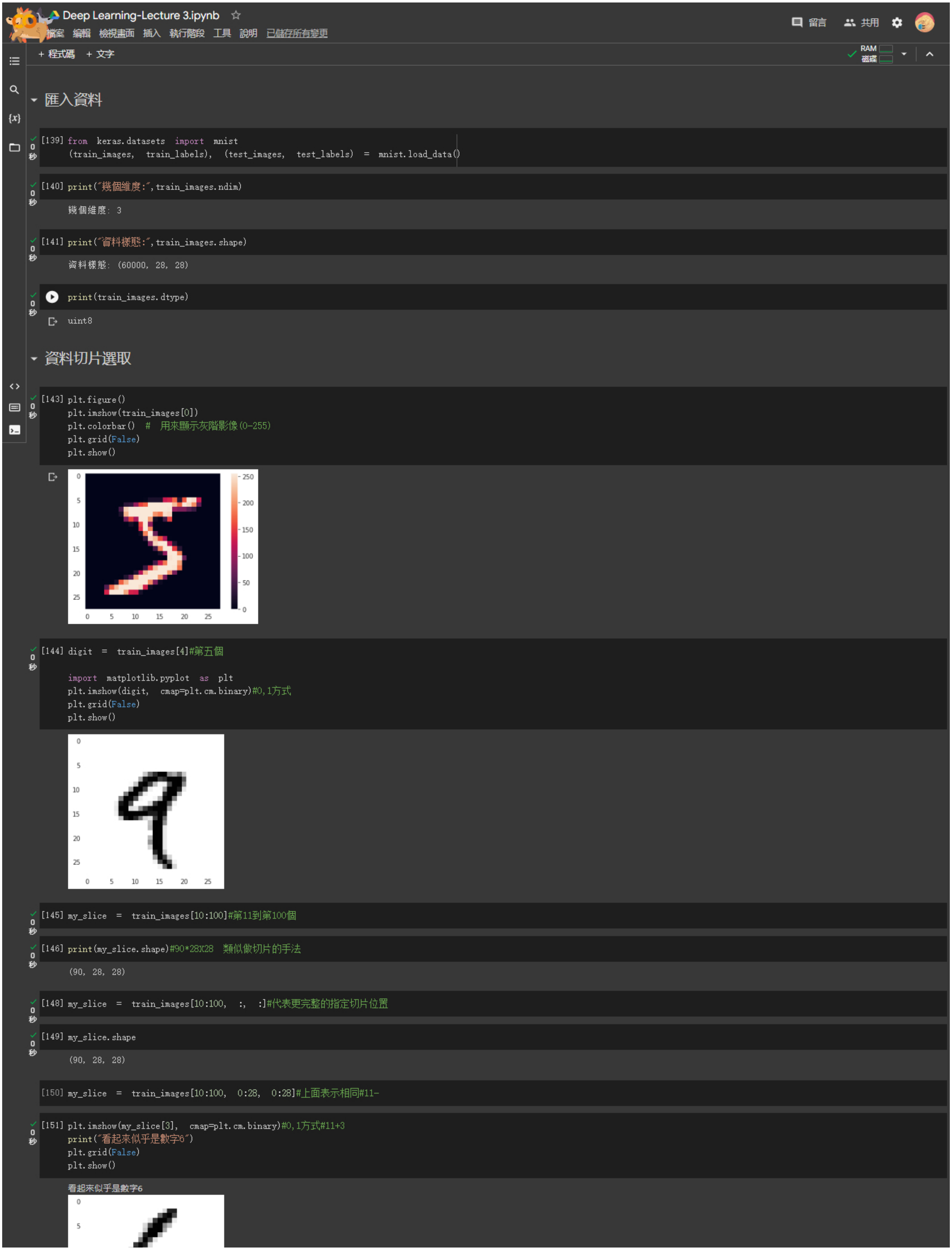

Deep Learning-Lecture 3.ipynb
檔案 編輯 檢視畫面 插入 執行階段 工具 說明 已儲存所有變更
留言
共用
+ 程式碼 + 文字
RAM
磁碟

匯入資料

```
[139] from keras.datasets import mnist
      (train_images, train_labels), (test_images, test_labels) = mnist.load_data()
```

```
[140] print("幾個維度:",train_images.ndim)
```

```
幾個維度: 3
```

```
[141] print("資料樣態:",train_images.shape)
```

```
資料樣態: (60000, 28, 28)
```

```
print(train_images.dtype)
```

```
uint8
```

