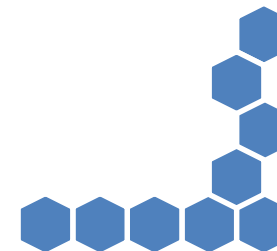


「智能理財與深度學習」訓練營

張景堯 老師



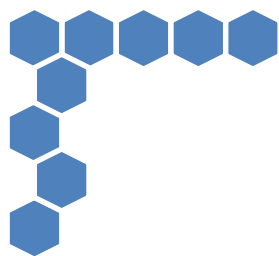
國立政治大學金融科技研究中心
智能理財與深度學習暑期訓練營



第一周課程內容

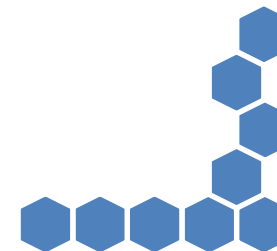
Outline

- Python 環境的安裝 (Anaconda + Python + TensorFlow + Keras)
- 利用 Jupyter Notebook 編輯程式
- Python 基礎
- Python 套件
 - Numpy 與 tensor 計算概念
 - Pandas, Matplotlib 資料整理與繪圖
- 利用 Linear Regression 做預測



Python 環境安裝

Creating Python environment

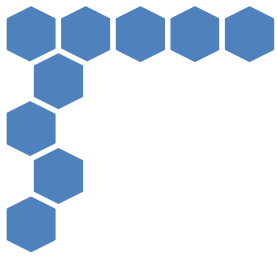


國立政治大學金融科技研究中心
智能理財與深度學習暑期訓練營

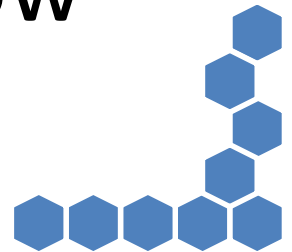


要安裝什麼

What we need

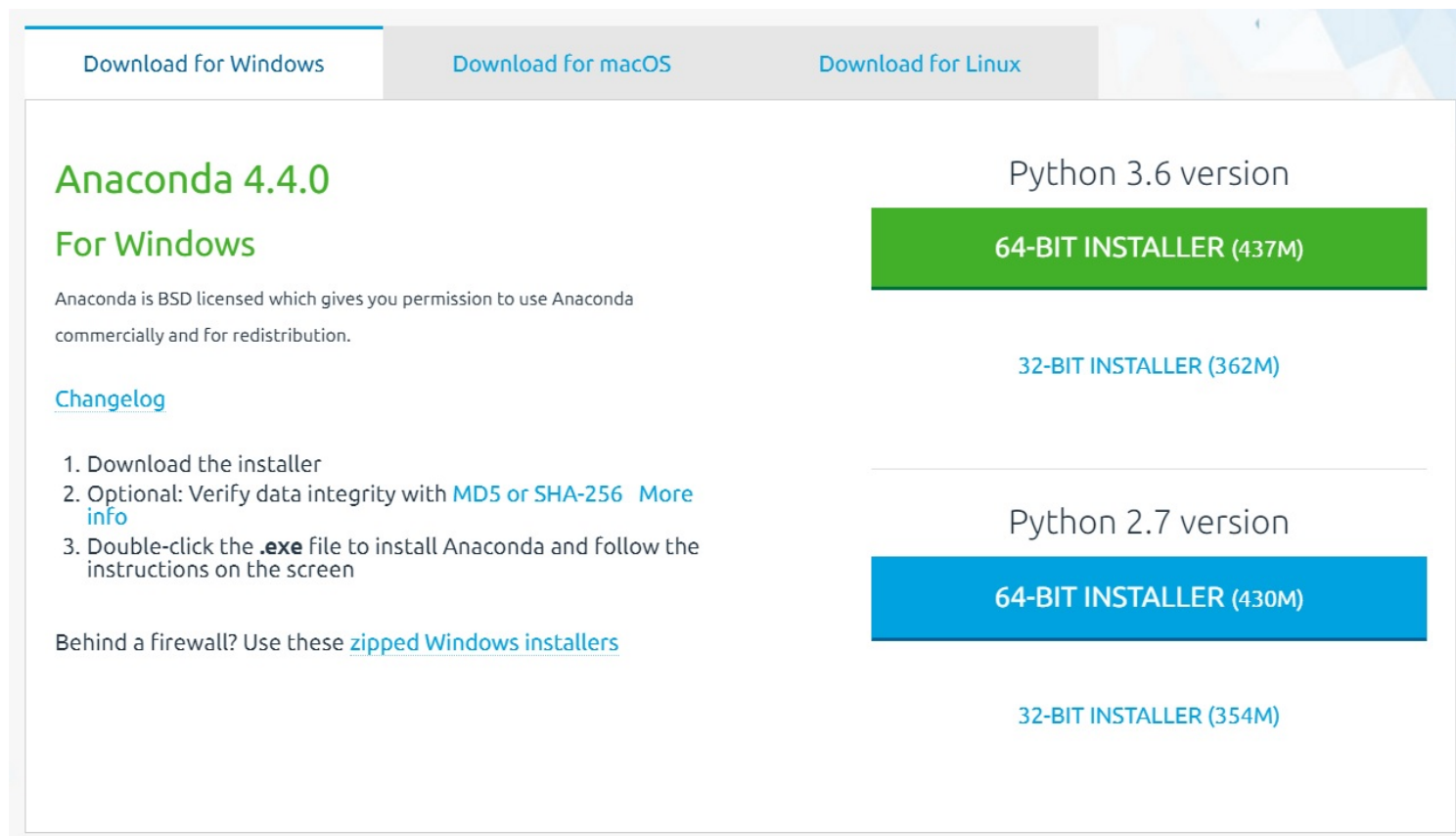


Anaconda(Python) + keras + tensorflow



Anaconda 官網

Anaconda official website



The screenshot shows the Anaconda download page for Windows. At the top, there are three tabs: "Download for Windows" (selected), "Download for macOS", and "Download for Linux". Below the tabs, the page displays "Anaconda 4.4.0 For Windows". A paragraph states: "Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution." Below this is a "Changelog" link. A list of three steps is provided: 1. Download the installer, 2. Optional: Verify data integrity with MD5 or SHA-256 (with a "More info" link), and 3. Double-click the .exe file to install Anaconda and follow the instructions on the screen. At the bottom left, there is a link for "Behind a firewall? Use these zipped Windows installers". On the right side, there are two sections for Python versions. The first section is for "Python 3.6 version" and contains two buttons: a green "64-BIT INSTALLER (437M)" button and a blue "32-BIT INSTALLER (362M)" button. The second section is for "Python 2.7 version" and contains two buttons: a blue "64-BIT INSTALLER (430M)" button and a blue "32-BIT INSTALLER (354M)" button.

Download for Windows Download for macOS Download for Linux

Anaconda 4.4.0

For Windows

Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution.

[Changelog](#)

1. Download the installer
2. Optional: Verify data integrity with [MD5 or SHA-256](#) [More info](#)
3. Double-click the **.exe** file to install Anaconda and follow the instructions on the screen

Behind a firewall? Use these [zipped Windows installers](#)

Python 3.6 version

64-BIT INSTALLER (437M)

[32-BIT INSTALLER \(362M\)](#)

Python 2.7 version

64-BIT INSTALLER (430M)

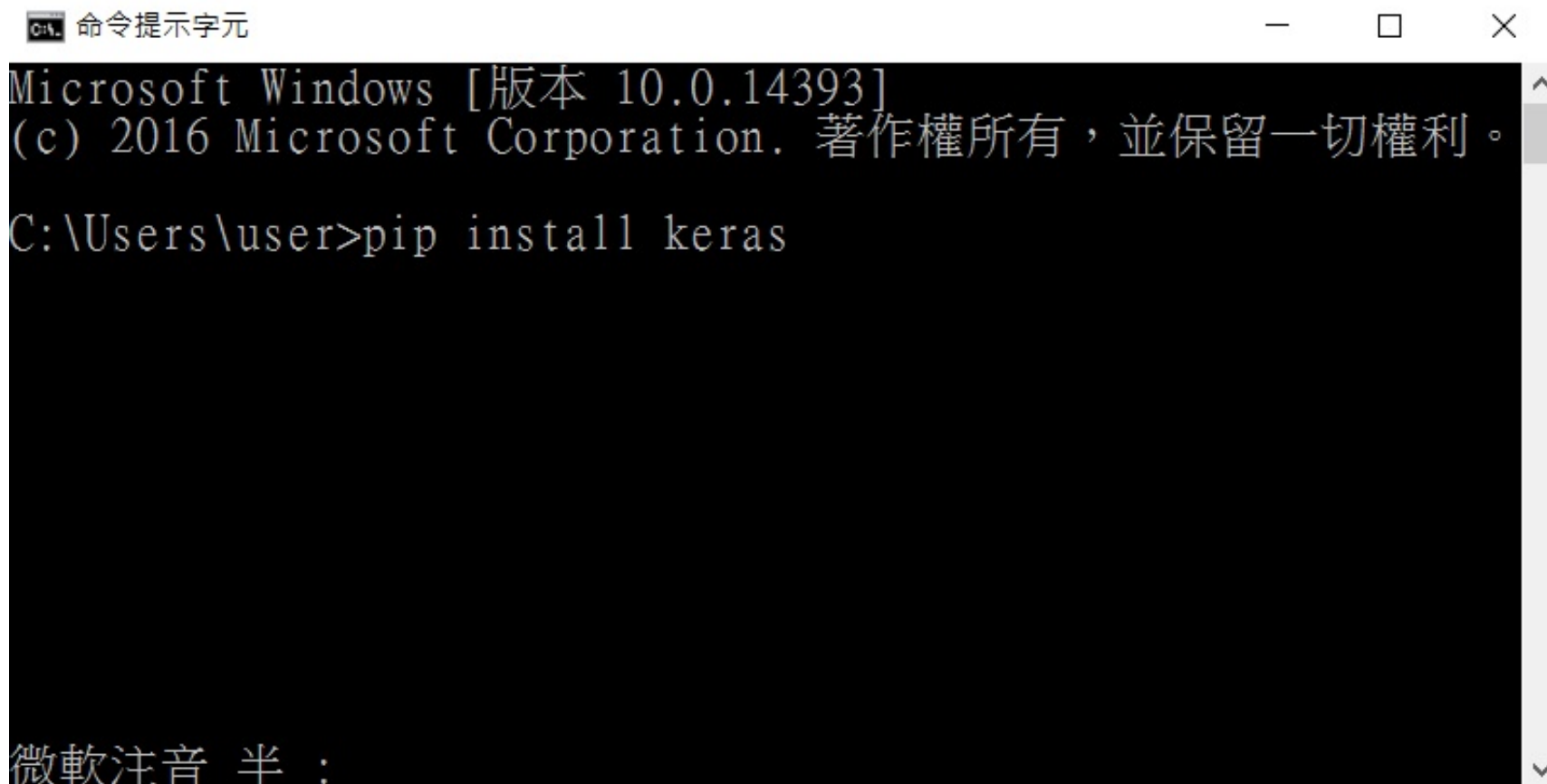
[32-BIT INSTALLER \(354M\)](#)

<https://www.continuum.io/downloads>

下載符合自己作業系統的版本並安裝
(Python 3.6)

打開terminal(CMD)

Open terminal(command line)

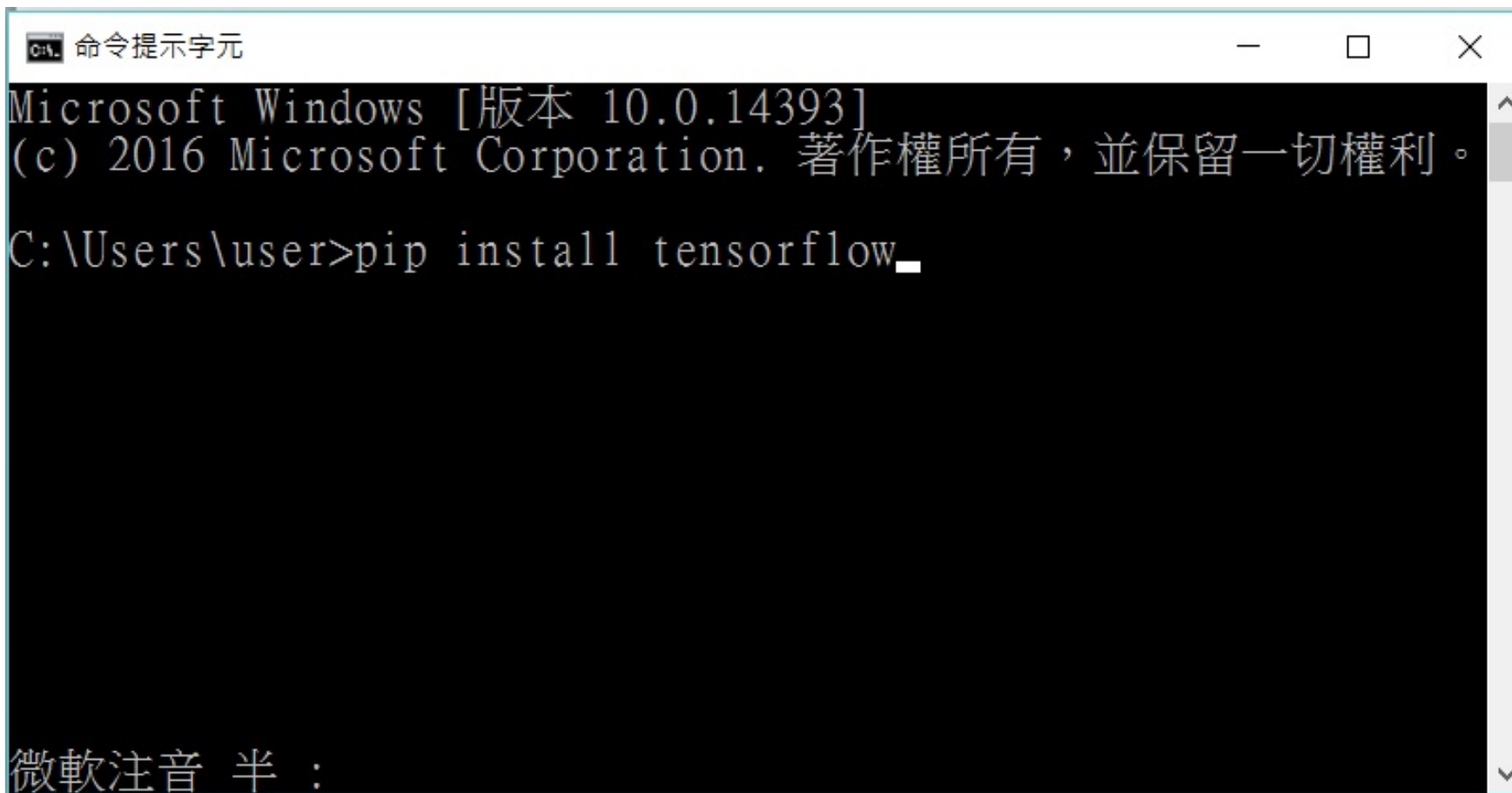


```
C:\> 命令提示字元
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation. 著作權所有，並保留一切權利。
C:\Users\user>pip install keras
```

