

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH
KHOA ĐIỆN-ĐIỆN TỬ



HCMUTE

MÔN HỌC THỰC TẬP PYTHON

TIÊU LUẬN CUỐI KỲ

GVHD: Hồ Nhựt Minh

SVTH: Nguyễn Đình Đức Thọ 20139092

TP Hồ Chí Minh, tháng 12 năm 2022

Mục Lục

1. Lập trình Python cơ bản
 - 1.1. Python Introduction
 - 1.2. For Loops and Conditional Statements in Python
 - 1.3. Dictionaries, Frequency Tables, and Functions in Python
 - 1.4. Python Functions
 - 1.5. Lists and For Loops Practice Problems
 - 1.6. Conditional Statements Practice Problems
 - 1.7. Functions Fundamentals Practice Problems
2. Object-Oriented Python
 - 2.1. Object-Oriented Python
 - 2.2. Object-Oriented Python Practice Problems
3. Numpy
4. Pandas
 - 4.1. Pandas Part 1
 - 4.2. Pandas Part 2
5. Pandas tiếp theo
 - 5.1. Introduction to pandas
 - 5.2. Exploring data with pandas fundamentals
 - 5.3. Exploring data with pandas intermediate
 - 5.4. Data cleaning basics
6. Visualization
 - 6.1. Line Graphs and Time Series
 - 6.2. Scatter Plots and Correlations
 - 6.3. Bar Plots Histograms and Distributions
 - 6.4. Pandas Visualizations and Grid Charts
7. Exercise
 - 7.1. Requests
 - 7.2. Pillow

- 7.3. BeautifulSoup
- 7.4. Matplotlib
- 7.5. Flask
- 7.6. Challenge
- 7.7. Solution
- 8. Đồ án cuối kỳ
 - 8.1. Giới thiệu đề tài
 - 8.2. Lấy dữ liệu từ API
 - 8.3. Xử lý và phân tích dữ liệu
 - 8.4. Kết quả

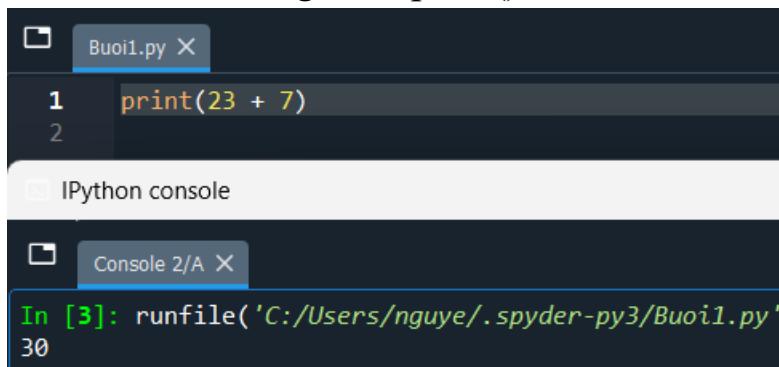
01 Python Introduction

1.	Python Programming	2
1.1.	The print() Function.....	2
1.2.	Python Syntax.....	2
1.3.	Computer Programs	3
1.4.	Code Comments	3
1.5.	Các phép toán	4
2.	Programming Python Variables	5
2.1.	Lưu các giá trị	5
2.2.	Variables	6
2.3.	Tên biến	6
2.4.	Updating Variables	8
2.5.	Syntax Shortcuts	8
3.	Python Data Types: Integers, Floats, Strings.....	10
3.1.	Integers and Floats.....	10
3.2.	Conversion Between Types	11
3.3.	Strings	13
3.4.	Các ký tự đặc biệt	13
3.5.	Nối chuỗi	14
3.6.	Chuyển đổi chuỗi	15
3.7.	Chuỗi nhiều dòng.....	15
4.	List trong Python.....	16
4.1.	Lưu trữ các phần tử rows thành các biến.....	16
4.2.	Chuyển rows thành lists.....	16
4.3.	Độ dài của list	17
4.4.	List indexing và lấy giá trị từ List	17
4.5.	Indexing đảo	19
4.6.	List Slicing.....	19
4.7.	Lists trong List.....	20
4.8.	Truy xuất Lists trong List	20

1. Python Programming

1.1. The print() Function

Ta thực hiện một phép tính đơn giản, để hiển thị kết quả của phép tính ra màn hình ta sử dụng hàm print()

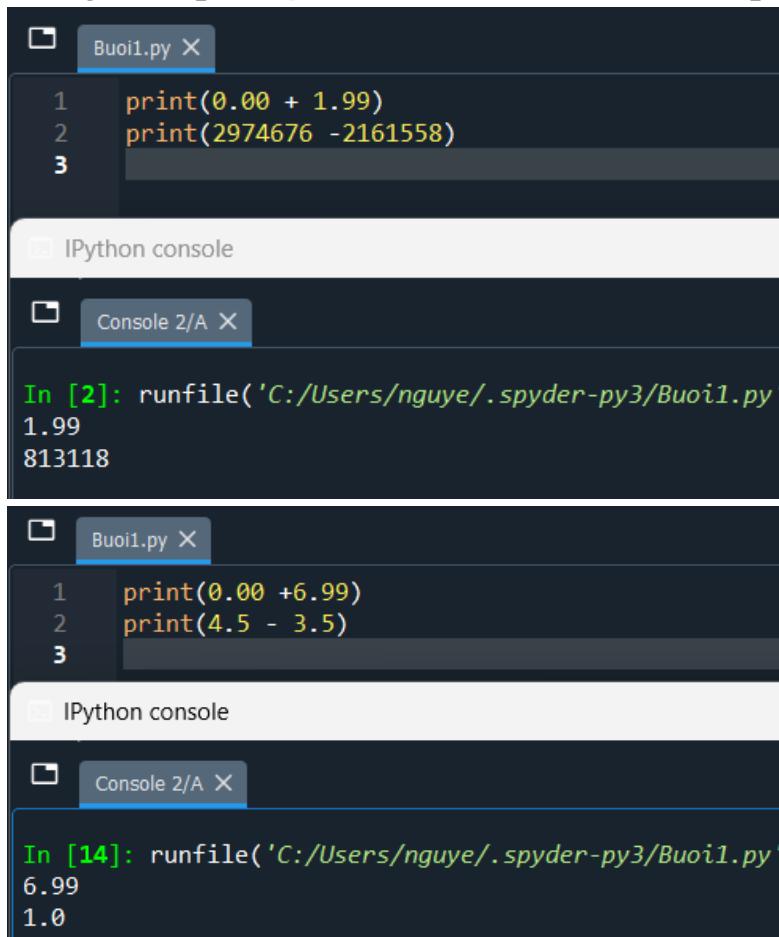


The screenshot shows the Spyder IDE interface. At the top, there is a code editor window titled "Buoi1.py" containing the following code:

```
1 print(23 + 7)
2
```

Below the code editor is an "IPython console" window. In the "Console 2/A" tab, the command "In [3]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py')" is entered, followed by the output "30".

Dùng hàm print() để thực hiện nhiều hơn 1 phép tính.



The screenshot shows the Spyder IDE interface. It displays three separate code editor windows, each titled "Buoi1.py". The first window contains:

```
1 print(0.00 + 1.99)
2 print(2974676 - 2161558)
3
```

The second window contains:

```
In [2]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py')
1.99
813118
```

The third window contains:

```
1 print(0.00 +6.99)
2 print(4.5 - 3.5)
3
```

Below these code editors are two IPython console windows. The first "Console 2/A" window shows the output of the first two print statements from the first code editor: "1.99" and "813118". The second "Console 2/A" window shows the output of the first two print statements from the third code editor: "6.99" and "1.0".

1.2. Python Syntax

Để thực hiện nhiều phép tính, ta buộc phải để mỗi dòng 1 phép tính, nếu không sẽ gặp lỗi như hình dưới.

Buoi1.py*

```

1 print(0.00 + 6.99) print(4.5 - 3.5)
2

```

IPython console

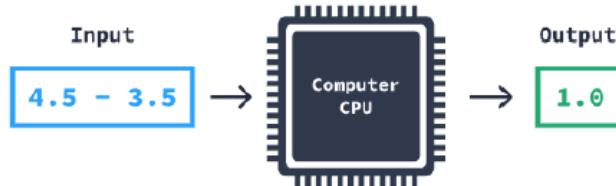
Console 2/A

```

In [15]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py',
File ~\spyder-py3\Buoi1.py:1
    print(0.00 + 6.99) print(4.5 - 3.5)
          ^
SyntaxError: invalid syntax

```

1.3. Computer Programs



Máy tính sẽ thực hiện tất cả các phép tính, như ví dụ dưới, chúng ta không thấy kết quả của $0.00 + 6.99$ vì chúng ta không sử dụng hàm `print()` để đưa kết quả ra màn hình.

Buoi1.py

```

1 0.00 + 6.99
2 print (4.5 - 3.5)
3

```

IPython console

Console 2/A

```

In [16]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py'
1.0

```

1.4. Code Comments

Để ghi chú gì đó ở chương trình, ta dùng ký tự `#`. Máy tính sẽ bỏ qua các ký tự khác bên cạnh ký tự `#`.

The screenshot shows the Spyder IDE interface with two code editors and two IPython consoles.

- Code Editor 1 (Top):** Contains the following Python code:


```
1 print (1.99)    # App cost for Fruit Ninja Classic
2 print (39583534)  # Rating count
3 print (4.5)      # Average user rating
4
```
- IPython Console 1:** Shows the output of the code from the first editor:


```
In [22]: runfile('C:/Users/nguye/.spyder-py3/Buoil.py', wdir='C:/Users/nguye/.spyder-py3')
1.99
39583534
4.5
```
- Code Editor 2 (Bottom):** Contains the following Python code:


```
1 # Minecraft: Pocket Edition cost, rating count, and user rating
2 print(6.99)
3 print (522012)
4 print (4.5)
```
- IPython Console 2:** Shows the output of the code from the second editor:


```
In [23]: runfile('C:/Users/nguye/.spyder-py3/Buoil.py', wdir='C:/Users/nguye/.spyder-py3')
6.99
522012
4.5
```

1.5. Các phép toán

Như những ví dụ trên, chúng ta đã có thể thực hiện các phép tính cộng trừ. Tuy nhiên chúng ta có thể tính các phép tính cơ bản khác. Ví dụ, để thực hiện phép nhân ta dùng ký tự * .

The screenshot shows the Spyder IDE interface with one code editor and one IPython console.

- Code Editor:** Contains the following Python code:


```
1 print ( 1.99 * 2 )
2
```
- IPython Console:** Shows the output of the code from the code editor:


```
In [24]: runfile('C:/Users/nguye/.spyder-py3/Buoil.py', wdir='C:/Users/nguye/.spyder-py3')
3.98
```

Phép chia ta dùng / .

The screenshot shows the Spyder IDE interface. At the top, there is a tab bar with 'Buoi1.py' selected. Below it, the code editor contains two lines of Python code: 'print (6.99 / 2)'. In the bottom right corner of the code editor, there is a small gray box containing the number '2'. To the right of the code editor is the IPython console window, titled 'Console 2/A'. It displays the command 'In [25]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py')' followed by the output '3.495'.

Ví dụ tính số mũ 3^2 ta dùng `**`.

The screenshot shows the Spyder IDE interface. At the top, there is a tab bar with 'Buoi1.py' selected. Below it, the code editor contains two lines of Python code: 'print (3 ** 2)'. In the bottom right corner of the code editor, there is a small gray box containing the number '2'. To the right of the code editor is the IPython console window, titled 'Console 2/A'. It displays the command 'In [26]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py')' followed by the output '9'.

Các phép toán trong Python cũng tuân thủ theo quy tắc trong ngoặc, mũ, nhân và chia, và cuối cùng là cộng trừ như toán học cơ bản.

The screenshot shows the Spyder IDE interface. At the top, there is a tab bar with 'Buoi1.py' selected. Below it, the code editor contains two lines of Python code: 'print (6.99 + 0.00 * 10)' and 'print ((6.99 + 0.00) * 10)'. In the bottom right corner of the code editor, there is a small gray box containing the number '2'. To the right of the code editor is the IPython console window, titled 'Console 2/A'. It displays the command 'In [29]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py')', followed by the output '6.99' on the first line and '69.9' on the second line.

2. Programming Python Variables

2.1. Lưu các giá trị

Chúng ta có thể đặt cho các giá trị 1 cái tên, ký tự. Như vậy sẽ tiện hơn trong việc thay đổi, cập nhập các giá trị.

The screenshot shows the Spyder IDE interface. At the top, there's a tab bar with 'Buoi1.py' selected. Below it is an 'IPython console' window containing the code: 'result = 3.98' and 'print (result)'. The output window below shows the result: 'In [34]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py', 3.98).

Chúng ta có thể đặt kết quả của phép tính vào biến mà chúng ta muốn, và hàm print() cũng có thể in giá trị của biến đó ra màn hình.

The screenshot shows the Spyder IDE interface. At the top, there's a tab bar with 'Buoi1.py' selected. Below it is an 'IPython console' window containing the code: 'result = 3.98 * 2' and 'print (result)'. The output window below shows the result: 'In [36]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py', 7.96).

2.2. Variables

Khi chạy code, giá trị 3.98 sẽ được lưu vào trong memory của máy tính. Để có thể truy xuất được giá trị này, ta đặt tên cho nó. Tên này gọi là tên biến.

Lưu ý, chúng ta cần viết tên biến bên trái của toán tử bằng, còn giá trị sẽ được ghi bên phải. chúng ta có thể đặt 1 tên bất kỳ thay cho chữ Result bên dưới.

The screenshot shows the Spyder IDE interface. At the top, there's a tab bar with 'Buoi1.py' selected. Below it is an 'IPython console' window containing the code: 'Result = 3.98', 'print (Result)', and 'print (Result + 1.99)'. The output window below shows the results: 'In [32]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py', 3.98, 5.97).

2.3. Tên biến

Việc đặt tên biến cũng phải tuân theo 1 số quy luật nhất định. Như hình bên dưới, chúng ta không thể dùng 1 khoảng trắng để đặt tên, kết quả chạy sẽ báo lỗi.

The screenshot shows the Spyder IDE interface. At the top, there's a tab bar with 'Buoi1.py' selected. Below it is an IPython console window containing the following text:

```
In [37]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py')
File ~\spyder-py3\Buoi1.py:1
    a result = 3.98
          ^
SyntaxError: invalid syntax
```

Đây là 2 quy tắc trong việc đặt tên biến.

- Chúng ta chỉ được sử dụng chữ cái, số, dấu gạch dưới (không thể dùng nháy đơn, dấu gạch ngang, dấu cách, ...)
- Tên biến không được bắt đầu bằng 1 số.

Dưới đây là một vài ví dụ hợp lệ và không hợp lệ:

<u>Valid names</u>	<u>Invalid names</u>	<u>Why invalid</u>
data_09_01_2018	1_data	→ starts with a number
RESULT_1	new data	→ has a white space character
Result_1	john's_salary	→ has an apostrophe
ouTput	old-data	→ has a hyphen
_Output	price_in_\$	→ has the \$ character

Và trong Python, máy tính cũng phân biệt chữ hoa và chữ thường. Như ví dụ dưới, result sẽ khác Result.

The screenshot shows the Spyder IDE interface. The top window is titled 'Buoi1.py' and contains the following Python code:

```
1 result = 3.98
2 Result = 39823
3 print (result)
4 print (Result)
```

The bottom window is titled 'IPython console' and shows the execution of the file:

```
In [38]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py', 3.98
39823
```

2.4. Updating Variables

Chúng ta có thể cập nhật giá trị của biến bằng các phép toán số học với chính nó, kết quả của phép tính đó sẽ được ghi đè lại tên biến đó.

The screenshot shows the Spyder IDE interface. The top window is titled 'Buoi1.py' and contains the following Python code:

```
1 result = 3.98
2 result = result + 10
3 print (result)
```

The bottom window is titled 'IPython console' and shows the execution of the file:

```
In [41]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py', 13.98
```

2.5. Syntax Shortcuts

The screenshot shows the Spyder IDE interface. The top window is titled 'Buoi1.py' and contains the following Python code:

```
1 result = 3.98
2 print (result)
3
4 print (result + 9.9)
5
6 result = result + 9.9
7 print (result)
```

The bottom window is titled 'IPython console' and shows the execution of the file:

```
In [45]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py', 3.98
13.88
13.88
```

The screenshot shows the Spyder IDE interface. The top window is titled 'Buoi1.py' and contains the following Python code:

```

1 result = 3.98
2 result += 9.9
3 print(result)

```

The bottom window is titled 'IPython console' and shows the output of running the script:

```

In [46]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py')
13.88

```

Như 2 ví dụ trên, ta thấy kết quả cho ra giống nhau, vì thế chúng ta có cập nhật các giá trị của biến bằng cách ghi tắt, một số ký tự tắt ở bảng này.

Syntax shortcut	Example code	Output
<code>+=</code>	<code>x = 4 x += 2</code>	<code>6</code>
<code>-=</code>	<code>x = 4 x -= 2</code>	<code>2</code>
<code>*=</code>	<code>x = 4 x *= 2</code>	<code>8</code>
<code>/=</code>	<code>x = 4 x /= 2</code>	<code>2</code>
<code>***=</code>	<code>x = 4 x ***= 2</code>	<code>16</code>

Lưu ý để sử dụng các ký tự tắt này, biến phải được định nghĩa trước đó, nếu không chương trình sẽ báo lỗi như hình dưới.

The screenshot shows the Spyder IDE interface. The top window is titled 'Buoi1.py' and contains the following Python code:

```

1 # Error:
2 result += 6

```

The bottom window is titled 'IPython console' and shows the resulting error message:

```

In [47]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py',
Traceback (most recent call last):
File ~\spyder-py3\Buoi1.py:2 in <module>
      result += 6
NameError: name 'result' is not defined

```

Để sửa lại, ta chỉ cần định nghĩa 1 giá trị trước khi dùng ký tự tắt cho biến đó.

The screenshot shows the Spyder IDE interface. At the top, there is a code editor window titled "Buoil.py" containing the following Python code:

```
1 # No Error:  
2 result = 10  
3 result += 6  
4 print(result)
```

Below the code editor is an "IPython console" window titled "Console 2/A". It displays the output of running the file:

```
In [48]: runfile('C:/Users/nguye/.spyder-py3/Buoil.py', 16)
```

3. Python Data Types: Integers, Floats, Strings

3.1. Integers and Floats

Trong Python, chúng ta có định nghĩa kiểu số nguyên và số thập phân.

The screenshot shows the Spyder IDE interface. At the top, there is a code editor window titled "Buoil.py" containing the following Python code:

```
1 print(6.99 * 1)  
2 print(0.0 + 1.99)
```

Below the code editor is an "IPython console" window titled "Console 2/A". It displays the output of running the file:

```
In [49]: runfile('C:/Users/nguye/.spyder-py3/Buoil.py', 6.99  
1.99)
```

Trong toán học, số nguyên khác số thập phân, và Python cũng biết điều đó. Để xem kiểu dữ liệu của 1 giá trị hay 1 biến ta dùng hàm type()

The screenshot shows the Spyder IDE interface. At the top, there is a code editor window titled "Buoil.py" containing the following Python code:

```
1 print(type(0))  
2 print(type(9.9))
```

Below the code editor is an "IPython console" window titled "Console 2/A". It displays the output of running the file:

```
In [55]: runfile('C:/Users/nguye/.spyder-py3/Buoil.py', <class 'int'>  
<class 'float'>)
```

Ta cũng có thể tính toán giữa 2 kiểu dữ liệu int và float này.

The screenshot shows the Spyder IDE interface. At the top, there is a code editor window titled "Buoi1.py" containing the following Python code:

```
1 print (0 + 8.78)
2 print (9 * 3.3)
```

Below the code editor is an "IPython console" window showing the output of the code:

```
In [58]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py',
8.78
29.7
```

Các biến có 2 kiểu dữ liệu này đều có thể cập nhật giá trị bằng các ký tự tắt mà chúng ta đã nói ở trên.

The screenshot shows the Spyder IDE interface. At the top, there is a code editor window titled "Buoi1.py" containing the following Python code:

```
1 bien1 = 6
2 bien1 += 7.5
3 bien2 = 8.8
4 bien2 *= 10
5
6 print(bien1)
7 print(bien2)
```

Below the code editor is an "IPython console" window showing the output of the code:

```
In [61]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py'
13.5
88.0
```

3.2. Conversion Between Types

Chúng ta có thể ép kiểu dữ liệu như chúng ta muốn. Để ép từ int sang float ta dùng lệnh float().

The screenshot shows the Spyder IDE interface. At the top, there is a code editor window titled "Buoi1.py" containing the following Python code:

```
1 print (float(10))
```

Below the code editor is an "IPython console" window showing the output of the code:

```
In [62]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py'
10.0
```

Và ngược lại có thể dùng int() để ép float về int.

The screenshot shows the Spyder IDE interface. The code editor window (Buoi1.py) contains the line: `print (int(8.38))`. The IPython console window (Console 2/A) shows the output: `In [63]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py', 8)`, which results in the value `8`.

Khi ép kiểu float sang int, phần thập phân sẽ được bỏ và giá trị đó sẽ được làm tròn xuống như ví dụ dưới đây.

The screenshot shows the Spyder IDE interface. The code editor window (Buoi1.py) contains the line: `print (int(8.99999999))`. The IPython console window (Console 2/A) shows the output: `In [64]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py', 8)`, which results in the value `8`.

Vậy để làm tròn đúng quy tắc làm tròn, ta dùng lệnh `round()`.

The screenshot shows the Spyder IDE interface. The code editor window (Buoi1.py) contains three lines of code: `print (round(8.999))`, `print (round(1.5))`, and `print (round(2.112))`. The IPython console window (Console 2/A) shows the output: `In [65]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py', 9)`, `9`, `2`, and `2`.

Và lệnh `round()` không phải dùng để ép kiểu dữ liệu, để muốn giữ kết quả làm tròn, ta có thể lưu tên biến đã đặt trước đó. Bên dưới là các ví dụ.

The screenshot shows the Spyder IDE interface. At the top, there is a tab bar with 'Buoi1.py X'. Below it is a code editor window containing the following Python code:

```

1 bien = 8.9
2
3 print(round(bien))
4 print(bien)
5
6 bien = round(bien)
7 print(bien)

```

Below the code editor is an 'IPython console' window titled 'Console 2/A X'. It displays the following output:

```

In [66]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py',
9
8.9
9

```

3.3. Strings

Ngoài kiểu dữ liệu int và float, chúng ta có kiểu string để lưu trữ dãy ký tự chữ, chữ và số, hoặc số... và dãy ký tự này được đặt trong dấu “” hoặc ‘’.

The screenshot shows the Spyder IDE interface. At the top, there is a tab bar with 'Buoi1.py X'. Below it is a code editor window containing the following Python code:

```

1 Mon = 'Python'
2 Lop = "20139A"
3 print (Mon)
4 print (Lop)

```

Below the code editor is an 'IPython console' window titled 'Console 2/A X'. It displays the following output:

```

In [67]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py',
Python
20139A

```

Str là kết quả trả về khi dùng lệnh type vs những dãy ký tự có kiểu dữ liệu là strings

The screenshot shows the Spyder IDE interface. At the top, there is a tab bar with 'Buoi1.py* X' and 'temp.py X'. Below it is a code editor window containing the following Python code:

```

1 Mon = 'Python'
2 print (type(Mon))

```

Below the code editor is an 'IPython Console' window titled 'Console 1/A X'. It displays the following output:

```

In [1]: runcell(0, 'C:/Users/nguye/.spyder-py3/Buoi1.py')
<class 'str'>

```

3.4. Các ký tự đặc biệt

Đôi khi chúng ta cần thay đổi linh hoạt giữa “” và ‘’. Nếu trong câu có “” thì chuỗi str chúng ta sẽ để trong dấu ‘’ (ví dụ bên dưới, dòng 3). Và ngược lại nếu trong câu có ‘’ thì chuỗi str chúng ta sẽ để trong dấu “” (ví dụ bên dưới, dòng 1).

Ở 1 vài trường hợp đặc biệt, chúng ta có thể thay dấu câu gốc chứa dấu ‘’ thành \‘\’ hay thay “” thành \“\” nếu trong câu có cả 2 dấu được sử dụng. Lúc này máy tính sẽ hiểu dấu được thay là 1 ký tự và in ra màn hình. (ví dụ dưới, dòng 5).

The screenshot shows the Spyder IDE interface. At the top, there are two tabs: 'Buoi1.py' (active) and 'temp.py'. Below the tabs is a code editor window containing the following Python code:

```
1 bien = "Hom nay troi 'rat dep'"
2 print (bien)
3 bien1 = 'Hom nay troi "rat dep"'
4 print (bien1)
5 bien2 = 'Hom nay troi \'rat dep\''
6 print (bien2)
```

Below the code editor is an 'IPython Console' window. It shows the command 'In [3]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py')' followed by three lines of output:

```
Hom nay troi 'rat dep'
Hom nay troi "rat dep"
Hom nay troi 'rat dep'
```

3.5. Nối chuỗi

Chúng ta có thể ghép nối các chuỗi ký tự str với nhau bằng phép cộng +. Ta cũng có thể copy lại ký tự với số lần lặp tùy thích bằng phép nhân *.

The screenshot shows the Spyder IDE interface. At the top, there are two tabs: 'Buoi1.py' (active) and 'temp.py'. Below the tabs is a code editor window containing the following Python code:

```
1 thoitiet = 'Hom' + ' nay' + ' troi dep '
2 print (thoitiet)
3 tt = thoitiet * 5
4 print (tt)
```

Below the code editor is an 'IPython Console' window. It shows the command 'In [6]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py', wdir='C:/Users/nguye/.spyder-py3')' followed by five lines of output:

```
Hom nay troi dep
Hom nay troi dep Hom nay troi dep Hom nay troi dep Hom nay troi dep Hom nay troi dep
```

3.6. Chuyển đổi chuỗi

The screenshot shows the Spyder IDE interface. In the top tab bar, 'Buoi1.py X' is active. Below it, the code editor contains the following Python script:

```
1 bien1 = str('88')
2 print(type(bien1))
3
4 bien2 = str('10.5')
5 print(type(bien2))
6
7 bien = bien1+bien2
8 print(bien)
9
10 kq = int(bien1)+float(bien2)
11 print(kq)
```

In the bottom IPython Console, the output of running the script is shown:

```
In [11]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py'
<class 'str'>
<class 'str'>
8810.5
98.5
```

Nếu những ký tự trong chuỗi là số chúng ta ép kiểu dữ liệu thành int hoặc float tương ứng, và ngược lại, chúng ta cũng có thể đưa số ở kiểu int hoặc float về lại str. (Như ví dụ ở trên).

The screenshot shows the Spyder IDE interface. In the top tab bar, 'Buoi1.py X' is active. Below it, the code editor contains the following Python script:

```
1 bien1 = 'Sai'
2 int(bien1)
```

In the bottom IPython Console, the output of running the script is shown:

```
In [12]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py', wdir='C:/U
Traceback (most recent call last):
File "C:\Users\nguye\.spyder-py3\Buoi1.py", line 2, in <module>
    int(bien1)
ValueError: invalid literal for int() with base 10: 'Sai'
```

Tuy nhiên, ta cần chú ý rằng không phải tất cả str có thể ép kiểu int hay float, nếu trong str có nhiều hơn 1 số, hoặc có ký tự chữ cái, ký tự khác thì không thể đưa sang kiểu dữ liệu int và float, như ví dụ trên chương trình sẽ báo lỗi.

3.7. Chuỗi nhiều dòng

Để in ra 1 chuỗi gồm nhiều dòng chỉ với 1 tên biến, ta có thể dùng dấu ‘ ‘ hoặc “ “ “ ở đầu và ở cuối các dòng đó.

```

1 bien1 = '''Hom
2 Nay
3 That
4 Dep'''
5 print(bien1)

```

IPython Console

```

In [15]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py')
Hom
Nay
That
Dep

```

4. List trong Python

4.1. Lưu trữ các phần tử rows thành các biến

List là 1 trong những cấu trúc dữ liệu phổ biến nhất trong python, giả sử, chúng ta có 1 hàng gồm nhiều thông tin với các kiểu dữ liệu khác nhau (như hình dưới), list có thể gom những biến đó thành 1 biến duy nhất để chúng ta có thể dễ kiểm soát hơn.

```

1 name_row1 = 'Dung'
2 age_row1 = 20.5
3 id_row1 = 12345

```

Variable Explorer

Name	Type	Size	
age_row1	float	1	20.5
id_row1	int	1	12345
name_row1	str	4	Dung

4.2. Chuyển rows thành lists

Vậy như đã nói ở trên, cách để chuyển nhiều biến nhỏ thành list là dùng lệnh với cú pháp: tên_list = [‘str’, int , float, int, float, ‘str’,] . Và khi dùng lệnh type() máy tính cũng cho chúng ta thấy nó sẽ tự động nhận cú pháp trên là 1 list.

The screenshot shows the Spyder IDE interface. At the top, there are two tabs: 'Buoi1.py X' and 'temp.py X'. Below the tabs, the code editor contains the following Python code:

```
1 row1 = ['Dung', 20.5, 12345]
2 print (row1)
3 print(type(row1))
```

Below the code editor is the 'IPython Console' tab, which is currently active. It displays the output of the code execution:

```
In [19]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py',
['Dung', 20.5, 12345]
<class 'list'>
```

4.3. Độ dài của list

Để đếm xem 1 list gồm bao nhiêu phần tử, ta có thể dùng lệnh len().

The screenshot shows the Spyder IDE interface. At the top, there are two tabs: 'Buoi1.py X' and 'temp.py X'. Below the tabs, the code editor contains the following Python code:

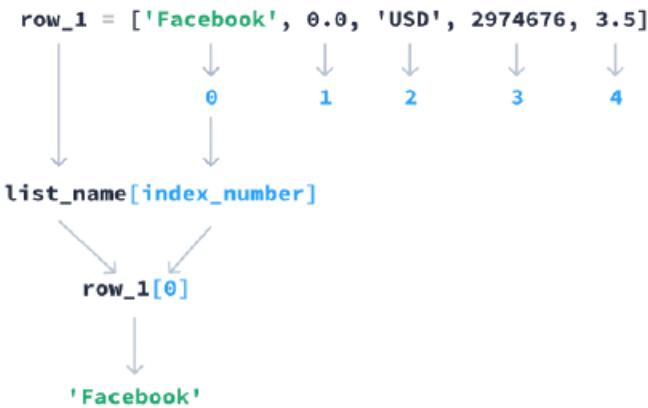
```
1 row1 = ['Dung', 20.5, 12345]
2 row2 = [1,5,6,3,2,45, "ten", "nine"]
3 print (row1)
4 print(len(row1))
5 print (row2)
6 print(len(row2))
```

Below the code editor is the 'IPython Console' tab, which is currently active. It displays the output of the code execution:

```
In [22]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py'
['Dung', 20.5, 12345]
3
[1, 5, 6, 3, 2, 45, 'ten', 'nine']
8
```

4.4. List indexing và lấy giá trị từ List

Mỗi phần tử trong list sẽ được gán tự động 1 số được gọi là index, bắt đầu từ số 0 đến n. Vậy mặc định ký tự đầu có index là 0 và số cuối cùng là có số index = len(list) – 1 .



Để dễ truy xuất đến từng phần tử trong list ta có thể dùng cú pháp sau: list_name[index], với index ứng với phần tử mà ta muốn gọi.

```

Buoi1.py X temp.py X
1 row1 = ['Dung', 20.5, 12345, 999]
2 print(row1[0])
3 print(row1[1])

IPython Console

Console 1/A X
In [29]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py',
Dung
20.5

```

Ta có thể ứng dụng việc truy xuất index trong việc tính toán qua lại giữa các list, list vs số, ...

```

Buoi1.py X temp.py X
1 row1 = ['Dung', 20.5, 12345]
2 row2 = ['Vy', 26, 98765]
3
4 row = row2[1] - row1[1]
5 print(row)

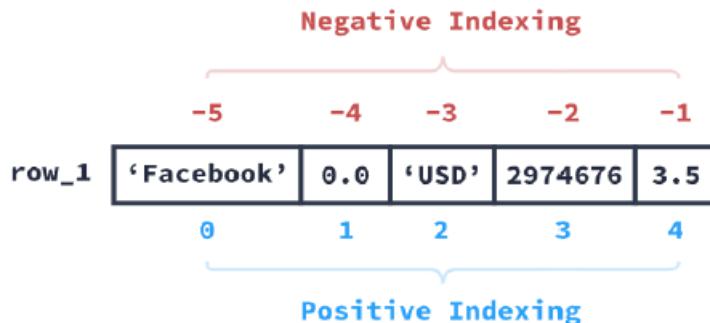
IPython Console

Console 1/A X
In [30]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py',
5.5

```

4.5. Indexing đảo

Ngoài ra, python còn hỗ trợ index đảo ngược, ứng với số cuối index là -1 và đếm ngược dần lên giảm 1 đơn vị.



