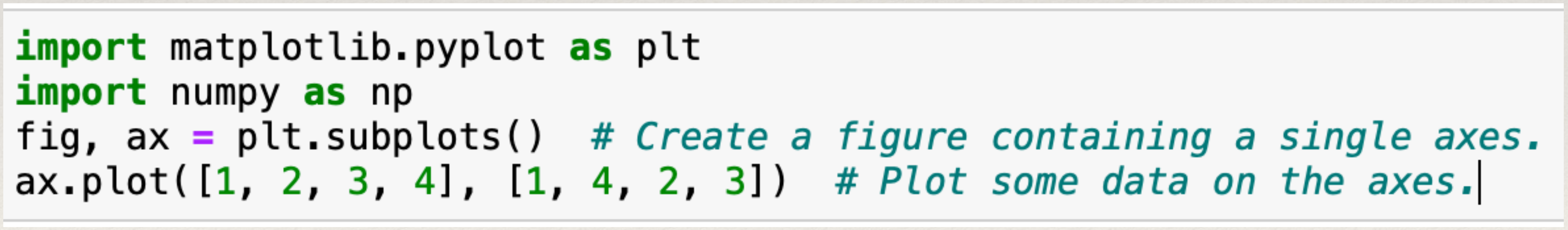
Lorem Ipsum Dolor

繪圖 Matplotlib

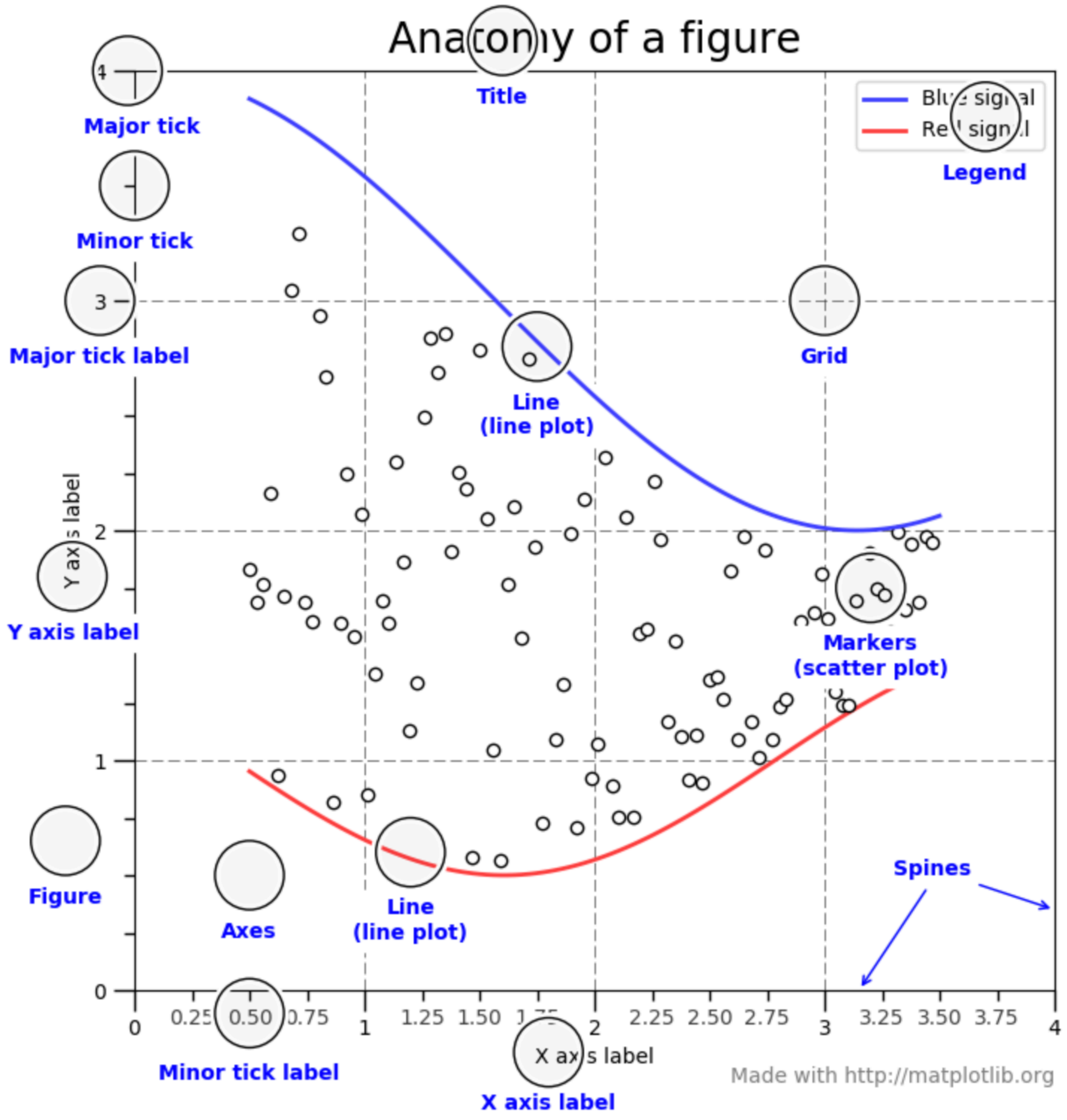
Telling Pan
telung@mac.com

簡單範例

- Matplotlib 使用於繪圖，圖形可以包含一個或多個 Axes (x, y 或是 x, y, z 的 3D 圖形)。
- 最簡單的方式是使用 pyplot.subplots 然後使用 Axes.plot ：



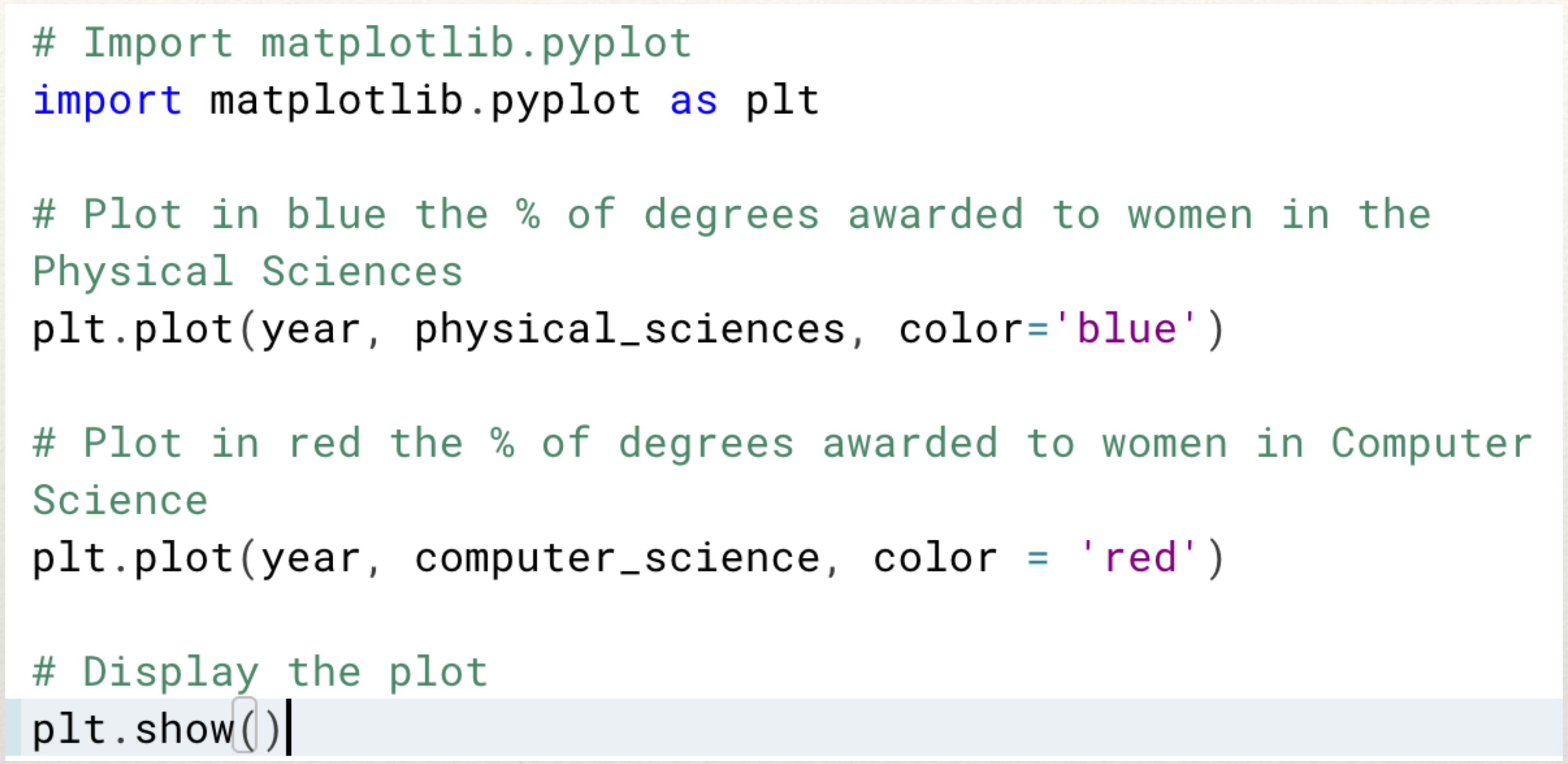
```python
import matplotlib.pyplot as plt
import numpy as np
fig, ax = plt.subplots()  # Create a figure containing a single axes.
ax.plot([1, 2, 3, 4], [1, 4, 2, 3])  # Plot some data on the axes.
```


Anatomy of a figure
Title
Major tick
Minor tick
Major tick label
Y axis label
Figure
Axes
Line (line plot)
Grid
Legend
Blue signal
Red signal
Markers (scatter plot)
Spines
Minor tick label
X axis label
Y axis label
0
0.25
0.50
0.75
1
1.25
1.50
1.75
2
2.25
2.50
2.75
3
3.25
3.50
3.75
4
Made with http://matplotlib.org

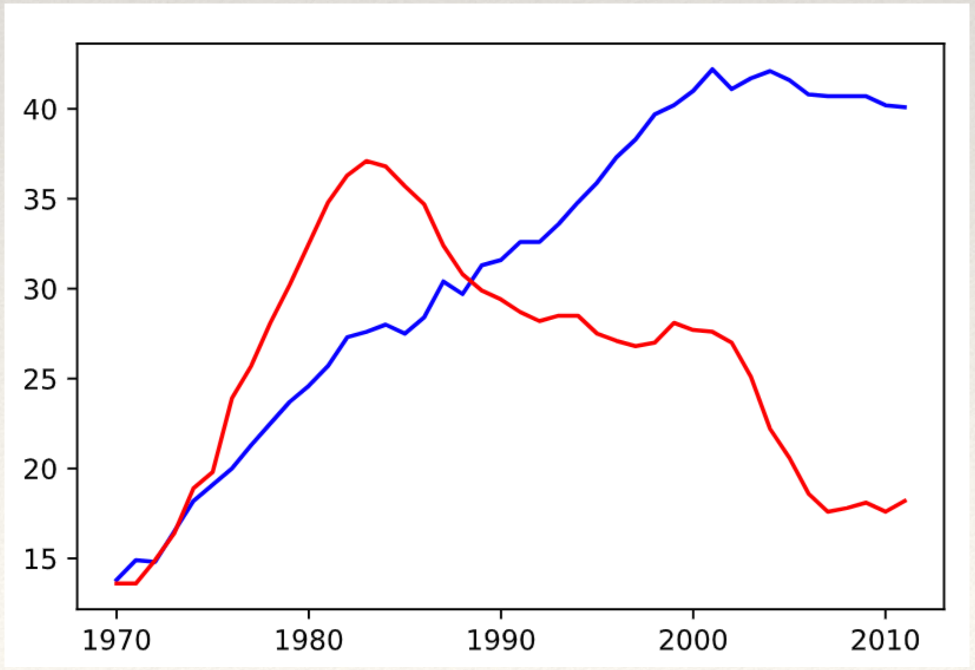
Multiple plots on single axis

- The data set here comes from records of undergraduate degrees awarded to women in a variety of fields from 1970 to 2011. You can compare trends in degrees most easily by viewing two curves on the same set of axes.
- Three NumPy arrays have been pre-loaded for you: year (enumerating years from 1970 to 2011 inclusive), physical_sciences (representing the percentage of Physical Sciences degrees awarded to women each in corresponding year), and computer_science (representing the percentage of Computer Science degrees awarded to women in each corresponding year).
- Two plt.plot() commands to draw line plots of different colors on the same set of axes. Here, year represents the x-axis,
  while physical_sciences and computer_science are the y-axes.

plot 語法範例 (此處省略資料的處理)

