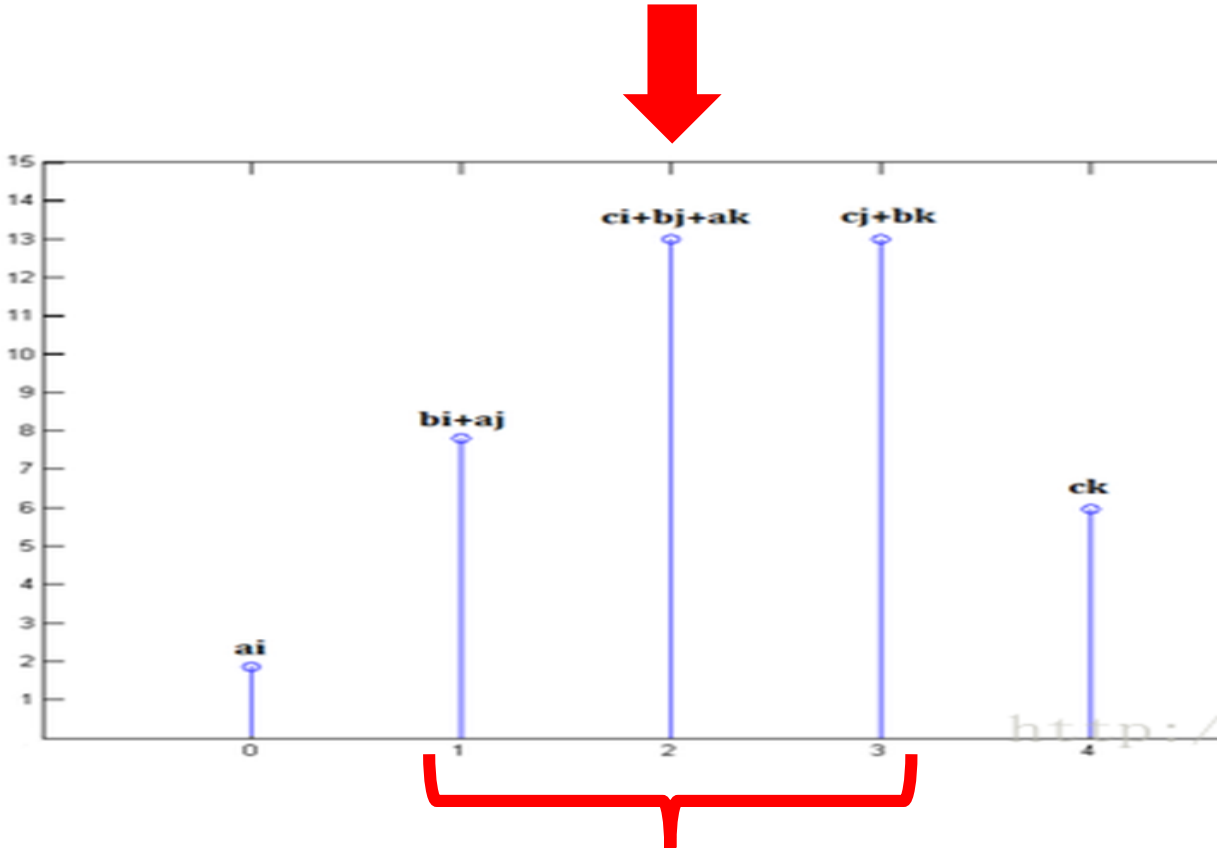
x[n]乘以y[2]并平移到位置2



三個圖疊加



Valid==返回的陣列長度為max(M,N)-min(M,N)+1,此時返回的是完全重疊的點。邊緣的點無效。



Same==>返回的陣列長度為max(M, N),邊際效應依舊存在

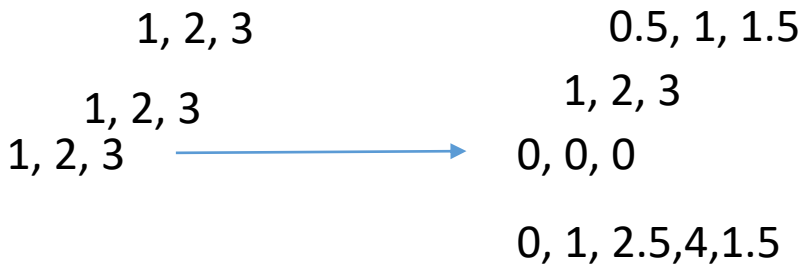
Full===預設值，返回每一個卷積值，長度是N+M-1,在卷積的邊緣處，信號不重疊，存在邊際效應

```
import numpy as np
np.convolve([1, 2, 3], [0, 1, 0.5])
```

Full===預設值

```
np.convolve([1,2,3],[0,1,0.5], 'same')

np.convolve([1,2,3],[0,1,0.5], 'valid')
```

```
import numpy as np
np.convolve([1, 2, 3], [0, 1, 0.5])
```



2.

Matplotlib
Data Visualization
資料視覺化

Data Visualization
資料視覺化

Data Visualization
資料視覺化

藉助於**圖形化手段**，
清晰有效地傳達與溝通訊息

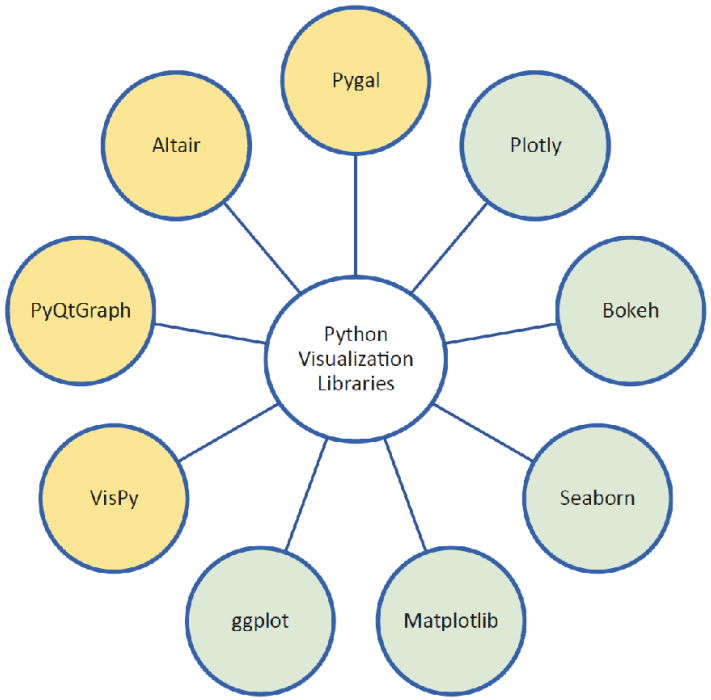
https://zh.wikipedia.org/wiki/資料視覺化

Data Visualization
資料視覺化

藉助於圖形化手段，
清晰有效地傳達與溝通訊息

Data Visualization
資料視覺化の有許多套件
請挑選你熟悉的….深入學習

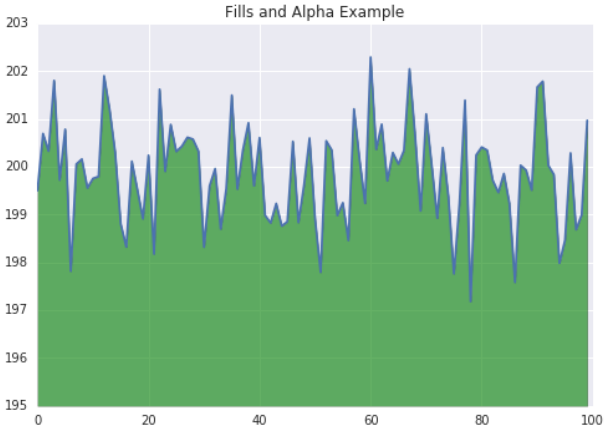
- **Matplotlib(本課程使用)**
- Seaborn
- Ggplot
- Bokeh
- Pyga
- Plotly

