

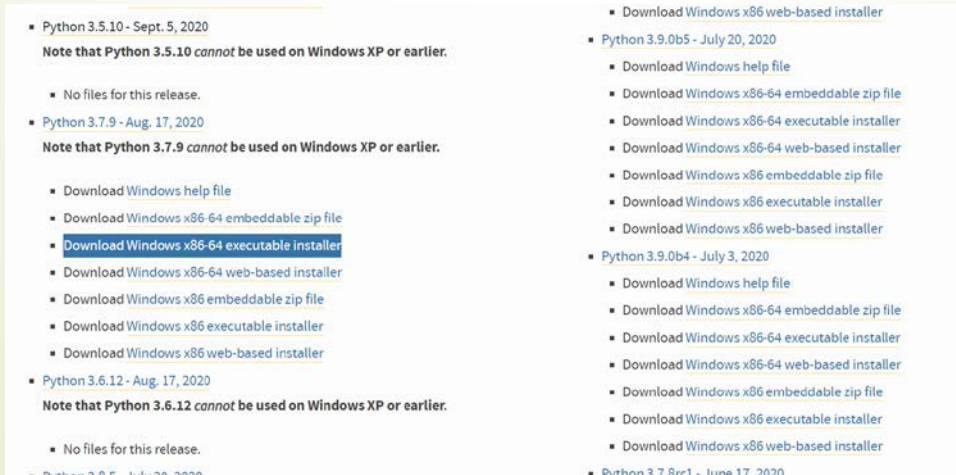
環境安裝

- Python 安裝(版本 3.7)
- Visual Studio Code安裝
- Python 擴充套件安裝
 - Jupyter Notebook安裝
 - Numpy 安裝
 - Pandas 安裝
 - Matplotlib安裝
 - OpenCv安裝
 - TensorFlow 安裝(2.0 版本)

Python 安裝(I)

- ➡ 下載3.7.9 安裝檔(不是最新版3.8)
(<https://www.python.org/downloads/windows/>)

Download Windows x86-64 executable installerS



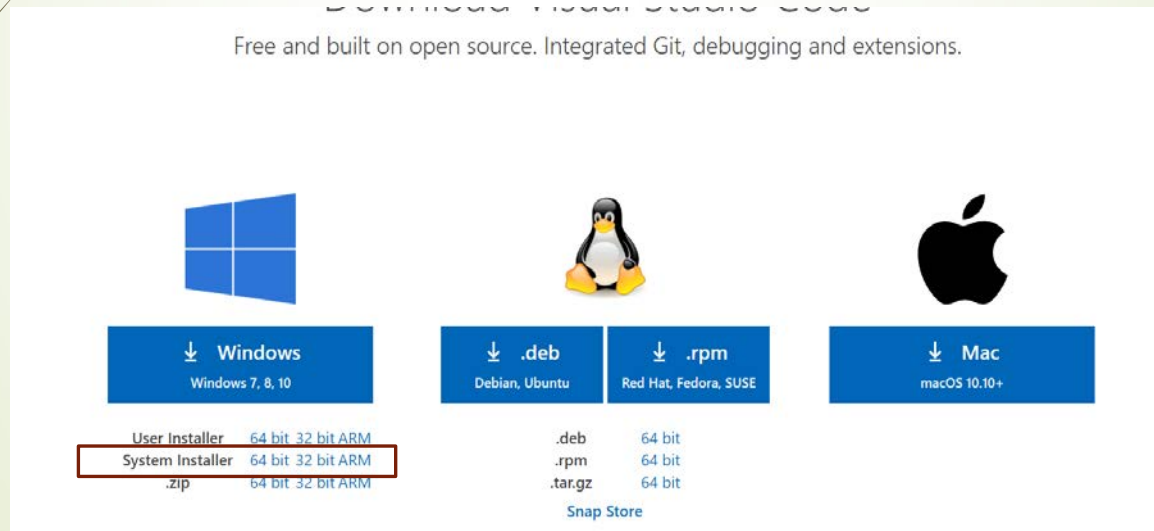
The screenshot displays the 'Downloads' section of the Python.org website, specifically the Windows download page. It lists several Python versions with their release dates and available download options. The version 3.7.9 is highlighted, and the 'Download Windows x86-64 executable installer' link is selected. The page also includes a note that Python 3.5.10 and 3.6.12 cannot be used on Windows XP or earlier.