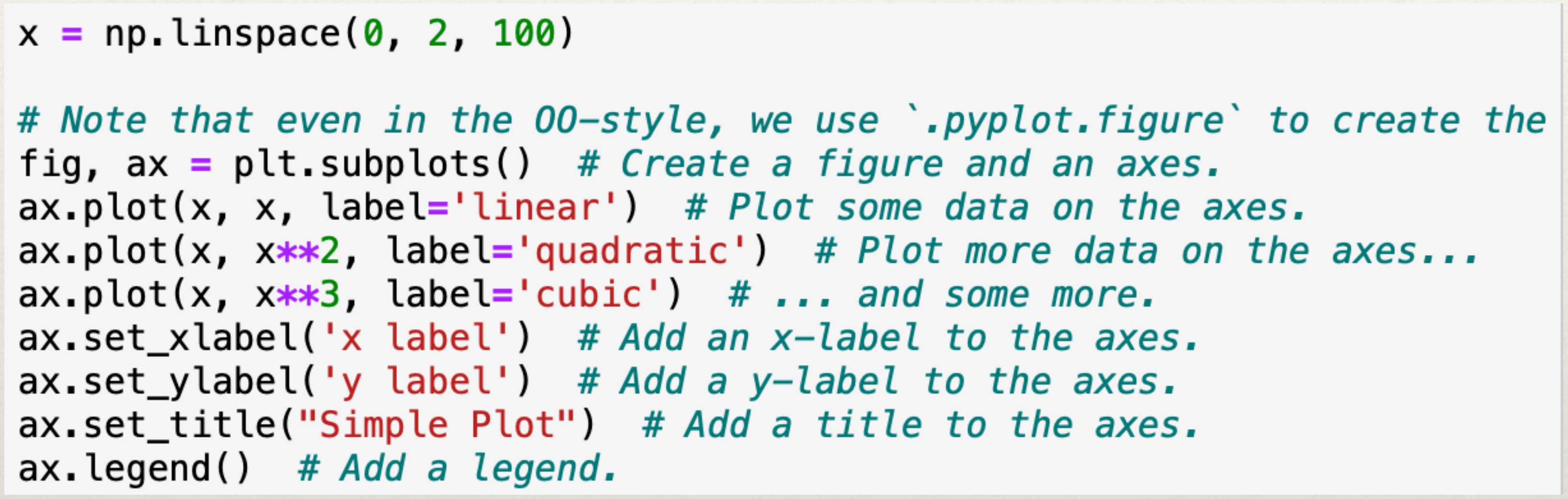


```
# Import matplotlib.pyplot
import matplotlib.pyplot as plt

# Plot in blue the % of degrees awarded to women in the
Physical Sciences
plt.plot(year, physical_sciences, color='blue')

# Plot in red the % of degrees awarded to women in Computer
Science
plt.plot(year, computer_science, color = 'red')

# Display the plot
plt.show()
```



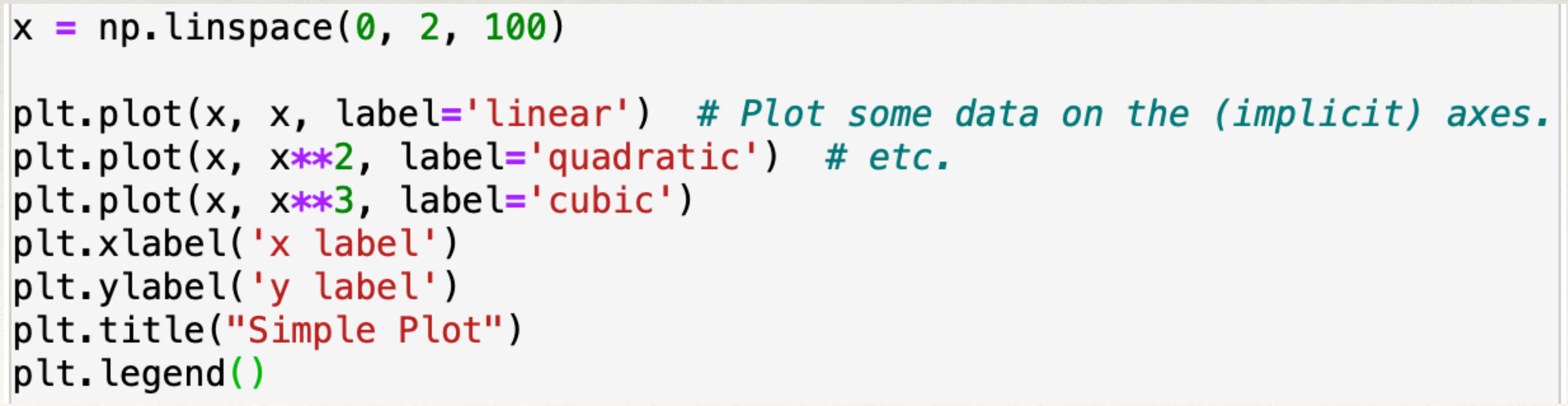
物件導向的介面

- 兩種建立方式：
  1. 明確建立 figures 與 axes



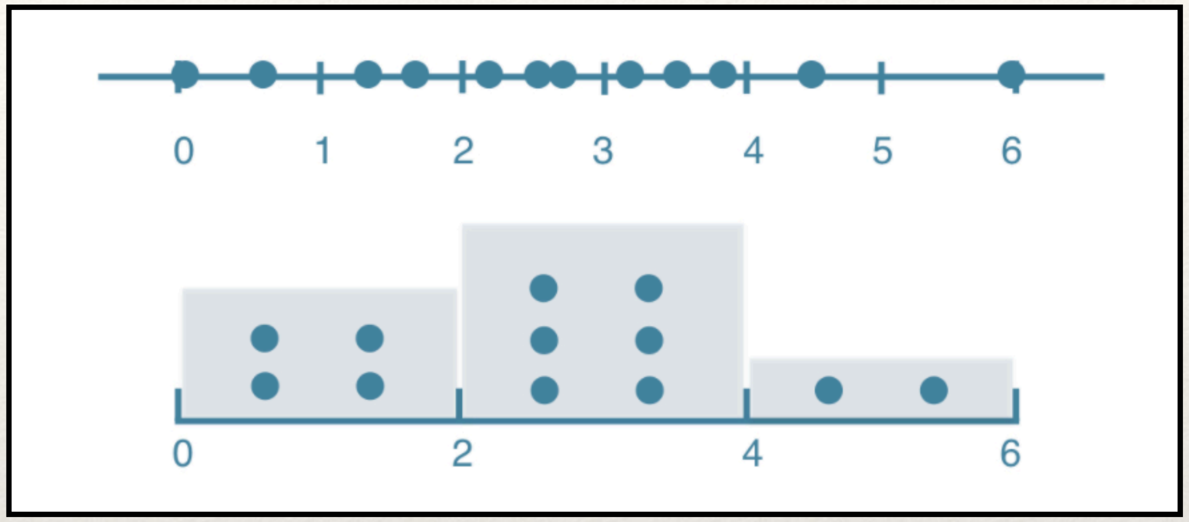
```python
x = np.linspace(0, 2, 100)

# Note that even in the OO-style, we use `.pyplot.figure` to create the
fig, ax = plt.subplots()  # Create a figure and an axes.
ax.plot(x, x, label='linear')  # Plot some data on the axes.
ax.plot(x, x**2, label='quadratic')  # Plot more data on the axes...
ax.plot(x, x**3, label='cubic')  # ... and some more.
ax.set_xlabel('x label')  # Add an x-label to the axes.
ax.set_ylabel('y label')  # Add a y-label to the axes.
ax.set_title("Simple Plot")  # Add a title to the axes.
ax.legend()  # Add a legend.
```

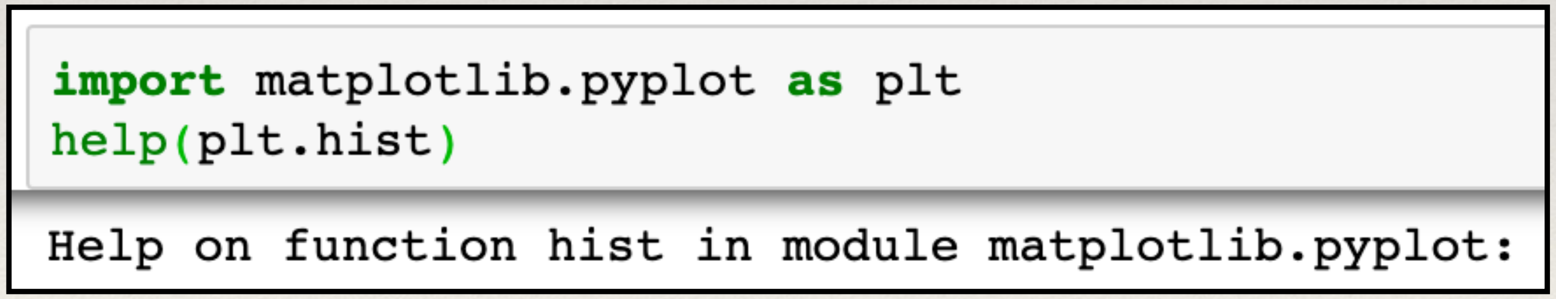
  2. 使用 pyplot



```python
x = np.linspace(0, 2, 100)

plt.plot(x, x, label='linear')  # Plot some data on the (implicit) axes.
plt.plot(x, x**2, label='quadratic')  # etc.
plt.plot(x, x**3, label='cubic')
plt.xlabel('x label')
plt.ylabel('y label')
plt.title("Simple Plot")
plt.legend()
```

直方圖 (Histogram)



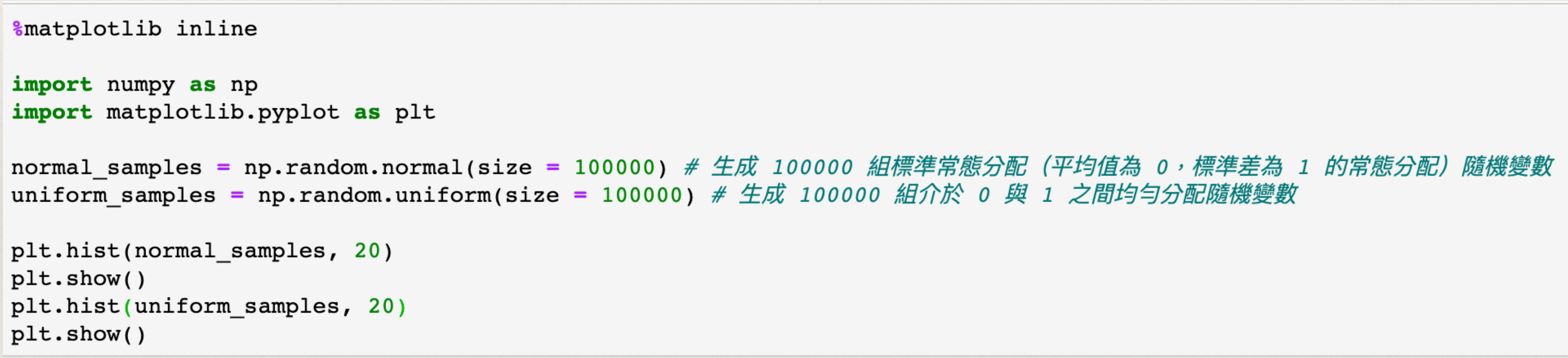
- 探索 dataset
- 瞭解資料分佈情形
- 

```
import matplotlib.pyplot as plt
help(plt.hist)
```

```
Help on function hist in module matplotlib.pyplot:
```

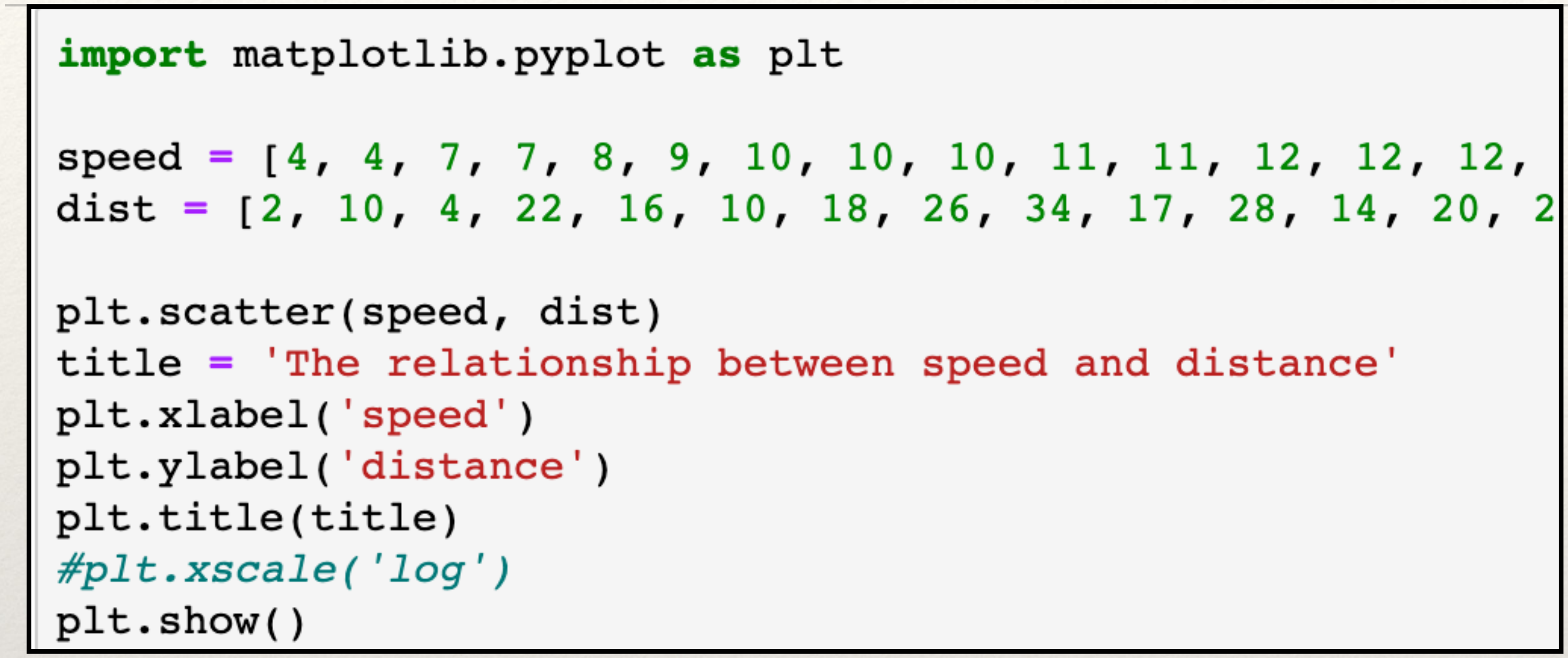
建立直方圖：bins

- Python 預設的 bin 為 10，太少無法呈現真正面貌，太多則過於複雜。



```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt

normal_samples = np.random.normal(size = 100000) # 生成 100000 組標準常態分配（平均值為 0，標準差為 1 的常態分配）隨機變數
uniform_samples = np.random.uniform(size = 100000) # 生成 100000 組介於 0 與 1 之間均勻分配隨機變數

plt.hist(normal_samples, 20)
plt.show()
plt.hist(uniform_samples, 20)
plt.show()
```

