

Big Data Analysis Platforms

SHYI-CHYI CHENG

Outline

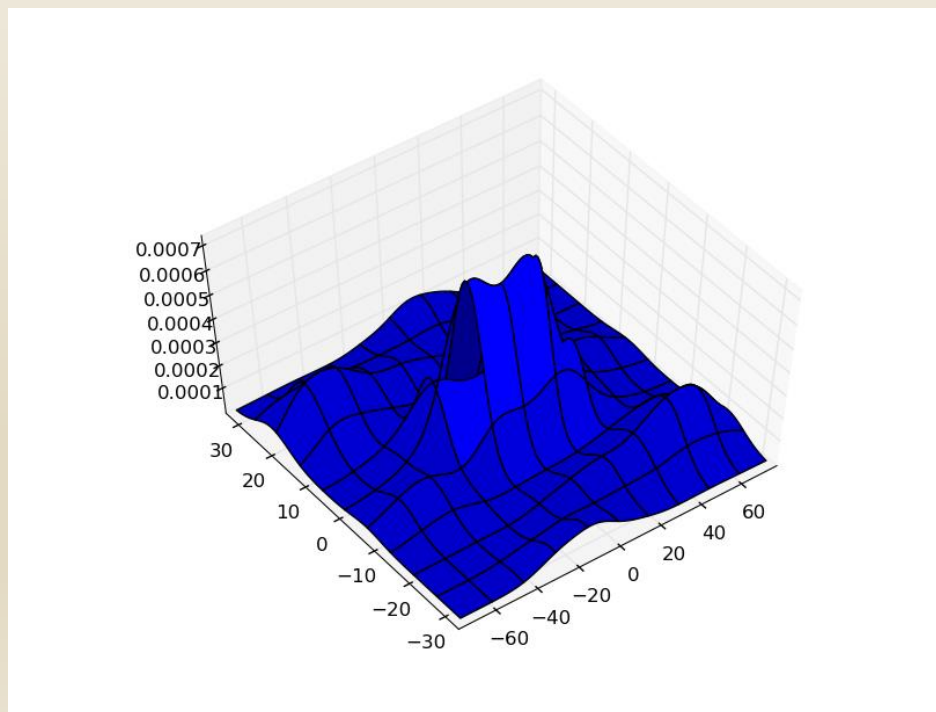
- Review of Virtual Machine (虛擬機器回顧)
- Hadoop Platform (運算分析系統架構)
- MapReduce
- Introduction to Python (Python入門簡介)
- Python Spark Platform (Python Spark運算分析架構)
- Parallel Programming With Spark

- CH2 • Python介紹
- python語言相對C/C++簡單易學，函式庫豐富
- Python在嵌入式計算領域的發展速度很快。
- Python以易於編寫、低錯誤率及可讀性著稱。
- Python，是一種物件導向、直譯式的電腦程式語言。
- 利用Python+NumPy(或SciPy)等執行數值計算(或統計處理)的高效率函式庫，讓Python在數據分析領域屹立不搖。
 - Caffe、Tensorflow、Chainer、Theano等深度學習框架均以Python為基礎架設。



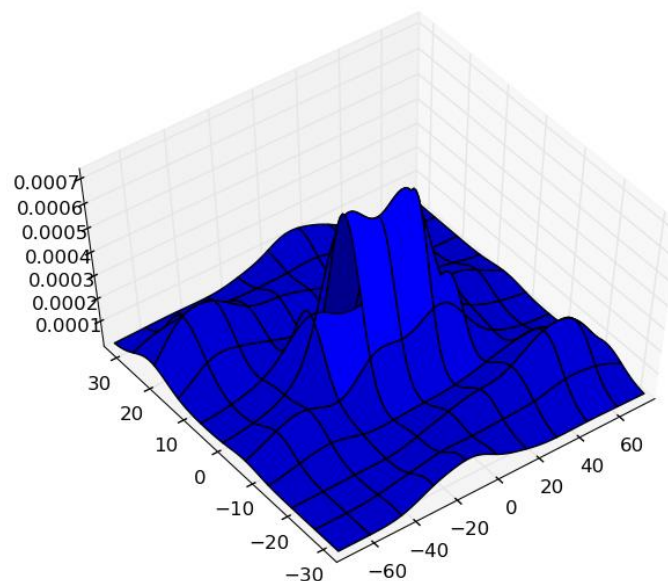
Python安裝

- Python版本:2.x與3.x (兩者不太相容)
- 基本Python外部程式:
 - NumPy提供高維陣列或矩陣的數值計算功能
 - Matplotlib是繪圖用的程式庫。
- Python在各種領域均有程式庫存在。
- 建議使用Anaconda套裝版本安裝Python
 - 請安裝最新版



Python安裝

- 安裝Anaconda套裝版本後，請繼續安裝如下機器學習相關套件：
 - `$conda install NumPy --update`
 - `$conda install matplotlib --update`
 - `$conda install SciPy`
 - `$conda install scikit-learn`
 - `$conda install pandas`

