

# 人工智慧系統

# Artificial Intelligence System

許志宇Chin-Yu Hsu

October 4, 2018

# 聽眾Audience

- 大三 third year ungraduate
- 基礎課程 Basic curriculum
- 61 students

# Objective

- 引導您了解如何自學。Guide you to know how to learn by yourself.
- 人工智慧系統（ Artificial Intelligence System ）的歷史是什麼？
- What is the history of the 人工智慧系統(AIS)?
  - Who are the important people AIS? Newton, Lagrange, Gauss and Euler.
  - What are the important events of AIS?
  - Where is the location of AIS?
  - Which objects are related to AIS?
  - When is AIS popular?

# 材料Materials

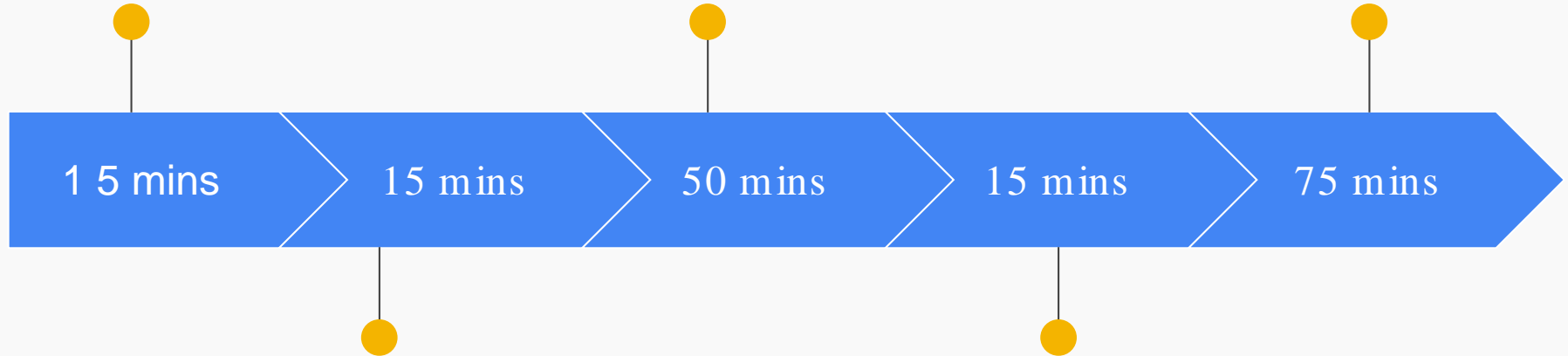
- What is AIS
- Python Language
- Tensor flow
- Project and report

程序Procedure

Introduction

Step by step to learn  
small concepts by  
doing

A project and report  
should be done by  
yourself



Demo:How to solve  
problem?

Conclusions

# 家庭作業1/Homework1

Learning efficiency depends on motivation

1. Why do you choose this course?
2. What kind of jobs for AIS?
3. What kind of jobs you want?

How to learn knowledge of AIS efficiently?

1. Python programming
2. Mathematics
3. Tensorflow, scikit learn and, keras

# Python programming

## Learning topics

1. Using Python as a Calculator
  - a. Arithmetic operations
2. Variables
3. if Statements
4. for Statements
5. The range() Function
6. Executing modules as scripts
7. Mathematics
8. Python Functions - W3Schools

## Learning Resources

1. [Welcome to Python.org](#)
2. [Download](#)
3. [Python For Beginners](#)
4. [The Python Tutorial](#)
5. [IntroductoryBooks](#)
6. [Python Functions - W3Schools](#)
7. [A Visual Introduction to Python](#)
8. [Try Jupyter](#)



# Python programming

## Learning topics

1. Python Functions

```
def my_function():  
    print("Hello from a function")
```

```
my_function()
```

## Learning Resources

1. [Python Functions - W3Schools](#)
2. [Run web python or Anaconda](#)
3. <http://jupyter.org/try>
4. [https://hub.mybinder.org/user/jupyterlab-jupyterlab-demo-yqivt6ba/lab#Integration-\(scipy.integrate\)](https://hub.mybinder.org/user/jupyterlab-jupyterlab-demo-yqivt6ba/lab#Integration-(scipy.integrate))