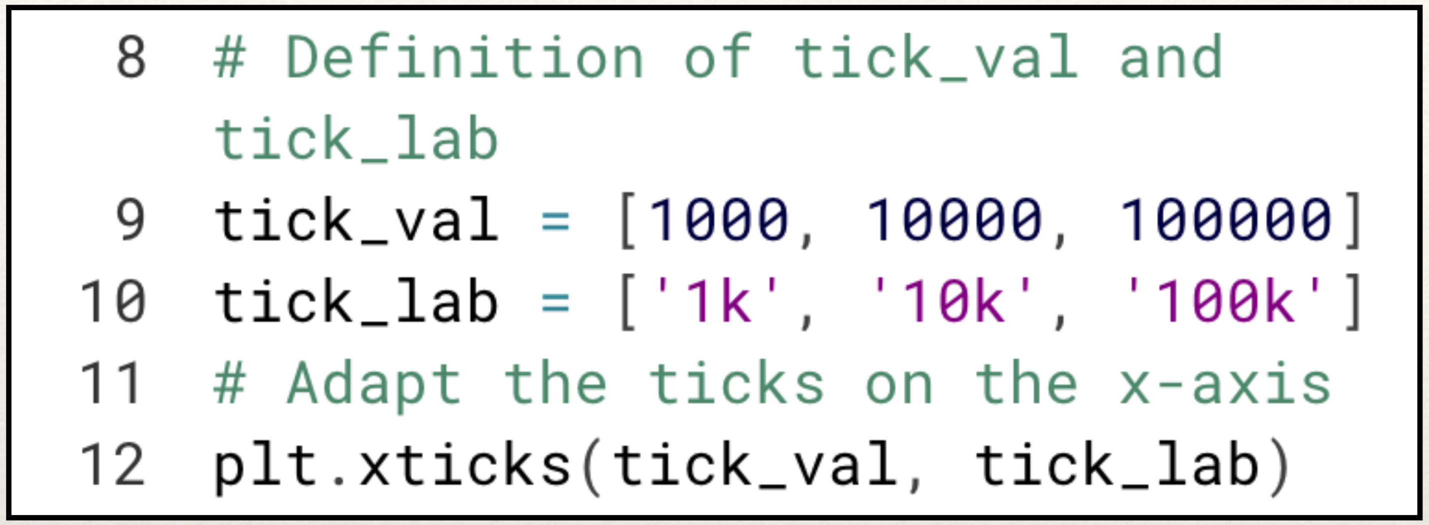
散佈圖（Scatter plot）



```
import matplotlib.pyplot as plt

speed = [4, 4, 7, 7, 8, 9, 10, 10, 10, 11, 11, 12, 12, 12,
dist = [2, 10, 4, 22, 16, 10, 18, 26, 34, 17, 28, 14, 20, 2

plt.scatter(speed, dist)
title = 'The relationship between speed and distance'
plt.xlabel('speed')
plt.ylabel('distance')
plt.title(title)
#plt.xscale('log')
plt.show()
```

speed = [4, 4, 7, 7, 8, 9, 10, 10, 10, 11, 11, 12, 12, 12, 12, 13, 13, 13, 13, 14, 14, 14, 14, 15, 15, 15, 16, 16, 17, 17, 17, 18, 18, 18, 18, 19, 19, 19, 20, 20, 20, 20, 20, 22, 23, 24, 24, 24, 24, 25]
dist = [2, 10, 4, 22, 16, 10, 18, 26, 34, 17, 28, 14, 20, 24, 28, 26, 34, 34, 46, 26, 36, 60, 80, 20, 26, 54, 32, 40, 32, 40, 50, 42, 56, 76, 84, 36, 46, 68, 32, 48, 52, 56, 64, 66, 54, 70, 92, 93, 120, 85]

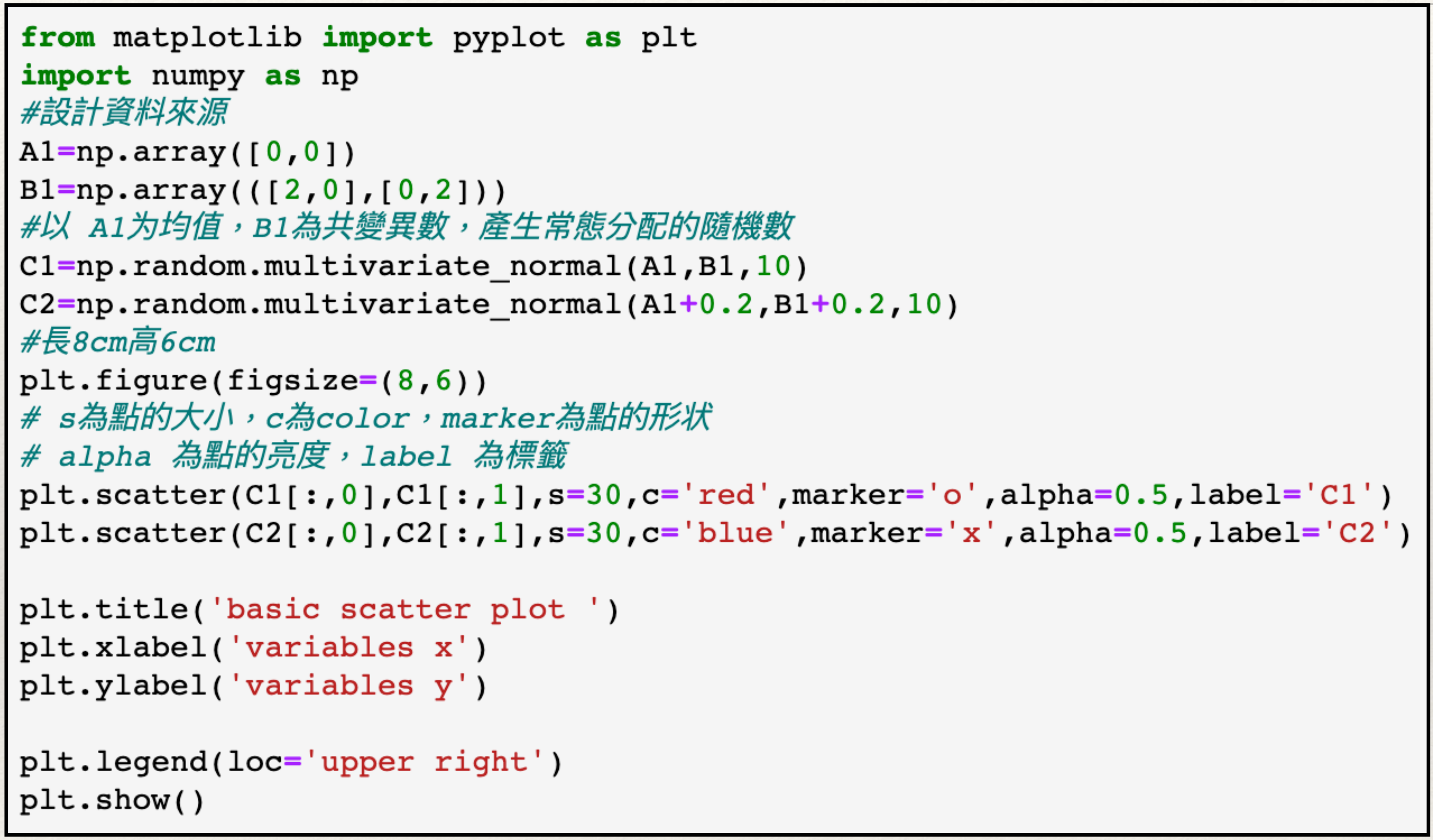
Ticks



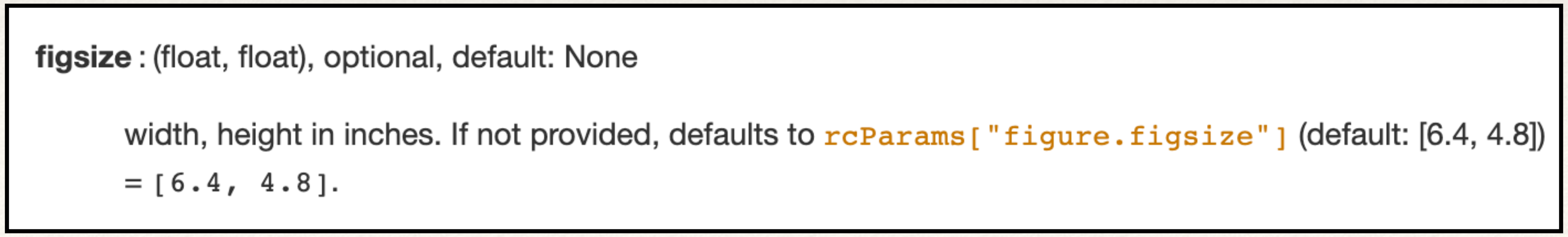
```python
# Definition of tick_val and tick_lab
tick_val = [1000, 10000, 100000]
tick_lab = ['1k', '10k', '100k']
# Adapt the ticks on the x-axis
plt.xticks(tick_val, tick_lab)
```

https://matplotlib.org/3.2.1/api/_as_gen/matplotlib.pyplot.figure.html

Scatter 案例練習 I



```python
from matplotlib import pyplot as plt
import numpy as np
#設計資料來源
A1=np.array([0,0])
B1=np.array(([2,0],[0,2]))
#以 A1为均值，B1為共變異數，產生常態分配的隨機數
C1=np.random.multivariate_normal(A1,B1,10)
C2=np.random.multivariate_normal(A1+0.2,B1+0.2,10)
#長8cm高6cm
plt.figure(figsize=(8,6))
# s為點的大小，c為color，marker為點的形狀
# alpha 為點的亮度，label 為標籤
plt.scatter(C1[:,0],C1[:,1],s=30,c='red',marker='o',alpha=0.5,label='C1')
plt.scatter(C2[:,0],C2[:,1],s=30,c='blue',marker='x',alpha=0.5,label='C2')

plt.title('basic scatter plot ')
plt.xlabel('variables x')
plt.ylabel('variables y')

plt.legend(loc='upper right')
plt.show()
```

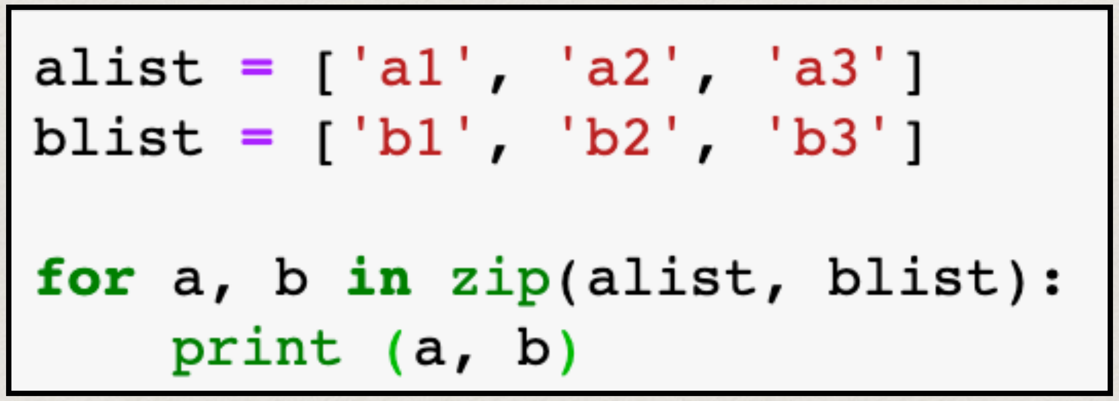


figsize : (float, float), optional, default: None

width, height in inches. If not provided, defaults to `rcParams["figure.figsize"]` (default: [6.4, 4.8]) `= [6.4, 4.8]`.

