Trường Đại học Sư phạm Kỹ thuật TP.HCM
Khoa Điện - Điện tử

Môn học: Cơ Sở Khoa Học Dữ Liệu

CHAPTER 6:
Data Loading, Storage, and File Formats

Giảng Viên: TS. Nguyễn Mạnh Hùng

Sinh viên: Võ Văn Khánh – 20145322

Trần Quốc Huy – 20139076

Vũ Quang Huy – 20139077

Trần Minh Khôi – 20139079

Nguyễn Đình Đức Thọ – 20139092

Nguyễn An Minh Triết – 20139093

Thành phố Hồ Chí Minh, tháng 11 năm 2022

Mục lục

Các công cụ trong cuốn sách này sẽ ít được sử dụng nếu bạn không thể dễ dàng nhập và xuất dữ liệu bằng Python. Ta sẽ tập trung vào đầu vào và đầu ra với các đối tượng pandas, mặc dù tất nhiên có rất nhiều công cụ trong các thư viện khác để hỗ trợ quá trình này. Ví dụ, NumPy có tính năng lưu trữ và tải dữ liệu nhị phân cấp thấp nhưng cực nhanh, bao gồm hỗ trợ cho mảng ánh xạ bộ nhớ.

Đầu vào và đầu ra thường thuộc một số danh mục chính: đọc tệp văn bản và các định dạng trên đĩa khác hiệu quả hơn, tải dữ liệu từ cơ sở dữ liệu và tương tác với các nguồn công việc mạng như API web.

1 Reading and Writing Data in Text Format

Python đã trở thành một ngôn ngữ được ưa chuộng để đọc văn bản và tệp vì cú pháp đơn giản của nó để tương tác với các tệp, cấu trúc dữ liệu trực quan và các tính năng tiện lợi như bộ đóng gói và giải nén. Pandas có một số chức năng để đọc dữ liệu dạng bảng dưới dạng đối tượng DataFrame. Bảng 6-1 có một bản tóm tắt của tất cả chúng, mặc dù `read_csv` và `read_table` có khả năng là những thứ bạn sẽ sử dụng nhiều nhất.

Function	Description
`read_csv`	Load delimited data from a file, URL, or file-like object. Use comma as default delimiter
`read_table`	Load delimited data from a file, URL, or file-like object. Use tab (`'\t'`) as default delimiter
`read_fwf`	Read data in fixed-width column format (that is, no delimiters)
`read_clipboard`	Version of `read_table` that reads data from the clipboard. Useful for converting tables from web pages

Hình 1: Bảng 6-1.Các hàm phân tích cú pháp trong pandas

Ta sẽ cung cấp tổng quan về cơ chế hoạt động của các chức năng này, nhằm chuyển đổi dữ liệu văn bản thành DataFrame. Các tùy chọn cho các chức năng này thuộc một số loại:

• Indexing: có thể coi một hoặc nhiều cột là DataFrame được trả về và có lấy tên cột từ tệp, người dùng hay không.

• Loại suy luận và chuyển đổi dữ liệu: điều này bao gồm chuyển đổi giá trị do người dùng xác định và danh sách tùy chỉnh các dấu giá trị bị thiếu.

• Phân tích cú pháp ngày giờ: bao gồm khả năng kết hợp, bao gồm kết hợp thông tin ngày và giờ trải rộng trên nhiều cột vào một cột duy nhất trong kết quả.

• Lặp lại: hỗ trợ lặp qua các khối tệp rất lớn.

• Các vấn đề về dữ liệu không rõ ràng: bỏ qua hàng hoặc chân trang, nhận xét hoặc các vấn đề nhỏ khác như dữ liệu số với hàng nghìn phân tách bằng dấu phẩy.

Loại suy luận là một trong những tính năng quan trọng hơn của các hàm này; điều đó có nghĩa là bạn không phải chỉ định cột nào là số, số nguyên, boolean hoặc chuỗi. Tuy nhiên, việc xử lý ngày tháng và các loại tùy chỉnh khác đòi hỏi nhiều nỗ lực hơn. Hãy bắt đầu với một tệp văn bản nhỏ được phân tách bằng dấu phẩy (CSV):

```
In [846]:
import numpy as np
import pandas as pd
np.random.seed(12345)
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(10, 6))
np.set_printoptions(precision=4, suppress=True)

with open('examples/ex1.csv') as f:
    for line in f:
        print(line)
a,b,c,d,message
1,2,3,4,hello
5,6,7,8,world
9,10,11,12,foo
```

Vì đây là `comma_delimited`, nên ta có thể sử dụng `read_csv` để đọc nó trong DataFrame:

```
In [847]: df = pd.read_csv('examples/ex1.csv')
In [848]: df
Out[848]:
             a   b   c   d  message
        0    1   2   3   4    hello
        1    5   6   7   8    world
        2    9  10  11  12      foo
```

Ta cũng có thể đã sử dụng `read_table` và chỉ định dấu phân cách:

```
In [849]: pd.read_table('examples/ex1.csv', sep=',')
Out[849]:
               a   b   c   d  message
          0    1   2   3   4    hello
```

```
          1   5   6   7   8    world
          2   9  10  11  12      foo
```

Ở đây ta sử dụng lệnh Unix cat shell để in nội dung thô của tệp vào màn hình. Nếu bạn đang sử dụng Windows, bạn có thể sử dụng type thay vì cat để đạt được hiệu quả tương tự. Một tệp sẽ không phải lúc nào cũng có hàng header. Hãy xem xét tệp này:

```
In [850]:
with open('examples/ex2.csv') as f:
    for line in f:
        print(line)
1,2,3,4,hello
5,6,7,8,world
9,10,11,12,foo
```

Để đọc điều này, bạn có một vài tùy chọn. Bạn có thể cho phép pandas gán mặc định tên cột hoặc bạn có thể tự chỉ định tên:

```
In [851]: pd.read_csv('examples/ex2.csv', header=None
Out[851]:
             X.1  X.2  X.3  X.4     X.5
          0    1    2    3    4   hello
          1    5    6    7    8   world
          2    9   10   11   12     foo
In [852]: pd.read_csv('examples/ex2.csv', names=['a', 'b', 'c', 'd',
    'message'])
Out[852]:
                a    b    c    d  message
          0    1    2    3    4    hello
          1    5    6    7    8    world
          2    9   10   11   12      foo
```

Giả sử ta muốn cột message là chỉ mục của DataFrame được trả về. Ta có thể cho biết cột ở mục 4 được đặt tên là 'message' bằng cách sử dụng `index_col` :

```
In [853]: names = ['a', 'b', 'c', 'd', 'message']
In [854]: pd.read_csv('examples/ex2.csv', names=names, index_col='message')
```

```
Out[854]:
          a   b   c   d
   message
   hello   1   2   3   4
   world   5   6   7   8
   foo     9  10  11  12
```

