國立雲林科技大學
電子工程系
Department of Electronic Engineering

教育部補助AI應用領域系列課程-
人工智慧計算晶片設計和應用人才培育

LAB2-Linear regression

國立雲林科技大學
夏世昌 特聘教授 / 電子工程系
王斯弘 助理教授 / 前瞻學位學士學程

2020, Fall Semester

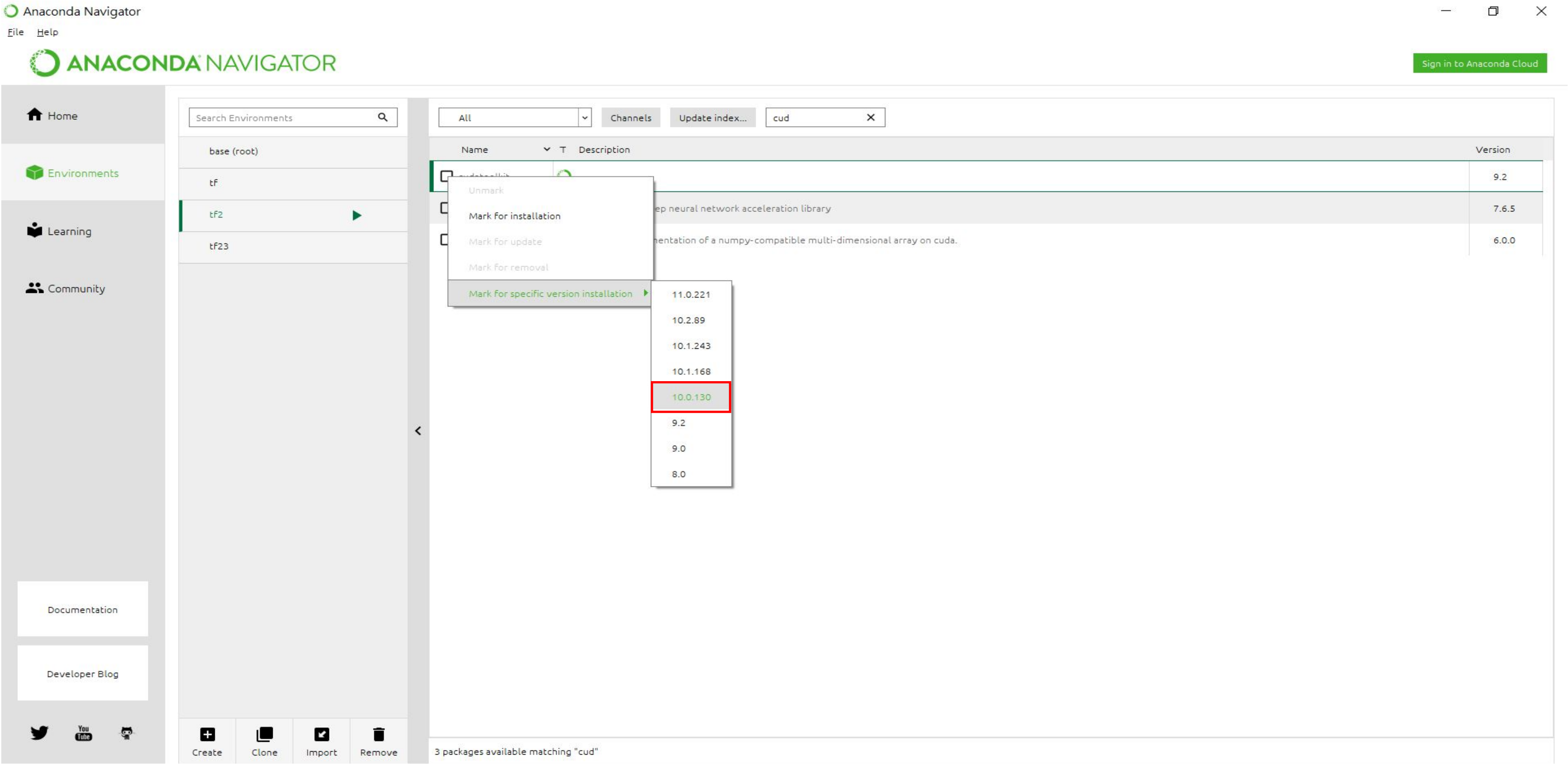
安裝TensorFlow

GPU版本

需在tf2環境下安裝 **codatoolkit** 和 **cudnn**

安裝TensorFlow

GPU版本codatoolkit

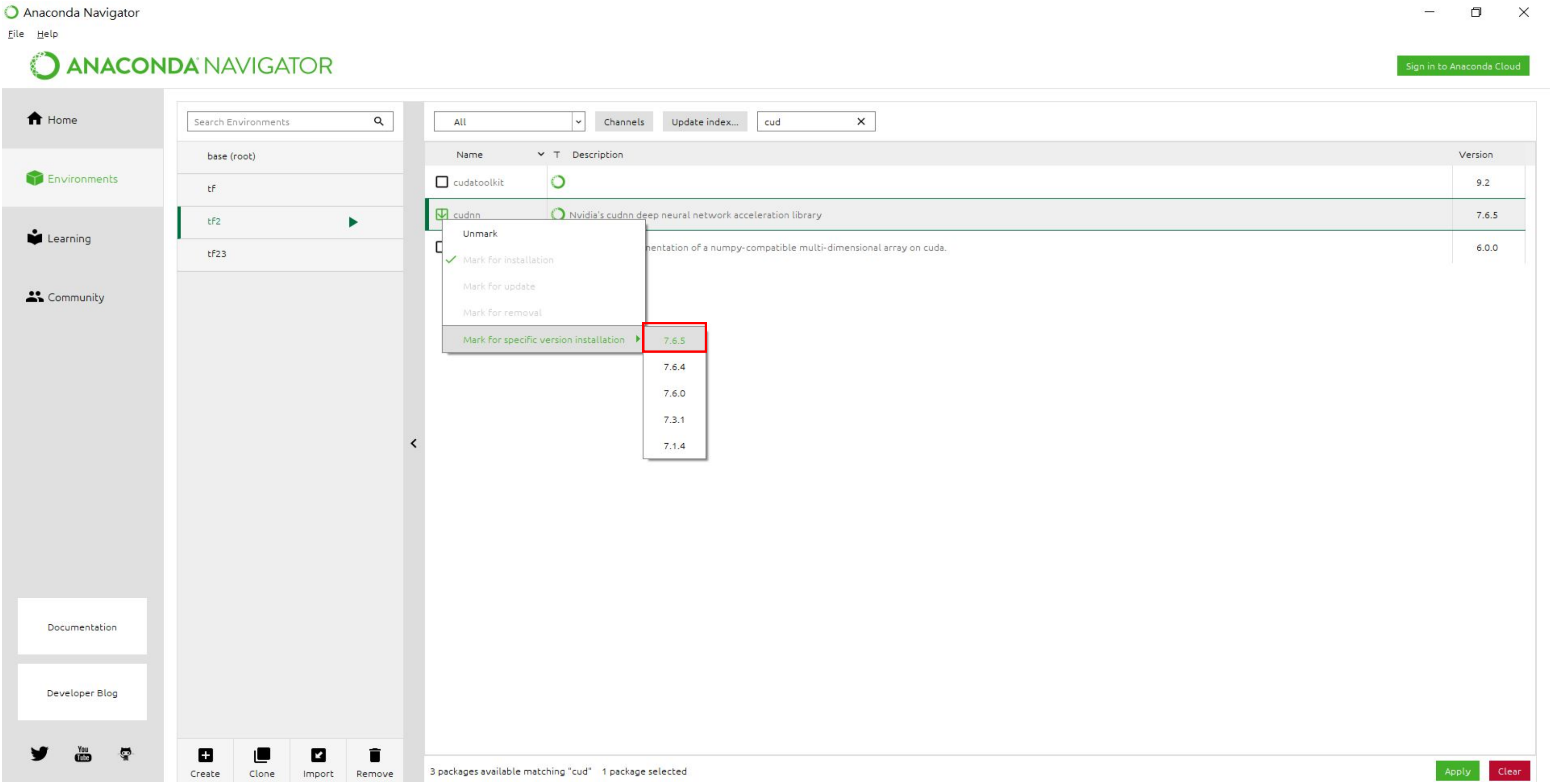


Name	T	Description	Version
cudatoolkit			9.2
		ep neural network acceleration library	7.6.5
		entation of a numpy-compatible multi-dimensional array on cuda.	6.0.0

3 packages available matching "cud"

