Chương 10. Projects

Nội dung của chương

Pygame

Data visualization

Django

Trực quan hóa dữ liệu

Sản sinh dữ liệu

Cài đặt Matplotlib

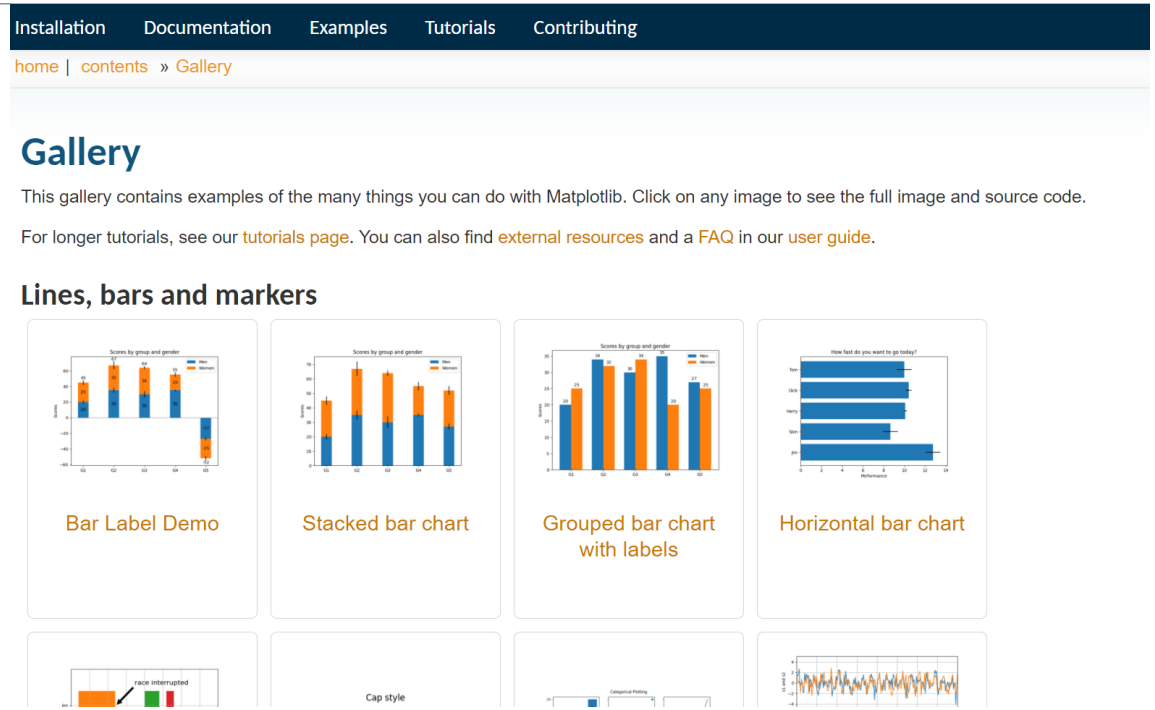
Dùng lệnh pip hoặc pip3

$ python -m pip install --user matplotlib

$ python3 -m pip install --user matplotlib

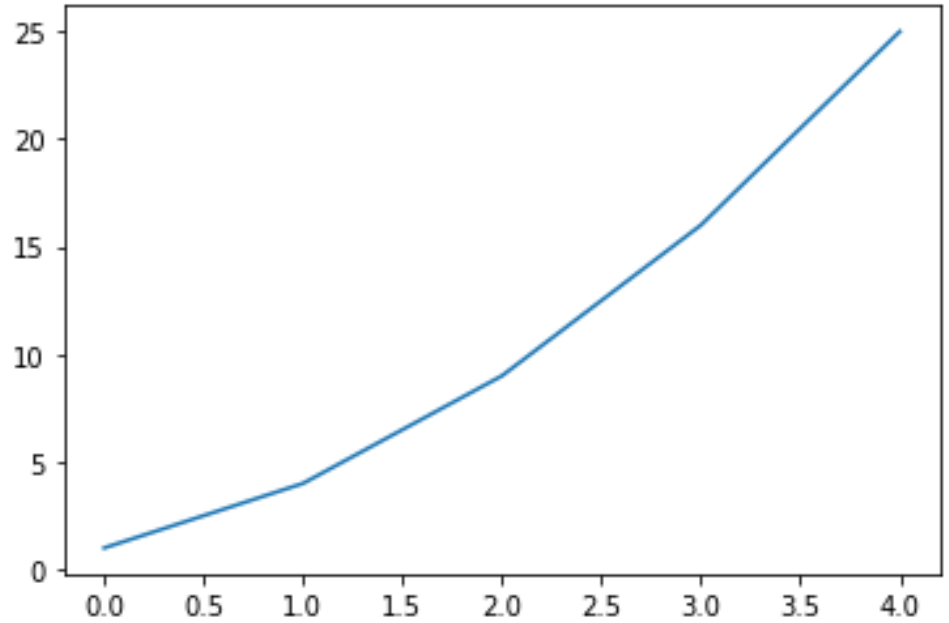
Thư viện hình ảnh:

https://matplotlib.org/gallery/.