輸入
pip install keras

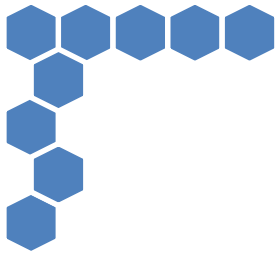
還是在terminal

Still on terminal

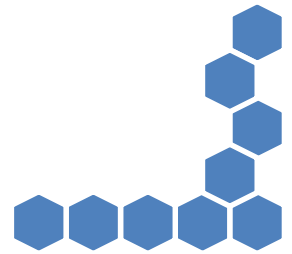


```
命令提示字元
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation. 著作權所有，並保留一切權利。
C:\Users\user>pip install tensorflow_
```

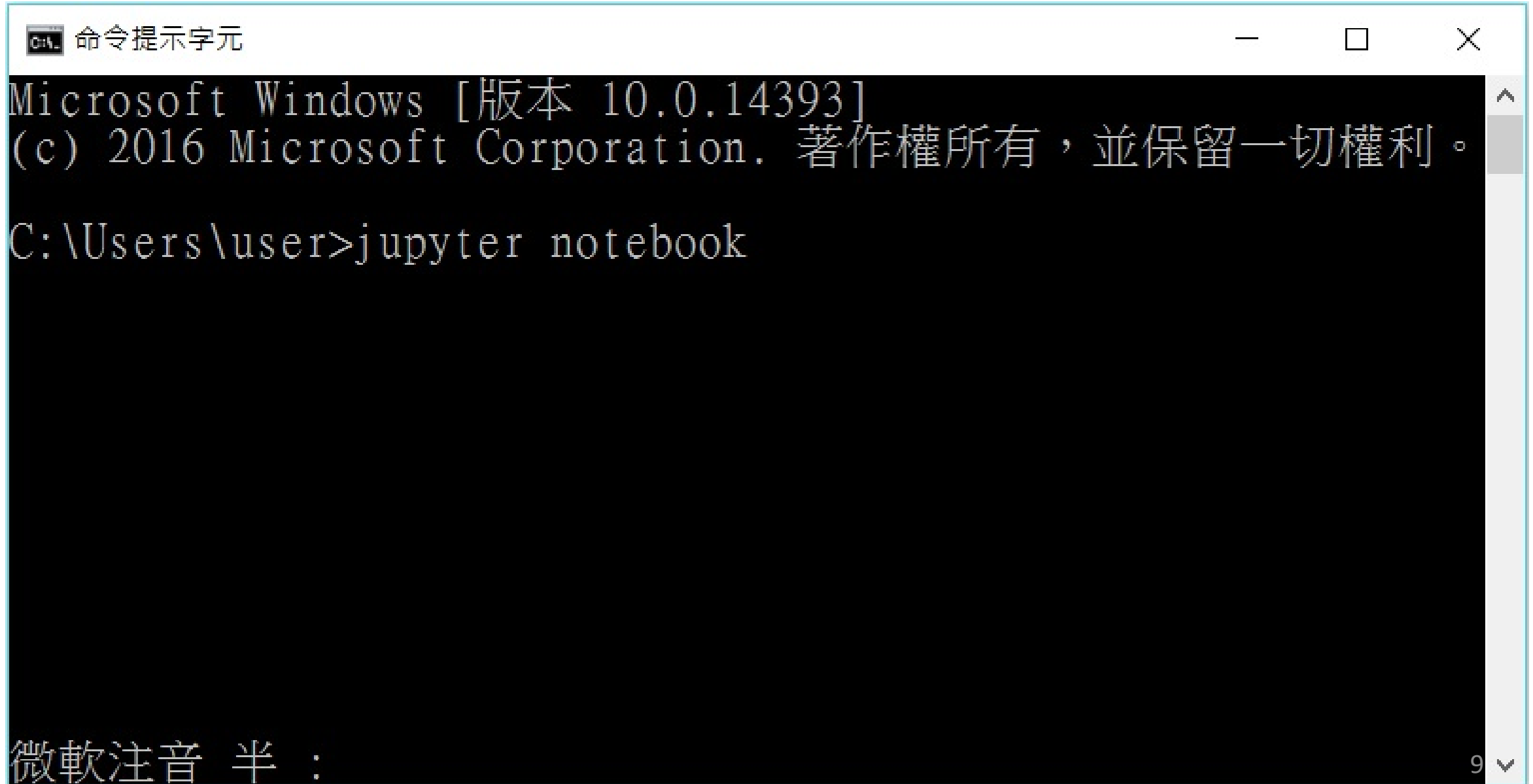
輸入
pip install tensorflow



We are done



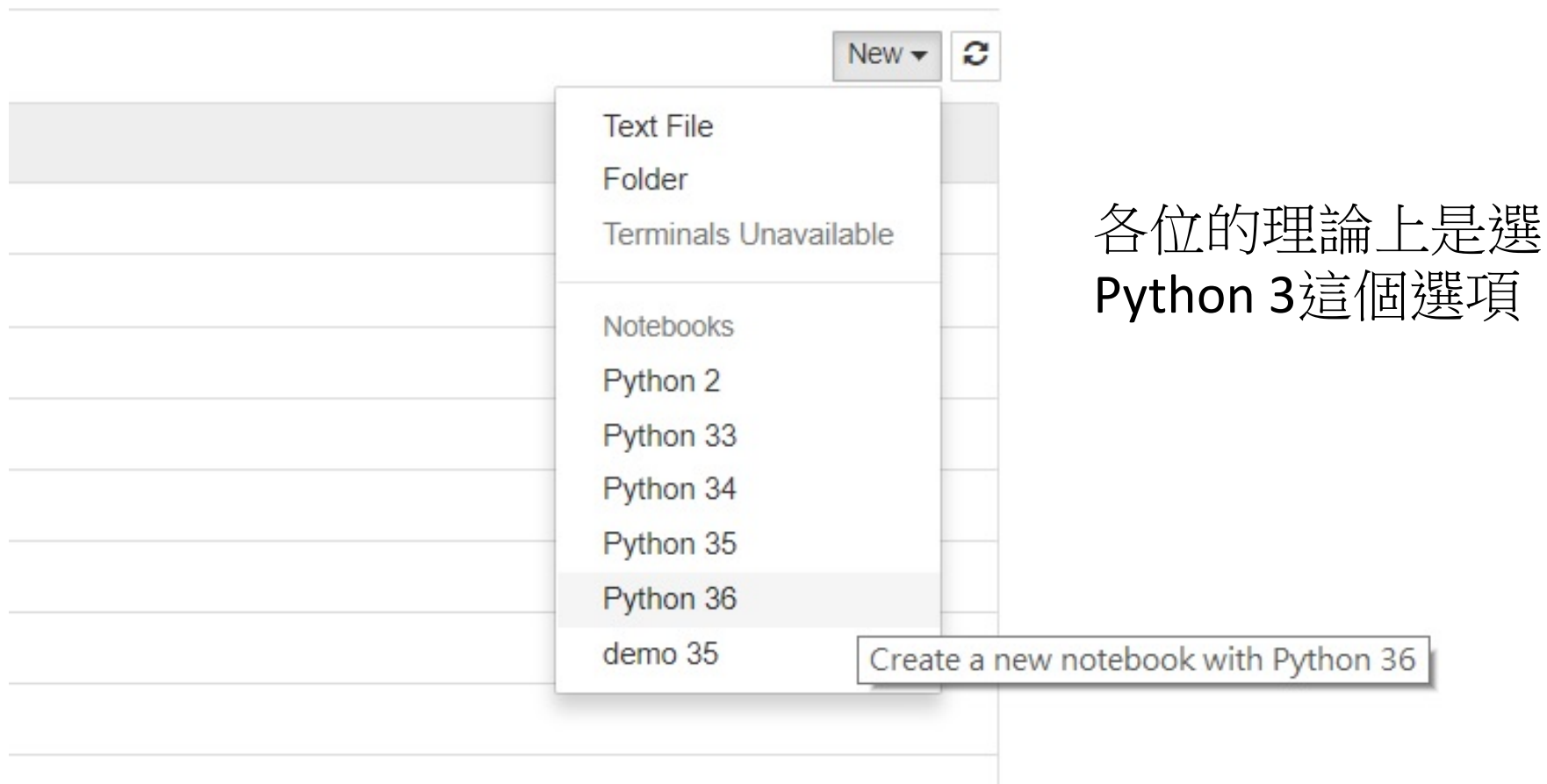
Test it on jupyter notebook



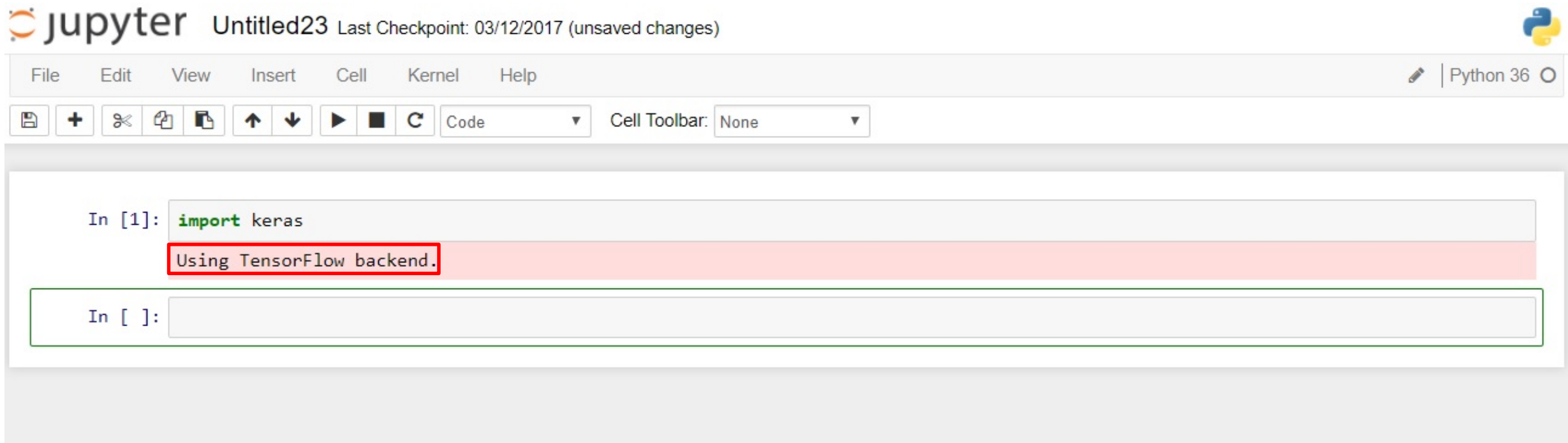
A screenshot of a Windows Command Prompt window. The title bar at the top reads "命令提示字元" (Command Prompt) with standard Windows window controls (minimize, maximize, close). The main area has a black background with white text. It displays the Windows version information: "Microsoft Windows [版本 10.0.14393] (c) 2016 Microsoft Corporation. 著作權所有，並保留一切權利。" followed by the command prompt "C:\Users\user>jupyter notebook". At the bottom left, there is a watermark "微軟注音 半" (Microsoft Pinyin Half). At the bottom right, there is a small "9" and a downward arrow icon.

```
命令提示字元
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation. 著作權所有，並保留一切權利。
C:\Users\user>jupyter notebook
微軟注音 半
9
```