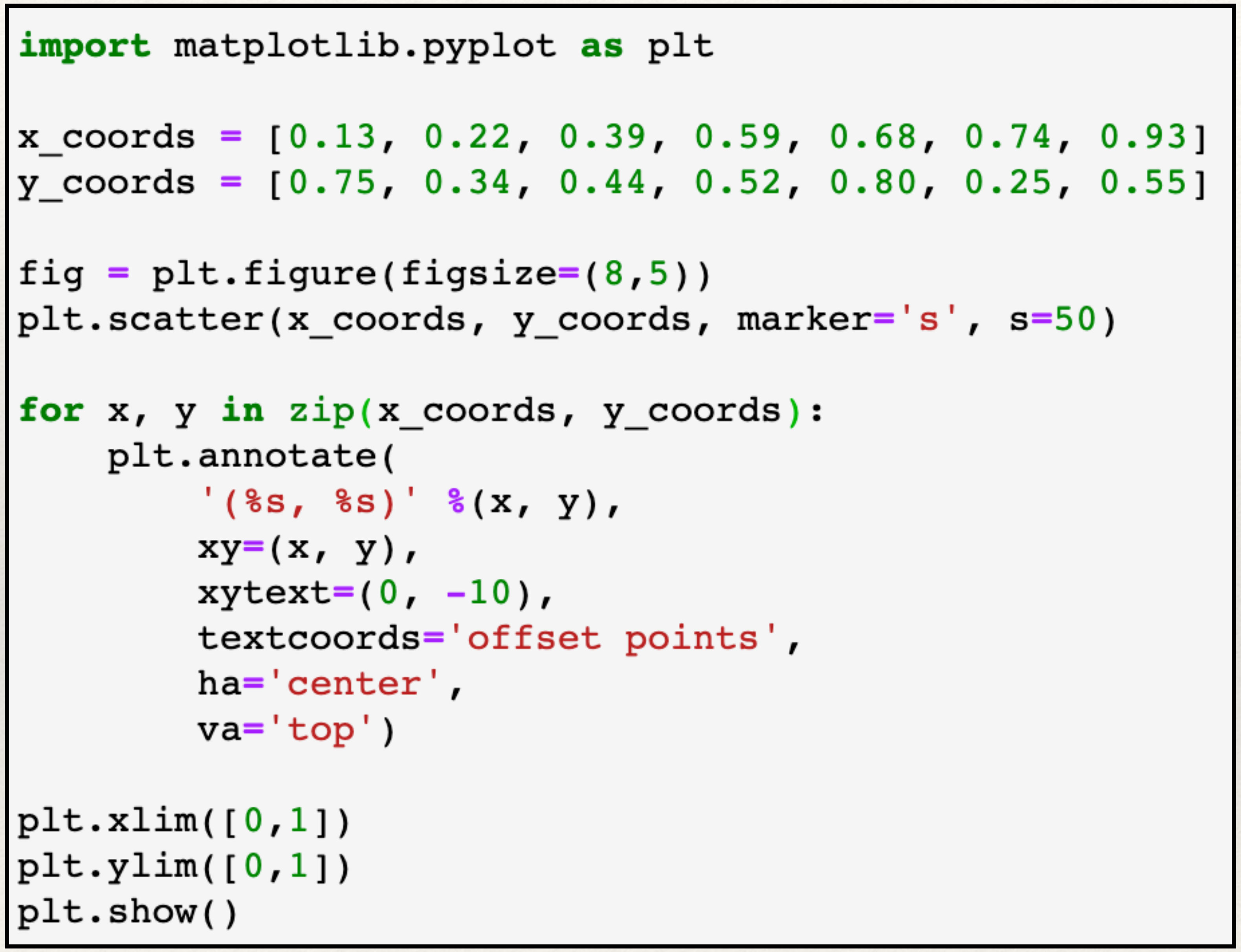
程式語法補充

- zip
- 迴圈



```
alist = ['a1', 'a2', 'a3']
blist = ['b1', 'b2', 'b3']

for a, b in zip(alist, blist):
    print (a, b)
```

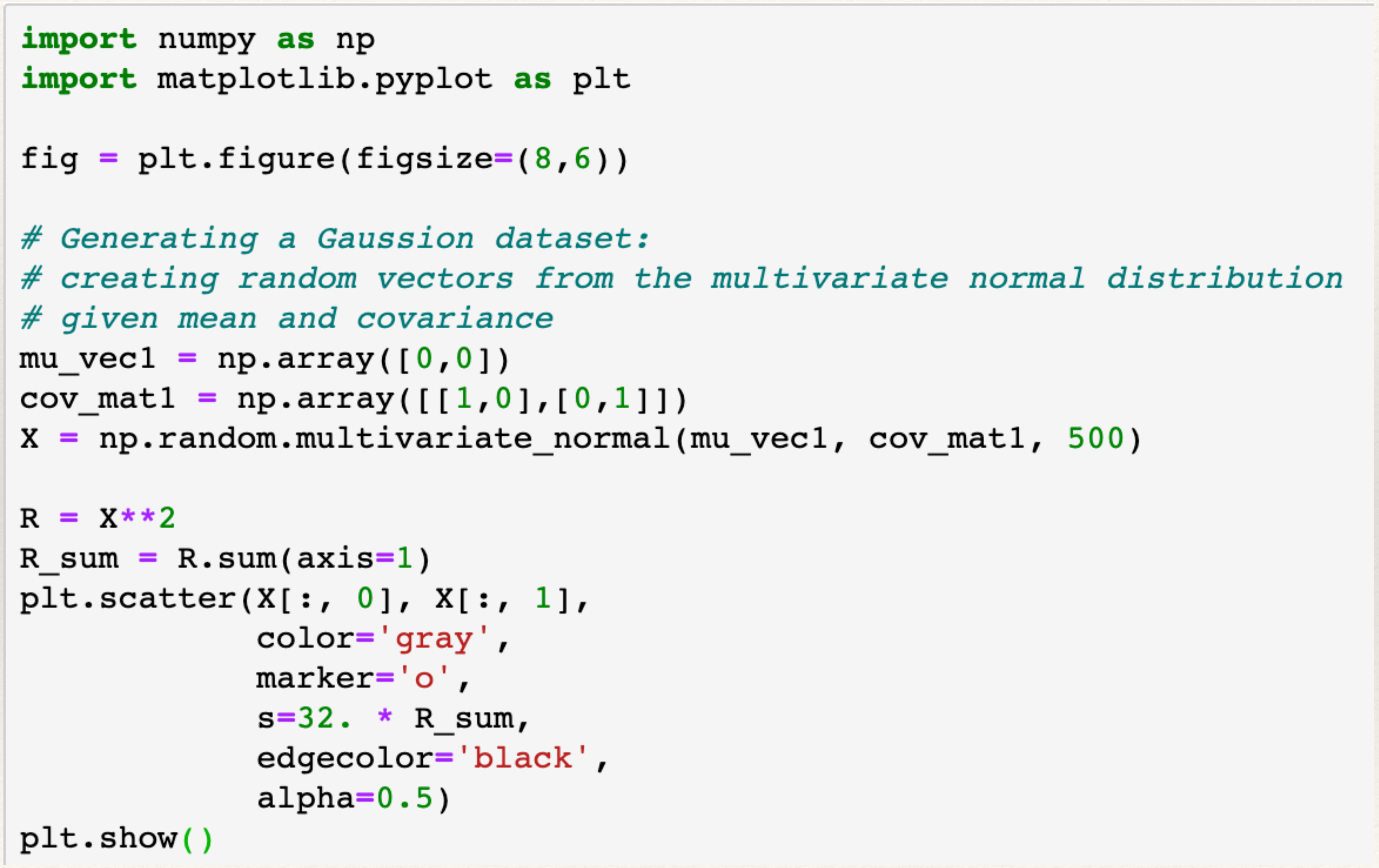
Scatter 案例練習 II



```python
import matplotlib.pyplot as plt

x_coords = [0.13, 0.22, 0.39, 0.59, 0.68, 0.74, 0.93]
y_coords = [0.75, 0.34, 0.44, 0.52, 0.80, 0.25, 0.55]

fig = plt.figure(figsize=(8,5))
plt.scatter(x_coords, y_coords, marker='s', s=50)

for x, y in zip(x_coords, y_coords):
    plt.annotate(
        '(%s, %s)' %(x, y),
        xy=(x, y),
        xytext=(0, -10),
        textcoords='offset points',
        ha='center',
        va='top')

plt.xlim([0,1])
plt.ylim([0,1])
plt.show()
```

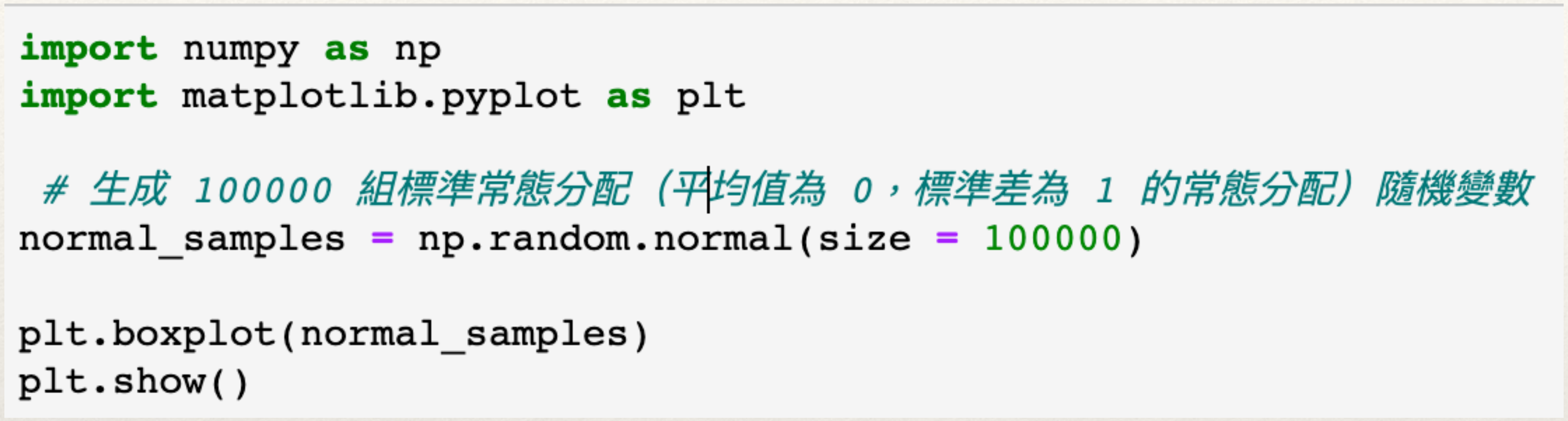
https://matplotlib.org/3.1.1/api/markers_api.html

Scatter 案例練習 III



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

fig = plt.figure(figsize=(8,6))

# Generating a Gaussion dataset:
# creating random vectors from the multivariate normal distribution
# given mean and covariance
mu_vec1 = np.array([0,0])
cov_mat1 = np.array([[1,0],[0,1]])
X = np.random.multivariate_normal(mu_vec1, cov_mat1, 500)

R = X**2
R_sum = R.sum(axis=1)
plt.scatter(X[:, 0], X[:, 1],
            color='gray',
            marker='o',
            s=32. * R_sum,
            edgecolor='black',
            alpha=0.5)
plt.show()
```

盒鬚圖（Box plot）

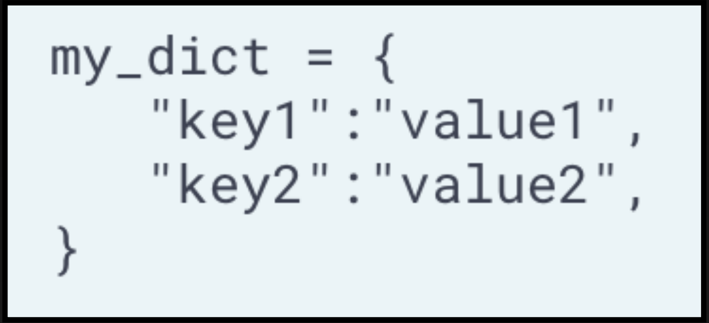


