

Ngôn Ngữ Lập Trình Python

Các thư viện phổ biến

Trịnh Tấn Đạt

Đại Học Sài Gòn

trinhtandat@sgu.edu.vn

<http://sites.google.com/site/ttdat88>

Nội Dung

- Numpy
- Matplotlib

Nội Dung

- Mảng và Ma Trận trong Python
- Giới thiệu về NumPy
- Các phép toán trên mảng
- Một số thao tác thông dụng

Mảng và Ma Trận trong Python

- **Ma trận** là cấu trúc dữ liệu hai chiều, trong đó các số được sắp xếp thành các hàng và cột.
- **Nested list** thường được dùng để trình bày ma trận trong Python. Biểu diễn như sau

```
A = [[1, 4, 5],  
      [-5, 8, 9]]
```

Mảng và Ma Trận trong Python

```
A = [[1, 4, 5, 12],  
      [-5, 8, 9, 0],  
      [-6, 7, 11, 19]]  
print("A =", A)  
print("A[1] =", A[1]) # [-5, 8, 9, 0]  
print("A[1][2] =", A[1][2]) # 9  
print("A[0][-1] =", A[0][-1]) # 12  
column = []  
for row in A:  
    column.append(row[2])  
print("Cột thứ 3 =", column)
```

Giới thiệu về NumPy

- Ngôn ngữ python có hệ thống các gói rất phong phú, hỗ trợ nhiều lĩnh vực khác nhau, từ xử lý ảnh/video/audio, xử lý văn bản, thống kê, máy học, đồ họa, web,...
- Sử dụng pip để tải các gói mới về từ internet
 - Ví dụ: open Command Prompt (cmd.exe)

`pip install numpy`

`pip install pandas`

`pip install matplotlib`

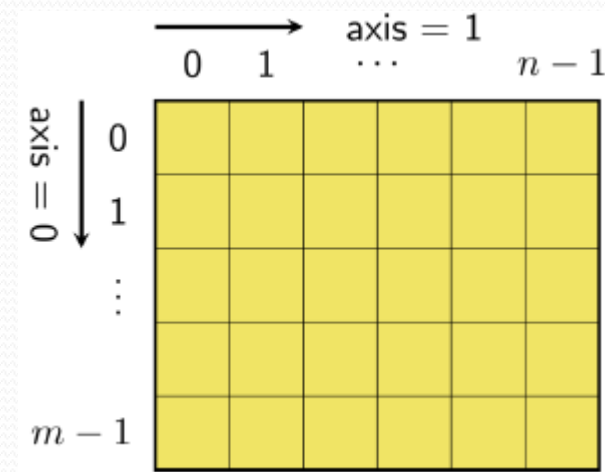
.....

Giới thiệu về NumPy

- NumPy (Numerical Python): là gói chuyên về xử lý dữ liệu số (nhiều chiều); gói cũng chứa các hàm đại số tuyến tính cơ bản, biến đổi fourier, sinh số ngẫu nhiên nâng cao,...
- NumPy là thư viện bổ sung của python, do không có sẵn, ta phải cài đặt:
open Command Prompt (cmd.exe)
pip install numpy
- Anaconda đã tích hợp sẵn numpy khi cài đặt.
- Cách đơn giản nhất để kiểm tra xem hệ thống đã cài numpy hay không là thử import gói xem có bị báo lỗi hay không: **import numpy as np**

Giới thiệu về NumPy

- Đối tượng chính của NumPy là các mảng đa chiều đồng nhất (homogeneous multidimension array)
 - Kiểu dữ liệu phần tử con trong mảng phải giống nhau
 - Mảng có thể một chiều hoặc nhiều chiều
 - Các chiều (axis) được đánh thứ tự từ 0 trở đi
 - Kiểu **ndarray** là lớp chính xử lý dữ liệu mảng nhiều chiều
 - Rất nhiều hàm và phương thức xử lý ma trận



Tạo mảng và truy cập

```
import numpy as np
a = np.array([1, 2, 3]) # tạo mảng 1 chiều
print(type(a)) # in "<class 'numpy.ndarray'>"
print(a.shape) # in "(3,)"
print(a[0], a[1], a[2]) # in "1 2 3"
a[0] = 5
print(a) # in "[5, 2, 3]"
b = np.array([[1, 2, 3],[4, 5, 6]]) # tạo mảng 2 chiều
print(b.shape) # in "(2, 3)"
print(b[0, 0], b[0, 1], b[1, 0]) # in "1 2 4"
print(np.diag([1, 3, 4])) #
```

```
[[1 0 0]
 [0 3 0]
 [0 0 4]]
```

Tạo mảng và truy cập

```
import numpy as np
x = np.arange(3) # mảng [0 1 2]
print(x)
a = np.zeros((2, 2)) # mảng 2x2 toàn số 0
print(a)
b = np.ones((1, 2)) # mảng 1x2 toàn số 1
print(b)
c = np.full((3, 2, 2), 9) # mảng 3x2x2 toàn số 9
print(c)
d = np.eye(2) # ma trận đơn vị 2x2
print(d)
e = np.random.rand(3, 2) # mảng 3x2 ngẫu nhiên [0,1)
print(e)
# mảng 2x3 điền các số từ 1 đến 6, kiểu số nguyên 32 bit
x = np.array([[1, 2, 3], [4, 5, 6]], np.int32)
print(x.ndim, x.size)
print(x.shape) # in "(2, 3)"
print(x.dtype) # in "dtype('int32')"
```

```
[0 1 2]
[[0. 0.]
 [0. 0.]]
[[1. 1.]]
[[[9 9]
  [9 9]]

 [[9 9]
  [9 9]]

 [[9 9]
  [9 9]]]
[[1. 0.]
 [0. 1.]]
[[0.7332557  0.64747164]
 [0.77964332 0.27683164]
 [0.55152677 0.84255606]]
2 6
(2, 3)
int32
```

Tạo mảng và truy cập

```
import numpy as np
A = np.arange(4)
print('A =', A)
B = np.arange(12).reshape(2, 6)
print('B =', B)
# Output:
#A = [0 1 2 3]
#B = [[ 0 1 2 3 4 5]
#      [ 6 7 8 9 10 11]]
```

Truy cập theo chỉ số (slicing)

```
import numpy as np
# mảng 3x4
a = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
print(a)
# mảng 2x2 trích xuất từ a, dòng 0+1, cột 1+2
b = a[:2, 1:3]
print(b)
# chú ý: mảng của numpy tham chiếu chứ không copy dữ liệu
print(a[0, 1]) # in "2"
b[0, 0] = 77 # b[0, 0] cũng là a[0, 1] !!! alias
print(a[0, 1]) # in "77"

a = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
c = a[:2, 1:3].copy()
c[0, 0] = 77 # b[0, 0] cũng là a[0, 1]
print(a[0, 1]) # in "2" !!! copy
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
[[2 3]
 [6 7]]
2
77
2
```

Truy cập theo chỉ số (slicing)

```
row_r1 = a[1, :] # mảng 1 chiều độ dài 4
row_r2 = a[1:2, :] # mảng 2 chiều 1x4
print(row_r1, row_r1.shape) # in ra "[5 6 7 8] (4,)"
print(row_r2, row_r2.shape) # in ra "[[5 6 7 8]] (1, 4)"
col_r1 = a[:, 1] # mảng 1 chiều độ dài 3
col_r2 = a[:, 1:2] # mảng 2 chiều 3x1
print(col_r1, col_r1.shape) # in ra "[ 2 6 10] (3,)"
print(col_r2, col_r2.shape) # in ra "[[ 2]
                             #         [ 6]
                             #         [10]] (3, 1)"
```

```
[5 6 7 8] (4,)
[[5 6 7 8]] (1, 4)
[ 2  6 10] (3,)
[[ 2]
 [ 6]
 [10]] (3, 1)
```

Các phép toán trên mảng

```
import numpy as np
x = np.array([[1, 2], [3, 4]], dtype=np.float64)
y = np.array([[5, 6], [7, 8]], dtype=np.float64)
print(x + y) # print(np.add(x, y)), xử lý khác list
print(x - y) # print(np.subtract(x, y))
print(x * y) # print(np.multiply(x, y))
print(x / y) # print(np.divide(x, y))
print(np.sqrt(x)) # khai căn tất cả các phần tử
print(2**x) # tính 2 mũ các phần tử trong x
# chú ý: phép nhân/chia thực hiện theo cặp phần tử của x và y
```

```
[[ 6.  8.]
 [10. 12.]]
[[-4. -4.]
 [-4. -4.]]
[[ 5. 12.]
 [21. 32.]]
[[0.2      0.33333333]
 [0.42857143 0.5      ]]
[[1.      1.41421356]
 [1.73205081 2.      ]]
```

```
[[ 2.  4.]
 [ 8. 16.]]
```

Các phép toán trên mảng

- Nhân ma trận (dot) và nghịch đảo

```
import numpy as np
x = np.array([[1, 2],[3, 4]])
y = np.array([[5, 6],[7, 8]])
v = np.array([1, 0])
w = np.array([0, 1])
print(v.dot(w)) # tương tự print(np.dot(v, w))
print(x.dot(v)) # tương tự print(np.dot(x, v))
print(y.dot(w)) # tương tự print(np.dot(y, w))
print(x.dot(y)) # tương tự print(np.dot(x, y))
print(np.linalg.inv(x)) # tính và in nghịch đảo của x
```

```
0
[1 3]
[6 8]
[[19 22]
 [43 50]]
[[-2.  1.]
 [ 1.5 -0.5]]
```


Các phép toán trên mảng

- Ma trận chuyển vị: `x.T` hoặc `x.transpose()`

```
import numpy as np
x = np.array([[1, 2], [3, 4]])
print(x) # in ra "[[1 2]
          #          [3 4]]"
print(x.T) # in ra "[[1 3]
                   #          [2 4]]"
# chú ý: mảng 1 chiều không có chuyển vị
y = np.array([1, 2, 3])
print(y) # in ra "[1 2 3]"
print(y.T) # in ra "[1 2 3]"
z = np.array([[1, 2, 3]])
print(z.T)
```

```
[[1 2]
 [3 4]]
[[1 3]
 [2 4]]
[1 2 3]
[1 2 3]
[[1]
 [2]
 [3]]
```