# Python programming

```
#factor function
def functiona(number):
    result=1
    for x in range(1,number+1):
        result=x*result
    print ('result of a:'+str(result))
```

```
functiona(36)
```

1. <http://jupyter.org/try>

# Python programming

```
#suming function
def functionb(number):
    result=0
    for x in range(0,number+1,1):
        result=x+result
    print ('result of b:'+str(result))
```

```
functionb(36)
```

1. <http://jupyter.org/try>

# Python programming

```
def myc(a,b):  
    if(a<b):  
        print("a smaller b")  
    elif(a==b):  
        print("a equals b")  
    else:  
        print("a bigger b")  
    return
```

myc(29,3)

myc(3,29)

myc(3,3)

1. <http://jupyter.org/try>

# Python programming

```
# %load sin_graph.py
import numpy as np
import matplotlib.pyplot as plt
```

```
# data
x = np.arange(0, 6, 0.1)
y = np.sin(x)
```

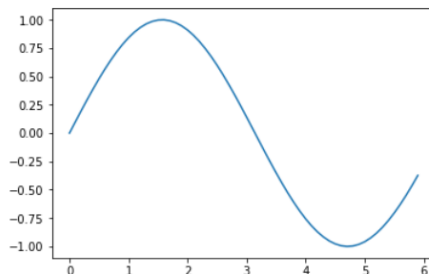
```
# plot graph
plt.plot(x, y)
plt.show()
```

<http://jupyter.org/try>

```
In [3]: # %load sin_graph.py
import numpy as np
import matplotlib.pyplot as plt

# data
x = np.arange(0, 6, 0.1)
y = np.sin(x)

# plot graph
plt.plot(x, y)
plt.show()
```



# Python programming

```
# coding: utf-8
```

```
import numpy as np
```

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

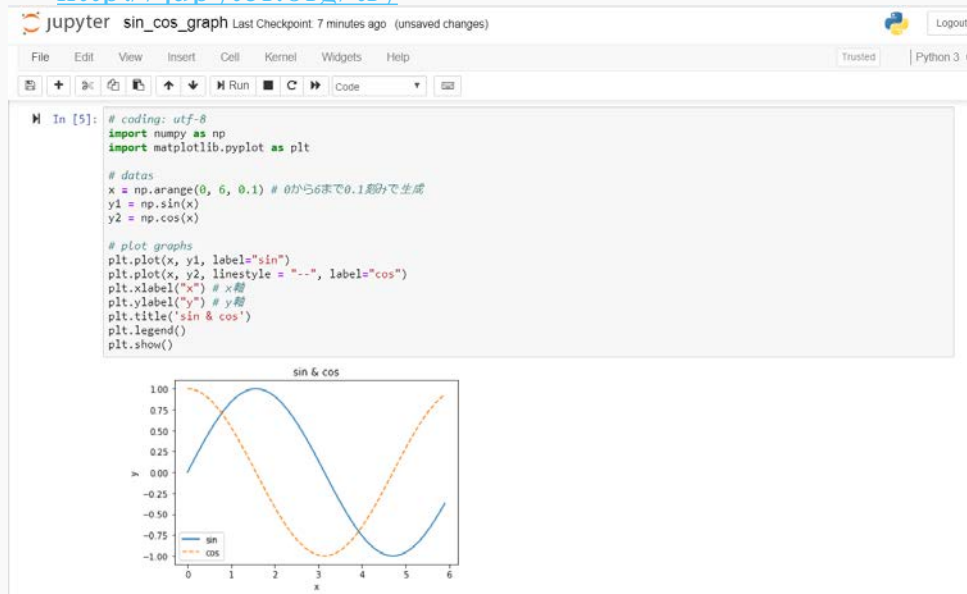
```
# datas
```

```
x = np.arange(0, 6, 0.1) # 0から6まで0.1刻みで  
生成
```

```
y1 = np.sin(x)
```

```
y2 = np.cos(x)
```