```python
import numpy as np
import matplotlib.pyplot as plt

 # 生成 100000 組標準常態分配（平均值為 0，標準差為 1 的常態分配）隨機變數
normal_samples = np.random.normal(size = 100000)

plt.boxplot(normal_samples)
plt.show()
```

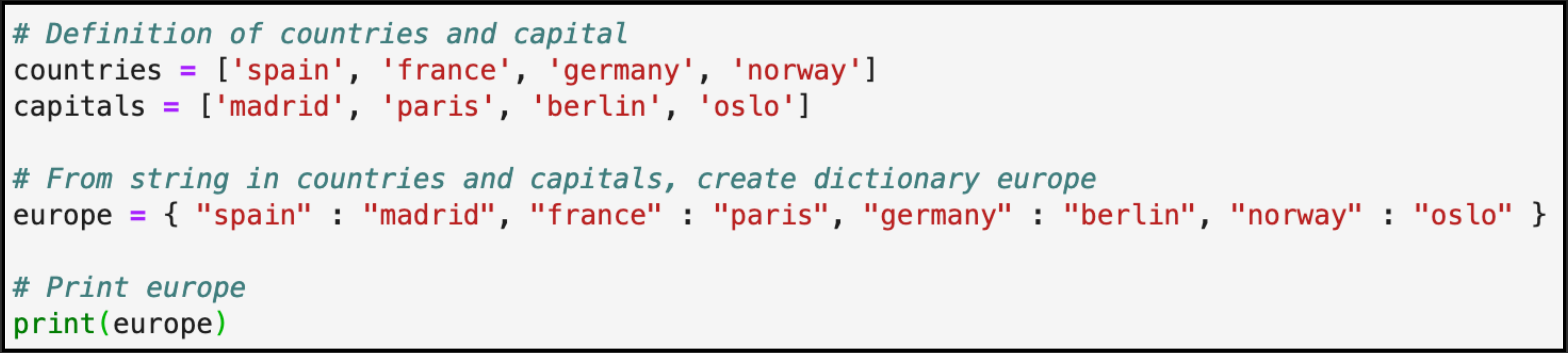
Dictionary

Dictionary

dictionary 的格式

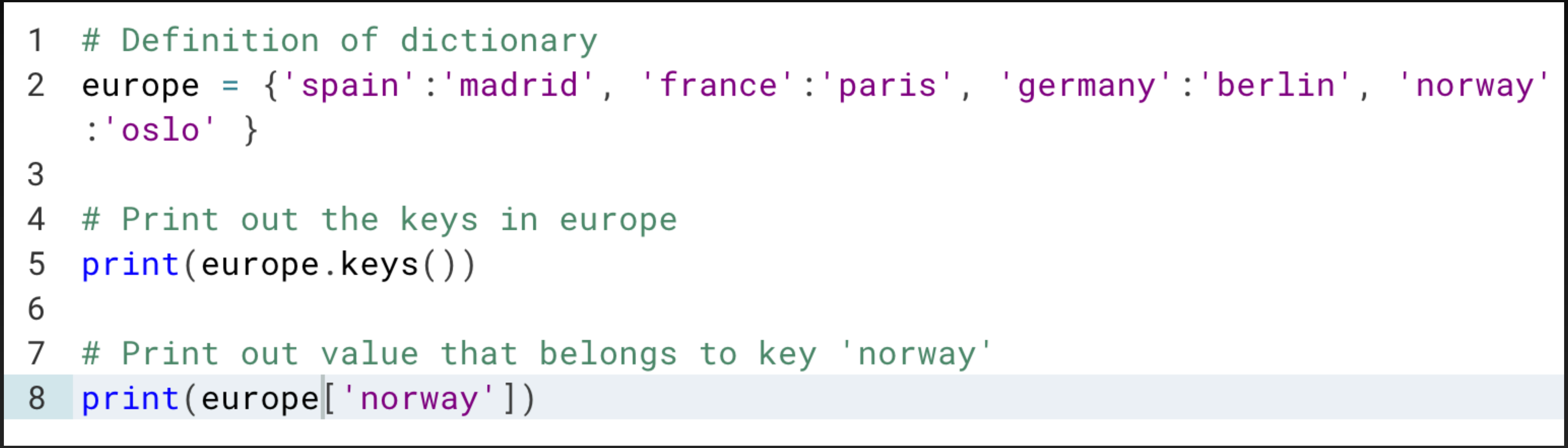


```
my_dict = {
    "key1":"value1",
    "key2":"value2",
}
```



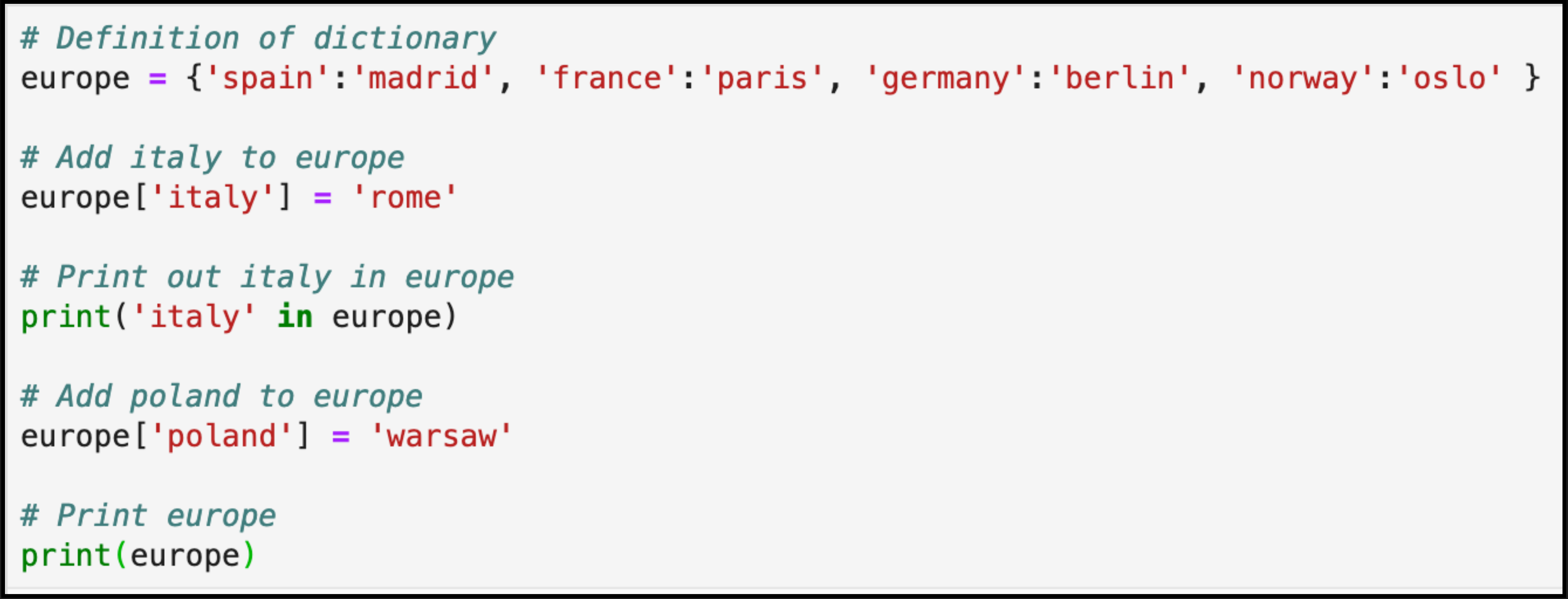
```
# Definition of countries and capital
countries = ['spain', 'france', 'germany', 'norway']
capitals = ['madrid', 'paris', 'berlin', 'oslo']

# From string in countries and capitals, create dictionary europe
europe = { "spain" : "madrid", "france" : "paris", "germany" : "berlin", "norway" : "oslo" }

# Print europe
print(europe)
```

取用 Dictionary I

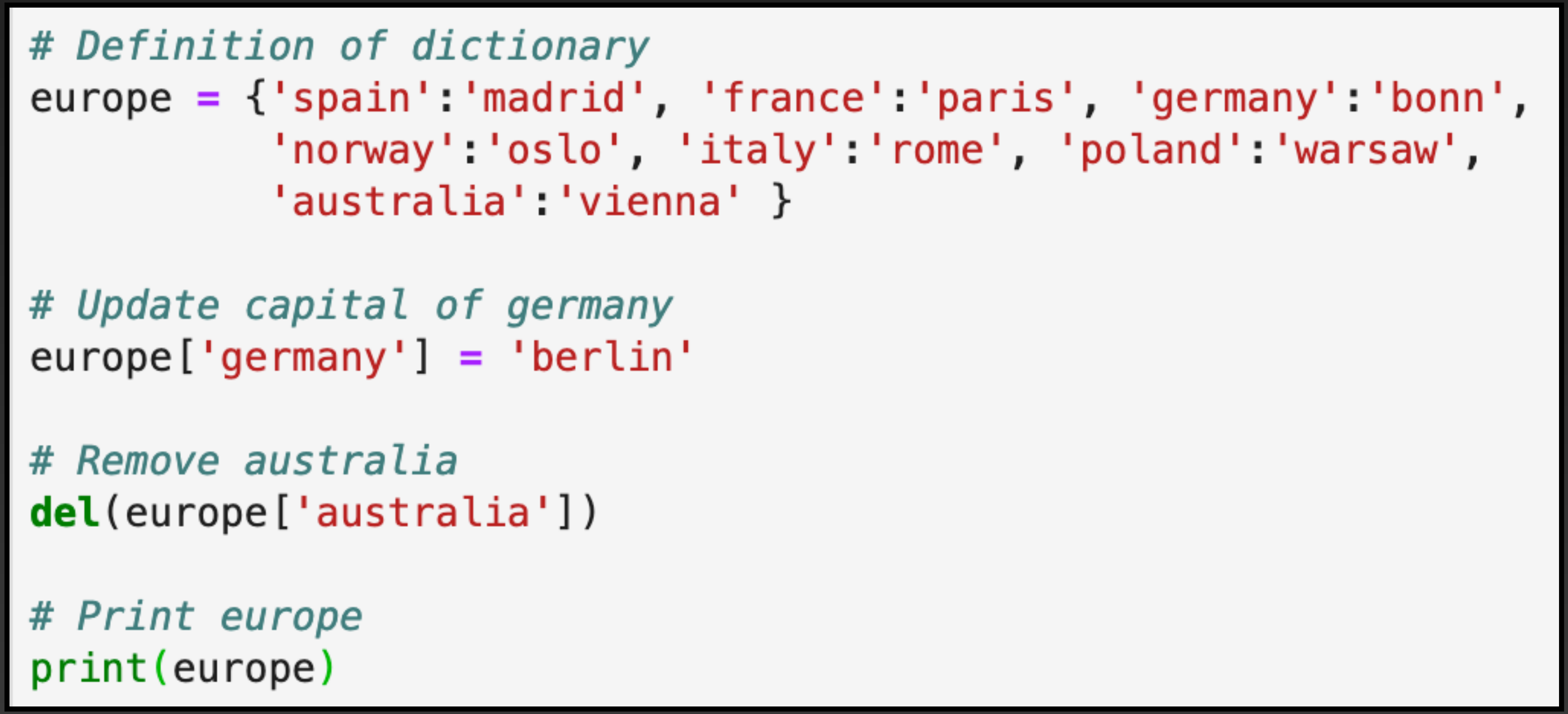


```python
# Definition of dictionary
europe = {'spain':'madrid', 'france':'paris', 'germany':'berlin', 'norway':'oslo' }

# Print out the keys in europe
print(europe.keys())

# Print out value that belongs to key 'norway'
print(europe['norway'])
```

存取 Dictionary II



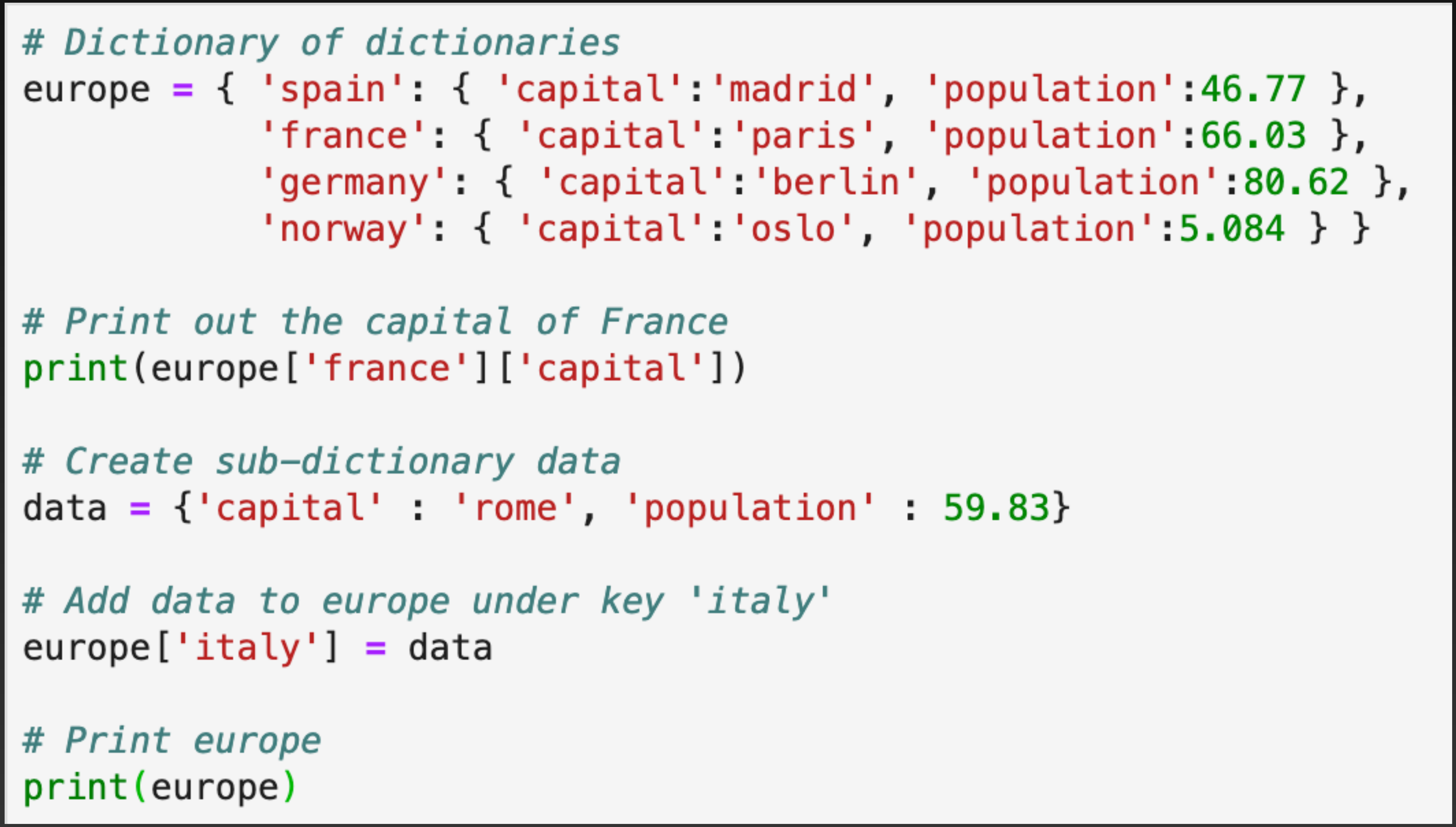
```python
# Definition of dictionary
europe = {'spain':'madrid', 'france':'paris', 'germany':'berlin', 'norway':'oslo' }

