THƯ VIỆN TRONG PYTHON HỖ TRỢ PHÂN TÍCH DỮ LIỆU

Nội dung

1. Thư viện numpy
2. Mảng và chỉ số
3. Thư viện matplotlib
4. Biểu đồ trong matplotlib
5. Thư viện Pandas

Thư viện trong python

- **os**: xử lý file và tương tác với hệ điều hành
- **network và graph**: làm việc với dữ liệu đồ thị, có thể làm việc với dữ liệu rất lớn (đồ thị lên đến hàng triệu đỉnh)
- **regular expressions**: tìm kiếm mẫu trong dữ liệu text
- **Beautiful Soup**: trích xuất dữ liệu từ file HTML hoặc từ website

3

Thư viện trong python

- **Pandas**: chuyên sử dụng cho quản lý và tương tác với dữ liệu có cấu trúc, được sử dụng rộng rãi trong việc thu thập và tiền xử lý dữ liệu
- **Scikit Learn**: chuyên về học máy (machine learning), dựa trên **NumPy, SciPy** và **matplotlib**; thư viện này có sẵn nhiều công cụ hiệu quả cho học máy: các thuật toán phân lớp, hồi quy, phân cụm và giảm chiều dữ liệu
- **Statsmodels**: cho phép người sử dụng khai phá dữ liệu, ước lượng mô hình thống kê và kiểm định

4

Thư viện trong python

- **SciPy (Scientific Python)**: dựa trên Numpy, cung cấp các công cụ mạnh, chẳng hạn như: biến đổi fourier rời rạc, đại số tuyến tính, tối ưu hóa và ma trận thưa
- **NumPy (Numerical Python)**: thư viện chuyên về xử lý dữ liệu số (nhiều chiều); thư viện cũng chứa các hàm đại số tuyến tính cơ bản, biến đổi fourier, sinh số ngẫu nhiên nâng cao,…
- **Matplotlib**: sử dụng để vẽ biểu đồ, hỗ trợ rất nhiều loại biểu đồ, đồ thị khác nhau.

5

Thư viện trong python

- **Theano**: chuyên dùng tính toán hiệu quả các mảng nhiều chiều, sử dụng rộng rãi trong học máy
- **Keras**: thư viện cấp cao chuyên về học máy, sử dụng Theano, TensorFlow làm phụ trợ
- **TensorFlow:** chuyên dùng cho học máy của Google, đặc biệt là các mạng thần kinh nhân tạo
- **Scrapy**: thu thập thông tin trên web, rất phù hợp với việc lấy các dữ liệu theo mẫu
- **SymPy:** tính toán chuyên ngành dùng cho số học, đại số, toán rời rạc và vật lý lượng tử

6

Thư viện trong python

- **Bokeh**: tạo các ô tương tác, biểu đồ tổng quan trên nền web, rất hiệu quả khi tương tác với dữ liệu lớn và trực tuyến.
- **Seaborn**: dựa trên matplotlib, công cụ trực quan hóa (visualization) dữ liệu thống kê.
- **Blaze**: gói dựa trên **Numpy** và **Pandas** hướng đến dữ liệu phân tán hoặc truyền phát =>công cụ mạnh tạo diễn thị về dữ liệu lớn (big data).

7

Numpy

- NumPy là thư viện bổ sung của python, do không có sẵn, ta phải cài đặt:

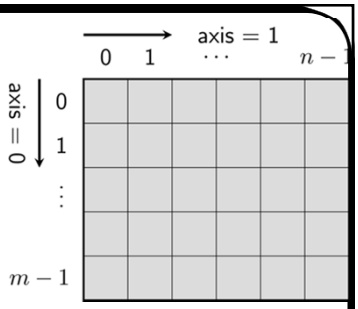
pip install numpy

- Để kiểm tra xem hệ thống đã cài numpy hay không => thử import gói xem có bị báo lỗi hay không:

import numpy as np

8

Numpy



- Đối tượng chính của NumPy là các mảng đa chiều đồng nhất (homogeneous multidimention 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
  - Có đến 24 kiểu số khác nhau
  - Kiểu **ndarray** là lớp chính xử lý dữ liệu mảng nhiều chiều