資料切片選取

```
[143] plt.figure()
      plt.imshow(train_images[0])
      plt.colorbar()  #  用來顯示灰階影像(0-255)
      plt.grid(False)
      plt.show()
```

0
5
10
15
20
25
0
5
10
15
20
25
250
200
150
100
50
0

```
[144] digit = train_images[4]#第五個

      import matplotlib.pyplot as plt
      plt.imshow(digit, cmap=plt.cm.binary)#0,1方式
      plt.grid(False)
      plt.show()
```

```
[145] my_slice = train_images[10:100]#第11到第100個
```

```
[146] print(my_slice.shape)#90*28X28  類似做切片的手法
```

```
(90, 28, 28)
```

```
[148] my_slice = train_images[10:100, :, :]#代表更完整的指定切片位置
```

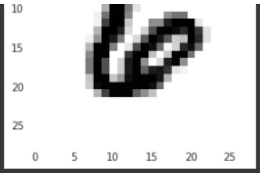
```
[149] my_slice.shape
```

```
(90, 28, 28)
```

```
[150] my_slice = train_images[10:100, 0:28, 0:28]#上面表示相同#11-
```

```
[151] plt.imshow(my_slice[3], cmap=plt.cm.binary)#0,1方式#11+3
      print("看起來似乎是數字6")
      plt.grid(False)
      plt.show()
```

```
看起來似乎是數字6
```



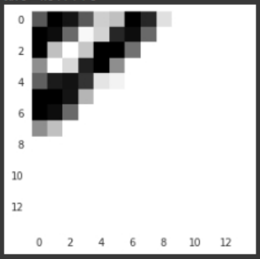
```
[152] my_slice.shape
```

```
(90, 28, 28)
```

```
[153] my_slice = train_images[:,14:,14:]#15-28
```

```
[154] plt.imshow(my_slice[13], cmap=plt.cm.binary)#0,1方式
      print("數字6的右下角")
      plt.grid(False)
      plt.show()
```

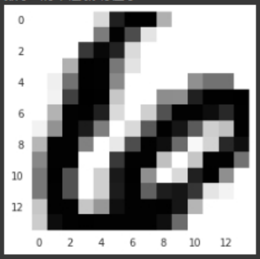
```
數字6的右下角
```



```
[155] my_slice = train_images[: ,7:-7, 7:-7]#原本第一個
```

```
[156] plt.imshow(my_slice[13], cmap=plt.cm.binary)#0,1方式
      print("數字6的中間裁切位子")
      plt.grid(False)
      plt.show()
```

```
數字6的中間裁切位子
```