Trong trường hợp ta muốn tạo chỉ mục phân cấp từ nhiều cột, chỉ cần chuyển danh sách số cột hoặc tên:

```
In [855]:
with open('examples/csv_mindex.csv') as f:
   for line in f:
      print(line)
key1,key2,value1,value2
one,a,1,2
one,b,3,4
one,c,5,6
one,d,7,8
two,a,9,10
two,b,11,12
two,c,13,14
two,d,15,16

In [856]: parsed = pd.read_csv('examples/csv_mindex.csv', index_col
['key1','key2'])

In [857]: parsed
Out[857]:
            value1 value2
   key1 key2
   one   a      1      2
         b      3      4
         c      5      6
         d      7      8
   two   a      9     10
```

```
b     11    12
c     13    14
d     15    16
```

Trong một số trường hợp, bảng có thể không có dấu phân cách cố định, sử dụng khoảng trắng hoặc một số cách khác mẫu để tách các trường. Trong những trường hợp này, ta có thể chuyển một biểu thức chính quy dưới dạng dấu phân cách `read_table`. Hãy xem xét một tệp văn bản trông như thế này:

```
In [858]: list(open('examples/ex3.txt'))
Out[858]:
   ['            A         B         C\n',
    'aaa  -0.264438 -1.026059 -0.619500\n',
    'bbb   0.927272  0.302904 -0.032399\n',
    'ccc  -0.264273 -0.386314 -0.217601\n',
    'ddd  -0.871858   348382  1.100491\n']
```

Mặc dù ta có thể thực hiện các munging bằng tay, nhưng trong trường hợp này, các trường được phân tách bằng một biến số lượng khoảng trắng. Điều này có thể được thể hiện bằng biểu thức chính quy s+, vì vậy ta có:

```
In [859]: result = pd.read_table('examples/ex3.txt', sep='\s+')
In [860]: result
Out[860]:
                A         B         C
   aaa  -0.264438 -1.026059 -0.619500
   bbb   0.927272  0.302904 -0.032399
   ccc  -0.264273 -0.386314 -0.217601
   ddd  -0.871858 -0.348382  1.100491
```

Bởi vì có ít hơn một tên cột so với số hàng dữ liệu, `read_table` suy ra rằng cột đầu tiên phải là chỉ mục của DataFrame trong trường hợp đặc biệt này.

Các hàm phân tích cú pháp có nhiều đối số bổ sung để giúp ta xử lý rộng nhiều định dạng tệp ngoại lệ xảy ra (xem Bảng 6-2). Ví dụ: ta có thể bỏ qua hàng đầu tiên, thứ ba và thứ tư của tệp có skiprows:

```
In [861]: with open('examples/ex4.csv') as f:
              for line in f:
                  print(line)


# hey!
a,b,c,d,message
# just wanted to make things more difficult for you
# who reads CSV files with computers, anyway?
1,2,3,4,hello
5,6,7,8,world
9,10,11,12,foo
In [862]: pd.read_csv('examples/ex4.csv', skiprows=[0, 2, 3])
Out[862]:
       a   b   c   d   message
   0   1   2   3   4     hello
   1   5   6   7   8     world
   2   9  10  11  12       foo
```

Xử lý các giá trị bị thiếu là một phần quan trọng và thường xuyên có sắc thái của phân tích cú pháp tệp quá trình. Dữ liệu bị thiếu thường không có mặt (chuỗi trống) hoặc được đánh dấu bởi một số giá trị trọng tâm. Theo mặc định, pandas sử dụng một tập hợp các lính canh thường xảy ra, chẳng hạn như NA, -1.#IND và NULL:

```
In [863]:
with open('examples/ex5.csv') as f:
    for line in f:
        print(line)
something,a,b,c,d,message
one,1,2,3,4,NA
two,5,6,,8,world
three,9,10,11,12,foo
In [864]: result = pd.read_csv('examples/ex5.csv')


In [865]: result
Out[865]:
```

```
      something a   b   c   d message
0         one  1   2   3   4     NaN
1         two  5   6 NaN   8   world
2       three  9  10  11  12     foo
In [866]: pd.isnull(result)
Out[866]:
      something     a      b      c      d  message
0       False  False  False  False  False     True
1       False  False  False   True  False    False
2       False  False  False  False  False    False
```

Option na_values có thể lấy một danh sách hoặc tập hợp các chuỗi để xem xét các giá trị bị thiếu:

```
In [867]: result = pd.read_csv('examples/ex5.csv', na_values=['NULL'])
In [868]: result
Out[868]:
      something a   b   c   d message
0         one  1   2   3   4     NaN
1         two  5   6 NaN   8   world
2       three  9  10  11  12     foo
```

Các lính canh NA khác nhau có thể được chỉ định cho mỗi cột trong một lệnh:

```
In [869]: sentinels = {'message': ['foo', 'NA'], 'something': ['two']}
In [870]: pd.read_csv('examples/ex5.csv', na_values=sentinels)
Out[870]:
    something a   b   c   d message
0         one  1   2   3   4     NaN
1         two  5   6 NaN   8   world
2       three  9  10  11  12     NaN
```

Argument	Description
path	String indicating filesystem location, URL, or file-like object
sep or delimiter	Character sequence or regular expression to use to split fields in each row
header	Row number to use as column names. Defaults to 0 (first row), but should be None if there is no header row
index_col	Column numbers or names to use as the row index in the result. Can be a single name/number or a list of them for a hierarchical index
names	List of column names for result, combine with header=None
skiprows	Number of rows at beginning of file to ignore or list of row numbers (starting from 0) to skip
na_values	Sequence of values to replace with NA
comment	Character or characters to split comments off the end of lines
parse_dates	Attempt to parse data to datetime; False by default. If True, will attempt to parse all columns. Otherwise can specify a list of column numbers or name to parse. If element of list is tuple or list, will combine multiple columns together and parse to date (for example if date/time split across two columns)
keep_date_col	If joining columns to parse date, drop the joined columns. Default True
converters	Dict containing column number of name mapping to functions. For example {'foo': f} would apply the function f to all values in the 'foo' column
dayfirst	When parsing potentially ambiguous dates, treat as international format (e.g. 7/6/2012 -> June 7, 2012). Default False
date_parser	Function to use to parse dates
nrows	Number of rows to read from beginning of file
iterator	Return a TextParser object for reading file piecemeal
chunksize	For iteration, size of file chunks
skip_footer	Number of lines to ignore at end of file
verbose	Print various parser output information, like the number of missing values placed in non-numeric columns
encoding	Text encoding for unicode. For example 'utf-8' for UTF-8 encoded text
squeeze	If the parsed data only contains one column return a Series
thousands	Separator for thousands, e.g. ',' or '.'