9

Ví dụ: Khởi tạo mảng

```
import numpy as np
x = np.range(3.0)                  # mảng [0. 1. 2.]
a = np.zeros((2, 2))               # mảng 2 x 2 toàn số 0
b = np.ones((1, 2))                #  mảng 1x2 toàn số 1
c = np.full((3, 2, 2), 9)          # mảng 3 x 2 x 2 toàn số 9
d = np.eye(2)                      # ma trận đơn vị 2 x 2
e = np.random.random(3,2)          # mảng 3 x 2 ngẫu nhiên [0,1)
# mảng 2 x 3 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')"
```

10

Ví dụ: Tạo và truy cập mảng

```
import numpy as np

a = np.array([21, 32,83])      # 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] = 1
print(a)
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"
```

11

Ví dụ:

a. Viết chương trình tạo ma trận 5 x5 có chứa toàn số 0.
b. Viết chương trình tạo ma trận 4 x 3 có chứa toàn số 1.

Hướng dẫn:

 # Câu a:

```
import numpy as np
a = np.zeros((5,5))
print(a)
```

 # Câu b:

(Download bộ thư viện nump xuống)

Truy cập theo chỉ số (slicing)

```
import numpy as np

# mảng 3 x 4
a = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
# mảng 2 x 2 trích xuất từ a, dòng 0+1, cột 1+2
b= a[:2, 1:3]
# 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]
print(a[0, 1])          # in "77"
```

13

Ví dụ: Truy cập theo chỉ số

```
row_r1 = a[1, :]
row_r2 = a[1:2, :]
print(row_r1, row_r1.shape)
print(row_r2, row_r2.shape)

col_r1 = a[:, 1]
col_r2 = a[:, 1:2]
print(col_r1, col_r1.shape)
print(col_r2, col_r2.shape)
```

```
# mảng 1 chiều độ dài 4
#  mảng 2 chiều 1 x 4
# in ra "[5 6 7 8] (4,)"
# in ra "[[5 6 7 8]] (1, 4)"

# mảng 1 chiều độ dài 3
# mảng 2 chiều 3 x 1
# in ra "[ 2  6 10] (3,)"
# in ra "[[ 2] [6] [10]] (3,1)"
```

14

Ví dụ: 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

# phép nhân/chia thực hiện theo cặp phần tử của x và y
```

15

Ví dụ: Nhân ma trận và nghịch đảo

```
import numpy as np

x = np.array([[1, 2], [3, 4]])
y = np.array([[5, 6],[7, 8]])

v = np.array([9, 10])
w= np.array([11, 12])

print(v.dot(w))          # tương tự print(np.dot(v, w))
print(x.dot(v))          # tương tự print(np.dot(x, v))
print(x.dot(y))          # tương tự print(np.dot(x, y))
print(np.linalg.inv(x))  # tính và in nghịch đảo của x
```

16

Ví dụ: Ma trận chuyển vị

```
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]]"

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)
```

17

Ví dụ: Đọc dữ liệu từ file

```
from io import StringIO
import numpy as np
c = StringIO("0 1\n2 3")
x = np.loadtxt(c)          # array([[ 0.,  1.],
                           #        [ 2.,  3.]])
d = StringIO("M 21 72\nF 35 58")
y = np.loadtxt(d, dtype={'names': ('gender', 'age', 'weight'),
'formats': ('S1', 'i4', 'f4')})
print(y)                   # [('M', 21, 72.0), ('F', 35, 58.0)]
```

18

Ví dụ: Đọc dữ liệu từ file

```
import numpyas np

x = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12 ]])
v = np.array([1, 0, 1])
y = x + v
print(y)        # in ra "[[ 2  2  4]
                #        [ 5  5  7]
                #        [ 8  8 10]
                #        [11 11 13]]"
```

Ví dụ: Tạo mảng ngẫu nhiên

