Buổi 2. Các thư viện trong Python

1 Thông tin chung

Mục tiêu buổi học

- Giới thiệu các thư viện phổ biến được dùng để phân tích dữ liệu trong Python.
- Hướng dẫn một số thao tác với thư viện Numpy, Pandas, Matplotlib, ...

Kiến thức và kỹ năng đạt được

- Biết cách sử dụng các hàm trong thư viện.
- Áp dụng cài đặt được các bài tập thực hành.

Công cụ thực hành

- Ngôn ngữ lập trình: Python
- Công cụ thực hành: Anaconda, colab

Thời gian thực hành: 3 tiết

2 Nội dung lý thuyết

Python là một ngôn ngữ lập trình mã nguồn mở và có rất nhiều thư viện mã nguồn mở. Trong lĩnh vực phân tích dữ liệu, nhiều thư viện đã được phát triển chia sẻ để mọi người có thể lập trình dễ dàng, nhanh chóng.

Một số thư viện phổ biến

- Numpy (https://numpy.org): hỗ trợ các tính toán khoa học và các thao tác với mảng, ma trận
- Pandas (https://pandas.pydata.org/): cung cấp chức năng đọc và thao tác trên dữ liệu từ nhiều nguồn khác nhau
- Matplotlib (https://matplotlib.org/): mô phỏng, vẽ biểu đồ dữ liệu

3 Nội dung thực hành

3.1 Numpy

3.1.1 Giới thiệu

Numpy là một thư viện hỗ trợ các tính toán trong khoa học

- Các mảng nhiều chiều

- Các hàm tính toán phức tạp
- Có thể tích hợp được ngôn ngữ C/C++, Pascal
- Dễ dàng thực hiện các phép toán trong đại số tuyến tính, biến đổi Fourier, sinh số ngẫu nhiên, ...

Cần phải thêm vào thư viện Numpy trước khi sử dụng.

```
import numpy as np
```

3.1.2 Numpy và các thao tác trên ma trận

Khai báo và khởi tạo ma trận Numpy cung cấp sẵn hàm để tạo mảng 2 chiều:

- numpy.array(): tạo mảng với giá trị trực tiếp từ các mảng 1 chiều hay các bộ

[56]:
```
import numpy as np

# Tạo mảng 2 chiều từ các mảng 1 chiều/danh sách
A = np.array([[1, 2, 3],
              [4, 5, 6],
              [7, 8, 9]])
print(A)
# Lấy số dòng, số cột của mảng 2 chiều
print(A.shape)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
(3, 3)
```

[57]:
```
import numpy as np
# Tạo mảng 2 chiều từ các bộ dữ liệu
A = np.array([(1, 2, 3),
              (4, 5, 6),
              (7, 8, 9)])
print(A)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

Ngoài ra, Numpy cung cấp một số hàm để sinh mảng tự động:

- arange(): tạo mảng từ một dãy số liên tiếp nhau
- linspace(): tạo mảng số thực trong một khoảng giá trị
- full(), ones(), zeros(), empty()
- eye(): tạo ma trận đơn vị
- diag(): tạo ma trận đường chéo
- max(), min(), sum()

[58]:
```
import numpy as np

# tạo mảng số nguyên từ dãy số liên tiếp nhau
A = np.arange(0, 100, step = 10)
print(A)
```

```
[ 0 10 20 30 40 50 60 70 80 90]
```

[59]:
```
import numpy as np

# tạo mảng n số thực từ dãy số liên tiếp nhau
A = np.linspace(0, 1, num = 5)
print(A)
```

```
[0.   0.25 0.5  0.75 1.  ]
```

[60]:
```
import numpy as np

# Tạo ma trận ngẫu nhiên kích thước gồm 3 dòng, 5 cột
A = np.random.rand(3, 5)
print(A)
```

```
[[0.37111464 0.09138428 0.54495336 0.08968621 0.96191124]
 [0.21032472 0.88281063 0.16785987 0.74982441 0.33826175]
 [0.64583049 0.99855621 0.5098262  0.0092711  0.23070573]]
```

[61]:
```
import numpy as np

# tạo ma trận đơn vị
A = np.eye(4)
print(A)
B = np.eye(4, k = 2)
print(B)
C = np.eye(4, k = -2)
print(C)

# tạo ma trận tất cả giá trị bằng 1
D = np.ones([3, 2])
print(D)

# tạo ma trận tất cả giá trị bằng 0
E = np.zeros([2, 3])
print(E)

# tạo ma trận tất cả giá trị bằng nhau
F = np.full((3, 3), 5)
print(F)
```

```
# tạo ma trận tất cả giá trị được sinh giá trị ngẫu nhiên
G = np.random.random((3, 3))
print(G)
```