Hình 2: Bảng 6-2. Đối số hàm `read_csv` / `read_table`

1.1 Reading Text Files in Pieces

Khi xử lý các tệp rất lớn hoặc tìm ra bộ đối số phù hợp chính xác xử lý một tệp lớn, ta có thể chỉ muốn đọc trong một phần nhỏ của tệp hoặc lặp lại qua các phần nhỏ hơn của tệp.

```
In [871]:
pd.options.display.max_rows = 10
result = pd.read_csv('examples/ex6.csv')
```

```
In [872]: result
Out[872]:
   <class 'pandas.core.frame.DataFrame'>
   Int64Index: 10000 entries, 0 to 9999
   Data columns:
   one       10000 non-null values
   two       10000 non-null values
   three     10000 non-null values
   four      10000 non-null values
   key       10000 non-null values
   dtypes: float64(4), object(1)
```

Nếu ta chỉ muốn đọc ra số lượng nhỏ các hàng (tránh đọc toàn bộ tệp), chỉ định rằng với nrows:

```
In [873]: pd.read_csv('examples/ex6.csv', nrows=5)
Out[873]:
            one       two     three      four  key
   0   0.467976 -0.038649 -0.295344 -1.824726    L
   1  -0.358893  1.404453  0.704965 -0.200638    B
   2  -0.501840  0.659254 -0.421691 -0.057688    G
   3   0.204886  1.074134  1.388361 -0.982404    R
   4   0.354628 -0.133116  0.283763 -0.837063    Q
```

Để đọc một tệp theo từng phần, hãy chỉ định một đoạn dưới dạng một số hàng:

```
In [874]: chunker = pd.read_csv('examples/ex6.csv', chunksize=1000)
In [875]: chunker
Out[875]: <pandas.io.parsers.TextParser at 0x8398150>
```

Đối tượng TextParser được trả về bởi read_csv cho phép bạn lặp qua các phần của tập tin theo chunksize. Ví dụ: chúng ta có thể lặp lại trên ex6.csv, tổng hợp giá trị được tính trong cột ”key” như sau:

```
chunker = pd.read_csv('examples/ex6.csv', chunksize=1000)


tot = Series([], dtype = 'float64')
```

```
for piece in chunker:
    tot = tot.add(piece['key'].value_counts(), fill_value=0)


tot = tot.order(ascending=False)
```

Và ta có:

```
In [877]: tot[:10]
Out[877]:
    E 368
    X 364
    L 346
    O 343
    Q 340
    M 338
    J 337
    F 335
    K 334
    H 330
```

TextParser cũng được trang bị một phương pháp get_chunk cho phép bạn đọc các phần có kích thước tùy ý.

1.2 Writing Data Out to Text Format

Dữ liệu cũng có thể xuất sang định dạng được phân tách. Hãy xem xét một trong các tệp CSV đã đọc ở trên:

```
In [878]: data = pd.read_csv('ex5.csv')
In [879]: data
Out[879]:
     something a   b   c   d message
    0       one 1   2   3   4     NaN
    1       two 5   6 NaN   8   world
    2     three 9  10  11  12     foo
```

Sử dụng phương thức `to_csv` của DataFrame, chúng ta có thể ghi dữ liệu ra một tệp được

phân tách bằng dấu phẩy :

```
In [880]: data.to_csv('out.csv')
In [881]: !type ch06/out.csv
Out[881]:
   ,something,a,b,c,d,message
   0,one,1,2,3.0,4,
   1,two,5,6,,8,world
   2,three,9,10,11.0,12,foo
```

Tất nhiên, các dấu phân cách khác có thể được sử dụng (ghi vào `sys.stdout` để nó chỉ in kết quả văn bản):

```
In [882]: data.to_csv(sys.stdout, sep='|')
Out[882]:
   |something|a|b|c|d|message
   0|one|1|2|3.0|4|
   1|two|5|6||8|world
   2|three|9|10|11.0|12|foo
```

Các giá trị bị thiếu xuất hiện dưới dạng các chuỗi trống trong đầu ra. Bạn có thể biểu thị chúng bằng một số giá trị trọng điểm khác:

```
In [883]: data.to_csv(sys.stdout, na_rep='NULL')
Out[883]:
   ,something,a,b,c,d,message
   0,one,1,2,3.0,4,NULL
   1,two,5,6,NULL,8,world
   2,three,9,10,11.0,12,foo
```

Nếu không có tùy chọn nào khác được chỉ định, cả nhãn hàng và nhãn cột đều được ghi lại. Cả hai thứ này đều có thể bị vô hiệu hóa khi:

```
In [884]: data.to_csv(sys.stdout, index=False, header=False)
Out[884]:
    one,1,2,3.0,4,
    two,5,6,,8,world
    three,9,10,11.0,12,foo
```

Bạn cũng có thể chỉ viết một tập hợp con của các cột theo thứ tự bạn chọn:

```
In [885]: data.to_csv(sys.stdout, index=False, cols=['a', 'b', 'c'])
Out[885]:
    a,b,c
    1,2,3.0
    5,6,
    9,10,11.0
```

Series cũng có phương thức `to_csv`:

```
In [886]: dates = pd.date_range('1/1/2000', periods=7)
In [887]: ts = Series(np.arange(7), index=dates)
In [888]: ts.to_csv('tseries.csv')
In [889]: !type tseries.csv
Out[889]
    2000-01-01 00:00:00,0
    2000-01-02 00:00:00,1
    2000-01-03 00:00:00,2
    2000-01-04 00:00:00,3
    2000-01-05 00:00:00,4
    2000-01-06 00:00:00,5
    2000-01-07 00:00:00,6
```

Với một chút lộn xộn (không có header, cột đầu tiên là chỉ mục), bạn vẫn có thể đọc phiên bản CSV của Series với `read_csv`, nhưng cũng có một phương pháp thuận tiện là `from_csv` giúp việc này đơn giản hơn một chút, đây là kết quả nhận được từ phiên bản cũ:

```
In [890]: Series.from_csv('tseries.csv', parse_dates=True)
Out[890]:
    2000-01-01 0
```

```
2000-01-02 1
2000-01-03 2
2000-01-04 3
2000-01-05 4
2000-01-06 5
2000-01-07 6
```

Và kể từ phiên bản 0.21.0, phương thức này không còn được hỗ trợ nữa. Đây là kết quả nhận được khi gõ dòng lệnh:

```
In [890]: Series.from_csv('tseries.csv', parse_dates=True)
Out[890]:
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [63], in <cell line: 1>()
----> 1 Series.from_csv('tseries.csv', parse_dates=True)


AttributeError: type object 'Series' has no attribute 'from_csv'
```

Xem thêm các tài liệu cho `to_csv` và `from_csv` trong IPython để biết thêm nhiều thông tin bổ ích khác.