```
np.random.random(3, 2)  # mảng 3 x 2 ngẫu nhiên trong [0,1)
np.random.randn()       # một số sinh theo phân phối chuẩn
np.random.randn(3)      #  mảng 3 số theo phân phối chuẩn
np.random.randn(3, 4)   #  mảng 3 x 4 theo phân phối chuẩn

# mảng 2 x 4 gồm các số nguyên trong [3,15)
np.random.randint(3, 15, ( 2 ,4))
# Tạo một dãy hoán vị ngẫu nhiên của dãy (0, 1, 2,…,19)
np.random.permutation(20)
```

20

Ví dụ: Tính tổng theo các trục

```
import numpy as np
x = np.array([[1, 2], [3, 4]]) # [1,2]
                               # [3,4]

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

21

Ví dụ: Các hàm thống kê

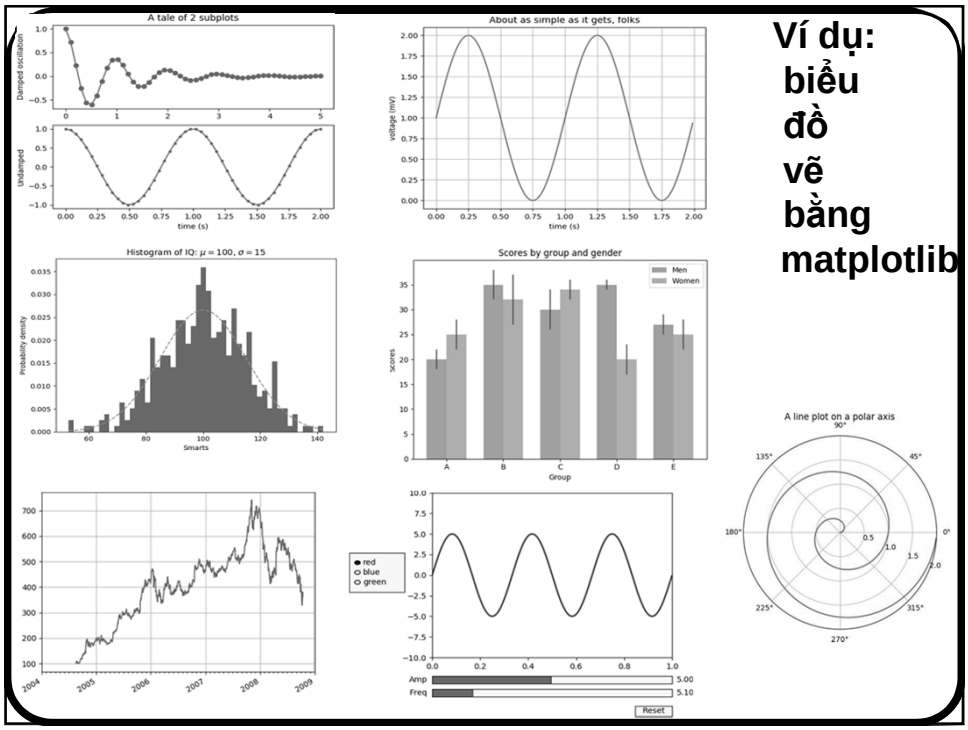
```
import numpy as np
    a = np.random.randn(3, 4)
# tính trung bình của cả ma trận a
    print(np.mean(a))
# tính trung vị của cột đầu tiên
    print(np.median(a[:,0]))
# tính độ lệch chuẩn của từng dòng
    print(a.std(axis=0))
# tính phương sai của từng cột
    print(a.var(axis=1))
```