Cụ thể hơn, ví dụ, nếu chúng ta muốn lấy ra phần tử thứ 3 trong list bên dưới ta đều có thể dùng với index là 2 hoặc -1 đều được.

```
Buoi1.py X temp.py X
1 row1 = ['Dung', 20.5, 12345]
2 print (row1[-1])
3 print (row1[2])

IPython Console
Console 1/A X
In [31]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py')
12345
12345
```

Và nếu chúng ta nhập index lớn hơn số index là máy cấp cho list đó, chương trình sẽ báo lỗi như hình dưới.

```
Buoi1.py X temp.py X
1 row1 = ['Dung', 20.5, 12345]
2 print (row1[3])

IPython Console
Console 1/A X
In [32]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py', wdir='C:/U
Traceback (most recent call last):
  File "C:\Users\nguye\.spyder-py3\Buoi1.py", line 2, in <module>
    print (row1[3])
IndexError: list index out of range
```

4.6. List Slicing

Để truy xuất ra 1 loạt các phần tử trong list, ta cần xác định index đầu và index cuối của những phần tử cần truy, sau đó ngăn dấu bằng dấu : .

The screenshot shows the Spyder IDE interface. In the top tab bar, 'Buoi1.py' is active. Below it, the IPython Console shows the output of running 'Buoi1.py'. The console output is:

```
In [38]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py')
['Dung', 20.5, 12345]
['Dung', 20.5, 12345]
```

Ta có in từ đầu tới 1 index bất kỳ bằng cách bỏ qua index đầu và ngược lại nếu cần in tới cuối list, ta có thể bỏ qua index cuối.

The screenshot shows the Spyder IDE interface. In the top tab bar, 'Buoi1.py' is active. Below it, the IPython Console shows the output of running 'Buoi1.py'. The console output is:

```
In [42]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py')
['Dung', 20.5, 12345]
['Hi', 9871]
```

4.7. Lists trong List

Ta có thể gộp nhiều list thành 1 list bằng cách gọi lần lượt chúng ra theo thứ tự ta muốn và đặt cho chúng 1 tên biến dưới kiểu list.

The screenshot shows the Spyder IDE interface. In the top tab bar, 'Buoi1.py' is active. Below it, the IPython Console shows the output of running 'Buoi1.py'. The console output is:

```
In [43]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py',
wdir='C:/Users/nguye/.spyder-py3')
[['Dung', 20.5, 12345, 'Hi'],
 ['Vy', 21.2, 45673, 'Fun'],
 ['Minh', 19.8, 78532, 'Win'],
 ['Nhan', 21.5, 98337, 'Fast']]
```

4.8. Truy xuất Lists trong List

Ta có thể gọi 1 list trong list lớn chứa nó bằng số index được cấp tương ứng. Tương tự việc gọi từng phần tử trong list như ta đã học ở trên.

The screenshot shows the Spyder IDE interface. At the top, there are two tabs: 'Buoi1.py X' and 'temp.py X'. The code in 'Buoi1.py' is:

```
1 row1 = ['Dung', 20.5, 12345, 'Hi']
2 row2 = ['Vy', 21.2, 45673, 'Fun']
3 row3 = ['Minh', 19.8, 78532, 'Win']
4 row4 = ['Nhan', 21.5, 98337, 'Fast']
5 row = [row1, row2, row3, row4]
6
7 print(row[1])
8 print(row[:2])
```

Below the code editor is the IPython Console tab, which contains the output of the run command:

```
In [46]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py', wdir='C:/Users/nguye/.spyder-py3')
['Vy', 21.2, 45673, 'Fun']
[['Dung', 20.5, 12345, 'Hi'], ['Vy', 21.2, 45673, 'Fun']]
```

Vậy để truy xuất từng phần tử trong list nhỏ ta có gọi index tương ứng với index theo hàng và cột như 1 mảng 2 chiều.

Cú pháp để gọi là: list_name[index_ hàng][index_cột]

The screenshot shows the Spyder IDE interface. At the top, there are two tabs: 'Buoi1.py X' and 'temp.py X'. The code in 'Buoi1.py' is identical to the previous one:

```
1 row1 = ['Dung', 20.5, 12345, 'Hi']
2 row2 = ['Vy', 21.2, 45673, 'Fun']
3 row3 = ['Minh', 19.8, 78532, 'Win']
4 row4 = ['Nhan', 21.5, 98337, 'Fast']
5 row = [row1, row2, row3, row4]
6
7 print(row[1][0])
```

Below the code editor is the IPython Console tab, which contains the output of the run command:

```
In [48]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py',
Vy
```

Và ta có thể ứng dụng nó để thuận tiện gọi từng phần tử trong nhiều list nhỏ ra để tính toán, in ra, thay đổi, ...

The screenshot shows the Spyder IDE interface. At the top, there are two tabs: 'Buoi1.py' (active) and 'temp.py'. Below the tabs is the code editor containing the following Python script:

```
1 row1 = ['Dung', 20.5, 12345, 'Hi']
2 row2 = ['Vy', 21.2, 45673, 'Fun']
3 row3 = ['Minh', 19.8, 78532, 'Win']
4 row4 = ['Nhan', 21.5, 98337, 'Fast']
5 row = [row1, row2, row3, row4]
6
7 avg_age = (row[0][1] + row[1][1] + row[2][1] + row[3][1]) / 4
8 print(avg_age)
```

Below the code editor is the IPython Console window, which displays the command runfile and its output:

```
In [49]: runfile('C:/Users/nguye/.spyder-py3/Buoi1.py', wdir='C:/Users/20.75
```

02 For Loops and Conditional Statements in Python

1.	Vòng lặp For trong python.....	2
1.1.	Vòng lặp For	2
1.2.	Cấu trúc vòng lặp for	2
1.3.	Mở File	3
1.4.	Một số lỗi khi sử dụng file.....	4
1.5.	Một cách khác để tính giá trị trung bình	5
2.	Các lệnh If, Else, Elif trong Python	6
2.1.	Lệnh if.....	6
2.2.	Booleans	7
2.3.	Nguyên tắc hoạt động của lệnh if	7
2.4.	Bài toán tính rating trung bình của gaming apps.....	9
3.	Lệnh If, Else, Elif : Nhiều điều kiện	10
3.1.	Nhiều điều kiện.....	10
3.2.	Toán tử logic : or	10
3.3.	Kết hợp nhiều toán tử logic	11
3.4.	Toán tử so sánh và ứng dụng.....	12
3.5.	Lệnh else	13
3.6.	Lệnh elif.....	14
3.7.	Else vs Elif.....	14

1. Vòng lặp For trong python

1.1. Vòng lặp For

Giả sử, ta có 1 list gồm n phần tử, để in ra từng phần tử ta có thể dùng vòng lặp để duyệt từng ký tự và đưa nó ra màn hình như ví dụ dưới.

The screenshot shows a Jupyter Notebook interface. The code editor window contains a file named 'Buoi2.py' with the following content:

```
list = [4, 5, 2, 9, 5, 10]
for i in list:
    print(i)
```

The IPython Console window below shows the output of running the script:

```
In [1]: runfile('C:/Users/nguye/Desktop/Buoi2.py')
4
5
2
9
5
10
```

Ta có thể dùng vòng lặp for để tính giá trị trung bình, tổng của cột, hàng... So với phương pháp thông thường, nếu ta có 1 file dữ liệu rất lớn (khoảng hơn vài ngàn dòng...) thì chúng ta không thể nào ghi từng phép tính ra được, vì thế vòng lặp rất quan trọng.

The screenshot shows a Jupyter Notebook interface. The code editor window contains a file named 'Buoi2.py' with the following content:

```
row1 = ['Vy', 20, 'Binh Dinh', 2011]
row2 = ['Trung', 23, 'Tay Ninh', 2009]
row3 = ['Chien', 17, 'Da Lat', 2010]
row4 = ['Minh', 25, 'HCM', 2006]

row = [row1, row2, row3, row4]
for i in row:
    namthamgia = i[-1]
    print(namthamgia)
```

The IPython Console window below shows the output of running the script:

```
In [5]: runfile('C:/Users/nguye/Desktop/Buoi2.py')
2011
2009
2010
2006
```

1.2. Cấu trúc vòng lặp for

Cấu trúc cơ bản của 1 vòng lặp for là:

for biến lặp in biến có thể lặp:

nội dung vòng lặp

Ví dụ như hình bên dưới.

script

```
a_list = [1, 3, 5]
for value in a_list:
    print(value)
    print(value - 1)
```

Iteration variable
Iterable variable
Body of the loop

Output

```
1
0
3
2
5
4
```

Ở đây ta áp dụng vòng lặp vô 1 bài tính tổng đơn giản:

Cụ thể ở đây i được dùng là biến để duyệt từng phần tử trong list, i dựa theo index của list. row1 là list mà ta muốn lặp. Chúng ta cần khai báo trước 1 biến = 0 và dùng biến đó để cộng với từng phần tử trong list. Vòng for sẽ có nhiệm vụ duyệt từng phần tử trong list đó bằng cách lấy biến của từng i (index).

The screenshot shows a Jupyter Notebook interface. The top part displays the code in a file named `Buoi2.py`:

```
1 row1 = [10, 30, 18, 48, 29]
2 sum_row1=0
3 for i in row1:
4     sum_row1 = i + sum_row1
5     print(sum_row1)
6 print (sum_row1)
```

The bottom part shows the output in the IPython Console:

```
In [8]: runfile('C:/Users/nguye/Desktop/Buoi2.py')
In [9]: runfile('C:/Users/nguye/Desktop/Buoi2.py')
10
40
58
106
135
135
```

1.3. Mở File

Ở đây ta dùng lệnh open('địa chỉ file') để mở file mà chúng ta cần xử lý. Tiếp theo ta có thể dùng hàm reader của thư viện csv để đọc và sau đó dùng list() để đưa dữ liệu đọc được về thành từng list nhỏ để dễ sử dụng.

The screenshot shows a Jupyter Notebook interface with two main sections: a code editor and an IPython Console.

Code Editor (Buoi2.py):

```
1 opened = open(r"C:\Users\nguye\Desktop\AppleStore.csv",encoding="utf8")
2 from csv import reader
3 read_file = reader(opened)
4 apps_data = list(read_file)
5 print (len(apps_data))
6 print ('\n')
7 print (apps_data[0])
8 print ('\n')
9 print(apps_data[1:3])
```

IPython Console:

```
In [20]: runfile('C:/Users/nguye/Desktop/Buoi2.py', wdir='C:/Users/nguye/Desktop')
7198

['id', 'track_name', 'size_bytes', 'currency', 'price', 'rating_count_tot',
'rating_count_ver', 'user_rating', 'user_rating_ver', 'ver', 'cont_rating', 'prime_genre',
'sup_devices.num', 'ipadSc_urls.num', 'lang.num', 'vpp_lic']

[[284882215, 'Facebook', '389879808', 'USD', '0', '2974676', '212', '3.5', '3.5', '95',
'4+', 'Social Networking', '37', '1', '29', '1'], [389801252, 'Instagram', '113954816',
'USD', '0', '2161558', '1289', '4.5', '4', '10.23', '12+', 'Photo & Video', '37', '0', '29',
'1]]
```

1.4. Một số lỗi khi sử dụng file

Buoi2.py

```
1 opened = open(r"C:/Users/nguye/Desktop/AppleStore.csv",encoding='utf-8')
2 from csv import reader
3 read_file = reader(opened)
4 apps_data = list(read_file)
5
6 header = apps_data[0]
7 app = apps_data[1:]
8 sum_rate=0
9 for i in app:
10     rate = i[7]
11     sum_rate += rate
```

IPython Console

Console 1/A

```
In [21]: runfile('C:/Users/nguye/Desktop/Buoi2.py', wdir='C:/Users/nguye/Desktop')
Traceback (most recent call last):

  File "C:/Users/nguye/Desktop/Buoi2.py", line 11, in <module>
    sum_rate += rate

TypeError: unsupported operand type(s) for +=: 'int' and 'str'
```

Khi ta lấy dữ liệu từ file ra, các phần tử được đưa về dạng str nên khi không ép kiểu lại thì python sẽ không thể tính toán giữa 2 kiểu dữ liệu khác nhau được

Buoi2.py

```
1 opened = open(r"C:/Users/nguye/Desktop/AppleStore.csv",encoding='utf-8')
2 from csv import reader
3 read_file = reader(opened)
4 apps_data = list(read_file)
5
6 header = apps_data[0]
7 app = apps_data[1:]
8 sum_rate=0
9 for i in app:
10     rate = float(i[7])
11     sum_rate += rate
12 print(sum_rate)
13 avg_rate = sum_rate / len(app)
14 print(avg_rate)
```

IPython Console

Console 1/A

```
In [26]: runfile('C:/Users/nguye/Desktop/Buoi2.py', wdir='C:/Users/nguye/Desktop')
25383.5
3.526955675976101
```

1.5. Một cách khác để tính giá trị trung bình

The screenshot shows a Jupyter Notebook interface. The top part is a code editor window titled "Buoi2.py*" containing Python code to calculate average app ratings from a CSV file. The bottom part is an "IPython Console" window showing the execution of the script and its output.

```

1  opened = open(r"C:/Users/nguye/Desktop/AppL
2  from csv import reader
3  read_file = reader(opened)
4  apps_data = list(read_file)
5  header = apps_data[0]
6  app = apps_data[1:]
7  sum_rate = []
8
9  for i in app:
10    rate = float(i[7])
11    sum_rate.append(rate)
12
13 avg_rate = sum(sum_rate) / len(app)
14 print(avg_rate)

```

In [30]: runfile('C:/Users/nguye/Desktop/Buoi2.py', 3.526955675976101)

Chúng ta có thể áp dụng những kỹ thuật kết hợp với 1 số lệnh sau để giải quyết bài toán:

Lệnh `list_name.append()` dùng để thêm 1 phần tử mới mà chúng ta muốn vô cuối list đó

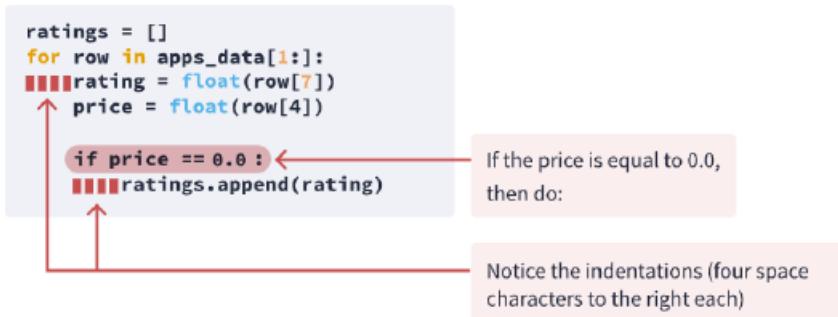
Lệnh `sum()` để tính tổng các số (kiểu int hay float) có trong list

Lưu ý khi sử dụng file, thông thường hàng đầu tiên là index của file, chúng ta không cần xử lý nó, phải xem dữ liệu bắt đầu từ hàng nào, cột nào.

2. Các lệnh If, Else, Elif trong Python

2.1. Lệnh if

Lệnh if dùng để so sánh các biến, thực hiện rẽ nhánh, kiểm tra điều kiện...



Lưu ý là trong python, python sẽ nhận dạng body của các vòng lặp, lệnh if bằng những khoảng trắng trước nó nên khi lập trình ta phải thụt bằng nhau (thường là 4 khoảng trắng) để python có thể đâu là body của lệnh , hàm gì.

The screenshot shows a Jupyter Notebook interface. The top part displays the code in a file named 'Buoi2.py':

```
1 opened = open(r"C:\Users\nguye\Desktop\Apple Store - Sales by Category.csv")
2 from csv import reader
3 read_file = reader(opened)
4 apps_data = list(read_file)
5
6 free_app_rate = []
7 for i in apps_data[1:]:
8     rate = float(i[7])
9     price= float(i[4])
10    if price == 0.0:
11        free_app_rate.append(rate)
12 avg = sum(free_app_rate)/len(free_app_rate)
13 print(avg)
```

The bottom part shows the output of running the code in the IPython Console:

```
In [31]: runfile('C:/Users/nguye/Desktop/Buoi2.py', 3.3767258382642997
```

2.2. Booleans

Trong khoa học máy tính, Boolean (đôi khi được rút ngắn thành Bool) là kiểu dữ liệu có một trong hai giá trị có thể có (thường được ký hiệu là true và false) nhằm biểu diễn hai giá trị chân lý của logic và đại số Boolean.

The screenshot shows a Jupyter Notebook interface. The top part displays the code in a file named 'untitled1.py*':

```
1 a = 1
2 print(a == 0)
3 print(a == 1)
```

The bottom part shows the output of running the code in the IPython Console:

```
In [33]: runfile('C:/Users/nguye/Desktop/untitled1.py'
False
True
```

Dấu == ở đây dùng để so sánh giữa 2 giá trị có bằng nhau không, nếu bằng trả về true, nếu sai thì false.

2.3. Nguyên tắc hoạt động của lệnh if

Lệnh if sẽ được thực hiện khi giá trị boolean được trả về là true trong các phép so sánh. Nếu giá trị là false, chương trình sẽ bỏ qua và không thực hiện những trong body của if.

The screenshot shows a Jupyter Notebook interface. At the top, there are two tabs: 'Buoi2.py X' and 'untitled1.py* X'. The code editor contains the following Python code:

```
1 if True:
2     print('Dung')
3 if False:
4     print('Sai')
```

Below the code editor is the 'IPython Console' tab, which is active. It shows the command 'In [34]: runfile('C:/Users/nguye/Desktop/untitled1.py')' followed by the output 'Dung'.

Dấu != để so sánh xem 2 giá trị đó có khác nhau không, nếu khác là true, còn nếu bằng sẽ trả về false.

The screenshot shows a Jupyter Notebook interface. At the top, there are two tabs: 'Buoi2.py X' and 'untitled1.py* X'. The code editor contains the following Python code:

```
1 print (2!=0)
2 print (2!=2)
```

Below the code editor is the 'IPython Console' tab, which is active. It shows the command 'In [35]: runfile('C:/Users/nguye/Desktop/untitled1.py')' followed by the output 'True' and 'False' on separate lines.

Áp dụng vô bài toán xử lý file, ta có thể dùng để so sánh các giá trị đúng mà thực hiện những phép tính, phép đếm...

```

1  opened = open(r"C:/Users/nguye/Desktop/AppleStore.csv")
2  from csv import reader
3  read_file = reader(opened)
4  apps_data = list(read_file)
5
6  non_free_app_rate = []
7  for i in apps_data[1:]:
8      rate = float(i[7])
9      price= float(i[4])
10     if price != 0.0:
11         non_free_app_rate.append(rate)
12 avg = sum(non_free_app_rate)/len(non_free_app_rate)
13 print(avg)

```

In [36]: runfile('C:/Users/nguye/Desktop/Buoi2.py', wdir='C:/Users/nguye/Desktop')
3.720948742438714

2.4. Bài toán tính rating trung bình của gaming apps

```

1  opened = open(r"C:/Users/nguye/Desktop/AppleStore.csv")
2  from csv import reader
3  read_file = reader(opened)
4  apps_data = list(read_file)
5
6  game_rate = []
7  for i in apps_data[1:]:
8      rate = float(i[7])
9      genre = i[11]
10     if genre == 'Games':
11         game_rate.append(rate)
12 avg = sum(game_rate)/len(game_rate)
13 print(avg)

```

In [38]: runfile('C:/Users/nguye/Desktop/Buoi2.py', wdir='C:/Users/nguye/Desktop')
3.6850077679958573

Áp dụng cú thẽ trong 2 ví dụ trên, ta có thẽ thấy phép so sánh == hoặc != không chỉ so sánh các số mà còn có thẽ so sánh 2 string với nhau.

The screenshot shows a Jupyter Notebook interface. The top tab bar has two tabs: 'Buoi2.py X' and 'untitled1.py* X'. The main area contains Python code to calculate the average rating of non-game applications from a CSV file. The code uses a for loop to iterate through the data, filtering out rows where the genre is 'Games'. It then calculates the average of the remaining ratings. The output cell below the code shows the result: 3.343928035982009.

```
1 opened = open(r"C:/Users/nguye/Desktop/Apple Store - Sheet1.csv", "r")
2 from csv import reader
3 read_file = reader(opened)
4 apps_data = list(read_file)
5
6 non_game_rate = []
7 for i in apps_data[1:]:
8     rate = float(i[7])
9     genre = i[11]
10    if genre != 'Games':
11        non_game_rate.append(rate)
12 avg = sum(non_game_rate)/len(non_game_rate)
13 print(avg)
```

IPython Console

Console 1/A X

In [37]: runfile('C:/Users/nguye/Desktop/Buoi2.py', 3.343928035982009)

3. Lệnh If, Else, Elif : Nhiều điều kiện

3.1. Nhiều điều kiện

Khi chúng ta cần 1 lúc nhiều điều kiện, chúng ta có thể dùng toán tử and để kết nối những điều kiện lại, lệnh and cũng dựa trên kết quả được trả về của booleans.

Như hình dưới là các trường hợp lệnh and sẽ gặp và giá trị mà chúng trả về.

The screenshot shows a Jupyter Notebook interface. The top tab bar has two tabs: 'Buoi2.py X' and 'untitled1.py* X'. The main area contains Python code that prints four boolean values: (True and True), (True and False), (False and True), and (False and False). The output cell below the code shows the results: True, False, False, and False respectively.

```
1 print (True and True)
2 print (True and False)
3 print (False and True)
4 print (False and False)
```

IPython Console

Console 1/A X

In [39]: runfile('C:/Users/nguye/Desktop/untitled1.py')

True
False
False
False

3.2. Toán tử logic : or

Ngoài ra ta cũng có toán or dùng khi ta có nhiều điều kiện, nhưng chỉ cần 1 điều kiện đúng là đủ. Or cũng như and, cũng sử dụng giá trị boolean được trả về.

Như hình dưới là các trường hợp lệnh or sẽ gặp và giá trị mà chúng trả về.

The screenshot shows a Jupyter Notebook interface. At the top, there are two tabs: 'Buoi2.py X' and 'untitled1.py X'. The 'untitled1.py' tab is active. Below the tabs is a code editor containing four lines of Python code:

```
1 print (True or True)
2 print (True or False)
3 print (False or True)
4 print (False or False)
```

Below the code editor is an 'IPython Console' section. It shows the output of the code execution:

```
In [2]: runfile('C:/Users/nguye/Desktop/untitled1.py')
True
True
True
False
```

3.3. Kết hợp nhiều toán tử logic



Khi gặp bài toán cần nhiều chọn lọc những điều kiện khác nhau, ta có thể kết hợp nhiều toán tử logic lại để chọn ra những điều kiện cần thiết.

```

1  opened = open(r"C:/Users/nguye/Desktop/AppleStore.csv",encoding="utf8")
2  from csv import reader
3  read_file = reader(opened)
4  apps_data = list(read_file)
5
6  not_free_games = []
7  for i in apps_data[1:]:
8      rate = float(i[7])
9      genre = i[11]
10     price = float(i[4])
11     if (genre == 'Games' or genre == 'Social Networking') and price != 0:
12         not_free_games.append(rate)
13 avg = sum(not_free_games)/len(not_free_games)
14 print(avg)

```

In [4]: runfile('C:/Users/nguye/Desktop/Buoi2.py', wdir='C:/Users/nguye/Desktop')
3.8904235727440146

3.4. Toán tử so sánh và ứng dụng

Comparison (text)	Comparison (operator)	Comparison (code)
A is equal to B	==	A == B
A is not equal to B	!=	A != B
A is greater than B	>	A > B
A is greater than or equal to B	>=	A >= B
A is less than B	<	A < B
A is less than or equal to B	<=	A <= B

Bên cạnh những toán tử logic, chúng ta cũng có những toán tử để so sánh những số.

The screenshot shows a Jupyter Notebook interface. The top part displays the code in `untitled1.py`:

```
1 x = 10
2 y = - 9.84
3 z = 4.6
4
5 if z > x:
6     print ('z lon hon x')
7 if y != x:
8     print('y khong bang x')
9 if x>=z or y <= x:
10    print(True)
```

The bottom part shows the output in the IPython Console:

```
In [5]: runfile('C:/Users/nguye/Desktop/untitled1.py')
y khong bang x
True
```

3.5. Lệnh else

Một lệnh else có thể được sử dụng kết hợp với lệnh if. Một lệnh else chứa khối code mà thực thi nếu biểu thức điều kiện trong lệnh if được ước lượng là 0 hoặc một giá trị false. Lệnh else là lệnh tùy ý và chỉ có duy nhất một lệnh else sau lệnh if.

The screenshot shows a Jupyter Notebook environment. The top navigation bar has two tabs: 'Buoi2.py X' and 'untitled1.py X'. The main code cell (cell 14) contains the following Python script:

```
1 opened = open(r"C:\Users\nguye\Desktop\AppleStore.csv",encoding="utf8")
2 from csv import reader
3 read_file = reader(opened)
4 apps_data = list(read_file)
5
6 for i in apps_data[1:]:
7     price = float(i[4])
8     if price == 0.0:
9         i.append('free')
10    else:
11        i.append('not free')
12
13 apps_data[0].append('free or not')
14 print(apps_data[:6])
15
```

The output of cell 14 is displayed in the IPython Console below:

```
In [7]: runfile('C:/Users/nguye/Desktop/Buoi2.py', wdir='C:/Users/nguye/Desktop')
[['id', 'track_name', 'size_bytes', 'currency', 'price', 'rating_count_tot', 'rating_count_ver', 'user_rating', 'user_rating_ver', 'ver', 'cont_rating', 'prime_genre', 'sup_devices.num', 'ipadSc_urls.num', 'lang.num', 'vpp_lic', 'free or not'], ['284882215', 'Facebook', '389879808', 'USD', '0', '2974676', '212', '3.5', '3.5', '95', '4+', 'Social Networking', '37', '1', '29', '1', 'free'], [389801252, 'Instagram', '113954816', 'USD', '0', '2161558', '1289', '4.5', '4', '10.23', '12+', 'Photo & Video', '37', '0', '29', '1', 'free'], [529479190, 'Clash of Clans', '116476928', 'USD', '0', '2130805', '579', '4.5', '4.5', '9.24.12', '9+', 'Games', '38', '5', '18', '1', 'free'], [420009108, 'Temple Run', '65921024', 'USD', '0', '1724546', '3842', '4.5', '4', '1.6.2', '9+', 'Games', '40', '5', '1', '1', 'free'], [284035177, 'Pandora - Music & Radio', '130242560', 'USD', '0', '1126879', '3594', '4', '4.5', '8.4.1', '12+', 'Music', '37', '4', '1', '1', 'free']]
```

3.6. Lệnh elif

Lệnh elif cho phép bạn kiểm tra nhiều điều kiện và thực thi khối code ngay khi một trong các điều kiện được ước lượng là true.

The screenshot shows a Jupyter Notebook interface. The top tab bar has two tabs: 'Buoi2.py X' and 'untitled1.py X'. The main code editor area contains Python code to calculate the average rating of four apps and categorize them as 'Below avg', 'Roughly avg', or 'Better than avg'. The code is as follows:

```
1 x = [ ['Facebook',3.7], ['Messenger', 3.1], ['Zalo',4.0], ['Telegram', 4.8]]
2
3 avg = (3.9 + 4.1 + 3.5 + 4.4)/4
4
5 for app in x:
6     rate = app[1]
7     if rate <= avg-0.5:
8         app.append('Below avg')
9     elif rate > avg-0.5 and rate <= avg+0.5:
10        app.append('Roughly avg')
11    elif rate > avg+0.5:
12        app.append('Better than avg')
13
14 print(x)
```

The code is run in the IPython Console, which outputs the result:

```
In [9]: runfile('C:/Users/nguye/Desktop/untitled1.py', wdir='C:/Users/nguye/Desktop')
[['Facebook', 3.7, 'Roughly avg'], ['Messenger', 3.1, 'Below avg'], ['Zalo', 4.0, 'Roughly avg'],
 ['Telegram', 4.8, 'Better than avg']]
```

3.7. Else vs Elif

Lệnh elif khác với else là sau lệnh if có thể có nhiều elif nhưng chỉ có 1 else. Lệnh elif cuối cùng có thể thay thế bằng else.