1.3 Manually Working with Delimited Formats

Hầu hết các dạng dữ liệu dạng bảng đều có thể được tải từ đĩa bằng cách sử dụng các hàm như `pandas.read_table`. Tuy nhiên, trong một số trường hợp, có thể cần phải xử lý thủ công. Không có gì lạ khi nhận được một tệp có một hoặc nhiều dòng không đúng định dạng gây lỗi `read_table`. Để minh họa cho các công cụ cơ bản, hãy xem xét một tệp CSV nhỏ:

```
In [891]: !type ex7.csv
Out[891]:
"a","b","c"
"1","2","3"
"1","2","3","4"
```

Đối với bất kỳ tệp nào có dấu phân cách một ký tự, bạn có thể sử dụng mô-đun csv tích hợp của Python. Để sử dụng nó, hãy chuyển bất kỳ tệp nào đang mở hoặc đối tượng giống tệp tới

csv.reader:

```
import csv
f = open('ex7.csv')
reader = csv.reader(f)
```

Lặp lại thông qua reader, giống như một tệp tạo ra các bộ giá trị trong mỗi bộ như thế, với bất kỳ ký tự trích dẫn nào bị xóa:

```
In [893]: for line in reader:
   .....:     print(line)
Out[893]:


   ['a', 'b', 'c']
   ['1', '2', '3']
   ['1', '2', '3', '4']
```

Từ đó, bạn có thể thực hiện các thao tác cần thiết để đưa dữ liệu vào biểu mẫu mà bạn cần. Ví dụ:

```
In [894]: lines = list(csv.reader(open('ex7.csv')))
In [895]: header, values = lines[0], lines[1:]
In [896]: data_dict = {h: v for h, v in zip(header, zip(*values))}
In [897]: data_dict
Out[897]: {'a': ('1', '1'), 'b': ('2', '2'), 'c': ('3', '3')}
```

Tệp CSV có nhiều dạng khác nhau. Việc xác định một định dạng mới bằng một dấu phân cách khác, quy ước trích dẫn chuỗi hoặc dấu kết thúc dòng được thực hiện bằng cách xác định một phân lớp đơn giản của csv.Dialect:

```
class my_dialect(csv.Dialect):
       lineterminator = '\n'
       delimiter = ';'
       quotechar = '"'
       quoting = csv.QUOTE_MINIMAL
reader = csv.reader(f, dialect=my_dialect)
```

Các tham số phương ngữ CSV riêng lẻ cũng có thể được cung cấp dưới dạng từ khóa cho

csv.reader mà không phải xác định một lớp con:

```
reader = csv.reader(f, delimiter='|')
```

Các tùy chọn khả thi (thuộc tính của csv.Dialect) và chức năng của chúng có thể được tìm thấy trong Bảng 6-3.

Argument	Description
delimiter	One-character string to separate fields. Defaults to ','.
lineterminator	Line terminator for writing, defaults to '\r\n'. Reader ignores this and recognizes cross-platform line terminators.
quotechar	Quote character for fields with special characters (like a delimiter). Default is '"'.
quoting	Quoting convention. Options include csv.QUOTE_ALL (quote all fields), csv.QUOTE_MINIMAL (only fields with special characters like the delimiter), csv.QUOTE_NONNUMERIC, and csv.QUOTE_NON (no quoting). See Python's documentation for full details. Defaults to QUOTE_MINIMAL.
skipinitialspace	Ignore whitespace after each delimiter. Default False.
doublequote	How to handle quoting character inside a field. If True, it is doubled. See online documentation for full detail and behavior.
escapechar	String to escape the delimiter if quoting is set to csv.QUOTE_NONE. Disabled by default

Hình 3: Bảng 6-3. Tùy chọn phương ngữ CSV

Đối với các tệp có dấu phân cách nhiều ký tự cố định hoặc phức tạp hơn, bạn sẽ không thể sử dụng mô-đun csv. Trong những trường hợp đó, bạn sẽ phải thực hiện tách dòng và dọn dẹp khác bằng cách sử dụng phương thức tách của chuỗi hoặc phương thức biểu thức chính quy re.split.

Để ghi các tệp được phân tách theo cách thủ công, bạn có thể sử dụng csv.writer. Nó chấp nhận một đối tượng tệp mở, có thể ghi cùng các tùy chọn định dạng và ngôn ngữ như csv.reader:

```
with open('mydata.csv', 'w') as f:
    writer = csv.writer(f, dialect=my_dialect)
    writer.writerow(('one', 'two', 'three'))
    writer.writerow(('1', '2', '3'))
    writer.writerow(('4', '5', '6'))
    writer.writerow(('7', '8', '9'))
```

Và đây là kết quả nhận được sau khi ghi tệp mydata.csv:

	A	B
1	one;two;three	
2	1;2;3	
3	4;5;6	
4	7;8;9	

Hình 4: Kết quả ghi tệp mydata.csv

1.4 JSON Data

JSON (viết tắt của JavaScript Object Notation) là một trong những định dạng chuẩn để gửi dữ liệu theo yêu cầu HTTP giữa trình duyệt web và các ứng dụng khác. Nó là định dạng dữ liệu linh hoạt hơn nhiều so với dạng văn bản dạng bảng như CSV.

Ví dụ:

```
obj = """
{"name": "Wes",
 "places_lived": ["United States", "Spain", "Germany"],
 "pet": null,
 "siblings": [{"name": "Scott", "age": 25, "pet": "Zuko"},
 {"name": "Katie", "age": 33, "pet": "Cisco"}]
}
"""
```

JSON là mã Python gần như hợp lệ ngoại trừ giá trị null null và một số sắc thái khác (chẳng hạn như không cho phép đặt dấu phẩy ở cuối danh sách). Cơ bản các loại là đối tượng (dicts), mảng (danh sách), chuỗi, số, booleans và null. tất cả các khóa trong một đối tượng phải là chuỗi. Có một số thư viện Python để đọc và ghi dữ liệu JSON. Tôi sẽ sử dụng json ở đây vì nó được tích hợp vào thư viện chuẩn của Python. Đến chuyển đổi một chuỗi JSON thành dạng Python, sử dụng json.loads:

```
In [899]: import json
In [900]: result = json.loads(obj)
In [901]: result
Out[901]:
   {u'name': u'Wes',
    u'pet': None,
    u'places_lived': [u'United States', u'Spain', u'Germany'],
    u'siblings': [{u'age': 25, u'name': u'Scott', u'pet': u'Zuko'},
    {u'age': 33, u'name': u'Katie', u'pet': u'Cisco'}]}
```

json.dumps mặt khác chuyển đổi một đối tượng Python trở lại JSON:

```
In [902]: asjson = json.dumps(result)
```