Numpy array

```
[157] import numpy as np
```

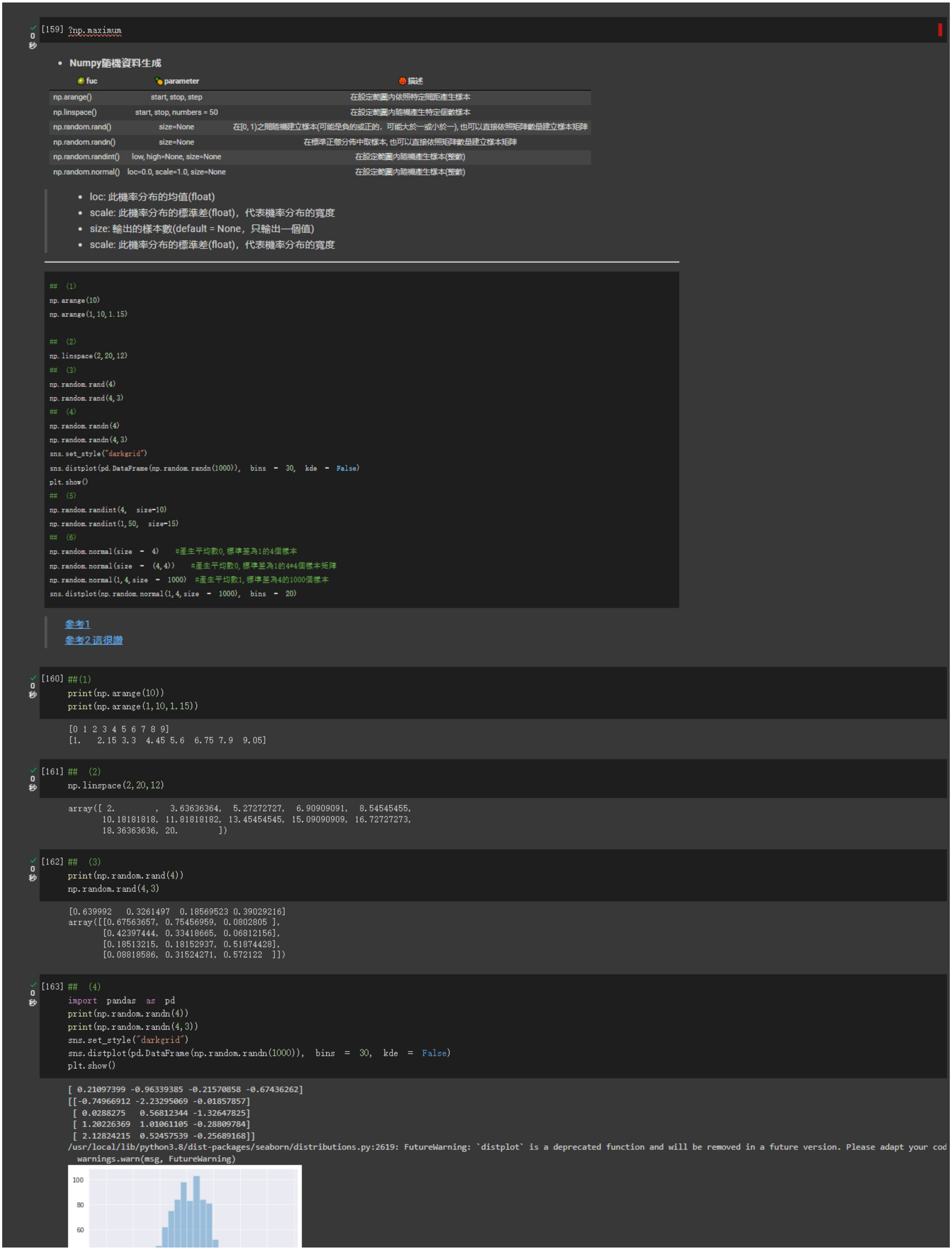
- **Numpy max() maximum()差別**
  - np.max() 最少接收一個參數，回傳所有內容的最大值
  - np.maximum(A, B) 最少接收兩個參數，回傳A與B逐個比較的最大值

```
import numpy as np
value = [1, 2, 3, 4, 9, 8, 7, 6]
print(np.max(value))

value = [1, 2, 3, 4, 5, 4, 9, 0, 8]
print(np.maximum(value, 4))
```

> 參考

```
[158] x = np.array([1,3,2,4]).reshape(2,2)
      y = np.array([-5,6,1,2])
      y.shape = (2,2)
      print("x:\n",x)
      print("y:\n",y)
      z = x + y
      z = np.maximum(z,0.)
      print("z:\n",z)
```

```
x:
 [[1 3]
 [2 4]]
y:
 [[-5  6]
 [ 1  2]]
z:
 [[0. 9.]
 [3. 6.]]
```



```
[159] ?np.maximum
```

- **Numpy隨機資料生成**

fuc	parameter	描述
np.arange()	start, stop, step	在設定範圍內依照特定間距產生樣本
np.linspace()	start, stop, numbers = 50	在設定範圍內隨機產生特定個數樣本
np.random.rand()	size=None	在[0, 1)之間隨機建立樣本(可能是負的或正的，可能大於一或小於一), 也可以直接依照矩陣數量建立樣本矩陣
np.random.randn()	size=None	在標準正態分佈中取樣本, 也可以直接依照矩陣數量建立樣本矩陣
np.random.randint()	low, high=None, size=None	在設定範圍內隨機產生樣本(整數)
np.random.normal()	loc=0.0, scale=1.0, size=None	在設定範圍內隨機產生樣本(整數)

> - loc: 此機率分布的均值(float)
> - scale: 此機率分布的標準差(float)，代表機率分布的寬度
> - size: 輸出的樣本數(default = None，只輸出一個值)
> - scale: 此機率分布的標準差(float)，代表機率分布的寬度

```
## (1)
np.arange(10)
np.arange(1,10,1.15)

## (2)
np.linspace(2,20,12)
## (3)
np.random.rand(4)
np.random.rand(4,3)
## (4)
np.random.randn(4)
np.random.randn(4,3)
sns.set_style("darkgrid")
sns.distplot(pd.DataFrame(np.random.randn(1000)), bins = 30, kde = False)
plt.show()
## (5)
np.random.randint(4, size=10)
np.random.randint(1,50, size=15)
## (6)
np.random.normal(size = 4)   #產生平均數0,標準差為1的4個樣本
np.random.normal(size = (4,4))   #產生平均數0,標準差為1的4*4個樣本矩陣
np.random.normal(1,4,size = 1000)  #產生平均數1,標準差為4的1000個樣本
sns.distplot(np.random.normal(1,4,size = 1000), bins = 20)
```