03 Dictionaries, Frequency Tables, and Functions in Python

1.	Dictionary trong python	2
1.1.	Dictionaries.....	2
1.2.	Indexing	2
1.3.	Thêm phần tử vào dictionary	2
1.4.	Cập key và giá trị của nó	2
2.	Dictionaries và bảng tần số	4
2.1.	Kiểm tra phần tử tồn tại trong dic.....	4
2.2.	Cập nhật giá trị của phần tử trong dic	4
2.3.	Tìm kiếm giá trị duy nhất	4
2.4.	Proportions và Percentages.....	5
2.5.	Dùng vòng lặp cho dictionary	5
2.6.	Giữ lại các phần tử của dictionary	6
2.7.	Bảng Frequency cho Numerical Columns.....	6
2.8.	Lọc theo intervals	6
3.	Python Functions: Sự dụng Built-in Functions, khởi tạo Functions.....	7
3.1.	Functions:	7
3.2.	Built-InFunctions	8
3.3.	Tạo Functions riêng cho bản thân	8
3.4.	Cấu trúc của 1 function.....	8
3.6.	Lệnh return.....	9
4.	Python Functions: Arguments, Parameters, và Debugging	10
4.1.	Lấy dữ liệu từ bất kỳ cột nào	10
4.2.	Tạo bảng frequency	10
4.3.	Viết 1 hàm độc lập	10

1. Dictionary trong python

1.1. Dictionaries

```
Buoi3.py X
1 num = {1: 'mot', 2: 'hai', 3: 'ba', 4 : 'bon'}
2 print(num)

IPython Console
Console 1/A X
In [1]: runfile('C:/Users/nguye/Desktop/Buoi3.py', wdir='C:/Users/nguye/Desktop')
{1: 'mot', 2: 'hai', 3: 'ba', 4: 'bon'}
```

1.2. Indexing

```
Buoi3.py X
1 list_num = [2002, 'Minh', 'UTE', 26.6 ]
2 #           nam   ten    truong diem
3 print(list_num[0])
4 dic_num = {'nam':2002, 'ten':'Minh', 'truong':"UTE", 'diem':26.6}
5 print(dic_num['nam'])

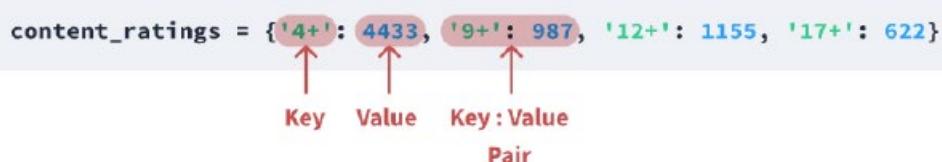
IPython Console
Console 1/A X
In [3]: runfile('C:/Users/nguye/Desktop/Buoi3.py', wdir='C:/Users/nguye/Desktop')
2002
2002
```

1.3. Thêm phần tử vào dictionary

```
Buoi3.py X
1 num ={ }
2 num[1]='mot'
3 num[2]='hai'
4 num[3]='ba'
5 print(num)

IPython Console
Console 1/A X
In [4]: runfile('C:/Users/nguye/Desktop/Buoi3.py',
{1: 'mot', 2: 'hai', 3: 'ba'})
```

1.4. Cập key và giá trị của nó



BuoI3.py

```
1 print(hash(4))
2 print(hash('four'))
3 print(hash(True))
```

IPython Console

Console 1/A

```
In [5]: runfile('C:/Users/nguye/Desktop/BuoI3.py',
4
7315452643962034845
1
```

BuoI3.py

```
1 dic1 = {1:'one', True: 'Boolean'}
2 dic2 = {0:'zero', True:'Boolean', False:'Bool', 1:'one', 2:'two'}
3 print(dic1)
4 print(dic2)
```

IPython Console

Console 1/A

```
In [10]: runfile('C:/Users/nguye/Desktop/BuoI3.py', wdir='C:/Users/nguye/Desktop')
{1: 'Boolean'}
{0: 'Bool', True: 'one', 2: 'two'}
```

BuoI3.py

```
1 dic1 = {1:'one', 'hai':2, 3:[1,2,3,4,5], 'bon':{1:10,2:20}}
2 print(dic1)
```

IPython Console

Console 1/A

```
In [11]: runfile('C:/Users/nguye/Desktop/BuoI3.py', wdir='C:/Users/nguye/Desktop')
{1: 'one', 'hai': 2, 3: [1, 2, 3, 4, 5], 'bon': {1: 10, 2: 20}}
```

2. Dictionaries và bảng tần số

2.1. Kiểm tra phần tử tồn tại trong dic

```
Buoi3.py
1 dic1 = {1: 'one', 'hai':2, 3:[1,2,3,4,5], 'bon':{1:10,2:20}}
2 print(1 in dic1)

IPython Console
Console 1/A
In [12]: runfile('C:/Users/nguye/Desktop/Buoi3.py', wdir='C:/Users/
True
```

2.2. Cập nhật giá trị của phần tử trong dic

```
Buoi3.py
1 dic1 = {1: 'one', 'hai':2, 3:[1,2,3,4,5], 'bon':{1:10,2:20}}
2 dic1['hai'] = 10
3 print(dic1)

IPython Console
Console 1/A
In [13]: runfile('C:/Users/nguye/Desktop/Buoi3.py', wdir='C:/Users/
{1: 'one', 'hai': 10, 3: [1, 2, 3, 4, 5], 'bon': {1: 10, 2: 20}}
```

2.3. Tìm kiếm giá trị duy nhất

```
Buoi3.py
1 opened = open(r"C:\Users\nguye\Desktop\Appl
2 from csv import reader
3 read_file = reader(opened)
4 apps_data = list(read_file)
5
6 genre_count = {}
7 for i in apps_data[1:]:
8     genre = i[10]
9     if genre in genre_count:
10         genre_count[genre] += 1
11     else:
12         genre_count[genre] = 1
13 print(genre_count)

IPython Console
Console 1/A
In [16]: runfile('C:/Users/nguye/Desktop/Buoi3.py',
Desktop')
{'4+': 4433, '12+': 1155, '9+': 987, '17+': 622}
```

2.4. Proportions và Percentages

Content rating	Number of apps (frequency)
4+	4,433
9+	987
12+	1,155
17+	622

2.5. Dùng vòng lặp cho dictionary

The screenshot shows a Jupyter Notebook interface with two tabs: 'Buoi3.py' and 'untitled2.py*'. The 'untitled2.py*' tab is active, displaying the following Python code:

```
1 content_rate = {'4+':4433, '9+':987, '12+':1155, '17':622}
2 total = 7197
3 for i in content_rate:
4     content_rate[i] /= total
5 print(content_rate)
```

Below the code, the 'IPython Console' tab is active, showing the output of the code execution:

```
In [17]: runfile('C:/Users/nguye/Desktop/untitled2.py', wdir='C:/Users/nguye/Desktop')
{'4+': 0.6159510907322495, '9+': 0.13714047519799916, '12+': 0.16048353480616923, '17': 0.08642489926358204}
```

2.6. Giữ lại các phần tử của dictionary

```
Buoi3.py X untitled2.py* X
1 content_rate = {'4+':4433, '9+':987, '12+':1155, '17':622}
2 total = 7197
3 rate_pro = {}
4 for i in content_rate:
5     pro = content_rate[i] / total
6     rate_pro[i] = pro
7
8 rate_per = {}
9 for i in rate_pro:
10    per = rate_pro[i]*100
11    rate_per[i] = per

Variable Explorer
```

Name	Type	Size	Value
content_rate	dict	4	{'4+':4433, '9+':987, '12+':1155, '17':622}
i	str	2	17
per	float	1	8.642489926358204
pro	float	1	0.08642489926358204
rate_per	dict	4	{'4+':61.59510907322495, '9+':13.714047519799916, '12+':16.0483534806 ...}
rate_pro	dict	4	{'4+':0.6159510907322495, '9+':0.13714047519799916, '12+':0.1604835348 ...}
total	int	1	7197

2.7. Bảng Frequency cho Numerical Columns

```
Buoi3.py X untitled2.py* X
1 opened = open(r"C:\Users\nguye\Desktop\AppleStore.csv",encoding="utf
2 from csv import reader
3 read_file = reader(opened)
4 apps_data = list(read_file)
5
6 data_sizes = []
7 for i in apps_data[1:]:
8     size = float(i[2])
9     data_sizes.append(size)
10
11 min_size = min(data_sizes)
12 max_size = max(data_sizes)

Variable Explorer
```

Name	Type	Size	Value
apps_data	list	7198	[['id', 'track_name', 'size_bytes', 'currency', 'price', ...], ['28488 ...
data_sizes	list	7197	[389879808.0, 113954816.0, 116476928.0, 65921024.0, 130242560.0, 74778 ...
i	list	16	['977965019', 'みんなのお弁当 by クックパッド ~お弁当をレシピ付きで記録・共有~', '51174400', 'USD' ...]
max_size	float	1	4025969664.0
min_size	float	1	589824.0
opened	TextIOWrapper	1	TextIOWrapper object of _io module
read_file	reader	1	reader object of _csv module
size	float	1	51174400.0

2.8. Lọc theo intervals

```
1  opened = open(r"C:\Users\nguye\Desktop\AppleStore.csv",encoding="utf8")
2  from csv import reader
3  read_file = reader(opened)
4  apps_data = list(read_file)
5  data_sizes = {'0-10mb':0, '10-50mb':0, '50-100mb':0, '100-500mb':0, '500mb+':0}
6  for i in apps_data[1:]:
7      data_sizes=float(i[2])
8      if data_sizes <=10000000:
9          data_sizes['0-10mb']+1
10     elif 10000000 < data_sizes <= 50000000:
11         data_sizes['10-50mb']+1
12     elif 50000000 < data_sizes <= 100000000:
13         data_sizes['50-100mb']+1
14     elif 100000000 < data_sizes <= 500000000:
15         data_sizes['100-500mb']+1
16     elif data_sizes > 500000000:
17         data_sizes['500mb+']+1
18  print(data_sizes)
{'0 - 10 MB': 285, '10 - 50 MB': 1639, '50 - 100MB': 1778, '100 - 500MB': 2894, '500MB + ': 601}
```

3. Python Functions: Sự dụng Built-in Functions, khởi tạo Functions

3.1. Functions:

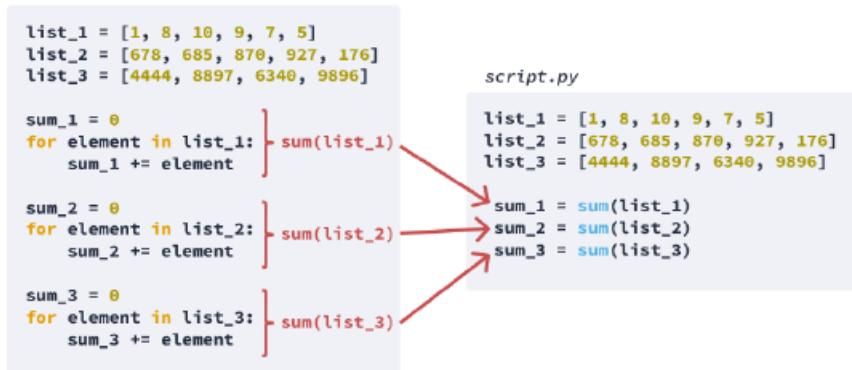
Một function hoạt động như sau:

- Đưa vào input
- Xử lý input
- Trả về output

Python hỗ trợ người dùng built-in functions vd như : print(), sum(), len(),

max()

3.2. Built-In Functions



3.3. Tạo Functions riêng cho bản thân

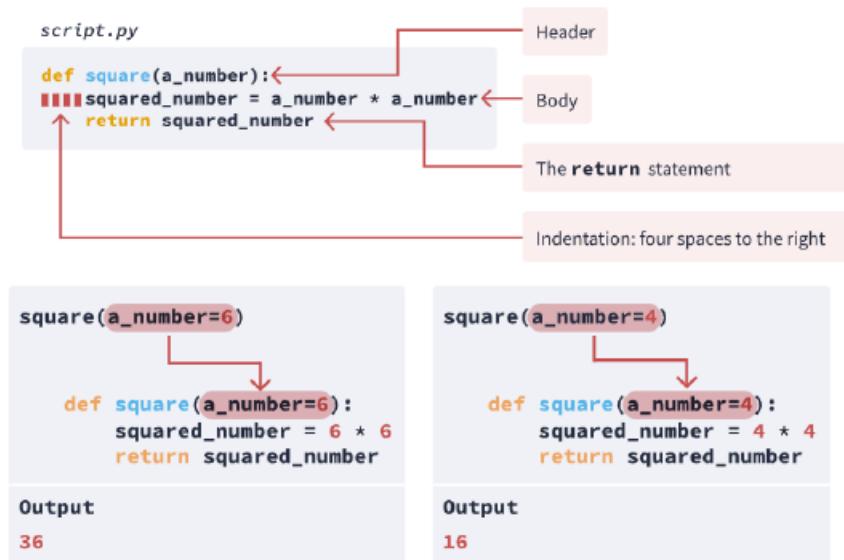
```
BuoI3.py X untitled2.py* X untitled

1 def hammu2(x):
2     kq = x*x
3     return kq
4 binh_6 =hammu2(6)
5 binh_7 =hammu2(7)
6 binh_100 =hammu2(100)

Variable Explorer
```

Name	Type	Size
binh_6	int	1
binh_7	int	1
binh_100	int	1

3.4. Cấu trúc của 1 function



3.5. Parameter và Argument

Vd ở đây:

- x làm tham số
- 6, 7 là đối số

```
1 def hammu2(x):
2     kq = x*x
3     return kq
4
5 binh_6 =hammu2(6)
6 binh_7 =hammu2(7)
```

3.6. Lệnh return

Lệnh return dùng để trả về giá trị sau khi thực hiện các lệnh trong hàm

```
1 def hammu2(x):
2     kq = x*x
3     return kq
4
5 binh_6 =hammu2(6)
6 binh_7 =hammu2(7)
```

Variable Explorer

Name	Type	Size	
binh_6	int	1	36
binh_7	int	1	49

4. Python Functions: Arguments, Parameters, và Debugging

4.1. Lấy dữ liệu từ bất kỳ cột nào

The screenshot shows a Jupyter Notebook interface with three tabs: Buoi3.py, untitled2.py*, and untitled3.py*. The Buoi3.py tab contains the following Python code:

```
1 opened = open(r"C:/Users/nguye/Desktop/AppleStore.csv",encoding="utf
2 from csv import reader
3 read_file = reader(opened)
4 apps_data = list(read_file)
5
6 def extract(i):
7     column = []
8     for row in apps_data[1:]:
9         value = row[i]
10        column.append(value)
11    return column
12 genres = extract(11)
```

The Variable Explorer table shows the following data:

Name	Type	Size	Value
apps_data	list	7198	[['id', 'track_name', 'size_bytes', 'currency', 'price', ...], ['28488 ...
genres	list	7197	['Social Networking', 'Photo & Video', 'Games', 'Games', 'Music', 'Soc ...
opened	TextIOWrapper	1	TextIOWrapper object of _io module
read_file	reader	1	reader object of _csv module

4.2. Tạo bảng frequency

The screenshot shows a Jupyter Notebook interface with three tabs: Buoi3.py, untitled2.py*, and untitled3.py*. The Buoi3.py tab contains the following Python code:

```
1 opened = open(r"C:/Users/nguye/Desktop/AppleStore.csv",encoding="utf
2 from csv import reader
3 read_file = reader(opened)
4 apps_data = list(read_file)
5
6 def extract(i):
7     column = []
8     for row in apps_data[1:]:
9         value = row[i]
10        column.append(value)
11    return column
12 genres = extract(11)
13 def freq(cot):
14     freq={}
15     for value in cot:
16         if value in freq:
17             freq[value]+=1
18         else:
19             freq[value]=1
20     return freq
21 genres_ft = freq(genres)
```

The Variable Explorer table shows the following data:

Name	Type	Size	Value
apps_data	list	7198	[['id', 'track_name', 'size_bytes', 'currency', 'price', ...], ['28488 ...
genres	list	7197	['Social Networking', 'Photo & Video', 'Games', 'Games', 'Music', 'Soc ...
genres_ft	dict	23	{'Social Networking':167, 'Photo & Video':349, 'Games':3862, 'Music':1 ...
opened	TextIOWrapper	1	TextIOWrapper object of _io module
read_file	reader	1	reader object of _csv module

4.3. Viết 1 hàm đọc lặp

The screenshot shows a Jupyter Notebook interface with three tabs at the top: 'Buoi3.py X', 'untitled2.py* X', and 'untitled3.py* X'. The main area displays Python code for reading a CSV file and calculating genre frequencies. A variable explorer sidebar shows the current state of variables.

```
1 opened = open(r"C:\Users\nguye\Desktop\AppleStore.csv",encoding="utf
2 from csv import reader
3 read_file = reader(opened)
4 apps_data = list(read_file)
5 def freq(cot):
6     freq={}
7     for row in apps_data[1:]:
8         value = row[cot]
9         if value in freq:
10             freq[value]+=1
11         else:
12             freq[value]=1
13     return freq
14 genres_ft = freq(7)
```

Variable Explorer

Name	Type	Size	Value
apps_data	list	7198	[['id', 'track_name', 'size_bytes', 'currency', 'price', ...], ['28488 ...
genres_ft	dict	10	{'3.5':702, '4.5':2663, '4':1626, '3':383, '5':492, '2.5':196, '2':106 ...
opened	TextIOWrapper	1	TextIOWrapper object of _io module
read_file	reader	1	reader object of _csv module

04 Python Functions

1. Python Functions: Built-in Functions and Multiple Return Statements.....	2
1.1. Interfering with the Built-In Functions	2
1.2. Variable Names and Built-In Functions	3
1.3. Default Arguments	3
1.4. Multiple Return Statements.....	4
1.5. Not Using the else Clause	5
2. Python Functions: Returning Multiple Variables and Function Scopes....	6
2.1. Returning Multiple Variables.....	6
2.2. Tuples	6
2.3. More About Tuples	9
2.4. No Return Statement.....	10
2.5. Function Behavior	11
2.6. Scopes — Global and Local.....	11

1. Python Functions: Built-in Functions and Multiple Return Statements

1.1. Interfering with the Built-In Functions

```
def find_sum(a_list):
    a_sum = 0
    for element in a_list:
        a_sum += float(element)
    return a_sum
list_1 = [5, 10, 15]
print (find_sum(list_1))
```

Output:

30.0

```
def sum(a_list):
    a_sum = 0
    for element in a_list:
        a_sum += float(element)
    return a_sum
list_1 = [5, 10, 15]
print (sum(list_1))
```

Output:

30.0

```
def sum(a_list):
    a_string = "This function donen't really return the sum"
    return a_string
list_1 = [5,10,15]
print(sum(list_1))
```

Output:

This function donen't really return the sum

```
list= [1,3,5,6,6]
max_val_test_0 = max(list)
print(max_val_test_0)
print (max(list))
def max(list):
    str='No max value returned'
    return str
```

Output:

6

6

1.2. Variable Names and Built-In Functions

```
sum=5+12  
print (sum)
```

Output:
17

```
sum=5+12  
list_1=[5, 10, 15]  
print (sum(list_1))
```

TypeError: 'int' object is not callable

```
sum=5+12  
list_1=[5, 10, 15]  
print (17(list_1))
```

TypeError: 'int' object is not callable

1.3. Default Arguments

```
def add_val(x, constant=10):  
    return x+ constant  
print (add_val(3))
```

Output:
13

```
def add_val(x, constant=10):  
    return x+ constant  
print (add_val())
```

TypeError: add_val() missing 1 required positional argument: 'x'

```
def add_val(x, constant=10):  
    return x+ constant  
print (add_val(3, constant=50))  
print (add_val(3, constant=26))  
print (add_val(3, constant=97))
```

Output:
53
29
100

```
def add_val(x=9, constant=10):
    return x+ constant
print (add_val())
```

Output:
19

```
def open_data (file_name = 'AppleStore.csv'):
    opened_file = open(file_name,encoding="utf8")
    from csv import reader
    read_file= reader(opened_file)
    data = list(read_file)
    return data
apps_data = open_data()
```

Output:

```
<class 'function'>
<function open_data at 0x0000019C2BCFA9D0>
<class 'list'>
[['id', 'track_name', 'size_bytes', 'currency', 'price', 'rating_count_tot', 'rating_count_ver',
'user_rating', 'user_rating_ver', 'ver', 'cont_rating', 'prime_genre', 'sup_devices.num',
'ipadSc_urls.num', 'lang.num', 'vpp_lic'], ['284882215', 'Facebook', '389879808', 'USD', '0.0',
'2974676', '212', '3.5', '3.5', '95.0', '4+', 'Social Networking', '37', '1', '29', '1'],
```

1.4. Multiple Return Statements

```
def sum_or_difference(a,b,return_sum=True):
    if return_sum:
        return a+b
    else:
        return a-b
print(sum_or_difference(10,5,return_sum=True))
print(sum_or_difference(10,5,return_sum=False))
```

Output:
15
5

```
def open_data(file_name='AppleStore.csv', header=True):
    opened_file = open(file_name,encoding="utf8")
    from csv import reader
    read_file=reader(opened_file)
    data = list(read_file)

    if header:
        return data[1:]
    else:
        return data
```

Output:

```
<class 'function'>
<function open_data at 0x0000019C2ED4DA60>
```

1.5. Not Using the else Clause

```
def sum_or_difference(a,b,return_sum=True):
    if return_sum:
        return a+b
    return a-b
print(sum_or_difference(10,5,return_sum=True))
print(sum_or_difference(10,5,return_sum=False))
```

Output:

```
15
5
```

```
def open_data (file_name = 'AppleStore.csv',header=True):
    opened_file = open(file_name,encoding="utf8")
    from csv import reader
    read_file= reader(opened_file)
    data = list(read_file)
    if header:
        return data[1:]
```

Output:

```
<class 'function'>
<function open_data at 0x0000019C2E3B88B0>
```

2. Python Functions: Returning Multiple Variables and Function Scopes

2.1. Returning Multiple Variables

```
def sum_and_difference (a,b):  
    a_sum = a+b  
    difference=a-b  
    return a_sum, difference  
  
sum_diff= sum_and_difference(15, 5)  
print(sum_diff)
```

Output:
(20, 10)

```
def sum_and_difference (a,b):  
    a_sum = a+b  
    difference=a-b  
    return difference,a_sum  
  
sum_diff= sum_and_difference(15, 5)  
print(sum_diff)
```

Output:
(10, 20)

```
def pythagorean(a,b):  
    a_squared = a*a  
    b_squared = b*b  
    c_squared = a_squared+b_squared  
    return a_squared, b_squared, c_squared  
print(pythagorean(5, 12))
```

Output:
(25, 144, 169)

2.2. Tuples

```
def sum_and_difference (a,b):  
    a_sum = a+b  
    difference=a-b  
    return a_sum,difference  
  
sum_diff= sum_and_difference(15, 5)  
print(type(sum_diff))
```

Output:
<class 'tuple'>

```
a_list = [1, 'a', 10.5]
a_tuple = (1, 'a', 10.5)

print(a_tuple)
print(type(a_tuple))
```

Output:
(1, 'a', 10.5)
<class 'tuple'>

```
a_list = [1, 'a', 10.5]
a_tuple = (1, 'a', 10.5)

print(a_tuple[0])
print(a_list[0])
print(a_tuple[-1])
print(a_list[-1])
```

Output:
1
1
10.5
10.5

```
a_list = [1, 'a', 10.5]
a_list[0]=99
print(a_list)
```

Output:
[99, 'a', 10.5]

```
a_tuple = (1, 'a', 10.5)
a_tuple[0]=99
print(a_tuple)
```

```
TypeError: 'tuple' object does not support item assignment
```

```

def open_dataset(file_name='Applestore.csv', header=True):
    opened_file = open(file_name, encoding="utf8")
    from csv import reader
    read_file = reader(opened_file)
    data = list(read_file)

    if header:
        return data[1:], data[0]
    else:
        return data

all_data = open_dataset()
header = all_data[1]
apps_data = all_data[0]

```

☒ all_data - Tuple (2 elements)

Index	Type	Size	Value
0	list	7197	[['284882215', 'Facebook', '389879808', 'USD', '0.0', ...], ['38980125 ...
1	list	16	['id', 'track_name', 'size_bytes', 'currency', 'price', 'rating_count_ ...

☒ apps_data - List (7197 elements)

Index	Type	Size	Value
0	list	16	['284882215', 'Facebook', '389879808', 'USD', '0.0', '2974676', '212', ...
1	list	16	['389801252', 'Instagram', '113954816', 'USD', '0.0', '2161558', '1289 ...
2	list	16	['529479190', 'Clash of Clans', '116476928', 'USD', '0.0', '2130805', ...
3	list	16	['420009108', 'Temple Run', '65921024', 'USD', '0.0', '1724546', '3842 ...

☒ header - List (16 elements)

Index	Type	Size	Value
0	str	2	id
1	str	10	track_name
2	str	10	size_bytes
3	str	8	currency
4	str	5	price
5	str	16	rating_count_tot
6	str	16	rating_count_ver

2.3. More About Tuples

```
a_tuple = (1, 'a')
print(a_tuple)
print(type(a_tuple))
```

Output:
(1, 'a')
<class 'tuple'>

```
a_tuple = [1, 'a']
print(a_tuple)
print(type(a_tuple))
```

Output:
(1, 'a')
<class 'tuple'>

```
def sum_and_difference (a, b):
    a_sum = a + b
    difference = a - b
    return a_sum, difference
sum_diff = sum_and_difference(15, 5)
print(sum_diff)
```

Output:
(20, 10)

```
def sum_and_difference (a, b):
    a_sum = a + b
    difference = a - b
    return [a_sum, difference]
sum_diff = sum_and_difference(15, 5)
print(sum_diff)
print(type(sum_diff))
```

Output:
[20, 10]
<class 'list'>

```
def sum_and_difference (a, b, do_sum=True):
    a_sum = a + b
    difference = a - b
    return a_sum, difference
a_sum, a_diff = sum_and_difference(15, 5)
print(a_sum)
print(a_diff)
```

Output:
20
10

```
def open_dataset(file_name='Applestore.csv', header=True):
    opened_file = open(file_name, encoding="utf8")
    from csv import reader
    read_file = reader(opened_file)
    data = list(read_file)

    if header:
        return data[1:], data[0]
    else:
        return data
```

Output:
<class 'function'>

2.4. No Return Statement

```
def print_constant():
    x = 3.14
    print(x)
print_constant()
```

Output:
3.14

```
def print_constant():
    x = 3.14
    print(x)
j = print_constant()
print(j)
print(type(j))
```

Output:
3.14
None
<class 'NoneType'>

```
def print_constant():
    x = 3.14
    print(x)
    |
    print(x)
```

NameError: name 'x' is not defined

```
def price(item, cost):
    print("The " + item + " costs $" + str(cost) + ".")  
  
price("chair", 40.99)
price("book", 12.84)
price("soap", 3.95)
```

Output:
The chair costs \$40.99.
The book costs \$12.84.
The soap costs \$3.95.

2.5. Function Behavior

```
def divide():
    []/'abc'
print('Code finished running, but no error was returned')
```

Output:
Code finished running, but no error was returned

```
def print_constant():
    x=3.14
    print(x)  
  
print(type(print_constant))
print_constant()
print(x)
```

Output:
<class 'function'>
3.14
Traceback (most recent call last):

 File "C:\Users\nguye\.spyder-py3\untitled5.py", line 9, in <module>
 print(x)

NameError: name 'x' is not defined

2.6. Scopes — Global and Local

```
x = 10
def print_constant():
    x = 3.14
    print(x)  
  
print_constant()
print(x)
```

Output:
3.14
10

```
e = 'mathematical constant'  
a_sum = 1000  
length = 50  
  
def exponential(x):  
    e = 2.72  
    print(e)  
result = exponential(5)  
print(e)|  
  
def divide():  
    print(a_sum)  
    print(length)  
result_2 = divide()
```

Output:
2.72
mathematical constant
1000
50

LISTS AND FOR LOOPS PRACTICE PROBLEMS

1. Printing All Values	2
2. Range For Loop	2
3. Print all numbers in a range	3
4. Incrementing all elements.....	3
5. Reversing a list	3
6. Understanding Code 1	4
7. Understanding Code 2	4
8. Understanding Code 3	4
9. Understanding Code 4	5
10. Understanding Code 5	5
11. Understanding Code 6	5
12. Matrix	5
13. Matrix number of Rows and Columns.....	6
14. Adding All Matrix Entries	6
15. Calculate the Average.....	7
16. Calculate the String Lengths.....	7
17. Printing String Characters.....	7
18. Read the File Into a List of Lists.....	8
19. Adding Data to Rows.....	8
20. Convert Numerical Values.....	9

1. Printing All Values

```
lines = ["My candle burns at both ends;", 'It will not last the night;', 'But ah, my foes, and o  
for i in lines:  
    print(i)
```

Output:

```
My candle burns at both ends;  
It will not last the night;  
But ah, my foes, and oh, my friend -  
It gives a lovely light.
```

2. Range For Loop

```
for i in range (12): # N = 11  
    print(i)
```

Output:

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11
```

3. Print all numbers in a range

```
for i in range (42, 60): #print all Num between 42 -> 59
    print(i)
```

Output:

```
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
```

4. Incrementing all elements

```
values = [16,1,7,2,19,12,5,20,2,10,10,14,17,14,1,16,19,7,9,19]
for i in range(len(values)):
    values[i] += 1
print (values)
```

Output:

```
[17, 2, 8, 3, 20, 13, 6, 21, 3, 11, 11, 15, 18, 15, 2, 17, 20, 8, 10, 20]
```

5. Reversing a list

```
values = [16,1,7,2,19,12,5,20,2,10,10,14,17,14,1,16,19,7,9,19]
reversed_values= []
for i in range(len(values)-1,-1,-1):
    reversed_values.append(values[i])
print (reversed_values)
```

Output:

```
[19, 9, 7, 19, 16, 1, 14, 17, 14, 10, 10, 2, 20, 5, 12, 19, 2, 7, 1, 16]
```

6. Understanding Code 1

```
values = [5,4,7,8,9,3]
values_copy = []
for v in values:
    values_copy = v
answer = values_copy
print(answer)
```

Output:
3

7. Understanding Code 2

```
values = [3,5,2,1]
total = 0
for x in values:
    total += x
answer_x = x
print(answer_x)
answer_total = total
print(answer_total)
```

Output:
1
11

8. Understanding Code 3

```
values = [3,5,2,1]
for x in values:
    x = 0
answer = x
print(answer)
```

Output:
0

9. Understanding Code 4

```
values = [3,5,2,1]
total = 0
for x in values:
    total = x
answer_x = x
answer_total = total
print(answer_x)
print(answer_total)
```

Output:

```
1
1
```

10. Understanding Code 5

```
answer = 0
for i in range(10):
    print(i)
    i = 11
    answer+=1
print('Answer = ', answer)
```

Output:

```
0
1
2
3
4
5
6
7
8
9
Answer = 10
```

11. Understanding Code 6

```
values = [5,4,7,8,9,3]
values_copy = []
for v in values:
    values_copy.append(v)
answer = values_copy

print(answer)
```

Output:

```
[5, 4, 7, 8, 9, 3]
```

12. Matrix

```

matrix_of_ones = []
n, m = 7, 3
for i in range(0, n):
    row = []
    for j in range(0, m):
        row.append(1)
    matrix_of_ones.append(row)
print(matrix_of_ones)

```

Output:
[[1, 1, 1], [1, 1, 1], [1, 1, 1], [1, 1, 1], [1, 1, 1], [1, 1, 1], [1, 1, 1]]

13. Matrix number of Rows and Columns

```

matrix = [[0,9,5,4,5,3,1,5,7],
          [8,2,1,7,3,1,5,7,0],
          [1,5,3,2,7,1,4,4,8],
          [2,5,6,2,0,4,1,9,3],
          [7,4,2,9,7,0,7,4,4,]]
num_rows=len(matrix)
num_cols=len(matrix[0])
print('num_rows = ', num_rows)
print('num_cols = ', num_cols)

```

Output:
num_rows = 5
num_cols = 9

14. Adding All Matrix Entries

```

matrix = [[0, 9, 5, 4],
          [8, 2, 3, 0],
          [1, 5, 3, 2]]
total=0
for rows in range(len(matrix)):
    for cols in matrix[rows]:
        total += cols
print(total)

```

Output:
42

15. Calculate the Average

```
values = [61, 20, 45, 63, 96, 71, 6, 8, 72, 22, 97, 7, 46, 11, 15, 74, 81, 69, 70, 26]
avg = sum(values)/len(values)
print('avg = ', avg)
```

Output:
avg = 48.0

16. Calculate the String Lengths

```
words = ['tissue', 'psychology', 'blind', 'assessment', 'dynamic', 'hero',
         'circulation', 'merchant', 'publication', 'interference', 'show',
         'joy', 'sour', 'aloof', 'grass', 'distortion', 'exclude', 'pressure',
         'bullet', 'calf']
word_len = []
for i in words:
    word_len.append(len(i))
print(word_len)
```

Output:
[6, 10, 5, 10, 7, 4, 11, 8, 11, 12, 4, 3, 4, 5, 5, 10, 7, 8, 6, 4]

17. Printing String Characters

```
sentence = 'I\'m practicing printing string characters!'
for i in sentence:
    print(i)
```

Output:

I	p
'	r
m	i
p	n
r	t
a	í
c	n
t	g
i	g
c	g
i	t
n	r
g	i

	c
	h
	a
	r
	a
	c
	t
	e
	r
	s
	!

18. Read the File Into a List of Lists

```
from csv import reader
opened_file = open('dq_unisex_names.csv', encoding="utf8")
read_file = reader(opened_file)
data = list(read_file)
headers = data[0]
rows = data[1:]
print('header: \n', headers)
print('rows: \n', rows[:5])
```

Output:

```
header:
['name', 'estimated_number']
rows:
[['Casey', '176544.328149'], ['Riley', '154860.66517300002'], ['Jessie', '136381.830656'], ['Jackie', '132928.78874000002'], ['Avery', '121797.41951600001']]
```

19. Adding Data to Rows

```
from csv import reader
opened_file = open('dq_unisex_names.csv', encoding="utf8")
read_file = reader(opened_file)
data = list(read_file)
headers = data[0]
rows = data[1:]

for i in range(len(rows)):
    rows[i].append(len(rows[i][0]))
print(rows[:5])
```

Output:

```
[['Casey', '176544.328149', 5], ['Riley', '154860.66517300002', 5], ['Jessie', '136381.830656', 6], ['Jackie', '132928.78874000002', 6], ['Avery', '121797.41951600001', 5]]
```

20. Convert Numerical Values

```
from csv import reader
opened_file = open('dq_unisex_names.csv',encoding="utf8")
read_file = reader(opened_file)
data = list(read_file)
headers = data[0]
rows = data[1:]

for i in range(len(rows)):
    rows[i][1] = float(rows[i][1])
print(rows[:5])
print('\nThe frist data row: ', type(rows[0][1]))
```

Output:

```
[['Casey', 176544.328149], ['Riley', 154860.66517300002], ['Jessie', 136381.830656], ['Jackie', 132928.78874000002], ['Avery', 121797.41951600001]]
```

```
The frist data row: <class 'float'>
```

CONDITIONAL STATEMENTS PRACTICE PROBLEMS

1. Understanding Control Flow 1	2
2. Understanding Control Flow 2	2
3. Understanding Control Flow 3	3
4. Understanding Control Flow 4	4
5. Understanding Control Flow 5	4
6. Understanding Control Flow 6	5
7. Understanding Control Flow 7	5
8. Understanding Control Flow 8	6
9. Counting Large Values.....	6
10. Counting Even and Odd Values.....	6
11. Reversed Lists.....	7
12. Common Characters	8
13. Maximum of a List	8
14. Minimum of a List	9
15. Finding the Pair.....	9
16. The Most Frequent Value	10
17. Counting Most Frequent Names	10
18. Longest Names	11
19. Most Common Name.....	11
20. Filter the List.....	12

1. Understanding Control Flow 1

```
answer1 = 'Nothing is printed.'
answer2 = 'Only A is printed.'
answer3 = 'Only B is printed.'
answer4 = 'Both A and B are printed.'
print(answer3)

x = 10
y = '10'
if x == y:
    print('A')
else:
    print('B')
```

Output:
Only B is printed.
B

2. Understanding Control Flow 2