安裝TensorFlow

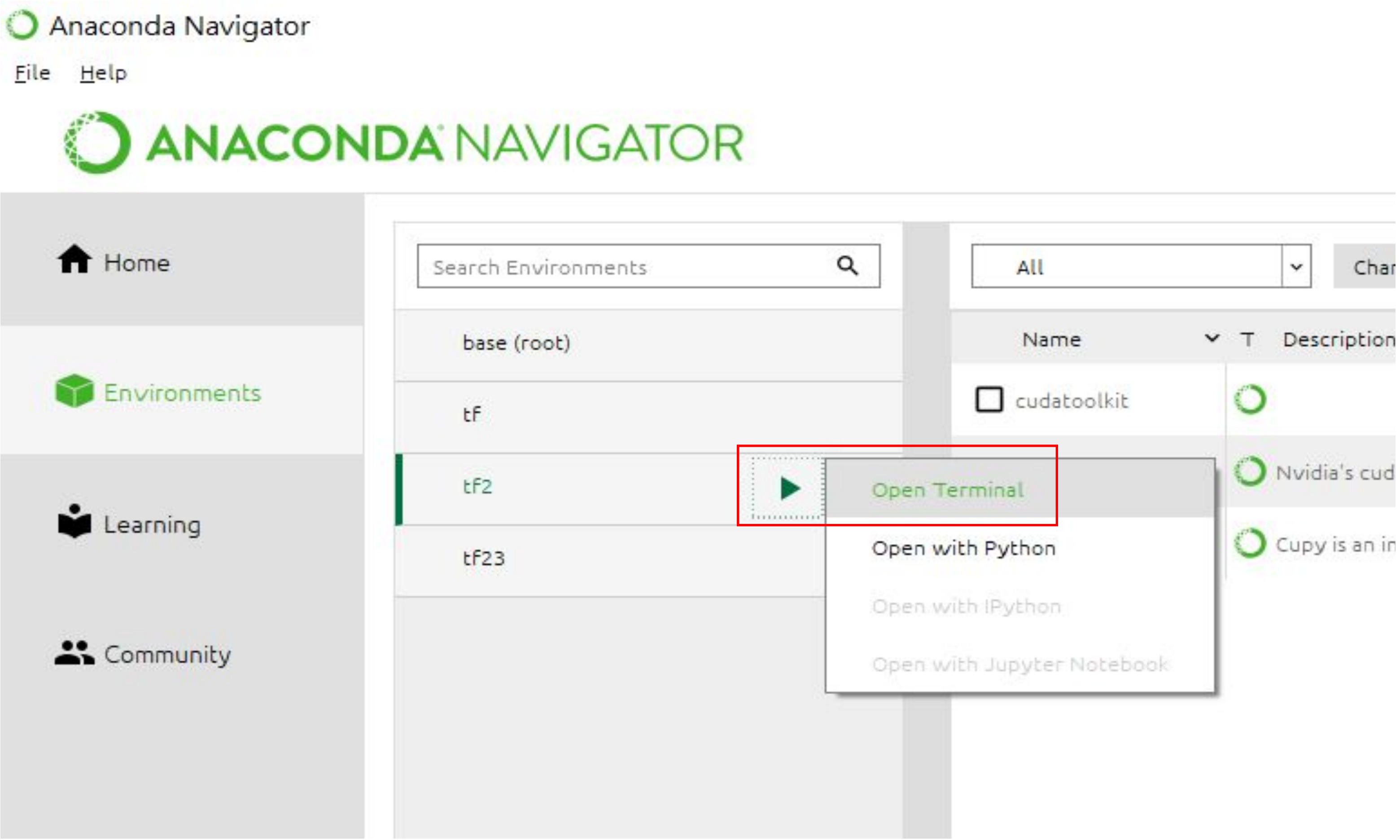
GPU版本cudnn



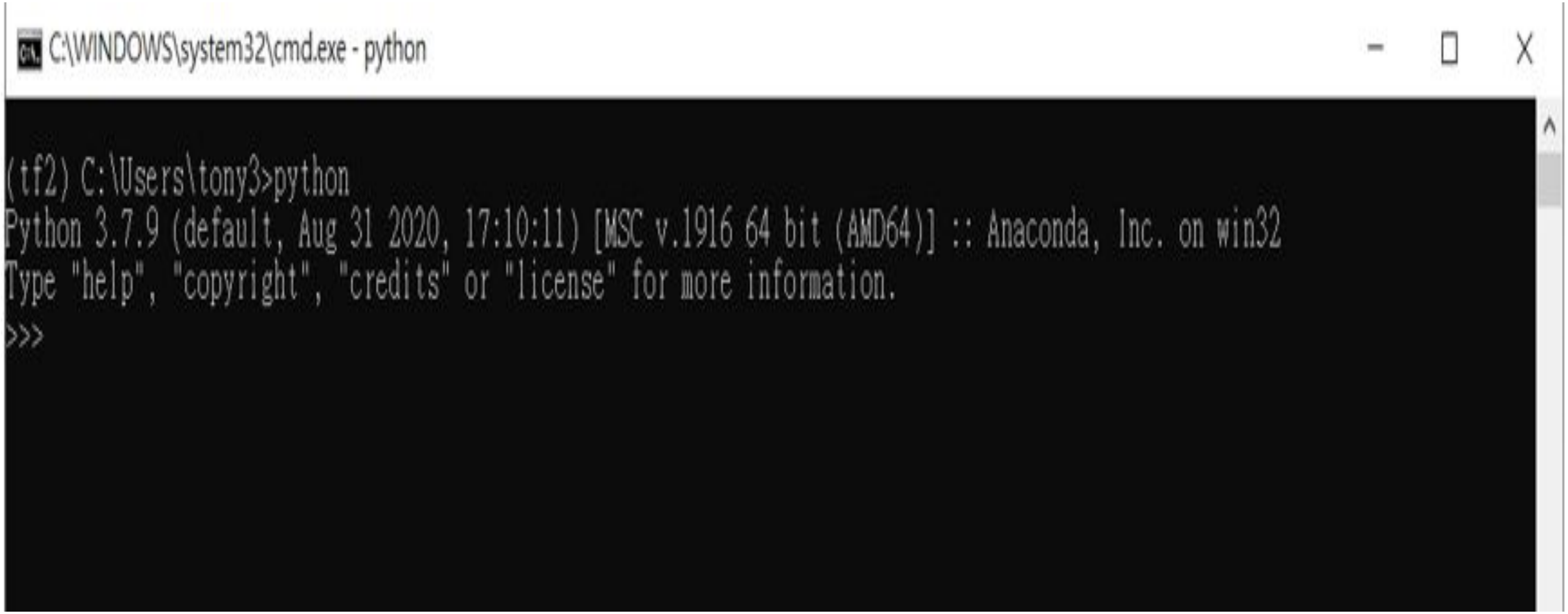
Name	T	Description	Version
□ cudatoolkit	○		9.2
☑ cudnn	○	Nvidia's cudnn deep neural network acceleration library	7.6.5
□		nentation of a numpy-compatible multi-dimensional array on cuda.	6.0.0

3 packages available matching "cud"   1 package selected

執行環境


Anaconda Navigator
File Help
ANACONDA NAVIGATOR
Home
Environments
Learning
Community
Search Environments
base (root)
tf
tf2
tf23
Open Terminal
Open with Python
Open with IPython
Open with Jupyter Notebook
All
Char
Name
T
Description
cudatoolkit
Nvidia's cud
Cupy is an ir

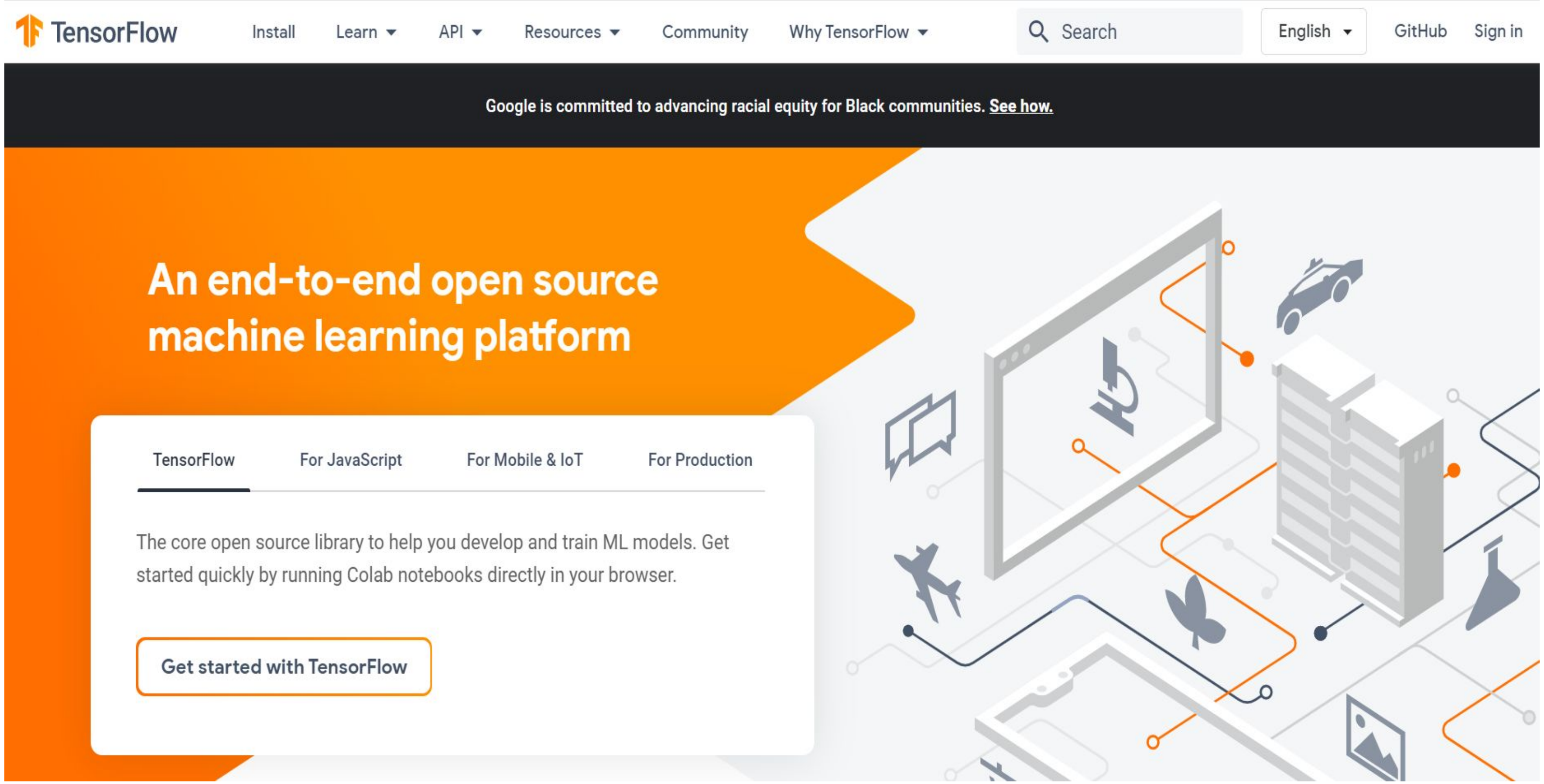
執行環境



```
(tf2) C:\Users\tony3>python
Python 3.7.9 (default, Aug 31 2020, 17:10:11) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

TensorFlow

https://www.tensorflow.org/



Google is committed to advancing racial equity for Black communities. See how.

An end-to-end open source machine learning platform

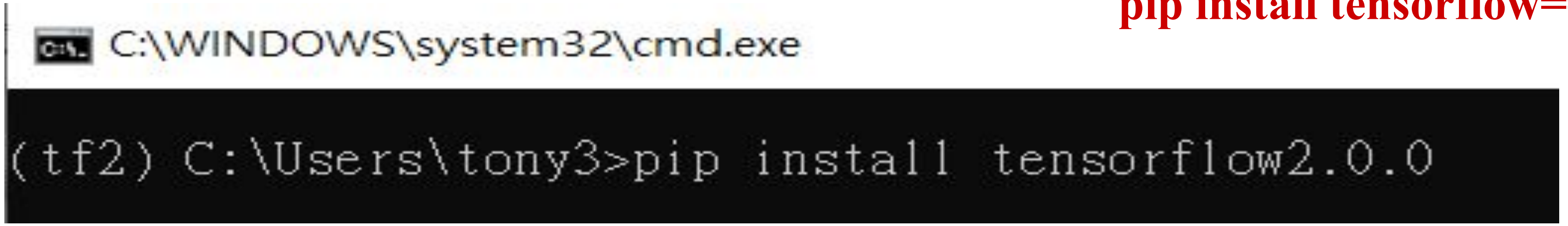
The core open source library to help you develop and train ML models. Get started quickly by running Colab notebooks directly in your browser.

TensorFlow

CPU版本

在 ES302 有些電腦請 tensorflow安裝2.5.0

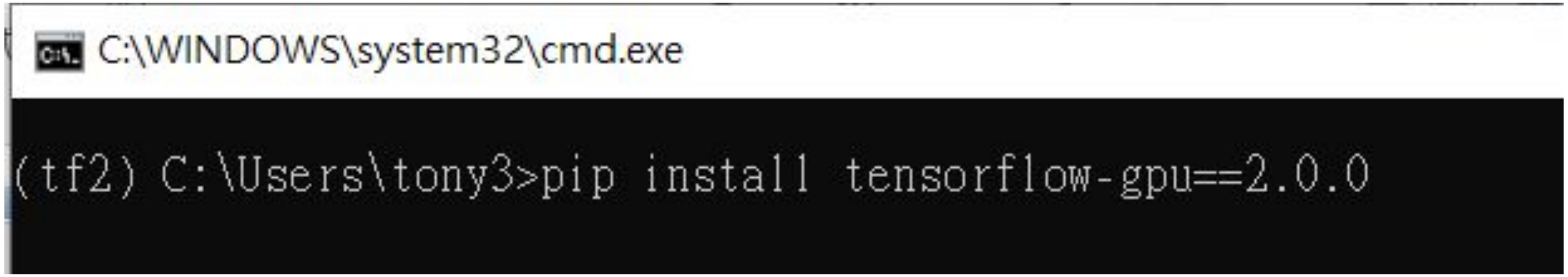
pip install tensorflow==2.5.0



C:\WINDOWS\system32\cmd.exe

```
(tf2) C:\Users\tony3>pip install tensorflow2.0.0
```

GPU版本

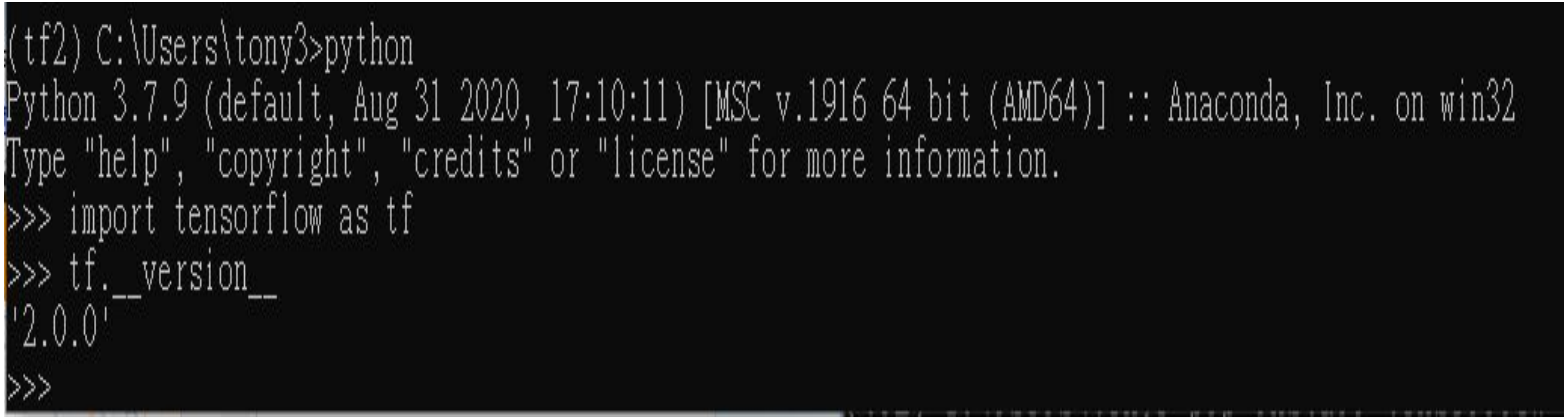


C:\WINDOWS\system32\cmd.exe