Vẽ biểu đồ một đường đơn giản

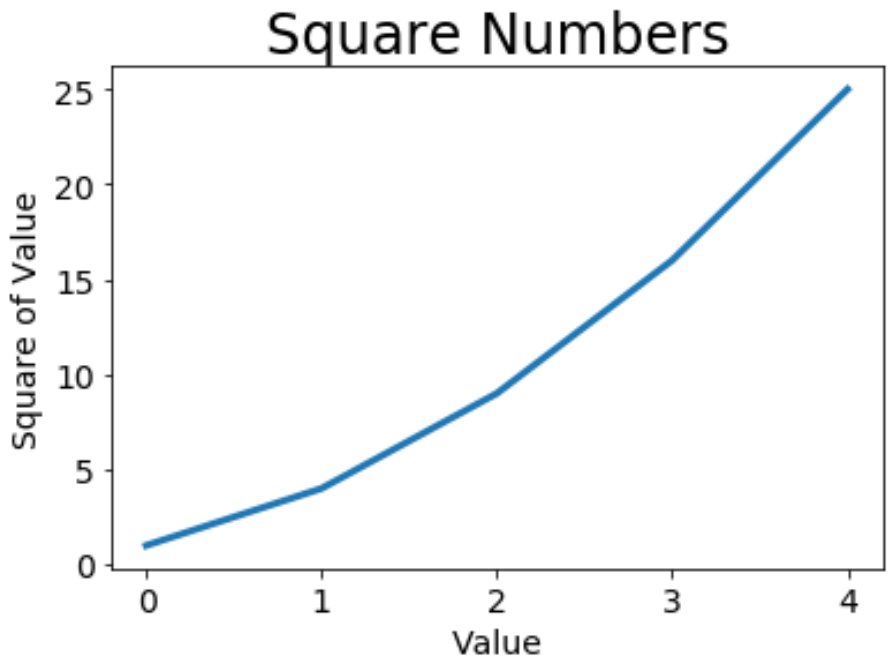
```
import matplotlib.pyplot as plt
squares = [1, 4, 9, 16, 25]
fig, ax = plt.subplots()
ax.plot(squares)
plt.show()
```



- Squares: danh sách các điểm
- fig: toàn bộ các hình vẽ
- ax: một biểu đồ
- hàm plot: vẽ biểu đồ với dữ liệu được cung cấp
- plt.show(): mở cửa sổ hiển thị

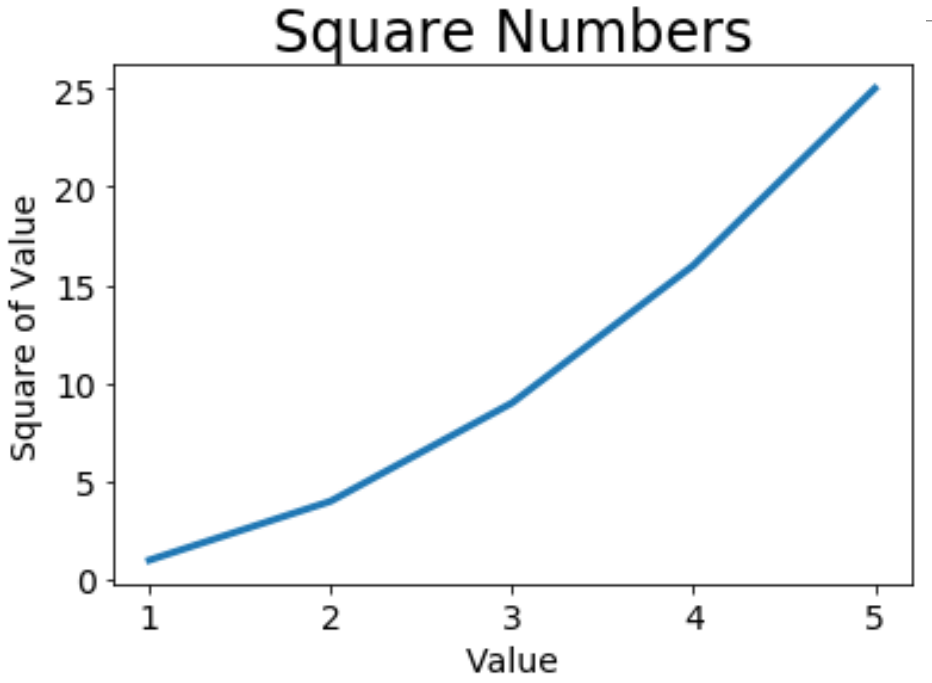
Thay đổi loại nhãn và độ dầy đường line

```
import matplotlib.pyplot as plt
squares = [1, 4, 9, 16, 25]
fig, ax = plt.subplots()
ax.plot(squares, linewidth=3)
# Set chart title and label axes.
ax.set_title("Square Numbers", fontsize=24)
ax.set_xlabel("Value", fontsize=14)
ax.set_ylabel("Square of Value", fontsize=14)
# Set size of tick labels.
ax.tick_params(axis='both', labelsize=14)
plt.show()
```



Chỉnh sửa các điểm

```
import matplotlib.pyplot as plt
input_values = [1, 2, 3, 4, 5]
squares = [1, 4, 9, 16, 25]
fig, ax = plt.subplots()
ax.plot(input_values, squares, linewidth=3)
# Set chart title and label axes.
--snip—
```

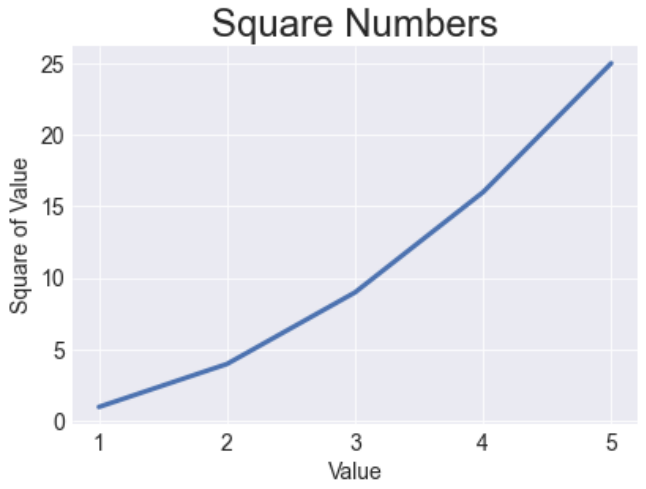


Sử dụng các định kiểu có sẵn (Build-in styles)

import matplotlib.pyplot as plt
print(plt.style.available)