```
answer1 = 'Nothing is printed.'
answer2 = 'Only A is printed.'
answer3 = 'Only B is printed.'
answer4 = 'Both A and B are printed.'
print(answer2)

x = 10
y = '10'
if str(x) == str(y):
    print('A')
else:
    print('B')
```

Output:
Only A is printed.
A

3. Understanding Control Flow 3

```
answer1 = 'Nothing is printed.'
answer2 = 'Only A is printed.'
answer3 = 'Only B is printed.'
answer4 = 'Only c is printed.'
answer5 = 'Both A and B are printed.'
answer6 = 'Both A and C are printed.'
answer7 = 'Both B and C are printed.'
answer8 = 'ALL A, B and C are printed.'
print(answer5)

x = 3
y = 2
z = 1
if x > y:
    print('A')
if y > z:
    print('B')
if z > x:
    print('C')
```

Output:
Both A and B are printed.
A
B

4. Understanding Control Flow 4

```
answer1 = 'Nothing is printed.'
answer2 = 'Only A is printed.'
answer3 = 'Only B is printed.'
answer4 = 'Only c is printed.'
answer5 = 'Both A and B are printed.'
answer6 = 'Both A and C are printed.'
answer7 = 'Both B and C are printed.'
answer8 = 'ALL A, B and C are printed.'
print(answer2)

x = 3
y = 2
z = 1
if x > y:
    print('A')
elif y > z:
    print('B')
elif z > x:
    print('C')
```

```
Output:
Only A is printed.
A
```

5. Understanding Control Flow 5

```
answer1 = 'Nothing is printed.'
answer2 = 'Only A is printed.'
answer3 = 'Only B is printed.'
answer4 = 'Both A and B are printed.'
print(answer2)

if True:
    print('A')
elif True:
    print('B')
```

```
Output:
Only A is printed.
A
```

6. Understanding Control Flow 6

```
answer1 = 'Only A is printed'
answer2 = 'Only D is not printed'
answer3 = 'Only B and c are printed'
print(answer1)

if True and True:
    print('A')
if True and False:
    print('B')
if False and True:
    print('C')
if False and False:
    print('D')
```

```
Output:
Only A is printed
A
```

7. Understanding Control Flow 7

```
answer1 = 'Only A is printed'
answer2 = 'Only D is not printed'
answer3 = 'Only B and c are printed'
print(answer2)

if True or True:
    print('A')
if True or False:
    print('B')
if False or True:
    print('C')
if False or False:
    print('D')
```

```
Output:
Only D is not printed
A
B
C
```

8. Understanding Control Flow 8

```
answer1 = 'Nothing is printed.'
answer2 = 'Only A is printed.'
answer3 = 'Only B is printed.'
answer4 = 'Both A and B are printed.'
print(answer3)

if False or True and False:
    print('A')
if True or False and False:
    print('B')
```

Output:
Only B is printed.
B

9. Counting Large Values

```
values = [80, 109, 111, 109, 94, 93, 108, 107, 81, 111,
          114, 102, 81, 107, 120, 108, 92, 113, 119, 97]
num_bigger = 0
for i in values:
    if i > 100:
        num_bigger += 1
print(num_bigger)
```

Output:
13

10. Counting Even and Odd Values

```
values = [80, 109, 111, 109, 94, 93, 108, 107, 81, 111,
          114, 102, 81, 107, 120, 108, 92, 113, 119, 97]
num_even = 0
num_odd = 0
for i in values:
    if i % 2 == 1:
        num_odd += 1
    else:
        num_even += 1
print('num_odd ', num_odd)
print('num_even ', num_even)
```

Output:
num_odd 8
num_even 12

11. Reversed Lists

```
values1 = [80, 109, 111, 109, 94,
           93, 108, 107, 81, 111,
           101, 114, 102, 81, 107,
           120, 108, 92, 113, 119, 97]
values2 = [97, 119, 113, 92, 108,
           120, 107, 81, 102, 114,
           110, 111, 81, 107, 108,
           93, 94, 109, 111, 109, 80]
re_val1 = []
for i in values1:
    re_val1 = [i] + re_val1
if re_val1 == values2:
    print(True)
else:
    print(False)
```

Output:

False

12. Common Characters

```
s1 = 'I have been busier these days due to having a lot on my plate.'  
s2 = 'You have been very supportive towards my recent endeavors.'  
s1 = list(s1)  
s2 = list(s2)  
common = set(s1) & set(s2)  
for i in common:  
    print(i)
```

Output:

```
y  
i  
d  
n  
p  
.o  
v  
t  
r  
s  
m  
  
b  
u  
a  
h  
e
```

13. Maximum of a List

```
values = [72, 48, 7, 66, 62, 32, 33, 75, 30, 85,  
         6, 85, 82, 88, 30, 32, 78, 39, 57, 96,  
         45, 57, 61, 10, 62, 48, 32, 96, 75, 15]  
maximum = max(values)  
print(maximum)
```

Output:

```
96
```

14. Minimum of a List

```
values = [72, 48, 7, 66, 62, 32, 33, 75, 30, 85,
          6, 85, 82, 88, 30, 32, 78, 39, 57, 96,
          45, 57, 61, 10, 62, 48, 32, 96, 75, 15]
minium = min(values)
print(minium)
```

Output:

```
6
```

15. Finding the Pair

```
values = [72, 50, 48, 50, 7, 66, 62, 32,
          33, 75, 30, 85, 6, 85, 82, 88,
          30, 32, 78, 39, 57, 96, 45, 57,
          61, 10, 62, 48, 32, 96, 75, 15, 50, 50]
values1 = 0
values2 = 0
for i in range(len(values)):
    for j in range(i, len(values)):
        if values[i] + values[j] == 100:
            values1 = values[i]
            values2 = values[j]
            print(values1, values2)
```

Output:

```
50 50
50 50
50 50
50 50
50 50
50 50
50 50
85 15
85 15
39 61
50 50
50 50
50 50
```

16. The Most Frequent Value

```
values = [72, 50, 48, 50, 7, 66, 62,
          32, 33, 75, 30, 85, 6, 85,
          82, 88, 30, 32, 78, 39, 57,
          96, 45, 57, 61, 32, 10, 62,
          48, 32, 96, 75, 15]
dic = {}
list1 = []
most_frequent = 0
for i in values:
    dic[i] = 0
for i in values:
    dic[i] += 1
for i in dic:
    list1.append(dic[i])
max_val = max(list1)
for i in dic:
    if dic[i] == max_val:
        most_frequent = i
        break
print(most_frequent)
```

Output:
32

17. Counting Most Frequent Names

```
from csv import reader
opened_file = open('dq_unisex_names.csv', encoding="utf8")
read_file = reader(opened_file)
data = list(read_file)
rows = data[1:]
count = 0
for i in range(len(rows)):
    rows[i][1] = float(rows[i][1])
for i in range(len(rows)):
    if rows[i][1] >= 100000:
        count += 1
print(count)
```

Output:
6

18. Longest Names

```
from csv import reader
opened_file = open('dq_unisex_names.csv',encoding="utf8")
read_file = reader(opened_file)
data = list(read_file)
rows = data[1:]
longest_names = []
for i in range(len(rows)):
    longest_names.append(rows[i][0])
    longest_names.append(len(rows[i][0]))
for i in range(1,len(longest_names),2):
    if longest_names[i] > 10:
        print(longest_names[i-1])
```

Output:
Oluwadamilola
Oluwanifemi
Oluwademilade
Oluwatomisin
Oluwadarasimi

19. Most Common Name

```
from csv import reader
opened_file = open('dq_unisex_names.csv',encoding="utf8")
read_file = reader(opened_file)
data = list(read_file)
largest = 0
save_i = 0
rows = data[1:]
for i in range(len(rows)):
    rows[i][1] = float(rows[i][1])
for i in range(len(rows)):
    if largest < rows[i][1]:
        largest = rows[i][1]
        save_i = i
print(rows[save_i][0])
```

Output:
Casey

20. Filter the List

```
from csv import reader
opened_file = open('dq_unisex_names.csv', encoding="utf8")
read_file = reader(opened_file)
data = list(read_file)
rows = data[1:]
rare_names = []
for i in range(len(rows)):
    rows[i][1] = float(rows[i][1])
for i in range(len(rows)):
    if rows[i][1] <= 1000:
        rare_names.append(rows[i][0])
print(rare_names[:5])
```

Output:
['Eliyah', 'Tonnie', 'Teagen', 'Rudi', 'Kamani']

FUNCTIONS: FUNDAMENTALS PRACTICE PROBLEMS

1. Understanding Function Scope 1.....	2
2. Understanding Function Scope 2.....	2
3. Understanding Function Scope 3.....	2
4. Understanding Function Scope 4.....	3
5. Understanding Function Scope 5.....	3
6. Formatting the Time	4
7. Palindrome.....	5
8. The Closest Restaurant	5
9. Finding Column Indexes	5
10. Checking Unique Column Values	7
11. Selecting Rows on a CSV	7
12. Anagrams.....	9

1. Understanding Function Scope 1

```
answer1=None
answer2=[]
answer3=[1,2,3]
x = []
def f():
    x = []
    x.append(1)
    x.append(2)
    x.append(3)
    return x
f()
print(x)
print(answer2)
```

Output:
[]
[]

2. Understanding Function Scope 2

```
answer1=None
answer2=[]
answer3=[1,2,3]
x = []
def f():
    x = []
    x.append(1)
    x.append(2)
    x.append(3)
f()
print(x)
print(answer2)
```

Output:
[]
[]

3. Understanding Function Scope 3

```
answer1=None
answer2=[]
answer3=[1,2,3]
x = []
def f():
    x = []
    x.append(1)
    x.append(2)
    x.append(3)
x = f()
print(x)
print(answer1)
```

Output:

```
None
None
```

4. Understanding Function Scope 4

```
answer1=None
answer2=[]
answer3=[1,2,3]
x = []
def f():
    x.append(1)
    x.append(2)
    x.append(3)
f()
print(x)
print(answer3)
```

Output:

```
[1, 2, 3]
[1, 2, 3]
```

5. Understanding Function Scope 5

```

answer1=None
answer2=[]
answer3=[1,2,3]
x = []
def f():
    x = []
    x.append(1)
    x.append(2)
    x.append(3)
    return x
x = f()
print(x)
print(answer3)

```

Output:

```

[1, 2, 3]
[1, 2, 3]

```

6. Formatting the Time

```

def format_time(hour, minute):
    hh = str(hour)
    mm = str(minute)
    if len(hh) < 2:
        hh = '0' + hh
    if len(mm) < 2:
        mm = '0' + mm
    time = hh + ':' + mm
    return time

hour_1 = 10
minute_1 = 30
print(format_time(hour_1,minute_1))      # answer to this input: 10:30
hour_2 = 5
minute_2 = 7
print(format_time(hour_2,minute_2))      # answer to this input: 05:07
hour_3 = 16
minute_3 = 9
print(format_time(hour_3,minute_3))      # answer to this input: 16:09

```

Output:

```

10:30
05:07
16:09

```

7. Palindrome

```
def is_palindrome(x):
    dna = str(x)
    check = 0
    for i in range(len(dna)):
        if dna[i] != dna[-i-1]:
            check = 1
            break
        elif dna[i] == dna[-i-1]:
            check = 0
    if check == 0:
        print(True)
    else:
        print(False)

is_palindrome("ATTA")
is_palindrome("AGATA")
is_palindrome("G")
```

Output:

```
True
False
True
```

8. The Closest Restaurant

```
import math
opened_file = open('restaurants.csv',encoding="utf8")
from csv import reader
read_file = reader(opened_file)
data = list(read_file)
rows = data[1:]

def closest_restaurant(x,y):
    x_1 = x
    y_1 = y
    m = 10**10
    for i in range(len(rows)):
        name = rows[i][0]
        x_2 = int(rows[i][3])
        y_2 = int(rows[i][4])
        check = math.sqrt((x_1 - x_2)**2 + (y_1 - y_2)**2)
        if m > check:
            m = check
            name_m = name
    print(name_m)

closest_restaurant(100,100)
```

Output:

```
LEADBETTERS II
```

9. Finding Column Indexes

```
def find_col_index(csv_filename, col_name):
    opened_file = open(csv_filename, encoding="utf8")
    from csv import reader
    read_file = reader(opened_file)
    data = list(read_file)
    header = data[0]
    check = True
    for i in range(len(header)):
        if col_name != header[i]:
            check = False
        elif col_name == header[i]:
            print(i)
            break
    if check :
        print(-1)

csv_filename = "users.csv"
col_name1 = "name"
find_col_index(csv_filename, col_name1)      # answer to this input: 2
col_name2 = "id"
find_col_index(csv_filename, col_name2)      # answer to this input: 0
col_name3 = "age"
find_col_index(csv_filename, col_name3)      # answer to this input: -1
```

Output:

```
2
0
-1
```

10. Checking Unique Column Values

```
def check_unique_values(csv_filename, col_name):
    opened_file = open(csv_filename, encoding="utf8")
    from csv import reader
    read_file = reader(opened_file)
    data = list(read_file)
    header = data[0]
    rows = data[1:]
    col = 0
    for i in range(len(header)):           #    tìm cột
        if col_name != header[i]:
            False
        elif col_name == header[i]:
            col = i
            break

    check = False                         #    check values trùng lặp
    for i in range(len(rows)):
        for j in range(i+1, len(rows)):
            if rows[i][col] == rows[j][col]:
                check = False
                break
            elif rows[i][col] != rows[j][col]:
                check = True
    if check == False :
        break
    if check == True:
        print(True)
    else:
        print(False)
csv_filename = "user_accounts.csv"
col_name1 = "name"
check_unique_values(csv_filename, col_name1)
col_name2 = "id"
check_unique_values(csv_filename, col_name2)
col_name3 = "email"
check_unique_values(csv_filename, col_name3)
```

Output:
False
True
False

11. Selecting Rows on a CSV

```

def select(csv_filename, col_name, values):
    opened_file = open(csv_filename, encoding="utf8")
    from csv import reader
    read_file = reader(opened_file)
    data = list(read_file)
    header = data[0]
    rows = data[1:]
    col = 0
    for i in range(len(header)):           #    find_col_index
        if col_name != header[i]:
            False
        elif col_name == header[i]:
            col = i
            break

    check = 0
    for i in range(len(rows)):             #    Tìm và in ra values cần tìm
        if values == rows[i][col]:
            print(rows[i])
            check += 1
    if check == 0:
        print(None)

csv_filename = "user_accounts.csv"
col_name = "name"
values = "Brian Williams"
select(csv_filename, col_name, values)

```

Output:

```

['10', 'brian.williams@hotmail.com', 'Brian Williams', '855 Burch Rue Suite 876 Carlosmouth NE 93014']
['1188', 'brian.williams@henson.com', 'Brian Williams', '48570 Davis Pike Apt. 080 Stevenland LA 25460']
['6529', 'brian.williams@johnson-white.com', 'Brian Williams', 'PSC 9141 Box 1910 APO AP 15743']

```

12. Anagrams

```
def are_anagrams(str1,str2):
    dic1 = {}
    dic2 = {}
    for i in str1:      # Gán các index cho dic1
        dic1[i] = 0
    for i in str1:      # Đếm các từ trong str1 gán cho values cho từng index
        dic1[i] += 1
    for i in str2:      # Gán các index cho dic2
        dic2[i] = 0
    for i in str2:      # Đếm các từ trong str2 gán cho values cho từng index
        dic2[i] += 1
    if dic1 == dic2:    # So sánh số lượng của từng ký tự trong dic1 và dic2
        print(True)
    else:
        print(False)

test1_string1 = 'gainly'
test1_string2 = 'laying'
are_anagrams(test1_string1,test1_string2)
test2_string1 = 'banana'
test2_string2 = 'bacana'
are_anagrams(test2_string1,test2_string2)
```

Output:

```
True
False
```

OBJECT-ORIENTED PYTHON

1. Introduction	2
2. Classes and Objects	2
3. Defining a Class	2
4. Instantiating a Class.....	3
5. Creating Methods	3
6. Understanding "Self"	3
7. Creating a Method that Accepts an Argument	4
8. Attributes and the Init Method	4
9. Creating an Append Method	5
10. Creating and Updating an Attribute.....	5

1. Introduction

```
l = [1, 2, 3]
s = "string"
d = {"a": 1, "b": 2}
print(type(l))
print(type(s))
print(type(d))
```

```
Output:
<class 'list'>
<class 'str'>
<class 'dict'>
```

2. Classes and Objects

Khi chúng ta sử dụng hàm type () trên màn hình trước, nó trả về các giá trị có nhãn "class":

```
Output:
<class 'list'>
<class 'str'>
<class 'dict'>
```

Điều này cho thấy cách chúng ta có thể sử dụng "type" và "class" thay thế cho nhau. Và chúng ta đã học class một thời gian.

3. Defining a Class

Chúng ta định nghĩa một class tương tự như cách chúng ta định nghĩa một hàm:

```
def my_function():
    # the details of the
    # function go here
```

```
class MyClass:
    # the details of the
    # class go here
```

Lưu ý rằng định nghĩa class không có dấu ngoặc đơn (). Đây là tùy chọn đối với class và quy ước là không sử dụng chúng. Tương tự, chúng ta thuật lè phần thân của lớp giống như phần thân của một hàm.

4. Instantiating a Class

```
class MyClass:  
    pass  
my_instance = MyClass()  
print(type(my_instance))
```

Output:
<class '__main__.MyClass'>

5. Creating Methods

```
class MyClass:  
    def first_method() :  
        return "This is my first method"  
my_instance = MyClass()
```

6. Understanding "Self"

```
class MyClass:  
    def first_method() :  
        return "This is my first method"  
  
my_instance = MyClass()  
result = my_instance.first_method()
```

Output:
Traceback (most recent call last):

File "C:\Users\nguye\.spyder-py3\untitled4.py", line 8, in <module>
 result = my_instance.first_method()

TypeError: first_method() takes 0 positional arguments but 1 was given

Để tránh lỗi trên, ta thêm parameter self vào method

```
class MyClass:  
    def first_method(self) :  
        return "This is my first method"  
  
my_instance = MyClass()  
result = my_instance.first_method()
```

Name	Type	Size	Value
result	str	23	This is my first method

7. Creating a Method that Accepts an Argument

```
class MyClass:  
    def first_method(self):  
        return "This is my first method"  
    def return_list(self, input_list):  
        return input_list  
my_instance = MyClass()  
result = my_instance.return_list([1, 2, 3])
```

Name	Type	Size	Value
result	list	3	[1, 2, 3]

8. Attributes and the Init Method

```
class ExampleClass:  
    def __init__(self, string):  
        self.my_attribute = string  
mc = ExampleClass("Hola!")  
print(mc.my_attribute)
```

Output:
Hola!

```
class myList:  
    def __init__(self, initial_data):  
        self.data = initial_data  
my_list = myList([1, 2, 3, 4, 5])  
print (my_list.data)
```

Output:
[1, 2, 3, 4, 5]

9. Creating an Append Method

```
class MyList:  
    def __init__(self, initial_data):  
        self.data = initial_data  
    def append(self, new_item):  
        self.data = self.data + [new_item]  
my_list = MyList([1, 2, 3, 4, 5])  
print(my_list.data)  
my_list.append(6)  
print(my_list.data)
```

Output:

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

10. Creating and Updating an Attribute

```
class MyList:  
    def __init__(self, initial_data):  
        self.data = initial_data  
        self.length = 0  
        for item in self.data:  
            self.length += 1  
    def append(self, new_item):  
        self.data = self.data + [new_item]  
  
my_list = MyList([1, 2, 3])  
print(my_list.length)  
my_list.append(4)  
print(my_list.length)
```

Output:

```
3  
3
```

OBJECT-ORIENTED PYTHON PRACTICE

PROBLEMS

1. Person Class.....	2
2. 2D Points.....	2
3. Time	3
4. Frequency Table.....	3
5. Supermarket Queue.....	4
6. Using a Class for CSV Rows	5

1. Person Class

```
class Person():
    def __init__(self, first_name, last_name):
        self.first_name = first_name[0].upper() + first_name[1:].lower()
        self.last_name = last_name[0].upper() + last_name[1:].lower()
    def __str__():
        return '{} {}'.format(self.first_name, self.last_name)
person = Person("Emilia", "Gomez")
print(person)

Output:
Emilia Gomez
```

2. 2D Points

```
import math
class Point2D():
    def __init__(self,x,y):
        self.x = x
        self.y = y
    def calculate_dis(self, other):
        dx = self.x - other.x
        dy = self.y - other.y
        return math.sqrt(dx * dx + dy * dy)
point1 = Point2D(3,4)
point2 = Point2D(9,5)
dis = point1.calculate_dis(point2)
print(dis)

Output:
6.082762530298219
```

3. Time

```
class Time():
    def __init__(self, hour, minute, second):
        self.hour = hour
        self.minute = minute
        self.second = second
    def total_second(self):
        return 3600*self.hour + 60*self.minute + self.second
    def __str__(self):
        s = ""
        if self.hour < 10:
            s += '0'
        s += str(self.hour)
        s += ':'
        if self.minute < 10:
            s += '0'
        s += str(self.minute)
        s += ':'
        if self.second < 10:
            s += '0'
        s += str(self.second)
        return s
mytime = Time(9,5,7)
mysecond = mytime.total_second()
print(mysecond)
print(mytime)
```

Output:

```
32707
09:05:07
```

4. Frequency Table

```
class FreqTable():
    def __init__(self):
        self.count = {}
    def add(self, ele):
        if not ele in self.count:
            self.count[ele] = 0
        self.count[ele] += 1
        return self.count[ele]
    def getcount(self,ele):
        if ele not in self.count:
            return 0
        return self.count[ele]
freqtable = FreqTable()
for i in range(3):
    freqtable.add(0)
print(freqtable.getcount(0))
print(freqtable.getcount(1))
```

Output:

```
3
0
```

5. Supermarket Queue

```
class SupermarketQueue():
    def __init__(self):
        self.elements = []
        self.front_index = 0
        self.num_elements = 0
    def add_to_back(self, element):
        self.elements.append(element)
        self.num_elements += 1
    def remove_from_front(self):
        ret = self.elements[self.front_index]
        self.front_index += 1
        self.num_elements -= 1
        return ret
    def __str__(self):
        tmp = self.elements[self.front_index: ].copy()
        tmp.reverse()
        return str(tmp)
    def __len__(self):
        return self.num_elements

queue = SupermarketQueue()
queue.add_to_back('Alice')
queue.add_to_back('Bob')
print(len(queue))
print(queue)
print(queue.remove_from_front())
```

Output:

```
2
['Bob', 'Alice']
Alice
```

6. Using a Class for CSV Rows

```
class User():                      # User class
    def __init__(self, row):
        self.id = row[0]
        self.email = row[1]
        self.name = row[2]
        self.address = row[3]
users = []                         # Read the CSV and create User instances
import csv
with open('user_accounts.csv') as f:
    reader = csv.reader(f)
    rows = list(reader)
    for row in rows[1:]:
        user = User(row)
        users.append(user)
last = users[-1]                   # Solution testing
print(last.id)
print(last.email)
print(last.name)
print(last.address)

Output:
9999
edward.jimenez@lee.org
Edward Jimenez
4445 Dylan Brook Apt. 099 South Lisa DE 87826
```

NUMPY

1.	Using NumPy	2
2.	Numpy Arrays	2
2.1.	Creating NumPy Arrays	2
2.2.	Built-in methods.....	2
2.3.	Random	3
2.4.	Array Attributes and Methods.....	4
2.5.	Reshape	4
2.6.	max, min , argmax, argmin	4
2.7.	Shape.....	5
2.8.	dtype.....	6
3.	NumPy Indexing and Selection	6
3.1.	Bracket Indexing and Selection.....	6
3.2.	Broadcasting.....	7
3.3.	Indexing a 2D array (matrices).....	8
3.4.	Fancy Indexing.....	8
3.5.	Selection.....	9
4.	NumPy Operations	10
4.1.	Arithmetic	10
4.2.	Universal Array Functions	11
5.	Numpy Exercises.....	11

1. Using NumPy

```
| import numpy as np
```

2. Numpy Arrays

2.1. Creating NumPy Arrays

```
my_list = [1,2,3]  
my_list
```

```
[1, 2, 3]
```

```
np.array(my_list)  
array([1, 2, 3])
```

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

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

```
np.array(my_matrix)  
array([[1, 2, 3],  
       [4, 5, 6],  
       [7, 8, 9]])
```

2.2. Built-in methods

arange

```
np.arange(0,10)  
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
np.arange(0,11,2)  
array([ 0,  2,  4,  6,  8, 10])
```

zeros and ones

```
np.zeros(3)  
array([ 0.,  0.,  0.])  
  
np.zeros((5,5))  
array([[ 0.,  0.,  0.,  0.,  0.],  
       [ 0.,  0.,  0.,  0.,  0.],  
       [ 0.,  0.,  0.,  0.,  0.],  
       [ 0.,  0.,  0.,  0.,  0.],  
       [ 0.,  0.,  0.,  0.,  0.]])
```

```
np.ones(3)
```

```
array([ 1.,  1.,  1.])
```

```
np.ones((3,3))
```

```
array([[ 1.,  1.,  1.],
       [ 1.,  1.,  1.],
       [ 1.,  1.,  1.]])
```

linspace

```
np.linspace(0,10,3)
```

```
array([ 0.,  5., 10.])
```

```
np.linspace(0,10,50)
```

```
array([ 0.          ,  0.20408163,  0.40816327,  0.6122449 ,
       0.81632653,  1.02040816,  1.2244898 ,  1.42857143,
       1.63265306,  1.83673469,  2.04081633,  2.24489796,
       2.44897959,  2.65306122,  2.85714286,  3.06122449,
       3.26530612,  3.46938776,  3.67346939,  3.87755102,
       4.08163265,  4.28571429,  4.48979592,  4.69387755,
       4.89795918,  5.10204082,  5.30612245,  5.51020408,
       5.71428571,  5.91836735,  6.12244898,  6.32653061,
       6.53061224,  6.73469388,  6.93877551,  7.14285714,
       7.34693878,  7.55102041,  7.75510204,  7.95918367,
       8.16326531,  8.36734694,  8.57142857,  8.7755102 ,
       8.97959184,  9.18367347,  9.3877551 ,  9.59183673,
       9.79591837,  10.        ])
```

eye

```
np.eye(4)
```

```
array([[ 1.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.],
       [ 0.,  0.,  1.,  0.],
       [ 0.,  0.,  0.,  1.]])
```

2.3. Random

rand

```
np.random.rand(2)
```

```
array([ 0.11570539,  0.35279769])
```

```
np.random.rand(5,5)
```

```
array([[ 0.66660768,  0.87589888,  0.12421056,  0.65074126,  0.60260888],
       [ 0.70027668,  0.85572434,  0.8464595 ,  0.2735416 ,  0.10955384],
       [ 0.0670566 ,  0.83267738,  0.9082729 ,  0.58249129,  0.12305748],
       [ 0.27948423,  0.66422017,  0.95639833,  0.34238788,  0.9578872 ],
       [ 0.72155386,  0.3035422 ,  0.85249683,  0.30414307,  0.79718816]])
```

randn

```
np.random.randn(2)
```

```
array([-0.27954018,  0.90078368])
```

```
np.random.randn(5,5)
```

```
array([[ 0.70154515,  0.22441999,  1.33563186,  0.82872577, -0.28247509],
       [ 0.64489788,  0.61815094, -0.81693168, -0.30102424, -0.29030574],
       [ 0.8695976 ,  0.413755 ,  2.20047208,  0.17955692, -0.82159344],
       [ 0.59264235,  1.29869894, -1.18870241,  0.11590888, -0.09181687],
       [-0.96924265, -1.62888685, -2.05787102, -0.29705576,  0.68915542]])
```

randint

```
np.random.randint(1,100)
```

```
44
```

```
np.random.randint(1,100,10)
```

```
array([13, 64, 27, 63, 46, 68, 92, 10, 58, 24])
```

2.4. Array Attributes and Methods

```
arr = np.arange(25)
```

```
ranarr = np.random.randint(0,50,10)
```

```
arr
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24])
```

```
ranarr
```

```
array([40, 11, 38, 32, 48,  5, 32,  9, 14, 31])
```

2.5. Reshape

```
arr.reshape(5,5)
```

```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

2.6. max, min , argmax, argmin

```
ranarr
```

```
array([40, 11, 38, 32, 48, 5, 32, 9, 14, 31])
```

```
ranarr.max()
```

```
48
```

```
ranarr.argmax()
```

```
4
```

```
ranarr.min()
```

```
5
```

```
ranarr.argmin()
```

```
5
```

2.7. Shape

```
# Vector
```

```
arr.shape
```

```
array([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24])
```

```
# Notice the two sets of brackets
```

```
arr.reshape(1,25)
```

```
array([[ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,
       16, 17, 18, 19, 20, 21, 22, 23, 24]])
```

```
arr.reshape(1,25).shape
```

```
(1, 25)
```

```
arr.reshape(25,1)
```

```
array([[ 0],  
       [ 1],  
       [ 2],  
       [ 3],  
       [ 4],  
       [ 5],  
       [ 6],  
       [ 7],  
       [ 8],  
       [ 9],  
       [10],  
       [11],  
       [12],  
       [13],  
       [14],  
       [15],  
       [16],  
       [17],  
       [18],  
       [19],  
       [20],  
       [21],  
       [22],  
       [23],  
       [24]])
```

```
arr.reshape(25,1).shape
```

```
(25, 1)
```

2.8. dtype

```
arr.dtype
```

```
dtype('int32')
```

3. NumPy Indexing and Selection

```
import numpy as np
```

```
#Creating sample array  
arr = np.arange(0,11)
```

```
#Show  
arr
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

3.1. Bracket Indexing and Selection

```
| #Get a value at an index  
arr[8]
```

```
8
```

```
| #Get values in a range  
arr[1:5]
```

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

```
| #Get values in a range  
arr[0:5]
```

```
array([0, 1, 2, 3, 4])
```

3.2. Broadcasting

```
| #Setting a value with index range (Broadcasting)  
arr[0:5]=100
```

```
#Show  
arr
```

```
array([100, 100, 100, 100, 100, 5, 6, 7, 8, 9, 10])
```

```
| # Reset array, we'll see why I had to reset in a moment  
arr = np.arange(0,11)
```

```
#Show  
arr
```

```
array([ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```

```
| #Important notes on Slices  
slice_of_arr = arr[0:6]
```

```
#Show slice  
slice_of_arr
```

```
array([0, 1, 2, 3, 4, 5])
```

```
| #Change Slice  
slice_of_arr[:]=99
```

```
#Show Slice again  
slice_of_arr
```

```
array([99, 99, 99, 99, 99, 99])
```

```
arr
```

```
array([99, 99, 99, 99, 99, 99, 6, 7, 8, 9, 10])
```

```
| #To get a copy, need to be explicit  
arr_copy = arr.copy()
```

```
arr_copy
```

```
array([99, 99, 99, 99, 99, 99, 7, 8, 9, 10])
```

3.3. Indexing a 2D array (matrices)

```
arr_2d = np.array(([5,10,15],[20,25,30],[35,40,45]))  
  
#Show  
arr_2d  
  
array([[ 5, 10, 15],  
       [20, 25, 30],  
       [35, 40, 45]])
```

```
#Indexing row  
arr_2d[1]
```

```
array([20, 25, 30])
```

```
# Format is arr_2d[row][col] or arr_2d[row,col]  
  
# Getting individual element value  
arr_2d[1][0]
```

```
20
```

```
# Getting individual element value  
arr_2d[1,0]
```

```
20
```

```
# 2D array slicing  
  
#Shape (2,2) from top right corner  
arr_2d[:2,1:]
```

```
array([[10, 15],  
       [25, 30]])
```

```
#Shape bottom row  
arr_2d[2]
```

```
array([35, 40, 45])
```

```
#Shape bottom row  
arr_2d[2,:]
```

```
array([35, 40, 45])
```