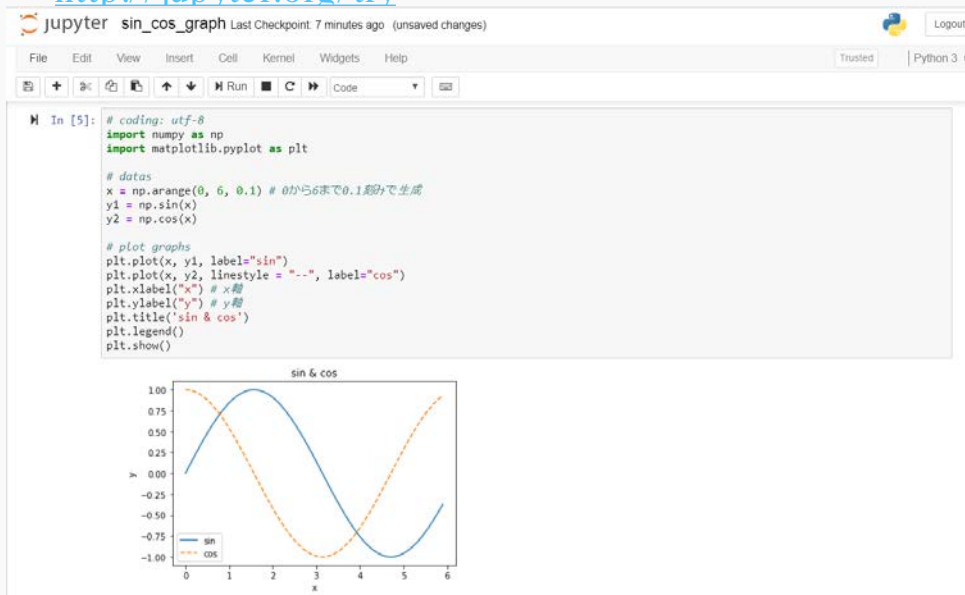
<http://jupyter.org/try>



# Python programming

```
# plot graphs
plt.plot(x, y1, label="sin")
plt.plot(x, y2, linestyle = "--", label="cos")
plt.xlabel("x") # x軸
plt.ylabel("y") # y軸
plt.title('sin & cos')
plt.legend()
plt.show()
```

<http://jupyter.org/try>



# Python programming

<http://jupyter.org/try>

```
class Man:
```

```
    """functions of class """
```

```
    def __init__(self, name):
```

```
        self.name = name
```

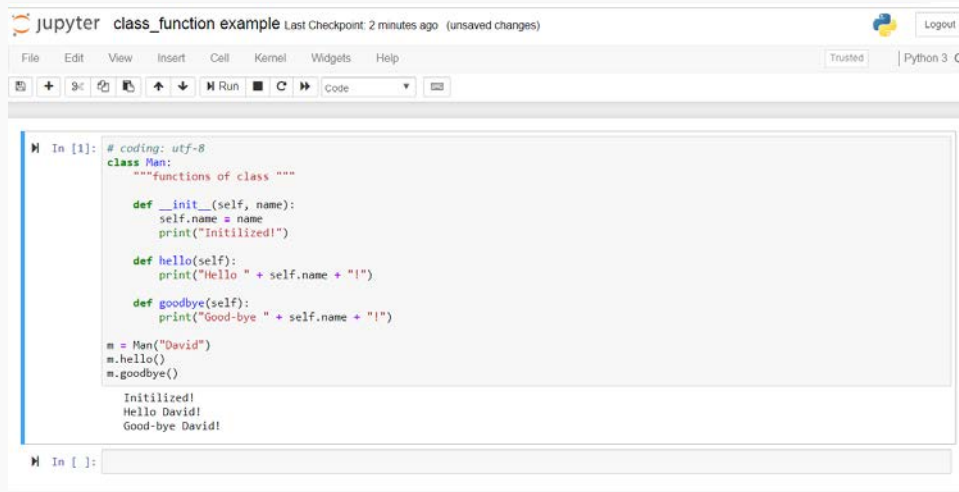
```
        print("Initilized!")
```

```
    def hello(self):
```

```
        print("Hello " + self.name + "!")
```

```
    def goodbye(self):
```

```
        print("Good-bye " + self.name + "!")
```