22

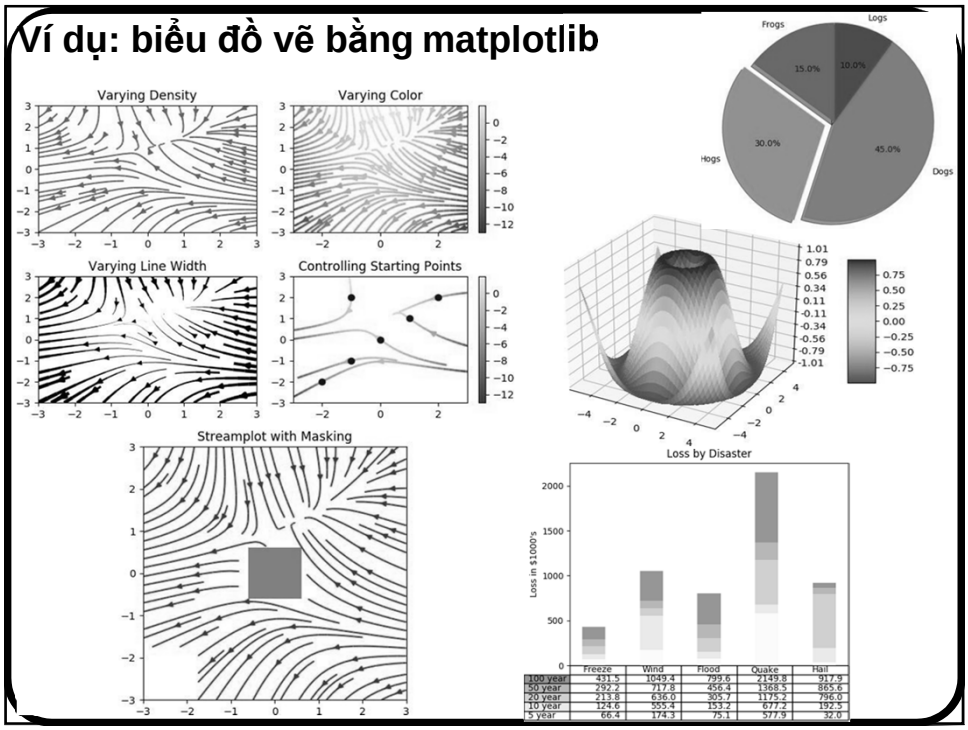
Thư viện matplotlib

Giới thiệu matplotlib

- Thư viện chuyên vẽ biểu đồ => mở rộng từ **numpy**
- Hỗ trợ nhiều loại biểu đồ: biểu đồ dòng, đường, tần suất (histograms), phổ, tương quan, errorcharts, scat erplots,…
- Ngoài API liên quan đến vẽ biểu đồ, matplotlib còn bao gồm một số interface: Object-Oriented API, The Scripting Interface (pyplot), The MATLAB Interface (pylab).
- Cài đặt: "**pip install matplotlib**"

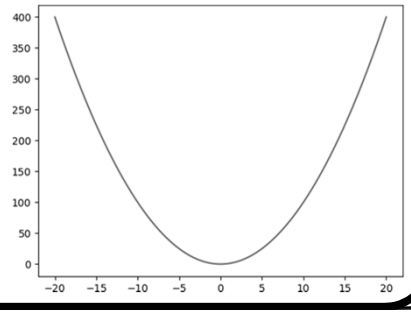


Ví dụ: biểu đồ vẽ bằng matplotlib



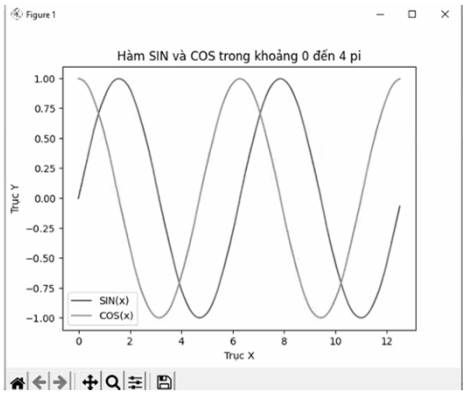
Ví dụ: vẽ biểu đồ $y = x^2$

```
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 * x

# vẽ biểu đồ quan hệ giữa x và y
plt.plot(x, y)
# hiển thị biểu đồ
plt.show()
```