3.4. Fancy Indexing

```
#Set up matrix
arr2d = np.zeros((10,10))
```

```
#Length of array
arr_length = arr2d.shape[1]
arr_length
```

```
10
```

```
#Set up array
for i in range(arr_length):
    arr2d[i] = i
```

```
arr2d
```

```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
       [2., 2., 2., 2., 2., 2., 2., 2., 2., 2.],
       [3., 3., 3., 3., 3., 3., 3., 3., 3., 3.],
       [4., 4., 4., 4., 4., 4., 4., 4., 4., 4.],
       [5., 5., 5., 5., 5., 5., 5., 5., 5., 5.],
       [6., 6., 6., 6., 6., 6., 6., 6., 6., 6.],
       [7., 7., 7., 7., 7., 7., 7., 7., 7., 7.],
       [8., 8., 8., 8., 8., 8., 8., 8., 8., 8.],
       [9., 9., 9., 9., 9., 9., 9., 9., 9., 9.]])
```

```
arr2d[[2,4,6,8]]
```

```
array([[2., 2., 2., 2., 2., 2., 2., 2., 2., 2.],
       [4., 4., 4., 4., 4., 4., 4., 4., 4., 4.],
       [6., 6., 6., 6., 6., 6., 6., 6., 6., 6.],
       [8., 8., 8., 8., 8., 8., 8., 8., 8., 8.]])
```

```
#Allows in any order
arr2d[[6,4,2,7]]
```

```
array([[6., 6., 6., 6., 6., 6., 6., 6., 6., 6.],
       [4., 4., 4., 4., 4., 4., 4., 4., 4., 4.],
       [2., 2., 2., 2., 2., 2., 2., 2., 2., 2.],
       [7., 7., 7., 7., 7., 7., 7., 7., 7., 7.]])
```

3.5. Selection

```
arr = np.arange(1,11)
arr
```

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

```
arr > 4
```

```
array([False, False, False, False,  True,  True,  True,  True,  True,
       True])
```

```
bool_arr = arr>4
```

```
bool_arr
```

```
array([False, False, False, False, True, True, True, True, True,
```

```
True])
```

```
arr[bool_arr]
```

```
array([ 5,  6,  7,  8,  9, 10])
```

```
arr[bool_arr]
```

```
array([ 5,  6,  7,  8,  9, 10])
```

```
arr[arr>2]
```

```
array([ 3,  4,  5,  6,  7,  8,  9, 10])
```

```
x = 2
```

```
arr[arr>x]
```

```
array([ 3,  4,  5,  6,  7,  8,  9, 10])
```

4. NumPy Operations

4.1. Arithmetic

```
import numpy as np
arr = np.arange(0,10)
```

```
arr + arr
```

```
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
```

```
arr * arr
```

```
array([ 0,  1,  4,  9, 16, 25, 36, 49, 64, 81])
```

```
arr - arr
```

```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
# Warning on division by zero, but not an error!
```

```
# Just replaced with nan
```

```
arr/arr
```

```
array([nan,  1.,  1.,  1.,  1.,  1.,  1.,  1.,  1.])
```

```
# Also warning, but not an error instead infinity
```

```
1/arr
```

```
array([
        inf, 1. , 0.5 , 0.33333333, 0.25 ,
        0.2 , 0.16666667, 0.14285714, 0.125 , 0.11111111])
```

```
arr**3
```

```
array([ 0, 1, 8, 27, 64, 125, 216, 343, 512, 729], dtype=int32)
```

4.2. Universal Array Functions

```
#Taking Square Roots  
np.sqrt(arr)
```

```
array([0. , 1. , 1.41421356, 1.73205081, 2.  
      2.23606798, 2.44948974, 2.64575131, 2.82842712, 3. ])
```

```
#Calcuating exponential (e^)  
np.exp(arr)
```

```
array([1.00000000e+00, 2.71828183e+00, 7.38905610e+00, 2.00855369e+01,  
      5.45981500e+01, 1.48413159e+02, 4.03428793e+02, 1.09663316e+03,  
      2.98095799e+03, 8.10308393e+03])
```

```
np.max(arr) #same as arr.max()
```

```
9
```

```
np.sin(arr)
```

```
array([ 0. , 0.84147098, 0.90929743, 0.14112001, -0.7568025 ,  
      -0.95892427, -0.2794155 , 0.6569866 , 0.98935825, 0.41211849])
```

```
np.log(arr)
```

```
array([-inf, 0. , 0.69314718, 1.09861229, 1.38629436,  
      1.60943791, 1.79175947, 1.94591015, 2.07944154, 2.19722458])
```

5. Numpy Exercises

```
import numpy as np
```

```
np.zeros(10)
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

```
np.ones(10)
```

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

```
np.ones(10) * 5
```

```
array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

```
np.arange(10,51)
```

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,  
      27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,  
      44, 45, 46, 47, 48, 49, 50])
```

```
np.arange(10,51,2)
```

```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
       44, 46, 48, 50])
```

```
np.arange(9).reshape(3,3)
```

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

```
np.eye(3)
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

```
np.random.rand(1)
```

```
array([0.61535346])
```

```
np.random.randn(25)
```

```
array([-2.36812666, -0.40230955, -0.86552875,  0.65452654, -0.71301621,
       0.4766957 ,  0.80902637, -0.08030906, -0.36346472, -0.72519298,
      -1.56093692, -0.80171591,  0.65808634, -1.13814778, -0.48713089,
      -0.48514401, -1.25926276, -0.70480916,  0.31073141,  0.2583337 ])
```

```
np.arange(1,101).reshape(10,10) / 100
```

```
array([[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ],
       [0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 ],
       [0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 ],
       [0.31, 0.32, 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 ],
       [0.41, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 ],
       [0.51, 0.52, 0.53, 0.54, 0.55, 0.56, 0.57, 0.58, 0.59, 0.6 ],
       [0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67, 0.68, 0.69, 0.7 ],
       [0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77, 0.78, 0.79, 0.8 ],
       [0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89, 0.9 ],
       [0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1. ]])
```

```
np.linspace(0,1,20)
```

```
array([0.          , 0.05263158, 0.10526316, 0.15789474, 0.21052632,
       0.26315789, 0.31578947, 0.36842105, 0.42105263, 0.47368421,
       0.52631579, 0.57894737, 0.63157895, 0.68421053, 0.73684211,
       0.78947368, 0.84210526, 0.89473684, 0.94736842, 1.        ])
```

```
mat = np.arange(1,26).reshape(5,5)
mat
```

```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

```
mat[2:,1:]
```

```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

```
mat[3,4]
```

```
20
```

```
mat[:3,1:2]
```

```
array([[ 2],
       [ 7],
       [12]])
```

```
mat[4,:]
```

```
array([21, 22, 23, 24, 25])
```

```
mat[3:5,:]
```

```
array([[16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

```
mat.sum()
```

```
325
```

```
mat.std()
```

```
7.211102550927978
```

```
mat.sum(axis=0)
```

```
array([55, 60, 65, 70, 75])
```

PANDAS PART 1

1. Series	2
1.1. Creating a Series	2
1.2. Data in a Series	3
1.3. Using an Index	3
2. DataFrames.....	4
2.1. Selection and Indexing.....	4
2.2. More index details.....	8
2.3. Multi-Index and Index Hierarchy.....	10
3. Missing Data.....	11
4. Groupby.....	12
5. Merging, Joining, and Concatenating.....	15
5.1. Concatenaion.....	16
5.2. Merging.....	17
5.3. Joining.....	19
6. Operations	19
6.1. Info on Unique Values	20
6.2. Selecting Data	20
6.3. Apply Functions.....	20
7. Data input and data output.....	23
7.1. CSV.....	23
7.2. Excel	24
7.3. HTML	24
7.4. SQL	24

1. Series

```
import numpy as np  
import pandas as pd
```

1.1.Creating a Series

```
labels = ['a','b','c']  
my_list = [10,20,30]  
arr = np.array([10,20,30])  
d = {'a':10, 'b':20, 'c':30}
```

Using lists

```
pd.Series(data=my_list)
```

```
0    10  
1    20  
2    30  
dtype: int64
```

```
pd.Series(data=my_list,index=labels)
```

```
a    10  
b    20  
c    30  
dtype: int64
```

```
pd.Series(my_list,labels)
```

```
a    10  
b    20  
c    30  
dtype: int64
```

Using Numpy Arrays

```
pd.Series(arr)
```

```
0    10  
1    20  
2    30  
dtype: int32
```

```
pd.Series(arr,labels)
```

```
a    10  
b    20  
c    30  
dtype: int32
```

Dictionary

```
pd.Series(d)
```

```
a    10  
b    20  
c    30  
dtype: int64
```

1.2.Data in a Series

```
pd.Series(data=labels)
```

```
0    a  
1    b  
2    c  
dtype: object
```

```
# Even functions (although unlikely that you will use this)  
pd.Series([sum,print,len])
```

```
0      <built-in function sum>  
1      <built-in function print>  
2      <built-in function len>  
dtype: object
```

1.3.Using an Index

```
ser1 = pd.Series([1,2,3,4],index = ['USA', 'Germany', 'USSR', 'Japan'])
```

```
ser1
```

```
USA      1  
Germany  2  
USSR     3  
Japan    4  
dtype: int64
```

```
ser2 = pd.Series([1,2,5,4],index = ['USA', 'Germany','Italy', 'Japan'])
```

```
ser2
```

```
USA      1  
Germany  2  
Italy    5  
Japan    4  
dtype: int64
```

```
ser1['USA']
```

```
1
```

```
ser1 + ser2
```

```
Germany    4.0
Italy      NaN
Japan     8.0
USA      2.0
USSR      NaN
dtype: float64
```

2. DataFrames

```
from numpy.random import randn
np.random.seed(101)
```

```
df = pd.DataFrame(randn(5,4),index='A B C D E'.split(),columns='W X Y Z'.split())
```

```
df
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

2.1. Selection and Indexing

```
df['W']
```

```
A    2.706850
B    0.651118
C   -2.018168
D    0.188695
E    0.190794
Name: W, dtype: float64
```

```
# Pass a list of column names
df[['W','Z']]
```

	W	Z
A	2.706850	0.503826
B	0.651118	0.605965
C	-2.018168	-0.589001
D	0.188695	0.955057
E	0.190794	0.683509

```
# SQL Syntax (NOT RECOMMENDED!)
df.W
```

```
A    2.706850
B    0.651118
C   -2.018168
D    0.188695
E    0.190794
Name: W, dtype: float64
```

```
type(df['W'])
```

```
pandas.core.series.Series
```

Creating a new column

```
df['new'] = df['W'] + df['Y']
```

```
df
```

	W	X	Y	Z	new
A	2.706850	0.628133	0.907969	0.503826	3.614819
B	0.651118	-0.319318	-0.848077	0.605965	-0.196959
C	-2.018168	0.740122	0.528813	-0.589001	-1.489355
D	0.188695	-0.758872	-0.933237	0.955057	-0.744542
E	0.190794	1.978757	2.605967	0.683509	2.796762

Removing Columns

```
df.drop('new', axis=1)
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

```
# Not inplace unless specified!  
df
```

	W	X	Y	Z	new
A	2.706850	0.628133	0.907969	0.503826	3.614819
B	0.651118	-0.319318	-0.848077	0.605965	-0.196959
C	-2.018168	0.740122	0.528813	-0.589001	-1.489355
D	0.188695	-0.758872	-0.933237	0.955057	-0.744542
E	0.190794	1.978757	2.605967	0.683509	2.796762

```
df.drop('new',axis=1,inplace=True)
```

```
df
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

```
df.drop('E',axis=0)
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057

```
df.loc['A']
```

W 2.706850
X 0.628133
Y 0.907969
Z 0.503826
Name: A, dtype: float64

```
df.iloc[2]
```

W -2.018168
X 0.740122
Y 0.528813
Z -0.589001
Name: C, dtype: float64

```
df.loc['B','Y']
```

-0.84807698340363147

```
df.loc[['A','B'],['W','Y']]
```

	W	Y
A	2.706850	0.907969
B	0.651118	-0.848077

Conditional Selection

```
df
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

```
df>0
```

	W	X	Y	Z
A	True	True	True	True
B	True	False	False	True
C	False	True	True	False
D	True	False	False	True
E	True	True	True	True

```
df[df>0]
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	NaN	NaN	0.605965
C	NaN	0.740122	0.528813	NaN
D	0.188695	NaN	NaN	0.955057
E	0.190794	1.978757	2.605967	0.683509

```
df[df['W']>0]
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

```
df[df['W']>0]['Y']
```

A 0.907969
B -0.848077
D -0.933237
E 2.605967
Name: Y, dtype: float64

```
df[df['W']>0][['Y','X']]
```

	Y	X
A	0.907969	0.628133
B	-0.848077	-0.319318
C	-0.933237	-0.758872
E	2.605967	1.978757

```
df[(df['W']>0) & (df['Y'] > 1)]
```

	W	X	Y	Z
E	0.190794	1.978757	2.605967	0.683509

2.2. More index details

```
df
```

	W	X	Y	Z
A	2.706850	0.628133	0.907969	0.503826
B	0.651118	-0.319318	-0.848077	0.605965
C	-2.018168	0.740122	0.528813	-0.589001
D	0.188695	-0.758872	-0.933237	0.955057
E	0.190794	1.978757	2.605967	0.683509

```
# Reset to default 0,1...n index  
df.reset_index()
```

	index	W	X	Y	Z
0	A	2.706850	0.628133	0.907969	0.503826
1	B	0.651118	-0.319318	-0.848077	0.605965
2	C	-2.018168	0.740122	0.528813	-0.589001
3	D	0.188695	-0.758872	-0.933237	0.955057
4	E	0.190794	1.978757	2.605967	0.683509

```
newind = 'CA NY WY OR CO'.split()
```

```
df['States'] = newind
```

```
df
```

	W	X	Y	Z	States
A	2.706850	0.628133	0.907969	0.503826	CA
B	0.651118	-0.319318	-0.848077	0.605965	NY
C	-2.018168	0.740122	0.528813	-0.589001	WY
D	0.188695	-0.758872	-0.933237	0.955057	OR
E	0.190794	1.978757	2.605967	0.683509	CO

```
df.set_index('States')
```

```
W X Y Z
```

States

CA	2.706850	0.628133	0.907969	0.503826
NY	0.651118	-0.319318	-0.848077	0.605965
WY	-2.018168	0.740122	0.528813	-0.589001
OR	0.188695	-0.758872	-0.933237	0.955057
CO	0.190794	1.978757	2.605967	0.683509

```
df
```

```
W X Y Z States
```

A	2.706850	0.628133	0.907969	0.503826	CA
B	0.651118	-0.319318	-0.848077	0.605965	NY
C	-2.018168	0.740122	0.528813	-0.589001	WY
D	0.188695	-0.758872	-0.933237	0.955057	OR
E	0.190794	1.978757	2.605967	0.683509	CO

```
df.set_index('States', inplace=True)
```

```
df
```

```
W X Y Z
```

States

CA	2.706850	0.628133	0.907969	0.503826
NY	0.651118	-0.319318	-0.848077	0.605965
WY	-2.018168	0.740122	0.528813	-0.589001
OR	0.188695	-0.758872	-0.933237	0.955057
CO	0.190794	1.978757	2.605967	0.683509

2.3. Multi-Index and Index Hierarchy

```
# Index Levels
outside = ['G1', 'G1', 'G1', 'G2', 'G2', 'G2']
inside = [1, 2, 3, 1, 2, 3]
hier_index = list(zip(outside, inside))
hier_index = pd.MultiIndex.from_tuples(hier_index)
```

```
hier_index
```

```
MultiIndex(levels=[[ 'G1', 'G2'], [1, 2, 3]],
           labels=[[0, 0, 0, 1, 1, 1], [0, 1, 2, 0, 1, 2]])
```

```
df = pd.DataFrame(np.random.randn(6,2), index=hier_index, columns=[ 'A', 'B'])
df
```

	A	B
1	0.153661	0.167638
G1 2	-0.765930	0.962299
3	0.902826	-0.537909
1	-1.549671	0.435253
G2 2	1.259904	-0.447898
3	0.266207	0.412580

```
df.loc['G1']
```

	A	B
1	0.153661	0.167638
2	-0.765930	0.962299
3	0.902826	-0.537909

```
df.loc['G1'].loc[1]
```

```
A      0.153661
B      0.167638
Name: 1, dtype: float64
```

```
df.index.names
```

```
FrozenList([None, None])
```

```
df.index.names = ['Group', 'Num']
```

```
df
```

	A	B
Group	Num	
	1	0.153661 0.167638
G1	2	-0.765930 0.962299
	3	0.902826 -0.537909
	1	-1.549671 0.435253
G2	2	1.259904 -0.447898
	3	0.266207 0.412580

```
df.xs('G1')
```

	A	B
Num		

1	0.153661 0.167638
2	-0.765930 0.962299
3	0.902826 -0.537909

```
df.xs(['G1', 1])
```

A 0.153661
B 0.167638
Name: (G1, 1), dtype: float64

```
df.xs(1, level='Num')
```

	A	B
--	---	---

Group		
G1	0.153661 0.167638	
G2	-1.549671 0.435253	

3. Missing Data

```
df = pd.DataFrame({'A':[1,2,np.nan],  
                  'B':[5,np.nan,np.nan],  
                  'C':[1,2,3]})
```

```
df
```

	A	B	C
0	1.0 5.0 1		
1	2.0 NaN 2		
2	NaN NaN 3		

```
df.dropna()
```

	A	B	C
0	1.0	5.0	1

```
| df.dropna(axis=1)
```

	C
0	1
1	2
2	3

```
df.dropna(thresh=2)
```

	A	B	C
0	1.0	5.0	1
1	2.0	NaN	2

```
df.fillna(value='FILL VALUE')
```

	A	B	C
0	1	5	1
1	2	FILL VALUE	2
2	FILL VALUE	FILL VALUE	3

```
df['A'].fillna(value=df['A'].mean())
```

```
0    1.0
1    2.0
2    1.5
Name: A, dtype: float64
```

4. Groupby

```
import pandas as pd
# Create dataframe
data = {'Company': ['GOOG', 'GOOG', 'MSFT', 'MSFT', 'FB', 'FB'],
        'Person': ['Sam', 'Charlie', 'Amy', 'Vanessa', 'Carl', 'Sarah'],
        'Sales': [200, 120, 340, 124, 243, 350]}
```

```
df = pd.DataFrame(data)
```

```
df
```

	Company	Person	Sales
0	GOOG	Sam	200
1	GOOG	Charlie	120
2	MSFT	Amy	340
3	MSFT	Vanessa	124
4	FB	Carl	243
5	FB	Sarah	350

```
df.groupby('Company')
```

```
<pandas.core.groupby.DataFrameGroupBy object at 0x113014128>
```

```
by_comp = df.groupby("Company")
```

```
by_comp.mean()
```

Sales

Company	Sales
FB	296.5
GOOG	160.0
MSFT	232.0

```
df.groupby('Company').mean()
```

Sales

Company	Sales
FB	296.5
GOOG	160.0
MSFT	232.0

```
by_comp.std()
```

Sales

Company

FB	75.660426
GOOG	56.568542
MSFT	152.735065

```
by_comp.min()
```

Person Sales

Company

FB	Carl	243
GOOG	Charlie	120
MSFT	Amy	124

```
by_comp.max()
```

Person Sales

Company

FB	Sarah	350
GOOG	Sam	200
MSFT	Vanessa	340

```
by_comp.count()
```

Person Sales

Company

FB	2	2
GOOG	2	2
MSFT	2	2

```
by_comp.describe()
```

Sales

Company

FB	count	2.000000	count	2.000000	count	2.000000
	mean	296.500000	mean	160.000000	mean	232.000000
	std	75.660426	std	56.568542	std	152.735065
	min	243.000000	min	120.000000	min	124.000000
	25%	269.750000	25%	140.000000	25%	178.000000
	50%	296.500000	50%	160.000000	50%	232.000000
	75%	323.250000	75%	180.000000	75%	286.000000
	max	350.000000	max	200.000000	max	340.000000

```
by_comp.describe().transpose()
```

pany	FB					GOOG					MSFT										
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean	std	min	25%	50%	75%	ma
Sales	2.0	296.5	75.660426	243.0	269.75	296.5	323.25	350.0	2.0	160.0	...	180.0	200.0	2.0	232.0	152.735065	124.0	178.0	232.0	286.0	340.

```
by_comp.describe().transpose()['GOOG']
```

	count	mean	std	min	25%	50%	75%	max
Sales	2.0	160.0	56.568542	120.0	140.0	160.0	180.0	200.0

5. Merging, Joining, and Concatenating

Example Dataframes

```
df1 = pd.DataFrame({'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3'],
                    'C': ['C0', 'C1', 'C2', 'C3'],
                    'D': ['D0', 'D1', 'D2', 'D3']},
                   index=[0, 1, 2, 3])
```

```
df2 = pd.DataFrame({'A': ['A4', 'A5', 'A6', 'A7'],
                    'B': ['B4', 'B5', 'B6', 'B7'],
                    'C': ['C4', 'C5', 'C6', 'C7'],
                    'D': ['D4', 'D5', 'D6', 'D7']},
                   index=[4, 5, 6, 7])
```

```
df3 = pd.DataFrame({'A': ['A8', 'A9', 'A10', 'A11'],
                    'B': ['B8', 'B9', 'B10', 'B11'],
                    'C': ['C8', 'C9', 'C10', 'C11'],
                    'D': ['D8', 'D9', 'D10', 'D11']},
                   index=[8, 9, 10, 11])
```

```
df1
```

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3

```
df2
```

	A	B	C	D
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7

```
df3
```

	A	B	C	D
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

5.1.Concatenation

```
pd.concat([df1,df2,df3])
```

	A	B	C	D
0	A0	B0	C0	D0
1	A1	B1	C1	D1
2	A2	B2	C2	D2
3	A3	B3	C3	D3
4	A4	B4	C4	D4
5	A5	B5	C5	D5
6	A6	B6	C6	D6
7	A7	B7	C7	D7
8	A8	B8	C8	D8
9	A9	B9	C9	D9
10	A10	B10	C10	D10
11	A11	B11	C11	D11

```
pd.concat([df1,df2,df3],axis=1)
```

	A	B	C	D	A	B	C	D	A	B	C	D
0	A0	B0	C0	D0	NaN							
1	A1	B1	C1	D1	NaN							
2	A2	B2	C2	D2	NaN							
3	A3	B3	C3	D3	NaN							
4	NaN	NaN	NaN	NaN	A4	B4	C4	D4	NaN	NaN	NaN	NaN
5	NaN	NaN	NaN	NaN	A5	B5	C5	D5	NaN	NaN	NaN	NaN
6	NaN	NaN	NaN	NaN	A6	B6	C6	D6	NaN	NaN	NaN	NaN
7	NaN	NaN	NaN	NaN	A7	B7	C7	D7	NaN	NaN	NaN	NaN
8	NaN	A8	B8	C8	D8							
9	NaN	A9	B9	C9	D9							
10	NaN	A10	B10	C10	D10							
11	NaN	A11	B11	C11	D11							

Example Dataframes

```
left = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                     'A': ['A0', 'A1', 'A2', 'A3'],
                     'B': ['B0', 'B1', 'B2', 'B3']})

right = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                      'C': ['C0', 'C1', 'C2', 'C3'],
                      'D': ['D0', 'D1', 'D2', 'D3']})
```

```
left
```

	A	B	key
0	A0	B0	K0
1	A1	B1	K1
2	A2	B2	K2
3	A3	B3	K3

```
right
```

	C	D	key
0	C0	D0	K0
1	C1	D1	K1
2	C2	D2	K2
3	C3	D3	K3

5.2.Merging

```
pd.merge(left,right,how='inner',on='key')
```

	A	B	key	C	D
0	A0	B0	K0	C0	D0
1	A1	B1	K1	C1	D1
2	A2	B2	K2	C2	D2
3	A3	B3	K3	C3	D3

```
left = pd.DataFrame({'key1': ['K0', 'K0', 'K1', 'K2'],
                     'key2': ['K0', 'K1', 'K0', 'K1'],
                     'A': ['A0', 'A1', 'A2', 'A3'],
                     'B': ['B0', 'B1', 'B2', 'B3']})

right = pd.DataFrame({'key1': ['K0', 'K1', 'K1', 'K2'],
                      'key2': ['K0', 'K0', 'K0', 'K0'],
                      'C': ['C0', 'C1', 'C2', 'C3'],
                      'D': ['D0', 'D1', 'D2', 'D3']})
```

```
pd.merge(left, right, on=['key1', 'key2'])
```

	A	B	key1	key2	C	D
0	A0	B0	K0	K0	C0	D0
1	A2	B2	K1	K0	C1	D1
2	A2	B2	K1	K0	C2	D2

```
pd.merge(left, right, how='outer', on=['key1', 'key2'])
```

	A	B	key1	key2	C	D
0	A0	B0	K0	K0	C0	D0
1	A1	B1	K0	K1	NaN	NaN
2	A2	B2	K1	K0	C1	D1
3	A2	B2	K1	K0	C2	D2
4	A3	B3	K2	K1	NaN	NaN
5	NaN	NaN	K2	K0	C3	D3

```
pd.merge(left, right, how='right', on=['key1', 'key2'])
```

	A	B	key1	key2	C	D
0	A0	B0	K0	K0	C0	D0
1	A2	B2	K1	K0	C1	D1
2	A2	B2	K1	K0	C2	D2
3	NaN	NaN	K2	K0	C3	D3

```
pd.merge(left, right, how='left', on=['key1', 'key2'])
```

	A	B	key1	key2	C	D
0	A0	B0	K0	K0	C0	D0
1	A1	B1	K0	K1	NaN	NaN
2	A2	B2	K1	K0	C1	D1
3	A2	B2	K1	K0	C2	D2
4	A3	B3	K2	K1	NaN	NaN

5.3.Joining

```
left = pd.DataFrame({'A': ['A0', 'A1', 'A2'],
                     'B': ['B0', 'B1', 'B2']},
                     index=['K0', 'K1', 'K2'])

right = pd.DataFrame({'C': ['C0', 'C2', 'C3'],
                      'D': ['D0', 'D2', 'D3']},
                      index=['K0', 'K2', 'K3'])
```

```
left.join(right)
```

	A	B	C	D
K0	A0	B0	C0	D0
K1	A1	B1	NaN	NaN
K2	A2	B2	C2	D2

```
left.join(right, how='outer')
```

	A	B	C	D
K0	A0	B0	C0	D0
K1	A1	B1	NaN	NaN
K2	A2	B2	C2	D2
K3	NaN	NaN	C3	D3

6. Operations

```
import pandas as pd
df = pd.DataFrame({'col1':[1,2,3,4], 'col2':[444,555,666,444], 'col3':['abc','def']}
df.head()
```

	col1	col2	col3
0	1	444	abc
1	2	555	def
2	3	666	ghi
3	4	444	xyz

6.1.Info on Unique Values

```
df['col2'].unique()  
array([444, 555, 666])
```

```
df['col2'].nunique()  
3
```

```
df['col2'].value_counts()  
444    2  
555    1  
666    1  
Name: col2, dtype: int64
```

6.2.Selecting Data

```
#Select from DataFrame using criteria from multiple columns  
newdf = df[(df['col1']>2) & (df['col2']==444)]
```

```
newdf
```

col1	col2	col3
3	4	444 xyz

6.3.Apply Functions

```
def times2(x):  
    return x*2
```

df['col1'].apply(times2)	df['col3'].apply(len)	df['col1'].sum()
0 2	0 3	10
1 4	1 3	
2 6	2 3	
3 8	3 3	
Name: col1, dtype: int64	Name: col3, dtype: int64	

```
del df['col1']
```

```
df
```

	col2	col3
0	444	abc
1	555	def
2	666	ghi
3	444	xyz

```
df.columns
```

```
Index(['col2', 'col3'], dtype='object')
```

```
df.index
```

```
RangeIndex(start=0, stop=4, step=1)
```

```
df
```

	col2	col3
0	444	abc
1	555	def
2	666	ghi
3	444	xyz

```
df.sort_values(by='col2')
```

	col2	col3
0	444	abc
3	444	xyz
1	555	def
2	666	ghi

```
df.isnull()
```

	col2	col3
0	False	False
1	False	False
2	False	False
3	False	False

```
# Drop rows with NaN Values  
df.dropna()
```

	col2	col3
0	444	abc
1	555	def
2	666	ghi
3	444	xyz

```
import numpy as np
```

```
df = pd.DataFrame({'col1':[1,2,3,np.nan],  
                   'col2':[np.nan,555,666,444],  
                   'col3':['abc','def','ghi','xyz']})  
df.head()
```

	col1	col2	col3
0	1.0	NaN	abc
1	2.0	555.0	def
2	3.0	666.0	ghi
3	NaN	444.0	xyz

```
df.fillna('FILL')
```

	col1	col2	col3
0	1	FILL	abc
1	2	555	def
2	3	666	ghi
3	FILL	444	xyz

```

data = {'A': ['foo', 'foo', 'foo', 'bar', 'bar', 'bar'],
        'B': ['one', 'one', 'two', 'two', 'one', 'one'],
        'C': ['x', 'y', 'x', 'y', 'x', 'y'],
        'D':[1,3,2,5,4,1]}

df = pd.DataFrame(data)

```

df

	A	B	C	D
0	foo	one	x	1
1	foo	one	y	3
2	foo	two	x	2
3	bar	two	y	5
4	bar	one	x	4
5	bar	one	y	1

```
df.pivot_table(values='D',index=['A', 'B'],columns=['C'])
```

	A	B	C	x	y
bar	one	4.0	1.0		
	two	Nan	5.0		
foo	one	1.0	3.0		
	two	2.0	Nan		

7. Data input and data output

Pandas có thể đọc rất nhiều kiểu file bằng cách sử dụng phương pháp pd.read.

7.1. CSV

CSV Input

```

df = pd.read_csv('example')
df

```

	a	b	c	d
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

CSV Output

```
df.to_csv('example',index=False)
```

7.2. Excel

Excel Input

```
pd.read_excel('Excel_Sample.xlsx',sheetname='Sheet1')
```

	a	b	c	d
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15

Excel Output

```
df.to_excel('Excel_Sample.xlsx',sheet_name='Sheet1')
```

7.3. HTML

Chúng ta cần tải thư viện lxml, html5lib, BeautifulSoup4 để đọc file.