在右上角的New建立新的notebook



輸入 import keras 再按下Shift + Enter



看到Using TensorFlow Backend就表示我們成功了

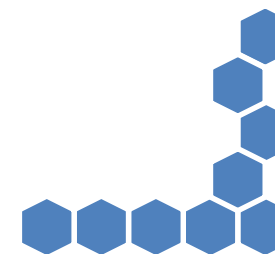


顯卡版本

GPU version

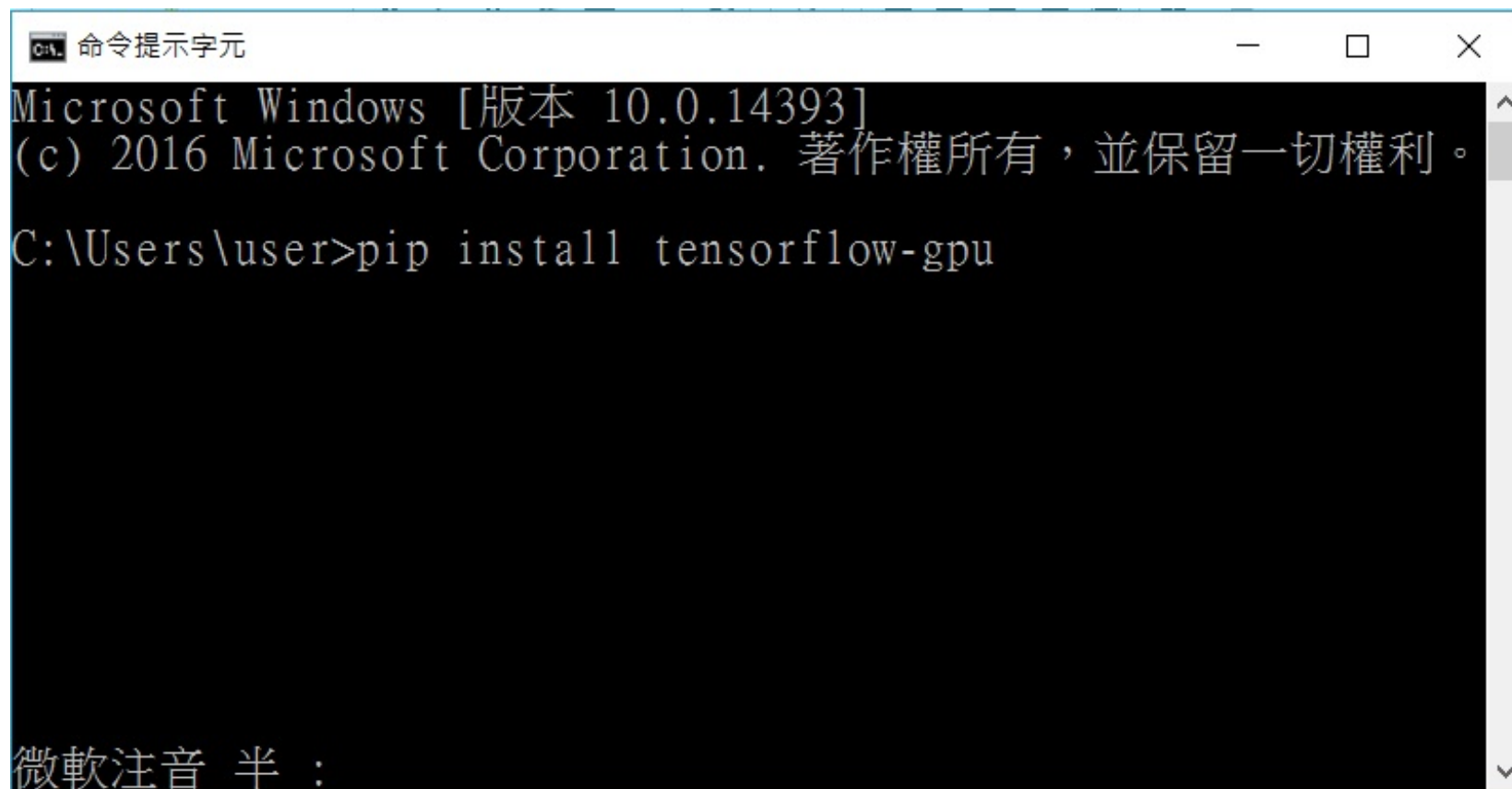


Anaconda(Python) + keras + tensorflow-gpu +
CUDA + cuDNN



前面都一樣

Change from pip install tensorflow



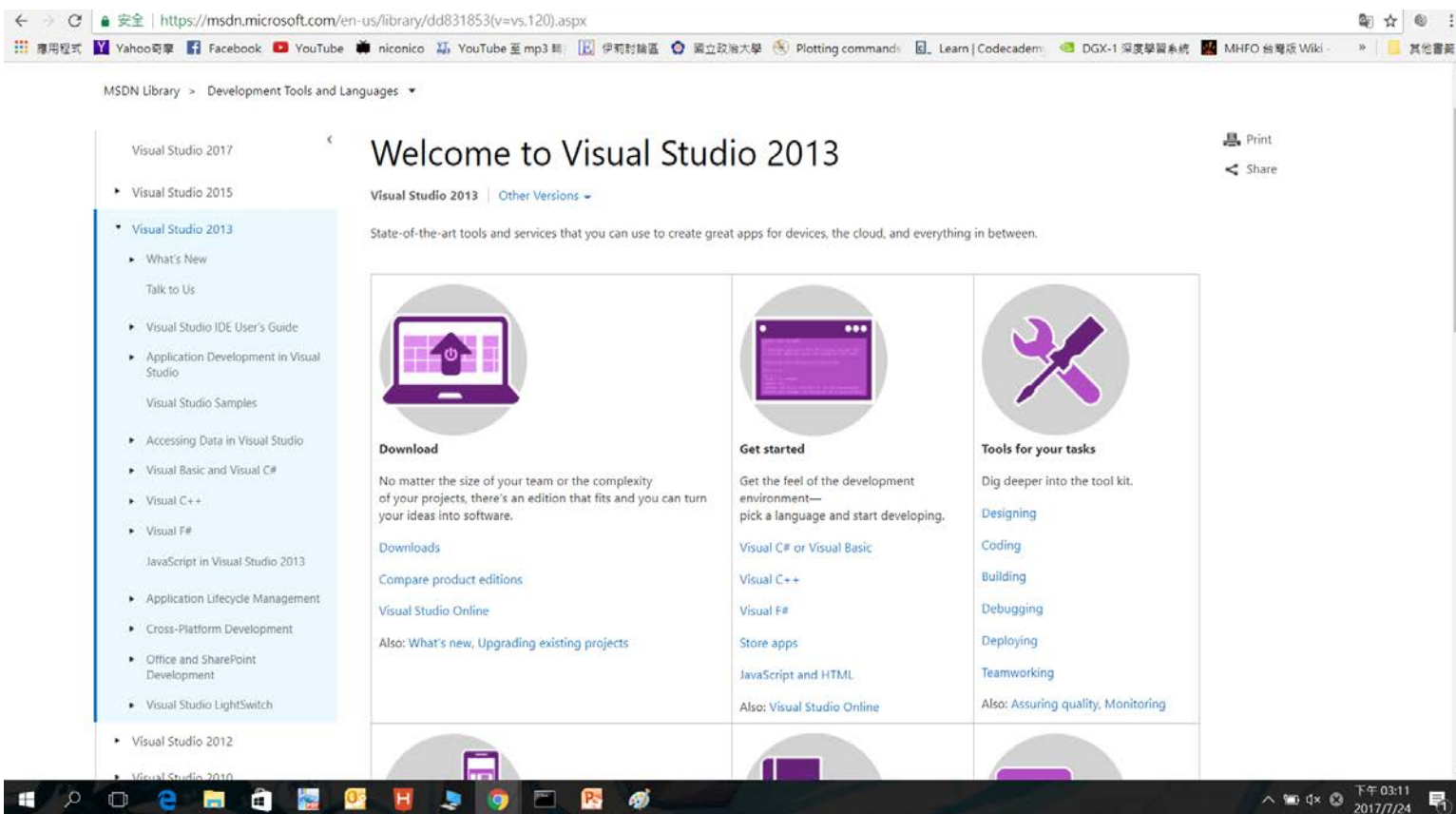
A screenshot of a Windows Command Prompt window. The title bar reads "命令提示字元" (Command Prompt). The window content shows the following text: "Microsoft Windows [版本 10.0.14393] (c) 2016 Microsoft Corporation. 著作權所有，並保留一切權利。 C:\Users\user>pip install tensorflow-gpu". At the bottom left, there is a small text overlay: "微軟注音 半 :".

```
命令提示字元
Microsoft Windows [版本 10.0.14393]
(c) 2016 Microsoft Corporation. 著作權所有，並保留一切權利。
C:\Users\user>pip install tensorflow-gpu
```

微軟注音 半 :

輸入
pip install tensorflow-gpu
(不用裝tensorflow)

CUDA toolkit (windows)



先安裝visual studio 2012 or 2013 or 2015

目前2017不行



CUDA toolkit (ubuntu)

需在終端機安裝

`sudo apt-get install libcupti-dev`

CUDA toolkit

CUDA Toolkit Download

[Home](#) > [ComputeWorks](#) > [CUDA Toolkit](#) > [CUDA Toolkit Download](#)

Select Target Platform ⓘ

Click on the green buttons that describe your target platform. Only supported platforms will be shown.

Operating System

Windows

Linux

Mac OSX

Architecture ⓘ

x86_64

Version

10

8.1

7

Server 2016

Server 2012 R2

Server 2008 R2

Installer Type ⓘ

exe [network]

exe [local]

Download Installers for Windows 10 x86_64

The base installer is available for download below.

There is 1 patch available. This patch requires the base installer to be installed first.

> Base Installer

Download [1.3 GB] 

Installation Instructions:

1. Double click cuda_8.0.61_win10.exe

選擇適當的版本下載並安裝

Windows一定要先裝visual
studio再裝cuda toolkit

cuDNN

[Home](#) > [ComputeWorks](#) > [Deep Learning](#) > [Software](#) > [cuDNN Download](#)

cuDNN Download

NVIDIA cuDNN is a GPU-accelerated library of primitives for deep neural networks.

☒ **I Agree To the Terms of the [cuDNN Software License Agreement](#)**

Please check your framework documentation to determine the recommended version of cuDNN.

If you are using cuDNN with a Pascal (GTX 1080, GTX 1070), version 5 or later is required.

[Download cuDNN v6.0 \(April 27, 2017\), for CUDA 8.0](#)

[Download cuDNN v6.0 \(April 27, 2017\), for CUDA 7.5](#)

[Download cuDNN v5.1 \(Jan 20, 2017\), for CUDA 8.0](#)

[Download cuDNN v5.1 \(Jan 20, 2017\), for CUDA 7.5](#)

[Download cuDNN v5 \(May 27, 2016\), for CUDA 8.0](#)

[Download cuDNN v5 \(May 12, 2016\), for CUDA 7.5](#)

[Download cuDNN v4 \(Feb 10, 2016\), for CUDA 7.0 and later.](#)

[Archived cuDNN Releases](#)

cuDNN版本跟顯卡和tensorflow
版本有關有關

下載後解壓縮並放到cuda
toolkit的資料夾內

在windows叫不出GPU的話可以改用theano

Tensorflow與windows的相容性較差

把tensorflow backend改成theano backend (keras不變)

一樣用pip install theano安裝

In [1]: `import keras`

```
Using Theano backend.  
WARNING (theano.sandbox.cuda): The cuda backend is deprecated and will be removed in the next release (v0.10). Please switch to  
the gpuarray backend. You can get more information about how to switch at this URL:  
https://github.com/Theano/Theano/wiki/Converting-to-the-new-gpu-back-end%28gpuarray%29
```

```
Using gpu device 0: GeForce GTX 760M (CNMeM is enabled with initial size: 80.0% of memory, cuDNN 5110)
```

↑成功的話會顯示顯卡名稱 (tensorflow和theano都會顯示)

更多資訊

For more information



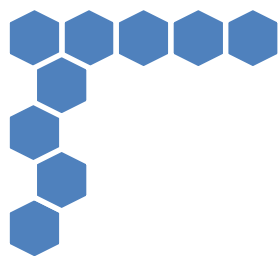
<https://www.continuum.io/>

<https://keras.io/>

<https://www.tensorflow.org/>

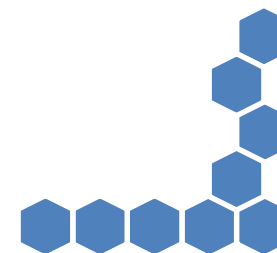
<http://deeplearning.net/software/theano/>





利用 Jupyter Notebook 編輯程式

善用雲端開發環境

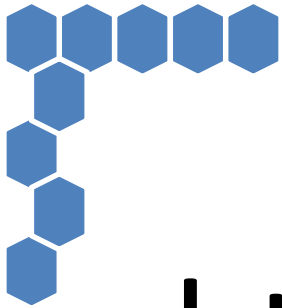


國立政治大學金融科技研究中心
智能理財與深度學習暑期訓練營

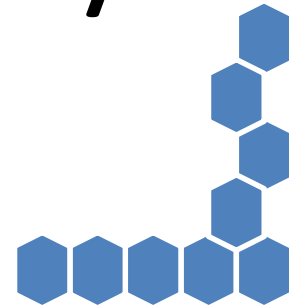




啟動方式




<http://dev.ifa.finance:3000/>



輸入帳密 開始使用

AI開發平台

[回首頁 |](#)

 jupyter

Sign in

Username:

Password:

Sign In

打包

編輯快捷鍵

Keyboard Shortcuts

jupyter Alfinance-Day1 (autosaved)

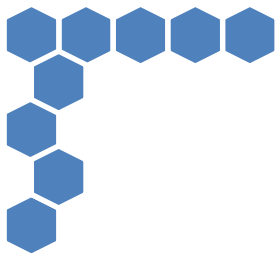
File Edit View Insert Cell Kernel Widgets Help