The screenshot shows a Jupyter Notebook window titled "class\_function example". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and saving. The code editor displays the following Python code:

```
# coding: utf-8
class Man:
    """functions of class """
    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()
```

Below the code editor, the output of the execution is shown:

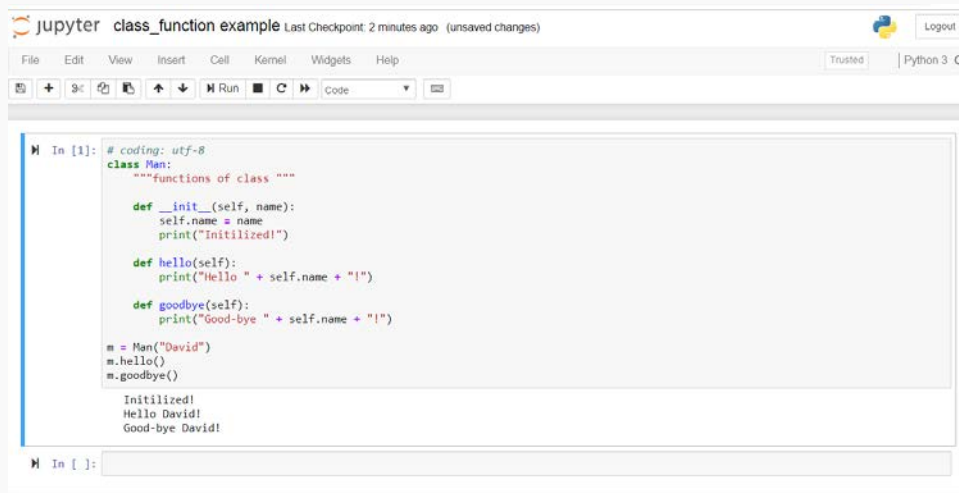
```
Initilized!
Hello David!
Good-bye David!
```



# Python programming

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

<http://jupyter.org/try>



The screenshot shows a Jupyter Notebook window titled "jupyter class\_function example". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a status bar at the bottom indicating "Python 3". The main area contains a code cell with the following Python code:

```
# coding: utf-8  
class Man:  
    """functions of class """  
  
    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()
```

Below the code, the output of the execution is displayed:

```
Initilized!  
Hello David!  
Good-bye David!
```

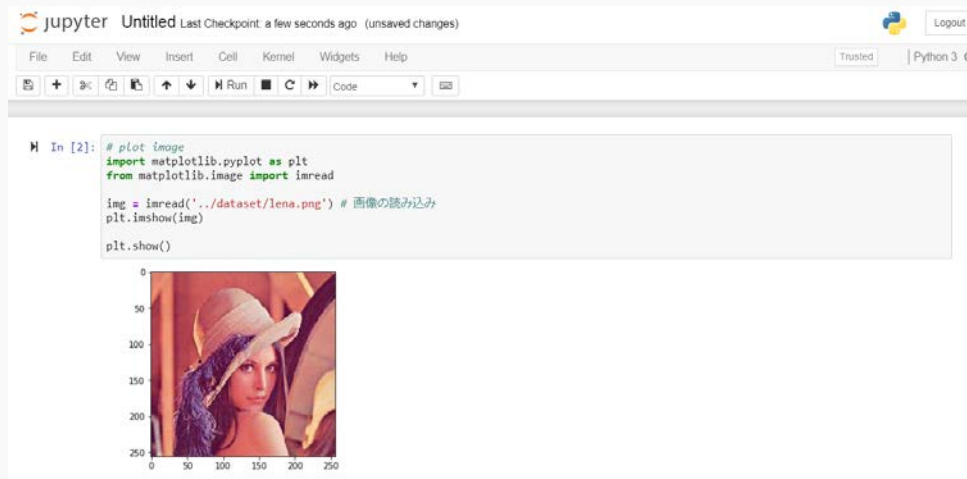
# Python programming

```
# plot image
import matplotlib.pyplot as plt
from matplotlib.image import imread
```

```
img = imread('../dataset/lena.png') # 画像の読み込み
plt.imshow(img)
```

```
plt.show()
```