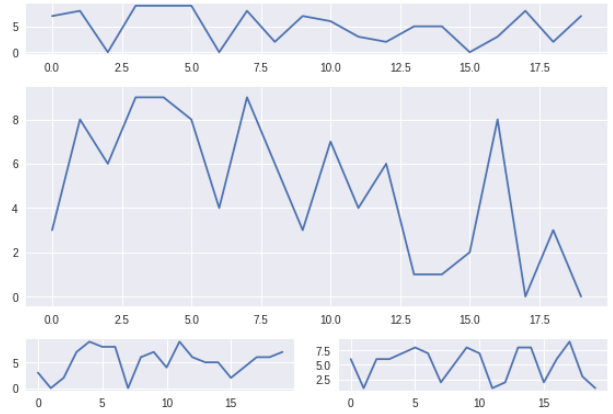


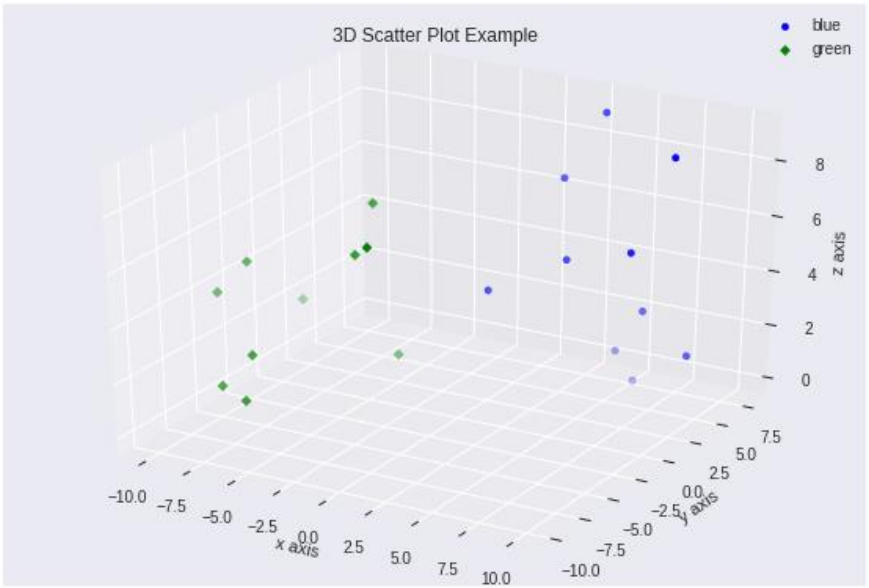
Google Colab有許多範利可以提供你自我學習

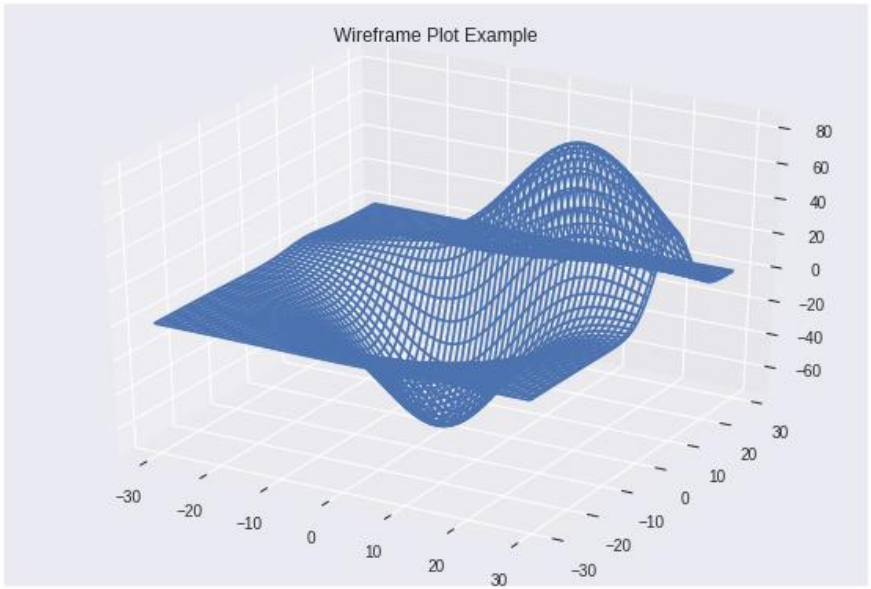
Charting in Colaboratory

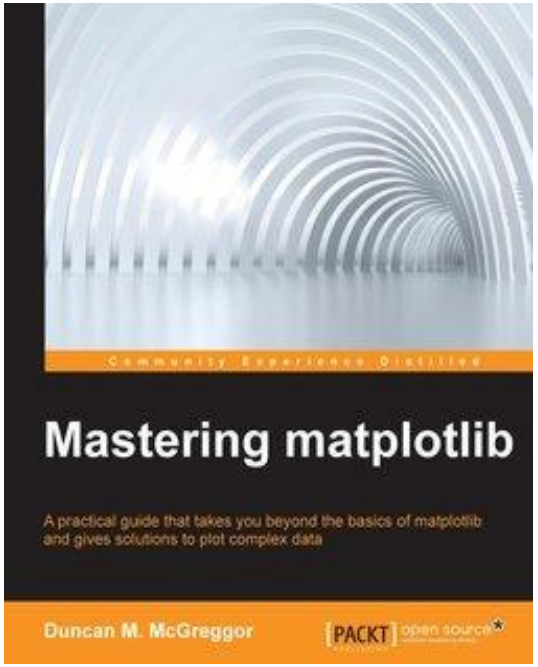
https://colab.research.google.com/notebooks/charts.ipynb