```
(tf2) C:\Users\tony3>pip install tensorflow-gpu==2.0.0
```

TensorFlow

安裝後可以在python中import tensorflow。
若沒有出現Error代表安裝成功。

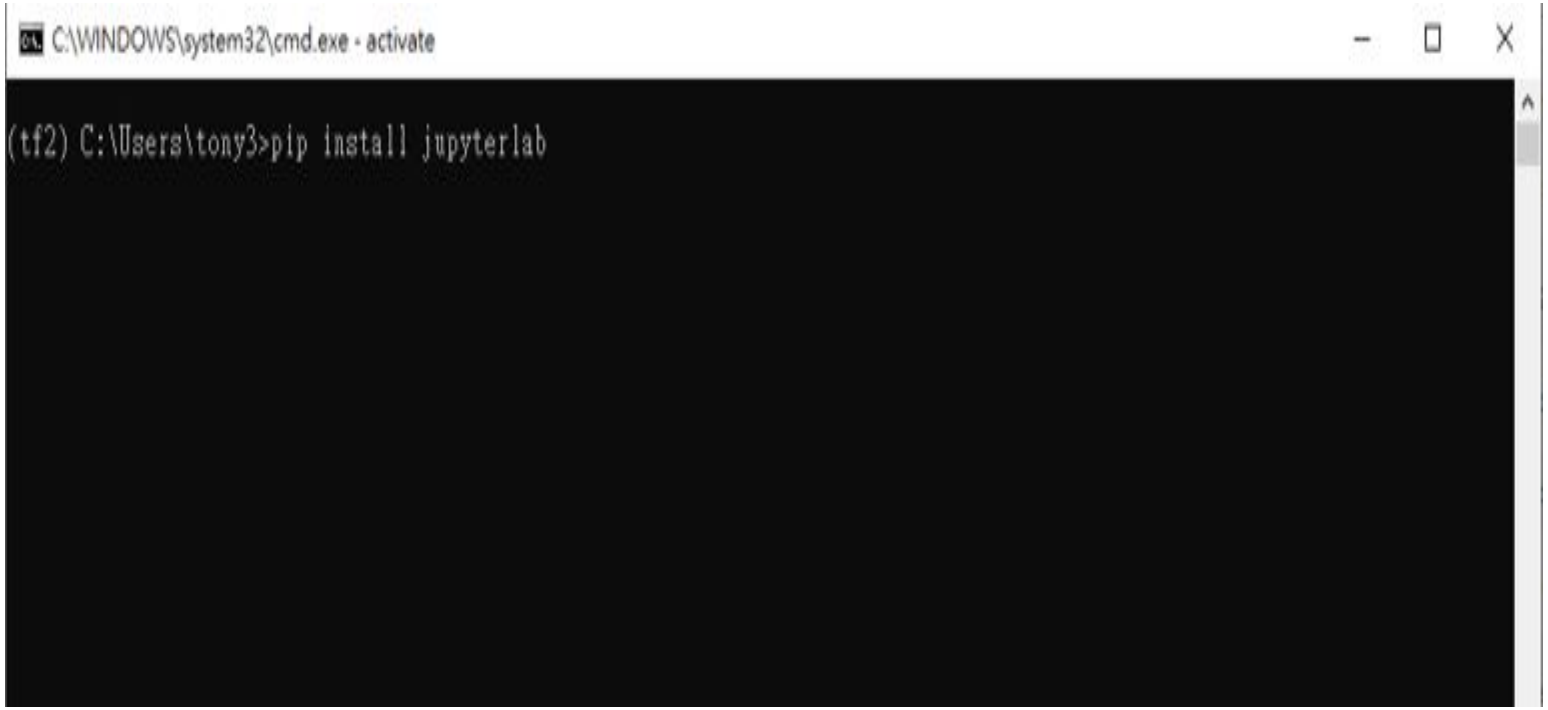


```
(tf2) C:\Users\tony3>python
Python 3.7.9 (default, Aug 31 2020, 17:10:11) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
>>> tf.__version__
'2.0.0'
>>>
```

JUPYTER LAB

建議可安裝jupyter lab

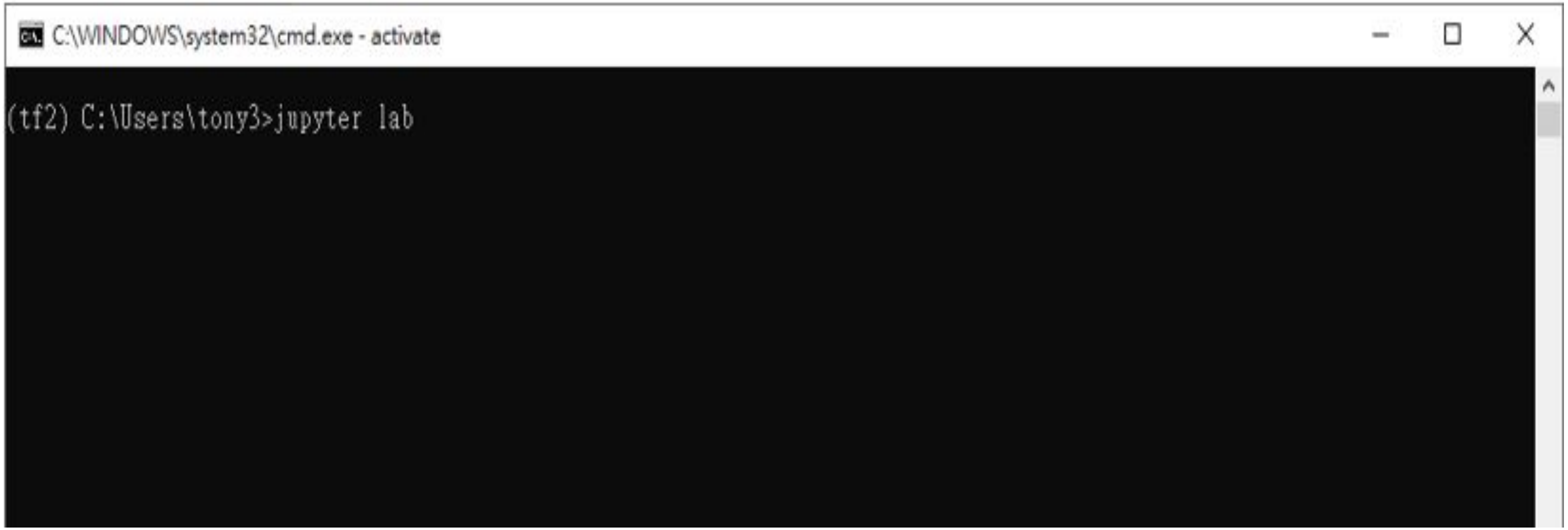
& pip install jupyterlab