Cách bạn chuyển đổi đối tượng JSON hoặc danh sách đối tượng thành DataFrame hoặc một số dữ liệu khác cấu trúc để phân tích sẽ tùy thuộc vào bạn. Thuận tiện, bạn có thể chuyển danh sách các đối tượng JSON đến hàm tạo DataFrame và chọn một tập hợp con của các trường dữ liệu:

```
In [903]: siblings = DataFrame(result['siblings'], columns=['name', 'age'])
In [904]: siblings
Out[904]:
     name age
   0 Scott 25
   1 Katie 33
```

1.5 XML and HTML: Web Scraping

Python có nhiều thư viện để đọc và ghi dữ liệu trong HTML phổ biến và các định dạng XML. lxml (http://lxml.de) là ứng dụng luôn có hiệu suất cao trong phân tích các tập tin rất lớn. lxml có nhiều programmer interface;

Ví dụ sử dụng lxml.html cho HTML, sau đó phân tích cú pháp một số XML bằng cách sử dụng lxml.objectify. Nhiều trang web cung cấp dữ liệu trong bảng HTML để xem trong trình duyệt, nhưng không có thể tải xuống dưới dạng định dạng dễ đọc bằng máy như JSON, HTML hoặc XML.

Đầu tiên trích xuất dữ liệu từ một URL, mở nó bằng urllib2 và phân tích cú pháp của stream bằng lxml như sau:

```
from lxml.html import parse
from urllib.request import urlopen,Request
url='http://finance.yahoo.com/q/op?s=AAPL+Options'
req=Request(url,headers={'User-Agent':'Mozilla/5.0'})
parsed=parse(urlopen(req))
doc=parsed.getroot()
doc
```

Sử dụng đối tượng này, ta có thể trích xuất tất cả các HTML tags thuộc một loại cụ thể, chẳng hạn như table tag chứa dữ liệu. Giả sử bạn muốn để lấy danh sách mọi URL được liên kết trong tài liệu; sử dụng phương thức findall của tài liệu gốc cùng với một XPath (một phương tiện thể hiện "truy vấn" trên tài liệu):

```
In [906]: links = doc.findall('.//a')
In [907]: links[15:20]
Out[907]:
    [<Element a at 0x2ae0a86d270>,
     <Element a at 0x2ae0a86f250>,
     <Element a at 0x2ae0a86d3b0>,
     <Element a at 0x2ae0a86d770>,
     <Element a at 0x2ae0a86d4f0>]
```

Nhưng đây là những đối tượng đại diện cho các phần tử HTML; để lấy URL và link text ta phải sử dụng phương thức get của từng phần tử (đối với URL) và phương thức text_content (đối với văn bản hiển thị):

```
In [908]: lnk = links[27]
In [909]: lnk
Out[909]: <Element a at 0x2ae0a86df40>
In [910]: lnk.get('href')
Out[910]: 'https://yahoo.uservoice.com/forums/382977'
In [911]: lnk.text_content()
Out[911]: 'Contact Us'
```

Do đó, việc nhận danh sách tất cả các URL trong tài liệu là vấn đề viết hiểu danh sách này:

```
In [912]: urls = [lnk.get('href') for lnk in doc.findall('.//a')]
In [913]: urls[-10:]
Out[913]:
    ['https://help.yahoo.com/kb/finance-for-web/SLN2310.html?locale=en_US',
 'https://help.yahoo.com/kb/finance-for-web',
 'https://yahoo.uservoice.com/forums/382977',
 'https://policies.oath.com/us/en/oath/privacy/index.html',
 'https://policies.oath.com/us/en/oath/privacy/adinfo/index.html',
 'https://legal.yahoo.com/us/en/yahoo/terms/otos/index.html',
 'https://finance.yahoo.com/sitemap/',
 'https://twitter.com/YahooFinance',
 'https://facebook.com/yahoofinance',
 'https://www.linkedin.com/company/yahoo-finance']
```

Một số các trang web làm cho nó dễ dàng hơn bằng cách cung cấp một table of interest một thuộc tính id. Đây là hai bảng chứa dữ liệu cuộc gọi và dữ liệu đặt tương ứng:

```
tables = doc.findall('.//table')
calls = tables[0]
puts = tables[1]
```

Mỗi bảng có một hàng header theo sau bởi mỗi hàng dữ liệu:

```
In [915]: rows = calls.findall('.//tr')
```

Đối với header cũng như các row data, chúng ta muốn trích xuất văn bản từ mỗi cell; Trong trường hợp của header, đây là các th cell và các td cell cho data:

```
def _unpack(row, kind='td'):
   elts = row.findall('.//%s' % kind)
   return [val.text_content() for val in elts]
```

```
In [917]: _unpack(rows[0], kind='th')
Out[917]: ['Contract Name',
 'Last Trade Date',
 'Strike',
```

```
 'Last Price',
 'Bid',
 'Ask',
 'Change',
 '% Change',
 'Volume',
 'Open Interest',
 'Implied Volatility']
In [918]: _unpack(rows[1], kind='td')
Out[918]:
    ['AAPL221202C00050000',
 '2022-11-25 12:34PM EST',
 '50.00',
 '98.00',
 '97.80',
 '98.55',
 '-1.91',
 '-1.91%',
 '2',
 '0',
 '309.38%']
```

Pandas có một lớp TextParser được sử dụng nội bộ trong read_csv và các parsing function khác để thực hiện chuyển đổi loại tự động thích hợp:

```
from pandas.io.parsers import TextParser
def parse_options_data(table):
   rows = table.findall('.//tr')
   header = _unpack(rows[0], kind='th')
   data = [_unpack(r) for r in rows[1:]]
   return TextParser(data, names=header).get_chunk()
```