Ví dụ: vẽ biểu đồ hình sin và cos

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

x = np.arange(0, 4 *np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

plt. plot( x,y_sin)
plt. plot( x,y_cos)
plt.xlabel('Trục X')
plt.ylabel('Trục Y')
plt.title('Hàm SIN và COS trong khoảng 0 đến 4 pi')
plt.legend(['SIN(x)', 'COS(x)'])
plt.show()
```



Các bước vẽ biểu đồ

- Đã có sẵn dữ liệu
- **Bước 1: Chọn loại biểu đồ phù hợp**
  - Tùy thuộc vào loại dữ liệu
  - Tùy thuộc vào mục đích sử dụng của người dùng
- **Bước 2: Thiết lập các thông số cho biểu đồ**
  - Thông số của các trục, ý nghĩa, tỉ lệ chia,…
  - Góc nhìn, mẫu tô, màu và các chi tiết khác
  - Các thông tin bổ sung
- **Bước 3: Vẽ biểu đồ**
- **Bước 4: Lưu ra file**

Line plot

- Biểu đồ thể hiện quan hệ giữa X và Y
- Cú pháp:
  - plot([x], y, [fmt], data=None,**kwargs)
  - plot([x], y, [fmt], [x2], y2, [fmt2], ..., **kwargs)

o "fmt" là quy cách vẽ đường

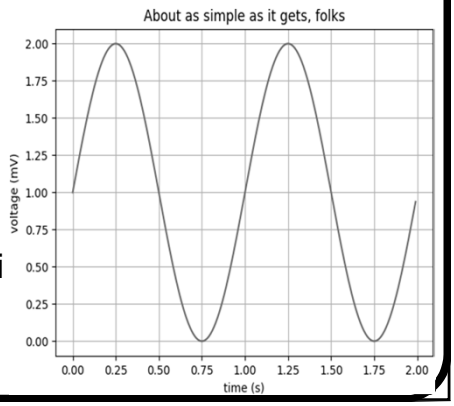
o "data" là nhãn của dữ liệu

o **kwargs: tham số vẽ đường

o Vẽ nhiều lần trên một biểu đồ

o Kết quả trả về là một list các đối

o tượng Line 2D

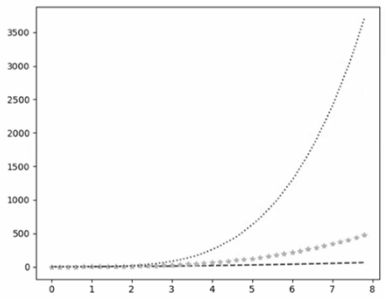


Line plot

- "fmt" gồm 3 phần **fmt = '[color][marker][line]'**
- [color]: 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
- [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

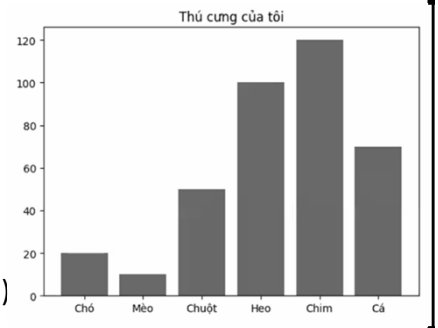
Ví dụ:

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

# chia đoạn 0-8 thành các bước 0.2
 x = np.arange(0., 8., 0.2)

# Vẽ 3 đường:
# màu đỏ nét đứt: y = x^2
# màu xanh dương, đánh dấu ô vuông: y = x^3
# màu xanh lá, đánh dấu tam giác: y = x^4
plt.plot(x,x**2,'r^', x, x**3, 'bp', x, x**4, 'g*')
 plt.show()
```



Ví dụ:

Cho tập dữ liệu thú cưng:
Data={'Chó':20, 'Mèo':10, 'Chuột':50, 'Heo':100, 'Chim':120, 'Cá':70}
Vẽ biểu đồ dạng cột thể hiện số lượng các thú cưng trên.