Một số thao tác thông dụng

- Cơ chế broadcasting

- Khi +, -, *, / matrix với vector thì python sẽ tự động biến đổi vector thành một matrix cùng kích thước với matrix kia, đó gọi là broadcasting. Tổng quát $(m, n) + - * / (1, n) \rightarrow (m, n)$ $(m, 1) \rightarrow (m, n)$

```
import numpy as np
x = np.array([1, 2, 4])
a = x + 1
# tự động broadcast 1 thành np.array([1, 1, 1])
print(a)
x = np.array([[1, 2, 4], [4, 2, 1]])
b = x + np.array([0, 1, 4])
# tự động broadcast np.array([0, 1, 4]) thành np.array([[0, 1, 4], [0, 1, 4]])
print(b)
x = np.array([[1, 2, 4], [4, 2, 1]])
c = x + np.array([[0], [1]])
# tự động broadcast np.array([[0], [1]]) thành np.array([[0, 0, 0], [1, 1, 1]])
print(c)
```

```
[2 3 5]
[[1 3 8]
 [4 3 5]]
[[1 2 4]
 [5 3 2]]
```

Một số thao tác thông dụng

- Tính tổng theo các trục

```
import numpy as np
x = np.array([[1, 2], [3, 4]])
print(np.sum(x)) # tính tổng toàn bộ x, in "10"
print(np.sum(x, axis=0)) # tính tổng mỗi cột, in "[4 6]"
print(np.sum(x, axis=1)) # tính tổng mỗi hàng, in "[3 7]"
```

Một số thao tác thông dụng

- Trích xuất dữ liệu theo dãy

```
import numpy as np
a = np.array([[1,2], [3, 4], [5, 6]])
print(a)
# Prints "[1 4 5]"
print(a[[0, 1, 2], [0, 1, 0]])
# Prints "[1 4 5]"
print(np.array([a[0, 0], a[1, 1], a[2, 0]]))
# Prints "[2 2]"
print(a[[0, 0], [1, 1]])
# Prints "[2 2]"
print(np.array([a[0, 1], a[0, 1]]))
```

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

Một số thao tác thông dụng

- Lọc phần tử theo chỉ số

```
import numpy as np
a = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]])
b = np.array([0, 2, 0, 1]) # b là mảng các chỉ số
print(a[np.arange(4), b]) # in ra "[1 6 7 11]"
# cộng tất cả các phần tử được lọc thêm 10
a[np.arange(4), b] += 10
print(a)
```

```
[ 1  6  7 11]
[[11  2  3]
 [ 4  5 16]
 [17  8  9]
 [10 21 12]]
```

Một số thao tác thông dụng

- Lọc dữ liệu theo điều kiện

```
import numpy as np
a = np.array([[1, 2], [3, 4], [5, 6]])
bool_idx = (a > 2)
print(bool_idx)
# lọc dữ liệu trong a, trả về một dãy
print(a[bool_idx]) # Prints "[3 4 5 6]"
print(type(a[bool_idx]), a[bool_idx].shape)
# có thể viết trực tiếp điều kiện (ngắn gọn hơn)
print(a[a > 2]) # Prints "[3 4 5 6]"
print(type(a[a > 2]), a[a > 2].shape)
```

```
[[False False]
 [ True  True]
 [ True  True]]
[3 4 5 6]
<class 'numpy.ndarray'> (4,)
[3 4 5 6]
<class 'numpy.ndarray'> (4,)
```

Một số thao tác thông dụng

- Điều chỉnh cỡ ma trận

```
import numpy as np
x = np.array([[1, 3], [4, 4], [4, 2]])
print(x)
print(x.shape) # (3, 2)
x = np.array([[1, 3], [4, 4], [4, 2]])
a = x.reshape(2, 3) # chỉnh thành 2x3
print(a) # array([[1, 3, 4], [4, 4, 2]])
x = np.array([[1, 3], [4, 4], [4, 2]])
b = x.reshape(2, -1) # tự tính chiều còn lại
print(b) # array([[1, 3, 4], [4, 4, 2]])
```

```
[[1 3]
 [4 4]
 [4 2]]
(3, 2)
[[1 3 4]
 [4 4 2]]
[[1 3 4]
 [4 4 2]]
```

Một số thao tác thông dụng

- Elementwise operation

```
import numpy as np
x = np.array([1, 2, 3])
print(np.log(x)) # lấy log cơ số e từng phần tử
print(np.abs(x)) # lấy trị tuyệt đối từng phần tử
print(np.maximum(x, 2)) # so sánh từng phần tử với 2 và lấy max
print(np.minimum(x, 2)) # so sánh từng phần tử với 2 và lấy min
print(x**2) # lũy thừa 2 từng phần tử
```

```
[0.         0.69314718 1.09861229]
[1 2 3]
[2 2 3]
[1 2 2]
[1 4 9]
```


Một số thao tác thông dụng

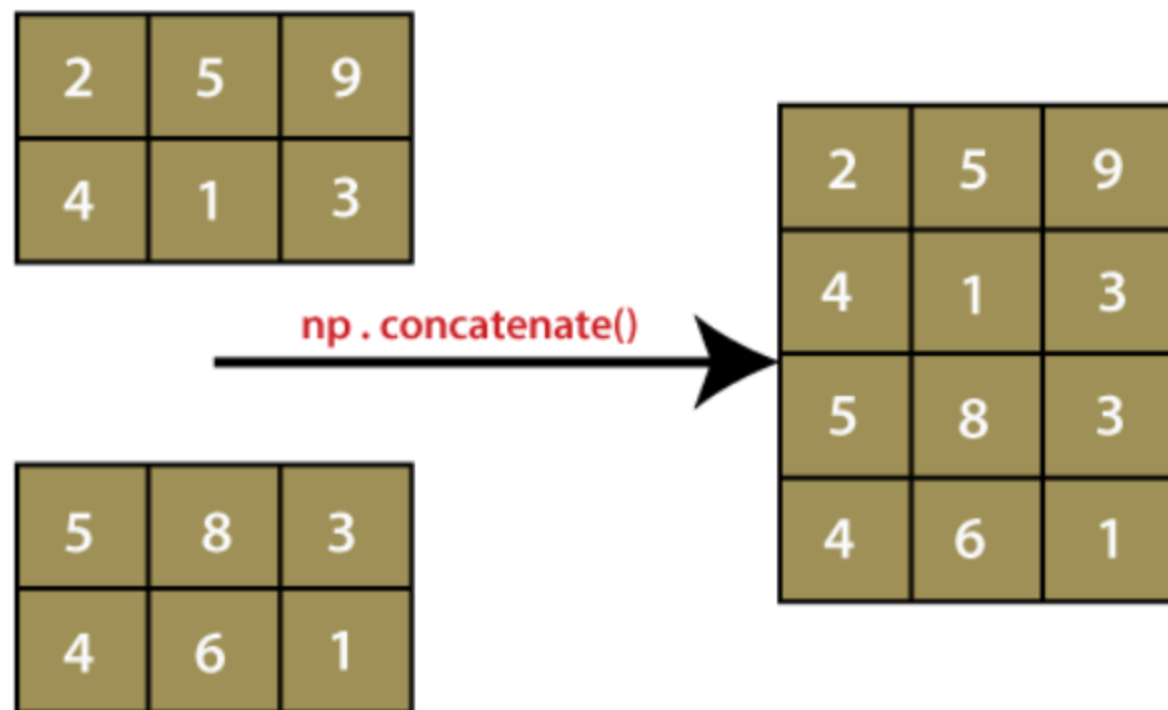
- Nối mảng (concatenation): nối các mảng với nhau theo chiều ngang hoặc chiều dọc

Cú pháp:

`numpy.concatenate ((a1, a2, ...), axis)`

Trong đó:

- a1, a2, a3 ... là các mảng có hình dạng giống nhau, ngoại trừ chiều tương ứng với trục.
- axis: Tham số này xác định trục mà mảng sẽ được nối với nhau. Theo mặc định, giá trị của nó là 0.



Ví dụ

```
In [1]: import numpy as np
x=np.array([[1,2],[3,4]])
y=np.array([[12,30]])
z=np.concatenate((x,y))
z
```

```
Out[1]: array([[ 1,  2],
               [ 3,  4],
               [12, 30]])
```

```
In [2]: import numpy as np
x=np.array([[1,2],[3,4]])
y=np.array([[12,30]])
z=np.concatenate((x,y), axis=0)
z
```

```
Out[2]: array([[ 1,  2],
               [ 3,  4],
               [12, 30]])
```

```
In [3]: import numpy as np
x=np.array([[1,2],[3,4]])
y=np.array([[12,30]])
z=np.concatenate((x,y.T), axis=1)
z
```

```
Out[3]: array([[ 1,  2, 12],
               [ 3,  4, 30]])
```

Ví dụ

- Ví dụ : `numpy.concatenate()` với `axis = None`

```
In [4]: import numpy as np
x=np.array([[1,2],[3,4]])
y=np.array([[12,30]])
z=np.concatenate((x,y), axis=None)
z
```

```
Out[4]: array([ 1,  2,  3,  4, 12, 30])
```

Một số thao tác thông dụng

- **Stacking: `numpy.stack()`**
- **Vertical & Horizontal Stacking:**
 - `hstack()` to stack along rows.
 - `vstack()` to stack along columns.

```
In [10]: import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.stack((arr1, arr2), axis=1)
print(arr)
```

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

```
In [11]: import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.hstack((arr1, arr2))
print(arr)
```

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

```
In [12]: import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.vstack((arr1, arr2))
print(arr)
```

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

Tham khảo

- <https://numpy.org/learn/>
- https://phamdinhkhanh.github.io/deepai-book/ch_appendix/appendix_numpy.html

Thực Hành

1. Tạo một ma trận 4x4 toàn các giá trị False (hint: `np.full()`)
2. Cho một dãy số nguyên 10 phần tử, hãy tách lấy tất cả những phần tử lẻ cho vào một mảng (**hint**: lọc dữ liệu theo điều kiện)
3. Cho một dãy số tự nhiên 10 phần tử, hãy thay thế tất cả những phần tử lẻ bằng số -1 (**hint**: lọc dữ liệu theo điều kiện)
4. Hai mảng a và b có cùng số dòng, hãy ghép chúng theo các dòng thành mảng c, các cột của a rồi đến các cột của b (**hint**: dùng `numpy.concatenate(,)`)
5. Mảng a và b có cùng số cột, hãy ghép chúng theo các cột thành mảng c, các dòng của a rồi đến của b (**hint**: dùng `numpy.concatenate(,)`)