HTML Input

```
In [5]: df = pd.read_html('http://www.fdic.gov/bank/individual/failed/banklist.html')
```

```
In [7]: df[0]
```

	Bank Name	City	ST	CERT	Acquiring Institution	Closing Date	Updated Date	Loss Share Type	Agreement Terminated	Termination Date
0	First CornerStone Bank	King of Prussia	PA	35312	First-Citizens Bank & Trust Company	May 6, 2016	July 12, 2016	none	NaN	NaN
1	Trust Company Bank	Memphis	TN	9956	The Bank of Fayette County	April 29, 2016	August 4, 2016	none	NaN	NaN
2	North Milwaukee State Bank	Milwaukee	WI	20364	First-Citizens Bank & Trust Company	March 11, 2016	June 16, 2016	none	NaN	NaN
3	Hometown National Bank	Longview	WA	35156	Twin City Bank	October 2, 2015	April 13, 2016	none	NaN	NaN
4	The Bank of Georgia	Peachtree City	GA	35259	Fidelity Bank	October 2, 2015	April 13, 2016	none	NaN	NaN
5	First Bank of America	Albuquerque	NM	35000	United Fidelity	July 10, 2015	July 12, 2015

7.4.SQL

Dùng các hàm sau:

- `read_sql_table(table_name, con[, schema, ...])`
 - Đọc database SQL thành một dataframe.
- `read_sql_query(sql, con[, index_col, ...])`
 - Đọc truy vấn SQL thành một dataframe.
- `read_sql(sql, con[, index_col, ...])`
 - Đọc database SQL và truy vấn SQL thành một dataframe.
- `DataFrame.to_sql(name, con[, flavor, ...])`
 - Viết một lưu trữ trong dataframe thành một database SQL.

```
from sqlalchemy import create_engine
engine = create_engine('sqlite:///memory:')
df.to_sql('data', engine)
sql_df = pd.read_sql('data', con=engine)
sql_df
```

index	a	b	c	d	
0	0	0	1	2	3
1	1	4	5	6	7
2	2	8	9	10	11
3	3	12	13	14	15

PANDAS PART 2

1.	Series	2
2.	Pandas DataFrame	3
3.	Pandas Data Cleaning	24
4.	Pandas Data Aggregation.....	37
5.	Pandas Feature Engineering.....	55
6.	Concatenating, Merging and Joining DataFrames in Pandas	55
7.	Basic Time Series Analysis	61
8.	Advanced – iterows, vectorization and numpy.....	67
9.	Advanced – map, zip and apply.....	69
10.	Advanced – Parallel Processing.....	71

1. Series

Serie là data dạng cột 1 chiều

```
# Creating a simple python list
```

```
a = [1,4,3,7,2,6,3]
```

```
a
```

```
[1, 4, 3, 7, 2, 6, 3]
```

```
# Now let's convert it into a pandas object called a series
```

```
# NOTE: A pandas series is a single dimension column of data e.g.
```

```
a = pd.Series(a)
```

```
a
```

```
0    1  
1    4  
2    3  
3    7  
4    2  
5    6  
6    3  
dtype: int64
```

```
type(a)
```

```
pandas.core.series.Series
```

Các toán tử cơ bản trong series

	a.sort_values()	a.sort_values(ascending=False)
a.max()	0 1 4 2 2 3 6 3 1 4 5 6 3 7 1	3 7 5 6 1 4 6 3 2 3 4 2 0 1 dtype: int64
7		
a.min()		

Serie sẽ không bị thay đổi khi dùng các lệnh này

```
a
```

```
0    1  
1    4  
2    3  
3    7  
4    2  
5    6  
6    3  
dtype: int64
```

Sắp xếp a lớn tới bé

```
a.sort_values(ascending=False, inplace=True)
```

```
a  
3    7  
5    6  
1    4  
2    3  
6    3  
4    2  
0    1  
dtype: int64
```

Đếm tần số xuất hiện

```
: a.value_counts()
```

```
: 3    2  
7    1  
6    1  
4    1  
2    1  
1    1  
dtype: int64
```

In ra serie từ đầu tới hàng 2

```
a[:2]
```

```
3    7  
5    6  
dtype: int64
```

2. Pandas DataFrame

Load data từ nguồn github

```
import pandas as pd

# Load our CSV file using pandas
file_name = "https://raw.githubusercontent.com/rajeevratan84/datascienceforbusiness/master/titanic.csv"
df = pd.read_csv(file_name)

df.head()
```

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	survived
0	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	1
1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	1
2	1	Allison, Miss. Helen Lorraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	0
3	1	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	0
4	1	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	113781	151.5500	C22 C26	S	0

Load đến 10 hàng

```
df.head(10)
```

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	survived
0	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	0
1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	0
2	1	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	0
3	1	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	0
4	1	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	113781	151.5500	C22 C26	S	0
5	1	Anderson, Mr. Harry	male	48.0000	0	0	19952	26.5500	E12	S	0
6	1	Andrews, Miss. Kornelia Theodosia	female	63.0000	1	0	13502	77.9583	D7	S	0
7	1	Andrews, Mr. Thomas Jr	male	39.0000	0	0	112050	0.0000	A36	S	0
8	1	Appleton, Mrs. Edward Dale (Charlotte Lamson)	female	53.0000	2	0	11769	51.4792	C101	S	0
9	1	Artagaveytia, Mr. Ramon	male	71.0000	0	0	PC 17609	49.5042	NaN	C	0

Lưu data trên vào file CSV hoặc là vào file Excel

```
# Saving to a CSV file
df.to_csv("myDataFrame2.csv")

# Of even an excel file
df.to_excel("myDataFrame2.xlsx")
```

In ra 5 hàng đầu của data

```
df.head()
```

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	survived
0	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	1
1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	1
2	1	Allison, Miss. Helen Lorraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	0
3	1	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	0
4	1	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	113781	151.5500	C22 C26	S	0

Xem mô tả về dataframe

```
df.describe()
```

	pclass	age	sibsp	parch	fare	survived
count	1309.000000	1046.000000	1309.000000	1309.000000	1308.000000	1309.000000
mean	2.294882	29.881135	0.498854	0.385027	33.295479	0.381971
std	0.837836	14.413500	1.041658	0.865560	51.758668	0.486055
min	1.000000	0.166700	0.000000	0.000000	0.000000	0.000000
25%	2.000000	21.000000	0.000000	0.000000	7.895800	0.000000
50%	3.000000	28.000000	0.000000	0.000000	14.454200	0.000000
75%	3.000000	39.000000	1.000000	0.000000	31.275000	1.000000
max	3.000000	80.000000	8.000000	9.000000	512.329200	1.000000

Xem thông tin về dataframe

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 11 columns):
pclass      1309 non-null int64
name        1309 non-null object
sex         1309 non-null object
age         1046 non-null float64
sibsp       1309 non-null int64
parch       1309 non-null int64
ticket      1309 non-null object
fare         1308 non-null float64
cabin        295 non-null object
embarked    1307 non-null object
survived    1309 non-null int64
dtypes: float64(2), int64(4), object(5)
memory usage: 112.6+ KB
#
```

Tạo ra dataframe không từ nguồn CSV nào

DataFrame có thể load từ excel, list python, dict và nhiều nữa

Tạo 1 list

```
list_a = ['a', 'b', 'c']
list_a
['a', 'b', 'c']
```

Biến nó thành dataframe

```
df_a = pd.DataFrame(list_a)
df_a
0
0 a
1 b
2 c
```

Thêm các giá trị cũng như tên hàng cột

```

dict_df = pd.DataFrame({
    'http_status': [200, 200, 404, 404, 301],
    'response_time': [0.04, 0.02, 0.07, 0.08, 1.0]})  

dict_df

```

	http_status	response_time
0	200	0.04
1	200	0.02
2	404	0.07
3	404	0.08
4	301	1.00

Kiểm tra DataFrame

In ra 10 hàng cuối

df.tail(10)												
	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	survived	
1299	3	Yasbeck, Mr. Antoni	male	27.0	1	0	2659	14.4542	NaN	C	0	
1300	3	Yasbeck, Mrs. Antoni (Selini Alexander)	female	15.0	1	0	2659	14.4542	NaN	C	1	
1301	3	Youseff, Mr. Gerious	male	45.5	0	0	2628	7.2250	NaN	C	0	
1302	3	Yousif, Mr. Wazli	male	NaN	0	0	2647	7.2250	NaN	C	0	
1303	3	Yousseff, Mr. Gerious	male	NaN	0	0	2627	14.4583	NaN	C	0	
1304	3	Zabour, Miss. Hileni	female	14.5	1	0	2665	14.4542	NaN	C	0	
1305	3	Zabour, Miss. Thamine	female	NaN	1	0	2665	14.4542	NaN	C	0	
1306	3	Zakarian, Mr. Mapriededer	male	26.5	0	0	2656	7.2250	NaN	C	0	
1307	3	Zakarian, Mr. Ortin	male	27.0	0	0	2670	7.2250	NaN	C	0	
1308	3	Zimmerman, Mr. Leo	male	29.0	0	0	315082	7.8750	NaN	S	0	

Xem kiểu dữ liệu của dataframe

```
df.dtypes
```

```
pclass      int64
name        object
sex         object
age       float64
sibsp      int64
parch      int64
ticket     object
fare       float64
cabin     object
embarked   object
survived   int64
dtype: object
```

Xem tên hàng

```
df.columns
```

```
Index(['pclass', 'name', 'sex', 'age', 'sibsp', 'parch', 'ticket', 'fare',
       'cabin', 'embarked', 'survived'],
      dtype='object')
```

Xem cột name

```
df[ 'name' ]
```

```
0          Allen, Miss. Elisabeth Walton
1          Allison, Master. Hudson Trevor
2          Allison, Miss. Helen Loraine
3          Allison, Mr. Hudson Joshua Creighton
4          Allison, Mrs. Hudson J C (Bessie Waldo Daniels)
5          Anderson, Mr. Harry
6          Andrews, Miss. Kornelia Theodosia
7          Andrews, Mr. Thomas Jr
8          Appleton, Mrs. Edward Dale (Charlotte Lamson)
9          Artagaveytia, Mr. Ramon
10         Astor, Col. John Jacob
11         Astor, Mrs. John Jacob (Madeleine Talmadge Force)
12         Aubart, Mme. Leontine Pauline
13         Barber, Miss. Ellen "Nellie"
14         Barkworth, Mr. Algernon Henry Wilson
15         Baumann, Mr. John D
16         Baxter, Mr. Quigg Edmond
17         Baxter, Mrs. James (Helene DeLaudeniere Chaput)
18         Bazzani, Miss. Albina
19         Beattie, Mr. Thomson
20         Beckwith, Mr. Richard Leonard
21         Beckwith, Mrs. Richard Leonard (Sallie Monypeny)
22         Behr, Mr. Karl Howell
23         Bidois, Miss. Rosalie
24         Bird, Miss. Ellen
25         Birnbaum, Mr. Jakob
26         Bishop, Mr. Dickinson H
```

Lọc ra các khoảng từ DataFrame

```
df.loc[1,"name"]
```

```
'Allison, Master. Hudson Trevor'
```

In ra toàn bộ 1 cột

```
df.loc[:, "name"]
```

```
0 Allen, Miss. Elisabeth Walton
1 Allison, Master. Hudson Trevor
2 Allison, Miss. Helen Loraine
3 Allison, Mr. Hudson Joshua Creighton
4 Allison, Mrs. Hudson J C (Bessie Waldo Daniels)
5 Anderson, Mr. Harry
6 Andrews, Miss. Kornelia Theodosia
7 Andrews, Mr. Thomas Jr
8 Appleton, Mrs. Edward Dale (Charlotte Lamson)
9 Artagaveytia, Mr. Ramon
10 Astor, Col. John Jacob
11 Astor, Mrs. John Jacob (Madeleine Talmadge Force)
12 Aubart, Mme. Leontine Pauline
13 Barber, Miss. Ellen "Nellie"
14 Barkworth, Mr. Algernon Henry Wilson
15 Baumann, Mr. John D
16 Baxter, Mr. Quigg Edmond
17 Baxter, Mrs. James (Helene DeLaudeniere Chaput)
18 Bazzani, Miss. Albina
19 Beattie, Mr. Thomson
20 Beckwith, Mr. Richard Leonard
21 Beckwith, Mrs. Richard Leonard (Sallie Monypeny)
22 Behr, Mr. Karl Howell
23 Bidois, Miss. Rosalie
24 Bird, Miss. Ellen
25 Birnbaum, Mr. Jakob
26 Bishop, Mr. Dickinson H
27 Bishop, Mrs. Dickinson H (Helen Walton)
28 Bissette, Miss. Amelia
29 Bjornstrom-Steffansson, Mr. Mauritz Hakan
...
1279 Vestrom, Miss. Hulda Amanda Adolfina
1280 Vovk, Mr. Janko
1281 Waelens, Mr. Achille
1282 Ware, Mr. Frederick
1283 Warren, Mr. Charles William
1284 Webber, Mr. James
1285 Wenzel, Mr. Linhart
1286 Whabee, Mrs. George Joseph (Shawneene Abi-Saab)
1287 Widegren, Mr. Carl/Charles Peter
```

```
names = df.loc[:, "name"]
```

```
names.head()
```

```
0 Allen, Miss. Elisabeth Walton
1 Allison, Master. Hudson Trevor
2 Allison, Miss. Helen Loraine
3 Allison, Mr. Hudson Joshua Creighton
4 Allison, Mrs. Hudson J C (Bessie Waldo Daniels)
Name: name, dtype: object
```

In ra 1 hàng

```
df.loc[0,:]
```

```
pclass          1  
name    Allen, Miss. Elisabeth Walton  
sex            female  
age             29  
sibsp            0  
parch            0  
ticket      24160  
fare        211.338  
cabin           B5  
embarked          S  
survived         1  
Name: 0, dtype: object
```

Chia nhỏ dữ liệu từ dataframe bằng câu lệnh iloc

df.iloc[số hàng, số cột]

```
df.iloc[0,1]
```

```
'Allen, Miss. Elisabeth Walton'
```

```
df.iloc[0]
```

```
pclass          1  
name    Allen, Miss. Elisabeth Walton  
sex            female  
age             29  
sibsp            0  
parch            0  
ticket      24160  
fare        211.338  
cabin           B5  
embarked          S  
survived         1  
Name: 0, dtype: object
```

Lưu ý: data 1 chiều của pandas là series, data 2 chiều của pandas là dataframe.

Chia nhỏ nhiều cột

```
df.loc[:,["name", "sex"]]
```

	name	sex
0	Allen, Miss. Elisabeth Walton	female
1	Allison, Master. Hudson Trevor	male
2	Allison, Miss. Helen Loraine	female
3	Allison, Mr. Hudson Joshua Creighton	male
4	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female
5	Anderson, Mr. Harry	male
6	Andrews, Miss. Kornelia Theodosia	female
7	Andrews, Mr. Thomas Jr	male
8	Appleton, Mrs. Edward Dale (Charlotte Lamson)	female
9	Artagaveytia, Mr. Ramon	male
10	Astor, Col. John Jacob	male
11	Astor, Mrs. John Jacob (Madeleine Talmadge Force)	female
12	Aubart, Mme. Leontine Pauline	female
13	Barber, Miss. Ellen "Nellie"	female
14	Barkworth, Mr. Algernon Henry Wilson	male
15	Baumann, Mr. John D	male
16	Baxter, Mr. Quigg Edmond	male
17	Baxter, Mrs. James (Helene DeLaudeniere Chaput)	female
18	Bazzani, Miss. Albina	female
19	Beattie, Mr. Thomson	male
20	Beckwith, Mr. Richard Leonard	male
21	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female

Một cách khác để chia nhỏ cột

```
df[["name", "sex"]]
```

	name	sex
0	Allen, Miss. Elisabeth Walton	female
1	Allison, Master. Hudson Trevor	male
2	Allison, Miss. Helen Loraine	female
3	Allison, Mr. Hudson Joshua Creighton	male
4	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female
5	Anderson, Mr. Harry	male
6	Andrews, Miss. Kornelia Theodosia	female
7	Andrews, Mr. Thomas Jr	male
8	Appleton, Mrs. Edward Dale (Charlotte Lamson)	female
9	Artagaveytia, Mr. Ramon	male
10	Astor, Col. John Jacob	male
11	Astor, Mrs. John Jacob (Madeleine Talmadge Force)	female
12	Aubart, Mme. Leontine Pauline	female
13	Barber, Miss. Ellen "Nellie"	female
14	Barkworth, Mr. Algernon Henry Wilson	male
15	Baumann, Mr. John D	male
16	Baxter, Mr. Quigg Edmond	male
17	Baxter, Mrs. James (Helene DeLaudeniere Chaput)	female
18	Bazzani, Miss. Albina	female
19	Beattie, Mr. Thomson	male
20	Beckwith, Mr. Richard Leonard	male
21	Beckwith, Mrs. Richard Leonard (Sallie Monypeny)	female
22	Behr, Mr. Karl Howell	male
23	Bidois, Miss. Rosalie	female
24	Bird, Miss. Ellen	female
25	Birnbaum, Mr. Jakob	male
26	Bishop, Mr. Dickinson H	male
27	Bishop, Mrs. Dickinson H (Helen Walton)	female

Chia nhỏ hàng cũng tương tự như thế

```
df.loc[2:4,:]
```

pclass		name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	survived
2	1	Allison, Miss. Helen Loraine	female	2.0	1	2	113781	151.55	C22 C26	S	0
3	1	Allison, Mr. Hudson Joshua Creighton	male	30.0	1	2	113781	151.55	C22 C26	S	0
4	1	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0	1	2	113781	151.55	C22 C26	S	0

Toán tử trong series

Tạo một dataframe

```
simple_df = pd.DataFrame({  
    'a': [1,2,3,4,5],  
    'b': [5, 15, 10, 20, 15]})  
simple_df
```

	a	b
0	1	5
1	2	15
2	3	10
3	4	20
4	5	15

Tách ra thành 1 serie a

```
a = simple_df['a']  
a
```

```
0    1  
1    2  
2    3  
3    4  
4    5  
Name: a, dtype: int64
```

Tách ra thành 1 serie b

```
b = simple_df['b']
```

```
b  
0    5  
1   15  
2   10  
3   20  
4   15  
Name: b, dtype: int64
```

Cộng 2 series a, b lại

```
c = a + b  
c
```

```
c  
0    6  
1   17  
2   13  
3   24  
4   20  
dtype: int64
```

Hàm min, max

```
a.max()
```

```
5
```

```
a.min()
```

```
1
```

Tìm đại lượng trung bình

```
a.mean()
```

```
3.0
```

```
a
```

```
a  
0    1  
1    2  
2    3  
3    4  
4    5  
Name: a, dtype: int64
```

Tìm trung vị trong series

```
a
```

```
0    1  
1    2  
2    3  
3    4  
4    5  
Name: a, dtype: int64
```

```
a  
a.median()
```

```
3.0
```

Tìm yếu vị trong series, giá trị xuất hiện nhiều nhất

```
b.mode()
```

```
0    15  
dtype: int64
```

```
b
```

```
0    5  
1    15  
2    10  
3    20  
4    15  
Name: b, dtype: int64
```

Đếm tần số

```
b.value_counts()
```

```
15    2  
5     1  
20    1  
10    1  
Name: b, dtype: int64
```

Thông tin mô tả series

```
b.describe()
```

```
count      5.000000
mean      13.000000
std       5.700877
min       5.000000
25%      10.000000
50%      15.000000
75%      15.000000
max      20.000000
Name: b, dtype: float64
```

Thông tin mô tả một trường mình chọn

```
df['sex'].describe()
```

```
# count - number of rows or values in that column
# unique - number of unique categories in the column
# top - most populous category
# freq - count of the most popular category
```

```
count      1309
unique      2
top       male
freq       843
Name: sex, dtype: object
```

Xem giá trị độc nhất

```
df['pclass'].unique()
```

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

Descriptive Stats on DataFrame

Tìm đại lượng trung bình từ 2 cột “fare” và “age”

```
df[["fare", "age"]].mean()
```

```
fare    33.295479
age     29.881135
dtype: float64
```

Câu lệnh trên tương đương với trực bằng 0, nếu chúng ta dùng axis = 1 thì chúng ta đang tính theo hàng

```
df[["fare", "age"]].mean(axis=1)
```

```
0      120.16875
1      76.23335
2      76.77500
3      90.77500
4      88.27500
...
1304    14.47710
1305    14.45420
1306    16.86250
1307    17.11250
1308    18.43750
Length: 1309, dtype: float64
```

Lọc dữ liệu bằng toán tử Boolean

```
df['pclass'] == 1
```

```
0      True
1      True
2      True
3      True
4      True
5      True
6      True
7      True
8      True
9      True
10     True
11     True
12     True
13     True
14     True
15     True
16     True
17     True
18     True
19     True
20     True
21     True
22     True
23     True
24     True
25     True
26     True
27     True
28     True
29     True
```

```
df['age'] > 70
```

```
0      False
1      False
2      False
3      False
4      False
...
1304    False
1305    False
1306    False
1307    False
1308    False
Name: age, Length: 1309, dtype: bool
```

Xem giá trị mình vừa lọc

```
df[df['age'] > 70]
```

	pclass		name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	survived
9	1	Artagaveytia, Mr. Ramon	male	71.0	0	0	PC 17609	49.5042	NaN	C	0	
14	1	Barkworth, Mr. Algernon Henry Wilson	male	80.0	0	0	27042	30.0000	A23	S	1	
61	1	Cavendish, Mrs. Tyrell William (Julia Florence...)	female	76.0	1	0	19877	78.8500	C46	S	1	
135	1	Goldschmidt, Mr. George B	male	71.0	0	0	PC 17754	34.6542	A5	C	0	
727	3	Connors, Mr. Patrick	male	70.5	0	0	370369	7.7500	NaN	Q	0	
1235	3	Svensson, Mr. Johan	male	74.0	0	0	347060	7.7750	NaN	S	0	

Lọc dữ liệu với giá trị cột có sẵn

```
df[df['name'] == 'Barkworth, Mr. Algernon Henry Wilson']
```

	pclass		name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	survived
14	1	Barkworth, Mr. Algernon Henry Wilson	male	80.0	0	0	27042	30.0	A23	S	1	

Điền vào những ô giá trị trống trong dataframe bằng câu lệnh isnull()

```
df.isnull()
```

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	survived
0	False	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False	False
...
1304	False	False	False	False	False	False	False	False	True	False	False
1305	False	False	False	True	False	False	False	False	True	False	False
1306	False	False	False	False	False	False	False	False	True	False	False
1307	False	False	False	False	False	False	False	False	True	False	False
1308	False	False	False	False	False	False	False	False	True	False	False

1309 rows × 11 columns

Kiểm tra cột “cabin”

```
df['cabin'].isnull()
```

```
0      False
1      False
2      False
3      False
4      False
...
1304     True
1305     True
1306     True
1307     True
1308     True
Name: cabin, Length: 1309, dtype: bool
```

```
df[df['cabin'].isnull()]
```

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	survived
9	1	Artagaveytia, Mr. Ramon	male	71.0	0	0	PC 17609	49.5042	NaN	C	
13	1	Barber, Miss. Ellen "Nellie"	female	26.0	0	0	19877	78.8500	NaN	S	
15	1	Baumann, Mr. John D	male	Nan	0	0	PC 17318	25.9250	NaN	S	
23	1	Bidois, Miss. Rosalie	female	42.0	0	0	PC 17757	227.5250	NaN	C	
25	1	Birnbaum, Mr. Jakob	male	25.0	0	0	13905	26.0000	NaN	C	
...
1304	3	Zabour, Miss. Hileni	female	14.5	1	0	2665	14.4542	NaN	C	
1305	3	Zabour, Miss. Thamine	female	Nan	1	0	2665	14.4542	NaN	C	
1306	3	Zakarian, Mr. Mapriededer	male	26.5	0	0	2656	7.2250	NaN	C	
1307	3	Zakarian, Mr. Ortin	male	27.0	0	0	2670	7.2250	NaN	C	
1308	3	Zimmerman, Mr. Leo	male	29.0	0	0	315082	7.8750	NaN	S	

1014 rows × 11 columns

```
df[df['cabin'].notnull()]
```

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	survived
0	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	1
1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	1
2	1	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	0
3	1	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	0
4	1	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	113781	151.5500	C22 C26	S	0
...
1189	3	Sandstrom, Miss. Marguerite Rut	female	4.0000	1	1	PP 9549	16.7000	G6	S	1
1217	3	Soholt, Mr. Peter Andreas Lauritz Andersen	male	19.0000	0	0	348124	7.6500	F G73	S	0
1230	3	Strom, Miss. Telma Matilda	female	2.0000	0	1	347054	10.4625	G6	S	0
1231	3	Strom, Mrs. Wilhelm (Elna Matilda Persson)	female	29.0000	1	1	347054	10.4625	G6	S	0
1249	3	Tobin, Mr. Roger	male	Nan	0	0	383121	7.7500	F38	Q	0

295 rows × 11 columns

Kết hợp toán tử Boolean để lọc dữ liệu

```

over_60 = df['age'] > 60
cabin_exists = df['cabin'].notnull()

over_60_cabin_exists = over_60 + cabin_exists

df[over_60_cabin_exists]

```

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	survived
0	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	1
1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	1
2	1	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	0
3	1	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	0
4	1	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	113781	151.5500	C22 C26	S	0
...
1230	3	Strom, Miss. Telma Matilda	female	2.0000	0	1	347054	10.4625	G6	S	0
1231	3	Strom, Mrs. Wilhelm (Elna Matilda Persson)	female	29.0000	1	1	347054	10.4625	G6	S	0
1235	3	Svensson, Mr. Johan	male	74.0000	0	0	347060	7.7750	NaN	S	0
1249	3	Tobin, Mr. Roger	male	NaN	0	0	383121	7.7500	F38	Q	0
1261	3	Turkula, Mrs. (Hedwig)	female	63.0000	0	0	4134	9.5875	NaN	S	1

310 rows × 11 columns

```

over_60_and_under_5 = (df['age'] > 60) | (df['age'] < 5)

```

```

df[over_60_and_under_5]

```

	pclass	name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	survived
1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	1
2	1	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	0
6	1	Andrews, Miss. Kornelia Theodosia	female	63.0000	1	0	13502	77.9583	D7	S	1
9	1	Artagaveytia, Mr. Ramon	male	71.0000	0	0	PC 17609	49.5042	NaN	C	0
14	1	Barkworth, Mr. Algernon Henry Wilson	male	80.0000	0	0	27042	30.0000	A23	S	1
...
1225	3	Storey, Mr. Thomas	male	60.5000	0	0	3701	NaN	NaN	S	0
1230	3	Strom, Miss. Telma Matilda	female	2.0000	0	1	347054	10.4625	G6	S	0
1235	3	Svensson, Mr. Johan	male	74.0000	0	0	347060	7.7750	NaN	S	0
1240	3	Thomas, Master. Assad Alexander	male	0.4167	0	1	2625	8.5167	NaN	C	1
1261	3	Turkula, Mrs. (Hedwig)	female	63.0000	0	0	4134	9.5875	NaN	S	1

84 rows × 11 columns

3. Pandas Data Cleaning

```
import pandas as pd

file_name = "https://raw.githubusercontent.com/rajeevratan84/datascienceforbusiness/master/amazon_fires.csv"
df = pd.read_csv(file_name, encoding = "ISO-8859-1")

df.tail()
```

	ano	mes	estado	numero	encontro
6449	2012	Dezembro	Tocantins	128	1/1/2012
6450	2013	Dezembro	Tocantins	85	1/1/2013
6451	2014	Dezembro	Tocantins	223	1/1/2014
6452	2015	Dezembro	Tocantins	373	1/1/2015
6453	2016	Dezembro	Tocantins	119	1/1/2016

Lọc ra dữ liệu độc nhất từ cột “estado”

```
df['estado'].unique()

array(['Acre', 'alagoas', 'Amapa', 'Amazonas', 'Bahia', 'Ceara',
       'Distrito Federal', 'Espirito Santo', 'Goias', 'Maranhao',
       'Mato Grosso', 'Minas Gerais', 'pará', 'Paraiba', 'Pernambuco',
       'Piau', 'Rio', 'rondonia', 'Roraima', 'Santa Catarina',
       'Sao Paulo', 'Sergipe', 'Tocantins'], dtype=object)
```

Đặt lại tên cho hàng

```
new_columns = {'ano' : 'year',
               'estado': 'state',
               'mes': 'month',
               'numero': 'number_of_fires',
               'encontro': 'date'}

df.rename(columns = new_columns, inplace=True)
```

```
df.head()
```

	year	month	state	number_of_fires	date
0	1998	Janeiro	Acre	0 Fires	1/1/1998
1	1999	Janeiro	Acre	0 Fires	1/1/1999
2	2000	Janeiro	Acre	0 Fires	1/1/2000
3	2001	Janeiro	Acre	0 Fires	1/1/2001
4	2002	Janeiro	Acre	0 Fires	1/1/2002

Lọc ra dữ liệu độc nhất từ cột “year”

```
df['year'].unique()

array([1998, 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008,
       2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017])
```

Thông tin về dataframe

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6454 entries, 0 to 6453
Data columns (total 5 columns):
year           6454 non-null int64
month          6454 non-null object
state          6454 non-null object
number_of_fires 6322 non-null object
date           6454 non-null object
dtypes: int64(1), object(4)
memory usage: 252.2+ KB
```

```
df.head()
```

	year	month	state	number_of_fires	date
0	1998	Janeiro	Acre	0 Fires	1/1/1998
1	1999	Janeiro	Acre	0 Fires	1/1/1999
2	2000	Janeiro	Acre	0 Fires	1/1/2000
3	2001	Janeiro	Acre	0 Fires	1/1/2001
4	2002	Janeiro	Acre	0 Fires	1/1/2002

Sắp xếp lại hàng

```
new_order = [4,1,0,2,3,]
df = df[df.columns[new_order]]
df.head()
```

	date	month	year	state	number_of_fires
0	1/1/1998	Janeiro	1998	Acre	0 Fires
1	1/1/1999	Janeiro	1999	Acre	0 Fires
2	1/1/2000	Janeiro	2000	Acre	0 Fires
3	1/1/2001	Janeiro	2001	Acre	0 Fires
4	1/1/2002	Janeiro	2002	Acre	0 Fires

In ra 25 hàng dữ liệu

```
df.head(25)
```

	date	month	year	state	number_of_fires
0	1/1/1998	Janeiro	1998	Acre	0 Fires
1	1/1/1999	Janeiro	1999	Acre	0 Fires
2	1/1/2000	Janeiro	2000	Acre	0 Fires
3	1/1/2001	Janeiro	2001	Acre	0 Fires
4	1/1/2002	Janeiro	2002	Acre	0 Fires
5	1/1/2003	Janeiro	2003	Acre	10 Fires
6	1/1/2004	Janeiro	2004	Acre	0 Fires
7	1/1/2005	Janeiro	2005	Acre	12 Fires
8	1/1/2006	Janeiro	2006	Acre	4 Fires
9	1/1/2007	Janeiro	2007	Acre	0 Fires
10	1/1/2008	Janeiro	2008	Acre	0 Fires
11	1/1/2009	Janeiro	2009	Acre	0 Fires
12	1/1/2010	Janeiro	2010	Acre	1 Fires
13	1/1/2011	Janeiro	2011	Acre	0 Fires
14	1/1/2012	Janeiro	2012	Acre	0 Fires
15	1/1/2013	Janeiro	2013	Acre	0 Fires
16	1/1/2014	Janeiro	2014	Acre	0 Fires
17	1/1/2015	Janeiro	2015	Acre	1 Fires
18	1/1/2016	Janeiro	2016	Acre	12 Fires
19	1/1/2017	Janeiro	2017	Acre	0 Fires
20	1/1/1998	Fevereiro	1998	Acre	0 Fires
21	1/1/1999	Fevereiro	1999	Acre	0 Fires
22	1/1/2000	Fevereiro	2000	Acre	0 Fires
23	1/1/2001	Fevereiro	2001	Acre	0 Fires
24	1/1/2002	Fevereiro	2002	Acre	1 Fires

In ra 5 hàng cuối dữ liệu

```
df.tail()
```

	date	month	year	state	number_of_fires
6449	1/1/2012	Dezembro	2012	Tocantins	128
6450	1/1/2013	Dezembro	2013	Tocantins	85
6451	1/1/2014	Dezembro	2014	Tocantins	223
6452	1/1/2015	Dezembro	2015	Tocantins	373
6453	1/1/2016	Dezembro	2016	Tocantins	119

Determining if a column contains numeric data

Xét xem có phải là numeric không

```
df['number_of_fires'].str.isnumeric()
```

```
0      False
1      False
2      False
3      False
4      False
...
6449     True
6450     True
6451     True
6452     True
6453     True
Name: number_of_fires, Length: 6454, dtype: object
```

Xét xem có phải là digit không

```
df['number_of_fires'].astype(str).str.isdigit()
```

```
0      False
1      False
2      False
3      False
4      False
...
6449     True
6450     True
6451     True
6452     True
6453     True
Name: number_of_fires, Length: 6454, dtype: bool
```

Loại bỏ những ký tự không cần thiết từ cột

```
df['number_of_fires'].str.strip(" Fires")
0      0
1      0
2      0
3      0
4      0
...
6449    128
6450     85
6451    223
6452    373
6453    119
Name: number_of_fires, Length: 6454, dtype: object
```

Thay thế cột với 1 cột trống

```
df['number_of_fires'] = df['number_of_fires'].str.strip(" Fires")
df.head()
```

	date	month	year	state	number_of_fires
0	1/1/1998	Janeiro	1998	Acre	0
1	1/1/1999	Janeiro	1999	Acre	0
2	1/1/2000	Janeiro	2000	Acre	0
3	1/1/2001	Janeiro	2001	Acre	0
4	1/1/2002	Janeiro	2002	Acre	0

Thông tin về dataframe

```
df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6454 entries, 0 to 6453
Data columns (total 5 columns):
date            6454 non-null object
month           6454 non-null object
year            6454 non-null int64
state           6454 non-null object
number_of_fires 6322 non-null object
dtypes: int64(1), object(4)
memory usage: 252.2+ KB
```

Chuyển cột “number_of_fires” thành kiểu dữ liệu float

```
df["number_of_fires"] = df["number_of_fires"].str.replace('','0').astype(float)
df.head()
```

	date	month	year	state	number_of_fires
0	1/1/1998	Janeiro	1998	Acre	0.0
1	1/1/1999	Janeiro	1999	Acre	0.0
2	1/1/2000	Janeiro	2000	Acre	0.0
3	1/1/2001	Janeiro	2001	Acre	0.0
4	1/1/2002	Janeiro	2002	Acre	0.0

Xử lý dữ liệu bị mất

```
file_name = "https://raw.githubusercontent.com/rajeevratan84/datascienceforbusiness/master/amazon_fires.csv"
df = pd.read_csv(file_name, encoding = "ISO-8859-1")
new_columns = {'ano' : 'year',
               'estado': 'state',
               'mes': 'month',
               'numero': 'number_of_fires',
               'encontro': 'date'}
df.rename(columns = new_columns, inplace=True)
df['number_of_fires'] = df['number_of_fires'].str.strip(" Fires")
# Creating a true copy of our dataframe
df_copy = df.copy()
df.head()
```

	year	month	state	number_of_fires	date
0	1998	Janeiro	Acre	0	1/1/1998
1	1999	Janeiro	Acre	0	1/1/1999
2	2000	Janeiro	Acre	0	1/1/2000
3	2001	Janeiro	Acre	0	1/1/2001
4	2002	Janeiro	Acre	0	1/1/2002