```
(tf2) C:\Users\tony3>pip install jupyterlab
```

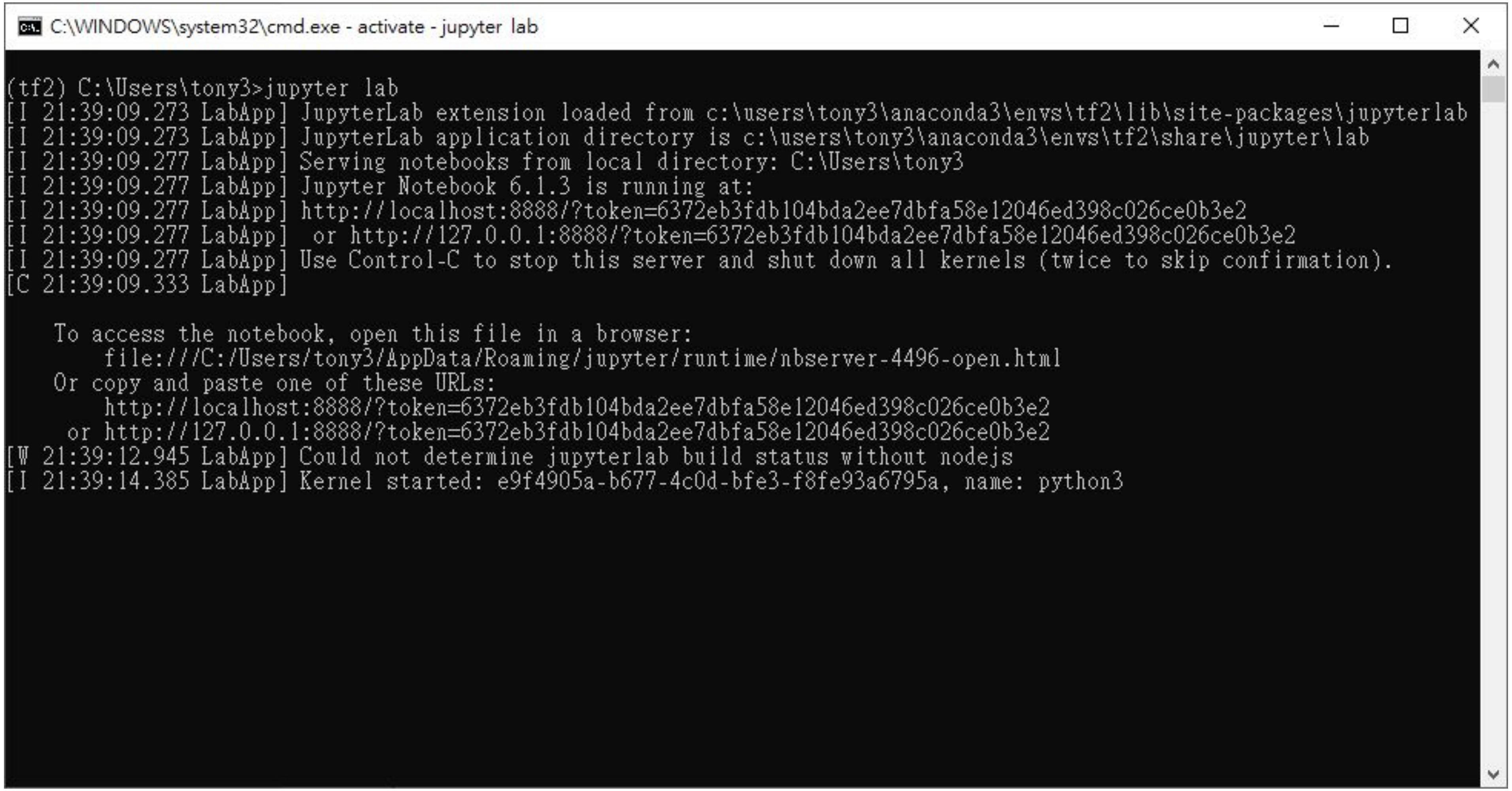
執行JUPYTER LAB

& jupyter lab



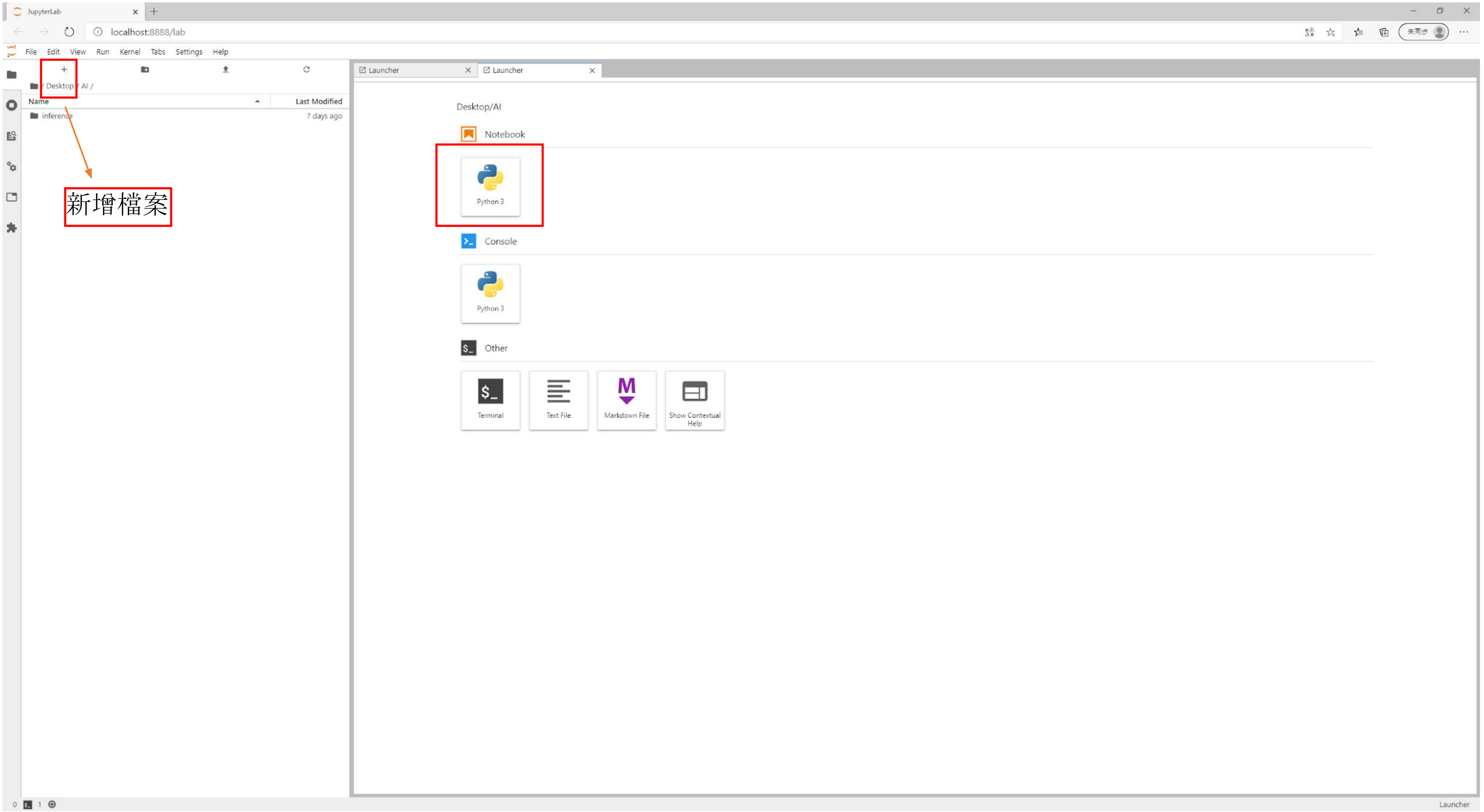
```
(tf2) C:\Users\tony3>jupyter lab
```

執行JUPYTER LAB



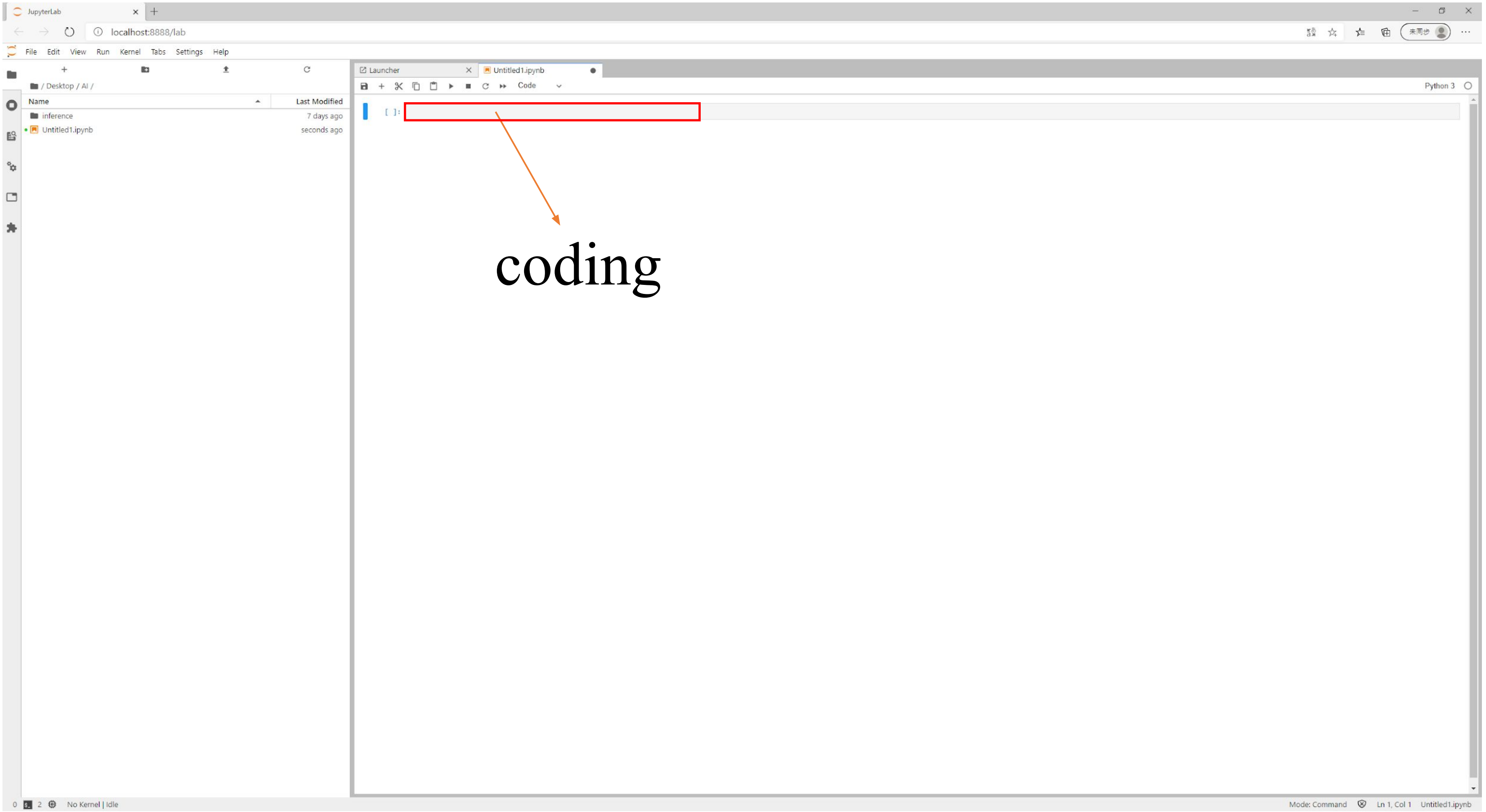
```
(tf2) C:\Users\tony3>jupyter lab
[I 21:39:09.273 LabApp] JupyterLab extension loaded from c:\users\tony3\anaconda3\envs\tf2\lib\site-packages\jupyterlab
[I 21:39:09.273 LabApp] JupyterLab application directory is c:\users\tony3\anaconda3\envs\tf2\share\jupyter\lab
[I 21:39:09.277 LabApp] Serving notebooks from local directory: C:\Users\tony3
[I 21:39:09.277 LabApp] Jupyter Notebook 6.1.3 is running at:
[I 21:39:09.277 LabApp] http://localhost:8888/?token=6372eb3fdb104bda2ee7dbfa58e12046ed398c026ce0b3e2
[I 21:39:09.277 LabApp]  or http://127.0.0.1:8888/?token=6372eb3fdb104bda2ee7dbfa58e12046ed398c026ce0b3e2
[I 21:39:09.277 LabApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 21:39:09.333 LabApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/tony3/AppData/Roaming/jupyter/runtime/nbserver-4496-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=6372eb3fdb104bda2ee7dbfa58e12046ed398c026ce0b3e2
     or http://127.0.0.1:8888/?token=6372eb3fdb104bda2ee7dbfa58e12046ed398c026ce0b3e2
[W 21:39:12.945 LabApp] Could not determine jupyterlab build status without nodejs
[I 21:39:14.385 LabApp] Kernel started: e9f4905a-b677-4c0d-bfe3-f8fe93a6795a, name: python3
```

執行JUPYTER LAB



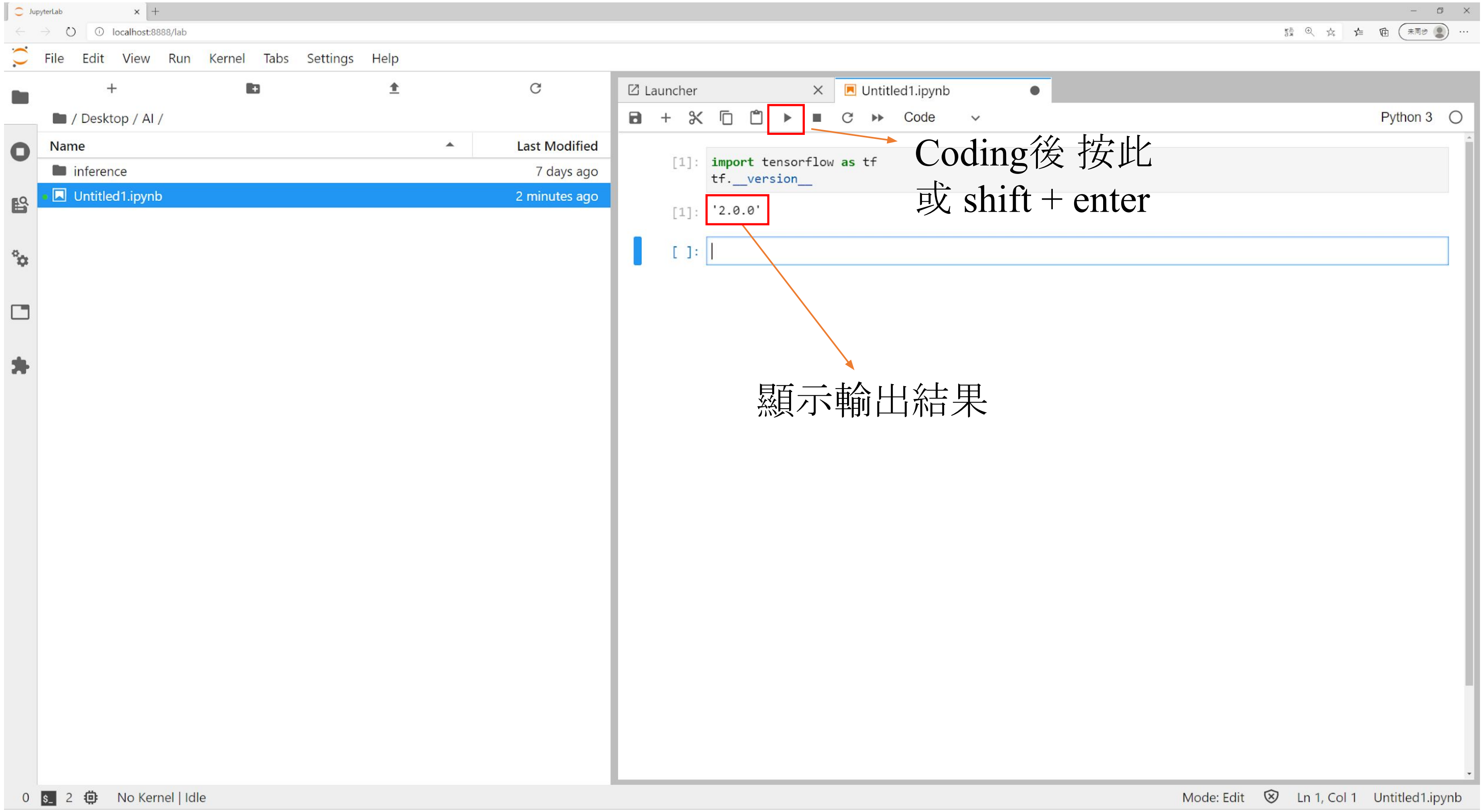
新增檔案

執行JUPYTER LAB



coding

執行JUPYTER LAB



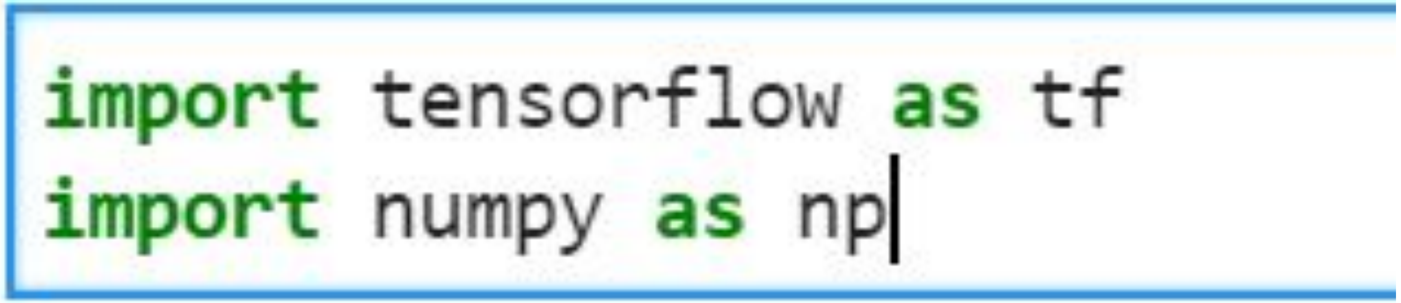
```
[1]: import tensorflow as tf
     tf.__version__
[1]: '2.0.0'
[ ]:
```

Coding後 按此
或 shift + enter

顯示輸出結果

START!!!

Import library



```
import tensorflow as tf
import numpy as np
```

Import library

```
import tensorflow as tf
```

TensorFlow是一個開源軟體庫, 用於各種感知和語言理解任務的機器學習。

https://www.tensorflow.org/api_docs/python/tf

Import library

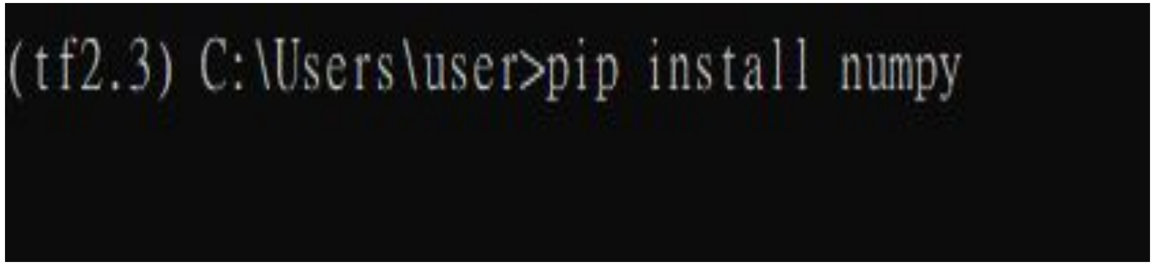
```
import numpy as np
```

Numpy 是 Python 的一個重要模組，主要用於資料處理上。

Import library

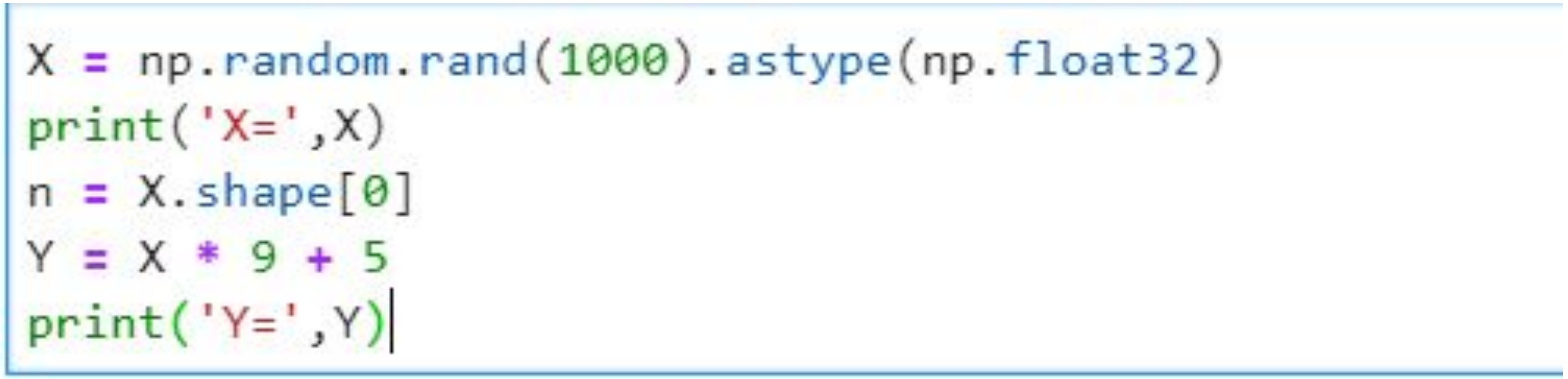
安裝numpy

& pip install numpy