Matplotlib

Nội Dung

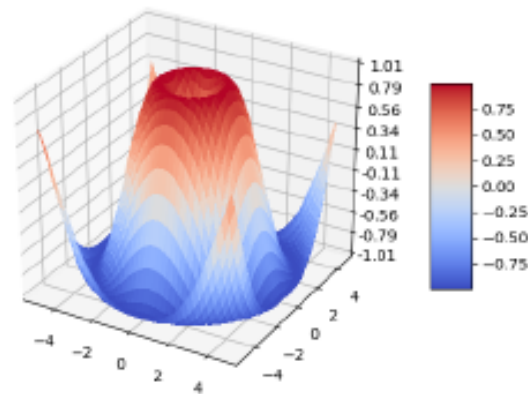
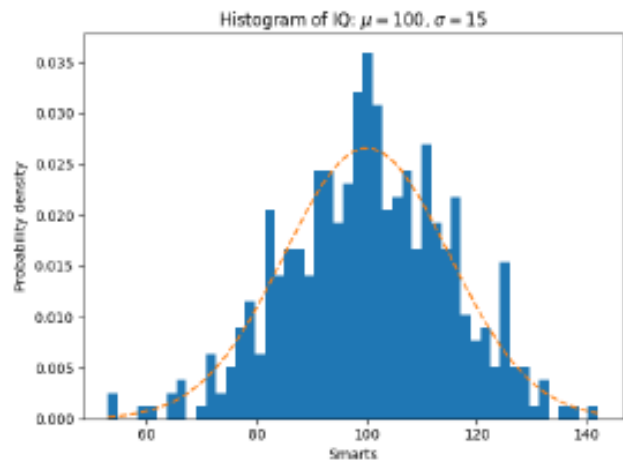
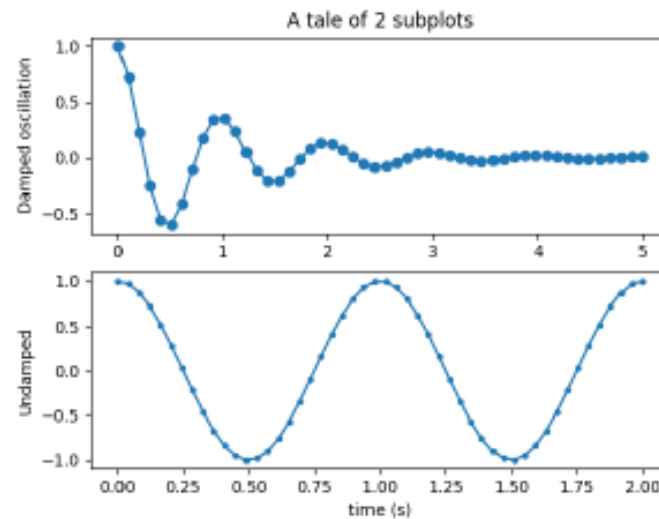
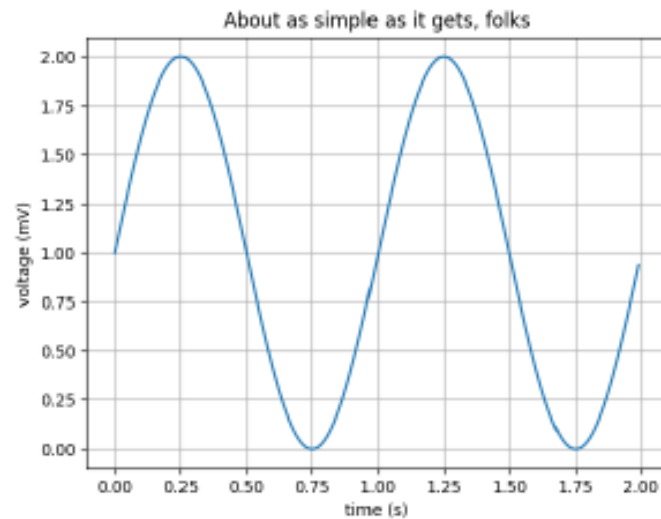
- Giới thiệu về matplotlib
- Các biểu đồ thông dụng

Giới thiệu về matplotlib

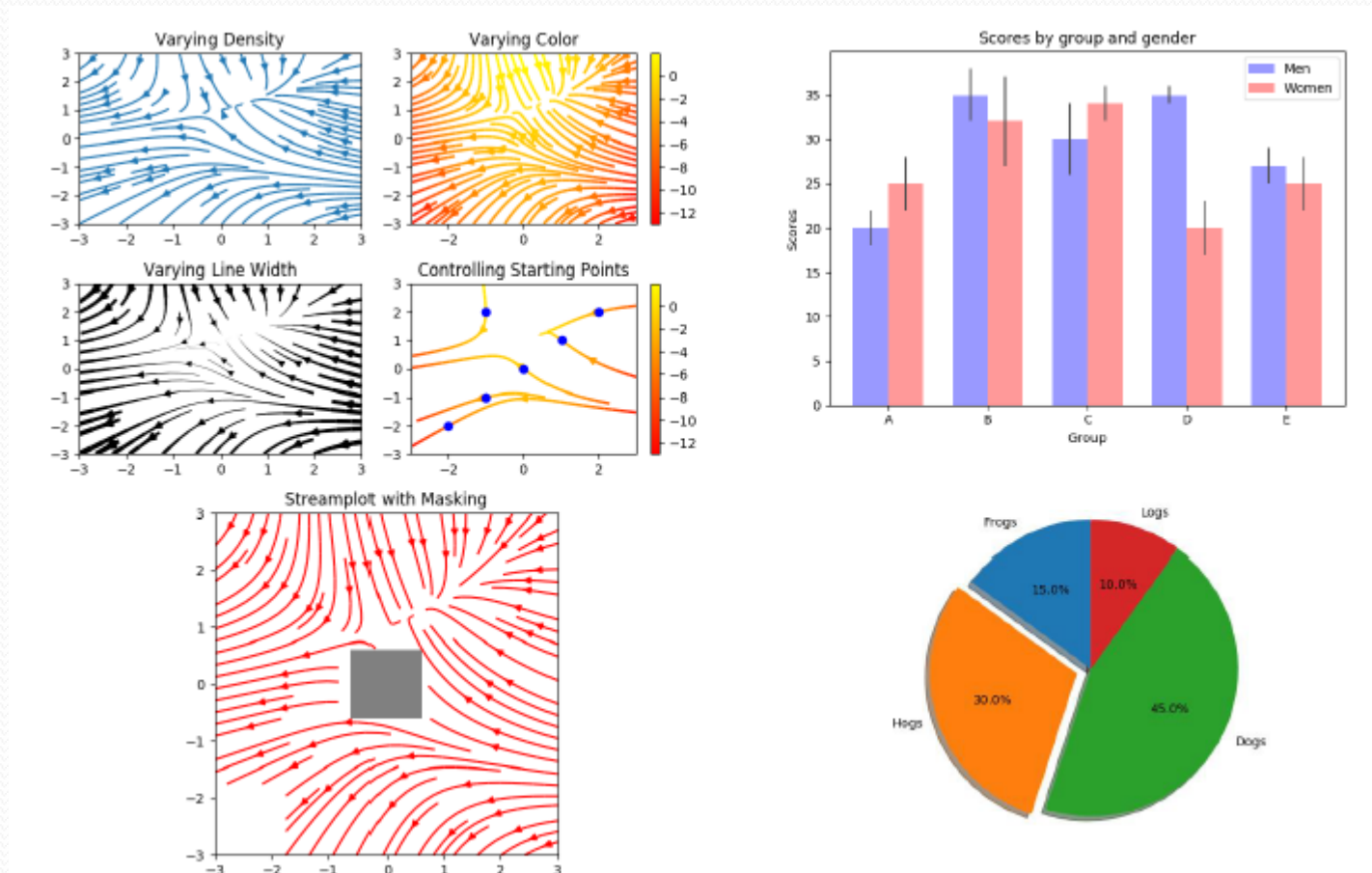
- “matplotlib” là thư viện chuyên về vẽ biểu đồ, mở rộng từ numpy.
- Hỗ trợ rất nhiều loại biểu đồ như biểu đồ dòng, đường, tần suất (histograms), phổ, tương quan, scatterplots,...
- Cài đặt :

`pip install matplotlib`

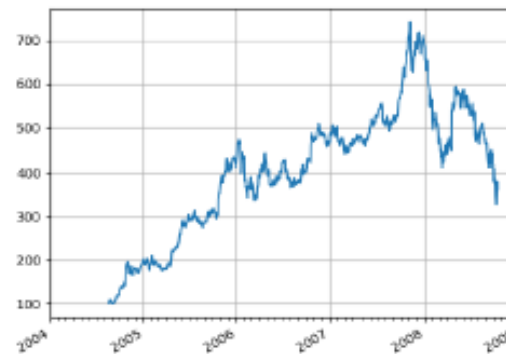
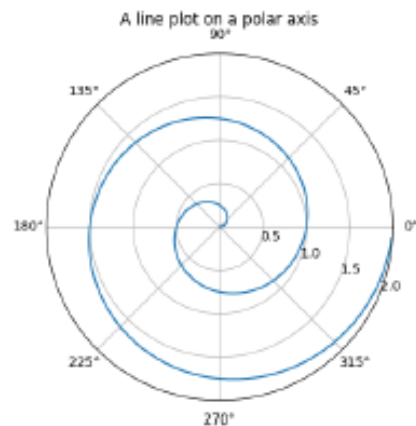
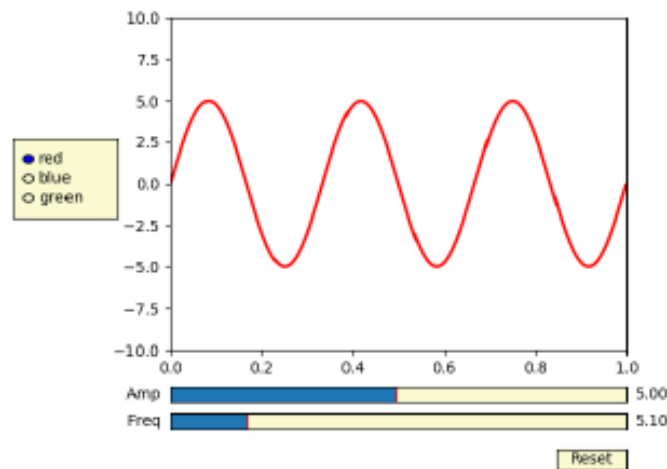
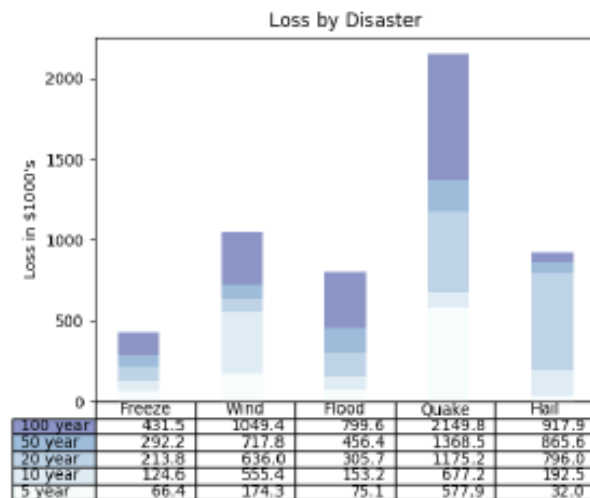
Giới thiệu về matplotlib