- Python 3.5.10 - Sept. 5, 2020
Note that Python 3.5.10 cannot be used on Windows XP or earlier.
 - No files for this release.
- Python 3.7.9 - Aug. 17, 2020
Note that Python 3.7.9 cannot be used on Windows XP or earlier.
 - Download Windows help file
 - Download Windows x86-64 embeddable zip file
 - Download Windows x86-64 executable installer
 - Download Windows x86-64 web-based installer
 - Download Windows x86 embeddable zip file
 - Download Windows x86 executable installer
 - Download Windows x86 web-based installer
- Python 3.6.12 - Aug. 17, 2020
Note that Python 3.6.12 cannot be used on Windows XP or earlier.
 - No files for this release.
- Python 3.9.0b5 - July 20, 2020
 - Download Windows x86 web-based installer
- Python 3.9.0b4 - July 3, 2020
 - Download Windows help file
 - Download Windows x86-64 embeddable zip file
 - Download Windows x86-64 executable installer
 - Download Windows x86-64 web-based installer
 - Download Windows x86 embeddable zip file
 - Download Windows x86 executable installer
 - Download Windows x86 web-based installer
- Python 3.7.8rc1 - June 17, 2020

Visual studio Code 安裝(I)

- 下載Visual Studio Code 安裝檔

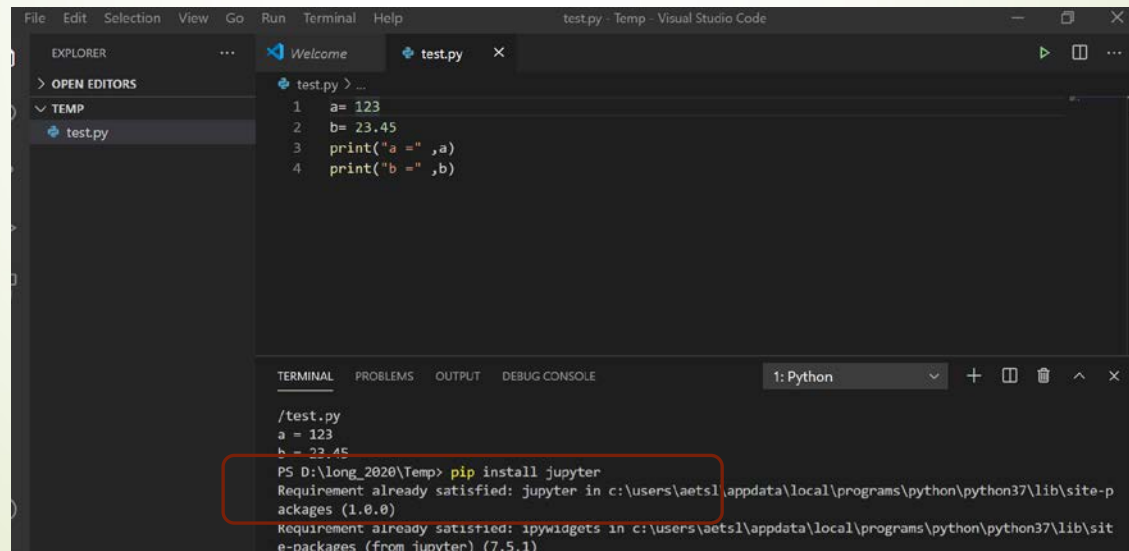
<https://code.visualstudio.com/download>



Jupyter Notebook安裝

➤ TERMINAL

- pip install jupyter
- pip install --upgrade pip (upgrade pip version)



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows a file named 'test.py' in the 'TEMP' directory. The Editor panel displays the contents of 'test.py':

```
1 a= 123
2 b= 23.45
3 print("a =" ,a)
4 print("b =" ,b)
```