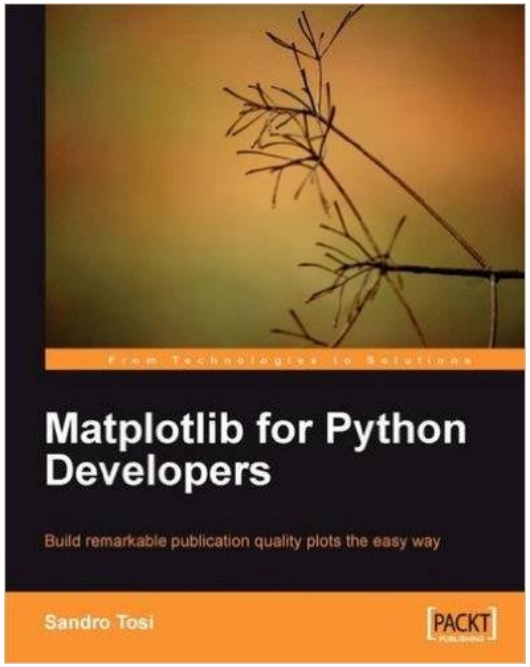


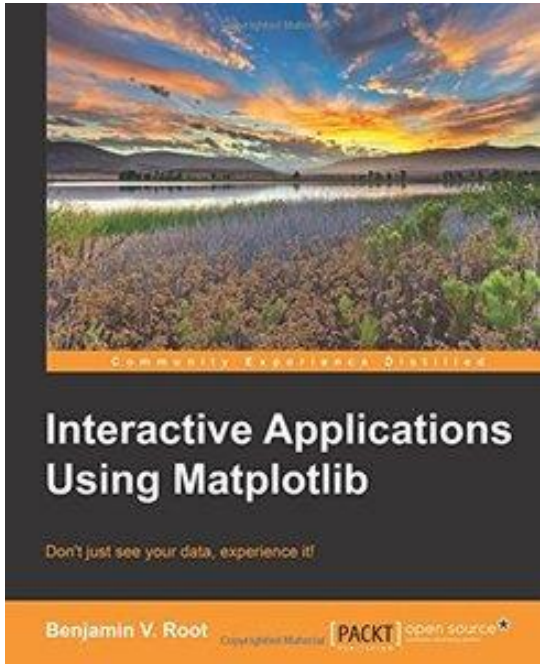


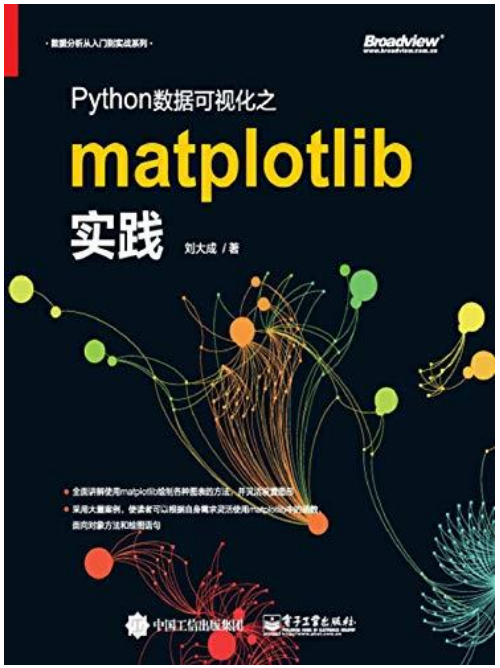





Community Experience Distilled
Mastering matplotlib
A practical guide that takes you beyond the basics of matplotlib and gives solutions to plot complex data
Duncan M. McGreggor
PACKT open source


From Technologies to Solutions
Matplotlib for Python Developers
Build remarkable publication quality plots the easy way
Sandro Tosi
PACKT


Community Experience Distilled
Interactive Applications Using Matplotlib
Don't just see your data, experience it!
Benjamin V. Root
PACKT open source