Gọi parsing function này trên các đối tượng bảng lxml và nhận DataFrame:

```
In [920]: call_data = parse_options_data(calls)
In [921]: put_data = parse_options_data(puts)
In [922]: call_data[:10]
```

```
Out[922]:


     Contract Name Last Trade Date Strike Last Price Bid Ask  Change  % Change
        Volume Open Interest Implied Volatility
0  AAPL221202C00050000 2022-11-25 12:34PM EST 50.0 98.00 97.80 98.55 -1.91
    -1.91% 2 0 309.38%
1  AAPL221202C00075000 2022-10-28 1:35PM EST 75.0 81.35 72.85 73.80 0.00 - 15
    0 233.40%
2  AAPL221202C00090000 2022-11-25 11:28AM EST 90.0 57.97 57.85 58.60 10.62
    +22.43% 15 0 161.33%
3  AAPL221202C00100000 2022-11-25 11:39AM EST 100.0 47.82 47.85 48.55 -0.88
    -1.81% 80 0 126.17%
4  AAPL221202C00105000 2022-11-17 9:36AM EST 105.0 42.48 42.85 43.60 0.00 - 1
    0 116.02%
5  AAPL221202C00110000 2022-11-25 11:06AM EST 110.0 38.44 37.90 38.45 -2.91
    -7.04% 60 0 94.14%
6  AAPL221202C00115000 2022-11-25 11:58AM EST 115.0 33.40 32.90 33.50 4.90
    +17.19% 1 0 85.55%
7  AAPL221202C00120000 2022-11-25 11:05AM EST 120.0 28.49 27.90 28.65 -2.16
    -7.05% 16 0 80.08%
8  AAPL221202C00123000 2022-11-25 11:34AM EST 123.0 24.79 24.90 25.65 -3.76
    -13.17% 1 0 72.07%
9  AAPL221202C00125000 2022-11-25 12:02PM EST 125.0 23.43 22.95 23.50 -1.69
    -6.73% 9 94 62.89%
```

Parsing XML with lxml.objectify

XML là từ viết tắt của từ Extensible Markup Language là ngôn ngữ đánh dấu mở rộng. XML có chức năng truyền dữ liệu và mô tả nhiều loại dữ liệu khác nhau. Tác dụng chính của XML là đơn giản hóa việc chia sẻ dữ liệu giữa các nền tảng và các hệ thống được kết nối thông qua mạng Internet...

Ví dụ dữ liệu hiệu suất được chứa trong một tập hợp các tệp XML. Mỗi dịch vụ xe lửa hoặc xe buýt có một tệp khác nhau (như Performance_MNR.xml cho Đường sắt MetroNorth) chứa dữ liệu hàng tháng dưới dạng một loạt bản ghi XML giống như sau:

```
<INDICATOR>
    <INDICATOR_SEQ>373889</INDICATOR_SEQ>
    <PARENT_SEQ></PARENT_SEQ>
    <AGENCY_NAME>Metro-North Railroad</AGENCY_NAME>
    <INDICATOR_NAME>Escalator Availability</INDICATOR_NAME>
    <DESCRIPTION>Percent of the time that escalators are operational
    systemwide. The availability rate is based on physical observations
        performed
    the morning of regular business days only. This is a new indicator the
        agency
    began reporting in 2009.</DESCRIPTION>
    <PERIOD_YEAR>2011</PERIOD_YEAR>
    <PERIOD_MONTH>12</PERIOD_MONTH>
    <CATEGORY>Service Indicators</CATEGORY>
    <FREQUENCY>M</FREQUENCY>
    <DESIRED_CHANGE>U</DESIRED_CHANGE>
    <INDICATOR_UNIT>%</INDICATOR_UNIT>
    <DECIMAL_PLACES>1</DECIMAL_PLACES>
    <YTD_TARGET>97.00</YTD_TARGET>
    <YTD_ACTUAL></YTD_ACTUAL>
    <MONTHLY_TARGET>97.00</MONTHLY_TARGET>
    <MONTHLY_ACTUAL></MONTHLY_ACTUAL>
</INDICATOR>
```

Sử dụng lxml.objectify, phân tích cú pháp file và nhận tham chiếu đến root node của file XML với getroot:

```
from lxml import objectify

path = 'Performance_MNR.xml'
parsed = objectify.parse(open(path))
root = parsed.getroot()
```

root.INDICATOR trả về một trình tạo mang lại mỗi phần tử XML <INDICATOR>. Mỗi bản ghi có thể được điền một lệnh tên tag (như YTD_ACTUAL) vào các giá trị dữ liệu:

```
data = []
skip_fields = ['PARENT_SEQ', 'INDICATOR_SEQ',
               'DESIRED_CHANGE', 'DECIMAL_PLACES']
for elt in root.INDICATOR:
 el_data = {}
 for child in elt.getchildren():
     if child.tag in skip_fields:
        continue
     el_data[child.tag] = child.pyval
     data.append(el_data)
```

Cuối cùng chuyển list of dicts thành dataframe:

```
In [927]: perf = DataFrame(data)
In [928]: perf.info()
Out[928]:
    <class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 12 columns):
 #   Column         Non-Null Count Dtype
---  ------         -------------- -----
 0   AGENCY_NAME    12 non-null    object
 1   INDICATOR_NAME 12 non-null    object
 2   DESCRIPTION    12 non-null    object
 3   PERIOD_YEAR    12 non-null    int64
 4   PERIOD_MONTH   12 non-null    int64
 5   CATEGORY       12 non-null    object
 6   FREQUENCY      12 non-null    object
 7   INDICATOR_UNIT 12 non-null    object
 8   YTD_TARGET     12 non-null    float64
 9   YTD_ACTUAL     12 non-null    object
 10  MONTHLY_TARGET 12 non-null    float64
 11  MONTHLY_ACTUAL 12 non-null    object
dtypes: float64(2), int64(2), object(8)
memory usage: 1.2+ KB
```

Dữ liệu XML có thể phức tạp hơn nhiều so với ví dụ này. Mỗi tag cũng có thể có metadata. Xem một HTML link tag cũng là XML hợp lệ:

```
from io import StringIO
tag = '<a href="http://www.google.com">Google</a>'


root = objectify.parse(StringIO(tag)).getroot()
```

Ta có thể truy cập bất kỳ field nào (như href) trong tag hoặc link text:

```
In [930]: root
Out[930]: <Element a at 0x2ae0c1e6fc0>


In [931]: root.get('href')
Out[931]: 'http://www.google.com'


In [932]: root.text
Out[932]: 'Google'
```

2 Binary Data Formats

Một trong những cách dễ dàng nhất để lưu trữ dữ liệu hiệu quả ở định dạng nhị phân là sử dụng tuần tự hóa pickle tích hợp sẵn của Python. Thuận tiện, tất cả các đối tượng pandas đều có một phương thức lưu để ghi dữ liệu vào bộ nhớ dưới dạng một pickle:

```
In [933]: frame = pd.read_csv('examples/ex1.csv')
In [934]: frame
Out[934]:
     a   b  c   d message
   0 1   2  3   4   hello
   1 5   6  7   8   world
   2 9 10 11 12     foo
In [935]: frame.to_pickle('examples/frame_pickle')
```

Bạn có thể đọc lại dữ liệu sang Python với pandas.load, một chức năng tiện lợi khác của pickle:

```
In [936]: pd.read_pickle('examples/frame_pickle')
Out[936]:
   a   b  c   d message
0  1   2  3   4   hello
1  5   6  7   8   world
2  9  10 11  12     foo
```

Pickle chỉ nên lưu ngắn hạn bởi vì nó khó đảm bảo ổn định theo thời gian. Một object đã được sử dụng có thể trong phiên bản mới hơn của thư viện sẽ bị lỗi.