The TERMINAL panel at the bottom shows the output of running the script and the command to install Jupyter:

```
/test.py
a = 123
b = 23.45
PS D:\long_2020\Temp> pip install jupyter
Requirement already satisfied: jupyter in c:\users\aelstl\AppData\Local\Programs\Python\Python37\lib\site-packages (1.0.0)
Requirement already satisfied: ipywidgets in c:\users\aelstl\AppData\Local\Programs\Python\Python37\lib\site-packages (from jupyter) (7.5.1)
```

A red box highlights the command and its output in the terminal.

安裝Python 擴充套件

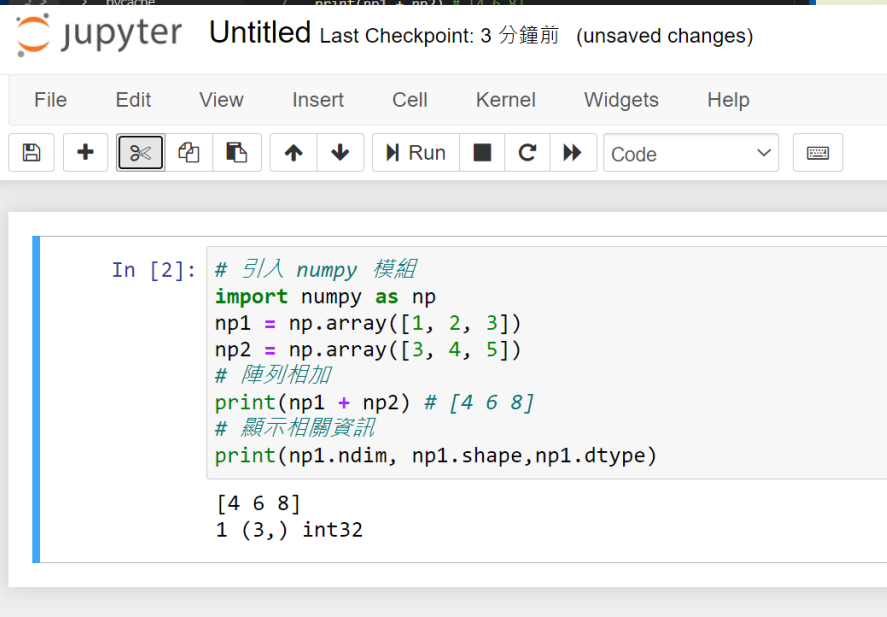
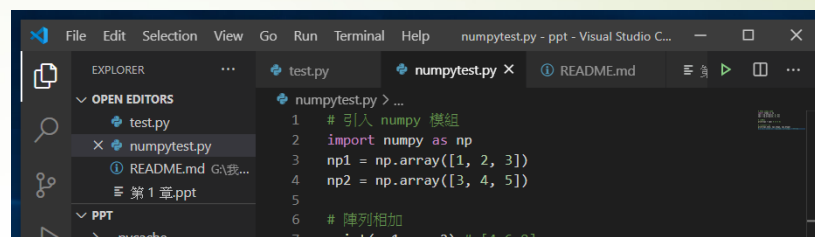
- `pip install numpy`
- `pip install pandas`
- `pip install Matplotlib`
- `pip install opencv-python`
- `pip install TensorFlow==2.0` (目前版本2.3)

Numpy 函式庫

- Numpy 是 Python 的一個重要模組（Python 是一個高階語言也是一種膠水語言，可以透過整合其他低階語言同時擁有效能和高效率的開發），主要用於資料處理上。Numpy 底層以 C 和 Fortran 語言實作，所以能快速操作多重維度的陣列。當 Python 處理龐大資料時，其原生 list 效能表現並不理想（但可以動態存異質資料），而 Numpy 具備平行處理的能力，可以將操作動作一次套用在大型陣列上。此外 Python 其餘重量級的資料科學相關套件（例如：Pandas、SciPy、Scikit-learn 等）都幾乎是奠基在 Numpy 的基礎上。因此學會 Numpy 對於往後學習其他資料科學相關套件打好堅實的基礎。