<http://jupyter.org/try>



# Python programming

Question:

Design a Class

With functions

plotsin()

plotsincos()

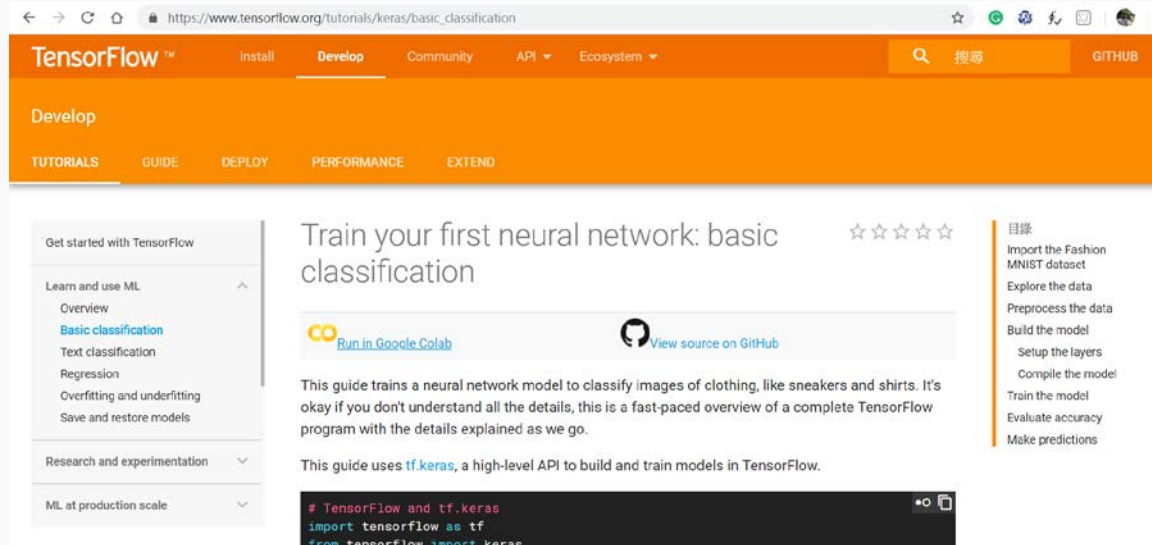
Plotimage()

<http://jupyter.org/try>

# Python programming

Train your first neural network:  
basic classification

[Run in Google Colab](https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tutorials/keras/basic_classification.ipynb)



The screenshot shows the TensorFlow website's tutorial page for 'Train your first neural network: basic classification'. The page has an orange header with navigation links: TensorFlow, Install, Develop, Community, API, and Ecosystem. Below the header is a sub-header with 'Develop' and a search bar. The main content area features a sidebar on the left with a table of contents, a central article area with a 'Run in Google Colab' button and a 'View source on GitHub' link, and a right sidebar with a '目錄' (Table of Contents) section. The article text describes the tutorial's purpose and mentions the use of tf.keras. A code snippet is visible at the bottom of the article.

TensorFlow™ Install Develop Community API Ecosystem

Develop

TUTORIALS GUIDE DEPLOY PERFORMANCE EXTEND

Get started with TensorFlow

Learn and use ML

- Overview
- Basic classification
- Text classification
- Regression
- Overfitting and underfitting
- Save and restore models

Research and experimentation

ML at production scale

## Train your first neural network: basic classification

☆☆☆☆☆

Run in Google Colab

View source on GitHub

This guide trains a neural network model to classify images of clothing, like sneakers and shirts. It's okay if you don't understand all the details, this is a fast-paced overview of a complete TensorFlow program with the details explained as we go.