Xem tổng các dữ liệu bị mất

```
df.isnull().sum()
```

```
year          0
month         0
state         0
number_of_fires    132
date          0
dtype: int64
```

Loại bỏ những giá trị NULL hoặc NaN

```
# Drop rows with NaN values
df = df.dropna()
df = df.reset_index() # reset's row indexes in case any rows were dropped
df.head()
```

	index	year	month	state	number_of_fires	date
0	0	1998	Janeiro	Acre	0	1/1/1998
1	1	1999	Janeiro	Acre	0	1/1/1999
2	2	2000	Janeiro	Acre	0	1/1/2000
3	3	2001	Janeiro	Acre	0	1/1/2001
4	4	2002	Janeiro	Acre	0	1/1/2002

Check lại tổng các dữ liệu bị mất

```
df.isnull().sum()
```

```
index          0
year           0
month          0
state          0
number_of_fires 0
date           0
dtype: int64
```

Tải lại dataframe để xem

```
file_name = "https://raw.githubusercontent.com/rajeevratan84/datascienceforbusiness/master/amazon_fires.csv"
df = pd.read_csv(file_name, encoding = "ISO-8859-1")
new_columns = {'ano' : 'year',
              'estado': 'state',
              'mes': 'month',
              'numero': 'number_of_fires',
              'encontro': 'date'}
df.rename(columns = new_columns, inplace=True)
df['number_of_fires'] = df['number_of_fires'].str.strip(" Fires")
df_copy = df.copy()
df.head()
```

	year	month	state	number_of_fires	date
0	1998	Janeiro	Acre	0	1/1/1998
1	1999	Janeiro	Acre	0	1/1/1999
2	2000	Janeiro	Acre	0	1/1/2000
3	2001	Janeiro	Acre	0	1/1/2001
4	2002	Janeiro	Acre	0	1/1/2002

Tạo một boolean index cho tất cả giá trị NULL

```
df['number_of_fires'].isnull()  
  
0      False  
1      False  
2      False  
3      False  
4      False  
...  
6449    False  
6450    False  
6451    False  
6452    False  
6453    False  
Name: number_of_fires, Length: 6454, dtype: bool
```

```
df[df['number_of_fires'].isnull()].head(10)
```

	year	month	state	number_of_fires	date
68	2006	Abril	Acre	NaN	1/1/2006
110	2008	Junho	Acre	NaN	1/1/2008
127	2005	Julho	Acre	NaN	1/1/2005
206	2004	Novembro	Acre	NaN	1/1/2004
217	2015	Novembro	Acre	NaN	1/1/2015
444	2002	Novembro	alagoas	NaN	1/1/2002
522	2001	Março	Amapa	NaN	1/1/2001
550	2009	Abril	Amapa	NaN	1/1/2009
614	2013	Julho	Amapa	NaN	1/1/2013
642	2001	Setembro	Amapa	NaN	1/1/2001

Những gì làm được với data bị mất

Dùng lệnh fillna() để điền những chỗ trống

```
df['number_of_fires'].fillna(0).head()  
  
0    0  
1    0  
2    0  
3    0  
4    0  
Name: number_of_fires, dtype: object
```

Điền ngược

```
df['number_of_fires'].fillna(method='ffill').head(70)
0    0
1    0
2    0
3    0
4    0
..
65   1
66   2
67   1
68   1
69   0
Name: number_of_fires, Length: 70, dtype: object
```

Lọc hàng 444 và hàng 445 để check thông tin

```
df.iloc[444]
year           2002
month        Novembro
state       alagoas
number_of_fires      NaN
date        1/1/2002
Name: 444, dtype: object
```

```
df.iloc[445]
year           2003
month        Novembro
state       alagoas
number_of_fires      17
date        1/1/2003
Name: 445, dtype: object
```

Fill dữ liệu NaN của 444 thành số 0

```
df = df_copy
# replace all missing or NaN values with 0
df['number_of_fires'] = df['number_of_fires'].fillna(0)
```

```
# Let's check to see if we did change our Nans to 0s
df.iloc[444]
```

```
year           2002
month        Novembro
state       alagoas
number_of_fires      0
date        1/1/2002
Name: 444, dtype: object
```

Assigning data types to our columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6454 entries, 0 to 6453
Data columns (total 5 columns):
year           6454 non-null int64
month          6454 non-null object
state          6454 non-null object
number_of_fires 6454 non-null object
date           6454 non-null object
dtypes: int64(1), object(4)
memory usage: 252.2+ KB
```

```
df["number_of_fires"] = df["number_of_fires"].str.replace('', '0').astype(float)
```

```
df.head()
```

	year	month	state	number_of_fires	date
0	1998	Janeiro	Acre	0.0	1/1/1998
1	1999	Janeiro	Acre	0.0	1/1/1999
2	2000	Janeiro	Acre	0.0	1/1/2000
3	2001	Janeiro	Acre	0.0	1/1/2001
4	2002	Janeiro	Acre	0.0	1/1/2002

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6454 entries, 0 to 6453
Data columns (total 5 columns):
year           6454 non-null int64
month          6454 non-null object
state          6454 non-null object
number_of_fires 6322 non-null float64
date           6454 non-null object
dtypes: float64(1), int64(1), object(3)
memory usage: 252.2+ KB
```

```
df['month'].unique()
```

```
array(['Janeiro', 'Fevereiro', 'Março', 'Abril', 'Maio', 'Junho', 'Julho',
       'Agosto', 'Setembro', 'Outubro', 'Novembro', 'Dezembro'],
      dtype=object)
```

Thay thế ký tự trong cột

```

month_translations = {'Janeiro': 'January',
'Fevereiro': 'February',
'Março': 'March',
'Abril': 'April',
'Maio': 'May',
'Junho': 'June',
'Julho': 'July',
'Agosto': 'August',
'Setembro': 'September',
'Outubro': 'October',
'Novembro': 'November',
'Dezembro': 'December'}

df["month"] = df["month"].map(month_translations)
df.head()

```

	year	month	state	number_of_fires	date
0	1998	January	Acre	0.0	1/1/1998
1	1999	January	Acre	0.0	1/1/1999
2	2000	January	Acre	0.0	1/1/2000
3	2001	January	Acre	0.0	1/1/2001
4	2002	January	Acre	0.0	1/1/2002

```
df.isnull().sum()
```

```

year          0
month         0
state         0
number_of_fires    132
date          0
dtype: int64

```

Further string functions on columns

```

df['state'] = df['state'].str.title()
df['state'].unique()

array(['Acre', 'Alagoas', 'Amapa', 'Amazonas', 'Bahia', 'Ceara',
       'Distrito Federal', 'Espirito Santo', 'Goias', 'Maranhao',
       'Mato Grosso', 'Minas Gerais', 'Pará', 'Paraiba', 'Pernambuco',
       'Piau', 'Rio', 'Rondonia', 'Roraima', 'Santa Catarina',
       'Sao Paulo', 'Sergipe', 'Tocantins'], dtype=object)

```

Removing columns

```
df.head()
```

	year	month	state	number_of_fires	date
0	1998	January	Acre	0.0	1/1/1998
1	1999	January	Acre	0.0	1/1/1999
2	2000	January	Acre	0.0	1/1/2000
3	2001	January	Acre	0.0	1/1/2001
4	2002	January	Acre	0.0	1/1/2002

```
df = df.drop("date", axis=1) # axis = 1 so that it works across our columns  
df.head()
```

	year	month	state	number_of_fires
0	1998	January	Acre	0.0
1	1999	January	Acre	0.0
2	2000	January	Acre	0.0
3	2001	January	Acre	0.0
4	2002	January	Acre	0.0

```
file_name = "https://raw.githubusercontent.com/rajeevratan84/datascienceforbusiness/master/amazon_fires.csv"  
df = pd.read_csv(file_name, encoding = "ISO-8859-1")  
new_columns = {'ano' : 'year',  
              'estado': 'state',  
              'mes': 'month',  
              'numero': 'number_of_fires',  
              'encontro': 'date'}  
df.rename(columns = new_columns, inplace=True)  
df['number_of_fires'] = df['number_of_fires'].str.strip(" Fires")  
df_copy = df.copy()  
df.head()
```

	year	month	state	number_of_fires	date
0	1998	Janeiro	Acre	0	1/1/1998
1	1999	Janeiro	Acre	0	1/1/1999
2	2000	Janeiro	Acre	0	1/1/2000
3	2001	Janeiro	Acre	0	1/1/2001
4	2002	Janeiro	Acre	0	1/1/2002

```
# Drop multiple columns  
df = df.drop(["year", "date"], axis=1)  
df.head()
```

	month	state	number_of_fires
0	Janeiro	Acre	0
1	Janeiro	Acre	0
2	Janeiro	Acre	0
3	Janeiro	Acre	0
4	Janeiro	Acre	0

Dropping Rows

Drop hàng đầu tiên

```
df = df.drop(df.index[0])
df = df.reset_index()
df.head()
```

	index	month	state	number_of_fires
0	1	Janeiro	Acre	0
1	2	Janeiro	Acre	0
2	3	Janeiro	Acre	0
3	4	Janeiro	Acre	0
4	5	Janeiro	Acre	10

Drop nhiều hàng

```
df = df.drop(df.index[[2,3]])
df.head()
```

	index	month	state	number_of_fires
0	1	Janeiro	Acre	0
1	2	Janeiro	Acre	0
4	5	Janeiro	Acre	10
5	6	Janeiro	Acre	0
6	7	Janeiro	Acre	12

Drop 1 khoảng cột

```
df = df.drop(df.index[1:4])
df.head()
```

	index	month	state	number_of_fires
0	1	Janeiro	Acre	0
6	7	Janeiro	Acre	12
7	8	Janeiro	Acre	4
8	9	Janeiro	Acre	0
9	10	Janeiro	Acre	0

4. Pandas Data Aggregation

```
import pandas as pd

file_name = "https://raw.githubusercontent.com/rajeevratan84/datasciencforbusiness/master/amazon_fires.csv"
df = pd.read_csv(file_name, encoding = "ISO-8859-1")
new_columns = {'ano': 'year',
               'estado': 'state',
               'mes': 'month',
               'numero': 'number_of_fires',
               'encontro': 'date'}
df.rename(columns = new_columns, inplace=True)
df['state'] = df['state'].str.title()

df['number_of_fires'] = df['number_of_fires'].str.strip(" Fires")
df['number_of_fires'] = df['number_of_fires'].fillna(0)
df['number_of_fires'] = df['number_of_fires'].astype(float)

# Introducing the Map Function
month_translations = {'Janeiro': 'January',
                      'Fevereiro': 'February',
                      'Março': 'March',
                      'Abril': 'April',
                      'Maio': 'May',
                      'Junho': 'June',
                      'Julho': 'July',
                      'Agosto': 'August',
                      'Sétembro': 'September',
                      'Outubro': 'October',
                      'Novembro': 'November',
                      'Dezembro': 'December'}

df["month"] = df["month"].map(month_translations)
df.head(15)
```

	year	month	state	number_of_fires	date
0	1998	January	Acre	0.0	1/1/1998
1	1999	January	Acre	0.0	1/1/1999
2	2000	January	Acre	0.0	1/1/2000
3	2001	January	Acre	0.0	1/1/2001
4	2002	January	Acre	0.0	1/1/2002
5	2003	January	Acre	10.0	1/1/2003
6	2004	January	Acre	0.0	1/1/2004
7	2005	January	Acre	12.0	1/1/2005
8	2006	January	Acre	4.0	1/1/2006
9	2007	January	Acre	0.0	1/1/2007
10	2008	January	Acre	0.0	1/1/2008
11	2009	January	Acre	0.0	1/1/2009
12	2010	January	Acre	1.0	1/1/2010
13	2011	January	Acre	0.0	1/1/2011
14	2012	January	Acre	0.0	1/1/2012

Using the Groupby function

Đây là một trong những công cụ phân tích dữ liệu hữu ích nhất do Pandas cung cấp. Nó cho phép chúng tôi phân đoạn dữ liệu chung và thực hiện một thao tác trên.

```
df.groupby('state')  
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7fe079853940>  
  
state_groups = df.groupby('state')  
state_groups.get_group('Rio')
```

	year	month	state	number_of_fires		date
4303	1998	January	Rio	0.0	1/1/1998	
4304	1999	January	Rio	0.0	1/1/1999	
4305	2000	January	Rio	0.0	1/1/2000	
4306	2001	January	Rio	0.0	1/1/2001	
4307	2002	January	Rio	0.0	1/1/2002	
...
5015	2012	December	Rio	38.0	1/1/2012	
5016	2013	December	Rio	62.0	1/1/2013	
5017	2014	December	Rio	31.0	1/1/2014	
5018	2015	December	Rio	42.0	1/1/2015	
5019	2016	December	Rio	79.0	1/1/2016	

717 rows × 5 columns

```

state_groups.groups

{'Acre': Int64Index([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9,
    ...
   229, 230, 231, 232, 233, 234, 235, 236, 237, 238],
  dtype='int64', length=239),
 'Alagoas': Int64Index([239, 240, 241, 242, 243, 244, 245, 246, 247, 248,
    ...
   469, 470, 471, 472, 473, 474, 475, 476, 477, 478],
  dtype='int64', length=240),
 'Amapa': Int64Index([479, 480, 481, 482, 483, 484, 485, 486, 487, 488,
    ...
   708, 709, 710, 711, 712, 713, 714, 715, 716, 717],
  dtype='int64', length=239),
 'Amazonas': Int64Index([718, 719, 720, 721, 722, 723, 724, 725, 726, 727,
    ...
   947, 948, 949, 950, 951, 952, 953, 954, 955, 956],
  dtype='int64', length=239),
 'Bahia': Int64Index([ 957,  958,  959,  960,  961,  962,  963,  964,  965,  966,
    ...
   1186, 1187, 1188, 1189, 1190, 1191, 1192, 1193, 1194, 1195],
  dtype='int64', length=239),
 'Ceara': Int64Index([1196, 1197, 1198, 1199, 1200, 1201, 1202, 1203, 1204, 1205,
    ...
   1425, 1426, 1427, 1428, 1429, 1430, 1431, 1432, 1433, 1434],
  dtype='int64', length=239),
 'Distrito Federal': Int64Index([1435, 1436, 1437, 1438, 1439, 1440, 1441, 1442, 1443, 1444,
    ...
   1664, 1665, 1666, 1667, 1668, 1669, 1670, 1671, 1672, 1673],
  dtype='int64', length=239),
 'Espírito Santo': Int64Index([1674, 1675, 1676, 1677, 1678, 1679, 1680, 1681, 1682, 1683,
    ...
   1903, 1904, 1905, 1906, 1907, 1908, 1909, 1910, 1911, 1912],
  dtype='int64', length=239),
 'Goiás': Int64Index([1913, 1914, 1915, 1916, 1917, 1918, 1919, 1920, 1921, 1922,
    ...
   2142, 2143, 2144, 2145, 2146, 2147, 2148, 2149, 2150, 2151],
  dtype='int64', length=239),
 'Maranhão': Int64Index([2152, 2153, 2154, 2155, 2156, 2157, 2158, 2159, 2160, 2161,
    ...
   2381, 2382, 2383, 2384, 2385, 2386, 2387, 2388, 2389, 2390],
  dtype='int64', length=239),
 'Mato Grosso': Int64Index([2391, 2392, 2393, 2394, 2395, 2396, 2397, 2398, 2399, 2400,
    ...
   2859, 2860, 2861, 2862, 2863, 2864, 2865, 2866, 2867, 2868],
  dtype='int64', length=478),
 'Minas Gerais': Int64Index([2869, 2870, 2871, 2872, 2873, 2874, 2875, 2876, 2877, 2878,
    ...
])

```

Dùng lệnh size() để đếm số lượng

```
state_groups.size()

state
Acre           239
Alagoas        240
Amapa          239
Amazonas       239
Bahia          239
Ceara          239
Distrito Federal  239
Espirito Santo   239
Goias          239
Maranhao        239
Mato Grosso     478
Minas Gerais    239
Paraiba         478
Pará            239
Pernambuco      239
Piau            239
Rio             717
Rondonia        239
Roraima         239
Santa Catarina   239
Sao Paulo       239
Sergipe         239
Tocantins       239
dtype: int64
```

Một số hàm như:

- mean()
- size()
- count()
- sum()
- min()
- max()

```
state_groups.sum()
```

state	year	number_of_fires
Acre	479783	17971.030
Alagoas	481800	4594.000
Amapa	479783	20108.576
Amazonas	479783	29890.129
Bahia	479783	43411.951
Ceara	479783	30395.042
Distrito Federal	479783	3501.000
Espirito Santo	479783	37002.276
Goias	479783	23362.852
Maranhao	479783	24839.169
Mato Grosso	959566	95566.975
Minas Gerais	479783	37002.276
Paraiba	959566	51723.918
Pará	479783	22124.144
Pernambuco	479783	22959.000
Piau	479783	37542.651
Rio	1439349	44799.865
Rondonia	479783	19239.429
Roraima	479783	23749.074
Santa Catarina	479783	23362.852
Sao Paulo	479783	49888.198
Sergipe	479783	3167.000
Tocantins	479783	33441.509

```

state_groups.mean()

# Note the top left cell named 'state' is now the index
# Previously the index was numbered numerically
# The columns year and number_of_fires are the grouped means

```

	year	number_of_fires
state		
Acre	2007.460251	75.192594
Alagoas	2007.500000	19.141667
Amapa	2007.460251	84.136301
Amazonas	2007.460251	125.063301
Bahia	2007.460251	181.639962
Ceara	2007.460251	127.175908
Distrito Federal	2007.460251	14.648536
Espirito Santo	2007.460251	154.821238
Goias	2007.460251	97.752519
Maranhao	2007.460251	103.929577
Mato Grosso	2007.460251	199.930910
Minas Gerais	2007.460251	154.821238
Paraiba	2007.460251	108.209033
Pará	2007.460251	92.569640
Pernambuco	2007.460251	96.062762
Piau	2007.460251	157.082222
Rio	2007.460251	62.482378
Rondonia	2007.460251	80.499703
Roraima	2007.460251	99.368510
Santa Catarina	2007.460251	97.752519
Sao Paulo	2007.460251	208.737230
Sergipe	2007.460251	13.251046
Tocantins	2007.460251	139.922632

Combining our groupby into one concise statement

```
df.groupby('state')['number_of_fires'].mean()

# In the above code:
# 'state' is what we're grouping
# 'number_of_fires' is column who's values we're running our aggregate mean function on
# our index will be 'state' after running this
```

```
state
Acre           75.192594
Alagoas        19.141667
Amapa          84.136301
Amazonas       125.063301
Bahia          181.639962
Ceara          127.175908
Distrito Federal 14.648536
Espirito Santo   154.821238
Goias           97.752519
Maranhao        103.929577
Mato Grosso     199.930910
Minas Gerais    154.821238
Paraiba         108.209033
Pará             92.569640
Pernambuco      96.062762
Piau             157.082222
Rio              62.482378
Rondonia        80.499703
Roraima          99.368510
Santa Catarina   97.752519
Sao Paulo        208.737230
Sergipe          13.251046
Tocantins        139.922632
Name: number_of_fires, dtype: float64
```

Pivot Table

```
import numpy as np  
  
df.pivot_table(values='number_of_fires', index='state', aggfunc=np.mean)
```

state	number_of_fires
Acre	75.192594
Alagoas	19.141667
Amapa	84.136301
Amazonas	125.063301
Bahia	181.639962
Ceara	127.175908
Distrito Federal	14.648536
Espirito Santo	154.821238
Goias	97.752519
Maranhao	103.929577
Mato Grosso	199.930910
Minas Gerais	154.821238
Paraiba	108.209033
Pará	92.569640
Pernambuco	96.062762
Piau	157.082222
Rio	62.482378
Rondonia	80.499703
Roraima	99.368510
Santa Catarina	97.752519
Sao Paulo	208.737230
Sergipe	13.251046
Tocantins	139.922632

```
df.pivot_table(values='number_of_fires', index='state', aggfunc=np.mean, margins=True)
```

state	number_of_fires
Acre	75.192594
Alagoas	19.141667
Amapa	84.136301
Amazonas	125.063301
Bahia	181.639962
Ceara	127.175908
Distrito Federal	14.648536
Espirito Santo	154.821238
Goias	97.752519
Maranhao	103.929577
Mato Grosso	199.930910
Minas Gerais	154.821238
Paraiba	108.209033
Pará	92.569640
Pernambuco	96.062762
Piau	157.082222
Rio	62.482378
Rondonia	80.499703
Roraima	99.368510
Santa Catarina	97.752519
Sao Paulo	208.737230
Sergipe	13.251046
Tocantins	139.922632
All	108.404542

```

# Can be useful when creating a plot

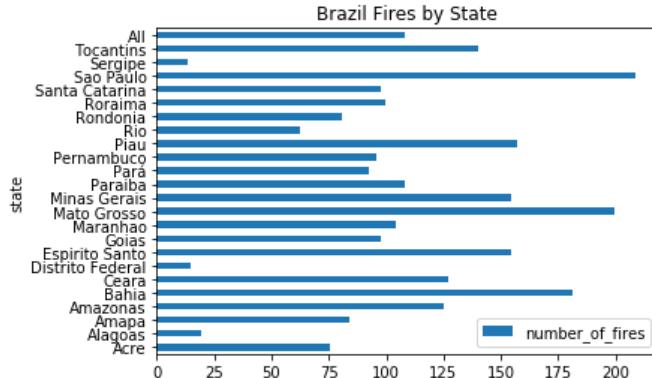
fires_per_state = df.pivot_table(values='number_of_fires', index='state', aggfunc=np.mean, margins=True)

# We push the horizontal scale to the right just to make it more aesthetic
max_val = fires_per_state["number_of_fires"].max() + 10

fires_per_state.plot(kind='barh', title='Brazil Fires by State', xlim=(0,max_val), legend=True)

```

<matplotlib.axes._subplots.AxesSubplot at 0x7fe0794e80b8>



Aggregating Multiple Columns Using Pivot Tables

df.shape[0]

6454

```

import numpy as np

df['severity'] = np.random.randint(1, 5, df.shape[0])
df.head(10)

```

	year	month	state	number_of_fires	date	severity
0	1998	January	Acre	0.0	1/1/1998	3
1	1999	January	Acre	0.0	1/1/1999	1
2	2000	January	Acre	0.0	1/1/2000	3
3	2001	January	Acre	0.0	1/1/2001	2
4	2002	January	Acre	0.0	1/1/2002	1
5	2003	January	Acre	10.0	1/1/2003	4
6	2004	January	Acre	0.0	1/1/2004	3
7	2005	January	Acre	12.0	1/1/2005	4
8	2006	January	Acre	4.0	1/1/2006	1
9	2007	January	Acre	0.0	1/1/2007	4

Format for aggregating multiple columns of data

```
DataFrame.pivot_table([column1_values, column2_values], column_index)
```

```
df.pivot_table(['number_of_fires', 'severity'], 'state')
```

state	number_of_fires	severity
Acre	75.192594	2.531381
Alagoas	19.141667	2.404167
Amapa	84.136301	2.468619
Amazonas	125.063301	2.422594
Bahia	181.639962	2.456067
Ceara	127.175908	2.502092
Distrito Federal	14.648536	2.426778
Espirito Santo	154.821238	2.393305
Goias	97.752519	2.548117
Maranhao	103.929577	2.401674
Mato Grosso	199.930910	2.577406
Minas Gerais	154.821238	2.497908
Paraiba	108.209033	2.468619
Pará	92.569640	2.456067
Pernambuco	96.062762	2.489540
Piau	157.082222	2.514644
Rio	62.482378	2.483961
Rondonia	80.499703	2.502092
Roraima	99.368510	2.543933
Santa Catarina	97.752519	2.410042
Sao Paulo	208.737230	2.476987
Sergipe	13.251046	2.468619
Tocantins	139.922632	2.594142

```
df.pivot_table(values=['number_of_fires', 'severity'], index='state', aggfunc=np.mean, margins=True)
```

state	number_of_fires	severity
Acre	75.192594	2.531381
Alagoas	19.141667	2.404167
Amapá	84.136301	2.468619
Amazonas	125.063301	2.422594
Bahia	181.639962	2.456067
Ceará	127.175908	2.502092
Distrito Federal	14.648536	2.426778
Espirito Santo	154.821238	2.393305
Goiás	97.752519	2.548117
Maranhão	103.929577	2.401674
Mato Grosso	199.930910	2.577406
Minas Gerais	154.821238	2.497908
Paraíba	108.209033	2.468619
Pará	92.569640	2.456067
Pernambuco	96.062762	2.489540
Piauí	157.082222	2.514644
Rio	62.482378	2.483961
Rondonia	80.499703	2.502092
Roraima	99.368510	2.543933
Santa Catarina	97.752519	2.410042
São Paulo	208.737230	2.476987
Sergipe	13.251046	2.468619
Tocantins	139.922632	2.594142
All	108.404542	2.483421

state	mean		amin		amax	
	number_of_fires	severity	number_of_fires	severity	number_of_fires	severity
Acre	75.192594	2.531381	0.0	1	960.0	4
Alagoas	19.141667	2.404167	0.0	1	162.0	4
Amapa	84.136301	2.468619	0.0	1	969.0	4
Amazonas	125.063301	2.422594	0.0	1	998.0	4
Bahia	181.639962	2.456067	0.0	1	995.0	4
Ceara	127.175908	2.502092	0.0	1	995.0	4
Distrito Federal	14.648536	2.426778	0.0	1	196.0	4
Espirito Santo	154.821238	2.393305	0.0	1	959.0	4
Goias	97.752519	2.548117	0.0	1	759.0	4
Maranhao	103.929577	2.401674	0.0	1	972.0	4
Mato Grosso	199.930910	2.577406	0.0	1	979.0	4
Minas Gerais	154.821238	2.497908	0.0	1	959.0	4
Paraiba	108.209033	2.468619	0.0	1	987.0	4
Pará	92.569640	2.456067	0.0	1	971.0	4
Pernambuco	96.062762	2.489540	0.0	1	859.0	4
Piau	157.082222	2.514644	0.0	1	943.0	4
Rio	62.482378	2.483961	0.0	1	885.0	4
Rondonia	80.499703	2.502092	0.0	1	969.0	4
Roraima	99.368510	2.543933	0.0	1	820.0	4
Santa Catarina	97.752519	2.410042	0.0	1	759.0	4
Sao Paulo	208.737230	2.476987	0.0	1	981.0	4
Sergipe	13.251046	2.468619	0.0	1	198.0	4
Tocantins	139.922632	2.594142	0.0	1	989.0	4
All	108.404542	2.483421	0.0	1	998.0	4

Learning to use GroupBy Agg

DataFrame.groupby(index)[column1].agg(function)

```
df.groupby('state')['number_of_fires'].agg(np.mean)
```

state	
Acre	75.192594
Alagoas	19.141667
Amapá	84.136301
Amazonas	125.063301
Bahia	181.639962
Ceará	127.175908
Distrito Federal	14.648536
Espírito Santo	154.821238
Goiás	97.752519
Maranhão	103.929577
Mato Grosso	199.930910
Minas Gerais	154.821238
Paraíba	108.209033
Pará	92.569640
Pernambuco	96.062762
Piauí	157.082222
Rio	62.482378
Rondônia	80.499703
Roraima	99.368510
Santa Catarina	97.752519
São Paulo	208.737230
Sergipe	13.251046
Tocantins	139.922632

Name: number_of_fires, dtype: float64

```
df.groupby('state')[['number_of_fires', 'severity']].agg(np.mean)
```

state	number_of_fires	severity
Acre	75.192594	2.560669
Alagoas	19.141667	2.412500
Amapa	84.136301	2.548117
Amazonas	125.063301	2.543933
Bahia	181.639962	2.585774
Ceara	127.175908	2.485356
Distrito Federal	14.648536	2.577406
Espirito Santo	154.821238	2.481172
Goias	97.752519	2.439331
Maranhao	103.929577	2.569038
Mato Grosso	199.930910	2.464435
Minas Gerais	154.821238	2.489540
Paraiba	108.209033	2.560669
Pará	92.569640	2.569038
Pernambuco	96.062762	2.460251
Piau	157.082222	2.518828
Rio	62.482378	2.567643
Rondonia	80.499703	2.552301
Roraima	99.368510	2.481172
Santa Catarina	97.752519	2.556485
Sao Paulo	208.737230	2.518828
Sergipe	13.251046	2.539749
Tocantins	139.922632	2.518828

```
df.groupby('state')[['number_of_fires', 'severity']].agg([np.mean, np.max])
```

state	number_of_fires		severity		D
	mean	amax	mean	amax	
Acre	75.192594	960.0	2.560669	4	
Alagoas	19.141667	162.0	2.412500	4	
Amapa	84.136301	969.0	2.548117	4	
Amazonas	125.063301	998.0	2.543933	4	
Bahia	181.639962	995.0	2.585774	4	
Ceara	127.175908	995.0	2.485356	4	
Distrito Federal	14.648536	196.0	2.577406	4	
Espirito Santo	154.821238	959.0	2.481172	4	
Goias	97.752519	759.0	2.439331	4	
Maranhao	103.929577	972.0	2.569038	4	
Mato Grosso	199.930910	979.0	2.464435	4	
Minas Gerais	154.821238	959.0	2.489540	4	
Paraiba	108.209033	987.0	2.560669	4	
Pará	92.569640	971.0	2.569038	4	
Pernambuco	96.062762	859.0	2.460251	4	
Piau	157.082222	943.0	2.518828	4	
Rio	62.482378	885.0	2.567643	4	
Rondonia	80.499703	969.0	2.552301	4	
Roraima	99.368510	820.0	2.481172	4	
Santa Catarina	97.752519	759.0	2.556485	4	
Sao Paulo	208.737230	981.0	2.518828	4	
Sergipe	13.251046	198.0	2.539749	4	
Tocantins	139.922632	989.0	2.518828	4	

Using custom Agg functions

```
def max_minus_mean(grouped_data):
    return(grouped_data.max() - grouped_data.mean())

df.groupby('state')['number_of_fires'].agg(max_minus_mean)
```

state	
Acre	884.807406
Alagoas	142.858333
Amapa	884.863699
Amazonas	872.936699
Bahia	813.360038
Ceara	867.824092
Distrito Federal	181.351464
Espirito Santo	804.178762
Goias	661.247481
Maranhao	868.070423
Mato Grosso	779.069090
Minas Gerais	804.178762
Paraiba	878.790967
Pará	878.430360
Pernambuco	762.937238
Piau	785.917778
Rio	822.517622
Rondonia	888.500297
Roraima	720.631490
Santa Catarina	661.247481
Sao Paulo	772.262770
Sergipe	184.748954
Tocantins	849.077368

D

Name: number_of_fires, dtype: float64

```
pd.DataFrame(df.groupby('state')['number_of_fires'].agg(max_minus_mean)).reset_index()
```

	state	number_of_fires
0	Acre	884.807406
1	Alagoas	142.858333
2	Amapa	884.863699
3	Amazonas	872.936699
4	Bahia	813.360038
5	Ceara	867.824092
6	Distrito Federal	181.351464
7	Espirito Santo	804.178762
8	Goiias	661.247481
9	Maranhao	868.070423
10	Mato Grosso	779.069090
11	Minas Gerais	804.178762
12	Paraiba	878.790967
13	Pará	878.430360
14	Pernambuco	762.937238
15	Piau	785.917778
16	Rio	822.517622
17	Rondonia	888.500297
18	Roraima	720.631490
19	Santa Catarina	661.247481
20	Sao Paulo	772.262770
21	Sergipe	184.748954
22	Tocantins	849.077368

D

5. Pandas Feature Engineering

```
import pandas as pd

file_name = "https://raw.githubusercontent.com/rajeevratan84/datascienceforbusiness/master/titanic.csv"
df = pd.read_csv(file_name)

df.head()
```

pclass		name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	survived
0	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	1
1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	1
2	1	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	0
3	1	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	0
4	1	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	113781	151.5500	C22 C26	S	0

- sibsp: # of siblings/spouse aboard
- parch: number of parents/child aboard

```
def get_family_size(sibsp, parch):
    family_size = sibsp + parch
    return family_size

df["family_size"] = df[["sibsp", "parch"]].apply(lambda x: get_family_size(x["sibsp"], x["parch"]), axis=1)
df.head()
```

pclass		name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	survived	family_size
0	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	1	0
1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	1	3
2	1	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	0	3
3	1	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	0	3
4	1	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	113781	151.5500	C22 C26	S	0	3