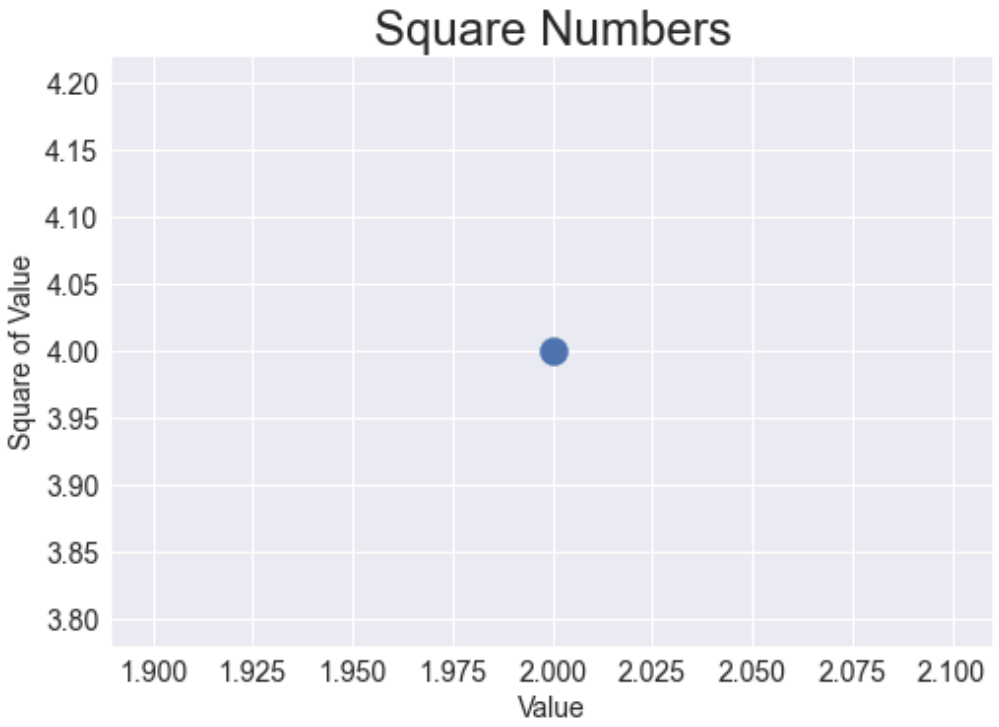
```
['Solarize_Light2', '_classic_test_patch', 'bmh', 'classic', 'dark_background', 'fast',
'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn', 'seaborn-bright', 'seaborn-colorblind',
'seaborn-dark', 'seaborn-dark-palette', 'seaborn-darkgrid', 'seaborn-deep', 'seaborn-muted',
'seaborn-notebook', 'seaborn-paper', 'seaborn-pastel', 'seaborn-poster', 'seaborn-talk', 'seaborn-
ticks', 'seaborn-white', 'seaborn-whitegrid', 'tableau-colorblind10']
```

```
import matplotlib.pyplot as plt

input_values = [1, 2, 3, 4, 5]

squares = [1, 4, 9, 16, 25]

plt.style.use('seaborn')

fig, ax = plt.subplots()

--snip--
```



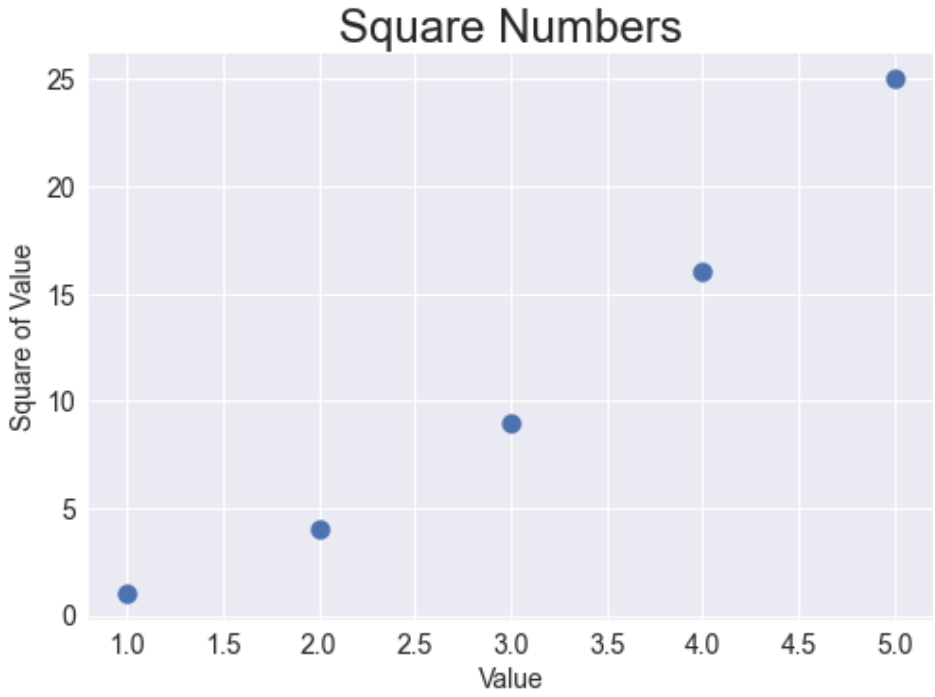
Vẽ biểu đồ và định kiểu riêng các điểm với scatter()

```
import matplotlib.pyplot as plt
plt.style.use('seaborn')
fig, ax = plt.subplots()
ax.scatter(2, 4, s=200)
# Set chart title and label axes.
ax.set_title("Square Numbers", fontsize=24)
ax.set_xlabel("Value", fontsize=14)
ax.set_ylabel("Square of Value", fontsize=14)
# Set size of tick labels.
ax.tick_params(axis='both', which='major', labelsize=14)
plt.show()
```