This guide uses [tf.keras](#), a high-level API to build and train models in TensorFlow.

```
# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras
```

目錄

- Import the Fashion MNIST dataset
- Explore the data
- Preprocess the data
- Build the model
- Setup the layers
- Compile the model
- Train the model
- Evaluate accuracy
- Make predictions

[https://www.tensorflow.org/tutorials/keras/basic\\_classification](https://www.tensorflow.org/tutorials/keras/basic_classification)

# Python programming

TensorFlow+Keras  
深度學習人工智慧實務應用

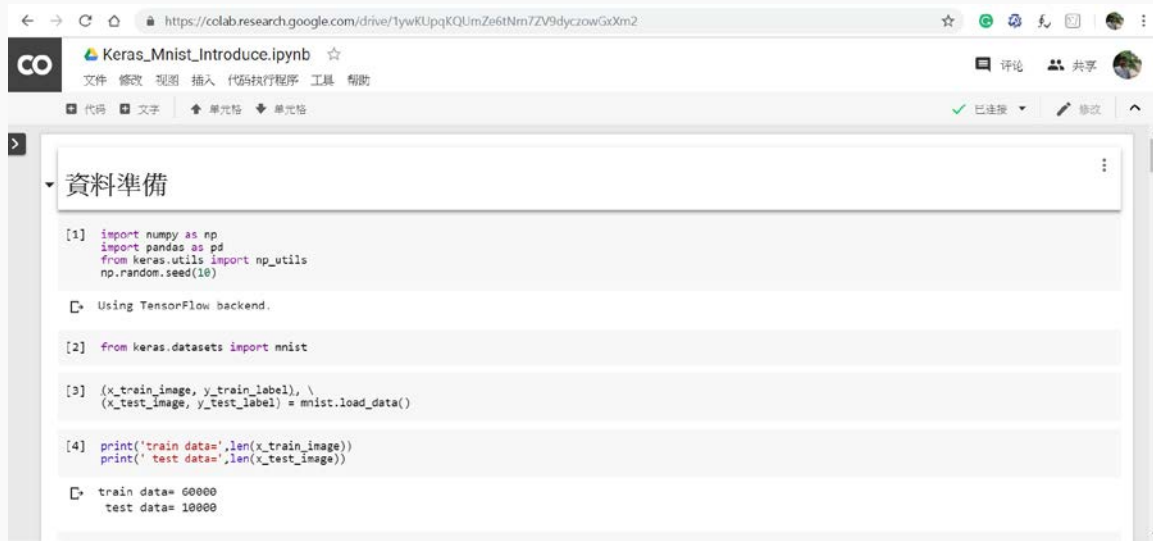


載範例程式（用下列URL

）[http://www.drmaster.com.tw/download/example/MP21710\\_example.zip](http://www.drmaster.com.tw/download/example/MP21710_example.zip)

# Python programming

Upload  
Keras\_Mnist\_Introduce.ipynb



```
[1] import numpy as np
import pandas as pd
from keras.utils import np_utils
np.random.seed(10)

[2] Using TensorFlow backend.

[2] from keras.datasets import mnist

[3] (x_train_image, y_train_label), \
(x_test_image, y_test_label) = mnist.load_data()

[4] print('train data', len(x_train_image))
print(' test data', len(x_test_image))

train data= 60000
test data= 10000
```