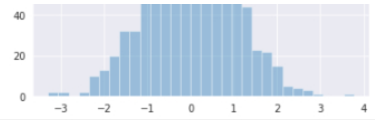
> 參考1
> 參考2 這很讚

```
[160] ##(1)
      print(np.arange(10))
      print(np.arange(1,10,1.15))

      [0 1 2 3 4 5 6 7 8 9]
      [1.   2.15 3.3  4.45 5.6  6.75 7.9  9.05]
```

```
[161] ##  (2)
      np.linspace(2,20,12)

      array([ 2.        ,  3.63636364,  5.27272727,  6.90909091,  8.54545455,
             10.18181818, 11.81818182, 13.45454545, 15.09090909, 16.72727273,
             18.36363636, 20.        ])
```

```
[162] ##  (3)
      print(np.random.rand(4))
      np.random.rand(4,3)

      [0.639992   0.3261497  0.18569523 0.39029216]
      array([[0.67563657, 0.75456959, 0.0802805 ],
             [0.42397444, 0.33418665, 0.06812156],
             [0.18513215, 0.18152937, 0.51874428],
             [0.08818586, 0.31524271, 0.572122  ]])
```

```
[163] ##  (4)
      import pandas as pd
      print(np.random.randn(4))
      print(np.random.randn(4,3))
      sns.set_style("darkgrid")
      sns.distplot(pd.DataFrame(np.random.randn(1000)), bins = 30, kde = False)
      plt.show()

      [ 0.21097399 -0.96339385 -0.21570858 -0.67436262]
      [[-0.74966912 -2.23295069 -0.01857857]
       [ 0.0288275   0.56812344 -1.32647825]
       [ 1.20226369  1.01061105 -0.28809784]
       [ 2.12824215  0.52457539 -0.25689168]]
      /usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your cod
        warnings.warn(msg, FutureWarning)
```



✓ 0 秒

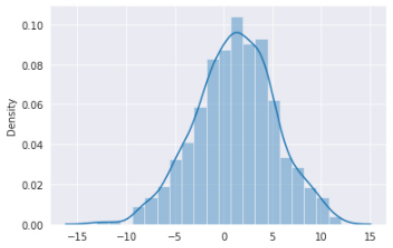
```
[164] ##  (5)
      print(np.random.randint(4,  size=10))
      np.random.randint(1,50,  size=15)
```

```
[0 0 2 3 2 0 2 2 1 3]
array([31, 22, 39, 42, 23, 16, 14, 32,  4, 18, 45, 34, 48, 34,  2])
```

✓ 0 秒

```
[165] ##  (6)
      import  seaborn  as  sns
      np.random.normal(size  =  4)    #產生平均數0,標準差為1的4個樣本
      np.random.normal(size  =  (4,4))    #產生平均數0,標準差為1的4*4個樣本矩陣
      np.random.normal(1,4,size  =  1000)  #產生平均數1,標準差為4的1000個樣本
      sns.distplot(np.random.normal(1,4,size  =  1000),  bins  =  20)
```

```
<AxesSubplot:ylabel='Density'>
```



✓ 0 秒

```
[166] x  =  np.random.random((64,  3,  32,  10))
      y  =  np.random.random((32,  10))
      z  =  np.maximum(x,  y)
      print(x.ndim)
      print(y.ndim)
      print(z.ndim)
```

```
4
2
4
```

- Python 內積 點積大不同
  - dot與inner

```
a=np.array([[1,2],[3,4]])
b=np.array([[11,12],[13,14]])
np.dot(a,b)
array([[37,  40],
       [85,  92]])
np.inner(a,b)
array([[35,  41],
       [81,  95]])
```

✓ 0 秒

```
[167] a=np.array([[1,2],[3,4]])
      b=np.array([[11,12],[13,14]])
      print("a:  \n",a)
      print("b:  \n",b)
      print("平常用到dot:  \n",np.dot(a,b))
      print("inner:  \n",np.inner(a,b))
      print("element-wise  product:  \n",a*b)
```