2.1 Using HDF5 Format

Có một số công cụ hỗ trợ hiệu quả việc đọc và ghi một lượng lớn dữ liệu khoa học ở định dạng nhị phân trên đĩa. Thư viện cấp ngành phổ biến cho điều này là HDF5, là thư viện C có giao diện bằng nhiều ngôn ngữ khác như Java, Python và MATLAB. "HDF" trong HDF5 là viết tắt của định dạng dữ liệu phân cấp. Mỗi tệp HDF5 chứa cấu trúc nút giống như hệ thống tệp nội bộ cho phép bạn lưu trữ nhiều bộ dữ liệu và hỗ trợ siêu dữ liệu. So với các định dạng đơn giản hơn, HDF5 hỗ trợ nén nhanh với nhiều loại máy nén khác nhau, cho phép lưu trữ dữ liệu với các mẫu lặp đi lặp lại hiệu quả hơn. Đối với các tập dữ liệu rất lớn không vừa với bộ nhớ, HDF5 là một lựa chọn tốt vì bạn có thể đọc và ghi các phần nhỏ của mảng lớn hơn nhiều một cách hiệu quả.

Không chỉ một mà tới hai giao diện cho thư viện HDF5 trong Python, PyTables và h5py, mỗi giao diện có một cách tiếp cận vấn đề khác nhau. h5py cung cấp giao diện trực tiếp, HDF5 API là giao diện cao cấp hơn, trong khi PyTables tóm tắt nhiều chi tiết của HDF5 để cung cấp nhiều vùng chứa dữ liệu linh hoạt, lập chỉ mục bảng, khả năng truy vấn và một số hỗ trợ cho các tính toán ngoài lõi.

Pandas có tối thiểu một class dict-like HDFStore, class này sử dụng PyTables để lưu trữ các đối tượng pandas:

```
In [937]: store = pd.HDFStore('mydata.h5')
In [938]: store['obj1'] = frame
In [939]: store['obj1_col'] = frame['a']
In [940]: store
Out[940]:
   <class 'pandas.io.pytables.HDFStore'>
```

```
File path: mydata.h5
obj1       DataFrame
obj1_col   Series
```

Các đối tượng có trong tệp HDF5 có thể được truy xuất theo kiểu giống như dict:

```
In [941]: store['obj1']
Out[941]:
     a   b  c   d message
   0 1   2  3   4   hello
   1 5   6  7   8   world
   2 9  10 11 12     foo
```

Nếu bạn làm việc với số lượng lớn dữ liệu, ta nên sử dụng PyTables và h5py để xem chúng có thể phù hợp với nhu cầu của mình như thế nào. Vì nhiều vấn đề phân tích dữ liệu bị ràng buộc IO (thay vì ràng buộc CPU), việc sử dụng một công cụ như HDF5 có thể tăng tốc hàng loạt các ứng dụng của chúng ta.

HDF5 không phải là cơ sở dữ liệu. Nó phù hợp nhất cho các bộ dữ liệu ghi một lần, đọc nhiều. Mặc dù dữ liệu có thể được thêm vào tệp bất kỳ lúc nào, nhưng nếu nhiều người ghi đồng thời làm như vậy, tệp có thể bị hỏng.

2.2 Reading Microsoft Excel Files

Pandas cũng hỗ trợ đọc đọc file chứa dữ liệu trong Excel 2003 (và cao hơn) sử dụng lớp ExcelFile. ExcelFile sử dụn gói xlrd và openpyxl, vì thế bạn phải tải chúng trước. Để dùng ExcelFile, tạo ra một instance bằng cách tạo đường dẫn tới xls hoặc xlsx file:

```
xls_file = pd.ExcelFile('data.xls')
```

Dữ liệu lưu trong sheet có thể được được thành DataFrame bằng cách sử dụng lệnh :

```
table = xls_file.parse('Sheet1')
```

3 Interacting with HTML and Web APIs

Nhiều trang web có APIs công cộng cung cấp data thông qua JSON hoặc một vài định dạng khác. Có một cơ số cách để truy cập APIs thông qua Python; một cách đơn giản mà được khuyên

dùng là yêu cầu gói (http://docs.python-requests.org). Để tìm từ "python pandas" trên Twitter, ta có thể tạo một HTTP GET request như sau:

```
In [944]: import requests
In [945]: url = "https://twitter154.p.rapidapi.com/search/search"
In [946]: querystring = {"query":"#python%20pandas","section":"top",
                         "min_retweets":"20","min_likes":"20",
                         "limit":"5","start_date":"2022-01-01",
                         "language":"en"}
In [947]: headers = {"X-RapidAPI-Key": "3ca3abc1f7mshfd96a9ce916a026p1dcaa
                     bjsnaf1ec9ecd60f","X-RapidAPI-Host":
                     "twitter154.p.rapidapi.com"}
In [948]: response = requests.request("GET", url, headers=headers, params =
                                      querystring)
In [949]: response
Out[949]: <Response [200]>
```

Thuộc tính text của đối tượng Response chứa nội dung của truy vấn GET. Nhiều web API sẽ trả về một chuỗi JSON được tải vào một đối tượng Python:

```
In [950]: import json
In [951]: data_tweet = json.loads(resp.text)
In [952]: data_tweet.keys()
Out[953]: dict_keys(['results', 'continuation_token'])
```

Trường kết quả results trong đối tượng response chứa danh sách các tweet, mỗi tweet được trình bày như 1 dict python:

```
{'tweet_id': '1500450317481222148',
 'creation_date': 'Sun Mar 06 12:36:47 +0000 2022',
 'text': 'I recently learned of a neat pandas.DataFrame method called
    `select_dtypes` which is a much cleaner way of filtering columns on type
    than checking each column type in a for loop! #python #Pandas
    https://t.co/Btelyg7T4U',
  'media_url': ['https://pbs.twimg.com/media/FNKrlq1XEAEvP1T.png'],
  'video_url': None,
  'user': {'creation_date': 'Sat Oct 23 02:05:18 +0000 2021',
```

```
 'user_id': '1451731326697156619',
 'username': 'pypeaday',
 'name': ' PypeADay',
 'follower_count': 340,
 'following_count': 130,
 'favourites_count': 1541,
 'is_private': False,
 'is_verified': False,
 'location': '',
 'profile_pic_url':
    'https://pbs.twimg.com/profile_images/ifYGM2vo_normal.jpg',
 'profile_banner_url':
    'https://pbs.twimg.com/profile_banners/1451731326697156619/1662732097',
 'description': "Jik. Pipes and Python - usually not at the same time. I'm
    well-grounded by my beautiful wife, awesome daughters, and trusty
    anti-static bracelet",
 'external_url': 'https://pypeaday.github.io/littlelink/',
 'number_of_tweets': 644,
 'bot': False,
 'timestamp': 1634954718,
 'has_nft_avatar': False},
 'language': 'en',
 'favorite_count': 2223,
 'retweet_count': 255,
 'reply_count': 22,
 'quote_count': 8,
 'retweet': False,
 'timestamp': 1646570207,
 'video_view_count': None}
```