Giới thiệu về matplotlib

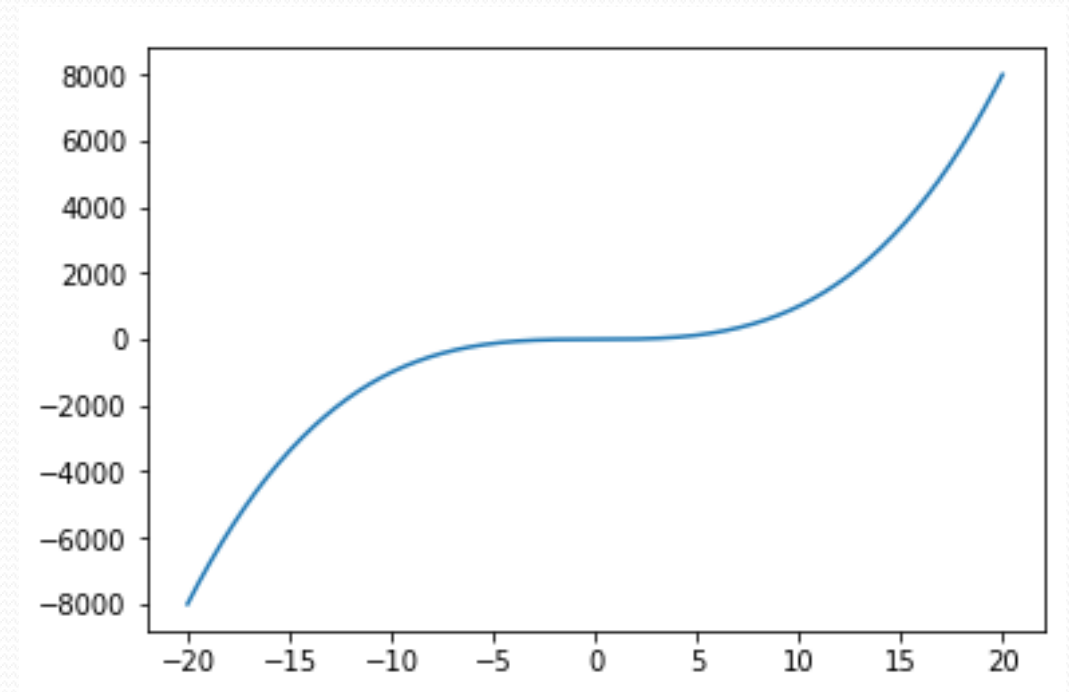


Giới thiệu về matplotlib



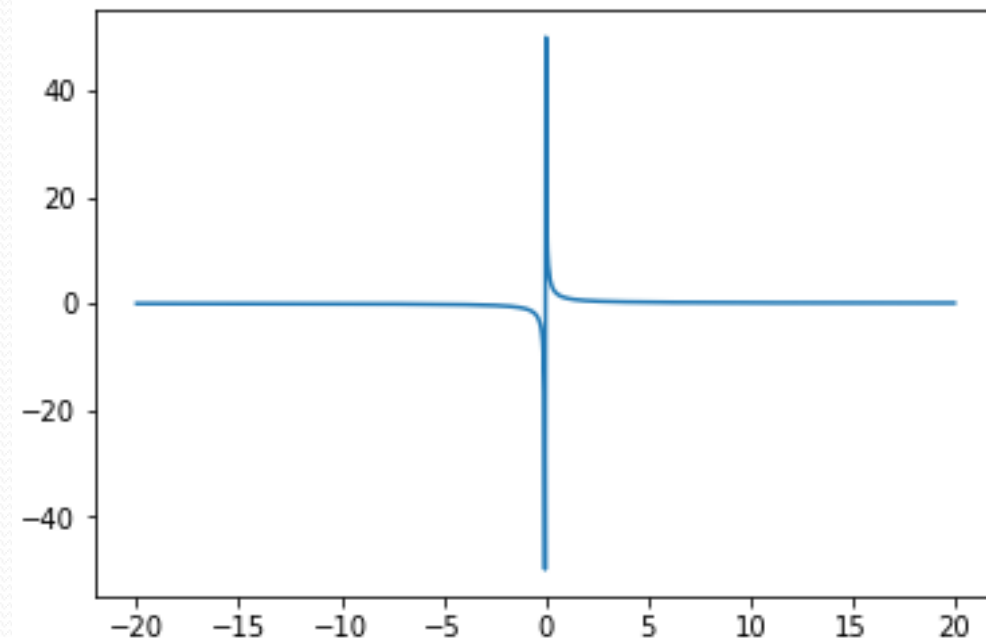
Các biểu đồ thông dụng

```
import numpy as np # thư viện numpy
import matplotlib.pyplot as plt # thư viện pyplot
# chia đoạn từ -20 đến 20 thành 1000 đoạn
x = np.linspace(-20, 20, 1000)
# tính y
y = x ** 3
# vẽ biểu đồ tương quan giữa x và y
plt.plot(x, y)
# hiển thị biểu đồ
plt.show()
```



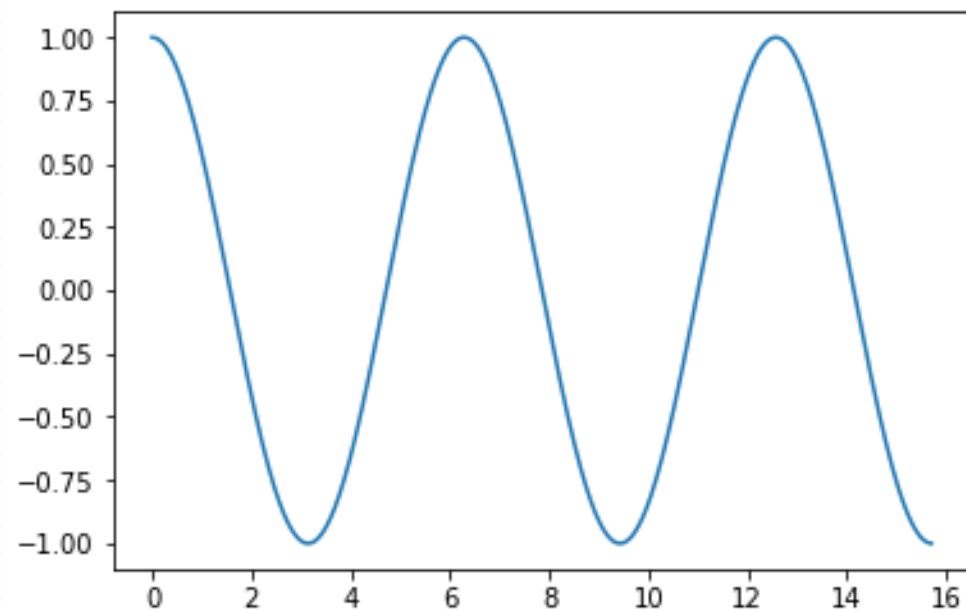
Các biểu đồ thông dụng

```
import numpy as np # thư viện numpy
import matplotlib.pyplot as plt # thư viện pyplot
# chia đoạn từ -20 đến 20 thành 1000 đoạn
x = np.linspace(-20, 20, 1000)
# tính y
y = x ** -1
# vẽ biểu đồ tương quan giữa x và y
plt.plot(x, y)
# hiển thị biểu đồ
plt.show()
```



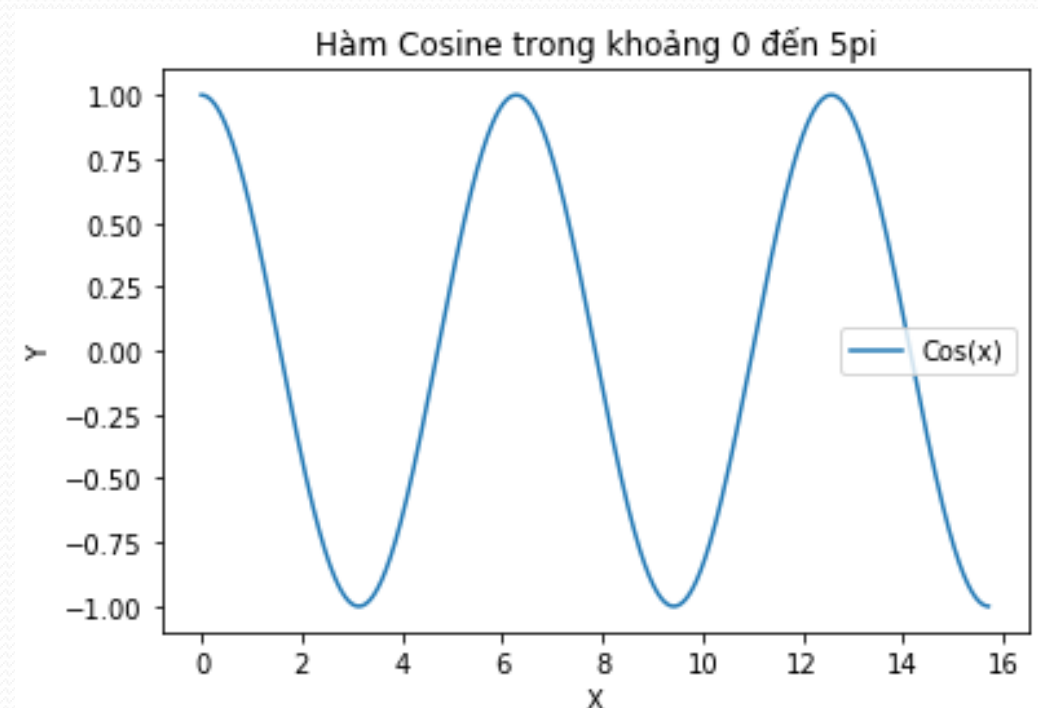
Các biểu đồ thông dụng

```
import numpy as np # thư viện numpy
import matplotlib.pyplot as plt # thư viện pyplot
# chia đoạn từ 0 đến 5pi thành các đoạn con 0.01
x = np.arange(0, 5 * np.pi, 0.01)
# tính cosine tương ứng với từng phần tử của x
y = np.cos(x)
# vẽ biểu đồ tương quan giữa x và y
plt.plot(x, y)
# hiển thị biểu đồ
plt.show()
```