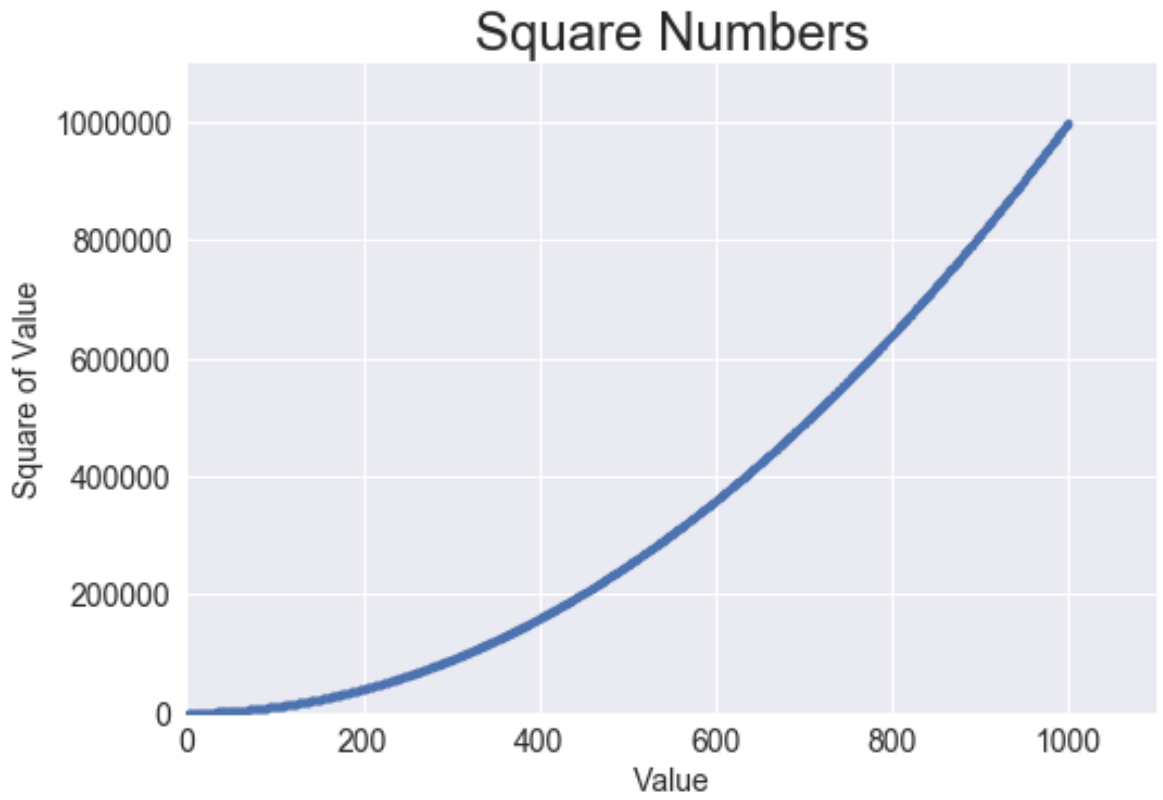
Vẽ tập điểm với scatter()

```
import matplotlib.pyplot as plt
x_values = [1, 2, 3, 4, 5]
y_values = [1, 4, 9, 16, 25]
plt.style.use('seaborn')
fig, ax = plt.subplots()
ax.scatter(x_values, y_values, s=100)
```



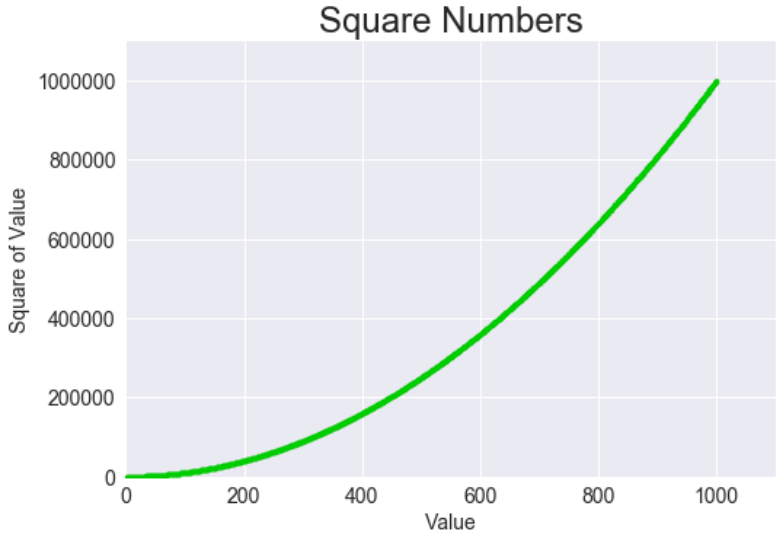
Tính toán dữ liệu tự động

```
import matplotlib.pyplot as plt
x_values = range(1, 1001)
y_values = [x**2 for x in x_values]
plt.style.use('seaborn')
fig, ax = plt.subplots()
ax.scatter(x_values, y_values, s=10)
# Set chart title and label axes.
--snip--
# Set the range for each axis.
ax.axis([0, 1100, 0, 1100000])
plt.show()
```

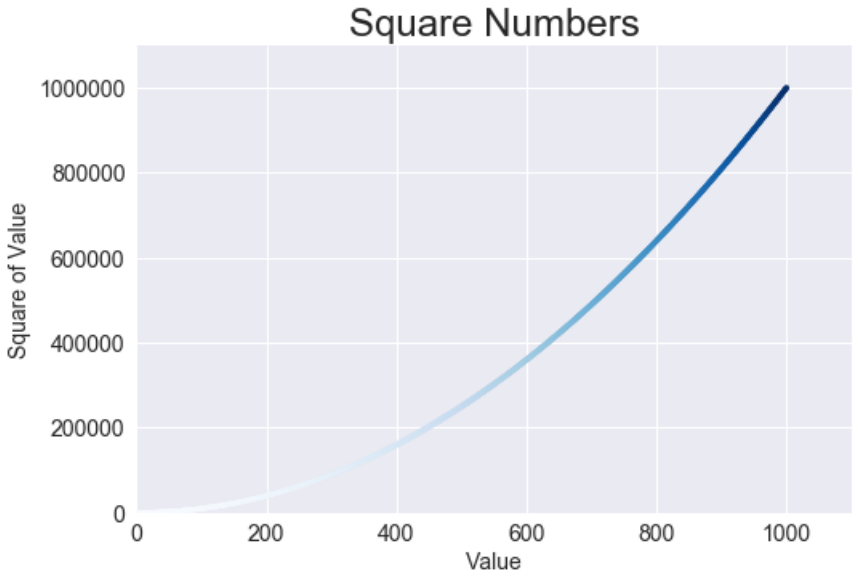


Tự định nghĩa mầu

```
ax.scatter(x_values, y_values, c=(0, 0.8, 0), s=10)
```




```
ax.scatter(x_values, y_values, c=y_values, cmap=plt.cm.Blues, s=10)
```