Broadview
Python数据可视化之
matplotlib
实践
刘大成 / 著

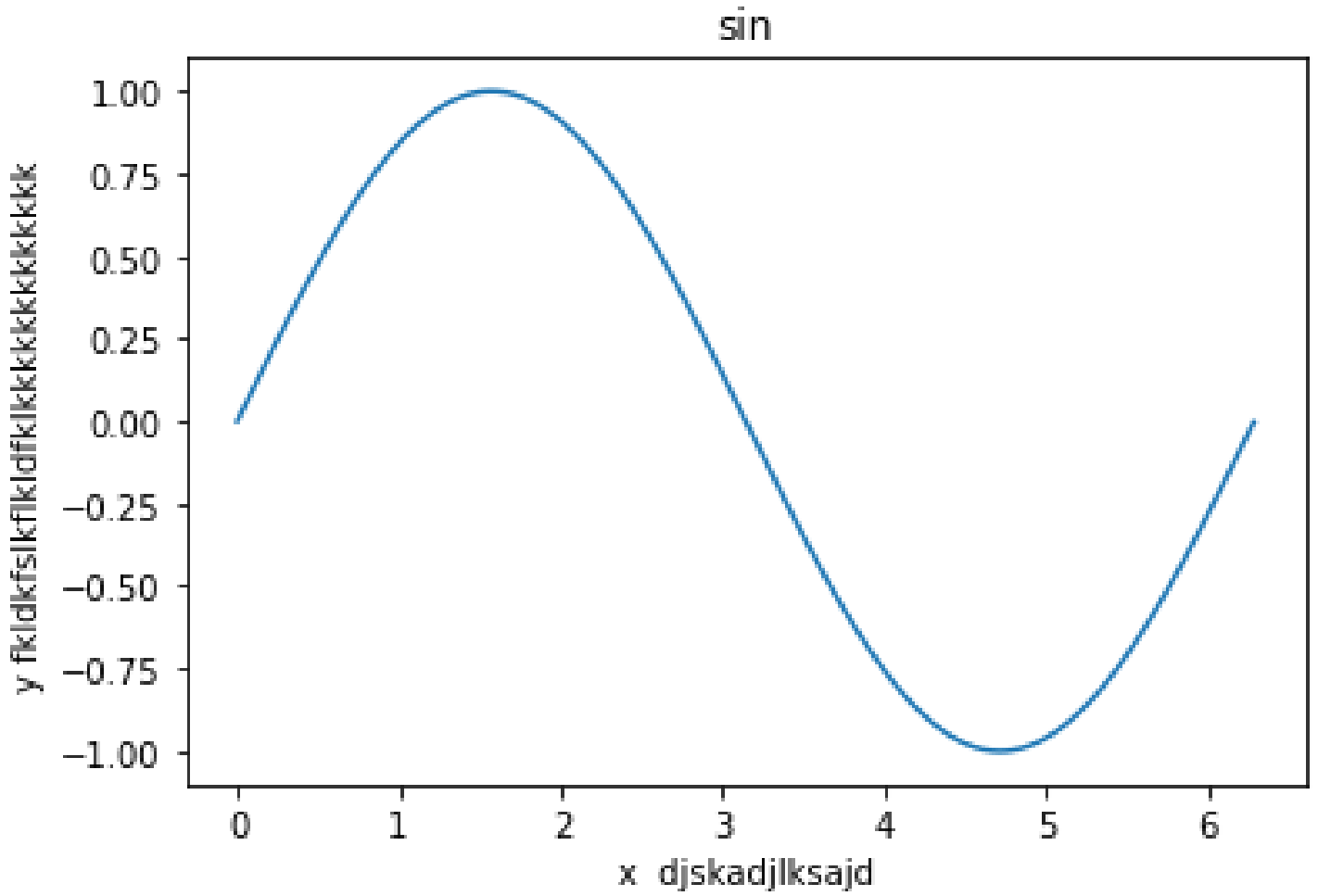
Data Visualization
資料視覺化の案例學習

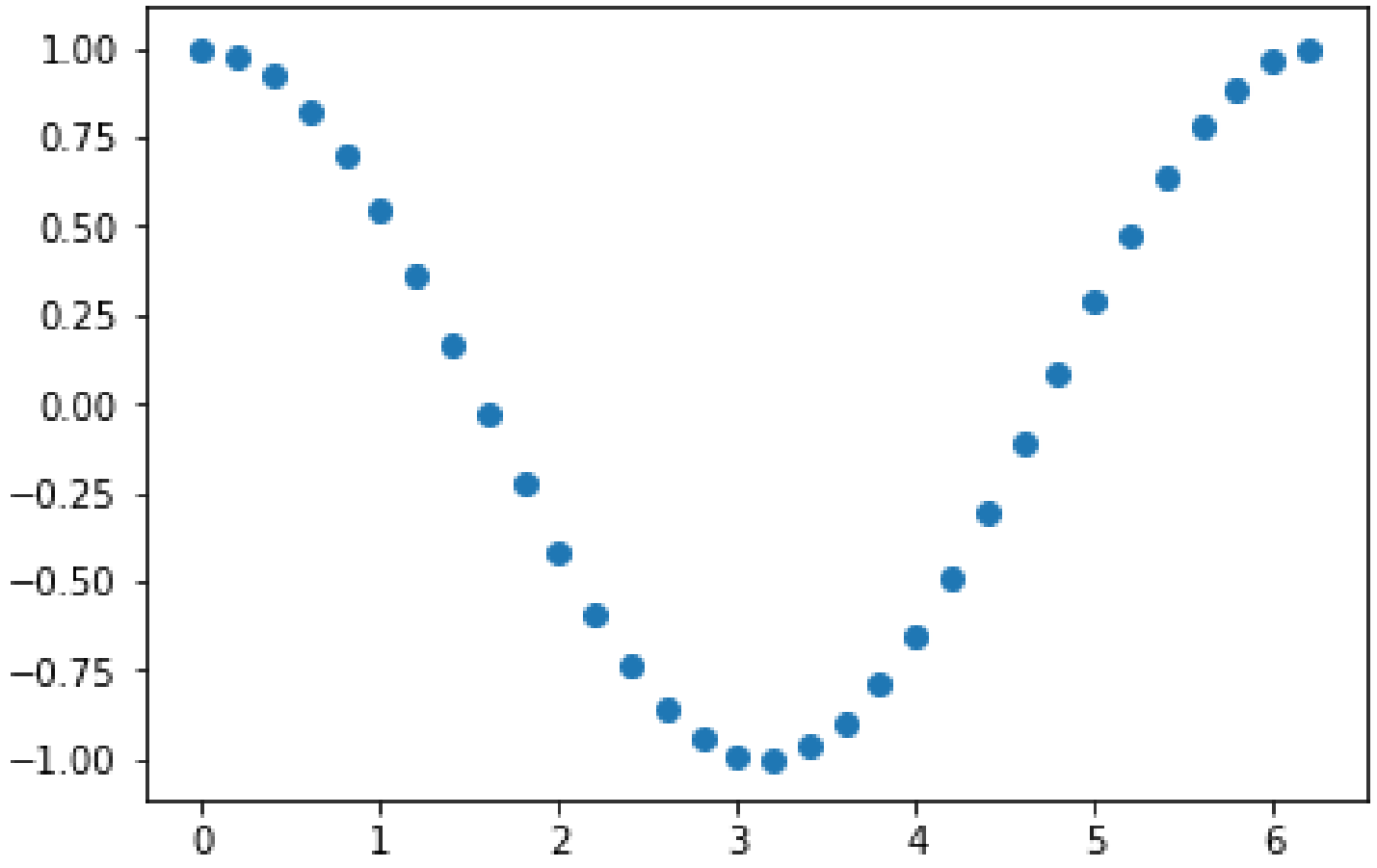
github

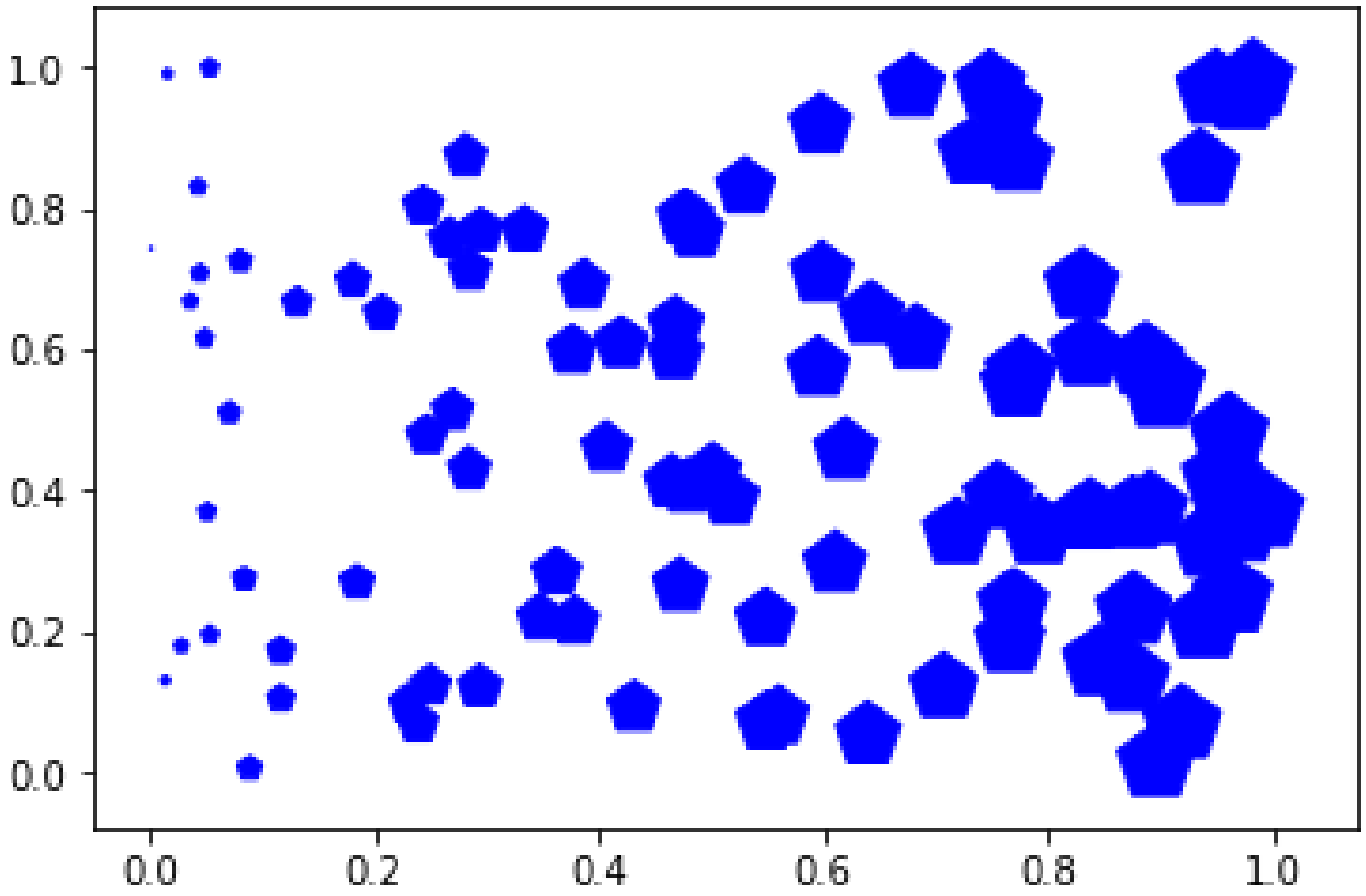
mydear**great**teache
r

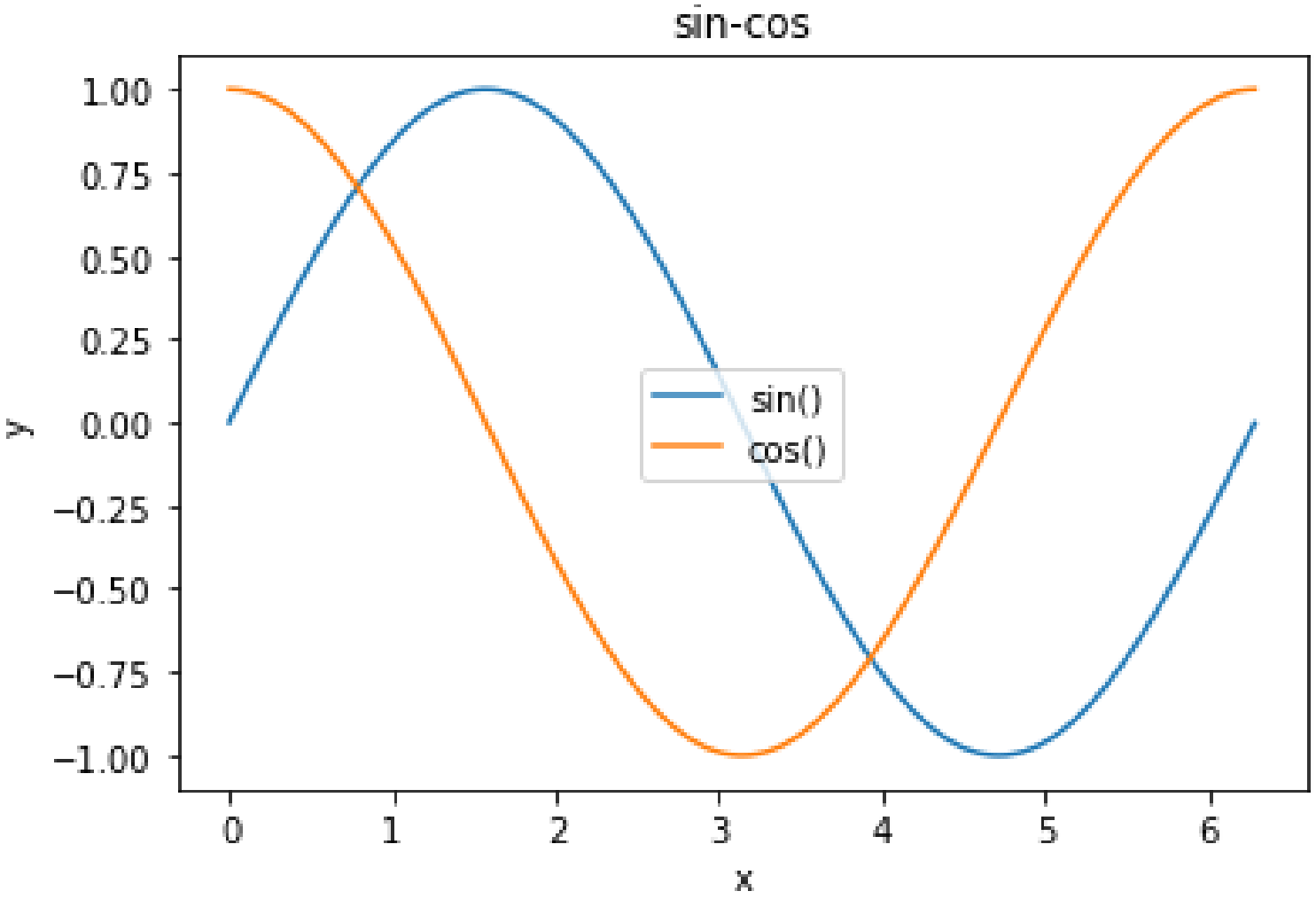
uTaipei2019