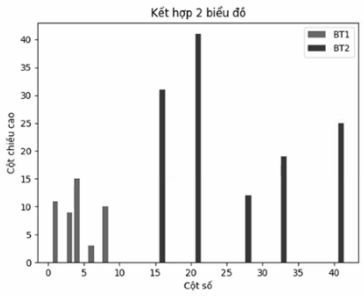


Hướng dẫn:

```
import matplotlib.pyplot as plt
Data={'Chó':20, 'Mèo':10, 'Chuột':50,
'Heo':100, 'Chim':120, 'Cá':70}
plt.bar(range(len(Data)),list(Data.values()))
plt.xticks(range(len(Data)),Data.keys())
plt.title('Thú cưng của tôi')
plt.show()
```

Ví dụ:

Giả sử có hai biểu đồ có giá trị:
BT1=([1,4,6,8,3],[11,15,3,10,9])
BT2=([21,41,16,28,33],[41,25,31,12,19])
Vẽ biểu đồ để ghép 2 biểu đồ trên thành 1 biểu đồ

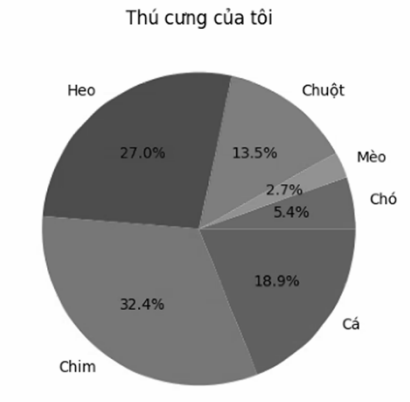


Hướng dẫn:

```
import matplotlib.pyplot as plt

plt.bar([1,4,6,8,3],[11,15,3,10,9], label ='BT1')
plt.bar([21,41,16,28,33],[41,25,31,12,19], label='BT2', color ='r')
plt.legend()
plt. xlabel('Cột số')
plt. ylabel('Cột chiều cao')
plt.title('Kết nối 2 biểu đồ')
plt.show()
```

Ví dụ

Cho tập dữ liệu thú cưng:
Data={‘Chó’:20, ‘Mèo’:10, ‘Chuột’:50, ‘Heo’:100, ‘Chim’:120, ‘Cá’:70}
Vẽ biểu đồ dạng tròn (bánh) thể hiện số lượng các thú cưng trên.



Hướng dẫn:

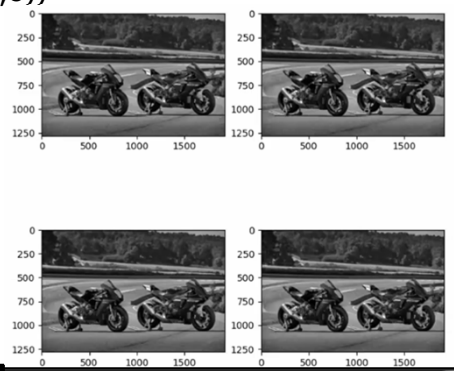
```
import numpy as np
import matplotlib.pyplot as mp
data = { 'Chó': 20, 'Mèo': 10,
"Chuột": 50, "Heo": 100, "Chim": 120,
   "Cá": 70}
mp.pie(list(data.values()), labels=list(data.keys()), autopct='%1.1f%%')
mp.title("Thú cưng của tôi")
mp.show()
```

Ví dụ

Đọc một ảnh từ máy và hiện lên màn hình, sau đó lưu ra file

Hướng dẫn:

```
import matplotlib.pyplot as plt
import matplotlib.image as Ima
image =Ima.imread(“abc.jpg”)
fig, axs=plt.subplots(2,2, figsize=(8,8))
axs[0,0].imshow(image)
axs[0,1].imshow(image)
axs[1,0].imshow(image)
axs[1,1].imshow(image)
# lưu file
plt.savefig(‘a.png’)
plt.savefig(‘a.pdf’)
plt.show()
```