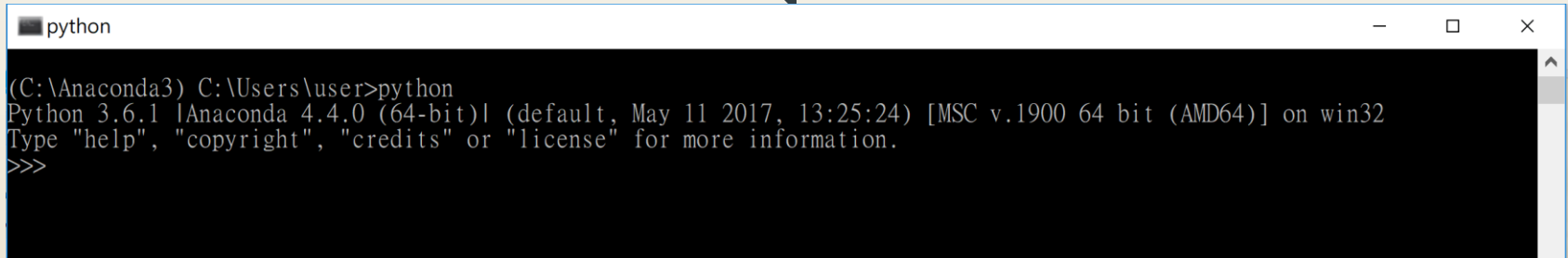


安裝Python

- 可使用以下指令查看是否有安裝Python 3.5版本

(C:\Anaconda3)C:\Users> python

需輸入在終端機的指令，會像這樣放在框框裡以\$表示。一個框框代表一行指令。



```
python
(C:\Anaconda3) C:\Users\user>python
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:25:24) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

↑出現此畫面代表已安裝

- 這個時候即進入Python3.6.1的開發環境，在>>>後輸入的指令必須為符合此版本的指令。
- 可以按Ctrl+Z離開python模式。
- 若無上述畫面，則請到下列網址下載適合個人OS的Anaconda版本進行Python安裝：

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

什麼是Python?

- Python語法簡單，與其它大多數程式設計語言使用大括弧不一樣，它使用**縮排**來定義語句塊。

```
9 if (a == 1){  
10     Serial.print('1');  
11 }
```

```
1 if a == 1:  
2     print ('1')
```

使用大括弧↑

- Python具備垃圾回收功能，能夠自動管理記憶體使用。
- 它經常被當作腳本語言用於處理系統管理任務和網路程式編寫，然而它也非常適合完成各種高階任務。
- Python虛擬機本身幾乎可以在所有的作業系統中運行。
- Python的官方直譯器是CPython，該直譯器用C語言編寫，是一個由社群驅動的自由軟體，目前由Python軟體基金會管理。
- Python支援命令式程式設計、物件導向程式設計、函數式編程、面向側面的程式設計、泛型編程多種編程範式。

使用縮排↑

執行Python程式

- Python的程式執行方式大致可分為兩種：

方式一.

說明：在終端機的模式下，進入 Python3.6.1的開發環境，直接下指令執行。

範例.

步驟1：在終端機下開啟Python