```
(tf2.3) C:\Users\user>pip install numpy
```

dataset



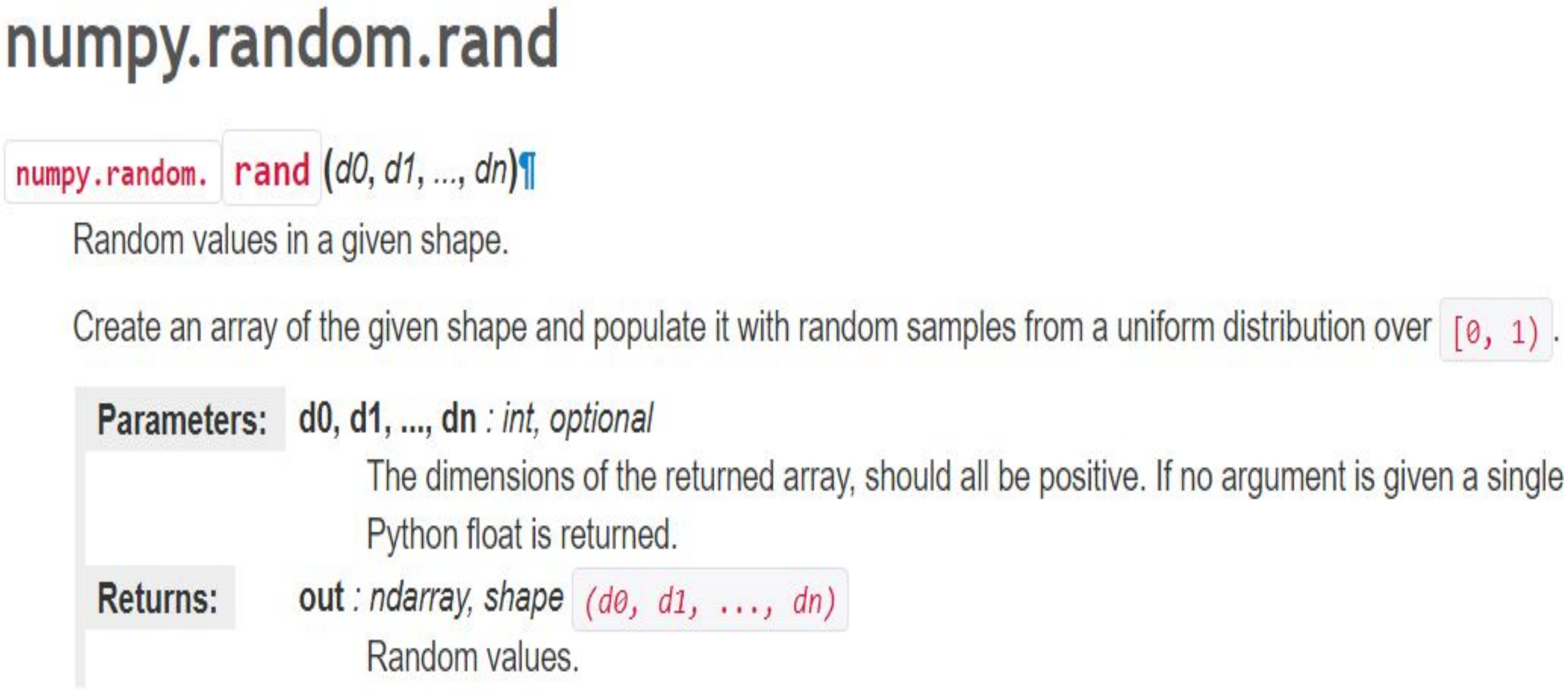
```
X = np.random.rand(1000).astype(np.float32)
print('X=',X)
n = X.shape[0]
Y = X * 9 + 5
print('Y=',Y)
```

X為訓練資料, 亂數產生1000筆

Y為對應X的答案

dataset

numpy.random.rand



numpy.random.rand

`numpy.random.rand(d0, d1, ..., dn)`¶

Random values in a given shape.

Create an array of the given shape and populate it with random samples from a uniform distribution over `[0, 1)`.

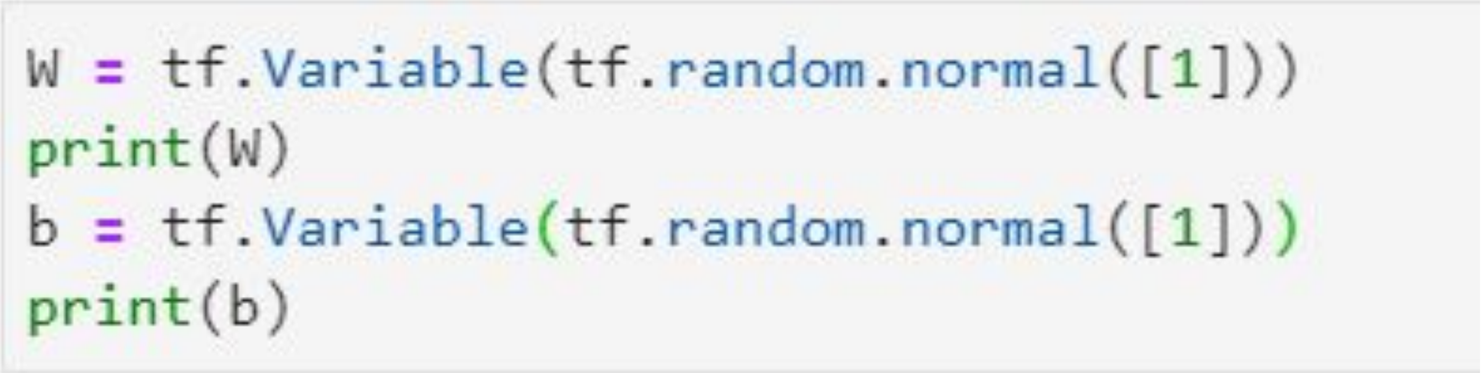
Parameters: **d0, d1, ..., dn** : *int, optional*

The dimensions of the returned array, should all be positive. If no argument is given a single Python float is returned.

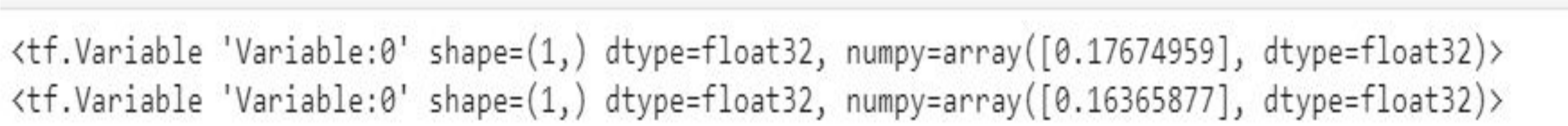
Returns: **out** : *ndarray, shape* `(d0, d1, ..., dn)`

Random values.

權重初始化



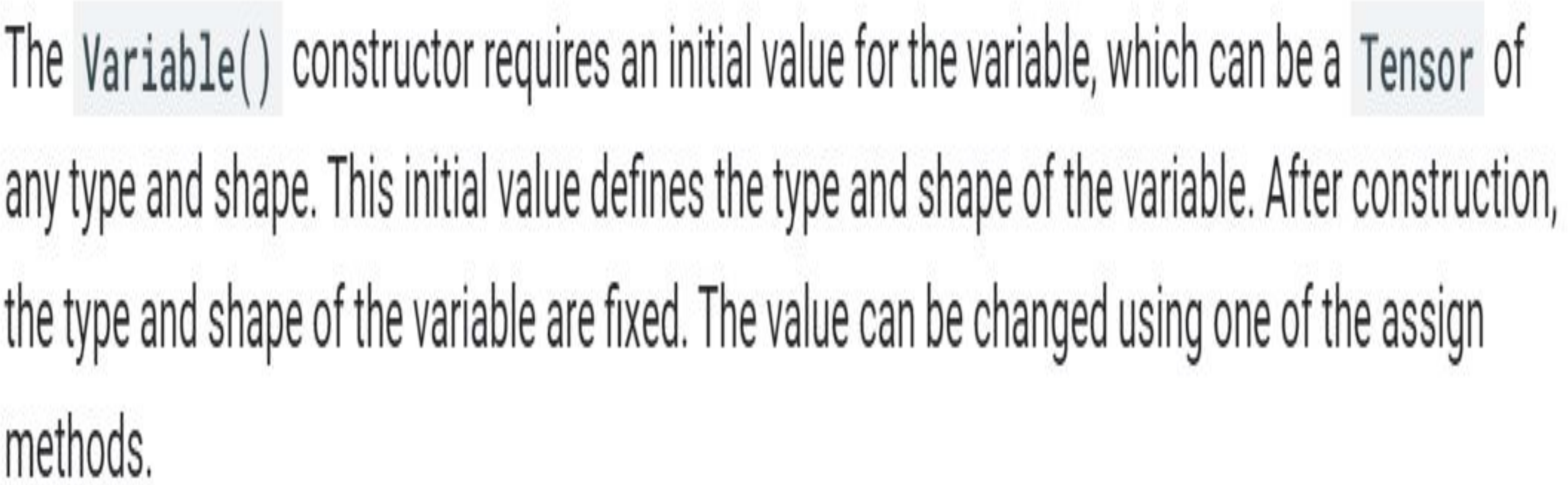
```
W = tf.Variable(tf.random.normal([1]))
print(W)
b = tf.Variable(tf.random.normal([1]))
print(b)
```



```
<tf.Variable 'Variable:0' shape=(1,) dtype=float32, numpy=array([0.17674959], dtype=float32)>
<tf.Variable 'Variable:0' shape=(1,) dtype=float32, numpy=array([0.16365877], dtype=float32)>
```

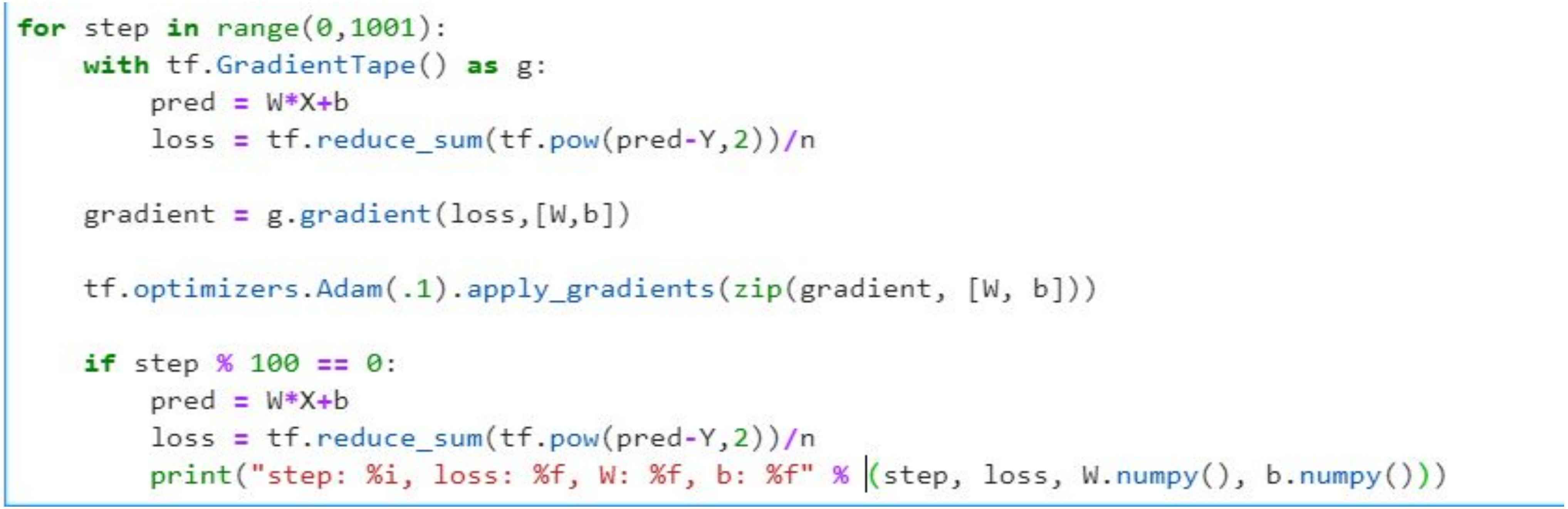
權重初始化

tf.Variable()



The `Variable()` constructor requires an initial value for the variable, which can be a `Tensor` of any type and shape. This initial value defines the type and shape of the variable. After construction, the type and shape of the variable are fixed. The value can be changed using one of the assign methods.

訓練