```
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
[[0. 0. 1. 0.]
 [0. 0. 0. 1.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [1. 0. 0. 0.]
 [0. 1. 0. 0.]]
[[1. 1.]
 [1. 1.]
 [1. 1.]]
[[0. 0. 0.]
 [0. 0. 0.]]
[[5 5 5]
 [5 5 5]
 [5 5 5]]
[[0.78459191 0.865257   0.58731164]
 [0.5719993  0.85951331 0.66102554]
 [0.64263514 0.21681524 0.21447238]]
```

3.1.3 Truy xuất dữ liệu

Sử dụng toán tử [] để truy xuất phần tử:

- Tại vị trí dòng i, cột j
- Theo dòng
- Theo cột
- Theo lát cắt

[62]:
```
import numpy as np

A = np.array([[1, 2, 3, 4],
              [5, 6, 7, 8],
              [9, 10, 11, 12],
             [13, 14, 15, 16]])
print(A)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
```

```
 [13 14 15 16]]
```

[63]:
```
print(A[1, 3])
```

```
8
```

[64]:
```
# row
print(A[1, :])
print(A[1:3, :])
```

```
[5 6 7 8]
[[ 5  6  7  8]
 [ 9 10 11 12]]
```

[65]:
```
# column
print(A[:, 1])
print(A[:, 1:3])
```

```
[ 2  6 10 14]
[[ 2  3]
 [ 6  7]
 [10 11]
 [14 15]]
```

[66]:
```
# slicing
print(A[1, 1:3])
print(A[1:3, 1])
print(A[1:, 1:])
print(A[:3, :3])
```

```
[6 7]
[ 6 10]
[[ 6  7  8]
 [10 11 12]
 [14 15 16]]
[[ 1  2  3]
 [ 5  6  7]
 [ 9 10 11]]
```

[67]:
```
print(A)
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]
```

[68]:
```
print(np.sum(A, axis = 0))
```

```
[28 32 36 40]
```

[69]:
```
print(np.sum(A, axis = 1))
```

```
[10 26 42 58]
```

3.1.4 Các thao tác xử lý dữ liệu

Phép cộng/trừ ma trận

Cho A là ma trận kích thước $m \times n$ và B là ma trận kích thước $n \times p$, phép cộng hai ma trận được định nghĩa bởi công thức:

$$C_{ij} = A_{ij} + B_{ij} \tag{1}$$

[70]:
```
import numpy as np

A = np.array([[1, 2, 3],
              [4, 5, 6]])
print(A)

B = np.array([[0, 1, 2],
              [3, 4, 5]])
print(B)

#C = A + B
C = np.add(A, B)
# C = np.substract(A, B)
print(C)
```

```
[[1 2 3]
 [4 5 6]]
[[0 1 2]
 [3 4 5]]
[[ 1  3  5]
 [ 7  9 11]]
```

Phép nhân ma trận

Cho A là ma trận kích thước $m \times n$ và một số vô hướng b, phép nhân được định nghĩa bởi công thức:

$$C_{ij} = b \times A_{ij} \tag{2}$$

Cho A là ma trận kích thước $m \times n$ và B là ma trận kích thước $n \times p$, phép nhân hai ma trận được định nghĩa bởi công thức:

$$C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj} \tag{3}$$

[71]:
```
import numpy as np

A = np.array([[1, 2, 3], [4, 5, 6]])
print(A)
print(2 * A)
```

```
[[1 2 3]
 [4 5 6]]
[[ 2  4  6]
 [ 8 10 12]]
```

[72]:
```
import numpy as np

A = np.array([[1, 2, 3],
              [4, 5, 6]])
print(A)

B = np.array([[0, 1],
              [2, 3],
              [4, 5]])
print(B)

C = np.dot(A, B)
print(C)
```

```
[[1 2 3]
 [4 5 6]]
[[0 1]
 [2 3]
 [4 5]]
[[16 22]
 [34 49]]
```

Các phép biến đổi trên ma trận

- transpose: chuyển vị ma trận
- reshape: thay đổi kích thước ma trận

[73]:
```
import numpy as np

import numpy as np

A = np.array([
    [1, 2, 3],
    [4, 5, 6]])
print(A)

print(A.transpose()) #print(A.T)
```

```
[[1 2 3]
 [4 5 6]]
[[1 4]
 [2 5]
 [3 6]]
```

[74]:
```
print(A.reshape(1, 6))
```