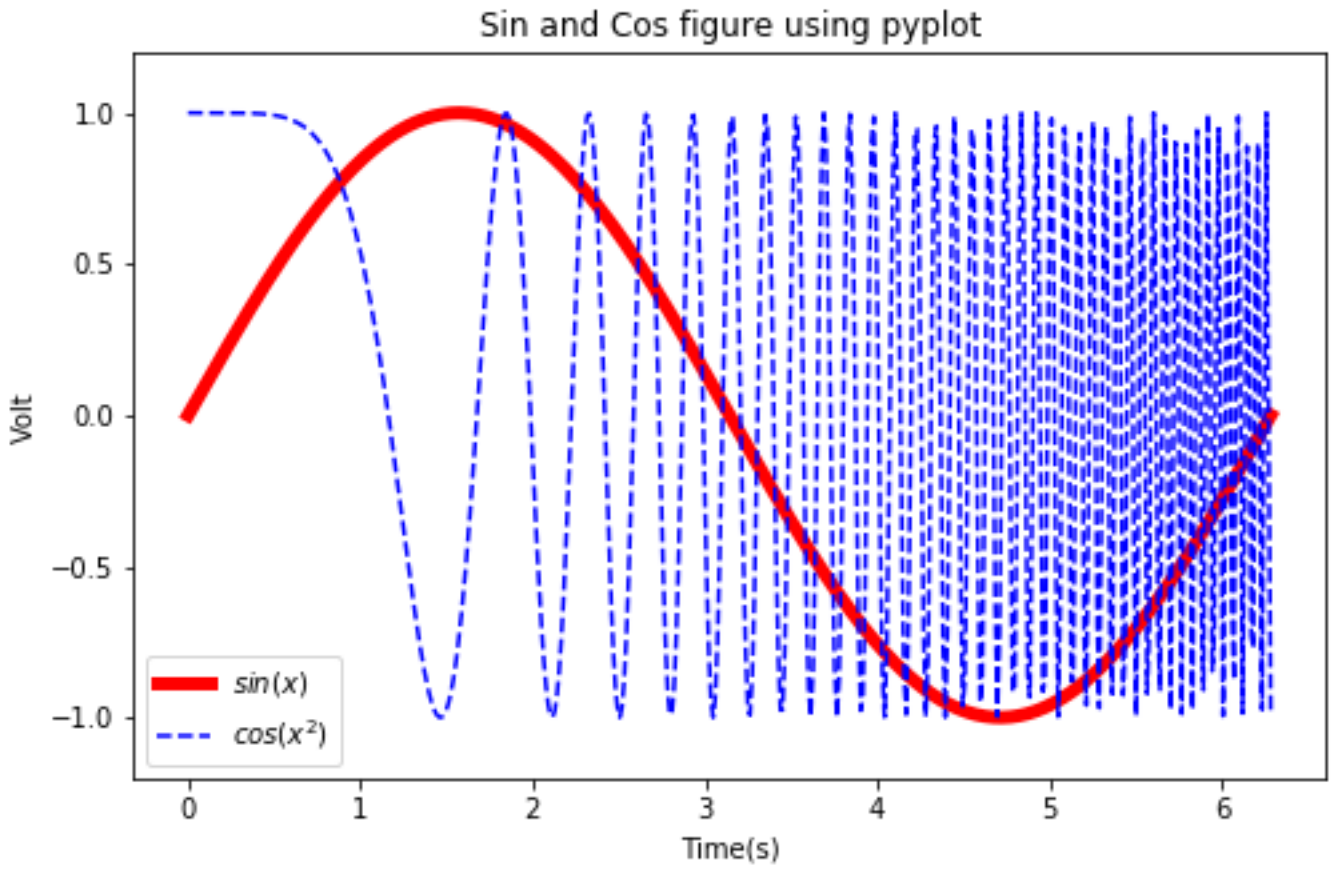
1_2_Matplotlib範例學習快速入門.ipynb

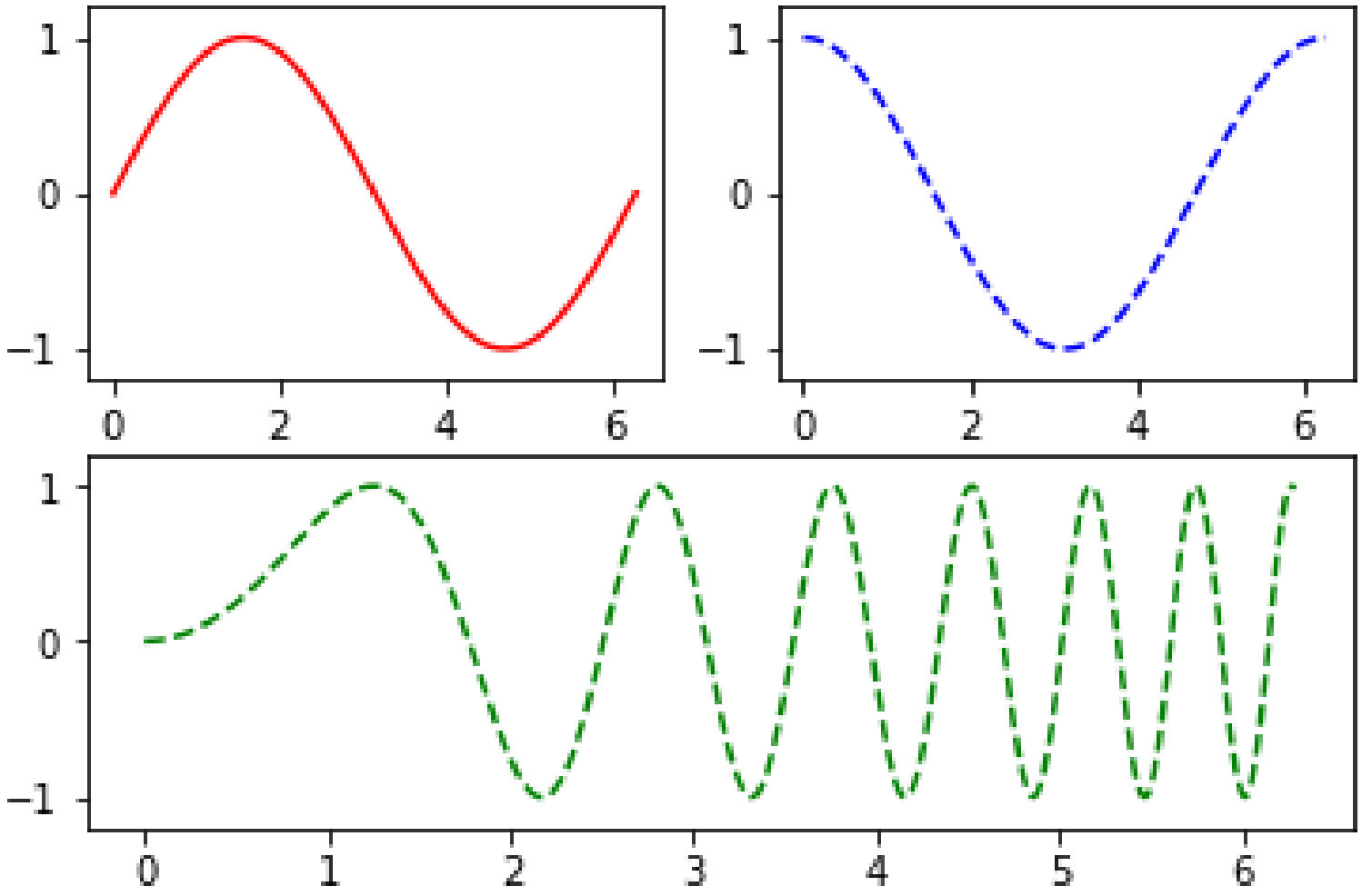

sin
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
y fkldkfslkflkldfklkkkkkkkkkkkk
0
1
2
3
4
5
6
x djskadjlksajd


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
1
2
3
4
5