簡易Numpy 程式

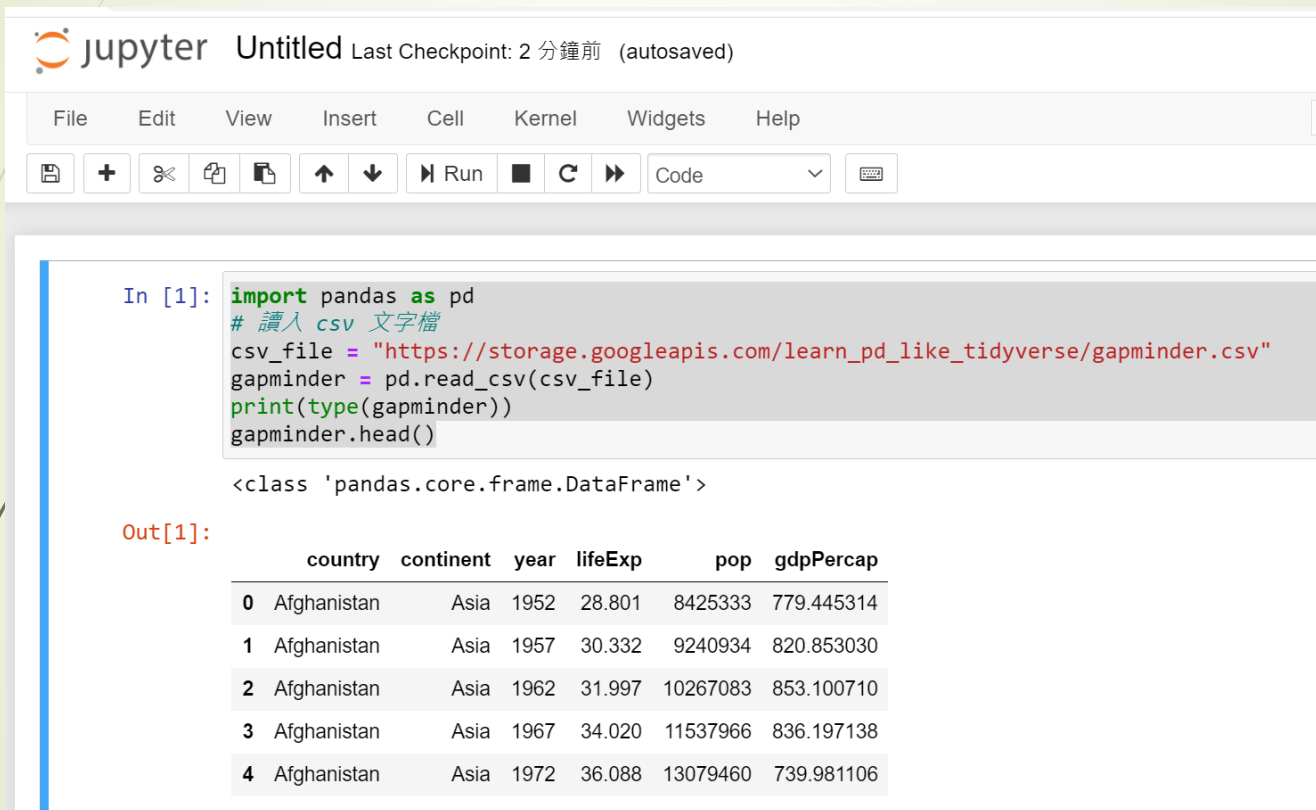
```
# 引入 numpy 模組
import numpy as np
np1 = np.array([1, 2, 3])
np2 = np.array([3, 4, 5])
# 陣列相加
print(np1 + np2) # [4 6 8]
# 顯示相關資訊
print(np1.ndim,
      np1.shape, np1.dtype)
```