Keyboard shortcuts

- Navigation**
 - 1**: to heading 1
 - 2**: to heading 2
 - 3**: to heading 3
 - 4**: to heading 4
 - 5**: to heading 5
 - 6**: to heading 6
 - K**: select cell above
 - Up**: select cell above
 - Down**: select cell below
 - J**: select cell below
 - Shift-K**: extend selected cells above
 - Shift-Up**: extend selected cells above
- Editing**
 - Ctrl-S**: Save and Checkpoint
 - S**: Save and Checkpoint
 - L**: toggle line numbers
 - O**: toggle output of selected cells
 - Shift-O**: toggle output scrolling of selected cells
 - H**: show keyboard shortcuts
 - I, I**: interrupt kernel
 - 0, 0**: restart the kernel (with dialog)
 - Esc**: close the pager
 - Q**: close the pager
 - Shift-L**: toggles line numbers in all cells, and persist the setting
 - Shift-Space**: scroll notebook up
 - Space**: scroll notebook down

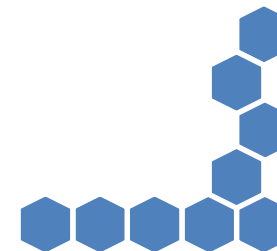
Edit Mode (press **Enter** to enable)

- Tab**: code completion or indent
- Shift-Tab**: tooltip
- Ctrl-J**: indent
- Ctrl-[**: dedent
- Ctrl-A**: select all
- Ctrl-Z**: undo
- Ctrl-Shift-Z**: redo
- Ctrl-Y**: redo
- Ctrl-Home**: go to cell start
- Ctrl-Up**: go to cell start
- Ctrl-End**: go to cell end
- Ctrl-Down**: go to cell end
- Ctrl-Left**: go one word left
- Ctrl-Right**: go one word right
- Ctrl-Backspace**: delete word before
- Ctrl-Delete**: delete word after
- Ctrl-M**: command mode
- Ctrl-Shift-F**: open the command palette
- Ctrl-Shift-P**: open the command palette
- Esc**: command mode
- Shift-Enter**: run cell, select below
- Ctrl-Enter**: run selected cells
- Alt-Enter**: run cell, insert below
- Ctrl-Shift-Minus**: split cell
- Ctrl-S**: Save and Checkpoint
- Down**: move cursor down
- Up**: move cursor up



Python 基礎

Python tutorial



國立政治大學金融科技研究中心
智能理財與深度學習暑期訓練營



大綱

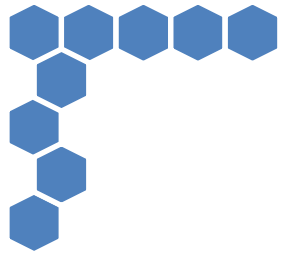
Agenda

Basic Python

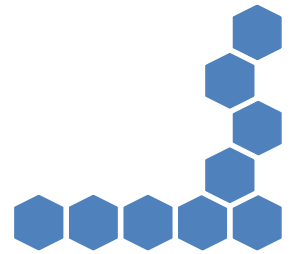
datatypes

bool in Python

for and while loops



Basic Python



1. 註解

comment

在前面加上#來為自己的程式碼註解，
方便他人及自己了解程式碼的意義。

```
#123456  
#abc
```

2.基本運算

basic calculation

$1+3*8$ #加減乘除就跟小算盤的一樣

>>25

$(11-3)/8$ #當然也可以加括號

>>1

$2**3$ #注意次方是兩個乘號

>>8

3. = 的意義

the meaning of =

```
S = 0
```

```
S = S+1
```

```
S = S+2
```

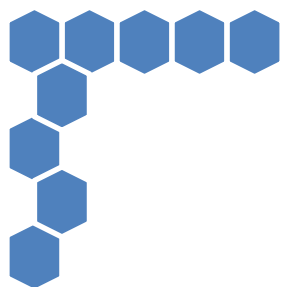
```
print(S)
```

```
>>3
```

#程式語言的=是指派的意思，
不是我們一般認知的等於

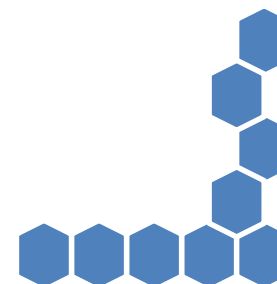
4.魔術指令

magic code



```
%pylab inline  
%matplotlib inline  
%save  
%run
```

注意魔術指令只能在jupyter notebook執行





```
%pylab inline(1)
```



```
%pylab inline
```

```
sin(pi/2)
```

#可以算三角函數

```
>>1
```

```
a = randn(15)
```

#可以取亂數

```
>>給一組15個亂數
```

```
(以0為平均，1為標準差的常態分配)
```



```
%pylab inline(2)
```

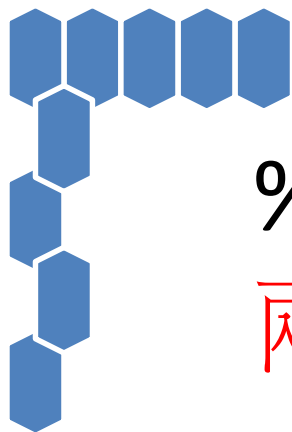


```
plot([-2, 10, 3, 13, 5]) #可以畫圖
```

>>一張摺線圖

```
plot(a) #畫出前一頁的亂數a的摺線圖
```


%pylab inline做了什麼？



%pylab引入了 **numpy** 和 **matplotlib**
兩個套件會在下節課介紹

inline與%matplotlib inline做同一件事
讓圖畫在notebook的頁面上
(不然會畫在另一個視窗)



%save用法

%save可將程式碼存成一個.py檔，語法為：

`%save filename n1-n2 n3-n4 ... n5 .. n6 ...`

```
In [1]: import numpy as np
import numpy.random as rn
import numpy.linalg as lg

In [4]: A=np.mat(rn.rand(3,3))

In [5]: A
Out[5]: matrix([[ 0.93360084,  0.21712287,  0.87538231],
 [ 0.52478396,  0.93068289,  0.78416744],
 [ 0.17330869,  0.16302538,  0.09078136]])

In [6]: Q=np.mat(np.zeros([3,3]))
R=np.mat(np.zeros([3,3]))

In [7]: R[0,0]=lg.norm(A[:,0])

In [8]: Q[:,0]=A[:,0]/R[0,0]

In [10]: R[0,1]=A[:,1].T*Q[:,0]

In [11]: q=A[:,1]-R[0,1]*Q[:,0]

In [12]: R[1,1]=lg.norm(q)

In [13]: Q[:,1]=q/R[1,1]
```

n1,n2等指的是紅框內的數字，跟程式碼在第幾行無關

%save示範

若要存此畫面所有程式碼，則程式碼為:

`%save demo.py 1 4-8 10-13`

```
In [1]: import numpy as np
import numpy.random as rn
import numpy.linalg as lg
```

```
In [4]: A=np.mat(rn.rand(3,3))
```

```
In [5]: A
```

```
Out[5]: matrix([[ 0.93360084,  0.21712287,  0.87538231],
 [ 0.52478396,  0.93068289,  0.78416744],
 [ 0.17330869,  0.16302538,  0.09078136]])
```

```
In [6]: Q=np.mat(np.zeros([3,3]))
R=np.mat(np.zeros([3,3]))
```

```
In [7]: R[0,0]=lg.norm(A[:,0])
```

```
In [8]: Q[:,0]=A[:,0]/R[0,0]
```

```
In [10]: R[0,1]=A[:,1].T*Q[:,0]
```

```
In [11]: q=A[:,1]-R[0,1]*Q[:,0]
```

```
In [12]: R[1,1]=lg.norm(q)
```

```
In [13]: Q[:,1]=q/R[1,1]
```



`%run`



```
%run demo.py
```

>>執行上一頁存的demo.py

#檔案必須與notebook在同
個目錄底下才可執行

5. 定義函數

define a function

#只要有:下一行就要空格

```
def foo(x,y):
```

#注意要空格

#空幾格沒特別規定但是要對齊

#也不能空太少格

```
    a = x**2 + y**2
```

```
    b = "foo"
```

```
    return a,b
```

6.指令的文檔

code documentation

```
#Try this
```

```
%pylab?
```

```
>>>一大串文字
```

```
#在指令後方加?可叫出該指令的文檔
```

Exercise(1)



已知bmi的算法為 $\frac{\text{體重}(kg)}{\text{身高}^2(m)}$

試著定義一個函數**bmi(height,weight)**
回傳bmi值



讓你的程式雲端執行

#將練習1的程式加上以下兩行程式碼

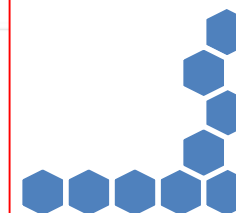
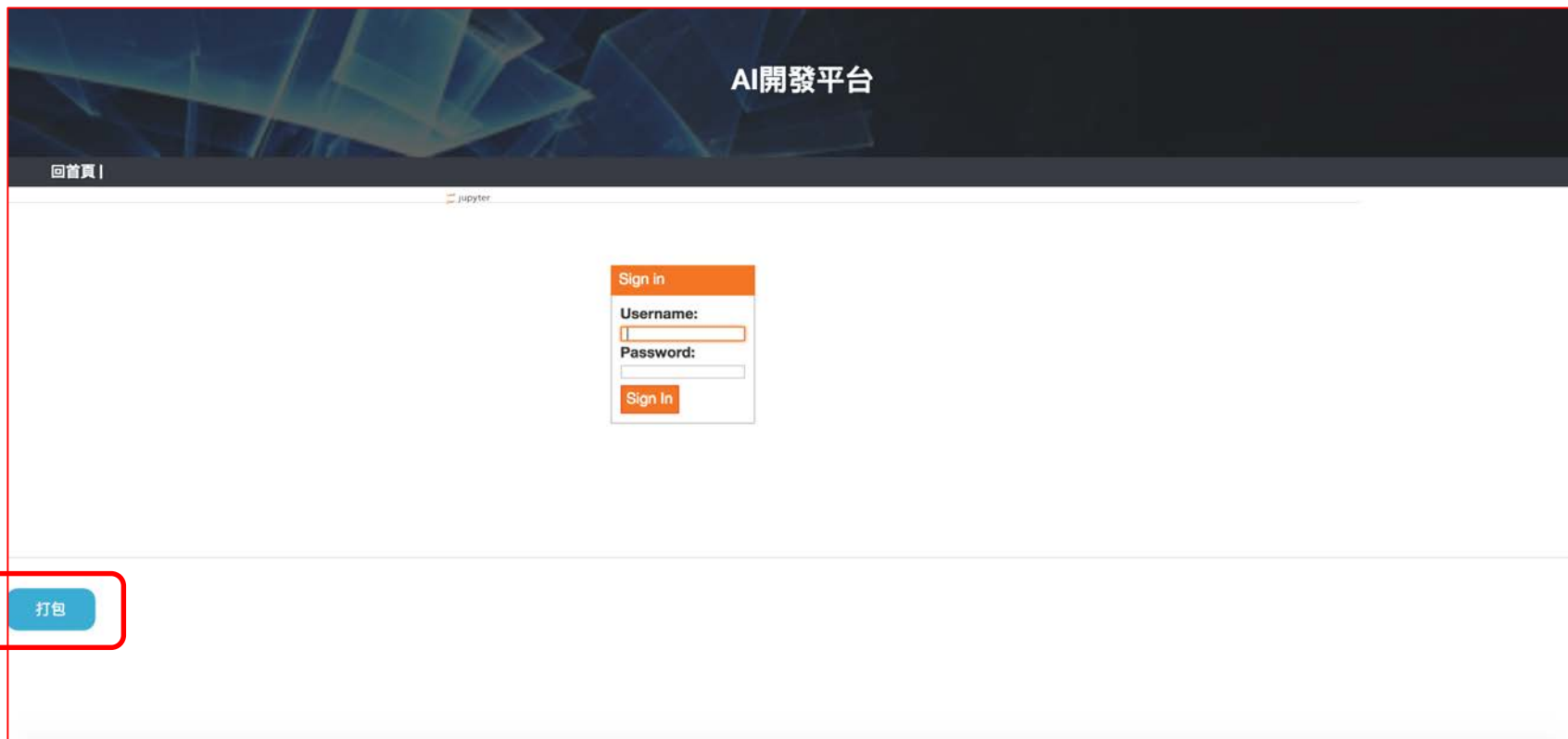
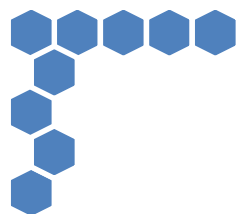
```
import gi  
gi.res(bmi(1.7, 75))
```

#透過gi.res方法將雲端主機執行bmi函數結果呈現在網頁上

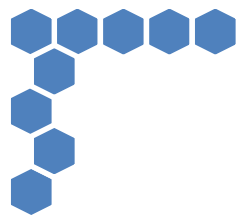
將你的程式儲存

```
%save ex1 練習1程式所在行數  
>>儲存練習1程式
```

啟動打包功能



把程式上傳





[回首頁 |](#)

專案名稱:

我的檔案清單:

aaa.py

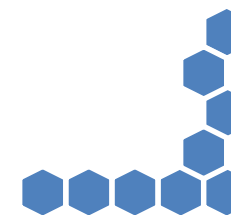
gi.py

test.py

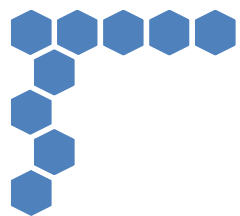
test2.py

test3.py

上傳



執行程式



project06

檔案名稱: aaa.py

作者: teacher

上傳日期: Tue Jul 25 2017 22:11:39 GMT+0800 (CST)