```
[[1 2 3 4 5 6]]
```

Tính ma trận nghịch đảo Định thức của ma trận

Cho A là ma trận vuông cấp 2 như sau:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

- Định thức của ma trận A được tính bởi công thức

$$\det A = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc \tag{4}$$

[75]:
```
import numpy as np

A = np.array([[1, -1, 1],
              [0, -2, 1],
              [-2, -3, 0]])
print(A)
print(np.linalg.det(A))
```

```
[[ 1 -1  1]
 [ 0 -2  1]
 [-2 -3  0]]
1.0
```

Nghịch đảo ma trận

Bài toán đặt ra: Giải phương trình ma trận $AX = B$?

Các khái niệm

Cho A là ma trận vuông cấp n, A được gọi là ma trận khả nghịch nếu tồn tại ma trận nghịch đảo A^{-1} thỏa điều kiện

$$AA^{-1} = A^{-1}A = I \tag{5}$$

[76]:
```
import numpy as np

A = np.array([[1, -1, 1],
              [0, -2, 1],
```

```
              [-2, -3, 0]])
print(A)
print("det = ", np.linalg.det(A))

# Sử dụng hàm inv
B = np.linalg.inv(A)
print(B)

C = np.dot(A, B)
print(A)
```

```
[[ 1 -1  1]
 [ 0 -2  1]
 [-2 -3  0]]
det =  1.0
[[ 3. -3.  1.]
 [-2.  2. -1.]
 [-4.  5. -2.]]
[[ 1 -1  1]
 [ 0 -2  1]
 [-2 -3  0]]
```

Giải hệ phương trình tuyến tính

$$\begin{aligned} a_{0,0}x_0 + a_{0,1}x_2 + \cdots + a_{0,n}x_n &= b_0 \\ a_{1,0}x_0 + a_{1,1}x_2 + \cdots + a_{1,n}x_n &= b_1 \\ &\vdots \\ a_{m,0}x_0 + a_{m,1}x_2 + \cdots + a_{m,n}x_n &= b_m \end{aligned} \tag{6}$$

$$A = \begin{bmatrix} a_{0,0} & a_{0,1} & \cdots & a_{0,n} \\ a_{1,0} & a_{1,1} & \cdots & a_{1,n} \\ \vdots & & & \vdots \\ a_{m,0} & a_{m,1} & \cdots & a_{m,n} \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_m \end{bmatrix} \tag{7}$$

[77]:
```
import numpy as np

A = np.array([[6, 4, 1],
              [1, 8, -2],
              [3, 2, 0]])
b = np.array([7, 6, 8])

x = np.linalg.solve(A, b)

print(x)
```

```
[ 4. -2. -9.]
```

3.2 Pandas

3.2.1 Giới thiệu

Pandas là một thư viện mã nguồn mở, thường được sử dụng trong phân tích dữ liệu.

- Xử lý được nhiều loại dữ liệu khác nhau: chuỗi thời gian, bảng dữ liệu, ...
- Có thể import từ nhiều nguồn dữ liệu khác nhau

Kiểu dữ liệu

- Chuỗi thời gian (time series): dữ liệu cột
- Bảng dữ liệu (data frame): dữ liệu gồm nhiều dòng, nhiều cột

Series Khai báo và khởi tạo

[78]:
```
import pandas as pd

s1 = pd.Series(data = [1, 3, 5, 7, 9])
print(s1)

s2 = pd.Series(data = [1, 3, 5, 7, 9], index = ["a", "b", "c", "d", "e"])
print(s2)
```

```
0    1
1    3
2    5
3    7
4    9
dtype: int64
a    1
b    3
c    5
d    7
e    9
dtype: int64
```

Truy xuất phần tử

[79]:
```
print(s2[2])
print(s2['e'])
# first + 2
print(s2[:2])
# last - 2
print(s2[-2:])
```

```
5
9
a    1
b    3
dtype: int64
```

```
d    7
e    9
dtype: int64
```

[80]:
```
import pandas as pd

# Chỉ mục tăng tự động
df1 = pd.DataFrame({"col1" : [1 ,2, 3],
                    "col2" : [4, 5, 6],
                    "col3" : [7, 8, 9]})
print(df1)

# Chỉ mục xác định
df2 = pd.DataFrame({"col1" : [1 ,2, 3],
                    "col2" : [4, 5, 6],
                    "col3" : [7, 8, 9]},
                   index = [2, 4, 6])
print(df2)