# Add italy to europe
europe['italy'] = 'rome'

# Print out italy in europe
print('italy' in europe)

# Add poland to europe
europe['poland'] = 'warsaw'

# Print europe
print(europe)
```



```python
# Definition of dictionary
europe = {'spain':'madrid', 'france':'paris', 'germany':'bonn',
          'norway':'oslo', 'italy':'rome', 'poland':'warsaw',
          'australia':'vienna' }

# Update capital of germany
europe['germany'] = 'berlin'

# Remove australia
del(europe['australia'])

# Print europe
print(europe)
```

Dictionary 的意義描述

- Dictionary 可以包含 key:value 配對，而其中的 value 又可以是 dictionary



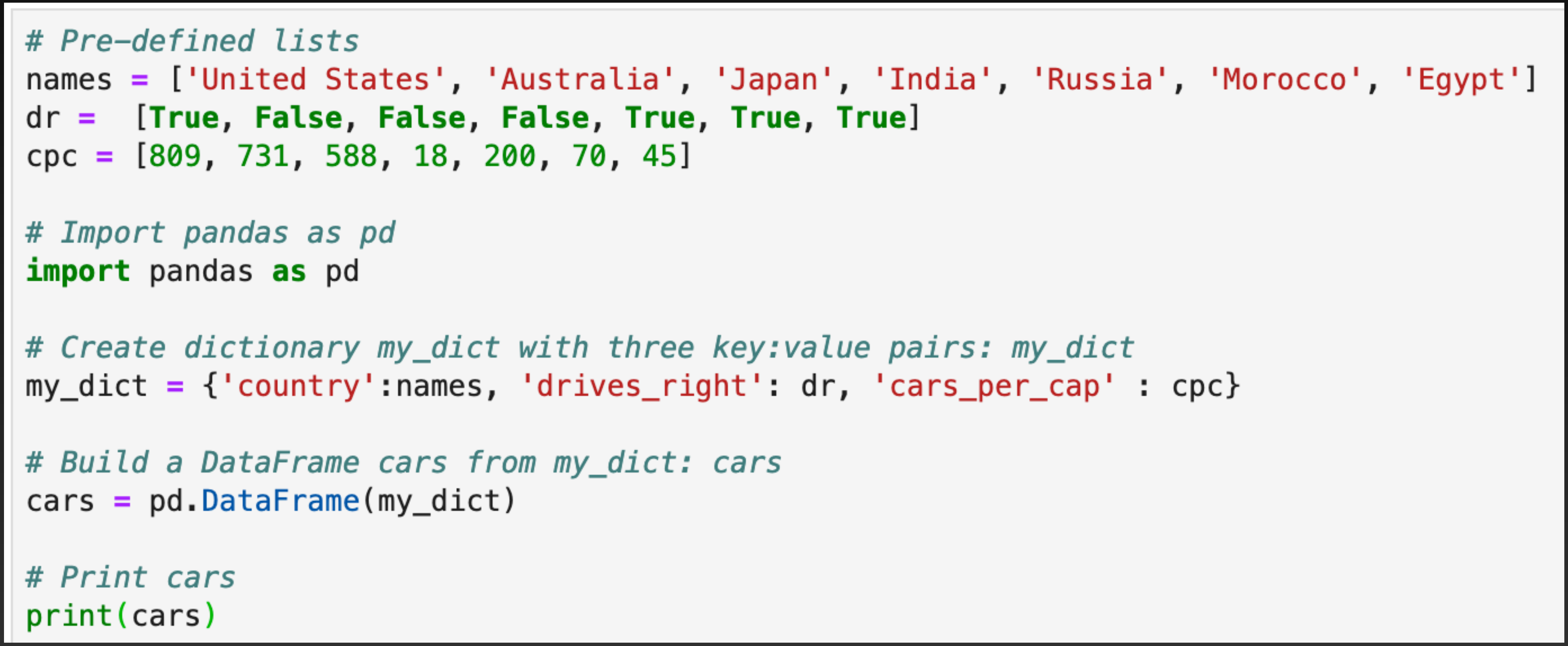
```
# Dictionary of dictionaries
europe = { 'spain': { 'capital':'madrid', 'population':46.77 },
           'france': { 'capital':'paris', 'population':66.03 },
           'germany': { 'capital':'berlin', 'population':80.62 },
           'norway': { 'capital':'oslo', 'population':5.084 } }

# Print out the capital of France
print(europe['france']['capital'])

# Create sub-dictionary data
data = {'capital' : 'rome', 'population' : 59.83}

# Add data to europe under key 'italy'
europe['italy'] = data

# Print europe
print(europe)
```

Dictionary to DataFrame (1)

- DataFrame 是 Pandas 最重要的資料結構，使用於儲存資料表，包含了欄與列的表頭。
- 可以直接由 dictionary 格式或是 csv 格式轉換成 DataFrame。
- 範例說明：來自不同國家的車輛，每比資料會對應到國家以及相關的欄位資訊，像是人均車輛數 (cpc)，右駕還是左駕 (dr) 等。
- 資料定義:
    - names：國家名稱
    - dr：使用真假值表示是否為右駕
    - cpc：在該國中每 1000 人擁有的車輛數
- 每個 dictionary key 是一個 column label 而每個 value 則是一個包含了欄位元素的 list。

Dictionary to DataFrame (1)

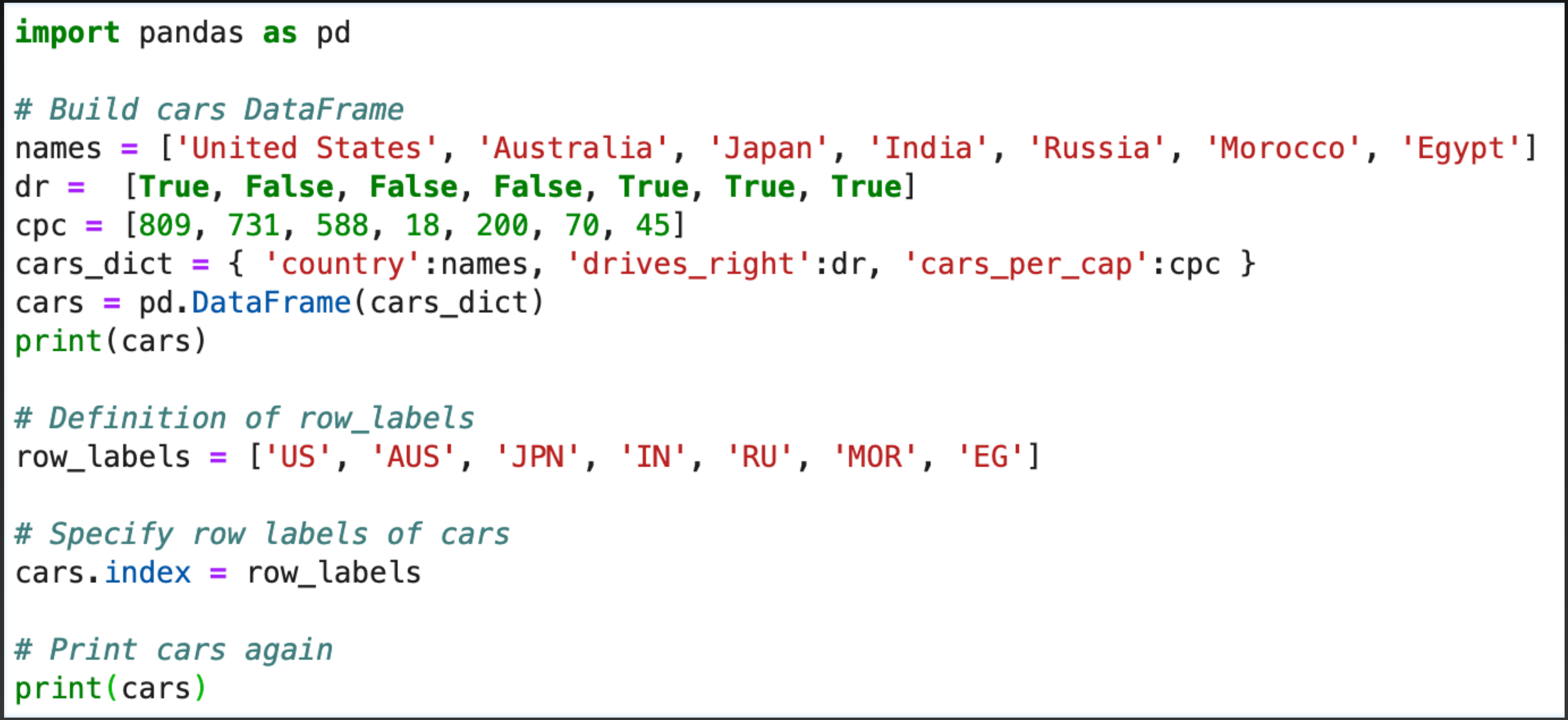
- 實作下列的 DataFrame 建構：



```python
# Pre-defined lists
names = ['United States', 'Australia', 'Japan', 'India', 'Russia', 'Morocco', 'Egypt']
dr =  [True, False, False, False, True, True, True]
cpc = [809, 731, 588, 18, 200, 70, 45]

# Import pandas as pd
import pandas as pd

# Create dictionary my_dict with three key:value pairs: my_dict
my_dict = {'country':names, 'drives_right': dr, 'cars_per_cap' : cpc}

# Build a DataFrame cars from my_dict: cars
cars = pd.DataFrame(my_dict)

# Print cars
print(cars)
```

Dictionary to DataFrame (2)

- 資料列標籤會自動設定成從 0 開始的整數。
- 下列的程式碼使用 cars.index 修改了資料列標籤。

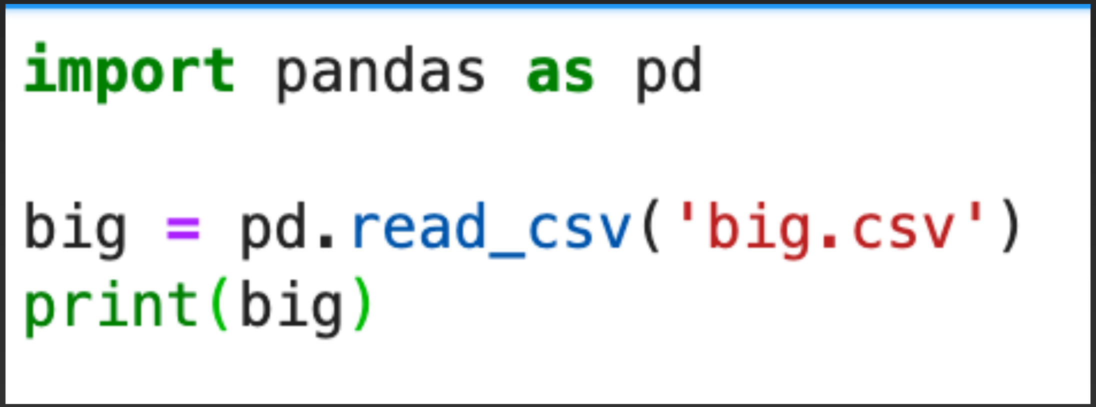


```
import pandas as pd

# Build cars DataFrame
names = ['United States', 'Australia', 'Japan', 'India', 'Russia', 'Morocco', 'Egypt']
dr =  [True, False, False, False, True, True, True]
cpc = [809, 731, 588, 18, 200, 70, 45]
cars_dict = { 'country':names, 'drives_right':dr, 'cars_per_cap':cpc }
cars = pd.DataFrame(cars_dict)
print(cars)

# Definition of row_labels
row_labels = ['US', 'AUS', 'JPN', 'IN', 'RU', 'MOR', 'EG']

# Specify row labels of cars
cars.index = row_labels

# Print cars again
print(cars)
```

CSV to DataFrame (1)

read_csv()

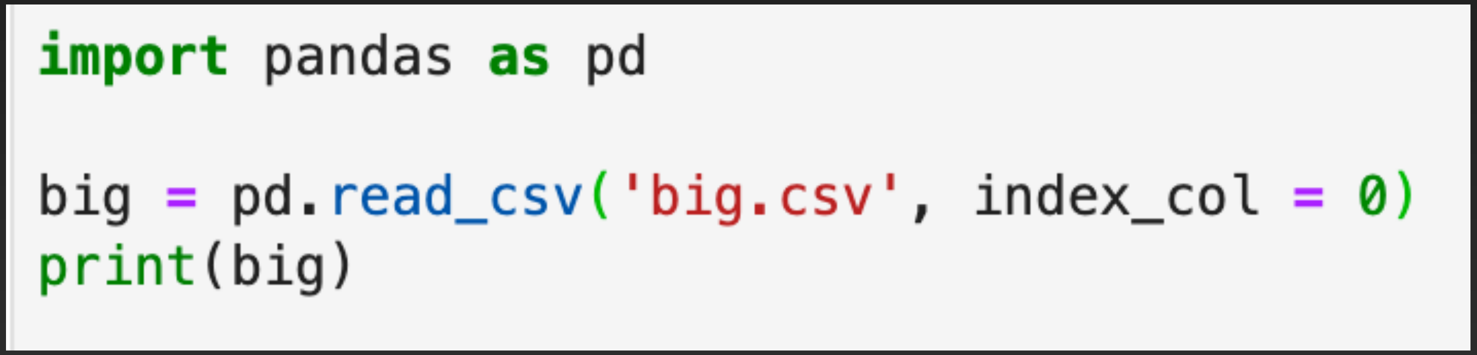
- 大部分實際情況，我D們會有實際資料，就可以使用 read_csv() 轉換成 DataFrame 然後進行後續處理。
- 現在我們取得一個較為複雜的巨量資料 big.csv
- 記得指定檔案名稱，有時需加上路徑，要使用 ‘big.csv’