執行

刪除

project05

檔案名稱: test2.py

作者: teacher

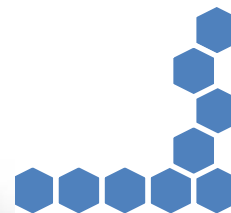
上傳日期: Tue Jul 25 2017 22:09:17 GMT+0800 (CST)

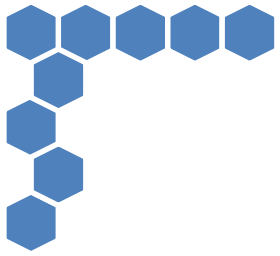
執行

刪除

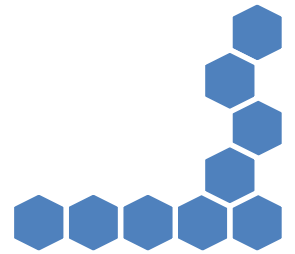
project02

檔案名稱: test3.py





datatypes



1.串列(1)

list

```
l = [-3, 7, 12, 0, 8, -20, 33]
```

```
l[0]
```

```
>>-3
```

```
l[3]
```

```
>>0
```

#Python幾乎都從0開始數

1.串列(2)

list

```
1[1:3]      #取到:右邊的數字-1
```

```
>>[7,12]
```

```
1[:3]      #不寫表示取到底
```

```
>>[-3,7,12]
```

```
1[-1]      #list可以倒著數
```

```
>>33
```

1.串列(3)

list

```
g = [3, 1, 2, 5]
```

```
l+g          #list可以相加，效果是新增附加
```

```
>>[-3, 7, 12, 0, 8, -20, 33, 3, 1, 2, 5]
```

```
g*2          #乘法是讓它重複
```

```
>>[3, 1, 2, 5, 3, 1, 2, 5]
```


1.串列(4)

list

```
g = [3, 1, 2, 5]
```

```
g.append(4)
```

```
print(g)
```

```
>>[3, 1, 2, 5, 4]
```

#append可在list後面新增新的物件

2.字串(1)

string

```
message = "Hello world"  
message[0]  
>>H          #邏輯與list一樣  
print(message)  
>>Hello world
```

2.字串(2)

string

‘‘‘今天
天氣
很好
’’’

#前後各三點可以做出跨行的大字串

2.字串(3)

string

‘你好’ == “你好” #一點跟兩點是一樣的

>>True

#但不能出現 ‘ “ 或 ” ’

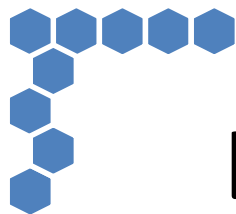
3.字典

dictionary

此為不良示範，盡量不要用
默認字當變數

```
dict = {'apple': '蘋果', 'banana': '香蕉'}  
dict['apple']  
>>'蘋果'           #像查字典一樣  
dict['orange'] = '橘子'  
dict                #可以新增  
>>{'apple': '蘋果', 'banana': '香蕉', 'orange': '橘子'}
```

Exercise(2)



```
List = [[0,1],[2,3],[5,6,7]]
```

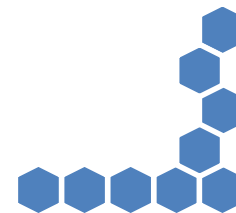
```
1.List[2][0]
```

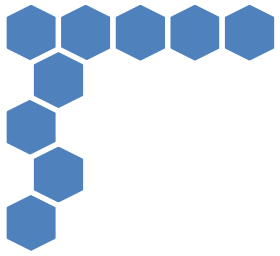
```
>> ???
```

```
2.List[a][b]
```

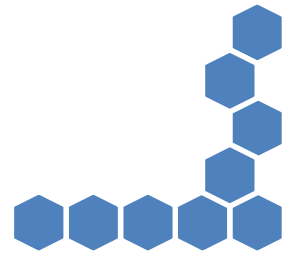
```
>>3
```

求a,b分別為多少





bool in Python



1.比大小(1)

Bigger or smaller

```
2 < 3
```

```
>>True
```

```
a = 7
```

```
a != 8
```

```
>>True
```

#Python的不等於是!=

1.比大小(2)

Bigger or smaller

```
a <= 9
```

```
>>True
```

```
a == 8
```

```
>>False
```

#我們一般認知的等於是==

1.比大小(3)

Bigger or smaller

“Z” > “B”

>>True

“apple” > “banana”

>>False

#文字字串也可以比大小
#邏輯是照英文字典的排序

in 的用法(1)

the use of in

```
keyword = “悲傷”  
message1 = “我今天很悲傷”  
keyword in message1  
>>True
```

in 的用法(2)

the use of in

```
keyword = “悲傷”  
message2 = “我今天很開心”  
keyword in message2  
>>False
```

in 的用法(3)

the use of in

```
list0 = [0,1,2,3]
```

```
2 in list0
```

```
>>True
```

```
4 in list0
```

```
>>False
```

if else 的用法(1)

the use of if and else

```
s = 60
```

```
if s >= 60:
```

#注意冒號且if也要空格，規則跟def一樣

```
    print("過了了不起哦")
```

```
else:
```

```
    print("好課值得一修再修")
```

if else 的用法(2)

the use of if and else

```
if s>60:
    print("過了了不起哦")
elif s==60:    #elif是else if的縮寫
    print("去好好感謝老師吧")
else:
    print("好課值得一修再修")
```



for and while loops



for 迴圈(1)

for loop

```
s = 0
```

```
grades = [77, 85, 56, 90, 66]
```

```
for i in grades:
```

```
    s = s + i
```

#for也要空格

```
print(s)
```

```
>>374
```

#grades的總和

for 迴圈(2)

for loop

```
s = 0
```

```
grades = [77, 85, 56, 90, 66]
```

```
for i in range(5):
```

#range(n)會從0取到n-1

```
    s = s + grades[i]
```

```
print(s/5)
```

```
>>74.8
```

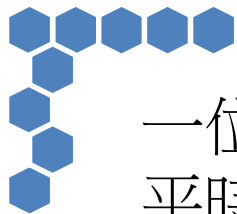
#grades的平均

while 迴圈

while loop

```
s = 0; i = 0          #Python也可以用分號
while i < len(grades): #只要有:下一行就要空格
    s = s + grades[i]
    i += 1            #Python其實也可以這樣寫
print(s)
>>374                #注意別讓它掉進無窮迴圈
```

Exercise(3)



一位老師成績這樣算的:

平時成績 20%

期中考 35%

期末考 45%

有位同學

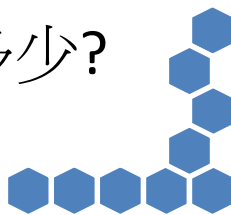
平時成績 85 分

期中 70 分

期末 80 分

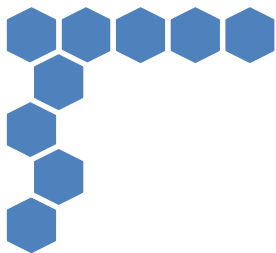
這位同學的學期成績是多少?

試著用for迴圈完成



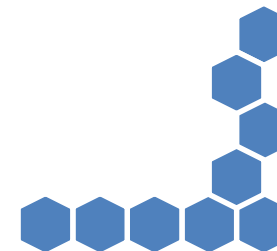
```
grades = [85, 70, 80]  
weights = [0.2, 0.35, 0.45]
```

⋮



Python 套件

Python site-packages



國立政治大學金融科技研究中心
智能理財與深度學習暑期訓練營



大綱

Agenda

How to import a package

numpy and tensor

matplotlib

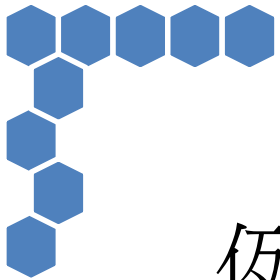
pandas



How to import a package



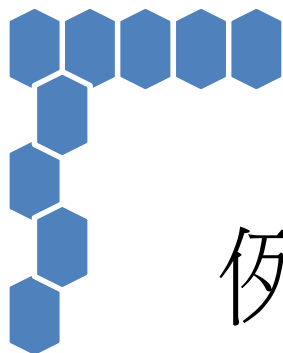
Import套件的兩種方式 (1)



from 套件 import 指定函數
例如: from numpy import sin
from math import *
(*表示所有函數)



Import套件的兩種方式 (2)



import 套件 (as 縮寫)

例如: import numpy

→ numpy.sin

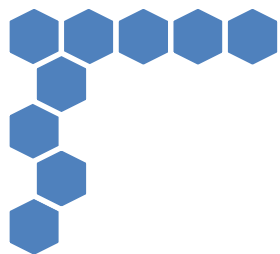
import numpy as np

→ np.sin

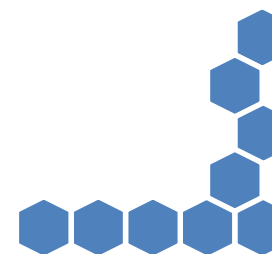
一般我們都用這種方式
(絕不是因為程式碼比較長看起來比較厲害)



%pylab inline不好嗎?




答案是: 不好

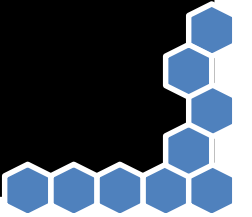


為什麼不好?

因為不同套件可能有同名但不同定義的函數



```
from random import *  
for i in range(50):  
    print(randint(1,10), end=', ')  
-----  
from numpy.random import *  
for i in range(50):  
    print(randint(1,10), end=', ')
```



發現不同了嗎？

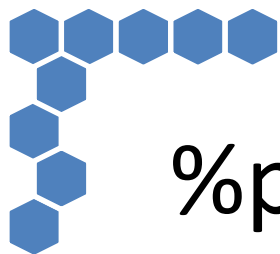


random的randint(1,10)有包含10
numpy的randint(1,10)只包含到9

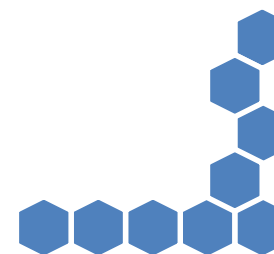
為了避免我們混淆不同套件的函數，我們選擇第二種引入套件的方式
(把套件名稱放在函數前面)



%pylab inline不好



%pylab inline會把numpy和matplotlib的
所有函數以第一種方式引入
所以它不是個好的方式





numpy and tensor



引入numpy



```
#這是標準引入法  
import numpy as np
```



What can numpy do?(1)



```
grades = [33, 74, 12, 87, 55]
```

```
np.average(grades)
```

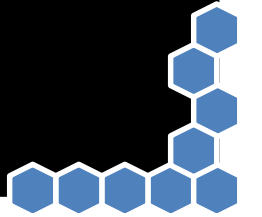
```
>>52.2000000000000000000003
```

#可以算平均

```
np.max(grades)
```

```
>>87
```

#可以找最大值



What can numpy do?(2)



```
np.std(grades)
```

```
>>27.12489631316588
```

#連標準差都可以算

```
np.pi
```

```
>>3.141592653589793
```

#有我們熟知的圓周率

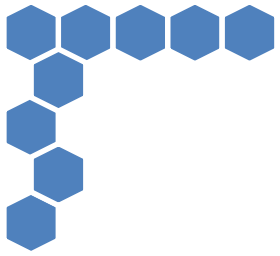
```
np.sin(3)
```

```
>>0.14112000805986721
```

#也有三角函數的數值計算



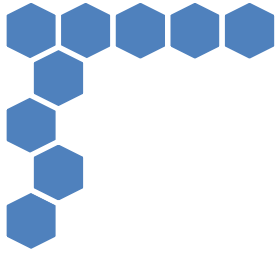
What is numpy?



numpy 是以array(陣列)為基礎的套件

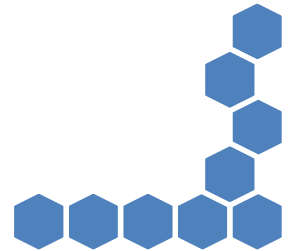


What is array?



array 是個像矩陣的資料型態

但注意它不是Python的矩陣(串列)



看看array能做些什麼!(1)

array裡不只可以放數字跟字串




```
s = np.array([-3, 2, 7, -5, 2])
```

```
s > 0
```

```
>>array([False, True, True, False, True], dtype=bool)
```

看看array能做些什麼!(2)

array讓複雜的事變簡單



```
s = np.array([-3, 2, 7, -5, 2])  
s[s > 0]  
>>array([2, 7, 2])
```



看看array能做些什麼!(3)

array是有形狀的



```
a = np.arange(28)
```

```
a.shape
```

```
>>(28,)
```

```
a.shape = (7,4)
```


```
>>將a變成7*4的陣列
```

```
#可以用print(a)看看它的樣子
```



看看array能做些什麼!(4)

特殊array生成法



```
np.zeros(10) #生成都是0的陣列  
np.zeros((3,4))  
np.eye(5) #生成像單位矩陣的陣列  
np.ones(5) #生成都是1的陣列
```



看看array能做些什麼!(5)

array的加法



```
A = np.arange(4)
```

#產生含循序數列0,1,2,3的array

```
B = np.arange(5, 9)
```

```
A.shape=(2, 2)
```

```
B.shape=(2, 2)
```

```
A+B
```


```
>>array([[ 5,  7],  
         [ 9, 11]])
```

#加法跟矩陣加法一樣



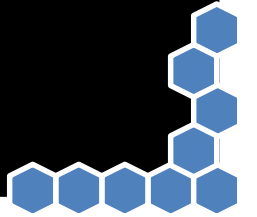
看看array能做些什麼!(6)

array的乘法



```
A = np.arange(4)
B = np.arange(5, 9)
A.shape=(2, 2)
B.shape=(2, 2)
A*B
>>array([[ 0,  6],
          [14, 24]])
```

#是對應的元素相乘，
不是矩陣乘法



補充:Python的矩陣乘法



```
mA = np.matrix(A)
```

```
mB = np.matrix(B)
```

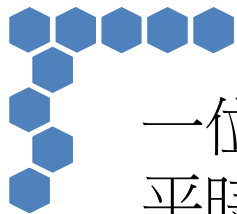
```
mA*mB
```

```
>>matrix([[ 7, 8], [31, 36]])
```

#沿用剛剛2*2的A,B陣列

#這才是我們熟知的矩陣乘法

Exercise(4)



一位老師成績這樣算的:

平時成績 20%

期中考 35%

期末考 45%

有位同學

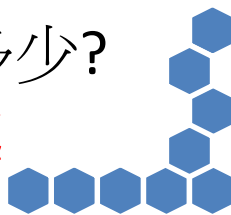
平時成績 85 分

期中 70 分

期末 80 分

這位同學的學期成績是多少?