```
for step in range(0,1001):
    with tf.GradientTape() as g:
        pred = W*X+b
        loss = tf.reduce_sum(tf.pow(pred-Y,2))/n

    gradient = g.gradient(loss,[W,b])

    tf.optimizers.Adam(.1).apply_gradients(zip(gradient, [W, b]))

    if step % 100 == 0:
        pred = W*X+b
        loss = tf.reduce_sum(tf.pow(pred-Y,2))/n
        print("step: %i, loss: %f, W: %f, b: %f" % (step, loss, W.numpy(), b.numpy()))
```

預測為W*X+b

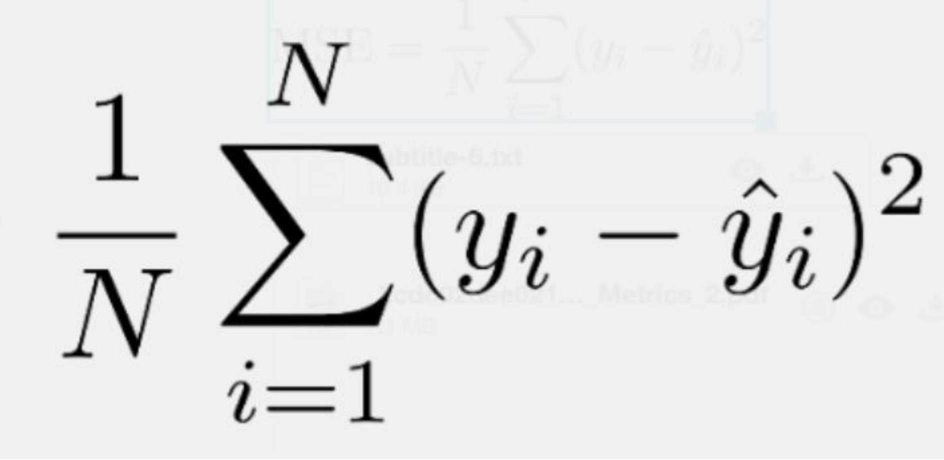
Loss採用mean square

Optimizer採用Adam

$$\frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y}_i)^2$$

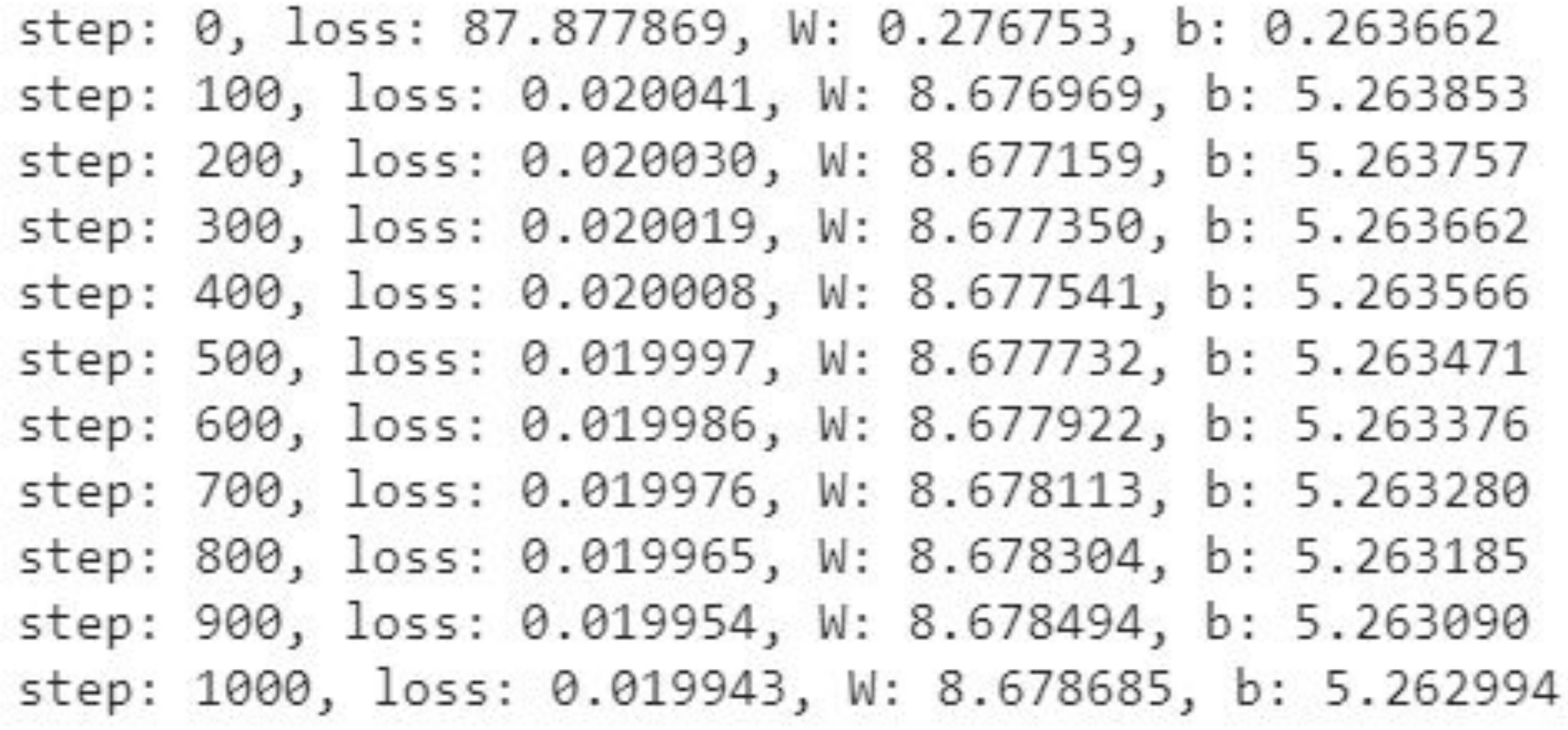
訓練

```
loss = tf.reduce_sum(tf.pow(pred-Y,2))/n
```



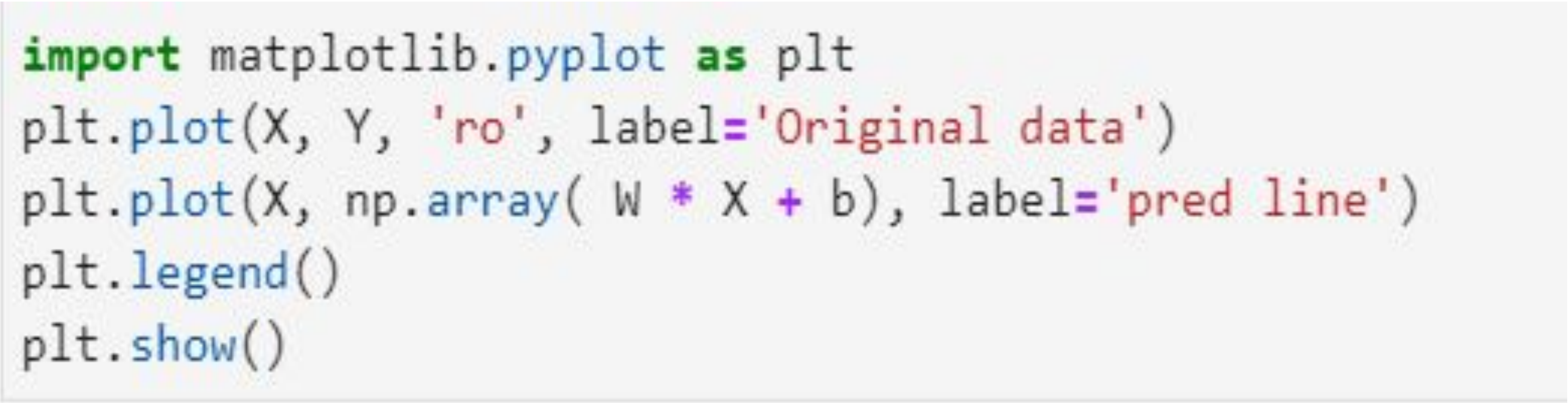
$$\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

訓練



```
step: 0, loss: 87.877869, W: 0.276753, b: 0.263662
step: 100, loss: 0.020041, W: 8.676969, b: 5.263853
step: 200, loss: 0.020030, W: 8.677159, b: 5.263757
step: 300, loss: 0.020019, W: 8.677350, b: 5.263662
step: 400, loss: 0.020008, W: 8.677541, b: 5.263566
step: 500, loss: 0.019997, W: 8.677732, b: 5.263471
step: 600, loss: 0.019986, W: 8.677922, b: 5.263376
step: 700, loss: 0.019976, W: 8.678113, b: 5.263280
step: 800, loss: 0.019965, W: 8.678304, b: 5.263185
step: 900, loss: 0.019954, W: 8.678494, b: 5.263090
step: 1000, loss: 0.019943, W: 8.678685, b: 5.262994
```

結果



```python
import matplotlib.pyplot as plt
plt.plot(X, Y, 'ro', label='Original data')
plt.plot(X, np.array( W * X + b), label='pred line')
plt.legend()
plt.show()
```

結果

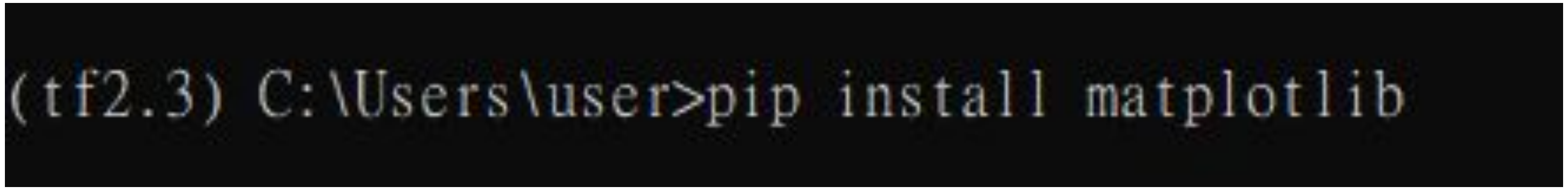
```
import matplotlib.pyplot as plt
```

matplotlib是Python程式語言及其數值數學擴展包 NumPy的可視化操作界面。

結果

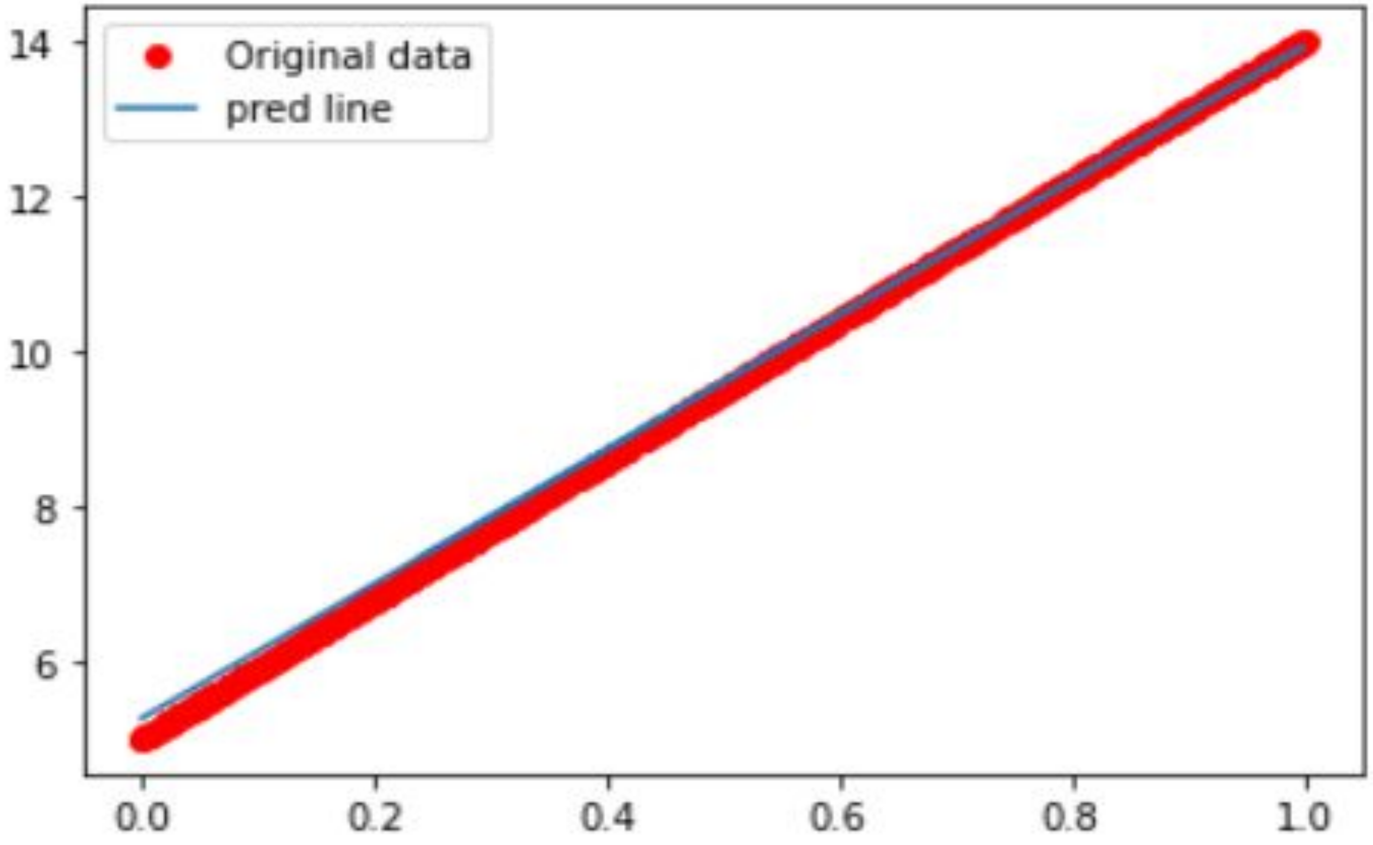
安裝matplotlib

& pip install matplotlib

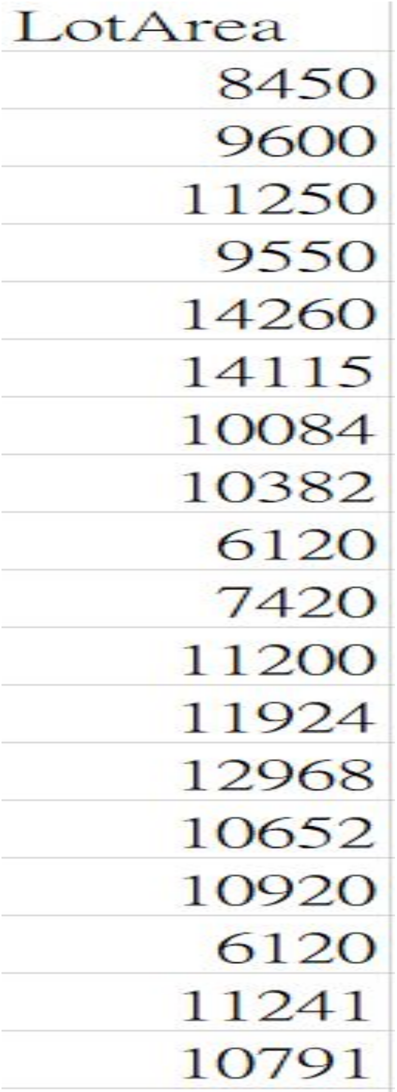