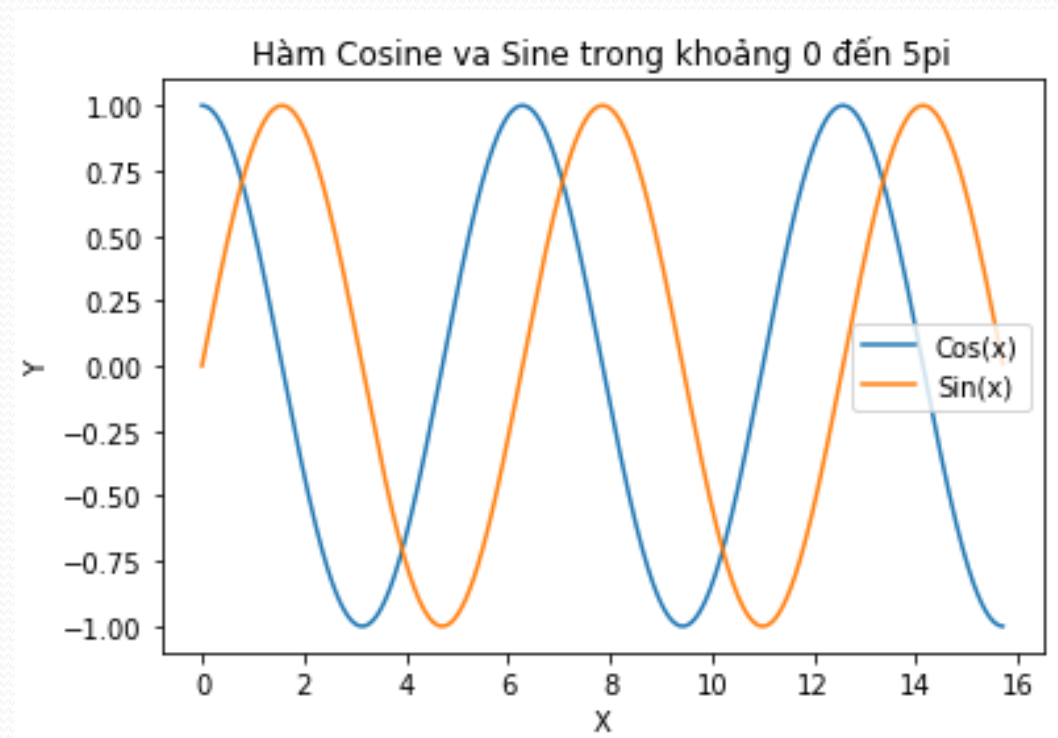
Các biểu đồ thông dụng

```
import numpy as np # thư viện numpy
import matplotlib.pyplot as plt # thư viện pyplot
x = np.arange(0, 5 * np.pi, 0.01)
y = np.cos(x)
plt.plot(x, y)
# các thông tin bổ sung cho biểu đồ
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Hàm Cosine trong khoảng 0 đến 5pi')
plt.legend(['Cos(x)'])
# hiển thị biểu đồ
plt.show()
```



Các biểu đồ thông dụng

```
import numpy as np # thư viện numpy
import matplotlib.pyplot as plt # thư viện pyplot
x = np.arange(0, 5 * np.pi, 0.01)
y = np.cos(x)
z = np.sin(x)
plt.plot(x, y)
plt.plot(x, z)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Hàm Cosine và Sine trong khoảng 0 đến 5pi')
plt.legend(['Cos(x)', 'Sin(x)'])
# hiển thị biểu đồ
plt.show()
```



Các biểu đồ thông dụng

- Điều kiện cần: đã có sẵn dữ liệu
- Có thể có 4 bước cơ bản:
 1. Chọn loại biểu đồ phù hợp
 - Tùy thuộc rất nhiều vào loại dữ liệu
 - Tùy thuộc vào mục đích sử dụng của người dùng
 2. Thiết lập các thông số cho biểu đồ
 - Thông số của các trục, ý nghĩa, tỉ lệ chia,...
 - Các điểm nhấn trên bản đồ
 - Góc nhìn, mẫu tô, màu và các chi tiết khác
 - Các thông tin bổ sung
 3. Vẽ biểu đồ
 4. Lưu ra file

Biểu đồ dạng đường (line plot)

- Biểu đồ thể hiện tương quan giữa X và Y
- Cú pháp:
 - `plot([x], y, [fmt], data=None, **kwargs)`
 - `plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)`
- “fmt” là quy cách vẽ đường
- “data” là nhãn của dữ liệu
- `**kwargs`: tham số vẽ đường
- Vẽ nhiều lần trên một biểu đồ
- Kết quả trả về là một list các đối tượng Line2D

https://matplotlib.org/3.1.0/api/_as_gen/matplotlib.pyplot.plot.html

Biểu đồ dạng đường (line plot)

- “fmt” gồm 3 phần `fmt = '[color][marker][line]'`
- `[color]` –viết tắt tên màu:
 - ‘b’ –blue
 - ‘g’ –green
 - ‘r’ –red
 - ‘c’ –cyan
 - ‘m’ –magenta
 - ‘y’ –yellow
 - ‘b’ –black
 - ‘w’ –white
 - `#rrggbb` –chỉ ra mã màu theo hệ RGB

Biểu đồ dạng đường (line plot)

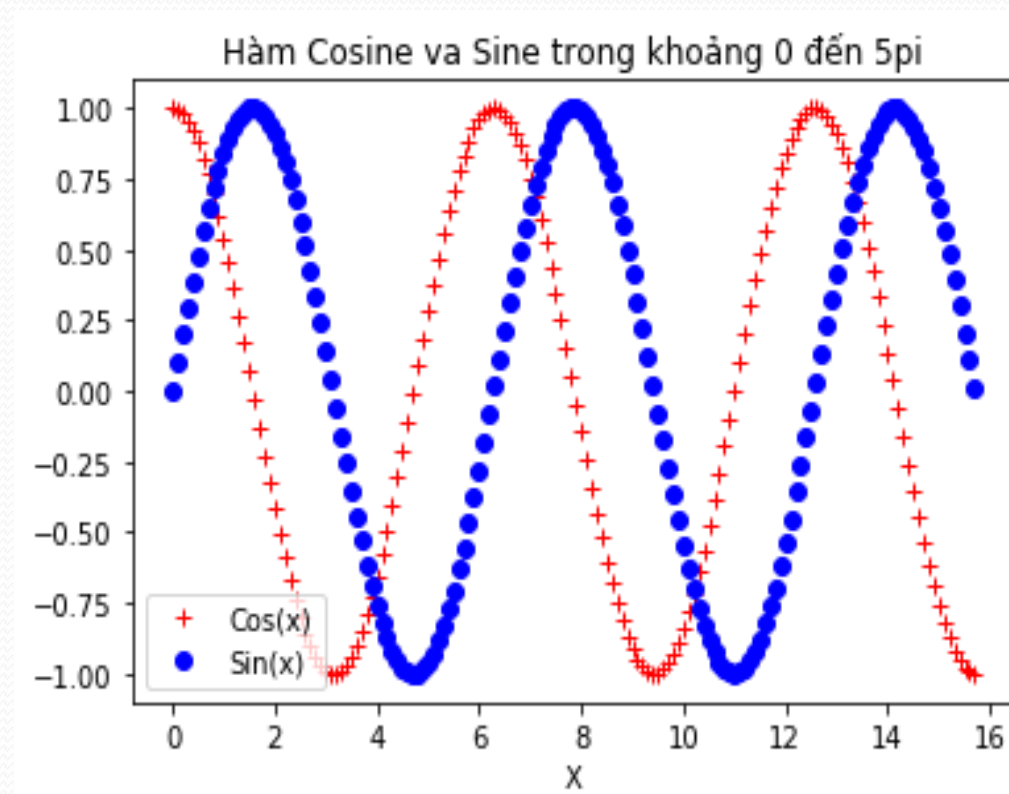
- [marker] –cách đánh dấu dữ liệu:
 - ‘o’ –hình tròn
 - ‘v’ –tam giác xuống (‘^’, ‘<’, ‘>’)
 - ‘*’ –ngôi sao
 - ‘.’ –chấm
 - ‘p’ –ngũ giác
- [line] –cách vẽ đường:
 - ‘-’ –nét liền
 - ‘--’ –nét đứt
 - ‘-.’ –gạch chấm
 - ‘:’ –đường chấm

Biểu đồ dạng đường (line plot)

- `plot(x, y)` # plot x and y using default line style and color
- `plot(x, y, 'bo')` # plot x and y using blue circle markers
- `plot(y)` # plot y using x as index array 0..N-1
- `plot(y, 'r+')` # ditto, but with red plusses

Biểu đồ dạng đường (line plot)

```
import numpy as np # thư viện numpy
import matplotlib.pyplot as plt # thư viện pyplot
x = np.arange(0, 5 * np.pi, 0.1)
y = np.cos(x)
z = np.sin(x)
plt.plot(x, y, 'r+')
plt.plot(x, z, 'bo')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Hàm Cosine và Sine trong khoảng 0 đến 5pi')
plt.legend(['Cos(x)', 'Sin(x)'])
# hiển thị biểu đồ
plt.show()
```

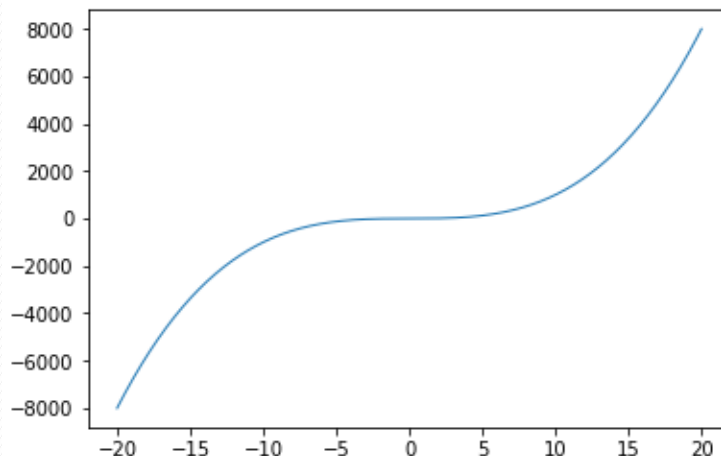


Biểu đồ dạng đường (line plot)