```
$ python
```

步驟2：進入Python模式後，直接輸入程式然後按Enter執行

執行結果：

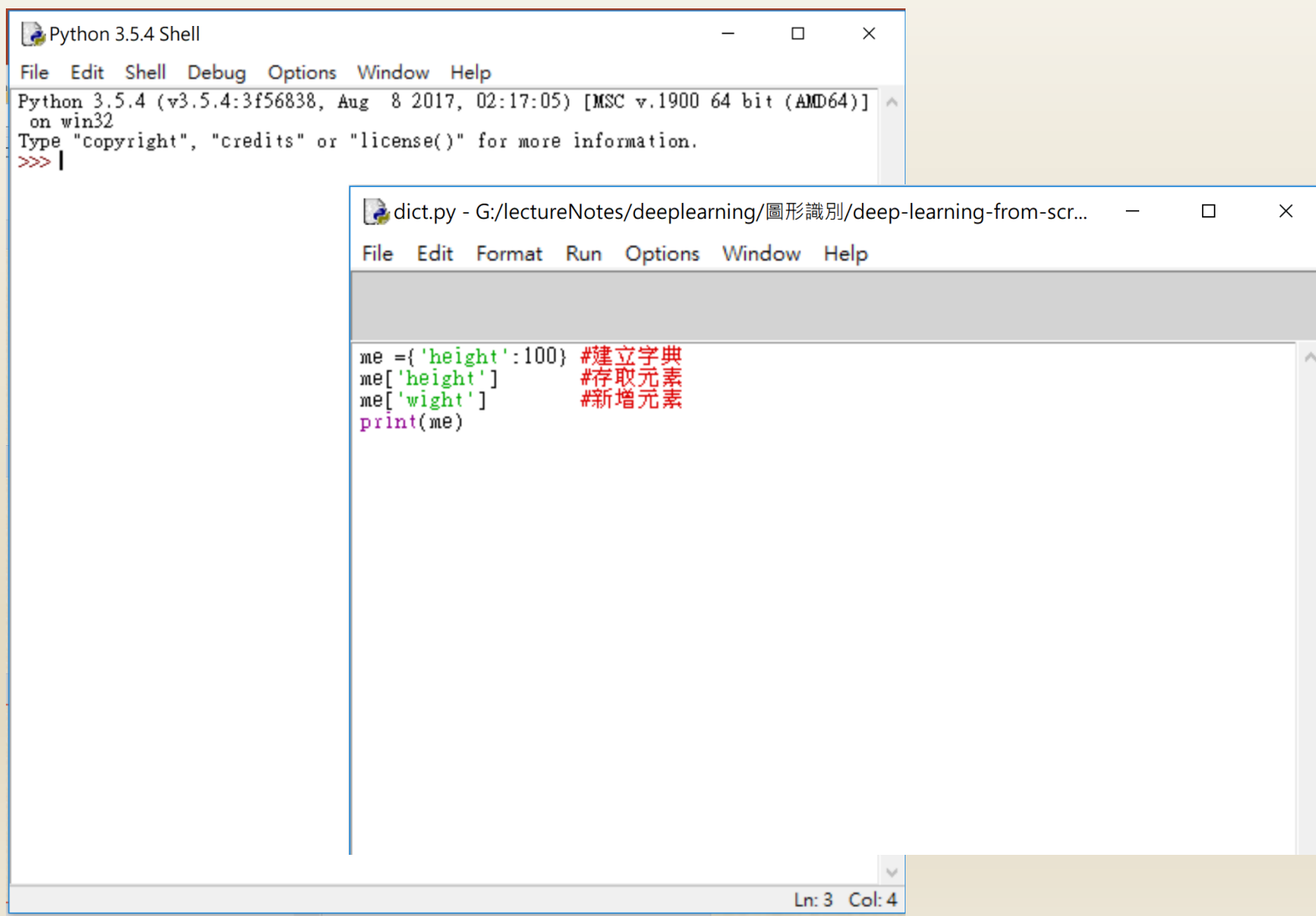
```
>>> print('hello world')
```

這邊的動作是透過print函數，把括號內的文字列印出來。

執行Python程式

方式二.

說明：透過Python IDE工具，先把程式寫在存文字檔案中，並存成.py檔。到時候我們再告訴Python程式，去讀取這個檔案並且執行。



Python語言_資料型態

- 在Python的程式語言之中，資料可以以變數加以儲存，並可透過Python程式語言中的數學計算處理。

語法：變數名=初始值

範例程式：

```
a=33  
b="abc"  
c=33.4  
print("Hello")  
print(a)  
print b  
print(c)  
print a+c
```

定義變數**a**為整數**33**

定義變數**b**為文字**abc**

定義變數**c**為浮點數**33.4**

顯示()內資料

顯示變數**a+c**結果

執行結果：

```
Hello  
33  
abc  
33.4  
66.4
```

Python語言_數學運算

- Python對數學的運算處理方法，跟一般C語言相同，可使用下面的範例程式了解。

數學符號	功能解釋	範例	答案
+	加法	3+2	5
-	減法	3-2	1
*	乘法	3*2	6
/	除法	5/3	1
%	取餘數	5%3	2
**	執行指數	3**2	9
//	除法	9//2	4

範例程式：

```
a=4  
b=2.2  
c="3.3"  
print(a+b)  
print(a<<1)  
print(a>>1)  
print(a%3)
```

執行結果：

```
6.2  
8  
2  
1
```

Python語言_陣列

- 陣列在程式語言中，是一種相同資料型態一直重複的資料結構，將連續、相同的資料型態放在連續的記憶體空間中。它可以讓程式碼表現更為簡單，加快開發速度。

語法1： 變數=['字串', '字串',.....]

範例程式：