```
(tf2.3) C:\Users\user>pip install matplotlib
```

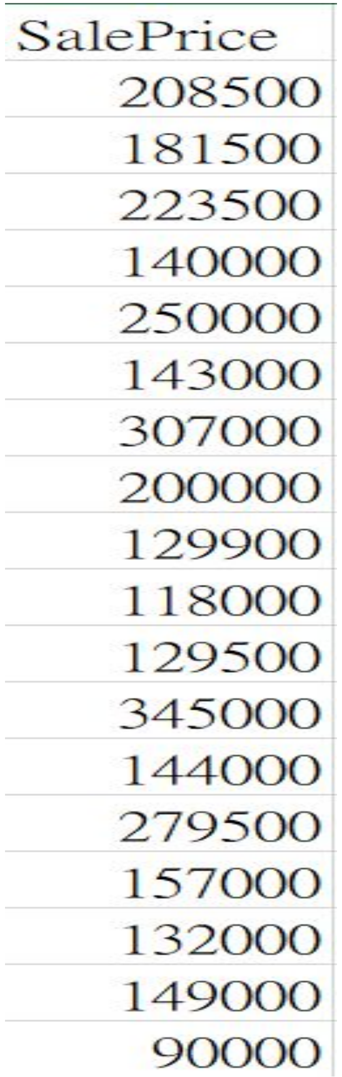
結果



訓練房價與面積關係

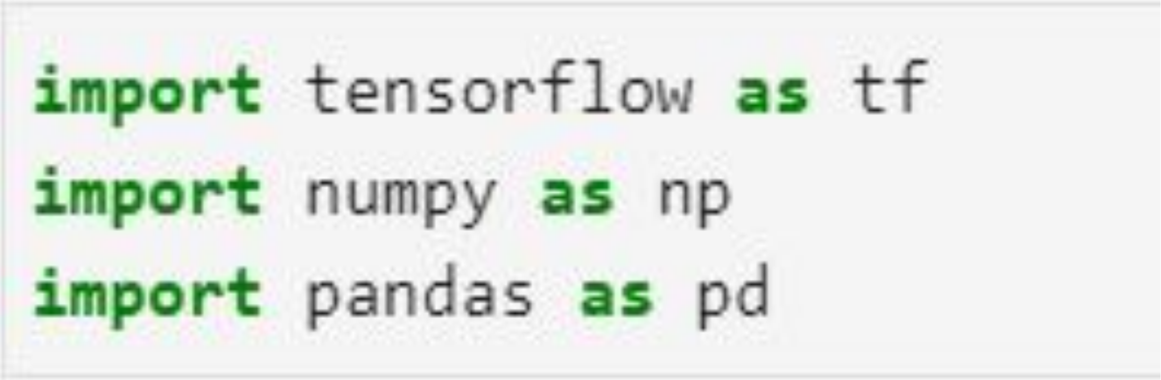


LotArea
8450
9600
11250
9550
14260
14115
10084
10382
6120
7420
11200
11924
12968
10652
10920
6120
11241
10791



SalePrice
208500
181500
223500
140000
250000
143000
307000
200000
129900
118000
129500
345000
144000
279500
157000
132000
149000
90000

Import library



```
import tensorflow as tf
import numpy as np
import pandas as pd
```

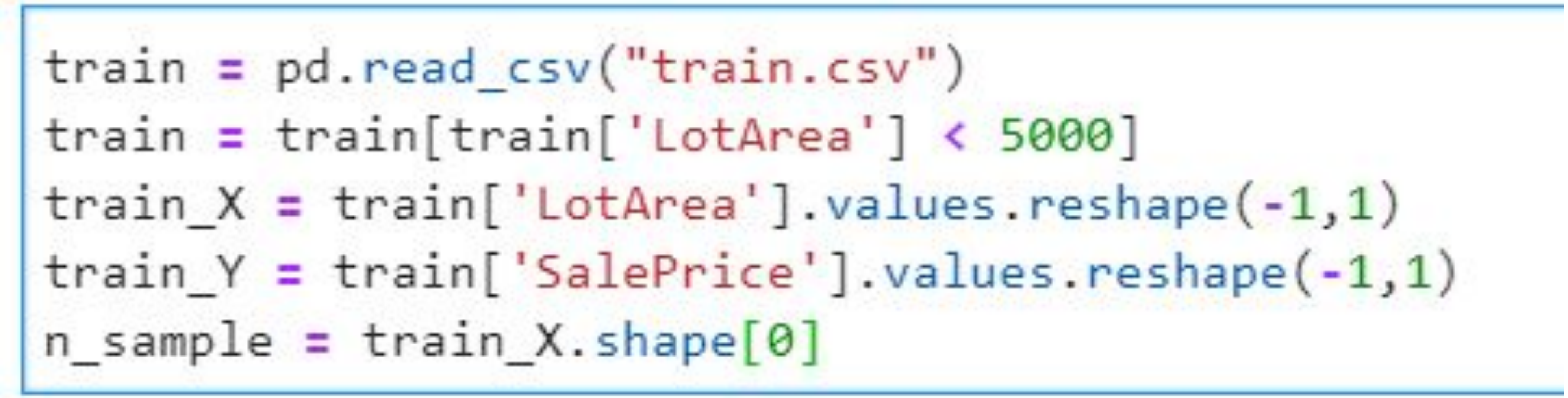
Import library

```
import pandas as pd
```

pandas是Python程式語言的用於數據操縱和分析的軟體庫。

pandas

dataset



```
train = pd.read_csv("train.csv")
train = train[train['LotArea'] < 5000]
train_X = train['LotArea'].values.reshape(-1,1)
train_Y = train['SalePrice'].values.reshape(-1,1)
n_sample = train_X.shape[0]
```

dataset

pd.read_csv("train.csv")

->讀取csv資料

train = train[train['LotArea'] < 5000]

->只讀取‘LotArea’中小於5000的資料

dataset

train_X = train['LotArea'].values.reshape(-1,1)

-> LotArea資料擺入train_X

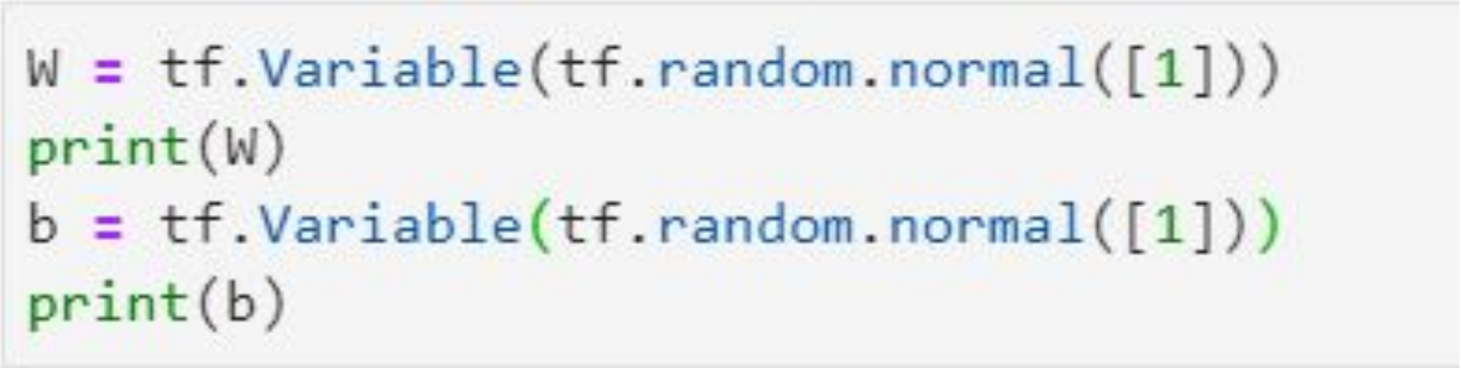
train_Y = train['SalePrice'].values.reshape(-1,1)

-> SalePrice資料擺入train_Y

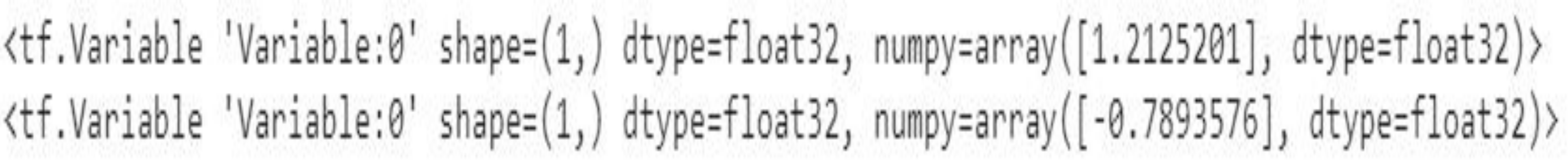
n_sample = train_X.shape[0]

-> n_sample 為train_X有幾筆資料

權重初始化

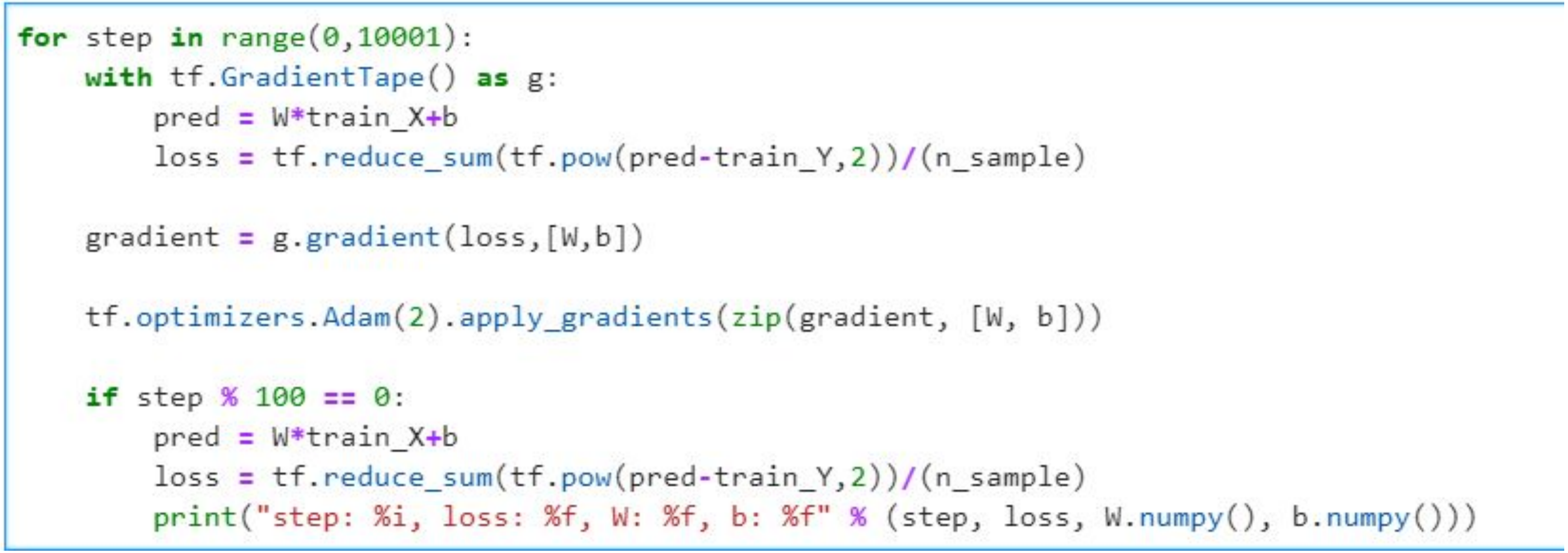