6


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


sin-cos
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
y
0
1
2
3
4
5
6
x
sin()
cos()

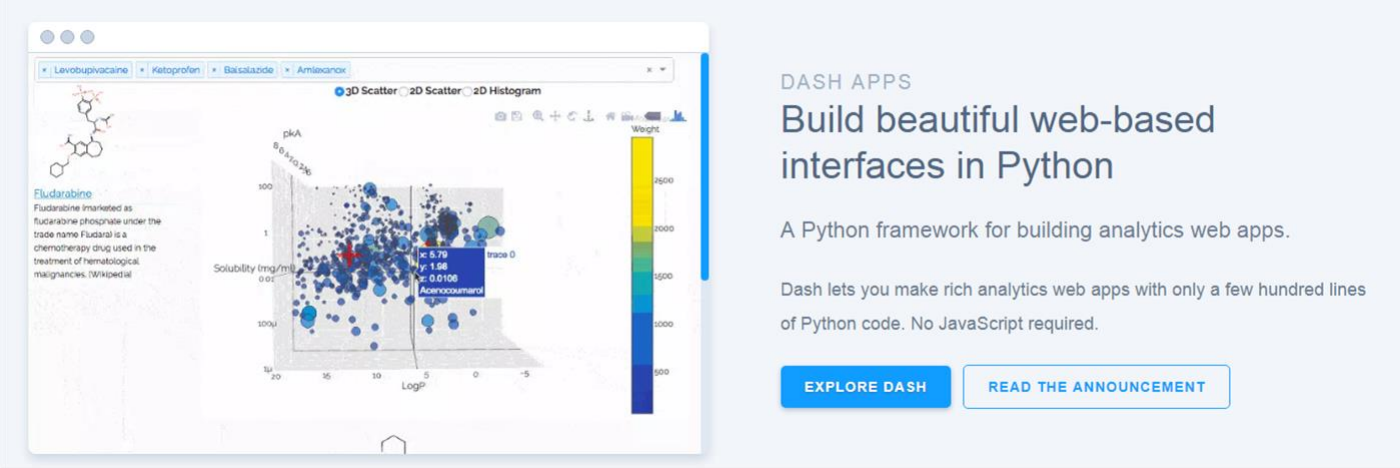

Sin and Cos figure using pyplot
1.0
0.5
0.0
−0.5
−1.0
Volt
0
1
2
3
4
5
6
Time(s)
sin(x)
cos(x²)