試著用numpy的函數完成



```
grades = np.array([85, 70, 80])  
weights = np.array([0.2, 0.35, 0.45])
```

⋮

What is tensor?

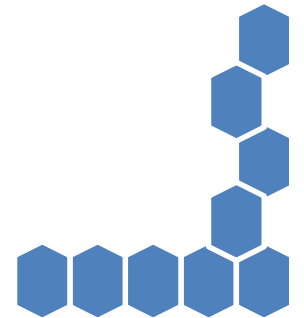


tensor的中文是張量，可當作廣義的向量

第零階張量可視為純量

第一階張量可視為向量

第二階張量可視為矩陣



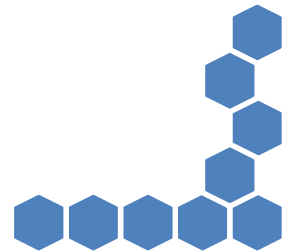
想了解更多可參考

<https://zh.wikipedia.org/wiki/%E5%BC%B5%E9%87%8F>


array跟tensor有什麼關係?



array裡的元素其實被視為向量
list裡的元素被視為物件
什麼意思?來做個實驗



回憶list的append




```
l = [0, 1, 2]
l.append(3)
l
>> [0, 1, 2, 3]
```

#三個物件新增為四個物件是合法的



array不能append

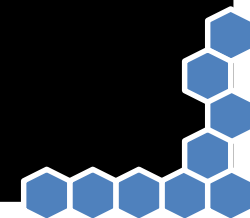


```
a = np.array([0,1,2])  
a.append(3)
```

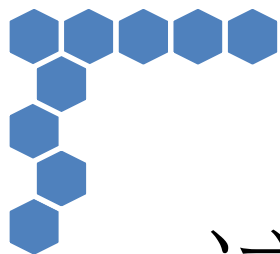
```
>>error message
```

#三維的向量變成四維的向量是不合法的

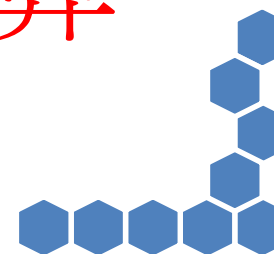
#array因為是向量導向的，所以不能任意新增物件改變維度



array這樣設計有什麼好處？



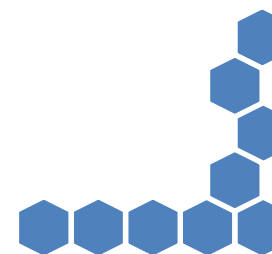
這樣的設計是為了平行運算



平行運算是什麼？



就是把一份工作切給很多人做，原則上CPU是幾核心就切成幾份。**numpy**會自動幫我們完成這複雜的工作!!



平行運算有差那麼多嗎？

```
import numpy as np
from datetime import datetime
```

```
a = np.random.randn(100)
b = np.random.randn(100)
T = 100000
```

```
def slow_dot_product(a,b):
    result = 0
    for e, f in zip(a, b):
        result += e*f
    return result
```

```
t0 = datetime.now()
for t in range(T):
    slow_dot_product(a, b)
dt1 = datetime.now() - t0
```

```
t0 = datetime.now()
for t in range(T):
    a.dot(b)
dt2 = datetime.now() - t0
print(dt1.total_seconds()/dt2.total_seconds())
>>39.897058447044365
```

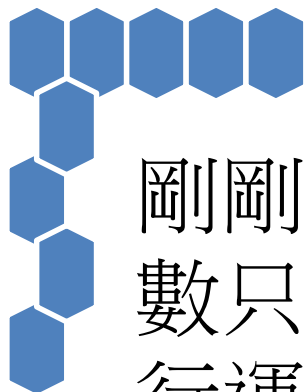
#這程式碼跑第一次會不準，
要多跑幾次

#用for迴圈算內積

#用numpy算內積

#最後算出兩者秒數的比值

時間比值傳達的意義

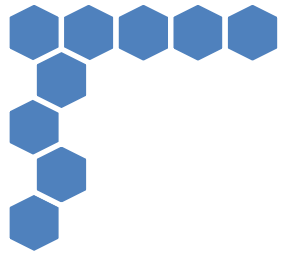


剛剛那個結果是用i7-4700HQ跑出來的結果，運算核心數只差了8倍但時間差了將近40倍，這就是為什麼要平行運算。

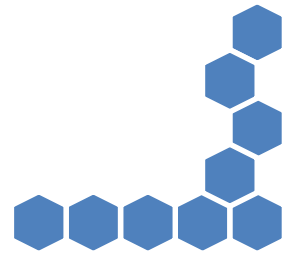
現在新的GPU的運算核心高達上千個，這是為什麼我們需要GPU加速。

目前還有TPU(Tensor Processing Unit)正在發展!!





matplotlib



引入matplotlib



#一般為了方便我們會這樣引入

```
%matplotlib inline
```

```
import matplotlib.pyplot as plt
```

畫基本摺線圖



#注意要打plt

```
plt.plot([1, 2, 3, 4], [3, -5, 7, 2])
```

畫像樣一點的函數圖(1)

$$f(x) = \sin(x) + x$$

#np.linspace(a,b,c)是在a,b區間切c個均勻格點

```
x = np.linspace(0, 5, 100)
plt.plot(x, np.sin(x)+x)
```

畫像樣一點的函數圖(2)

$$f(x) = \sin(x) + x$$

#也可以用定義函數的方式呈現

```
def f(x):  
    return np.sin(x) + x  
x = np.linspace(0, 5, 100)  
plt.plot(x, f(x))
```


兩個函數畫在同一張圖

$$f(x) = \sin(x) + x$$

$$g(x) = \cos(3x)$$



```
x = np.linspace(0, 5, 100)
plt.plot(x, np.sin(x) + x)
plt.plot(x, np.cos(3*x))
```



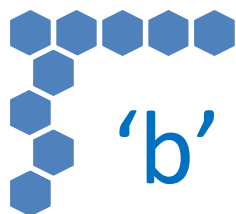
用顏色區分兩條函數

$$f(x) = \sin(x) + x$$

$$g(x) = \cos(3x)$$

```
x = np.linspace(0, 5, 100)
plt.plot(x, np.sin(x) + x, 'r')
plt.plot(x, np.cos(3*x), 'b')
```

常用的顏色串列



'b'

blue

'g'

green

'r'

red

'y'

yellow

'k'


black



能不能一次畫兩張圖？

$$f(x) = \sin(x) + x$$

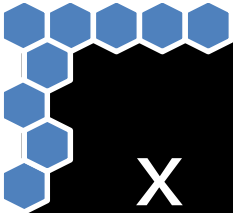
$$g(x) = \cos(3x)$$



```
x = np.linspace(0, 10, 100)
plt.subplot(211)
plt.plot(x, np.sin(x) + x)
plt.subplot(212)
plt.plot(x, np.cos(3*x))
```

#subplot()前兩個
個數字表示圖的
排列是幾個Row，
幾個Column，最
後一個數字表示
圖的位置

畫散佈圖



```
x = np.linspace(0, 5, 100)
y = np.random.randn(100)
plt.scatter(x, y, c='#daa73e')
```

#顏色也可以用RGB表示

在函數上打點

$$f(x) = \sin(x) + x$$

```
x = np.linspace(0, 5, 100)
plt.plot(x, np.sin(x)+x, marker='o',
markevery=10)
```

#每十個點就打上一個marker

各式長條圖(1)

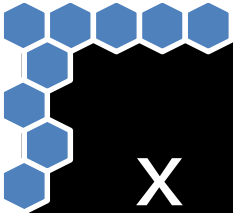


```
plt.bar(range(1, 6),  
np.random.randint(1, 30, 5))
```

#最基本的長條圖

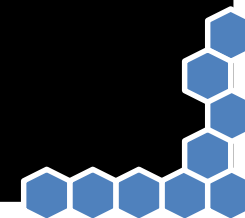


各式長條圖(2)

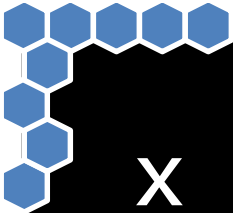


```
x = np.arange(1,6)
plt.bar(x - 0.2, [3, 10, 8, 12, 6],
width=0.4, fc='#e63946')
plt.bar(x + 0.2, [6, 3, 12, 5, 8],
width=0.4, fc='#7fb069')
```

#兩個並排的長條圖

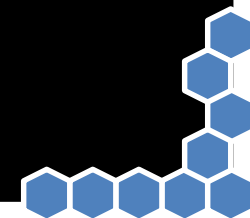


各式長條圖(3)



```
x = np.arange(0, 6)
plt.barh(x, np.random.randint(1,15,6),
fc= '#e55934' )
```

#橫式的長條圖



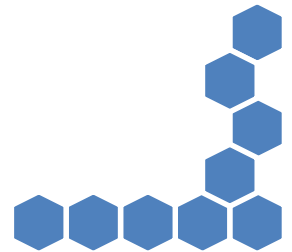
Exercise(5)



畫出

$$f(x) = \sin(2x) + \exp(x/2)$$

Hint: numpy有exponential函數
linspace -5~5 比較看得出來



讓你的程式在 雲端主機上畫圖

#將練習5的程式加上以下程式碼

```
import matplotlib.pyplot as plt  
plt.switch_backend('agg')
```

...

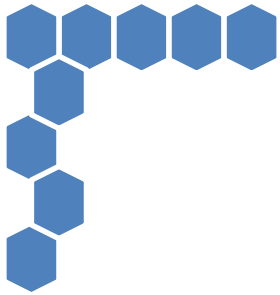
```
import gi  
gi.res(gi.base64(plt))
```

#透過gi.base64方法所繪製的圖送至雲端主機呈現



更多資訊

For more information



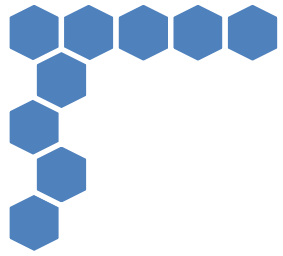
matplotlib還可以畫很多種奇怪的圖

<http://matplotlib.org/>

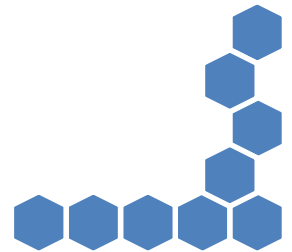
seaborn是另一種畫圖套件

<https://seaborn.pydata.org/>





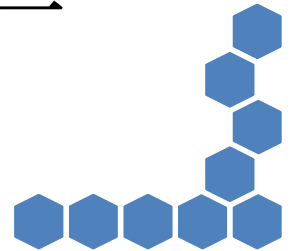
pandas



What is pandas for?



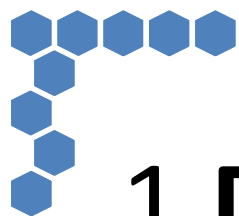
pandas是用來處理資料非常好用的工具，功能跟**microsoft excel**有點像，但是比較自由!!