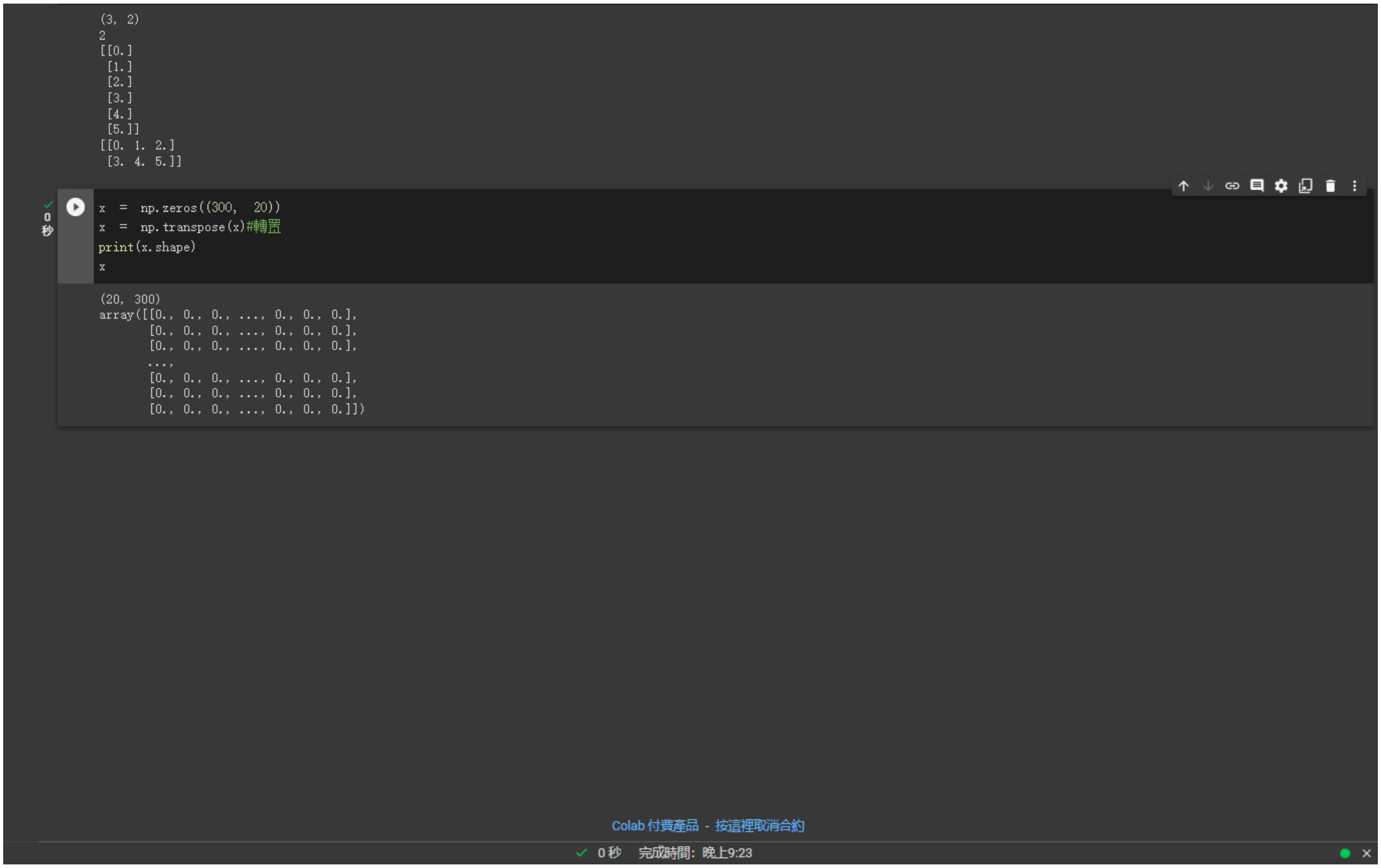
```
a:
 [[1 2]
 [3 4]]
b:
 [[11 12]
 [13 14]]
平常用到dot:
 [[37 40]
 [85 92]]
inner:
 [[35 41]
 [81 95]]
element-wise product:
 [[11 24]
 [39 56]]
```

✓ 0 秒

```
[168] import  numpy  as  np
```

✓ 0 秒

```
[169] x  =  np.array([[0.,  1.],[2.,  3.],[4.,  5.]])
      print(x.shape)
      print(x.ndim)
      x  =  x.reshape((6,  1))
      print(x)
      x  =  x.reshape((2,  3))
      print(x)
```



```
(3, 2)
2
[[0.]
 [1.]
 [2.]
 [3.]
 [4.]
 [5.]]
[[0. 1. 2.]
 [3. 4. 5.]]
```

```
x = np.zeros((300, 20))
x = np.transpose(x)#轉置
print(x.shape)
x
```

```
(20, 300)
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```