```
import pandas as pd

big = pd.read_csv('big.csv')
print(big)
```

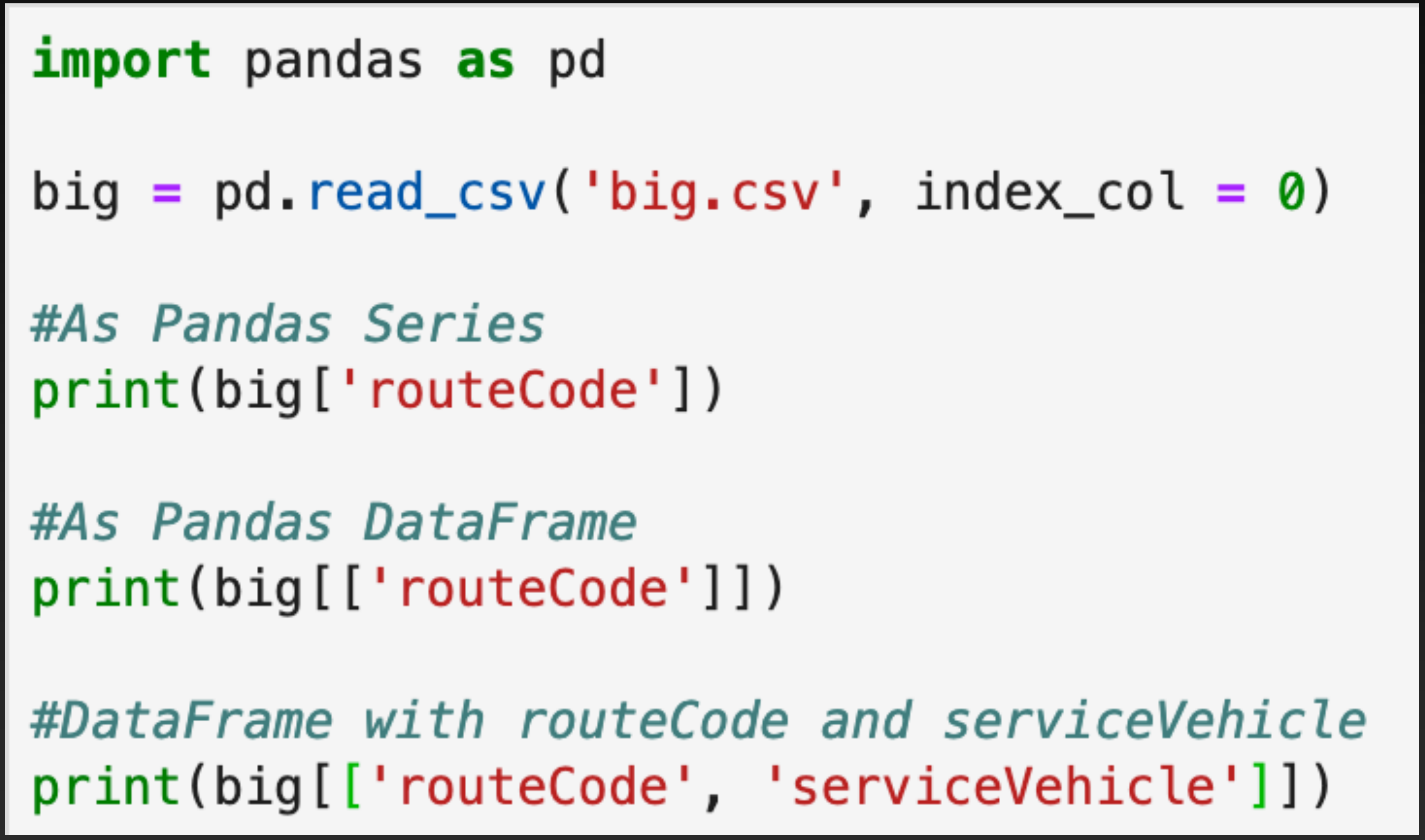
CSV to DataFrame (2)

- read_csv() call to import the CSV data
- index_col, an argument of read_csv(), that you can use to specify which column in the CSV file should be used as a row label



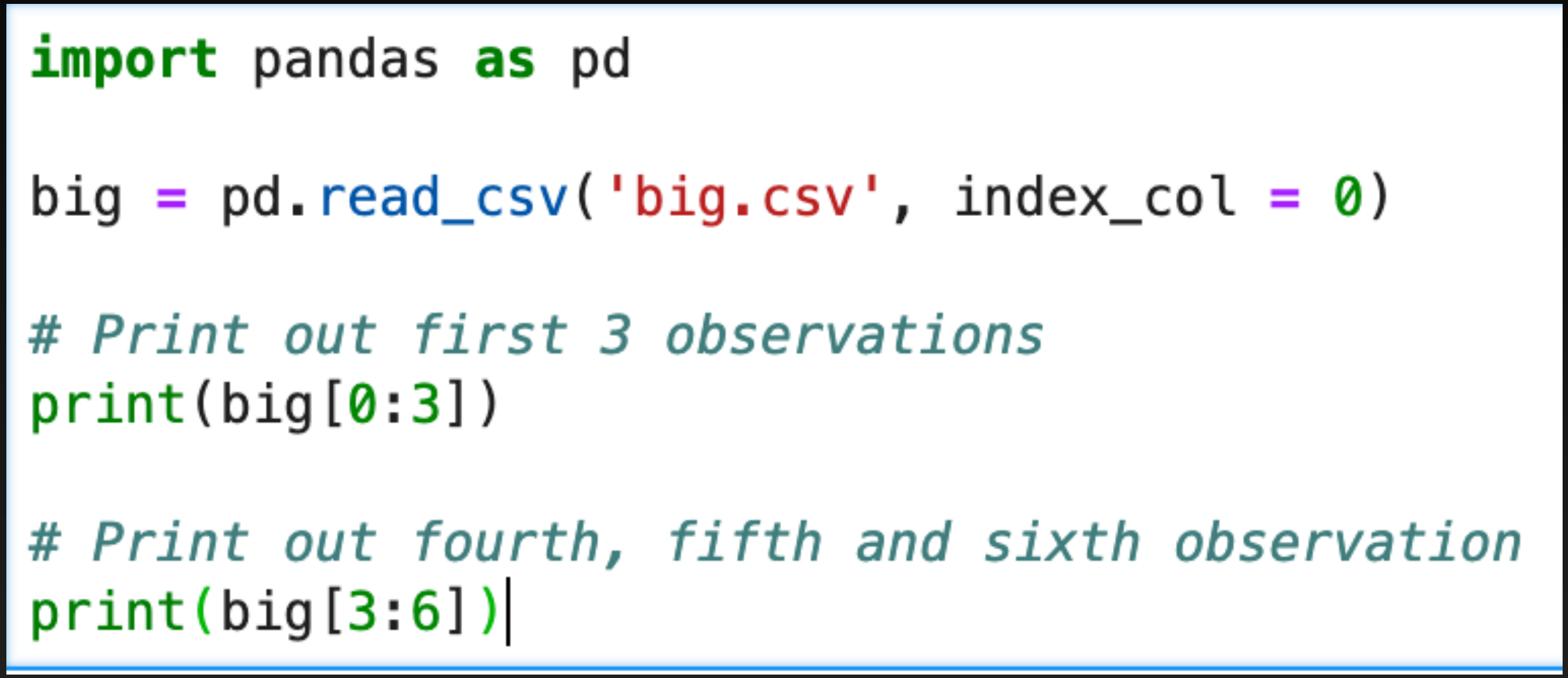
```
import pandas as pd

big = pd.read_csv('big.csv', index_col = 0)
print(big)
```

Square Brackets (1)

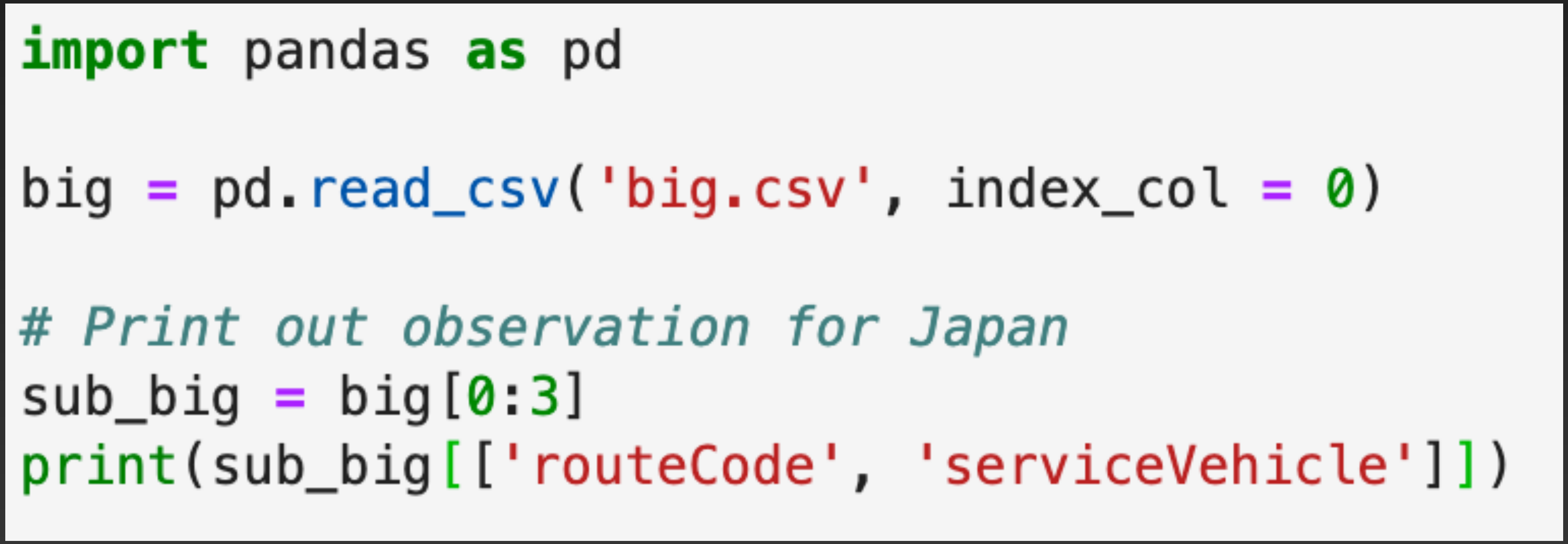


```python
import pandas as pd

big = pd.read_csv('big.csv', index_col = 0)

#As Pandas Series
print(big['routeCode'])

#As Pandas DataFrame
print(big[['routeCode']])

#DataFrame with routeCode and serviceVehicle
print(big[['routeCode', 'serviceVehicle']])
```

Square Brackets (2)



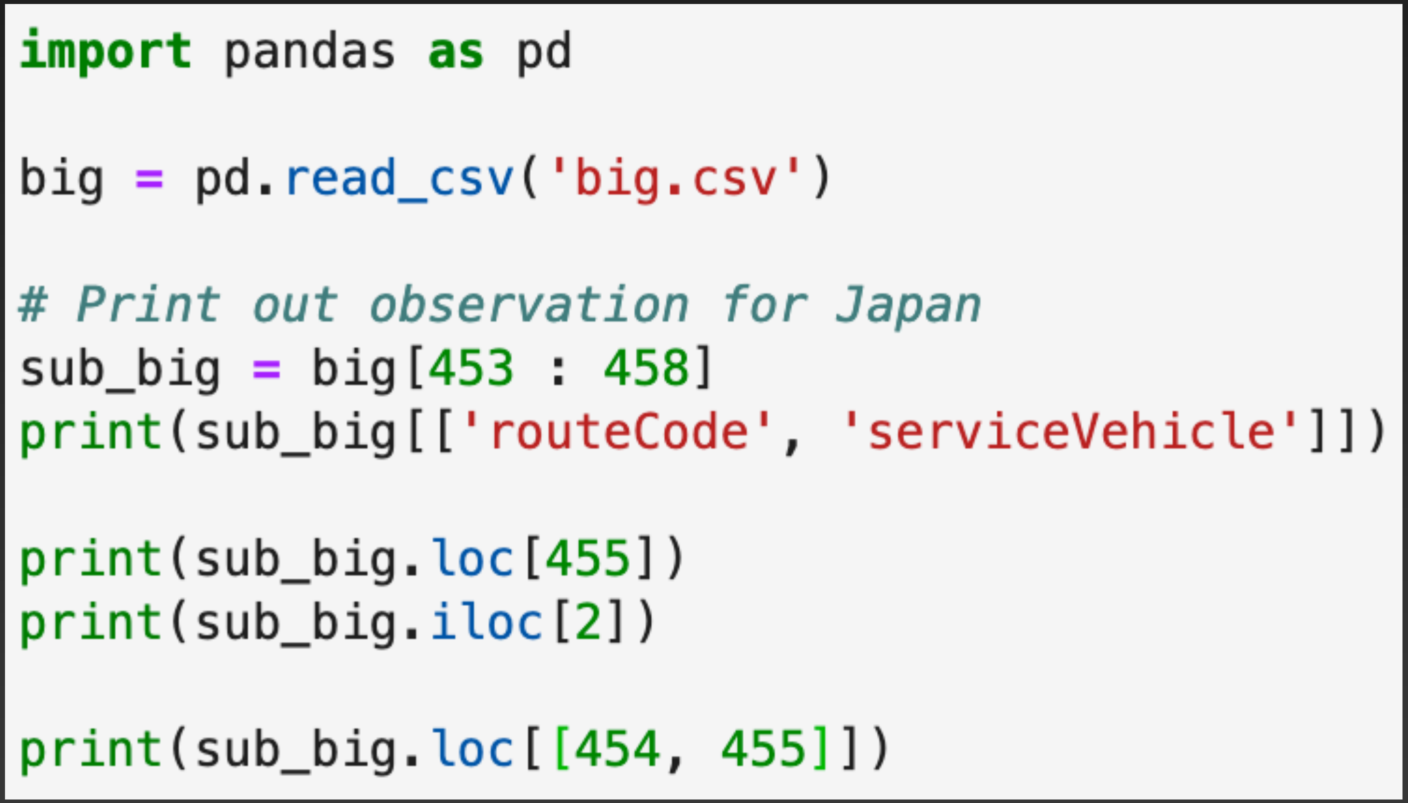
```python
import pandas as pd

big = pd.read_csv('big.csv', index_col = 0)