```
W = tf.Variable(tf.random.normal([1]))
print(W)
b = tf.Variable(tf.random.normal([1]))
print(b)
```



```
<tf.Variable 'Variable:0' shape=(1,) dtype=float32, numpy=array([1.2125201], dtype=float32)>
<tf.Variable 'Variable:0' shape=(1,) dtype=float32, numpy=array([-0.7893576], dtype=float32)>
```

訓練

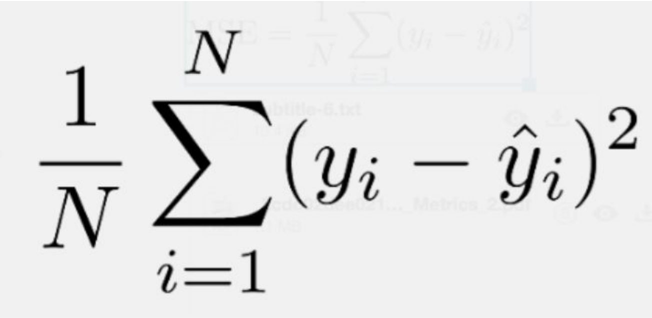


```
for step in range(0,10001):
    with tf.GradientTape() as g:
        pred = W*train_X+b
        loss = tf.reduce_sum(tf.pow(pred-train_Y,2))/(n_sample)

    gradient = g.gradient(loss,[W,b])

    tf.optimizers.Adam(2).apply_gradients(zip(gradient, [W, b]))

    if step % 100 == 0:
        pred = W*train_X+b
        loss = tf.reduce_sum(tf.pow(pred-train_Y,2))/(n_sample)
        print("step: %i, loss: %f, W: %f, b: %f" % (step, loss, W.numpy(), b.numpy()))
```

預測為W*train_X+b

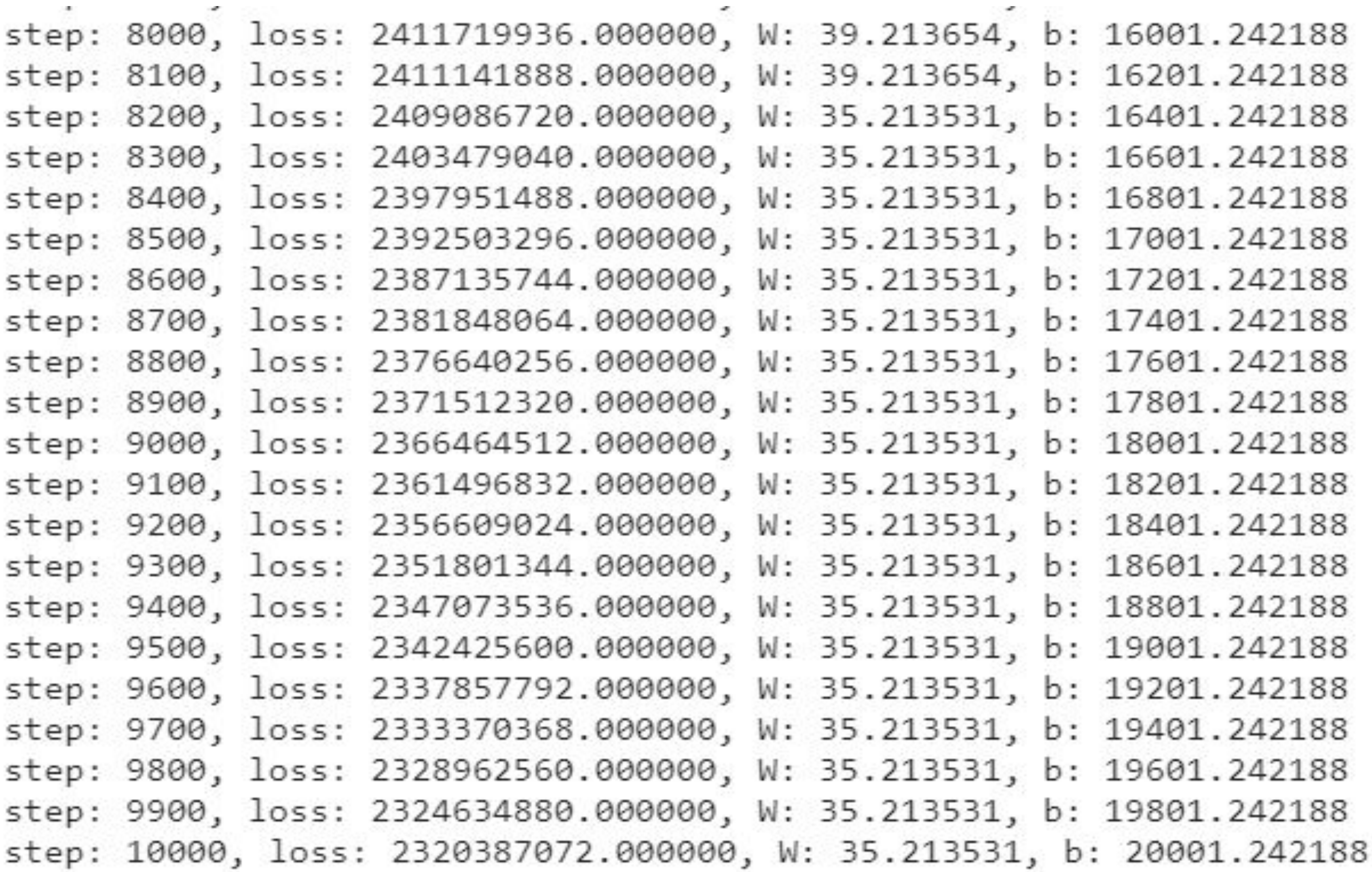
Loss採用mean square

Optimizer採用Adam



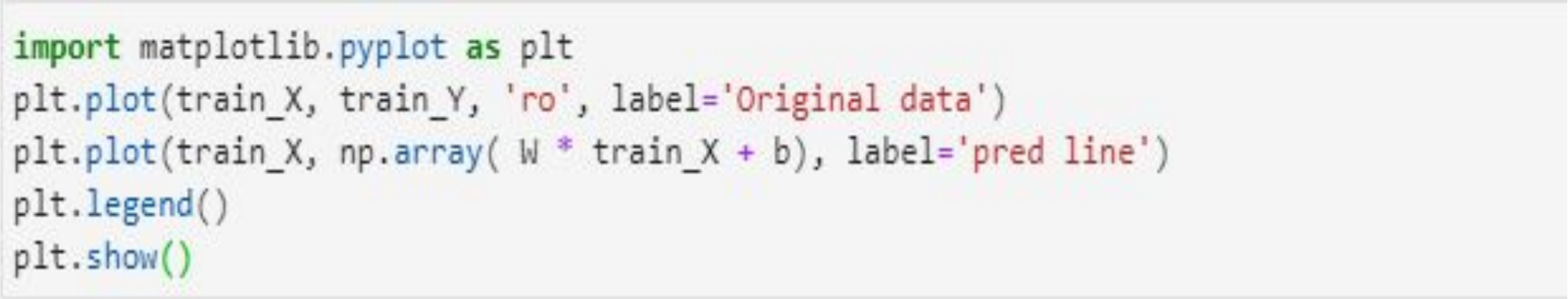
$$\frac{1}{N}\sum_{i=1}^{N}(y_i-\hat{y}_i)^2$$

訓練



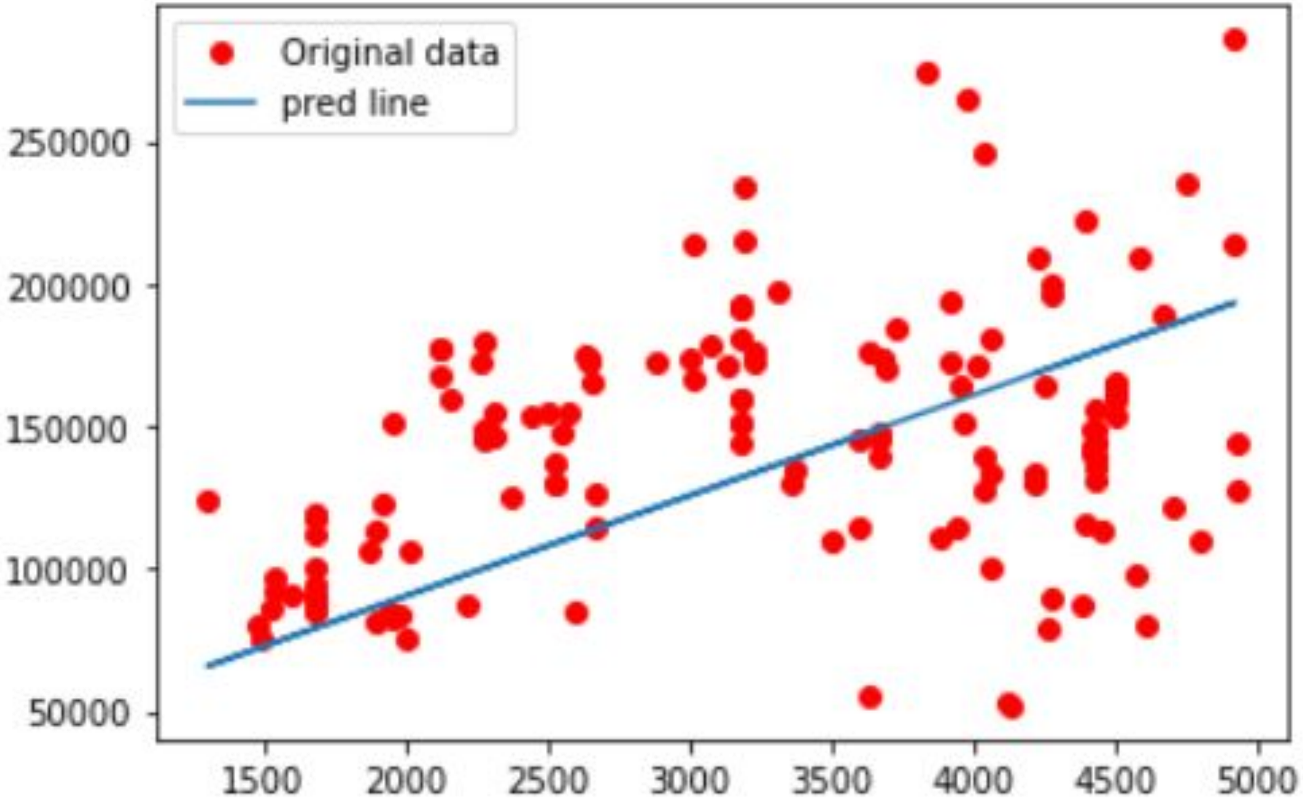
```
step: 8000, loss: 2411719936.000000, W: 39.213654, b: 16001.242188
step: 8100, loss: 2411141888.000000, W: 39.213654, b: 16201.242188
step: 8200, loss: 2409086720.000000, W: 35.213531, b: 16401.242188
step: 8300, loss: 2403479040.000000, W: 35.213531, b: 16601.242188
step: 8400, loss: 2397951488.000000, W: 35.213531, b: 16801.242188
step: 8500, loss: 2392503296.000000, W: 35.213531, b: 17001.242188
step: 8600, loss: 2387135744.000000, W: 35.213531, b: 17201.242188
step: 8700, loss: 2381848064.000000, W: 35.213531, b: 17401.242188
step: 8800, loss: 2376640256.000000, W: 35.213531, b: 17601.242188
step: 8900, loss: 2371512320.000000, W: 35.213531, b: 17801.242188
step: 9000, loss: 2366464512.000000, W: 35.213531, b: 18001.242188
step: 9100, loss: 2361496832.000000, W: 35.213531, b: 18201.242188
step: 9200, loss: 2356609024.000000, W: 35.213531, b: 18401.242188
step: 9300, loss: 2351801344.000000, W: 35.213531, b: 18601.242188
step: 9400, loss: 2347073536.000000, W: 35.213531, b: 18801.242188
step: 9500, loss: 2342425600.000000, W: 35.213531, b: 19001.242188
step: 9600, loss: 2337857792.000000, W: 35.213531, b: 19201.242188
step: 9700, loss: 2333370368.000000, W: 35.213531, b: 19401.242188
step: 9800, loss: 2328962560.000000, W: 35.213531, b: 19601.242188
step: 9900, loss: 2324634880.000000, W: 35.213531, b: 19801.242188
step: 10000, loss: 2320387072.000000, W: 35.213531, b: 20001.242188
```

結果



```
import matplotlib.pyplot as plt
plt.plot(train_X, train_Y, 'ro', label='Original data')
plt.plot(train_X, np.array( W * train_X + b), label='pred line')
plt.legend()
plt.show()
```

結果


Original data
pred line
250000
200000
150000
100000
50000
1500
2000
2500
3000
3500
4000
4500
5000

END!!!