Pandas 函式庫

- ▶ [pandas](#) 是 [Python](#) 的一個資料分析函式庫，提供如 [DataFrame](#) 等十分容易操作的資料結構，是近年做數據分析時不可或需的工具之一。
- ▶ pandas 可以支援多種文字、二進位檔案與資料庫的資料載入，常見的 txt、csv、excel 試算表、MySQL 或 PostgreSQL 都難不倒，如果對詳細的清單有興趣，可以參考 [pandas 0.21.0 documentation](#)。

Pandas 測試



The image shows a Jupyter Notebook interface. The title bar says "jupyter Untitled" and "Last Checkpoint: 2 分鐘前 (autosaved)". The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. The toolbar has icons for saving, adding cells, undo, redo, copy, paste, up/down arrows, run, and a dropdown menu set to "Code".

The code cell contains the following Python code:

```
In [1]: import pandas as pd
# 讀入 csv 文字檔
csv_file = "https://storage.googleapis.com/learn_pd_like_tidyverse/gapminder.csv"
gapminder = pd.read_csv(csv_file)
print(type(gapminder))
gapminder.head()
```

The output of the code cell is:

```
<class 'pandas.core.frame.DataFrame'>
```

Below the output, the first five rows of the DataFrame are displayed as a table:

| | country | continent | year | lifeExp | pop | gdpPercap |
|---|-------------|-----------|------|---------|----------|------------|
| 0 | Afghanistan | Asia | 1952 | 28.801 | 8425333 | 779.445314 |
| 1 | Afghanistan | Asia | 1957 | 30.332 | 9240934 | 820.853030 |
| 2 | Afghanistan | Asia | 1962 | 31.997 | 10267083 | 853.100710 |
| 3 | Afghanistan | Asia | 1967 | 34.020 | 11537966 | 836.197138 |
| 4 | Afghanistan | Asia | 1972 | 36.088 | 13079460 | 739.981106 |

Matplotlib函式庫

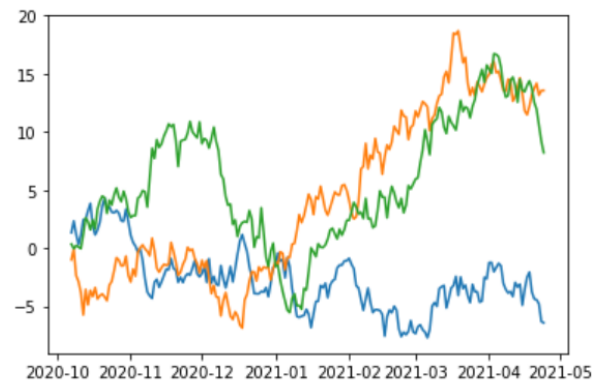
Python 的視覺化套件有靜態的 [Matplotlib](#)、[Seaborn](#) 和 [ggplot](#) (借鏡於 [R](#) 的 [ggplot2](#)) 套件以及動態的 [Bokeh](#) 套件 (類似於 D3.js)。其中 Matplotlib 是 Python 的一個重要模組 (Python 是一個高階語言也是一種膠水語言，可以透過整合其他低階語言同時擁有效能和高效率的開發)，主要用於資料視覺化上。一般來說使用 Matplotlib 有兩種主要方式：直接和 Matplotlib 的全域 pyplot 模組互動操作，第二種則是物件導向形式的操作方式。若是只有一張圖的話使用全域 pyplot 很方便，若是有多張圖的話用物件導向操作。一般來說 Matplotlib 預設值並不理想，但它的優點在於很容易在上面外包一層提供更好的預設值或是自己修改預設值。

Matplotlib測試

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
x = pd.period_range(pd.datetime.now(),
periods=200, freq='d')
x = x.to_timestamp().to_pydatetime()
# 產生三組，每組 200 個隨機常態分布元素
y = np.random.randn(200, 3).cumsum(0)
plt.plot(x, y)
plt.show()
```