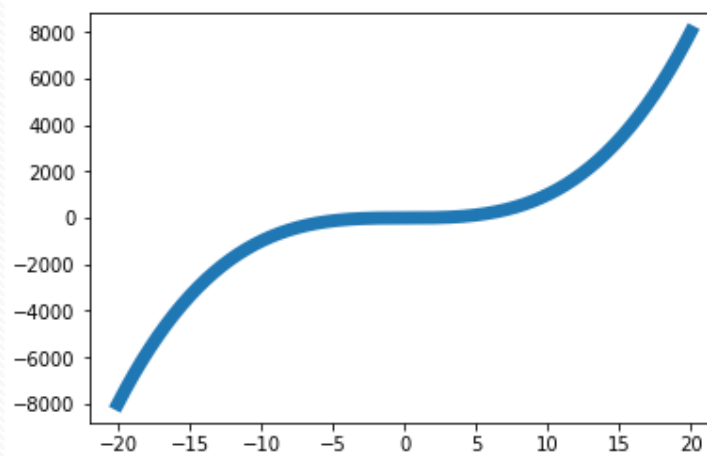
- `plot(x, y, 'go--', linewidth=2, markersize=12)`
- `plot(x, y, color='green', marker='o', linestyle='dashed', ... linewidth=2, markersize=12)`

```
import numpy as np # thư viện numpy
import matplotlib.pyplot as plt # thư viện pyplot
# chia đoạn từ -20 đến 20 thành 1000 đoạn
x = np.linspace(-20, 20, 1000)
# tính y
y = x ** 3
# vẽ biểu đồ tương quan giữa x và y
plt.plot(x, y, linewidth=7)
# hiển thị biểu đồ
plt.show()
```

linewidth=1

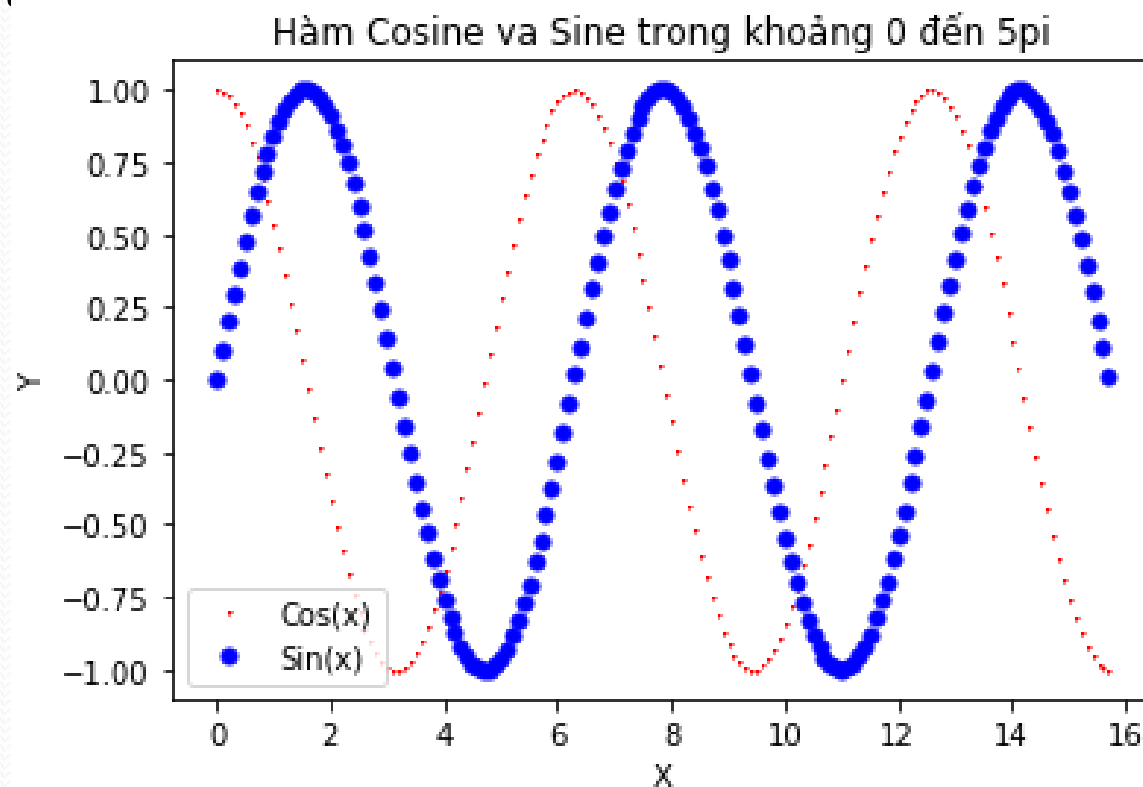


linewidth=7



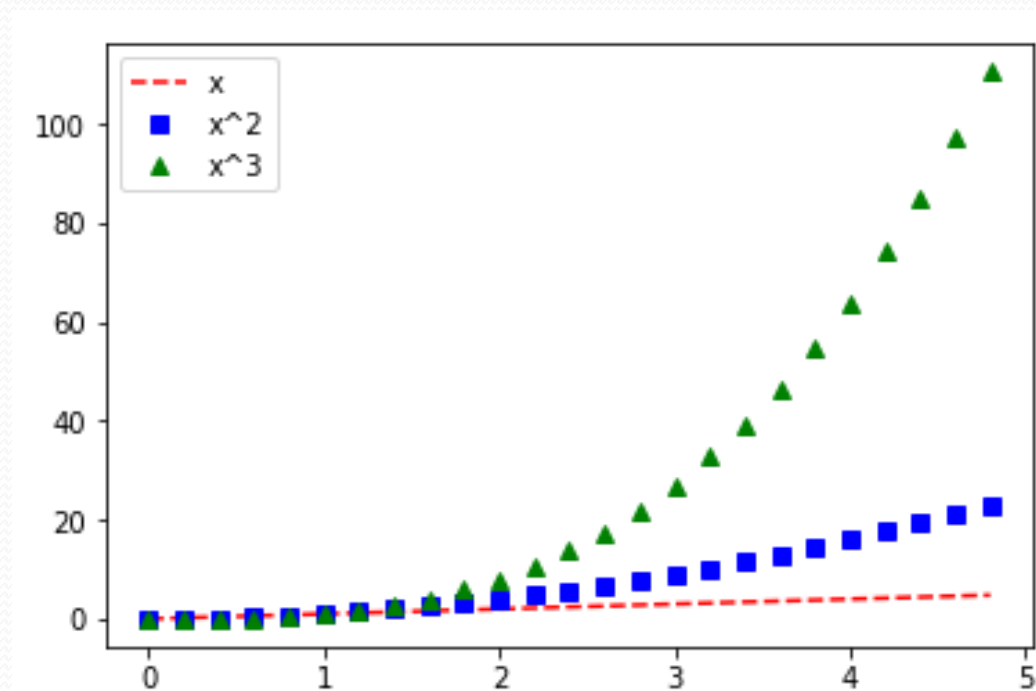
Biểu đồ dạng đường (line plot)

```
import numpy as np # thư viện numpy
import matplotlib.pyplot as plt # thư viện pyplot
x = np.arange(0, 5 * np.pi, 0.1)
y = np.cos(x)
z = np.sin(x)
plt.plot(x, y, 'r+', markersize=1)
plt.plot(x, z, 'bo', markersize=5)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Hàm Cosine và Sine trong khoảng 0 đến 5pi')
plt.legend(['Cos(x)', 'Sin(x)'])
# hiển thị biểu đồ
plt.show()
```



Biểu đồ dạng đường (line plot)

```
import numpy as np
import matplotlib.pyplot as plt
# chia đoạn 0-5 thành các bước 0.2
x = np.arange(0., 5., 0.2)
# Vẽ 3 đường:
# -màu đỏ nét đứt:  $y = x$ 
# -màu xanh dương, đánh dấu ô vuông:  $y = x^2$ 
# -màu xanh lá, đánh dấu tam giác:  $y = x^3$ 
plt.plot(x, x, 'r--', x, x**2, 'bs', x, x**3, 'g^')
plt.legend(['x', 'x^2', 'x^3'])
plt.show()
```



Biểu đồ dạng cột (bar plot)

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

```
D = { 'CNTT': 500,
```

```
'Toan': 310,
```

```
'Hoa': 150,
```

```
'Sinh': 280,
```

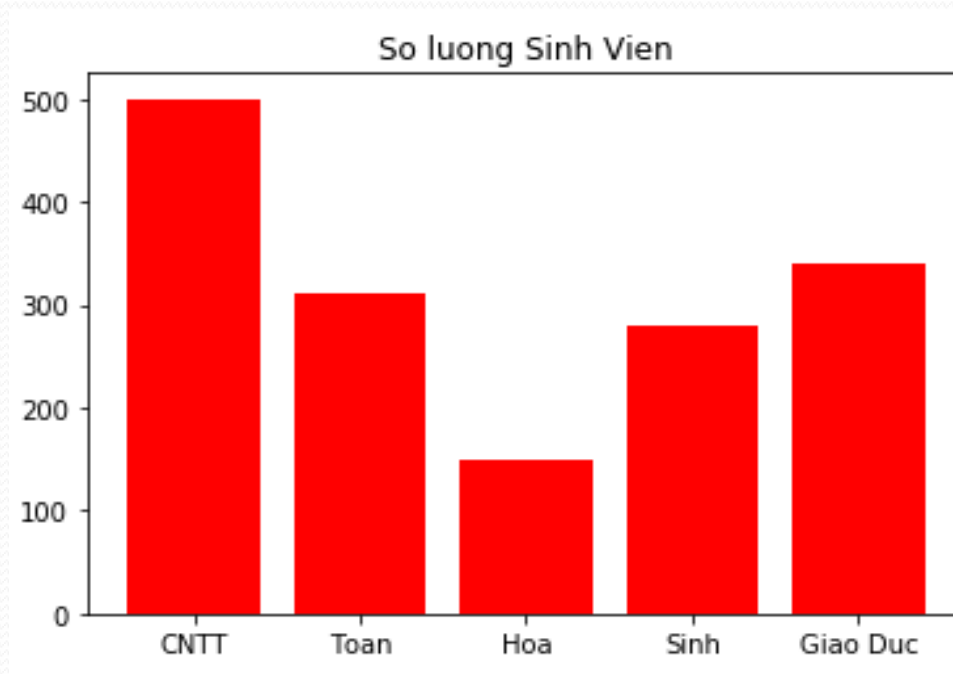
```
'Giao Duc': 340}
```

```
plt.bar(range(len(D)), list(D.values()),color='r')
```

```
plt.xticks(range(len(D)), D.keys())
```

```
plt.title('So luong Sinh Vien')
```

```
plt.show()
```



Biểu đồ dạng cột (bar plot)

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

```
D = { 'CNTT': 500,
```

```
'Toan': 310,
```

```
'Hoa': 150,
```

```
'Sinh': 280,
```