# Print out first 3 observations
print(big[0:3])

# Print out fourth, fifth and sixth observation
print(big[3:6])
```



```python
import pandas as pd

big = pd.read_csv('big.csv', index_col = 0)

# Print out observation for Japan
sub_big = big[0:3]
print(sub_big[['routeCode', 'serviceVehicle']])
```

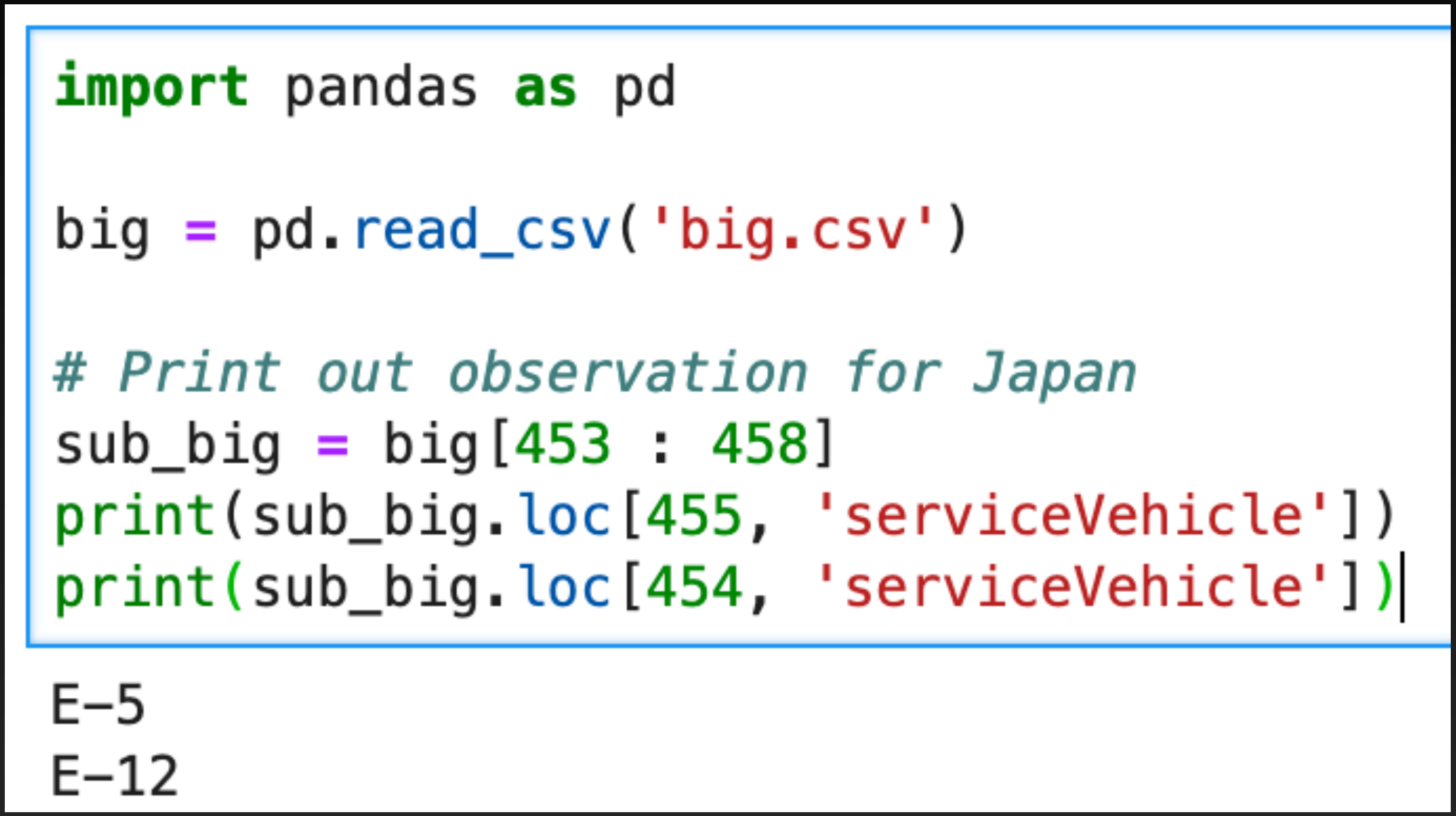
loc and iloc (1)

- With ***loc*** and ***iloc*** you can do practically any data selection operation on DataFrames.
- ***loc*** is label-based, which means that you have to specify rows and columns based on their row and column labels.
- ***iloc*** is integer index based.



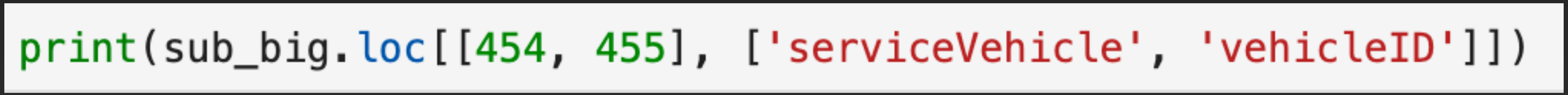
```
import pandas as pd

big = pd.read_csv('big.csv')

# Print out observation for Japan
sub_big = big[453 : 458]
print(sub_big[['routeCode', 'serviceVehicle']])

print(sub_big.loc[455])
print(sub_big.iloc[2])

print(sub_big.loc[[454, 455]])
```

loc and iloc (2)



```python
import pandas as pd

big = pd.read_csv('big.csv')

# Print out observation for Japan
sub_big = big[453 : 458]
print(sub_big.loc[455, 'serviceVehicle'])
print(sub_big.loc[454, 'serviceVehicle'])
```

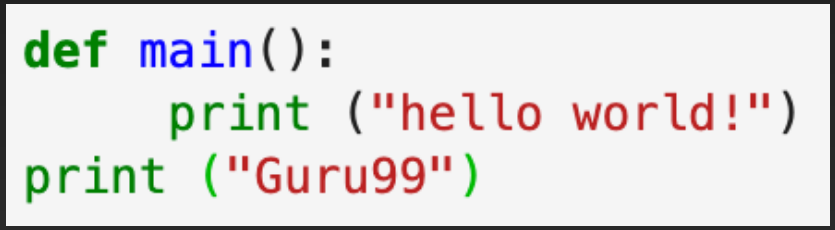
```
E-5
E-12
```



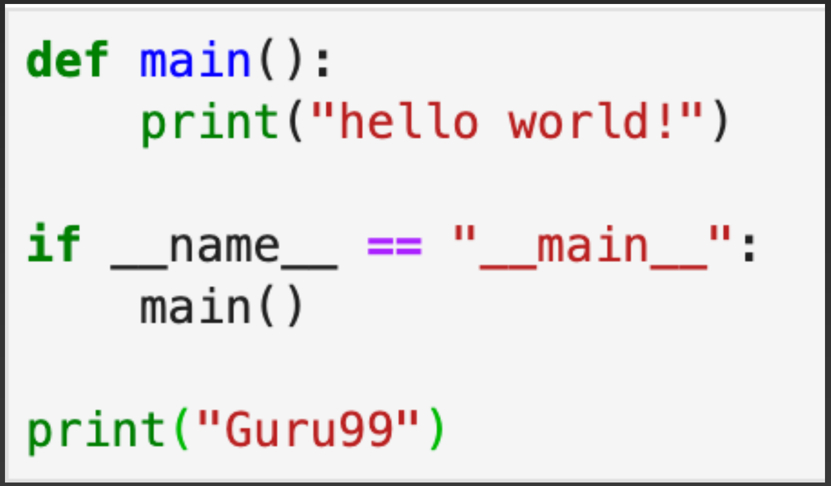
```python
print(sub_big.loc[[454, 455], ['serviceVehicle', 'vehicleID']])
```

Python Main Function

- PYTHON MAIN FUNCTION is a starting point of any program. When the program is run, the python interpreter runs the code sequentially. Main function is executed only when it is run as a Python program. It will not run the main function if it imported as a module.



```
def main():
     print ("hello world!")
print ("Guru99")
```



```
def main():
    print("hello world!")

if __name__ == "__main__":
    main()

print("Guru99")
```

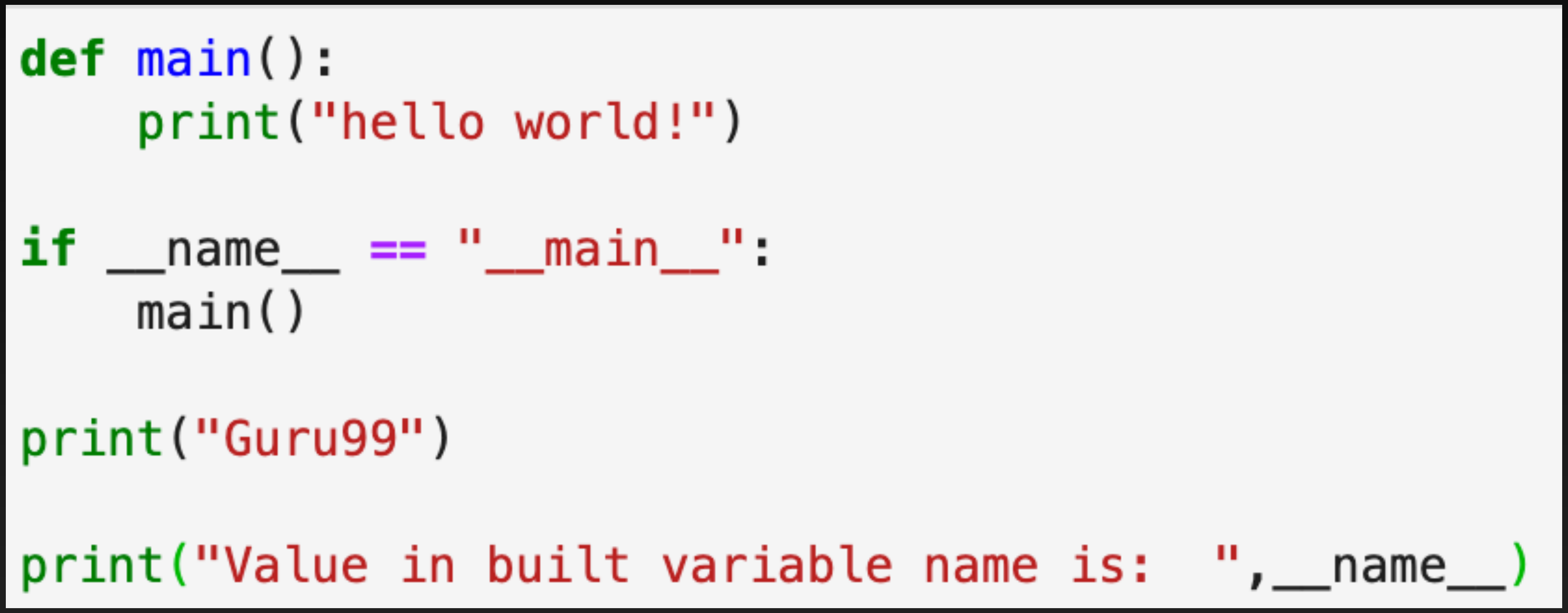
- When Python interpreter reads a source file, it will execute all the code found in it.

- When Python runs the "source file" as the main program, it sets the special variable (__name__) to have a value ("__main__").

- When you execute the main function, it will then read the "if" statement and checks whether __name__ does equal to __main__.

- In Python "if__name__== "__main__" allows you to run the Python files either as reusable modules or standalone programs.

The __name__ variable and Python Module



```
def main():
    print("hello world!")

if __name__ == "__main__":
    main()

print("Guru99")

print("Value in built variable name is:  ",__name__)
```

程式範例

```python
def __init__(self, speed=0):
    self.speed = speed
    self.odometer = 0
    self.time = 0

def say_state(self):
    print("I'm going {} kph!".format(self.speed))

def accelerate(self):
    self.speed += 5

def brake(self):
    self.speed -= 5

def step(self):
    self.odometer += self.speed
    self.time += 1

def average_speed(self):
    if self.time != 0:
        return self.odometer / self.time
    else:
        pass
```

```python
if __name__ == '__main__':

    my_car = Car()
    print("I'm a car!")
    while True:
        action = input("What should I do? [A]ccelerate, [B]rake, "
                 "show [O]dometer, or show average [S]peed?").upper()
        if action not in "ABOS" or len(action) != 1:
            print("I don't know how to do that")
            continue
        if action == 'A':
            my_car.accelerate()
        elif action == 'B':
            my_car.brake()
        elif action == 'O':
            print("The car has driven {} kilometers".format(my_car.odometer))
        elif action == 'S':
            print("The car's average speed was {} kph".format(my_car.average_speed()))
        my_car.step()
        my_car.say_state()
```