```
In [5]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
x = pd.period_range(pd.datetime.now(), periods=200, freq='d')
x = x.to_timestamp().to_pydatetime()
# 產生三組，每組 200 個隨機常態分布元素
y = np.random.randn(200, 3).cumsum(0)
plt.plot(x, y)
plt.show()
```

c:\users\ae1sl\appdata\local\programs\python\python37\lib\site-p
time class is deprecated and will be removed from pandas in a fu
after removing the cwd from sys.path.



TensorFlow 函式庫

針對 TensorFlow 2.0，TensorFlow 團隊聽取了開發者關於「簡化 API、減少多餘並改進檔案和示例」的建議來設計，將 TensorFlow 2.0 Alpha 版更新重點放在簡單和易用性，主要有以下更新：

- 使用 Keras 和 eager execution，輕鬆建立簡單的模型並執行。
- 在任何平台達成生產環境的模型部署。
- 為研究提供強大的實驗工具。
- 清除不建議使用的 API 和減少重複來簡化 API。

TensorFlow 測試

```
#import tensorflow, 無論 CPU 或 GPU 版本都是 import tensorflow as tf
import tensorflow as tf
#將MNIST 手寫數字資料讀進來
mnist = tf.keras.datasets.mnist

# mnist 的load_data()會回傳已經先分割好的training data 和 testing data
# 並且將每個 pixel 的值從 Int 轉成 floating point 同時做normalize(這是很常見的
preprocessing)
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
print(len(x_train)) #training data 總共有60000張圖片
print(x_train[0].shape) #每張圖片 ( 拿第一張當樣本 ) 大小為 28x28
# 開始搭建model
# 利用 "Sequential" 把每層 layer 疊起來
```

TensorFlow 測試 (continued)

```
# input 大小為 28 x 28
# 最後的 Dense(10) 且 activation 用 softmax
# 代表最後 output 為 10 個 class ( 0~9 ) 的機率
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])

# model 每層定義好後需要經過 compile
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
# 將搭好的 model 去 fit 我們的 training data
# 並 evaluate 在 testing data 上
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test, verbose=2)
```

TensorFlow 測試(continued)

```
In [1]: #import tensorflow, 無論 CPU 或 GPU 版本都是 import tensorflow as tf
import tensorflow as tf
```

```
In [2]: #將MNIST 手寫數字資料讀進來
mnist = tf.keras.datasets.mnist

# mnist 的load_data()會回傳已經先分割好的training data 和 testing data
# 並且將每個 pixel 的值從 Int 轉成 floating point 同時做normalize(這是很常見的preprocessing)
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

```
In [3]: print(len(x_train)) #training data 總共有60000張圖片
print(x_train[0].shape) #每張圖片 (拿第一張當樣本) 大小為 28x28
```

```
60000
(28, 28)
```

TensorFlow 測試 (continued)

```
In [4]: # 開始搭建model
# 利用 "Sequential" 把每層 layer 疊起來

# input 大小為 28 x 28

# 最後的 Dense(10) 且 activation 用 softmax
# 代表最後 output 為 10 個 class (0~9) 的機率
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
])

# model 每層定義好後需要經過compile
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```


TensorFlow 測試 (continued)

```
In [5]: # 將搭好的 model 去 fit 我們的 training data  
# 並evaluate 在 testing data 上  
model.fit(x_train, y_train, epochs=5)  
model.evaluate(x_test, y_test, verbose=2)
```

```
Train on 60000 samples  
Epoch 1/5  
60000/60000 [=====] - 5s 78us/sample - loss:  
0.3006 - accuracy: 0.9127  
Epoch 2/5  
60000/60000 [=====] - 4s 67us/sample - loss:  
0.1432 - accuracy: 0.9574  
Epoch 3/5  
60000/60000 [=====] - 4s 70us/sample - loss:  
0.1074 - accuracy: 0.9666  
Epoch 4/5  
60000/60000 [=====] - 4s 67us/sample - loss:  
0.0882 - accuracy: 0.9732  
Epoch 5/5  
60000/60000 [=====] - 4s 68us/sample - loss:  
0.0755 - accuracy: 0.9765  
10000/1 - 0s - loss: 0.0391 - accuracy: 0.9768
```

```
Out[5]: [0.07640696551923175, 0.9768]
```