Lưu biểu đồ tự động

```
plt.savefig('squares_plot.png',
bbox_inches='tight')
```

Tải dữ liệu về

Chúng ta sẽ sử dụng mô-đun csv của Python để xử lý dữ liệu thời tiết được lưu trữ trong định dạng CSV (các giá trị được phân tách bằng dấu phẩy) và phân tích nhiệt độ cao và thấp theo thời gian ở hai vị trí khác nhau.

Sau đó, chúng ta sẽ sử dụng Matplotlib để tạo biểu đồ dựa trên dữ liệu đã tải xuống của chúng ta để hiển thị các biến thể về nhiệt độ trong hai môi trường khác nhau

Ở phần sau của chương, chúng ta sẽ sử dụng mô-đun json để truy cập dữ liệu động đất được lưu trữ ở định dạng JSON và sử dụng Plotly để vẽ bản đồ thế giới hiển thị vị trí và cường độ của các trận động đất gần đây.

Đến cuối mục này, ta sẽ chuẩn bị làm việc với các loại và định dạng tập dữ liệu, đồng thời ta sẽ hiểu sâu hơn về cách để xây dựng các hình dung phức tạp

Phân tích định dạng của tệp CSV

```
import csv
filename = 'data/sitka_weather_07-2018_simple.csv'
with open(filename) as f:
        reader = csv.reader(f)
        header_row = next(reader)
print(header_row)
```

Module csv chứa hàm next() trả về dòng tiếp theo của tệp khi ta truyền vào đối tượng reader. Phần xử lý này, ta gọi next() chỉ một lần để trả về dòng đầu tiên của tệp, dòng này chứa header của tệp.

```
['STATION', 'NAME', 'DATE', 'PRCP', 'TAVG', 'TMAX', 'TMIN']
```

Header rows

```
import csv
filename = 'data/sitka_weather_07-2018_simple.csv'
with open(filename) as f:
    reader=csv.reader(f)
    header_row=next(reader)
for index, column_header in enumerate(header_row):
    print(index, column_header)
```

```
0 STATION
1 NAME
2 DATE
3 PRCP
4 TAVG
5 TMAX
6 TMIN
```

Ở đây, chúng ta thấy rằng ngày và nhiệt độ cao của chúng được lưu trữ trong cột 2 và 5. Để khám phá dữ liệu này, ta sẽ xử lý từng hàng dữ liệu trong sitka_weather_07-2018_simple.csv và trích xuất các giá trị với các chỉ mục 2 và 5.

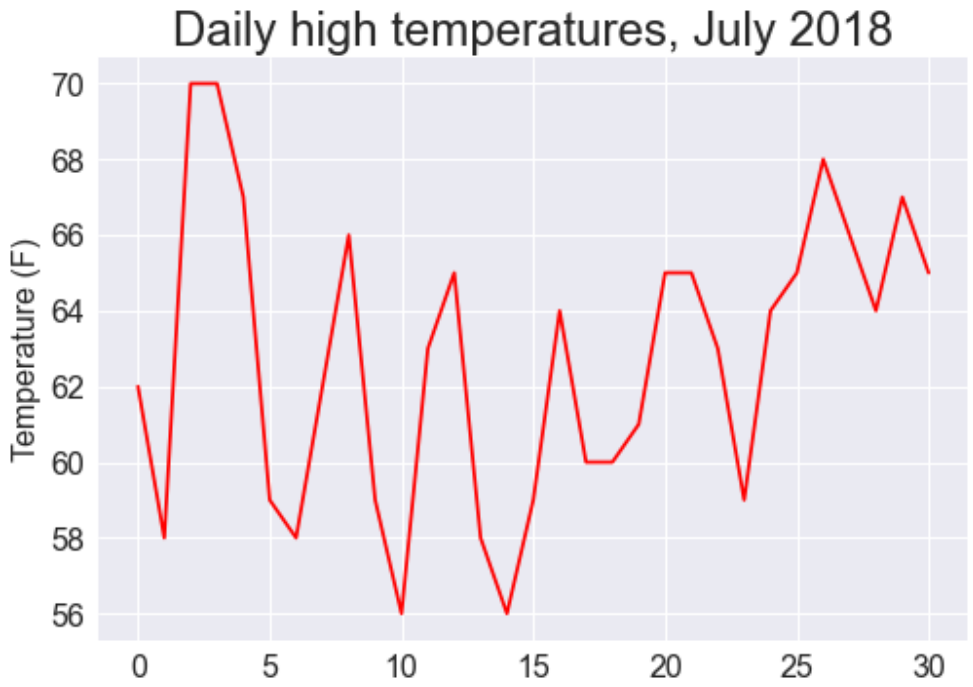
Trích xuất và đọc dữ liệu

```
import csv
filename = 'data/sitka_weather_07-2018_simple.csv'
with open(filename) as f:
    reader=csv.reader(f)
    header_row=next(reader)
    print(header_row)
    for index, column_header in enumerate(header_row):
        print(index, column_header)
    #get hight temperature from file
    highs=[]
    for row in reader:
        high=int(row[5])
        highs.append(high)
print(highs)
```

```
['STATION', 'NAME', 'DATE', 'PRCP', 'TAVG',
'TMAX', 'TMIN']
0 STATION
1 NAME
2 DATE
3 PRCP
4 TAVG
5 TMAX
6 TMIN
[62, 58, 70, 70, 67, 59, 58, 62, 66, 59, 56,
63, 65, 58, 56, 59, 64, 60, 60, 61, 65, 65,
63, 59, 64, 65, 68, 66, 64, 67, 65]
```