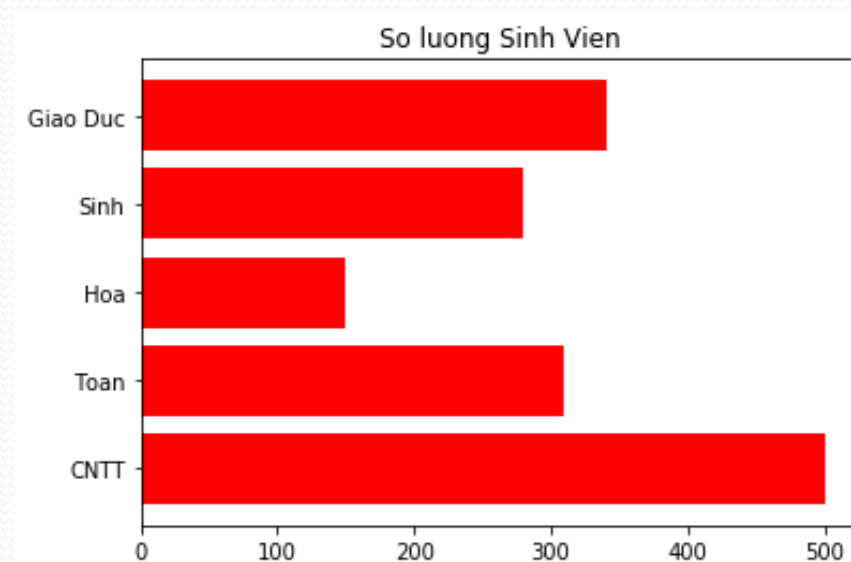
```
'Giao Duc': 340}
```

```
plt.barh(range(len(D)), list(D.values()),color='r')
```

```
plt.yticks(range(len(D)), D.keys())
```

```
plt.title('So luong Sinh Vien')
```

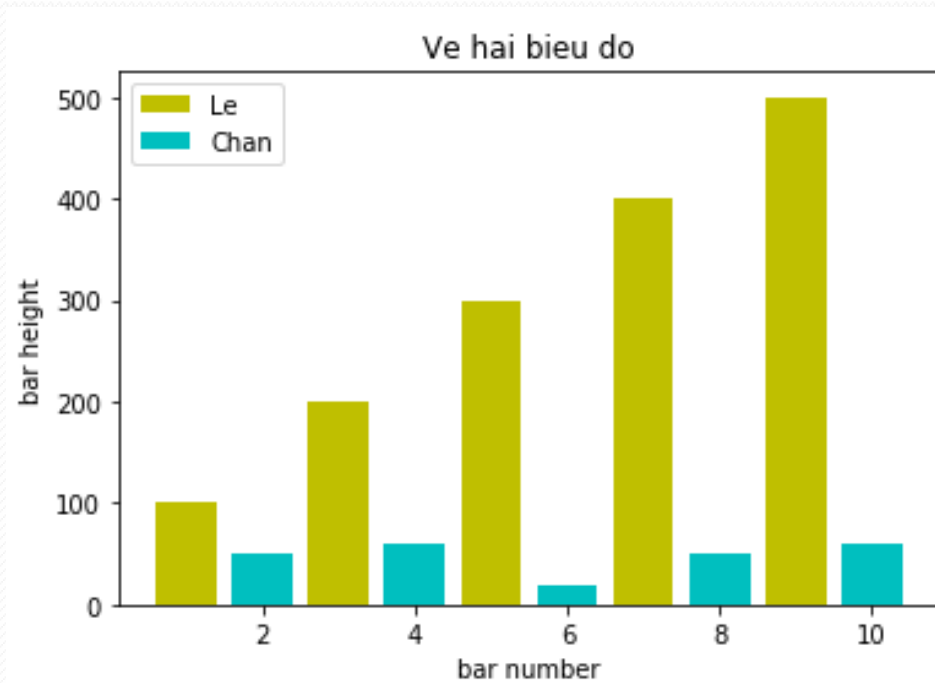
```
plt.show()
```



barh(): Plot a horizontal bar plot.

Biểu đồ dạng cột (bar plot)

```
import matplotlib.pyplot as plt
plt.bar([1,3,5,7,9],[100,200,300,400,500], label="Le",color='y')
plt.bar([2,4,6,8,10],[50,60,20,50,60], label="Chan", color='c')
plt.legend()
plt.xlabel('bar number')
plt.ylabel('bar height')
plt.title('Ve hai bieu do')
plt.show()
```



Biểu đồ bánh (pie chart)

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

```
D = { 'CNTT': 500,
```

```
'Toan': 310,
```

```
'Hoa': 150,
```

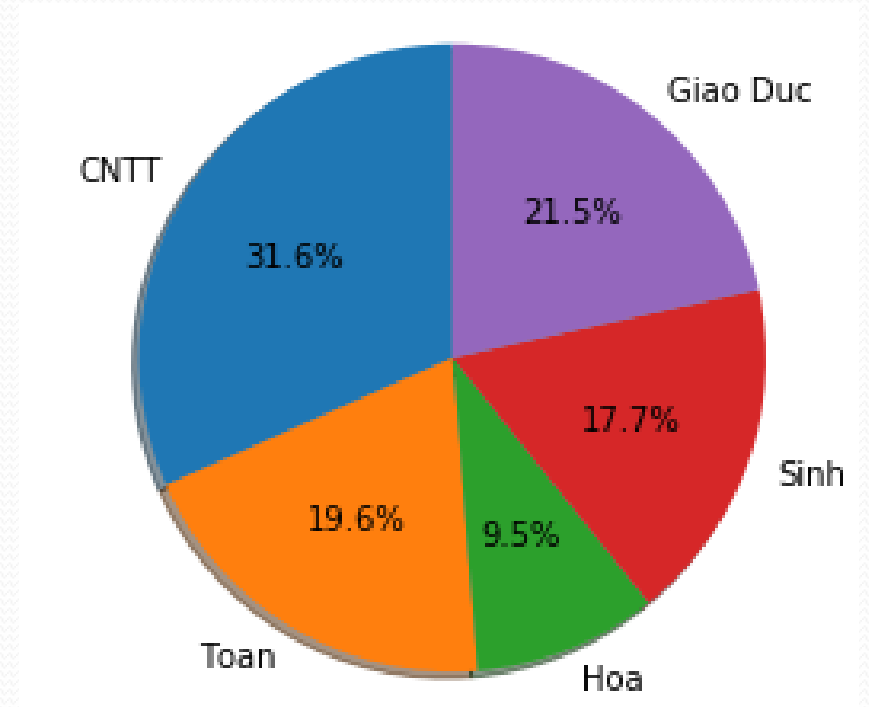
```
'Sinh': 280,
```

```
'Giao Duc': 340}
```

```
plt.pie(D.values(), labels=D.keys(), autopct='%1.1f%%', shadow=True, startangle=90)
```

```
plt.axis('equal')# trục x = trục y
```

```
plt.show()
```



Biểu đồ bánh (pie chart)

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

```
D = { 'CNTT': 500,
```

```
'Toan': 310,
```

```
'Hoa': 150,
```

```
'Sinh': 280,
```

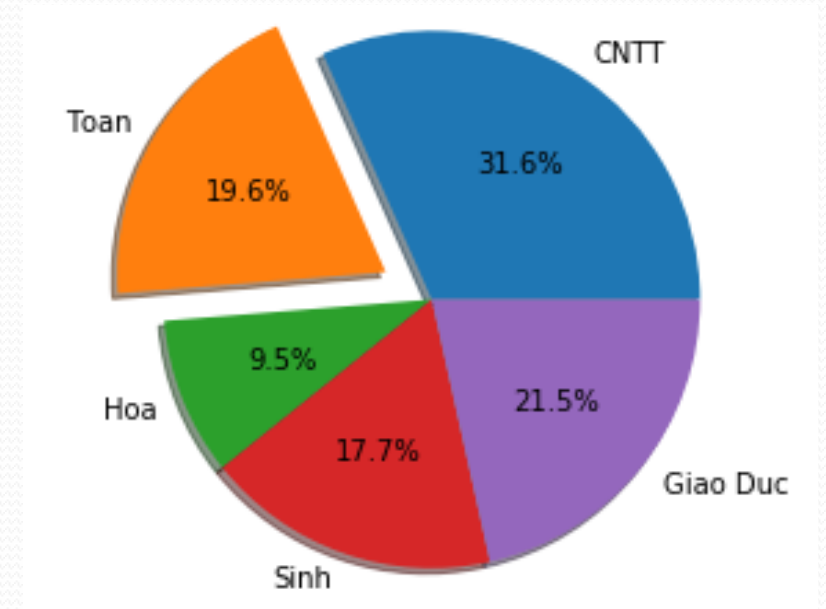
```
'Giao Duc': 340}
```

```
explode = (0, 0.2, 0, 0, 0) # only "explode" the 2nd slice (i.e. 'Toan')
```

```
plt.pie(D.values(),explode=explode, labels=D.keys(), autopct='%1.1f%%', shadow=True, startangle=0)
```

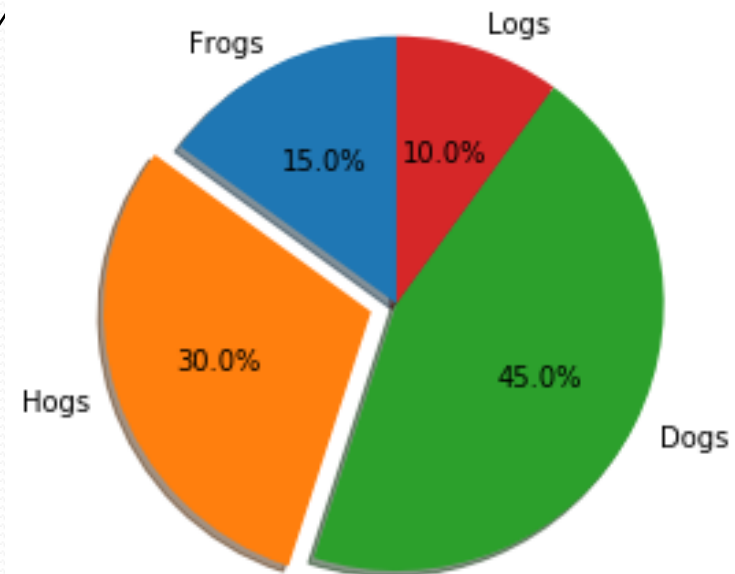
```
plt.axis('equal')# trục x = trục y
```

```
plt.show()
```