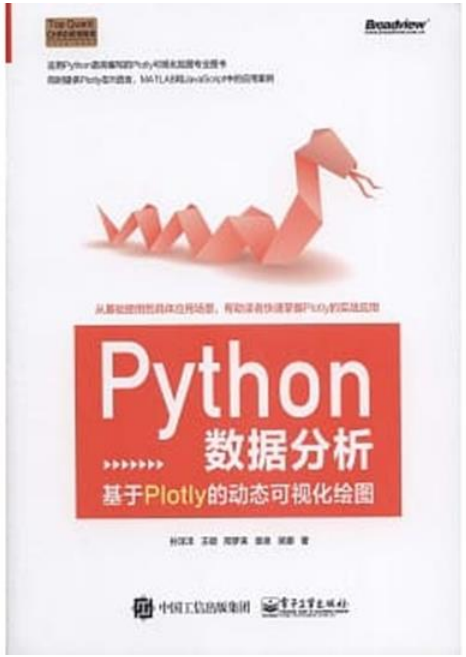

1
0
−1
0
2
4
6
1
0
−1
0
2
4
6
1
0
−1
0
1
2
3
4
5
6

延伸學習
Data Visualization
資料視覺化の各種套件

作業

到官方網址https://plot.ly/看看互動式資料視覺化成果





延伸閱讀:推薦的教科書

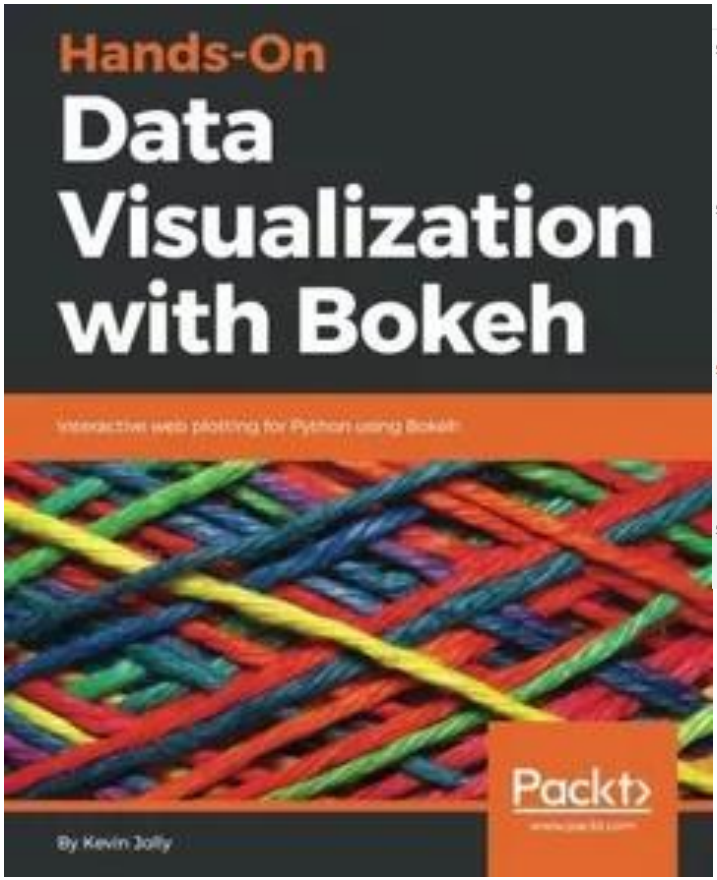
Python數據分析：基於Plotly的動態可視化繪圖

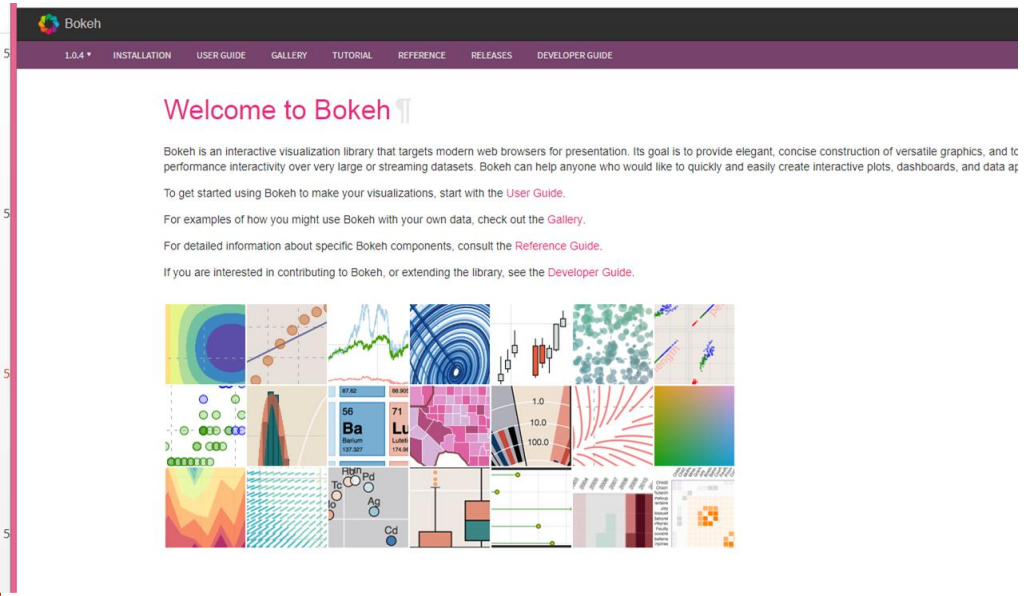
作者： 孫洋洋, 王碩, 邢夢來, 袁泉, 吳娜

電子工業出版社

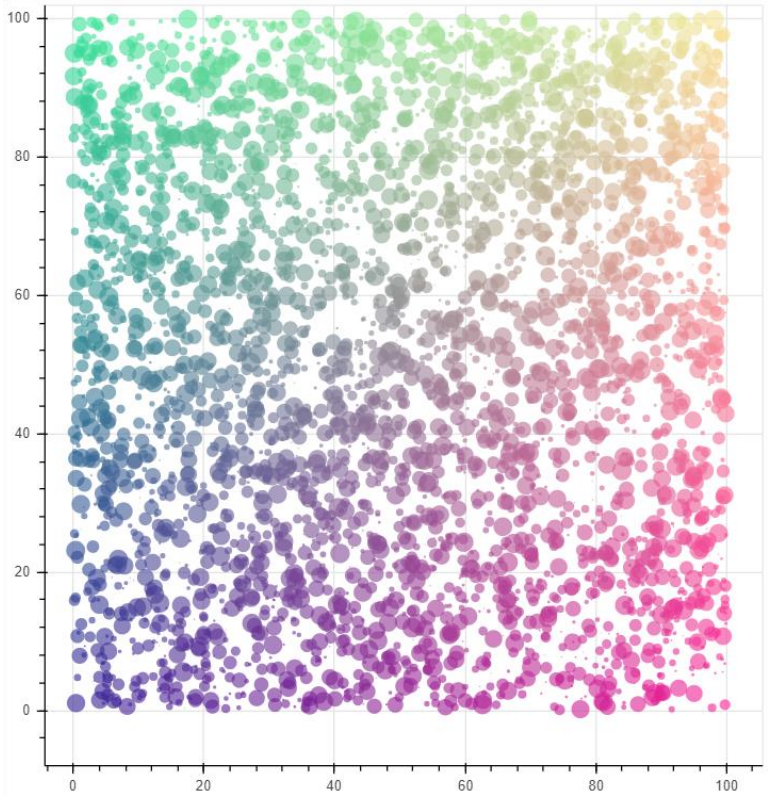
https://github.com/sunshe35/PythonPlotlyCodes