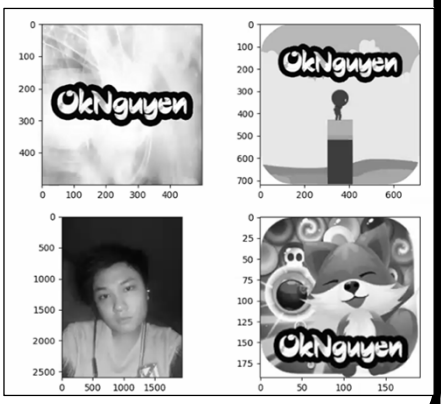


Ví dụ

Đọc bốn ảnh từ máy và hiện lên màn hình, sau đó lưu ra file

Hướng dẫn:

```
import matplotlib.pyplot as plt
import matplotlib.image as img
image1 = img.imread("abc1.png")
image2 = img.imread("abc2.png")
image3 = img.imread("abc3.jpg")
image4 = img.imread("abc4.png")
fig, axs = plt.subplots(2, 2, figsize = (8, 8))
axs[0, 0].imshow(image1)
axs[0, 1].imshow(image2)
axs[1, 0].imshow(image3)
axs[1, 1].imshow(image4)
# lưu file
plt.savefig('a.png')
plt.savefig('a.pdf')
plt.show()
```



Thư viện Pandas

Giới thiệu Pandas

- "pandas" là thư viện mở rộng từ **numpy**, chuyên để xử lý dữ liệu cấu trúc dạng bảng
- "pandas" => dạng số nhiều của "panel data"
- Đọc dữ liệu từ **nhiều định dạng**
- Liên kết dữ liệu và tích hợp xử lý dữ liệu bị thiếu
- Xoay, chuyển đổi chiều của dữ liệu dễ dàng
- Tách, đánh chỉ mục và chia nhỏ các tập dữ liệu lớn dựa trên nhãn
- Có thể nhóm dữ liệu cho các mục đích hợp nhất và chuyển đổi
- Lọc dữ liệu và thực hiện query trên dữ liệu
- Xử lý dữ liệu chuỗi thời gian và lấy mẫu

Câu trúc dữ liệu trong pandas

- Dữ liệu của pandas có 3 cấu trúc chính:
  - **Series (loạt)**: cấu trúc 1 chiều, mảng dữ liệu đồng nhất
  - **Dataframe (khung)**: cấu trúc 2 chiều, dữ liệu trên các cột là đồng nhất (có phần giống như table trong SQL, nhưng với các dòng được đặt tên)
  - **Panel (bảng):** cấu trúc 3 chiều, có thể xem như một tập các dataframe với thông tin bổ sung
- Dữ liệu series gần giống kiểu array trong numpy, nhưng có điểm khác biệt quan trọng:
  - Chấp nhận dữ liệu thiếu (**NaN: không xác định**)

Cấu trúc dataframe

- Dữ liệu 2 chiều
- Các cột có tên
- Dữ liệu trên cột là đồng nhất
- Các dòng có thể có tên
- Có thể có ô thiếu dữ liệu

	country	population	area	capital
BR	Brazil	200	8515767	Brasilia
RU	Russia	144	17098242	Moscow
IN	India	1252	3287590	New Delhi
CH	China	1357	9596961	Beijing
SA	South Africa	55	1221037	Pretoria

Cấu trúc panel

- Dữ liệu 3 chiều
- Một tập các data frame
- Các data frame có cấu trúc tương đồng
- Có thể có các thông tin bổ sung cho từng dataframe

		Open	Close
Major	Minor		
3/31/2015	IBM	23.602	132.903
	APPL	421.412	212.665
	CVX	568.055	409.201
	BHP	487.414	515.413
4/30/2015	IBM	150.868	457.895
	APPL	204.729	957.179
	CVX	90.679	888.687
	BHP	831.527	714.202
5/31/2015	IBM	788.582	922.422
	APPL	329.716	304.964
	CVX	36.578	981.508
	BHP	313.848	882.293

Tạo dữ liệu series

```
import pandas as pd
import numpy as np
S= pd.Series(np.random.randint(500, size =10))
print(S)
print(S.index)
print(S.values)
```

```
File  Edit  Shell  Debug  Options  Window  Help
Python 3.9.5 (tags/v3.9.5:0a7dcbd, May  3 2021, 17:27:52)
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more
>>>
========== RESTART: D:\Documents\Python\OkNguyen.
0    103
1    180
2    227
3    307
4    168
5    103
6    399
7     14
8    134
9    286
dtype: int32
RangeIndex(start=0, stop=10, step=1)
[103 180 227 307 168 103 399  14 134 286]
>>>
```