# Chỉ mục dạng chuỗi
df3 = pd.DataFrame({"col1" : [1 ,2, 3],
                    "col2" : [4, 5, 6],
                    "col3" : [7, 8, 9]},
                   index = ["row1", "row2", "row3"])
print(df3)
```

```
   col1  col2  col3
0     1     4     7
1     2     5     8
2     3     6     9
   col1  col2  col3
2     1     4     7
4     2     5     8
6     3     6     9
      col1  col2  col3
row1     1     4     7
row2     2     5     8
row3     3     6     9
```

DataFrame Khai báo và khởi tạo

[81]:
```
import pandas as pd

cities = {"city": ["London", "Berlin", "Madrid", "Rome",
                   "Paris", "Vienna", "Bucharest", "Hamburg",
                   "Budapest", "Warsaw", "Barcelona",
                   "Munich", "Milan"],
```

```
        "population": [8615246, 3562166, 3165235, 2874038,
                       2273305, 1805681, 1803425, 1760433,
                       1754000, 1740119, 1602386, 1493900,
                       1350680],
        "area" : [1572, 891.85, 605.77, 1285,
                  105.4, 414.6, 228, 755,
                  525.2, 517, 101.9, 310.4,
                  181.8]
}

# Tạo DataFrame từ dữ liệu có sẵn
df = pd.DataFrame(cities,
                  columns = ["population", "area"],
                  index = cities["city"])
df
```

[81]:

	population	area
London	8615246	1572.00
Berlin	3562166	891.85
Madrid	3165235	605.77
Rome	2874038	1285.00
Paris	2273305	105.40
Vienna	1805681	414.60
Bucharest	1803425	228.00
Hamburg	1760433	755.00
Budapest	1754000	525.20
Warsaw	1740119	517.00
Barcelona	1602386	101.90
Munich	1493900	310.40
Milan	1350680	181.80

[82]:
```
df.head()
#df.head(10)
```

[82]:

	population	area
London	8615246	1572.00
Berlin	3562166	891.85
Madrid	3165235	605.77
Rome	2874038	1285.00
Paris	2273305	105.40

[83]:
```
df.tail()
```

[83]:

	population	area
Budapest	1754000	525.2
Warsaw	1740119	517.0
Barcelona	1602386	101.9

```
Munich        1493900  310.4
Milan         1350680  181.8
```

Cập nhật dữ liệu

[84]:
```
# Thêm thuộc tính date
df['date'] = '12/12/2010'
df
```

[84]:
```
           population     area        date
London        8615246  1572.00  12/12/2010
Berlin        3562166   891.85  12/12/2010
Madrid        3165235   605.77  12/12/2010
Rome          2874038  1285.00  12/12/2010
Paris         2273305   105.40  12/12/2010
Vienna        1805681   414.60  12/12/2010
Bucharest     1803425   228.00  12/12/2010
Hamburg       1760433   755.00  12/12/2010
Budapest      1754000   525.20  12/12/2010
Warsaw        1740119   517.00  12/12/2010
Barcelona     1602386   101.90  12/12/2010
Munich        1493900   310.40  12/12/2010
Milan         1350680   181.80  12/12/2010
```

[85]:
```
# Xóa thuộc tính date
df.drop(['date'], axis = 1)
```

[85]:
```
           population     area
London        8615246  1572.00
Berlin        3562166   891.85
Madrid        3165235   605.77
Rome          2874038  1285.00
Paris         2273305   105.40
Vienna        1805681   414.60
Bucharest     1803425   228.00
Hamburg       1760433   755.00
Budapest      1754000   525.20
Warsaw        1740119   517.00
Barcelona     1602386   101.90
Munich        1493900   310.40
Milan         1350680   181.80
```

Truy xuất dữ liệu

-
-
- [i:, :j]:

- loc: lấy theo index
- iloc: lấy theo vị trí/thứ tự

[86]:
```
# Tên cột/thuộc tính
print(df["area"])
print(df.area)
# Danh sách chứa nhiều cột
print(df[["area", "population"]])
# tên cột và chỉ mục
print(df["area"]["London"])
print(df.area["London"])
```