```
a=['Apple', 'Banana','Watermelon']  
Print a[1]
```

執行結果： Banana

語法2： 變數=[[數字,數字,...],[數字,數字,...]]

範例程式：

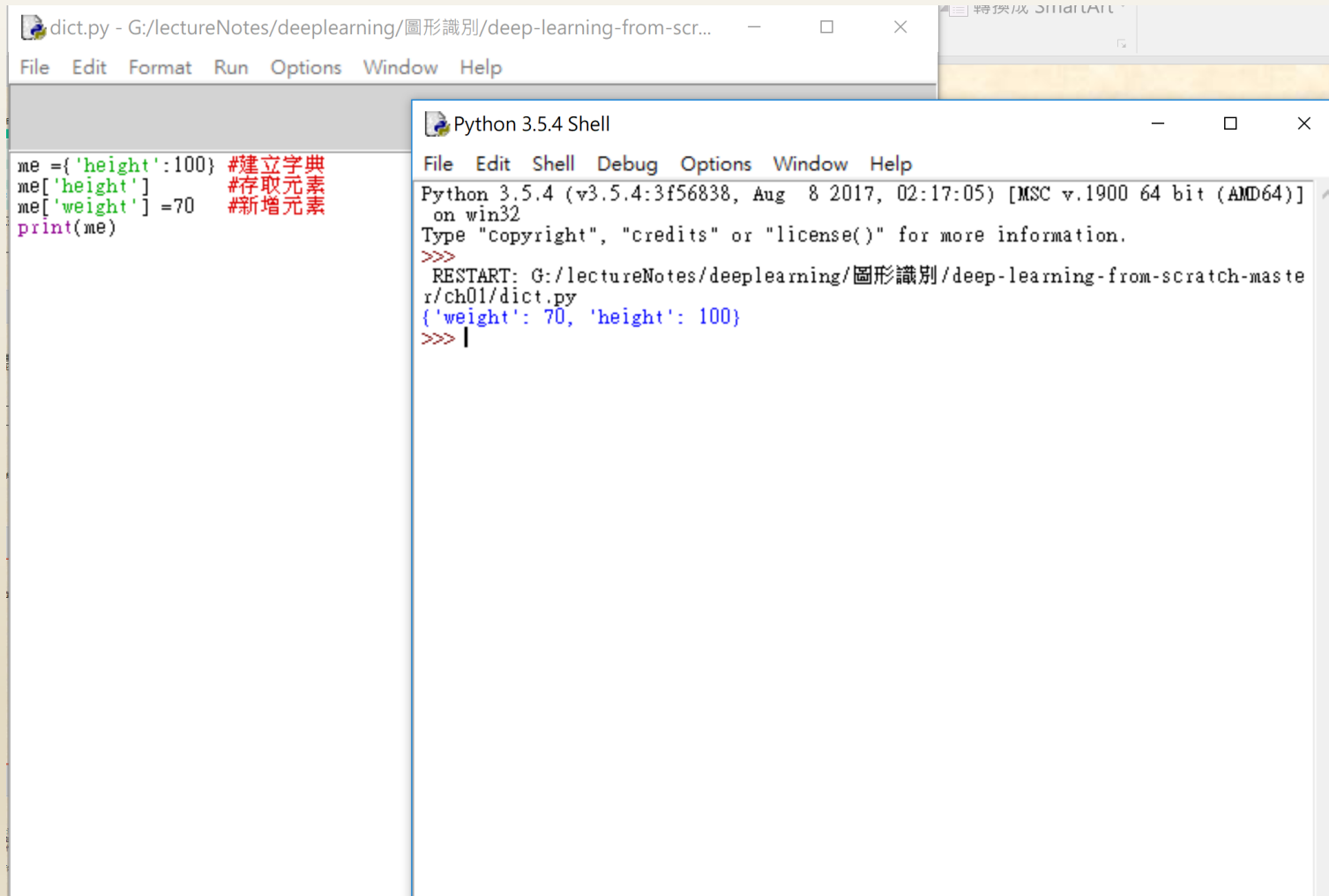
```
a=[[11,22,33],  
    [44,55,66],  
    [77,88,99]]  
Print a[1][0]
```

執行結果： 44

Python語言_字典型態

- 字典(Dictionary)是將key跟value配對，再儲存資料。

範例程式：



The screenshot displays a Python IDE with two windows. The left window, titled 'dict.py', contains the following code:

```
me = {'height': 100} #建立字典  
me['height']         #存取元素  
me['weight'] = 70    #新增元素  
print(me)
```

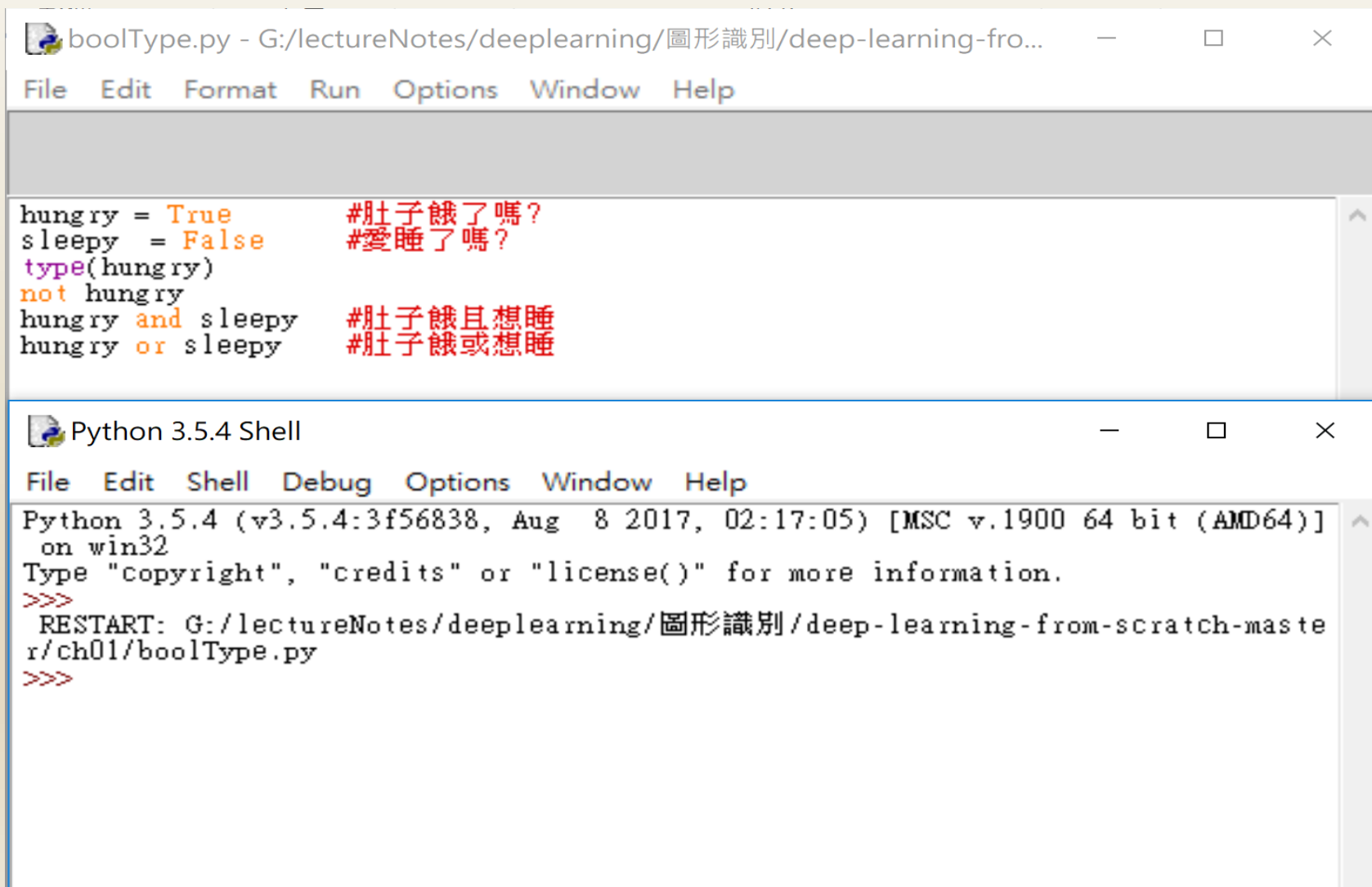
The right window, titled 'Python 3.5.4 Shell', shows the execution output:

```
Python 3.5.4 (v3.5.4:3f56838, Aug 8 2017, 02:17:05) [MSC v.1900 64 bit (AMD64)]  
on win32  
Type "copyright", "credits" or "license()" for more information.  
>>>  
RESTART: G:/lectureNotes/deeplearning/圖形識別/deep-learning-from-scratch-master/ch01/dict.py  
{'weight': 70, 'height': 100}  
>>> |
```

Python語言_布林型態

- 布林值只有True跟False。

範例程式：



The image shows two windows from a Python IDE. The top window, titled 'boolType.py', contains a Python script with variables 'hungry' and 'sleepy' set to 'True' and 'False' respectively. It then uses 'type()' to check their types, 'not' to invert 'hungry', and 'and'/'or' to combine them. The bottom window, titled 'Python 3.5.4 Shell', shows the execution of the script, including the version information and the file path.

```
boolType.py - G:/lectureNotes/deeplearning/圖形識別/deep-learning-fro...
File Edit Format Run Options Window Help

hungry = True           #肚子餓了嗎?
sleepy = False          #愛睡了嗎?
type(hungry)
not hungry
hungry and sleepy       #肚子餓且想睡
hungry or sleepy        #肚子餓或想睡

Python 3.5.4 Shell
File Edit Shell Debug Options Window Help
Python 3.5.4 (v3.5.4:3f56838, Aug 8 2017, 02:17:05) [MSC v.1900 64 bit (AMD64)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: G:/lectureNotes/deeplearning/圖形識別/deep-learning-from-scratch-master/ch01/boolType.py
>>>
```

Python語言_if...else條件判斷

- If語句與比較運算子(==、<、>...)一起用於檢測某個條件是否達成。
- If語句語法：

If 某變數 > 10 : 某動作

此語句為當某變數的值大於10時，執行某動作。
- 換句話說，只要if後的條件為真，則繼續執行if內程式；若為假，則跳過。

- 語法1：

If 條件判斷語句： 要做的事情1

- 語法2：

If 條件判斷語句： 要做的事情1 else: 要做的事情2

- 語法3：

If 條件判斷語句1： 要做的事情1 elif 條件判斷語句2： 要做的事情2 else: 其他要做的事情
--

Python語言_if...else條件判斷

- 條件判斷語句：

`x==y`(x等於y)

`x != y`(x不等於y)

`x < y`(x小於y)

`x > y`(x大於y)

`x<=y`(x小於等於y)

`x>=y`(x大於等於y)

- 範例程式：