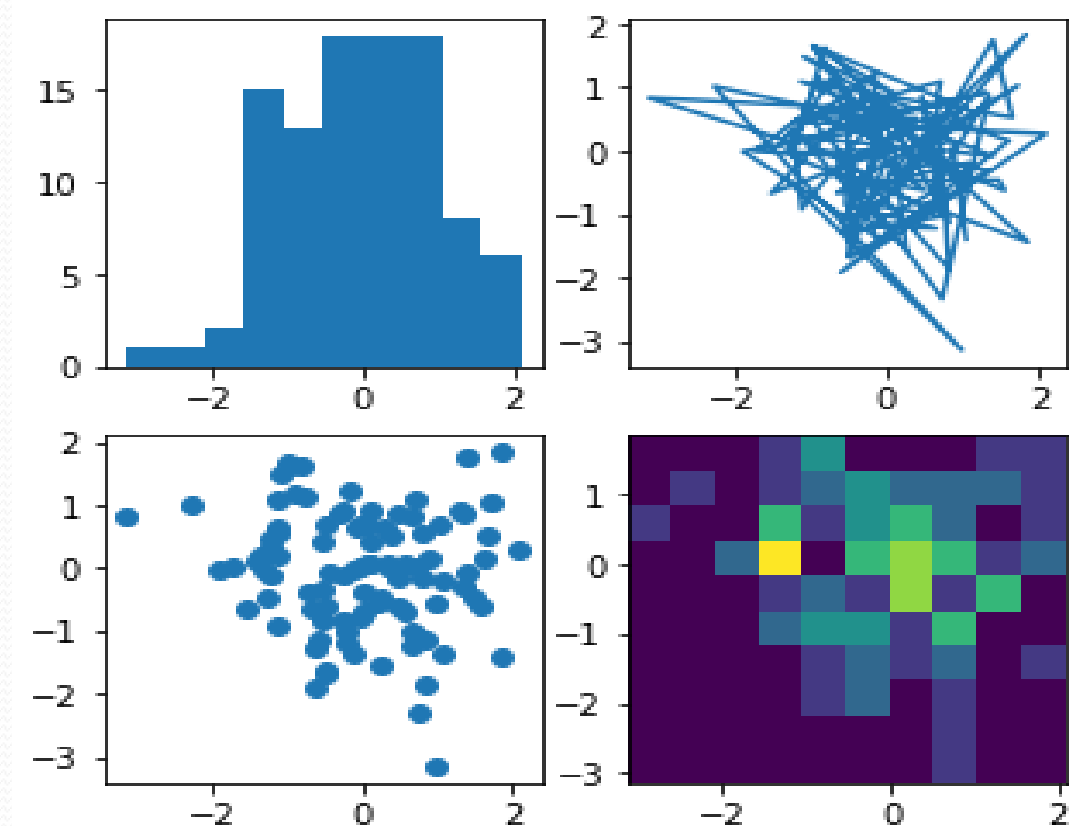
Biểu đồ bánh (pie chart)

```
import matplotlib.pyplot as plt
# Pie chart, where the slices will be ordered and plotted counter-clockwise:
labels = 'Frogs', 'Hogs', 'Dogs', 'Logs'
sizes = [15, 30, 45, 10]
explode = (0, 0.1, 0, 0) # only "explode" the 2nd slice (i.e. 'Hogs')
plt.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%',
        shadow=True, startangle=90)
plt.axis('equal')
plt.show()
```



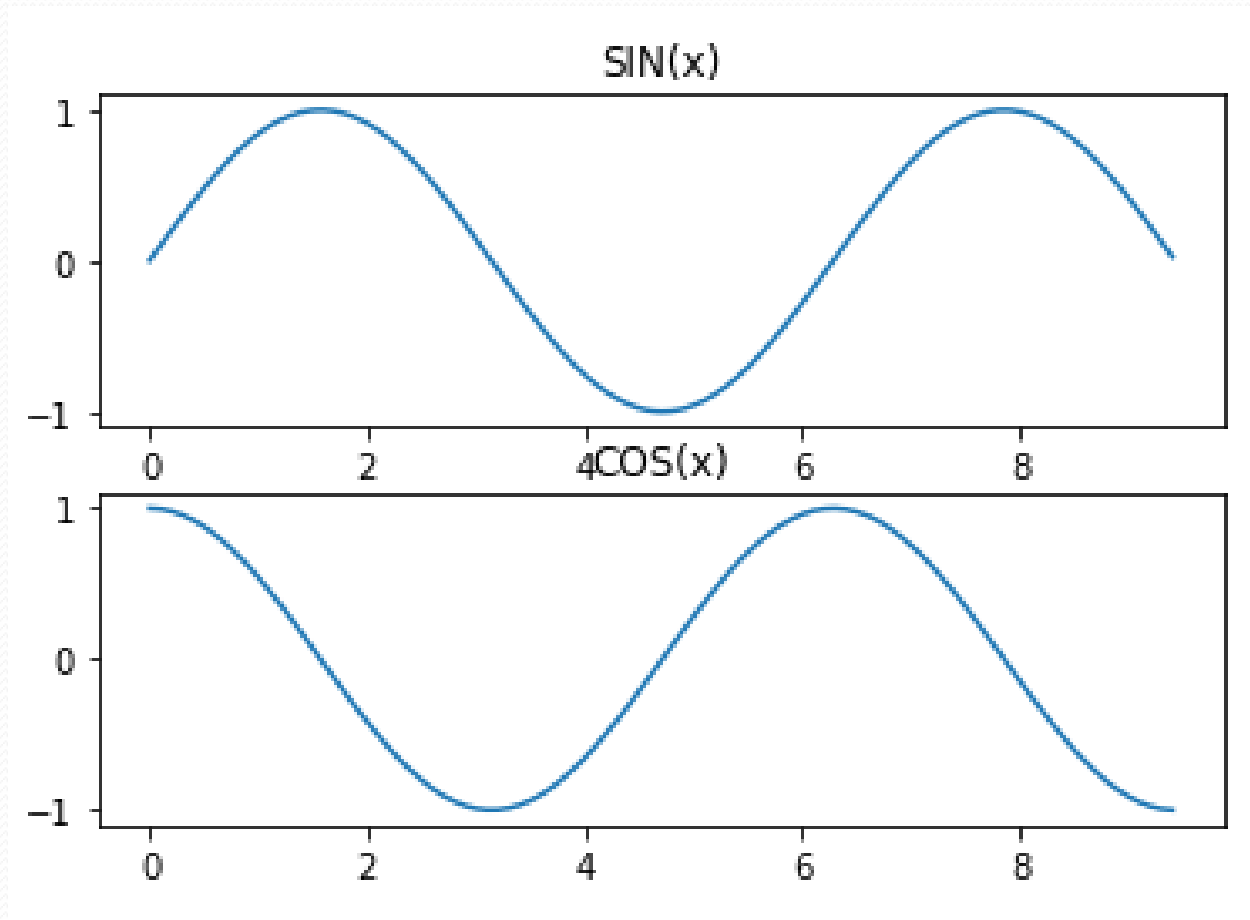
Chia thành các biểu đồ con

```
import matplotlib.pyplot as plt
import numpy as np
np.random.seed(19680801)
data = np.random.randn(2, 100)
fig, axs = plt.subplots(2, 2, figsize=(5, 5))
axs[0, 0].hist(data[0])
axs[1, 0].scatter(data[0], data[1])
axs[0, 1].plot(data[0], data[1])
axs[1, 1].hist2d(data[0], data[1])
plt.show()
```



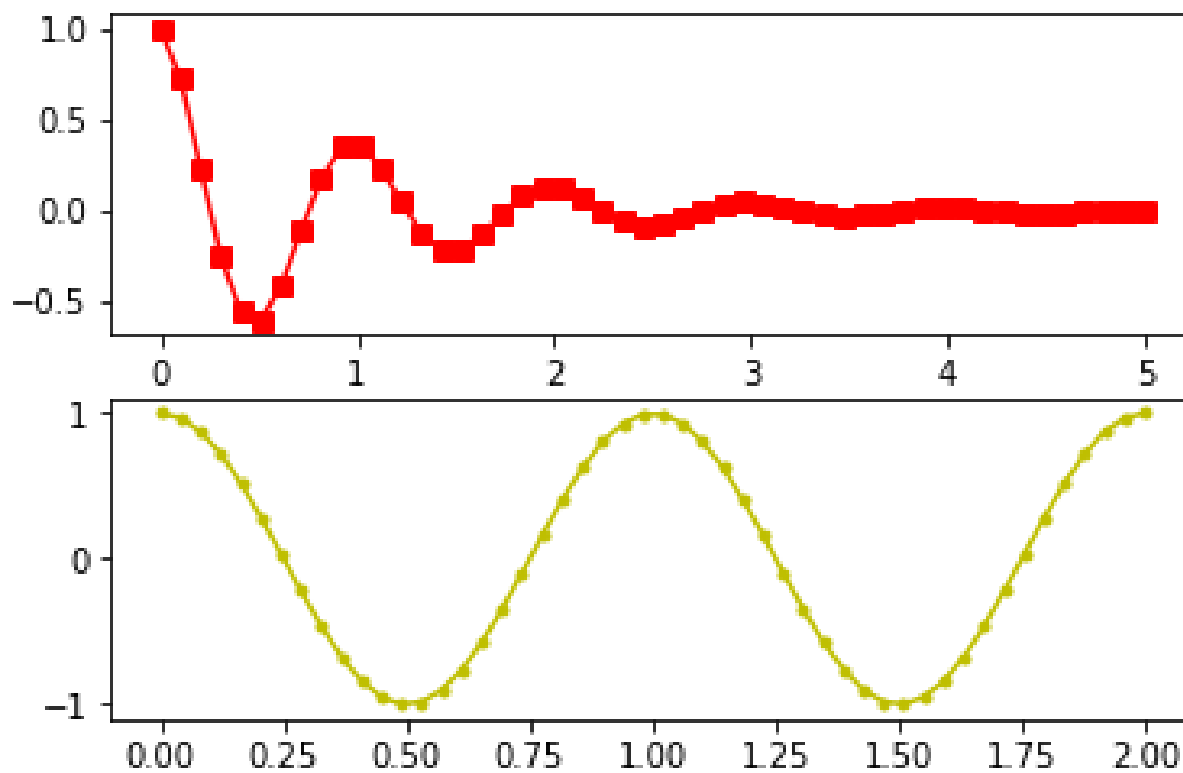
Chia thành các biểu đồ con

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)
plt.subplot(2, 1, 1) # biểu đồ 1
plt.plot(x, y_sin)
plt.title('SIN(x)')
plt.subplot(2, 1, 2) # biểu đồ 2
plt.plot(x, y_cos)
plt.title('COS(x)')
plt.show()
```



Chia thành các biểu đồ con

```
import numpy as np
import matplotlib.pyplot as plt
x1 = np.linspace(0.0, 5.0)
x2 = np.linspace(0.0, 2.0)
y1 = np.cos(2 * np.pi * x1) * np.exp(-x1)
y2 = np.cos(2 * np.pi * x2)
plt.subplot(2, 1, 1)
plt.plot(x1, y1, 'rs-')
plt.subplot(2, 1, 2)
plt.plot(x2, y2, 'y.-')
plt.show()
```



Tham khảo

- <https://matplotlib.org/>
- <https://matplotlib.org/stable/tutorials/index>
- https://phamdinhkhanh.github.io/deepai-book/ch_appendix/appendix_matplotlib.html