Lập đồ thị dữ liệu biểu đồ nhiệt độ

```
# Plot the high temperatures.
plt.style.use('seaborn')
fig, ax = plt.subplots()
ax.plot(highs, c='red')

# Format plot.
plt.title("Daily high temperatures, July 2018", fontsize=24)
plt.xlabel('', fontsize=16)
plt.ylabel("Temperature (F)", fontsize=16)
plt.tick_params(axis='both', which='major', labelsize=16)
```



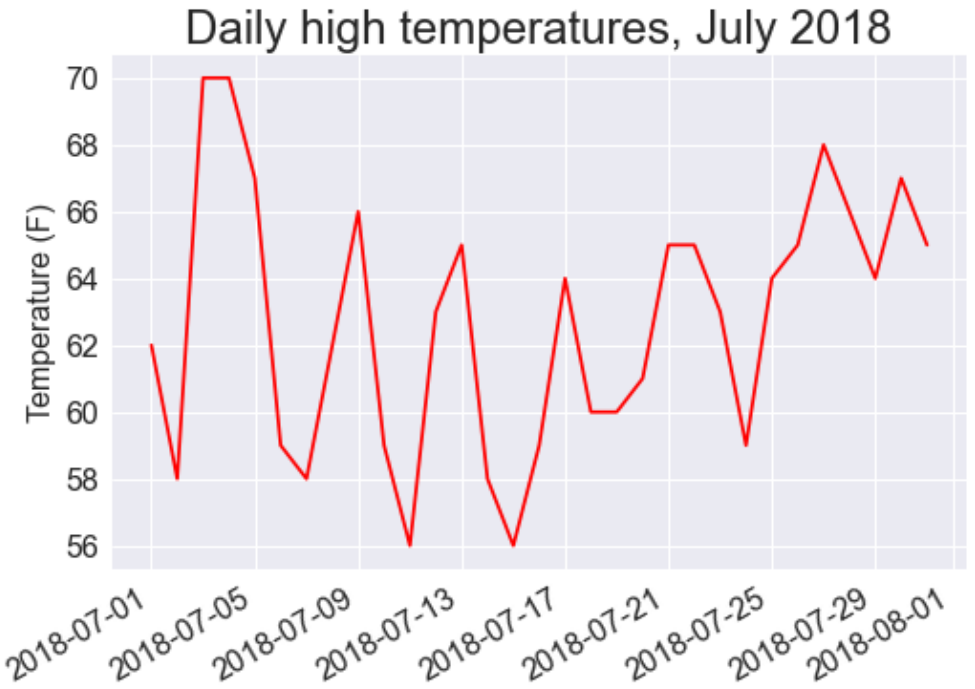
Module datetime

```
>>> from datetime import datetime
>>> first_date = datetime.strptime('2018-07-01', '%Y-%m-%d')
>>> print(first_date)
2018-07-01 00:00:00
```

Tham số	Ý nghĩa
%A	Weekday name, such as Monday
%B	Month name, such as January
%m	Month, as a number (01 to 12)
%d	Day of the month, as a number (01 to 31)
%Y	Four-digit year, such as 2019
%y	Two-digit year, such as 19
%H	Hour, in 24-hour format (00 to 23)
%I	Hour, in 12-hour format (01 to 12)
%p	am or pm
%M	Minutes (00 to 59)
%S	Seconds (00 to 61)

Vẽ biểu đồ thời gian

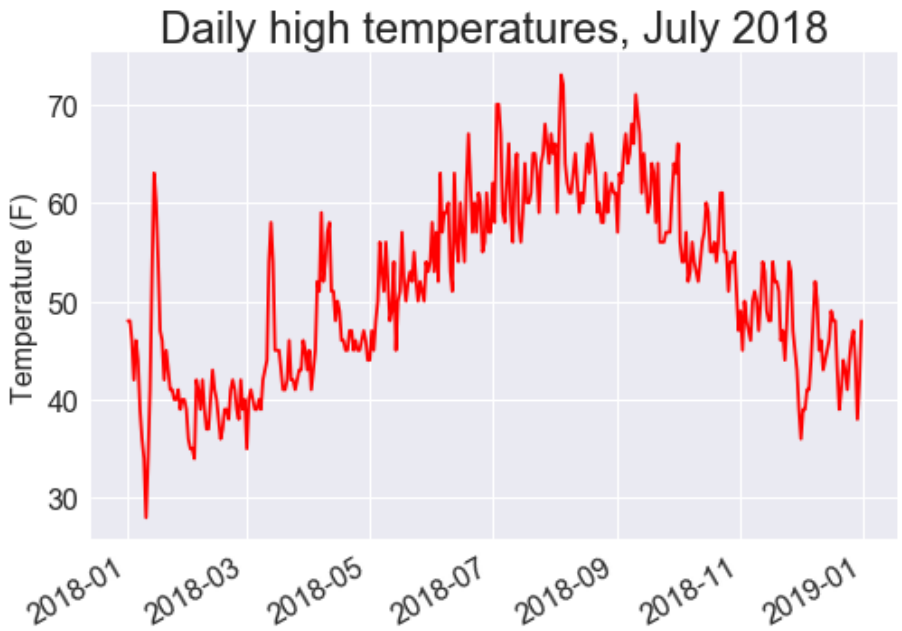
```
with open(filename) as f:
    dates=[]
    highs=[]
    for row in reader:
        current_date = datetime.strptime(row[2],'%Y-%m-%d')
        dates.append(current_date)
        high=int(row[5])

fig, ax = plt.subplots()
ax.plot(dates,highs, c='red')
# Format plot.
plt.title("Daily high temperatures, July 2018", fontsize=24)
plt.xlabel('', fontsize=16)
fig.autofmt_xdate()
```



hàm fig.autofmt_xdate() để vẽ các nhãn của trục x theo đường chéo để tránh việc các nhãn này đè lên nhau