```
a=4
If  a==1 :
    print('1')
elif a==2 :
    print('2')
else:
    print('3')
```

- 執行結果：

3

Python語言_range範圍

- 在Python迴圈中常會用到range範圍的寫法，因此在介紹迴圈之前，須先了解什麼是range範圍。

語法1：`range(範圍)`

範例程式：`a=range(10)
print(a)`

執行結果：`[0,1,2,3,4,5,6,7,8,9]`

語法2：`range(範圍開始,範圍結束)`

範例程式：`a=range(2,6)
print(a)`

執行結果：`[2,3,4,5]`

語法3：`range(範圍開始,範圍結束,每次相差)`

範例程式：`a=range(0,6,2)
print(a)`

執行結果：`[0, 2, 4]`

Python語言_for迴圈

- For語句用於重複執行一段程式，一般而言程式是在{ }內，但Python沒有{ }，它是透過空白來代替，所以同一個範圍內的程式，前面都要有相同的空白行數。

語法1： For 變數 in range(範圍)

範例程式：

```
for x in range(10):  
    print x  
    print "end"
```

執行結果：

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
end
```

語法2： For 變數 in range(範圍開始,範圍結束)

範例程式：

```
for x in range(2,6):  
    print x  
    print "end"
```

執行結果：

```
2  
3  
4  
5  
end
```

Python語言_for迴圈

語法3 : For 變數 in range(範圍開始,範圍結束,每次相差)

範例程式 :

```
for x in range(6,0,-2):  
    print x  
    print "end"
```

執行結果 :

```
6  
4  
2  
end
```

語法4 : For 變數 in 矩陣

範例程式 :

```
a=['Apple', 'Banana']  
for x in a:  
    print x  
    print "end"
```

執行結果 :

```
Apple  
Banana  
end
```

Python語言_while迴圈

while語句用於重複執行一段程式，而程式是放在相同空白行數的程式碼，while迴圈會無限的循環，直到括弧內的判斷式為否。

語法1：`while` 判斷的條件：
要做的事情

語法2：`while` 判斷的條件：
要做的事情

範例程式：

```
x=0
while x<5:
    print x
    x=x+1
print "end"
```

執行結果：

```
0
1
2
3
4
end
```

範例程式：

```
x=0
while x<=20:
    print x
    x=x+5
print "end"
```

執行結果：

```
0
5
10
15
20
end
```

Python語言_while迴圈

語法3：同上，透過while迴圈印出九九乘法表

範例程式：

```
x=0
while x<9:
    x=x+1
    y=1
    while y<10:
        print str(x)+"*"+str(y)+"="+str(x*y)
        y=y+1
    print "end"
```

執行結果：

```
1*1=1
1*2=2
.
.
.
9*7=63
9*8=72
9*9=81
end
```

Python語言_def函數

在Python中要定義函式，是使用def來定義，方式如下：

語法1：`def 函數名稱():`——建立函數，並在程式中呼叫該函數。

範例程式：

```
def fun1():  
    print "this is function1"  
fun1()
```

執行結果：`this is function1`

語法2：`def 函數名稱(參數):`——建立函數，並代參數進入函數中。

範例程式：

```
def fun2(num):  
    print "this is function2="+str(num);  
fun2(100)
```