引入pandas

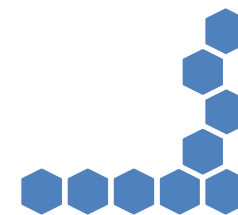
```
import pandas as pd
```

Pandas的基本結構



1.DataFrame: 可以想成一個表格。

2.Series: 表格的某一系列、某一行, 基本上就是我們以前的 **list** 或 **array**



dataframe的基本結構

	姓名	國文	英文	數學	自然	社會
0	劉俊安	9	10	15	10	10
1	胡玉華	10	10	10	8	9
2	黃淑婷	13	15	8	11	14
3	陳上紫	10	10	8	9	14
4	崔靜成	13	12	14	12	13

columns

index

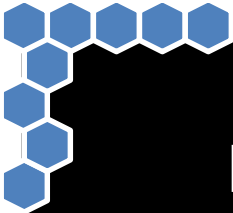
pandas可以直接讀csv檔



```
df = pd.read_csv("檔案路徑")
```

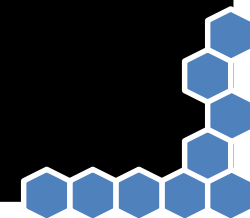
```
# 這樣就可以將csv檔變成一個dataframe
```

建立一個dataframe



```
mydata = np.random.randn(4, 3)
df1 = pd.DataFrame(mydata,
columns=list("ABC"))
df1
```

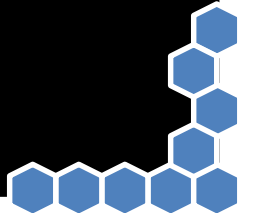
#這樣就完成了一個亂數生成的dataframe



dataframe的相接(1)



```
df2 = pd.DataFrame(np.random.randn(3, 3)  
    , columns=list("ABC"))  
df3 = pd.concat([df1, df2], axis=0)  
df3.index = range(7) #重整一下亂掉的index  
df3
```



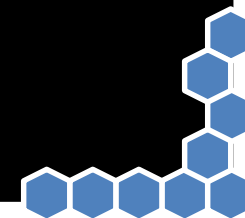
dataframe的相接(2)




```
df4 = pd.concat([df1, df2], axis=1)
```

```
df4
```

#axis=1會變橫的接



拿剛剛生成的dataframe實作(1)



```
df = df3
```

```
df.head()
```

#顯示最前面幾筆資料

```
df['A']
```


#顯示A這一行的值

```
df.A
```

#也可以這樣寫



拿剛剛生成的dataframe實作(2)



```
df.A.plot() #要畫圖就這麼簡單
```

```
df.A.hist(bins=20) #也可以畫長條圖
```

拿剛剛生成的dataframe實作(3)



```
df.A.mean()
```

#這樣就有平均了

```
df.A.std()
```

#標準差就這樣算

```
df.describe()
```

#多重願望一次滿足

拿剛剛生成的dataframe實作(4)



```
df.corr()
```

#可以算ABC彼此之間的相關係數

```
df.A.corr(df.B)
```

#AB之間的相關係數

一個重要的函數loc(1)



```
df.loc[2:3, "B":"C"]
```

#這樣就可以顯示2到3、B到C的範圍

```
df.loc[(2), ("B")]
```

#只有一列或一行就用括號!

一個重要的函數loc(2)



```
df.loc[(2), ("B")] = -1
```

```
df
```


#loc可用來改變某一個區域的值

```
df.loc[df.B > 0, "C"] = 0
```

```
df
```

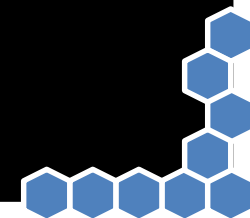
#讓B>0那一列的C值變成0

生成一個大一點的dataframe來玩(1)




```
df_grades = pd.DataFrame(  
    np.random.randint(6,16,(100,5)),  
    columns=["國文", "英文", "數學",  
            "社會", "自然"])
```

#這邊我們生成一個假的學測成績資料



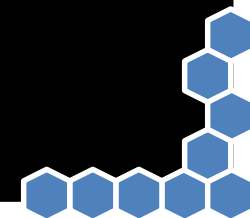
生成一個大一點的dataframe來玩(2)



```
df_grades["總級分"] = df_grades[["國文", "英文", "數學", "社會",  
"自然"]].sum(1)
```

```
# 要一列一列加起來是sum(0),  
一行一行加起來是sum(1)
```

```
#檢查一下df_grades會發現多了一行
```



生成一個大一點的dataframe來玩(3)




```
# 假設某科系想看數學*1.5+ 英文
```

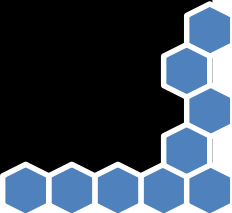
```
df_grades["主科"]=df_grades.數學*1.5+df_grades.英文  
df_grades
```

```
# df_grades現在多了兩行
```


生成一個大一點的dataframe來玩(4)



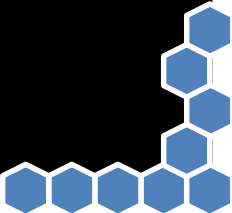
```
df_grades.sort_values(by=["總級分"],  
ascending=False).head(20)  
# 用總級分來排序(由大到小)
```



生成一個大一點的dataframe來玩(5)



```
df_grades.sort_values(by=["主科",  
"總級分"], ascending=True).head(20)  
# 先排主科再排總級分 (由小到大)
```



生成一個大一點的dataframe來玩(6)



#再檢查df_grades會發現它的順序根本沒變

```
df_grades.sort_values(by=["主科",  
"總級分"], ascending=True, inplace=True)  
df_grades.head(20)
```

#一般來說不建議用inplace=True，因為會蓋掉原本的dataframe，比較建議新建一個dataframe來存取

ETF資料介紹



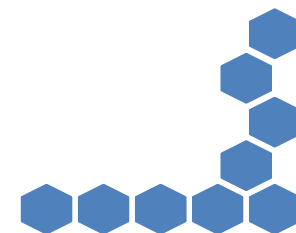
所有的ETF資料裡面都包含六種數值，分別為開盤價(Open)、收盤價(Close)、當日最高價(High)、當日最低價(Low)、成交量(Volume)、調整後收盤價(Adj_Close)。



Exercise(6)

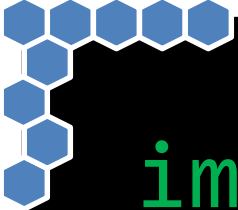


我們開啟AADR.csv來算ma10
(10天的收盤價移動平均)



Hint : 1.pd.read_csv('檔案路徑\\AAADR.csv')
2.可用for迴圈算
3.可用np.average()
4.下一頁還有更好的算法...

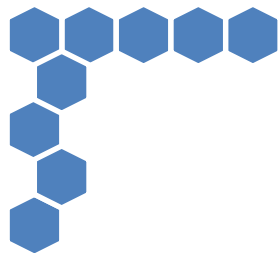
用Pandas的Series來計算Exercise(6)



```
import pandas as pd
df = pd.read_csv('AADR.csv')
df['MA10'] = pd.Series(df.iloc[::-1]
['Close']).rolling(10).mean()
```

#Series數列格式提供rolling函數可針對某個數的連續數列進行運算。
#.iloc[::-1]只是為了倒轉資料。





用線性回歸來預測

Prediction with linear regression



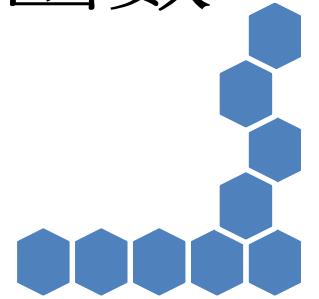
國立政治大學金融科技研究中心
智能理財與深度學習暑期訓練營



回歸的意義



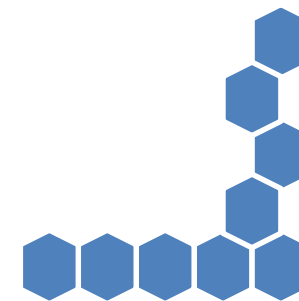
回歸就是在找一個與原始資料相去不遠函數



何謂相去不遠?



函數值與原始資料的誤差越小越好
一般我們取的是square error當誤差



換成數學的語言



假設我們有 n 筆資料 $(x_i, y_i) \quad 1 \leq i \leq n$

我們要找一個函數 $f(x)$ 使得

$$(f(x_i) - y_i)^2$$

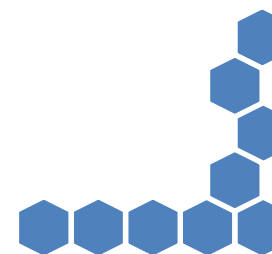
最小



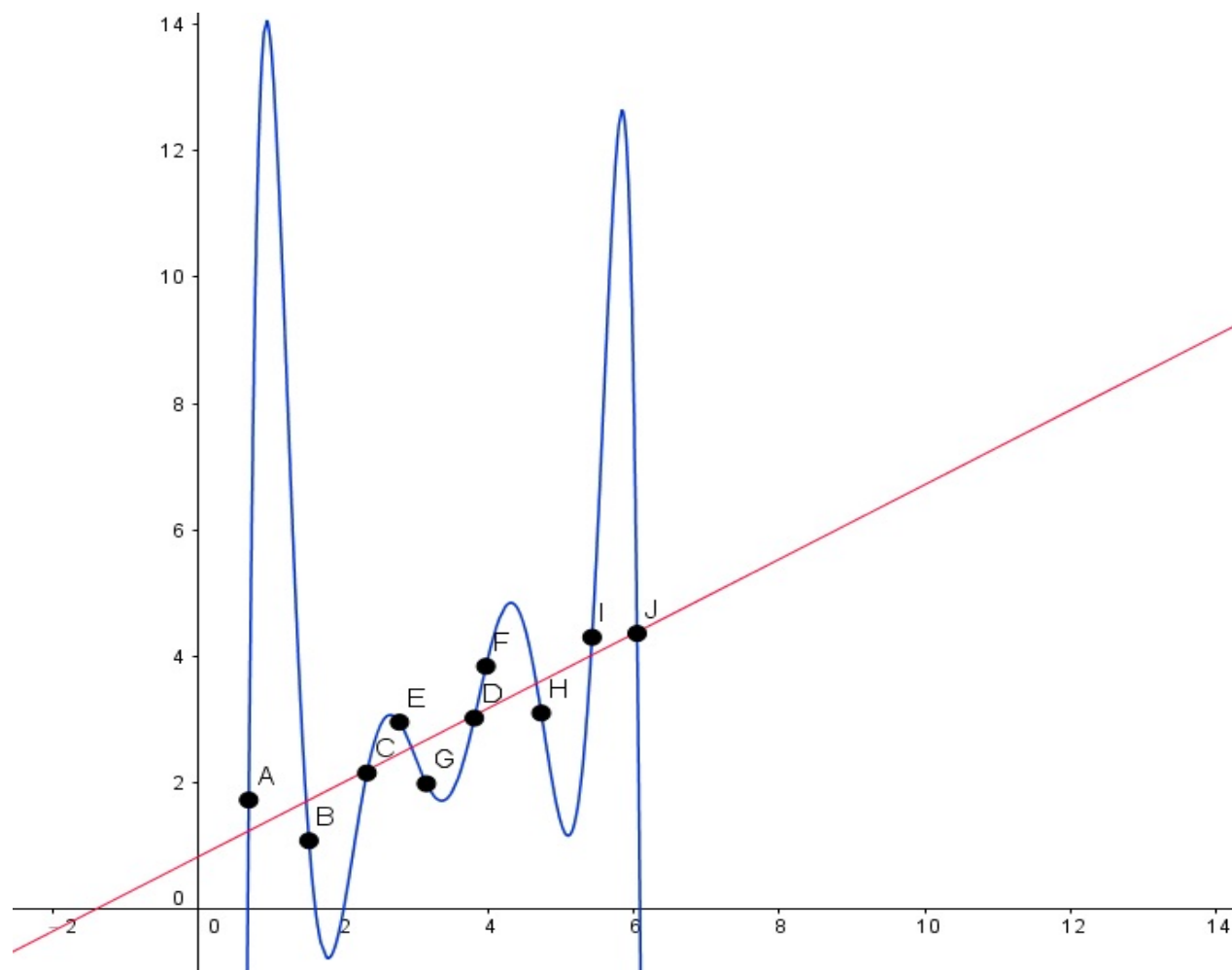
既然要最小何不通過所有點



通過所有點的函數不一定是好的函數
(大多數的情況都不是)