Vẽ một khung thời gian dài hơn

```
--snip--
filename = 'data/sitka_weather_2018_simple.csv'
with open(filename) as f:
--snip--
# Format plot.
plt.title("Daily high temperatures - 2018", fontsize=24)
plt.xlabel('', fontsize=16)
--snip
```

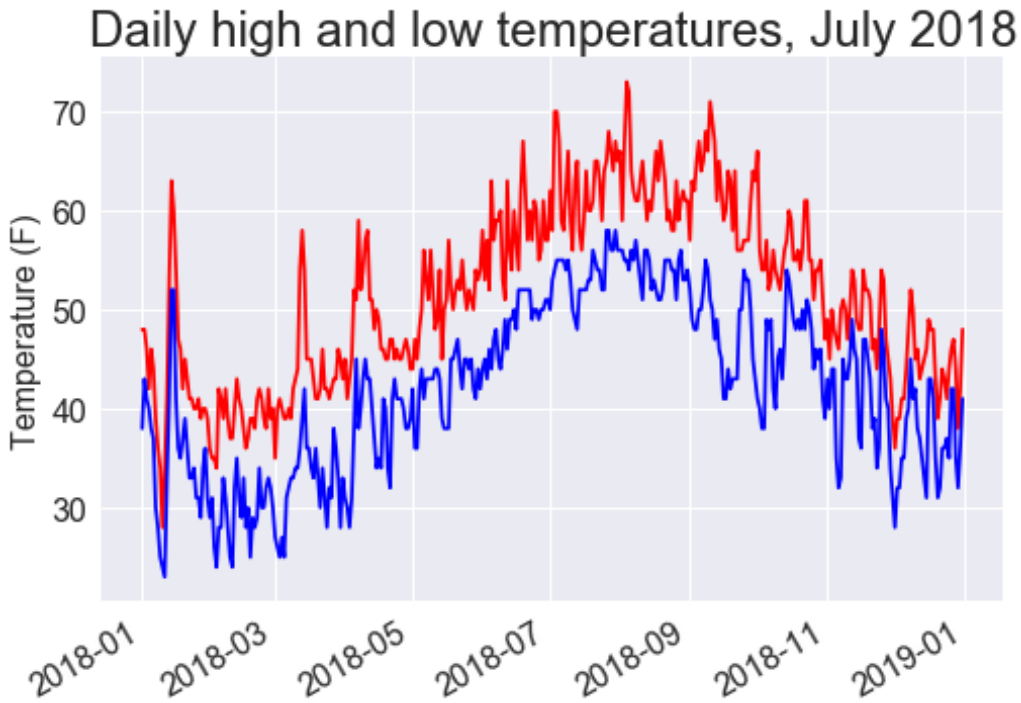


Có hai phần thay đổi so với đoạn mã ở trên. Phần thứ nhất, chúng ta thay đổi tệp dữ liệu từ dữ liệu của tháng 7 thành dữ liệu của cả năm (file sitka_weather_2018_simple.csv). Phần thứ hai, chúng ta thay đổi tiêu đề của biểu đồ từ July -2018 thành 2018 (bỏ July).

Vẽ biểu đồ chuỗi dữ liệu thứ hai

```
with open(filename) as f:
    reader=csv.reader(f)
    header_row=next(reader)
    print(header_row)

#    for index, column_header in enumerate(header_row):
#        print(index, column_header)
    #get hight temperature from file
    dates=[]
    highs=[]
    lows=[]
    for row in reader:
        current_date = datetime.strptime(row[2],'%Y-%m-%d')
        high=int(row[5])
        highs.append(high)
        dates.append(current_date)
        low=int(row[6])
        lows.append(low)
# Plot the high temperatures.
```

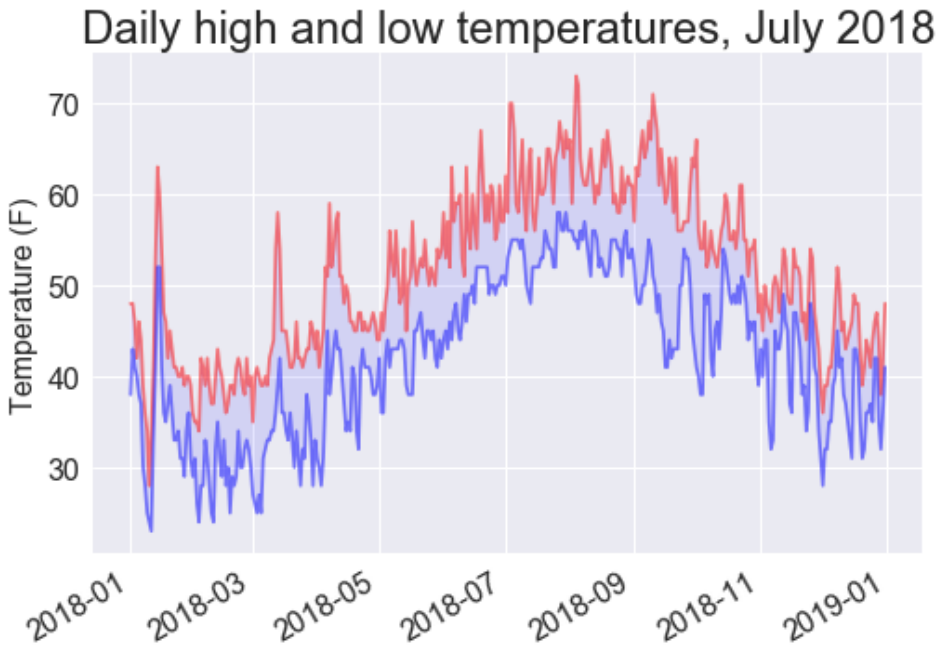
```
plt.style.use('seaborn')
fig, ax = plt.subplots()
ax.plot(dates,highs, c='red')
ax.plot(dates,lows, c='blue')

plt.title("Daily high and low temperatures, July 2018", fontsize=24)
```