執行結果：`this is function2=100`

Python語言_def函數

在Python中要定義函式，是使用def來定義，方式如下：

語法3：
`def 函數名稱():`
 要做的事情
 return 回傳值

建立函數的動作，代參數進入函數中，並回傳資料。

範例程式：
`def fun3(num1,num2):`
 `print "this is function3";`
 `return num1+num2`

 `print fun3(1,2)`

執行結果：
this is function3
3

Python語言_class

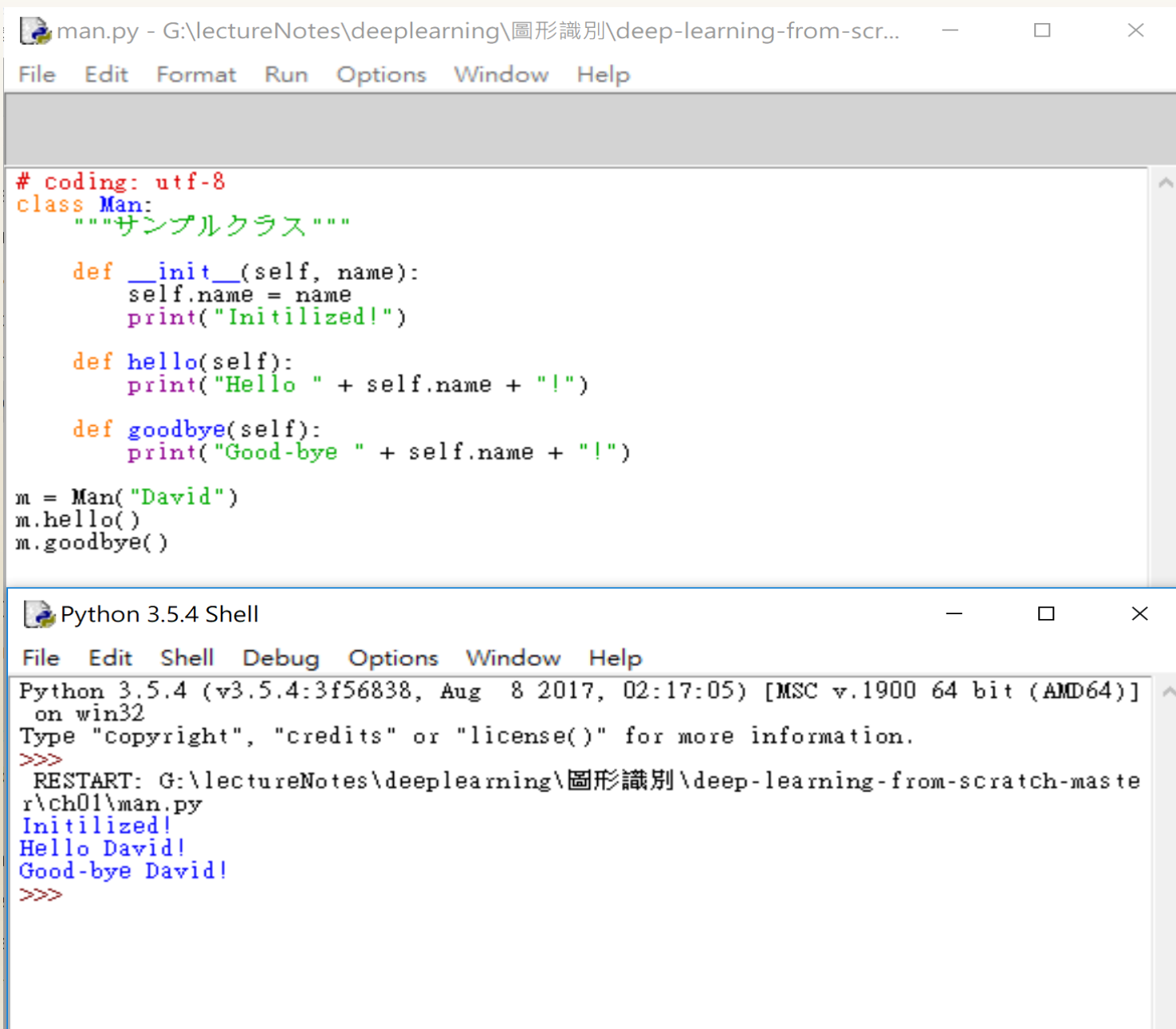
在Python中要自行定義類別，是使用class來定義，方式如下：

語法：

```
class 類別名稱:  
    def __init__(self,參數,...)    #建構子  
    ...  
    def 方法名稱1(self,參數,...)  #方法1  
    ...  
    def 方法名稱2(self,參數,...)  #方法2  
    ...
```


Python語言_class

範例程式：



The image shows a screenshot of a Python IDE. The top window, titled 'man.py', contains a Python class definition for 'Man'. The code is as follows:

```
# coding: utf-8
class Man:
    """サンプルクラス"""

    def __init__(self, name):
        self.name = name
        print("Initilized!")

    def hello(self):
        print("Hello " + self.name + "!")

    def goodbye(self):
        print("Good-bye " + self.name + "!")

m = Man("David")
m.hello()
m.goodbye()
```

The bottom window, titled 'Python 3.5.4 Shell', shows the execution of the code. The output is as follows:

```
Python 3.5.4 (v3.5.4:3f56838, Aug 8 2017, 02:17:05) [MSC v.1900 64 bit (AMD64)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>>
RESTART: G:\lectureNotes\deeplearning\圖形識別\deep-learning-from-scratch-master\ch01\man.py
Initilized!
Hello David!
Good-bye David!
>>>
```

Python語言_NumPy

NumPy是外部程式，使用前須先載入，方式如下：

語法：

```
>>>import numpy as np
```

範例程式1：

python

```
(C:\Anaconda3) C:\Users\user>python
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:25:24) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> x= np.array([1.0,2.0,3.0])
>>> print(x)
[ 1.  2.  3.]
>>> type(x)
<class 'numpy.ndarray'>
>>> .
```

Python語言_Numpy

NumPy是外部程式，使用前須先載入，方式如下：

語法：

```
>>>import numpy as np
```

範例程式2：

```
>>> x= np.array([1.0,2.0,3.0])
>>> y=np.array([2.0,4.0,6.0])
>>> x+y
array([3.0,6.0,9.0])
>>> x-y
array([-1.0,-2.0,-3.0])
>>> x*y          # element-wise product
array([2.0,8.0,18.0])
>>> x/y
array([0.5,0.5,0.5])
```