Sau đó, ta có thể tạo danh sách các trường tweet quan tâm, và chuyển danh sách kết quả tới DataFrame:

```
In [954]: tweet_fields = ['creation_date', 'text']
In [955]: tweets = pd.DataFrame(data_tweet['results'], columns=tweet_fields)
In [956]: tweets
```

```
Out[956]:
                  creation_date                                        text
 0  Sun Mar 06 12:36:47 +0000 2022   I recently learned of a neat pandas...
 1 Sun Jul 03 02:25:55 +0000 2022   Anatomy of Pandas data structures\n...
 2 Mon May 09 11:27:09 +0000 2022   Want to speedup Pandas DataFrame op...
 3 Sat Mar 19 17:37:00 +0000 2022   I have seen lots of great EDA tips ...
 4 Tue Apr 05 14:08:36 +0000 2022   If you have ever wanted to convert ...
```

Mỗi hàng trong DataFrame hiện có dữ liệu được trích xuất từ mỗi tweet:

```
In [957]: tweets.iloc[3]
Out[957]:
   creation_date                    Sat Mar 19 17:37:00 +0000 2022
   text             I have seen lots of great EDA tips for working i...
   Name: 3, dtype: object
```

Với một vài bước đơn giản, bạn có thể tạo một số giao tiếp cấp cao hơn cho web API trả về các đối tượng DataFrame để phân tích dễ dàng.

4 Interacting with Databases

Các ứng dụng hiếm khi truy cập data từ các text file, vì sự luồng dữ liệu qua lớn. Các databases server PostgresSQL, MySQL.. được sự dụng là giải pháp thay thế. Việc lựa chọn database dựa các yếu tố như hiệu năng, khả năng tích hợp và mở rộng theo yêu cầu của ứng dụng.

Việc truy suất data từ SQL thành DataFrame khá đơn giản, được hộ trở bởi một thư viện pandas. Ví dụ dưới đây sự dụng SQL-lit database:

```
import sqlite3
query = """
CREATE TABLE test
(a VARCHAR(20), b VARCHAR(20),
c REAL, d INTEGER );"""
con = sqlite3.connect(':memory:')
con.execute(query)
con.commit()
```

Sau đó , thêm vài hàng vào data table

```
data = [('Atlanta', 'Georgia', 1.25, 6), ('Tallahassee', 'Florida', 2.6, 3),
    ('Sacramento', 'California', 1.7, 5)]
stmt = "INSERT INTO test VALUES(?, ?, ?, ?)"
con.executemany(stmt, data)
con.commit()
```

Hầu hết các Python Sql Drivers trả về một list của các tuples khi lấy data từ bảng:

```
In [958]: cursor = con.execute('select * from test')
In [959]: rows = cursor.fetchall()
In [960]: rows
Out[960]:
    [(u'Atlanta', u'Georgia', 1.25, 6), (u'Tallahassee', u'Florida', 2.6, 3),
       (u'Sacramento', u'California', 1.7, 5)]
```

Bạn có thể chuyển list của các Tuples vào DataFrame khởi tạo, và yêu cầu thêm cả column name, ở đây nó được chứa trong thuộc tính descripton của con trỏ:

```
In [961]: cursor.description
Out[961]:
    (('a', None, None, None, None, None, None),
    ('b', None, None, None, None, None, None),
    ('c', None, None, None, None, None, None),
    ('d', None, None, None, None, None, None))
In [962]: a = np.array(cursor.description)
In [963]: pd.DataFrame(rows,columns=a[:,0])
Out[963]:
           a          b         c   d
    0 Atlanta   Georgia      1.25  6
    1 Tallahassee Florida   2.60  3
    2 Sacramento California 1.70  5
```

Tuy vậy pandas có hàm read_sql() giúp đơn giản hoá quá trình.

```
In [964]: import pandas.io.sql as sql
In [965]: sql.read_sql('select * from test', con)
```

```
Out[965]:
          a          b         c  d
   0 Atlanta    Georgia      1.25  6
   1 Tallahassee Florida    2.60  3
   2 Sacramento California 1.70  5
```

4.1 Storing and Loading Data in MongoDB

NoSql databases có thể gồm nhiều dạng khác nhau. Các kiểu lưu trữ dict-like-value đơn giản như BrerkeleyDB hoặc Tokyo Cabinet, trong khi các kiểu khác là document-based, với dict-like object là một đơn vị lưu trữ. Ví dụ dưới đây sự dụng MongoDB, ta khởi tạo một mongodb instance trong máy và kết nối nó với cổng mặc định bằng pymongo, offical driver của MongoDbB:

```
from pymongo import MongoClient
client = MongoClient('localhost', port=27017)
```

Documents được lưu trữ trong MongoDB đường tìm thấy trong collections bên trong databases. Mỗi instance của server có nhiều databases, mỗi database có nhiều collections. Giả sử lưu trữ API twitter data từ chapter trước. Đầu tiên ta có thể truy cập tweets collection hiện tại:

```
tweets_collection = client.db.tweets_database_collection
```

Sau đó load list tweets từ api và viết nó vào mộ sưu tập sự dụng tweets.insert_one:

```
for tweet in data_tweet['results']:
    tweets_collection.insert_one(tweet)
```

Bây giờ để truy xuất toàn bộ hoặc bất kỳ tweets nào từ bộ sưu tập ta có thể hiện với cú pháp sau:

```
cursor = tweets_collection.find({"user.username": "thedataprof"})
```

Con trỏ được trả về là đối tượng vòng lặp duyệt document như là một dict. Như ở trên ta có thể chuyển đổi kết quả thành dataframe:

```
tweet_fields = ['creation_date', 'text']
result = pd.DataFrame(list(cursor), columns=tweet_fields)
```

5 Link Github và tài liệu tham khảo

[1] Link github các code trong file jupyter đã thực hiện chạy và có kết quả của nhóm: https://github.com/minhkhoi245/Ch6giuaky.git

[2] Wes McKinney, ”Python for Data Analytics”, 1005 Gravenstein Highway North, Sebastopol, CA 95472, O’Reilly Media, Inc, October 2012.