```
London       1572.00
Berlin        891.85
Madrid        605.77
Rome         1285.00
Paris         105.40
Vienna        414.60
Bucharest     228.00
Hamburg       755.00
Budapest      525.20
Warsaw        517.00
Barcelona     101.90
Munich        310.40
Milan         181.80
Name: area, dtype: float64
London       1572.00
Berlin        891.85
Madrid        605.77
Rome         1285.00
Paris         105.40
Vienna        414.60
Bucharest     228.00
Hamburg       755.00
Budapest      525.20
Warsaw        517.00
Barcelona     101.90
Munich        310.40
Milan         181.80
Name: area, dtype: float64
              area  population
London     1572.00     8615246
Berlin      891.85     3562166
Madrid      605.77     3165235
Rome       1285.00     2874038
Paris       105.40     2273305
Vienna      414.60     1805681
```

```
Bucharest   228.00     1803425
Hamburg     755.00     1760433
Budapest    525.20     1754000
Warsaw      517.00     1740119
Barcelona   101.90     1602386
Munich      310.40     1493900
Milan       181.80     1350680
1572.0
1572.0
```

[87]:
```
# Tên cột/thuộc tính
print(df.loc["London"])
# Lấy dữ liệu từ dòng 5 -> cuối
print(df.iloc[5:])
# Lấy dữ liệu dòng: đầu -> 3
print(df.iloc[:3])
# Lấy dữ liệu từ dòng 1 -> 4
print(df.iloc[1:5])
```

```
population       8615246
area                1572
date           12/12/2010
Name: London, dtype: object
           population   area        date
Vienna        1805681  414.6  12/12/2010
Bucharest     1803425  228.0  12/12/2010
Hamburg       1760433  755.0  12/12/2010
Budapest      1754000  525.2  12/12/2010
Warsaw        1740119  517.0  12/12/2010
Barcelona     1602386  101.9  12/12/2010
Munich        1493900  310.4  12/12/2010
Milan         1350680  181.8  12/12/2010
        population     area        date
London     8615246  1572.00  12/12/2010
Berlin     3562166   891.85  12/12/2010
Madrid     3165235   605.77  12/12/2010
        population     area        date
Berlin     3562166   891.85  12/12/2010
Madrid     3165235   605.77  12/12/2010
Rome       2874038  1285.00  12/12/2010
Paris      2273305   105.40  12/12/2010
```

Các thao tác xử lý dữ liệu Lấy thông tin từ dữ liệu

[88]:
```
# Lấy số dòng, số cột của bảng dữ liệu
rows , cols = df.shape
print(rows)
```

```
print(cols)
```

```
13
3
```

[89]:
```
print(df.columns)
print(df.columns.values)
```

```
Index(['population', 'area', 'date'], dtype='object')
['population' 'area' 'date']
```

[90]:
```
print(df.index)
print(df.index.values)
```

```
Index(['London', 'Berlin', 'Madrid', 'Rome', 'Paris', 'Vienna', 'Bucharest',
       'Hamburg', 'Budapest', 'Warsaw', 'Barcelona', 'Munich', 'Milan'],
      dtype='object')
['London' 'Berlin' 'Madrid' 'Rome' 'Paris' 'Vienna' 'Bucharest' 'Hamburg'
 'Budapest' 'Warsaw' 'Barcelona' 'Munich' 'Milan']
```

[91]:
```
df.dtypes
```

[91]:
```
population      int64
area          float64
date           object
dtype: object
```

[92]:
```
# Lấy thông tin chung của dữ liệu
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 13 entries, London to Milan
Data columns (total 3 columns):
population    13 non-null int64
area          13 non-null float64
date          13 non-null object
dtypes: float64(1), int64(1), object(1)
memory usage: 1.0+ KB
```

[93]:
```
# lấy thông tin chung của dữ liệu
df.rank()
```

[93]:

	population	area	date
London	13.0	13.0	7.0
Berlin	12.0	11.0	7.0
Madrid	11.0	9.0	7.0
Rome	10.0	12.0	7.0
Paris	9.0	2.0	7.0

```
Vienna            8.0   6.0   7.0
Bucharest         7.0   4.0   7.0
Hamburg           6.0  10.0   7.0
Budapest          5.0   8.0   7.0
Warsaw            4.0   7.0   7.0
Barcelona         3.0   1.0   7.0
Munich            2.0   5.0   7.0
Milan             1.0   3.0   7.0
```

Lọc dữ liệu

[94]:
```
df.area > 500
```

[94]:
```
London         True
Berlin         True
Madrid         True
Rome           True
Paris         False
Vienna        False
Bucharest     False
Hamburg        True
Budapest       True
Warsaw         True
Barcelona     False
Munich        False
Milan         False
Name: area, dtype: bool
```

[95]:
```
# Lọc dữ liệu: dân số > 3000000 và diện tích > 500
df[(df.population > 3000000) & (df.area > 500)]
```

[95]:
```
          population     area        date
London       8615246  1572.00  12/12/2010
Berlin       3562166   891.85  12/12/2010
Madrid       3165235   605.77  12/12/2010
```

[96]:
```
df.sort_values(by = 'area')
```

[96]:
```
            population     area        date
Barcelona      1602386   101.90  12/12/2010
Paris          2273305   105.40  12/12/2010
Milan          1350680   181.80  12/12/2010
Bucharest      1803425   228.00  12/12/2010
Munich         1493900   310.40  12/12/2010
Vienna         1805681   414.60  12/12/2010
Warsaw         1740119   517.00  12/12/2010
Budapest       1754000   525.20  12/12/2010
```