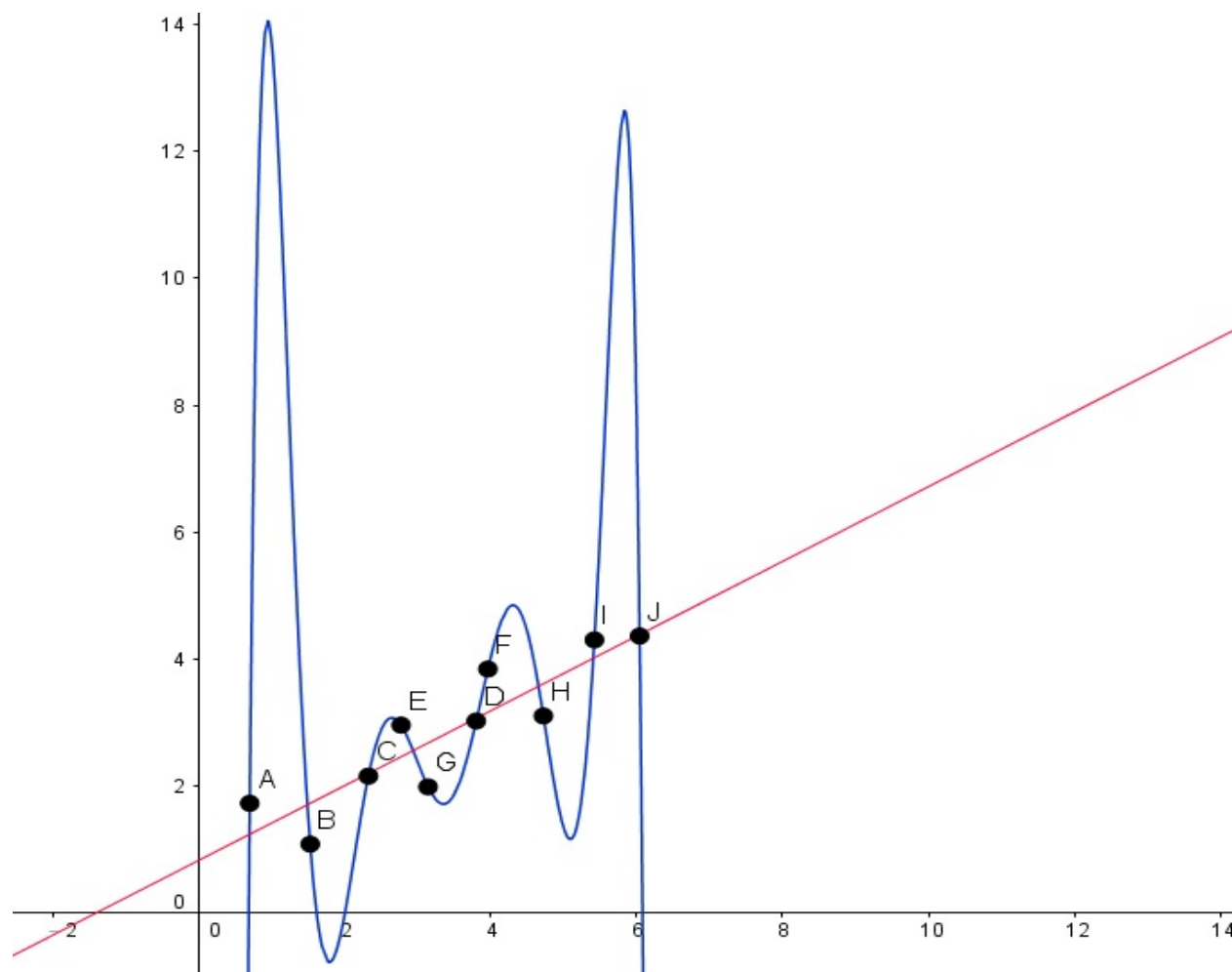


給各位選擇一個預測函數



黑點是原始資料
各位覺得紅色比較準
還是藍色比較準？

給各位選擇一個預測函數



黑點是原始資料
各位覺得紅色比較準
還是藍色比較準？

相信大部分的人會選擇紅色

可是藍色有通過所有點啊
問題在哪？



問題在我們覺得它不是我們在找的函數
我們雖然不知道我們在找的函數實際上是什麼 (知道的話就不用預測了)，但我們覺得它應該不是長那樣



那 $f(x)$ 該長什麼形狀



基本上隨便我們猜
我們覺得是什麼就是什麼

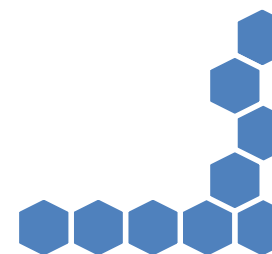


單變量線性回歸

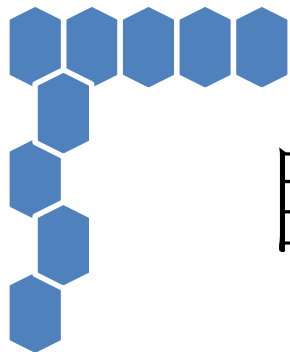


顧名思義我們假設這函數是一條線

$$f(x) = a * x + b$$



有了 $f(x)$ 的樣子就剩計算了

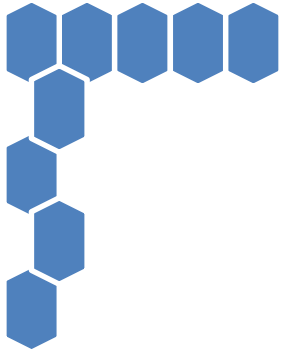


目標是

$$\min_{a,b} \sum_{i=1}^n (y_i - a - bx_i)^2$$



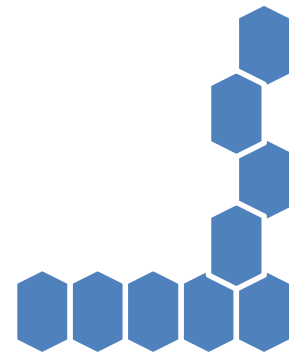
經過前人的努力



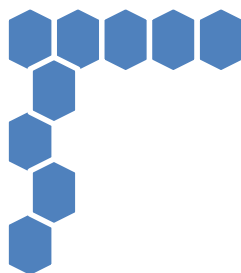
$$a = \bar{y} - b\bar{x}$$

$$b = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

\bar{x}, \bar{y} 分別為 x_i, y_i 的平均



真的要這樣算喔？

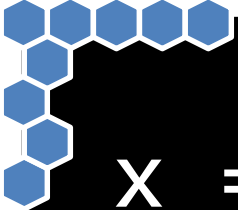


當然不會這麼殘忍

我們有sklearn可以幫我們算



實作一個線性回歸(1)



```
x = np.linspace(0, 5, 50)
y = 1.2*x + 0.8
plt.scatter(x,y)
plt.plot(x, 1.2*x+0.8, 'r')
```

我們會發現這條線完全吻合



實作一個線性回歸(2)



為了讓資料貼近現實，我們加入一些干擾

```
y = 1.2*x+0.8+0.6*np.random.randn(50)
plt.scatter(x,y)
plt.plot(x, 1.2*x + 0.8, 'r')
```


實作一個線性回歸(3.1)



```
# 開始尋找我們的目標函數
```

```
from sklearn.linear_model import LinearRegression  
model = LinearRegression()  
X = x.reshape(len(x),1)  
model.fit(X,y)
```

實作一個線性回歸(3.2)

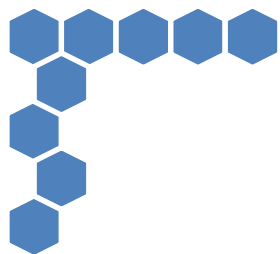


```
Y = model.predict(X)
plt.scatter(x, y)
plt.plot(x, Y, 'r')
```

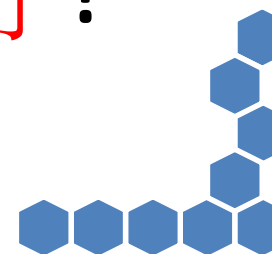
我們會發現這條線長得很像原始函數，但也有點不一樣



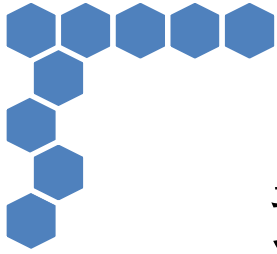
問題來了



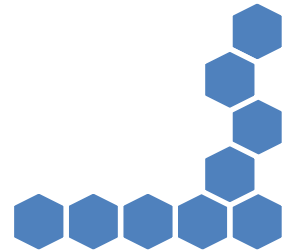
我們怎麼知道這預測函數好不好?



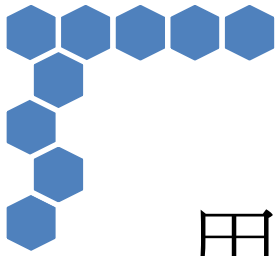
試試看就好了



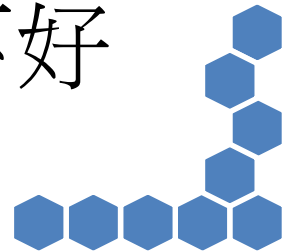
我們在原始資料動點手腳：
把它分成訓練資料和測試資料



概念上是這樣




用訓練資料來找預測函數
用測試資料來檢驗我們的預測函數好不好




注意測試資料絕對不能拿去找函數

開始檢驗我們的預測函數 (1)



```
# train_test_split這個函數可以幫我們切割原始資料
from sklearn.model_selection import train_test_split
x = np.linspace(0, 5, 50)
y = 1.2*x+0.8+0.6*np.random.randn(50)
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state=1)
```



開始檢驗我們的預測函數 (2)



看看訓練資料的樣子

```
plt.scatter(x_train, y_train)
```

再來就要開始找函數了


```
model = LinearRegression()
```

```
X_train = x_train.reshape(len(x_train),1)
```

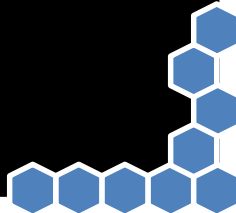
```
model.fit(X_train,y_train)
```




開始檢驗我們的預測函數 (3)



```
Y_train = model.predict(X_train)
plt.scatter(x_train, y_train)
plt.plot(x_train, Y_train, 'r')
# 紅色的就是我們的預測函數
```

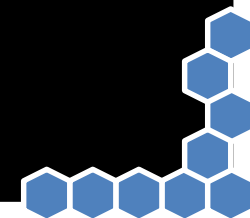


開始檢驗我們的預測函數 (4)



```
X_test = x_test.reshape(len(x_test),1)
Y_test = model.predict(X_test)
mse=np.sum((Y_test-y_test)**2)/len(y_test)
mse
```

可以算出mean square error來衡量預測函數好不好



開始檢驗我們的預測函數 (5)



也可以從圖上看點的差距

```
plt.scatter(x_test, y_test)  
plt.scatter(x_test, Y_test, c='r')
```

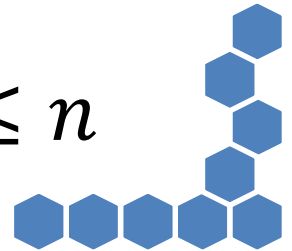
紅點為預測的結果

遇到多變量的情況怎麼辦

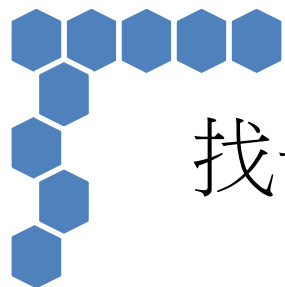


假設我們有一筆資料

$$([x_{i1}, x_{i2}, \dots, x_{ip}], y_i) \quad 1 \leq i \leq n$$



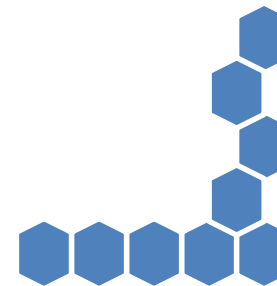
$$f(X) : R^p \rightarrow R$$



找一個函數 $f(X)$ 使得

$$(y_i - f([x_{i1}, x_{i2}, \dots, x_{ip}]))^2$$

最小

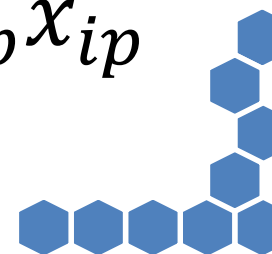


多變量線性回歸

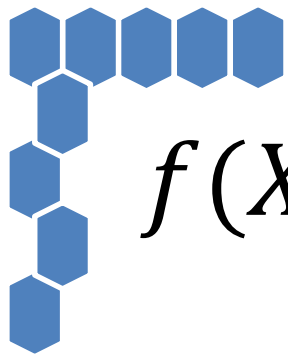


我們假設 $f(X)$ 為:

$$f(X) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip}$$



換成矩陣型式



$$f(X) = X\beta$$

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{bmatrix}$$

$$\beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$$



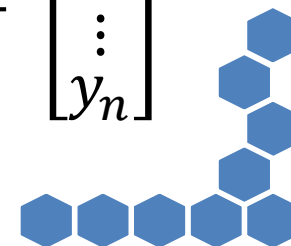
β 是多少

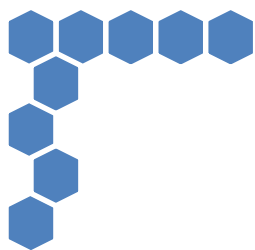


以最小平方法估計:

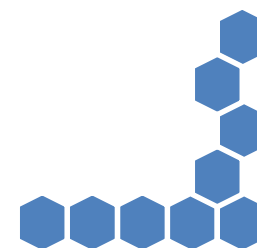
$$\beta = (X^T X)^{-1} X^T y$$

where $y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$





一樣用sklearn幫我們算



用一組假資料來實作(1)



```
x_train = [[6, 2], [8, 1], [10, 0], [14, 2], [18, 0]]
```


```
y_train = [[7], [9], [13], [17.5], [18]]
```

```
X_test = [[8, 2], [9, 0], [11, 2], [16, 2], [12, 0]]
```

```
y_test = [[11], [8.5], [15], [18], [11]]
```

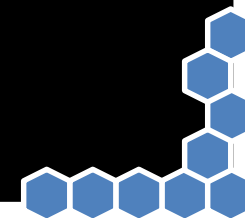
括號都不能自動省略

用一組假資料來實作(2)



```
from sklearn.linear_model import LinearRegression  
model = LinearRegression()  
model.fit(x_train, y_train)
```

這樣就訓練完了



用一組假資料來實作(3)



來看看成果如何

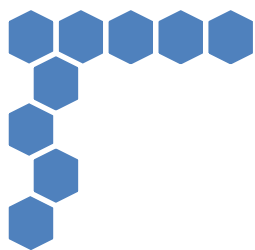
```
predictions = model.predict(X_test)
```

```
for i in range(len(predictions)):
```

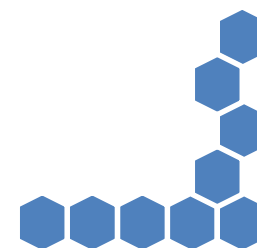
```
    print('Predicted: %s, Target: %s' %(predictions[i],  
y_test[i]))
```

```
print('R-squared: %.2f'%model.score(X_test,y_test))
```





本週作業練習



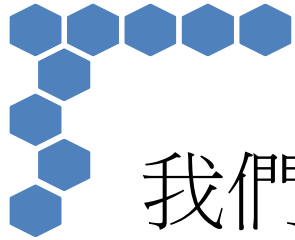
題目(1)



任選一組ETF資料，算出ma10，ma20，ma60，ma120，ma250，並將它的原始價格，ma10，ma20，ma60，ma120，ma250用不同顏色畫出來。



題目(2)



我們選其中一個叫AADR.csv的資料，用開盤價(Open)、收盤價(Close)、當日最高價(High)、當日最低價(Low)用線性迴歸的方式預測下一次(隔天)的開盤價(Open)。



提示



```
df = pd.read_csv("檔案路徑\\AADR.csv")
```

f(Open, Close, High, Low) = 下一次(隔天)的開盤價(Open)

試著用測試資料的觀念檢視預測函數