Tô bóng một vùng trong biểu đồ

sử dụng phương thức fill_between(), phương thức này nhận một chuỗi giá trị x và hai chuỗi giá trị y và lấp đầy khoảng trống giữa hai chuỗi giá trị y:

```
--snip--
# Plot the high and low temperatures.
plt.style.use('seaborn')
fig, ax = plt.subplots()
ax.plot(dates, highs, c='red', alpha=0.5)
ax.plot(dates, lows, c='blue', alpha=0.5)
plt.fill_between(dates, highs, lows,
facecolor='blue', alpha=0.1)
--snip—
```



Định dạng JSON

Trong phần này, ta sẽ tải xuống tập dữ liệu đại diện cho tất cả các trận động đất đã xảy ra trên thế giới trong tháng trước.

Sau đó, ta sẽ tạo một bản đồ hiển thị vị trí của những trận động đất này và mức độ quan trọng mỗi trận động đất đó.

Tải dữ liệu về động đất

Sao chép fle eq_1_day_m1.json vào thư mục đang lưu trữ dữ liệu cho các chương trình.

Các trận động đất được phân loại theo độ lớn của chúng trên thang độ Richter.

Chương trình này bao gồm dữ liệu cho tất cả các trận động đất với độ lớn M1 hoặc lớn hơn diễn ra trong 24 giờ qua (tại thời điểm của văn bản này).

Kiểm tra dữ liệu JSON

eq_1_day_m1.json


```
{"type":"FeatureCollection","metadata":{"generated":1550361461000,...
{"type":"Feature","properties":{"mag":1.2,"place":"11km NNE of Nor...
{"type":"Feature","properties":{"mag":4.3,"place":"69km NNW of Ayn...
{"type":"Feature","properties":{"mag":3.6,"place":"126km SSE of Co...
{"type":"Feature","properties":{"mag":2.1,"place":"21km NNW of Teh...
{"type":"Feature","properties":{"mag":4,"place":"57km SSW of Kakto...
--snip—
```

Sử dụng Web API

Git và Github

Yêu cầu dữ liệu sử dụng lệnh gọi API

https://api.github.com/search/repositories?q=language:python&sort=stars


```
{
  "total_count": 7485398,
  "incomplete_results": false,
  "items": [
    {
      "id": 83222441,
      "node_id": "MDEwOlJlcG9zaXRvcnk4MzIyMjQ0MQ==",
      "name": "system-design-primer",
      "full_name": "donnemartin/system-design-primer",
      "private": false,
      "owner": {
        "login": "donnemartin",
        "id": 5458997,
        "node_id": "MDQ6VXNlcjU0NTg5OTc=",
        "avatar_url": "https://avatars.githubusercontent.com/u/5458997?v=4",
        "gravatar_id": "",
        "url": "https://api.github.com/users/donnemartin",
        "html_url": "https://github.com/donnemartin",
        "followers_url": "https://api.github.com/users/donnemartin/followers",
        "following_url": "https://api.github.com/users/donnemartin/following{/other_user}",
        "gists_url": "https://api.github.com/users/donnemartin/gists{/gist_id}",
        "starred_url": "https://api.github.com/users/donnemartin/starred{/owner}{/repo}",
        "subscriptions_url": "https://api.github.com/users/donnemartin/subscriptions",
        "organizations_url": "https://api.github.com/users/donnemartin/orgs",
        "repos_url": "https://api.github.com/users/donnemartin/repos",
        "events_url": "https://api.github.com/users/donnemartin/events{/privacy}",
        "received_events_url": "https://api.github.com/users/donnemartin/received_events",
        "type": "User",
        "site_admin": false
      },
```

Cài đặt Requests

```
$ python -m pip install --user requests
```

Gói Requests cho phép một chương trình Python dễ dàng yêu cầu thông tin từ một trang web và kiểm tra phản hồi.

Dòng trên yêu cầu Python chạy mô-đun pip và cài đặt gói Requests vào thiết lập Python của người dùng hiện tại. Nếu ta sử dụng python3 hoặc một lệnh khác khi chạy chương trình hoặc cài đặt gói, hãy đảm bảo bạn sử dụng cùng một lệnh như trên.

Xử lý API Response

```
import requests
# Make an API call and store the response.
url = 'https://api.github.com/search/repositories?q=language:python&sort=stars'
headers = {'Accept': 'application/vnd.github.v3+json'}
r = requests.get(url, headers=headers)
print(f"Status code: {r.status_code}")

# Store API response in a variable.
response_dict = r.json()
# Process results.
print(response_dict.keys())
```

```
Status code: 200
dict_keys(['total_count', 'incomplete_results', 'items'])
```

Làm việc với từ điển Response

```
# Process results.
print(response_dict.keys())
print(f"Total repositories: {response_dict['total_count']}")
# Explore information about the repositories.
repo_dicts = response_dict['items']
print(f"Repositories returned: {len(repo_dicts)}")
# Examine the first repository.
repo_dict = repo_dicts[0]
print(f"\nKeys: {len(repo_dict)}")
for key in sorted(repo_dict.keys()):
    print(key)
```

```
Total repositories: 7486266
Repositories returned: 30

Keys: 74
archive_url
archived
assignees_url
blobs_url
-snip-
```

Tổng hợp các kho hàng đầu

Giám sát giới hạn tốc độ API

Trực quan kho dữ liệu sử dụng Ploty

Tinh chỉnh biểu đồ Plotly