Python語言_Numpy

Numpy是外部程式，使用前須先載入，方式如下：

語法：

```
>>>import numpy as np
```

範例程式3：

```
>>> A= np.array([[1,2],[3,4]])
>>> print(A)
[[1, 2],
 [3, 4]]
>>> A.shape
(2,2)
>>> A.dtype
dtype( 'int64' )
>>> B= np.array([[3,0],[0,6]])
>>> A+B
array([[4,2],
       [3, 0]])
```

```
>>> A*B
array([[3,0],
       [0,24]])
>>> A*10
array([[10, 20],
       [30, 40]])
>>> A[0]
array([1,2])
>>> A[0][1]
1
>>> for row in A
      print(row)

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

Python語言_Numpy

NumPy是外部程式，使用前須先載入，方式如下：

語法：

```
>>>import numpy as np
```

範例程式4：

```
>>> A= np.array([[1,2],[3,4]])  
>>> A= A.flatten()  
>>> print(A)  
[1, 2, 3, 4]  
>>> A > 2  
array([False, False, True, True], dtype = bool)  
>>> A[A>2]  
array([3, 4])
```

Python語言_Matplotlib

Matplotlib是外部程式，使用前須先載入，方式如下：

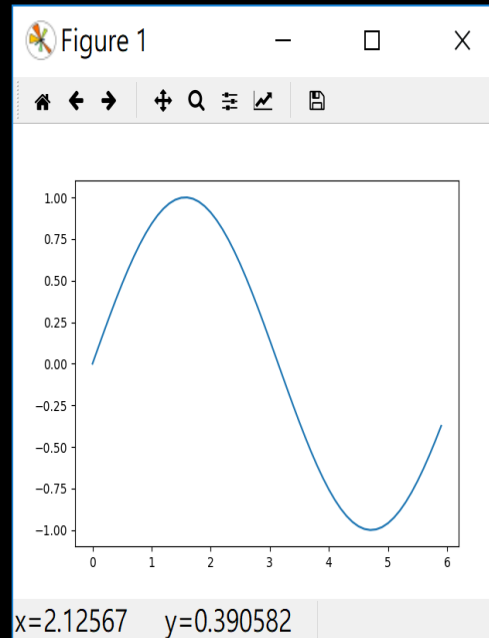
語法：

```
>>>import numpy as np
>>>import matplotlib.pyplot as plt
```

Anaconda Prompt - python

```
(C:\Anaconda3) C:\Users\user>python
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:25:24) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
```

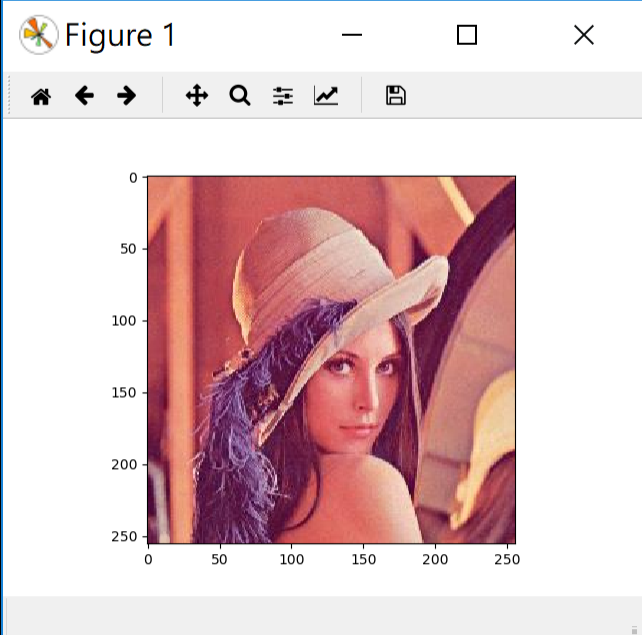
```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> x = np.arange(0, 6, 0.1) # 0から6まで0.1刻みで生成
>>> y = np.sin(x)
>>> plt.plot(x, y)
[<matplotlib.lines.Line2D object at 0x0000029D99DD4438>]
>>> plt.show()
```



Python語言_Matplotlib

Matplotlib是外部程式，使用前須先載入，方式如下：

```
Anaconda Prompt - python
(C:\Anaconda3) G:\lectureNotes\deeplearning\圖形識別>python
Python 3.6.1 |Anaconda 4.4.0 (64-bit)| (default, May 11 2017, 13:25:24) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> # coding: utf-8
... import matplotlib.pyplot as plt
>>> from matplotlib.image import imread
>>>
>>> img = imread('G:/DATASET/lena.png')
>>> plt.imshow(img)
<matplotlib.image.AxesImage object at 0x000002BFD55B9390>
>>>
>>> plt.show()
```



Any Questions?