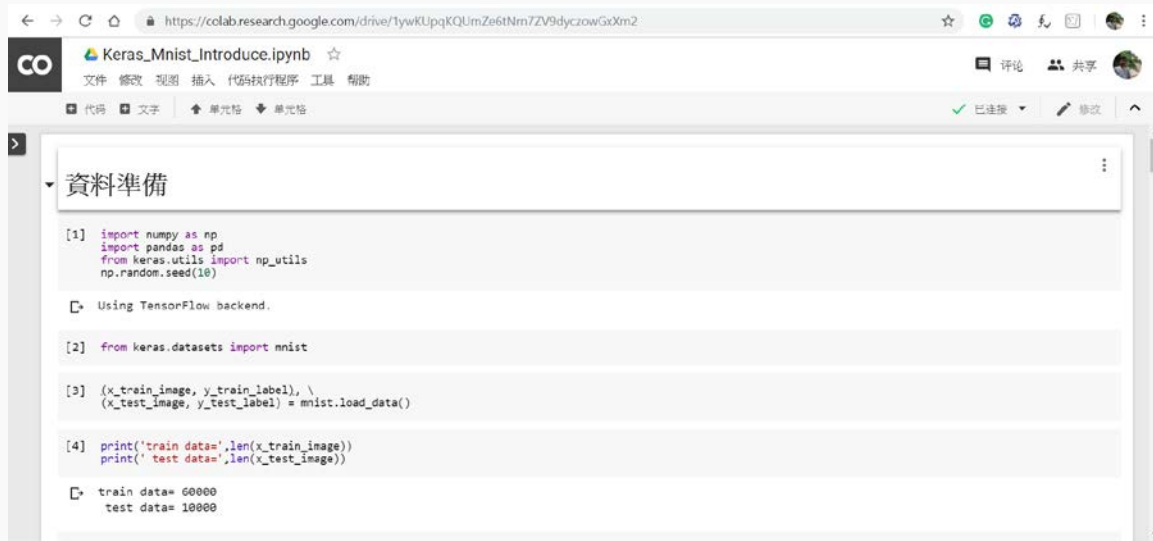
載範例程式（用下列URL

） [http://www.drmaster.com.tw/download/example/MP21710\\_example.zip](http://www.drmaster.com.tw/download/example/MP21710_example.zip)

# Python programming

Upload

Keras\_Mnist\_MLP\_h256.ipynb



```
[1] import numpy as np
import pandas as pd
from keras.utils import np_utils
np.random.seed(10)

[2] Using TensorFlow backend.

[2] from keras.datasets import mnist

[3] (x_train_image, y_train_label), \
(x_test_image, y_test_label) = mnist.load_data()

[4] print('train data', len(x_train_image))
print(' test data', len(x_test_image))

train data= 60000
test data= 10000
```

載範例程式（用下列URL

） [http://www.drmaster.com.tw/download/example/MP21710\\_example.zip](http://www.drmaster.com.tw/download/example/MP21710_example.zip)

# Python programming

- predictive models with sklearn pipelines

[Notebook](#) [Code](#) [Data \(1\)](#) [Comments \(1\)](#) [Log](#) [Versions \(2\)](#) [Forks \(5\)](#) [Fork Notebook](#)

## Building End to End predictive models with sklearn pipelines

Sklearn pipelines have been around for a while now. I always thought pipelines as a nifty feature that can take out the back forth scrolling in kernel notebooks between data cleaning, feature engineering and prediction steps, and have a single object/pipe that can take in data and give out predictions. I also found that this was a promiseland, it was not a easy task to create a practical/useful pipeline, that took in a DataFrame and gave out predictions.

But now, after a revisit to the land of pipelines, I could bet you that pipelines are back in action. No, I'm serious, I bet that it's possible to acheive the most organized and concise code for a problem by adapting pipelines. I'm willing to bet my upvote on it :)

I've explained every step along the way, So it should be easy to follow.

### What can you expect in this notebook

**Beginner** - I've never heard of sklearn pipelines - You are in for a treat. You'll be introduced to a new/clean/easy coding model to build predictive models

**Intermediate** - I've dabbled with pipes, haven't used them end to end - I'll introduce you to new techniques that can help you use pipelines better.

<https://www.kaggle.com/gautham11/building-predictive-models-with-sklearn-pipelines>



# Python programming

- [How to predict input image using trained model in Keras?](https://stackoverflow.com/questions/43469281/how-to-predict-input-image-using-trained-model-in-keras?)

<https://stackoverflow.com/questions/43469281/how-to-predict-input-image-using-trained-model-in-keras>