到官方網址https://bokeh.pydata.org/en/latest/
看看互動式資料視覺化成果





!pip install bokeh

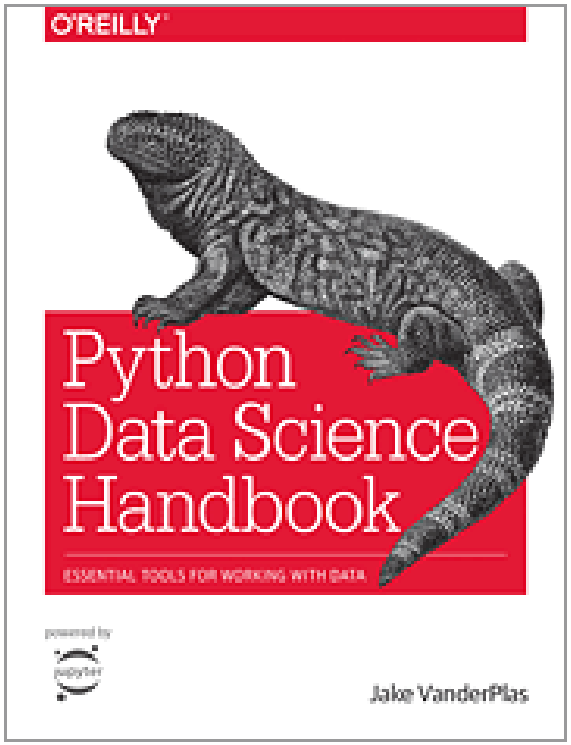


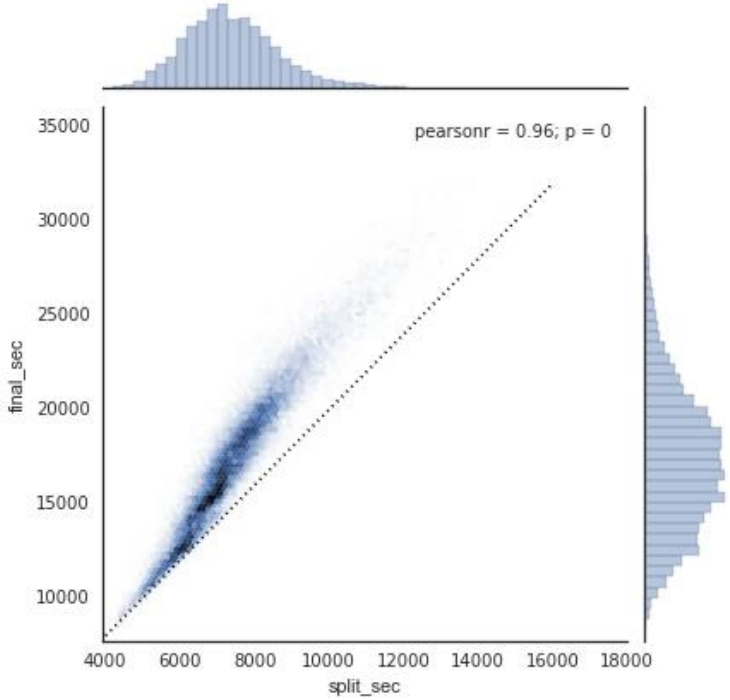

```
import numpy as np
from bokeh.plotting import figure, show
from bokeh.io import output_notebook

N = 4000

x = np.random.random(size=N) * 100
y = np.random.random(size=N) * 100
radii = np.random.random(size=N) * 1.5
colors = ["#%02x%02x%02x" % (r, g, 150) for r, g in zip(np.floor(50+2*x).astype(int),
np.floor(30+2*y).astype(int))]

output_notebook()
p = figure()
p.circle(x, y, radius=radii, fill_color=colors, fill_alpha=0.6, line_color=None)
show(p)
```

https://colab.research.google.com/notebooks/charts.ipynb

在Google Colab學習seaborn

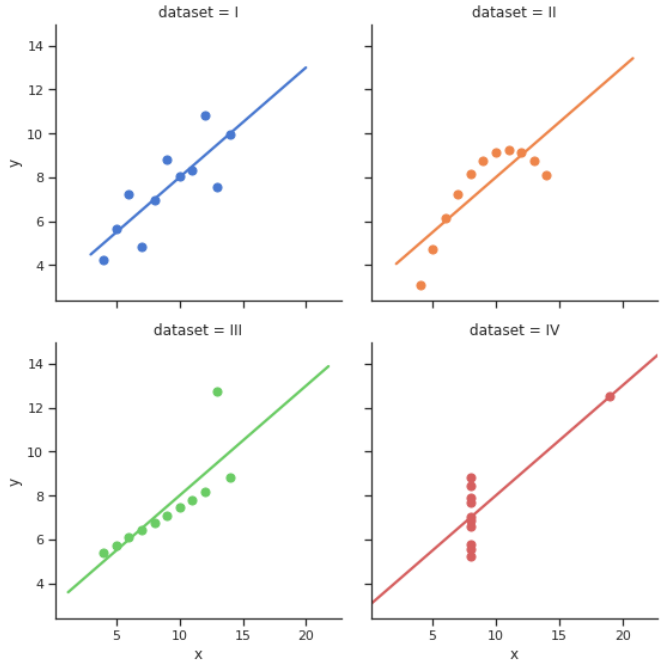
```
!pip install seaborn==0.9.0
```





範例學習1:
https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/04.14-Visualization-With-Seaborn.ipynb

範例學習2:
https://colab.research.google.com/drive/1o6MijFkNHiTPeS8Y5n59j2cH4-Mf2wX3

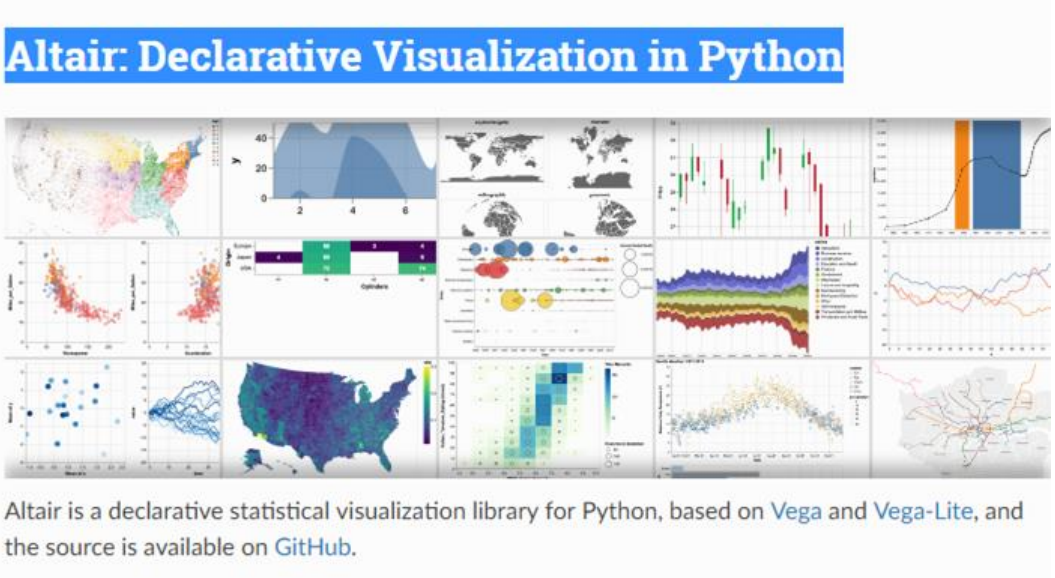



```
import seaborn as sns
sns.set(style="ticks")

# Load the example dataset for Anscombe's quartet
df = sns.load_dataset("anscombe")

# Show the results of a linear regression within each dataset
sns.lmplot(x="x", y="y", col="dataset", hue="dataset", data=df,
           col_wrap=2, ci=None, palette="muted", height=4,
           scatter_kws={"s": 50, "alpha": 1});
```

https://www.data-insights.cn/?p=179

Altair: Declarative Visualization in Python

到官方網址https://altair-viz.github.io/
看看資料視覺化成果




```
import altair as alt
from vega_datasets import data
cars = data.cars()

alt.Chart(cars).mark_point().encode(
    x='Horsepower',
    y='Miles_per_Gallon',
    color='Origin',
).interactive()
```