Ví dụ

```
import pandas as pd
import numpy as np
chi_so = [“Chó", “Mèo", “Chuột", “Heo“, “Chim”, “Cá”]
gia_ tri = [20, 10, 50,100, 120, 70]
S= pd.Series(gia_tri, index=chi_so)
print(S)
print(S.index)
print(S.values)
```

```
IDLE Shell 3.9.5
File  Edit  Shell  Debug  Options  Window  Help
Python 3.9.5 (tags/v3.9.5:0a7dcbd, May  3 2021, 17:27:52) [MSC v.1928 64 bit
D64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
========== RESTART: D:\Documents\Python\OkNguyen.py ==========
Chó       20
Mèo       10
Chuột     50
Heo      100
Chim     120
Cá        70
dtype: int64
Index(['Chó', 'Mèo', 'Chuột', 'Heo', 'Chim', 'Cá'], dtype='object')
[ 20  10  50 100 120  70]
>>>
```

Ví dụ:

Cho tập dữ liệu sau:
Data={ “Xuat Sac”: 1, “Tot”:2, “Kha”:3, “Trung Bình”: 4, “Yeu”:5, “Kem”:6” }
Viết code xuất tập Data trên ra màn hình.

Phép toán trên series

- Thực hiện phéptoán trên series nhưsau:
  - Nếu là phép toán giữa 2 series, thì cácgiá trị cùng chỉ số sẽ thực hiện phép toán với nhau, trường hợp không có giá trị ở cả 2 seriesthì trả về NaN
  - Nếu là phép toán giữa series và 1 số, thì thực hiện phép toán trên số đó với tất cả các giá trịtrong series

Một số phương thức

- S.axes: trả về danh sách các chỉ mục của S
- S.dtype: trả về kiểu dữ liệu các phần tử của S
- S.empty: trả về True nếu S rỗng
- S.ndim: trả về số chiều của S (1)
- S.size: trả về số phần tử của S
- S.values: trả về list các phần tử của S
- S.head(n): trả về n phần tử đầu tiên của S
- S.tail(n): trả về n phần tử cuối cùng của S

Khởi tạo dataframe

pandas.DataFrame(data,index,columns,dtype,copy)

- Trong đó:
  - 'data': nhận giá trị từ nhiều kiểu khác nhau như list, dictionary, ndarray, series,… và cả các DataFrame khác
  - 'index': nhãn chỉ mục hàng của dataframe
  - 'columns': nhãn chỉ mục cột của dataframe
  - 'dtype': kiểu dữ liệu cho mỗi cột
  - 'copy' : nhận giá trị True/False để chỉ dữ liệu có được copy sang vùng nhớ mới không, mặc định là False

Ví dụ:

Tạo dataframe từ list

```
A = [['Anh',1],["Em",2],["Chị",200]]
df = pd.DataFrame(A)
print(df)
```

Tạo dataframe từ dictionary các list

```
crimes_rates = {
  "Year":[1960,1961,1962,1963,1964],
  "Population":[179323175,182992000,185771000,188483000,191141
  000],  "Total":[3384200,3488000,3752200,4109500,4564600],
  "Violent":[288460,289390,301510,316970,364220]
}
crimes_dataframe = pd.DataFrame(crimes_rates)
print(crimes_dataframe)
```

Đọc dữ liệu từ file.csv

```
import pandas as pd
d = pd.read_csv("a.csv")
print(d)
```

Xem dữ liệu

```
import pandas as pd
import matplotlib.pyplot as plt
d = pd.read_csv("a.csv", index_col = 0)
 d.describe()
```

Kết hợp giữa pandas và matplotlib

```
import pandas as pd
import matplotlib.pyplot as plt
d = pd.read_csv("a.csv", index_col = 0)
d.area.plot(kind='bar')
plt.show()
```