```
Madrid        3165235   605.77  12/12/2010
Hamburg       1760433   755.00  12/12/2010
Berlin        3562166   891.85  12/12/2010
Rome          2874038  1285.00  12/12/2010
London        8615246  1572.00  12/12/2010
```

Thống kê cơ bản

- sum, max, min
- mean, std
- describle

[97]:
```
#df["area"].max()
df.area.max()
```

[97]:
```
1572.0
```

[98]:
```
df.area.mean()
```

[98]:
```
576.4553846153846
```

[99]:
```
df.area.min()
```

[99]:
```
101.9
```

[100]:
```
df.describe()
```

[100]:
```
         population         area
count  1.300000e+01    13.000000
mean   2.600047e+06   576.455385
std    1.932512e+06   452.103468
min    1.350680e+06   101.900000
25%    1.740119e+06   228.000000
50%    1.803425e+06   517.000000
75%    2.874038e+06   755.000000
max    8.615246e+06  1572.000000
```

Phát hiện dữ liệu thiếu, lỗi

- notnull
- isnull
- dropna
- fillna

[101]:
```
import numpy as np
import pandas as pd

test_df = pd.DataFrame({'A':[1, 2, np.nan],
                        'B':[4, np.nan, np.nan],
```

```
                        'C':[7, 8, 9]})

print(test_df)
```

```
     A    B  C
0  1.0  4.0  7
1  2.0  NaN  8
2  NaN  NaN  9
```

[102]:
```
# Kiểm tra tất cả dữ liệu có giá trị NaN hay ko?
test_df.isnull()
```

[102]:
```
       A      B      C
0  False  False  False
1  False   True  False
2   True   True  False
```

[103]:
```
# Xóa tất cả dòng có giá trị NaN
test2df = test_df.dropna()
test2df
```

[103]:
```
     A    B  C
0  1.0  4.0  7
```

[104]:
```
# Điền tất cả giá trị NaN bằng giá trị 0
test2df = test_df.fillna(1)
test2df
```

[104]:
```
     A    B  C
0  1.0  4.0  7
1  2.0  1.0  8
2  1.0  1.0  9
```

Đọc và ghi dữ liệu với DataFrame

- read_csv(filename)
- to_csv(filename)

[105]:
```
df.to_csv("population.csv")
```

[106]:
```
df = pd.read_csv("population.csv")
df
```

[106]:
```
   Unnamed: 0  population     area        date
0      London     8615246  1572.00  12/12/2010
1      Berlin     3562166   891.85  12/12/2010
2      Madrid     3165235   605.77  12/12/2010
3        Rome     2874038  1285.00  12/12/2010
```

```
4        Paris     2273305   105.40  12/12/2010
5       Vienna     1805681   414.60  12/12/2010
6    Bucharest     1803425   228.00  12/12/2010
7      Hamburg     1760433   755.00  12/12/2010
8     Budapest     1754000   525.20  12/12/2010
9       Warsaw     1740119   517.00  12/12/2010
10   Barcelona     1602386   101.90  12/12/2010
11      Munich     1493900   310.40  12/12/2010
12       Milan     1350680   181.80  12/12/2010
```

3.3 Mathplotlib

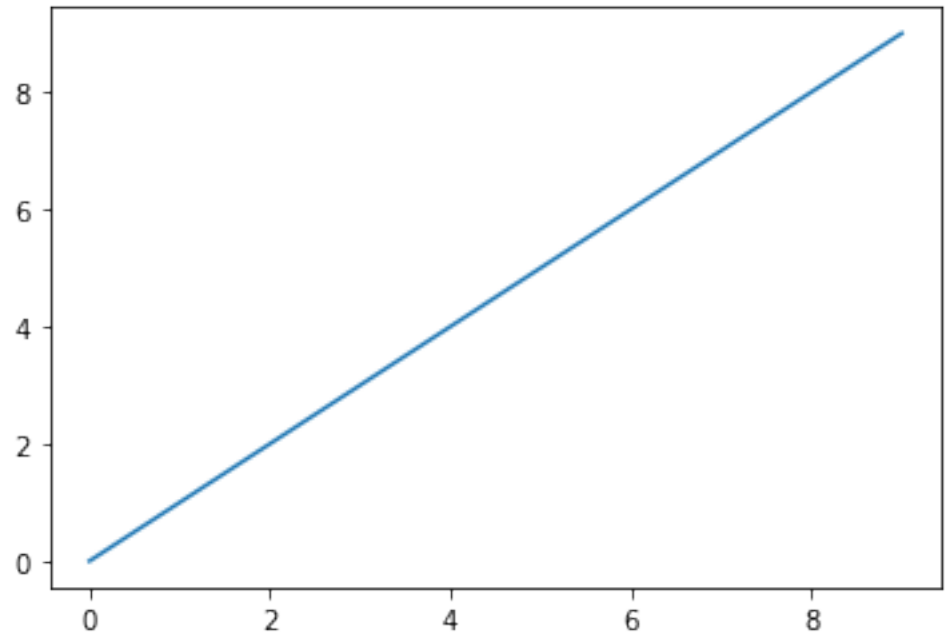
3.3.1 Giới thiệu

- Là thư viện hỗ trợ vẽ các biểu đồ trong Python.
- Xử lý tốt đối với các dữ liệu Numpy.

3.3.2 Các thao tác vẽ biểu đồ

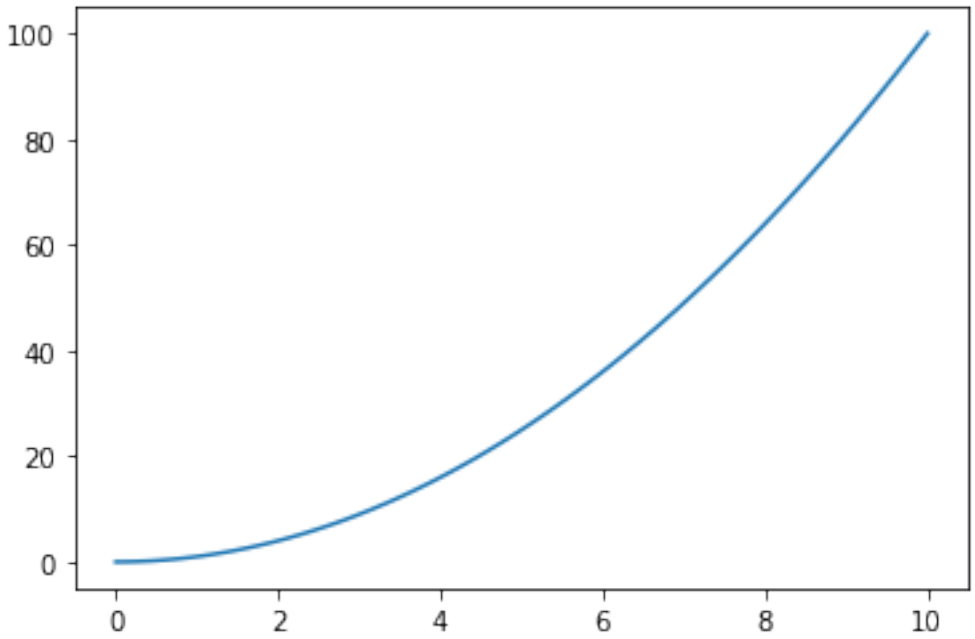
[107]:
```
import numpy as np
import matplotlib.pyplot as plt

data = np.arange(10)

plt.plot(data)
```

[107]:
```
[<matplotlib.lines.Line2D at 0x275c1019f48>]
```



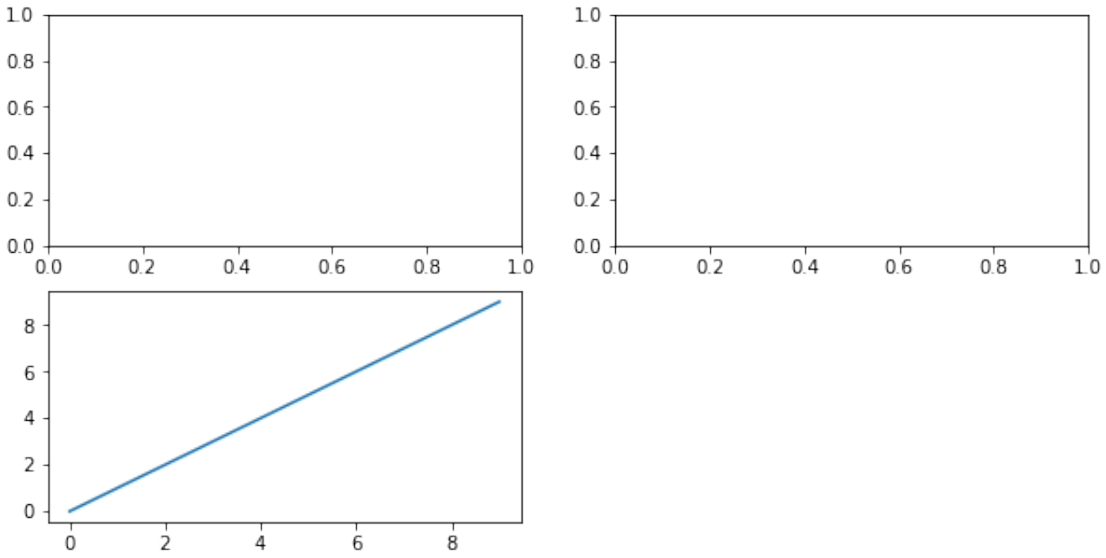
[108]:
```
# y = x^2
x = np.linspace(0, 10, 100)
y = np.power(x, 2)

plt.plot(x, y)
```

[108]:
```
[<matplotlib.lines.Line2D at 0x275c2020a88>]
```

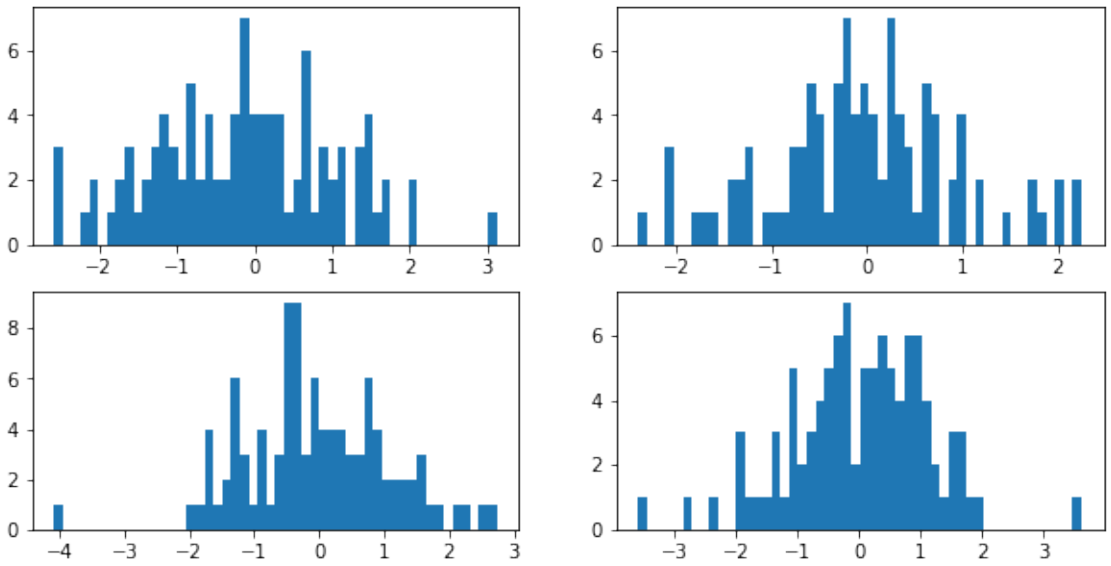


- Các biểu đồ có thể vẽ trên cùng một Figure như sau:

[109]:
```
fig = plt.figure(figsize = (10, 5))

ax1 =fig.add_subplot(2, 2, 1)
ax2 = fig.add_subplot(2, 2, 2)
ax3 = fig.add_subplot(2, 2, 3)

# Vẽ biểu đồ thứ 3
data = np.arange(10)
plt.plot(data)
plt.show()
```



[110]:
```
#
fig, ax = plt.subplots(figsize = (10, 5), nrows = 2, ncols = 2)
for i in range(2):
    for j in range(2):
        ax[i, j].hist(np.random.normal(size = 100), bins = 50)

plt.show()
```



4 Bài tập

Cho tập dữ liệu khách hàng Mall_Customer.csv, hãy thực hiện các thao tác thống kê cơ bản đối với dữ liệu này.