How do we use apply and lambda?

```
df['new_col'] = df.apply(lambda x: function(x.col_1, x.col_2), axis=1)
```

```
df["family_size2"] = df["sibsp"] + df["parch"]
df.head()
```

pclass		name	sex	age	sibsp	parch	ticket	fare	cabin	embarked	survived	family_size	family_size2
0	1	Allen, Miss. Elisabeth Walton	female	29.0000	0	0	24160	211.3375	B5	S	1	0	0
1	1	Allison, Master. Hudson Trevor	male	0.9167	1	2	113781	151.5500	C22 C26	S	1	3	3
2	1	Allison, Miss. Helen Loraine	female	2.0000	1	2	113781	151.5500	C22 C26	S	0	3	3
3	1	Allison, Mr. Hudson Joshua Creighton	male	30.0000	1	2	113781	151.5500	C22 C26	S	0	3	3
4	1	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.0000	1	2	113781	151.5500	C22 C26	S	0	3	3

6. Concatenating, Merging and Joining DataFrames in Pandas

1. Concat
2. Append
3. Merge
4. Join

```
import pandas as pd

df1 = pd.DataFrame({'item': ['A', 'B', 'C', 'D'],
                    'value': [1, 2, 3, 5]})

df2 = pd.DataFrame({'item': ['E', 'F', 'G', 'H'],
                    'value': [5, 6, 7, 8]})

print(df1)
print(df2)
```

	item	value
0	A	1
1	B	2
2	C	3
3	D	5

	item	value
0	E	5
1	F	6
2	G	7
3	H	8

df1

	item	value
0	A	1
1	B	2
2	C	3
3	D	5

```
pd.concat([df1, df2])
```

	item	value
0	A	1
1	B	2
2	C	3
3	D	5
0	E	5
1	F	6
2	G	7
3	H	8

Similarly we can use Append to add multiple dataFrames onto another

```

df1 = pd.DataFrame({'item': ['A', 'B', 'C', 'D'],
                    'value': [1, 2, 3, 5]})

df2 = pd.DataFrame({'item': ['D', 'F', 'G', 'H'],
                    'quanity': [2, 2, 1, 5]})

df3 = pd.DataFrame({'item': ['I', 'J', 'K', 'L'],
                    'quanity': [3, 4, 7, 25]})

df1.append([df2, df3])

```

```
C:\Users\minhk\AppData\Local\Temp\ipykernel_544\140
hod is deprecated and will be removed from pandas :
df1.append([df2, df3])
```

	item	value	quanity
0	A	1.0	NaN
1	B	2.0	NaN
2	C	3.0	NaN
3	D	5.0	NaN
0	D	NaN	2.0
1	F	NaN	2.0
2	G	NaN	1.0
3	H	NaN	5.0
0	I	NaN	3.0
1	J	NaN	4.0
2	K	NaN	7.0
3	L	NaN	25.0

Pandas merge

one-to-one joins: for example when joining two DataFrame objects on their indexes (which must contain unique values).

many-to-one joins: for example when joining an index (unique) to one or more columns in a different DataFrame.

many-to-many joins: joining columns on columns.

```
dfa = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                    'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3']})
```

```
dfB = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                    'C': ['C0', 'C1', 'C2', 'C3'],
                    'D': ['D0', 'D1', 'D2', 'D3']})
```

```
dfa
```

	key	A	B
0	K0	A0	B0
1	K1	A1	B1
2	K2	A2	B2
3	K3	A3	B3

```
dfB
```

	key	C	D
0	K0	C0	D0
1	K1	C1	D1
2	K2	C2	D2
3	K3	C3	D3

```
pd.merge(dfa, dfB, on='key')
```

	key	A	B	C	D
0	K0	A0	B0	C0	D0
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3

```
dfa = pd.DataFrame({'key': ['K0', 'K1', 'K2', 'K3'],
                    'A': ['A0', 'A1', 'A2', 'A3'],
                    'B': ['B0', 'B1', 'B2', 'B3']})
```

```
dfB = pd.DataFrame({'key': ['K4', 'K1', 'K2', 'K3'],
                    'C': ['C0', 'C1', 'C2', 'C3'],
                    'D': ['D0', 'D1', 'D2', 'D3']})
```

```
pd.merge(dfa, dfB, how='left', on='key')
```

	key	A	B	C	D
0	K0	A0	B0	NaN	NaN
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3

```
# Inner Joins
```

```
pd.merge(dFA, dfB, how='inner', on='key')
```

	key	A	B	C	D
0	K1	A1	B1	C1	D1
1	K2	A2	B2	C2	D2
2	K3	A3	B3	C3	D3

```
# Outer Joins
```

```
pd.merge(dFA, dfB, how='outer', on='key')
```

	key	A	B	C	D
0	K0	A0	B0	NaN	NaN
1	K1	A1	B1	C1	D1
2	K2	A2	B2	C2	D2
3	K3	A3	B3	C3	D3
4	K4	NaN	NaN	C0	D0

Joining

```
left = pd.DataFrame({'A': ['A0', 'A1', 'A2'],
                     'B': ['B0', 'B1', 'B2']},
                     index=['K0', 'K1', 'K2'])
```

```
left
```

	A	B
K0	A0	B0
K1	A1	B1
K2	A2	B2

```
right = pd.DataFrame({'C': ['C0', 'C2', 'C3'],
                      'D': ['D0', 'D2', 'D3']},
                      index=['K0', 'K2', 'K3'])
```

```
right
```

	C	D
K0	C0	D0
K2	C2	D2
K3	C3	D3

```
left.join(right)
```

	A	B	C	D
K0	A0	B0	C0	D0
K1	A1	B1	NaN	NaN
K2	A2	B2	C2	D2

```
left.join(right, how='outer')
```

	A	B	C	D
K0	A0	B0	C0	D0
K1	A1	B1	NaN	NaN
K2	A2	B2	C2	D2
K3	NaN	NaN	C3	D3

7. Basic Time Series Analysis

```
import pandas as pd
from datetime import datetime
import numpy as np

# Let's create a pandas series that logs time every hour from 1st Nov'19 to 7th Nov'19
df = pd.date_range(start='11/01/2019', end='11/07/2019', freq='H')
df

DatetimeIndex(['2019-11-01 00:00:00', '2019-11-01 01:00:00',
               '2019-11-01 02:00:00', '2019-11-01 03:00:00',
               '2019-11-01 04:00:00', '2019-11-01 05:00:00',
               '2019-11-01 06:00:00', '2019-11-01 07:00:00',
               '2019-11-01 08:00:00', '2019-11-01 09:00:00',
               ...
               '2019-11-06 15:00:00', '2019-11-06 16:00:00',
               '2019-11-06 17:00:00', '2019-11-06 18:00:00',
               '2019-11-06 19:00:00', '2019-11-06 20:00:00',
               '2019-11-06 21:00:00', '2019-11-06 22:00:00',
               '2019-11-06 23:00:00', '2019-11-07 00:00:00'],
              dtype='datetime64[ns]', length=145, freq='H')
```

- yyyy-mm-dd hh:mm:ss

```
len(df)
```

```
145
```

```
#Now let's turn our series into a dataframe
df = pd.DataFrame(df, columns=['date'])

# And add a 'made up' column for sales data
df['sales'] = np.random.randint(0,1000,size=(len(df)))
df.head()
```

	date	sales
0	2019-11-01 00:00:00	591
1	2019-11-01 01:00:00	43
2	2019-11-01 02:00:00	941
3	2019-11-01 03:00:00	850
4	2019-11-01 04:00:00	540

Selecting using dates

```
# Set your date as the index  
df = df.set_index('date')  
df.head()
```

```
sales  
date  
2019-11-01 00:00:00    591  
2019-11-01 01:00:00     43  
2019-11-01 02:00:00   941  
2019-11-01 03:00:00   850  
2019-11-01 04:00:00   540
```

Using .loc[] to index specific dates

```
# Selecting using date - getting exact value for cell  
df.loc['2019-11-01 03:00:00', 'sales']
```

```
850
```

```
# Selecting using date to return the row corresponding to that date  
df.loc['2019-11-01 03:00:00']
```

```
sales    850  
Name: 2019-11-01 03:00:00, dtype: int64
```

```
# Selecting an entire day  
df.loc['2019-11-01']  
  
# Similary you can use df.loc['2019-11'] to select an entire month
```

sales	
date	
2019-11-01 00:00:00	591
2019-11-01 01:00:00	43
2019-11-01 02:00:00	941
2019-11-01 03:00:00	850
2019-11-01 04:00:00	540
2019-11-01 05:00:00	75
2019-11-01 06:00:00	849
2019-11-01 07:00:00	479
2019-11-01 08:00:00	826
2019-11-01 09:00:00	342
2019-11-01 10:00:00	302
2019-11-01 11:00:00	47
2019-11-01 12:00:00	11
2019-11-01 13:00:00	613
2019-11-01 14:00:00	485
2019-11-01 15:00:00	69
2019-11-01 16:00:00	710
2019-11-01 17:00:00	304
2019-11-01 18:00:00	406
2019-11-01 19:00:00	441
2019-11-01 20:00:00	642
2019-11-01 21:00:00	283
2019-11-01 22:00:00	589
2019-11-01 23:00:00	523

```
# Selecting a range of dates  
df.loc['2019-11-01':'2019-11-02']
```

sales	
date	
2019-11-01 00:00:00	591
2019-11-01 01:00:00	43
2019-11-01 02:00:00	941
2019-11-01 03:00:00	850
2019-11-01 04:00:00	540
2019-11-01 05:00:00	75
2019-11-01 06:00:00	849
2019-11-01 07:00:00	479
2019-11-01 08:00:00	826
2019-11-01 09:00:00	342
2019-11-01 10:00:00	302
2019-11-01 11:00:00	47
2019-11-01 12:00:00	11
2019-11-01 13:00:00	613
2019-11-01 14:00:00	485
2019-11-01 15:00:00	69
2019-11-01 16:00:00	710
2019-11-01 17:00:00	304
2019-11-01 18:00:00	406
2019-11-01 19:00:00	441
2019-11-01 20:00:00	642
2019-11-01 21:00:00	283
2019-11-01 22:00:00	589
2019-11-01 23:00:00	523

Resampling

Summary States - we can use Statistical methods over different time intervals

mean(), sum(), count(), min(), max()

Down-sampling

reduce datetime rows to longer frequency

Up-sampling

increase datetime rows to shorter frequency

```
# Using resample to get the average for each day per hour  
df.resample('D').mean()  
#df.resample('D').sum()
```

sales	
	date
2019-11-01	10961
2019-11-02	11824
2019-11-03	11379
2019-11-04	10978
2019-11-05	13363
2019-11-06	9479
2019-11-07	256

```
# Using resample to get the average for each minute (downsampling)
```

```
df.resample('T').mean()
```

```
sales  
date  
----  
2019-11-01 00:00:00    591.0  
2019-11-01 00:01:00    NaN  
2019-11-01 00:02:00    NaN  
2019-11-01 00:03:00    NaN  
2019-11-01 00:04:00    NaN  
...      ...  
2019-11-06 23:56:00    NaN  
2019-11-06 23:57:00    NaN  
2019-11-06 23:58:00    NaN  
2019-11-06 23:59:00    NaN  
2019-11-07 00:00:00    256.0
```

8641 rows × 1 columns

Resampling frequencies

- 'min', 'T' - minute
- 'H' - hour
- 'D' - day
- 'B' - business day
- 'W' - week
- 'M' - month
- 'Q' - quarter
- 'A' - year

Parsing dates

```
df = pd.DataFrame({'year': [2015, 2016],
                   'month': [2, 3],
                   'day': [4, 5]})
```

```
df
```

	year	month	day
0	2015	2	4
1	2016	3	5

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2 entries, 0 to 1
Data columns (total 3 columns):
year      2 non-null int64
month     2 non-null int64
day       2 non-null int64
dtypes: int64(3)
memory usage: 176.0 bytes
```

```
pd.to_datetime(df)
```

```
0    2015-02-04
1    2016-03-05
dtype: datetime64[ns]
```

```
pd.to_datetime('2019-01-01', format='%Y-%m-%d', errors='ignore')
```

```
Timestamp('2019-01-01 00:00:00')
```

8. Advanced – iterrows, vectorization and numpy

```
import numpy as np

# Define a basic Haversine distance formula
def haversine(lat1, lon1, lat2, lon2):
    MILES = 3959
    lat1, lon1, lat2, lon2 = map(np.deg2rad, [lat1, lon1, lat2, lon2])
    dlat = lat2 - lat1
    dlon = lon2 - lon1
    a = np.sin(dlat/2)**2 + np.cos(lat1) * np.cos(lat2) * np.sin(dlon/2)**2
    c = 2 * np.arcsin(np.sqrt(a))
    total_miles = MILES * c
    return total_miles
```

```
# Define a function to manually loop over all rows and return a series of distances
def haversine_looping(df):
    distance_list = []
    for i in range(0, len(df)):
        d = haversine(40.671, -73.985, df.iloc[i]['latitude'], df.iloc[i]['longitude'])
        distance_list.append(d)
    return distance_list
```

```
import pandas as pd

file_name = "https://raw.githubusercontent.com/rajeevratan84/datascienceforbusiness/master/new_york_hotels.csv"
df = pd.read_csv(file_name, encoding = "ISO-8859-1")
df.head()
```

	ean_hotel_id	name	address1	city	state_province	postal_code	latitude	longitude	star_rating	high_rate	low_rate
0	269955	Hilton Garden Inn Albany/SUNY Area	1389 Washington Ave	Albany	NY	12206	42.68751	-73.81643	3.0	154.0272	124.0216
1	113431	Courtyard by Marriott Albany Thruway	1455 Washington Avenue	Albany	NY	12206	42.68971	-73.82021	3.0	179.0100	134.0000
2	108151	Radisson Hotel Albany	205 Wolf Rd	Albany	NY	12205	42.72410	-73.79822	3.0	134.1700	84.1600
3	254756	Hilton Garden Inn Albany Medical Center	62 New Scotland Ave	Albany	NY	12208	42.65157	-73.77638	3.0	308.2807	228.4597
4	198232	CrestHill Suites SUNY University Albany	1415 Washington Avenue	Albany	NY	12206	42.68873	-73.81854	3.0	169.3900	89.3900

```
df.shape
```

```
(1631, 11)
```

```
# This is a built in magic ipython command that provides timing for code executed in the cell
%%timeit

# Run the haversine Looping function
df['distance'] = haversine_looping(df)
```

```
1 loop, best of 3: 627 ms per loop
```

Hạn chế dùng vòng lặp trong pandas

Looping with iterrows()

```
%%timeit

# Haversine applied on rows via iteration
haversine_series = []
for index, row in df.iterrows():
    haversine_series.append(haversine(40.671, -73.985, row['latitude'], row['longitude']))
df['distance'] = haversine_series
```

```
1 loop, best of 3: 219 ms per loop
```

```
%%timeit

# Timing apply on the Haversine function
df['distance'] = df.apply(lambda row: haversine(40.671, -73.985, row['latitude'], row['longitude']), axis=1)
```

```
10 loops, best of 3: 60.3 ms per loop
```

```

%%timeit

# Vectorized implementation of Haversine applied on Pandas series
df['distance'] = haversine(40.671, -73.985, df['latitude'], df['longitude'])

```

The slowest run took 14.31 times longer than the fastest. This could mean that an intermediate result is being cached.
100 loops, best of 3: 2.52 ms per loop

```

# Run our Line profiler to inspect further
%lprun -f haversine df['distance'] = haversine(40.671, -73.985, df['latitude'], df['longitude'])

```

```

%%timeit

# Vectorized implementation of Haversine applied on NumPy arrays (note we use .values to access the num
df['distance'] = haversine(40.671, -73.985, df['latitude'].values, df['longitude'].values)

```

The slowest run took 8.09 times longer than the fastest. This could mean that an intermediate result is being cached.
1000 loops, best of 3: 274 µs per loop

9. Advanced – map, zip and apply

```

numbers = [5, 2, 7, 3, 4, 6, 8, 1]

# Let's make a function that calculates squares
def square_number(x):
    return x**2

```

```

# A non-pythonic approach is using a for Loop to iterate over each element in our list and append the r
squares = []
for num in numbers:
    squares.append(square_number(num))
print('Non-Pythonic Approach: ', squares)

```

Non-Pythonic Approach: [25, 4, 49, 9, 16, 36, 64, 1]

```

# A pythonic approach would be to use Map. It takes the function and the list as its input arguments.
x = map(square_number, numbers)
print('Pythonic Approach: ', list(x))

```

Pythonic Approach: [25, 4, 49, 9, 16, 36, 64, 1]

Let's now explore zip()

```

a = [5, 2, 7, 3, 4]
b = [3, 8, 6, 8, 1]

# Iterate over two or more lists simultaneously
for x, y in zip(a, b):
    print(x,y)

```

5 3
2 8
7 6
3 8
4 1

Now let's look at filter().

```
numbers = [5, 2, 7, 3, 4, 6, 8, 1, 9, 0, 10]

# Will return true if input number is even
def isEven(x):
    return x % 2 == 0

# A non-pythonic approach would be to use a for loop with an if statement inside
even_numbers = []
for num in numbers:
    if isEven(num):
        even_numbers.append(num)
print('Non-Pythonic Approach: ', even_numbers)

Non-Pythonic Approach: [2, 4, 6, 8, 0, 10]

# A pythonic approach would be to use filter()
even_numbers = filter(isEven, numbers)
print('Pythonic Approach: ', list(even_numbers))

Pythonic Approach: [2, 4, 6, 8, 0, 10]
```

10. Advanced – Parallel Processing

```
import random
import pandas as pd
import numpy as np
from multiprocessing import Pool

import pandas as pd

file_name = "https://raw.githubusercontent.com/rajeevratan84/datascienceforbusiness/master/new_york_hotels.csv"
df = pd.read_csv(file_name, encoding = "ISO-8859-1")
df.head()
```

	ean_hotel_id	name	address1	city	state_province	postal_code	latitude	longitude	star_rating	high_rate	low_rate
0	269955	Hilton Garden Inn Albany/SUNY Area	1389 Washington Ave	Albany	NY	12206	42.68751	-73.81643	3.0	154.0272	124.0216
1	113431	Courtyard by Marriott Albany Thruway	1455 Washington Avenue	Albany	NY	12206	42.68971	-73.82021	3.0	179.0100	134.0000
2	108151	Radisson Hotel Albany	205 Wolf Rd	Albany	NY	12205	42.72410	-73.79822	3.0	134.1700	84.1600
3	254756	Hilton Garden Inn Albany Medical Center	62 New Scotland Ave	Albany	NY	12208	42.65157	-73.77638	3.0	308.2807	228.4597
4	198232	CrestHill Suites SUNY University Albany	1415 Washington Avenue	Albany	NY	12206	42.68873	-73.81854	3.0	169.3900	89.3900

```
# Define a basic Haversine distance formula
def haversine(df):
    lat1, lon1, lat2, lon2 = 40.671, -73.985, df['latitude'].values, df['longitude'].values
    MILES = 3959
    lat1, lon1, lat2, lon2 = map(np.deg2rad, [lat1, lon1, lat2, lon2])
    dlat = lat2 - lat1
    dlon = lon2 - lon1
    a = np.sin(dlat/2)**2 + np.cos(lat1) * np.cos(lat2) * np.sin(dlon/2)**2
    c = 2 * np.arcsin(np.sqrt(a))
    total_miles = MILES * c
    df["distance"] = total_miles
    return df
```

```
%%timeit

# Vectorized implementation of Haversine applied on NumPy arrays (note we remove the values to access them)
distance_df = haversine(df)
```

The slowest run took 8.71 times longer than the fastest. This could mean that an intermediate result is being cached.
1000 loops, best of 3: 280 µs per loop

```
def parallelize_dataframe(df, func, n_cores=4):
    df_split = np.array_split(df, n_cores)
    pool = Pool(n_cores)
    df = pd.concat(pool.map(func, df_split))
    pool.close()
    pool.join()
    return df
```

```
%%timeit

distance_df = parallelize_dataframe(df, haversine)
```

10 loops, best of 3: 134 ms per loop

Introduction to pandas

1. Pandas and NumPy.....	2
2. Introduction to the Data.....	2
3. Introducing DataFrames	3
4. Selecting a Column From a DataFrame by Label	4
5. Introduction to Series	5
6. Selecting Columns From a DataFrame by Label Continued.	6
7. Selecting Rows From a DataFrame by Label.....	7
8. Series vs DataFrames	8
9. Value Counts Method.....	9
10. Selecting Items From a Series by Label.....	10

1. Pandas and NumPy

Pandas không phải là sự thay thế cho NumPy, mà là phần mở rộng của NumPy. Mã cơ bản cho pandas sử dụng rộng rãi thư viện NumPy, điều đó có nghĩa là các khái niệm bạn đã học sẽ có ích khi bạn bắt đầu tìm hiểu thêm về pandas. Cấu trúc dữ liệu chính trong pandas được gọi là dataframe.

Dataframes là pandas tương đương với ndarray Numpy 2D, với một vài điểm khác biệt chính:

- Giá trị trực có thể có nhãn chuỗi, không chỉ nhãn số.
- Dataframe có thể chứa các cột có nhiều kiểu dữ liệu: bao gồm số nguyên, số float và chuỗi.

The diagram illustrates the structure of a Pandas DataFrame. It consists of a grid of data with row and column labels. The columns are labeled 'rank', 'revenues', 'profits', and 'country'. The rows are labeled with company names: 'Walmart', 'State Grid', 'Sinopec Group', 'China Natural Petroleum', and 'Toyota Motor'. A red arrow labeled 'Index Axis' points vertically down the first column. A blue arrow labeled 'Column Labels' points horizontally across the top row. A red arrow labeled 'Column Axis' points horizontally to the right from the top edge of the grid. Below the grid, arrows indicate data types: a blue arrow labeled 'Row Labels' points to the company names; a blue arrow labeled 'Integer Type' points to the 'rank' column; a green arrow labeled 'Float Type' points to the 'profits' column; and a green arrow labeled 'String Type' points to the 'country' column.

	rank	revenues	profits	country
Walmart	1	485873	13643.0	USA
State Grid	2	315199	9571.3	China
Sinopec Group	3	267518	1257.9	China
China Natural Petroleum	4	262573	1867.5	China
Toyota Motor	5	254694	16899.3	Japan

2. Introduction to the Data

Giống như quy ước nhập cho NumPy (nhập numpy dưới dạng np), quy ước nhập cho pandas là:

```

import pandas as pd
f500 = pd.read_csv('f500.csv', index_col=0)
f500.index.name = None
f500_type=type(f500)
f500_shape=f500.shape
print(f500_shape)
print(f500_type)
f500

```

(500, 16)
<class 'pandas.core.frame.DataFrame'>

	rank	revenues	revenue_change	profits	assets	profit_change	ceo	industry	sector	previous_rank	country	hq_location
Walmart	1	485873	0.8	13643.0	198825	-7.2	C. Douglas McMillon	General Merchandisers	Retailing	1	USA	Bentonville, AR
State Grid	2	315199	-4.4	9571.3	489838	-6.2	Kou Wei	Utilities	Energy	2	China	Beijing, China
Sinopec Group	3	267518	-9.1	1257.9	310726	-65.0	Wang Yupu	Petroleum Refining	Energy	4	China	Beijing, China
China National Petroleum	4	262573	-12.3	1867.5	585619	-73.7	Zhang Jianhua	Petroleum Refining	Energy	3	China	Beijing, China
Toyota Motor	5	254694	7.7	16899.3	437575	-12.3	Akio Toyoda	Motor Vehicles and Parts	Motor Vehicles & Parts	8	Japan	Toyota, Japan
...
Teva Pharmaceutical Industries	496	21903	11.5	329.0	92890	-79.3	Yitzhak Peterburg	Pharmaceuticals	Health Care	0	Israel	Petach Tikva, Israel
New China Life Insurance	497	21796	-13.3	743.9	100609	-45.6	Wan Feng	Insurance: Life, Health (stock)	Financials	427	China	Beijing, China
Wm. Morrison Supermarkets	498	21741	-11.3	406.4	11630	20.4	David T. Potts	Food and Drug Stores	Food & Drug Stores	437	Britain	Bradford, Britain
TUI	499	21655	-5.5	1151.7	16247	195.5	Friedrich Joussen	Travel Services	Business Services	467	Germany	Hanover, Germany
AutoNation	500	21609	3.6	430.5	10060	-2.7	Michael J. Jackson	Specialty Retailers	Retailing	0	USA	Lauderdale, Florida

500 rows × 16 columns

3. Introducing DataFrames

Đối tượng DataFrame - hoặc chỉ dataframe, cấu trúc dữ liệu chính của pandas. Có thể sử dụng phương thức DataFrame.head(). Theo mặc định, nó trả về năm hàng đầu tiên trong dataframe

```
f500_head = f500.head(6)
f500_head
```

	rank	revenues	revenue_change	profits	assets	profit_change	ceo	industry	sector	previous_rank	country	hq_location
Walmart	1	485873	0.8	13643.0	198825	-7.2	C. Douglas McMillon	General Merchandisers	Retailing	1	USA	Bentonville, AR
State Grid	2	315199	-4.4	9571.3	489838	-6.2	Kou Wei	Utilities	Energy	2	China	Beijing, China
Sinopec Group	3	267518	-9.1	1257.9	310726	-65.0	Wang Yupu	Petroleum Refining	Energy	4	China	Beijing, China
China National Petroleum	4	262573	-12.3	1867.5	585619	-73.7	Zhang Jianhua	Petroleum Refining	Energy	3	China	Beijing, China
Toyota Motor	5	254694	7.7	16899.3	437575	-12.3	Akio Toyoda	Motor Vehicles and Parts	Motor Vehicles & Parts	8	Japan	Toyota, Japan
Volkswagen	6	240264	1.5	5937.3	432116	Nan	Matthias Muller	Motor Vehicles and Parts	Motor Vehicles & Parts	7	Germany	Wolfsburg, Germany

Có thể sử dụng phương thức DataFrame.tail() để hiển thị cho chúng tôi các hàng cuối cùng của dataframe:

```
f500_tail = f500.tail(8)
f500_tail
```

	rank	revenues	revenue_change	profits	assets	profit_change	ceo	industry	sector	previous_rank	country
Telecom Italia	493	21941	-17.4	1999.4	74295	NaN	Flavio Cattaneo	Telecommunications	Telecommunications	404	Italy
Xiamen ITG Holding Group	494	21930	34.3	35.6	12161	-25.1	Xu Xiaoxi	Trading	Wholesalers	0	China
Xinjiang Guanghui Industry Investment	495	21919	31.1	251.8	31957	49.9	Shang Jiqiang	Trading	Wholesalers	0	China
Teva Pharmaceutical Industries	496	21903	11.5	329.0	92890	-79.3	Yitzhak Peterburg	Pharmaceuticals	Health Care	0	Israel
New China Life Insurance	497	21796	-13.3	743.9	100609	-45.6	Wan Feng	Insurance: Life, Health (stock)	Financials	427	China
Wm. Morrison Supermarkets	498	21741	-11.3	406.4	11630	20.4	David T. Potts	Food and Drug Stores	Food & Drug Stores	437	Britain
TUI	499	21655	-5.5	1151.7	16247	195.5	Friedrich Joussen	Travel Services	Business Services	467	Germany
AutoNation	500	21609	3.6	430.5	10060	-2.7	Michael J. Jackson	Specialty Retailers	Retailing	0	USA

```
f500.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 500 entries, Walmart to AutoNation
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   rank             500 non-null    int64  
 1   revenues         500 non-null    int64  
 2   revenue_change  498 non-null    float64 
 3   profits          499 non-null    float64 
 4   assets           500 non-null    int64  
 5   profit_change   436 non-null    float64 
 6   ceo              500 non-null    object  
 7   industry         500 non-null    object  
 8   sector            500 non-null    object  
 9   previous_rank   500 non-null    int64  
 10  country          500 non-null    object  
 11  hq_location      500 non-null    object  
 12  website           500 non-null    object  
 13  years_on_global_500_list  500 non-null    int64  
 14  employees        500 non-null    int64  
 15  total_stockholder_equity 500 non-null    int64  
dtypes: float64(3), int64(7), object(6)
memory usage: 66.4+ KB
```

4. Selecting a Column From a DataFrame by Label

```
f500.loc[:, "rank"]
```

```
Walmart                  1
State Grid                2
Sinopec Group              3
China National Petroleum      4
Toyota Motor                 5
...
Teva Pharmaceutical Industries  496
New China Life Insurance      497
Wm. Morrison Supermarkets      498
TUI                         499
AutoNation                   500
Name: rank, Length: 500, dtype: int64
```

```

industries = f500["industry"]
industries_type = type(industries)
print(industries_type)
industries

<class 'pandas.core.series.Series'>

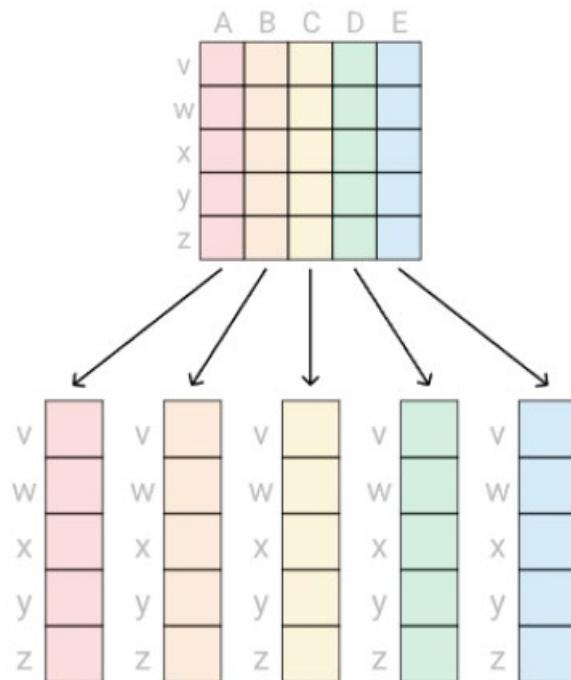
Walmart           General Merchandisers
State Grid        Utilities
Sinopec Group    Petroleum Refining
China National Petroleum  Petroleum Refining
Toyota Motor     Motor Vehicles and Parts
...
Teva Pharmaceutical Industries Pharmaceuticals
New China Life Insurance Insurance: Life, Health (stock)
Wm. Morrison Supermarkets Food and Drug Stores
TUI             Travel Services
AutoNation       Specialty Retailers
Name: industry, Length: 500, dtype: object

```

5. Introduction to Series

Sê-ri là loại pandas cho đối tượng một chiều. Bất cứ khi nào bạn nhìn thấy một đối tượng pandas 1D, nó sẽ là một chuỗi.

Bất cứ khi nào bạn nhìn thấy một đối tượng pandas 2D, nó sẽ là một khung dữ liệu. Trên thực tế, bạn có thể coi một khung dữ liệu là một tập hợp các đối tượng sê-ri, tương tự như cách pandas lưu trữ dữ liệu đằng sau hậu trường.



6. Selecting Columns From a DataFrame by Label Continued.

```
countries = f500["country"]
countries

Walmart           USA
State Grid        China
Sinopetc Group   China
China National Petroleum   China
Toyota Motor     Japan
...
Teva Pharmaceutical Industries  Israel
New China Life Insurance  China
Wm. Morrison Supermarkets  Britain
TUI               Germany
AutoNation        USA
Name: country, Length: 500, dtype: object
```

```
revenues_years = f500[["revenues", "years_on_global_500_list"]]
revenues_years
```

	revenues	years_on_global_500_list
Walmart	485873	23
State Grid	315199	17
Sinopetc Group	267518	19
China National Petroleum	262573	17
Toyota Motor	254694	23
...
Teva Pharmaceutical Industries	21903	1
New China Life Insurance	21796	2
Wm. Morrison Supermarkets	21741	13
TUI	21655	23
AutoNation	21609	12

500 rows × 2 columns

```
ceo_to_sector = f500.loc[:, "ceo":"sector"]
ceo_to_sector
```

	ceo	industry	sector
Walmart	C. Douglas McMillon	General Merchandisers	Retailing
State Grid	Kou Wei	Utilities	Energy
Sinopetc Group	Wang Yupu	Petroleum Refining	Energy
China National Petroleum	Zhang Jianhua	Petroleum Refining	Energy
Toyota Motor	Akio Toyoda	Motor Vehicles and Parts	Motor Vehicles & Parts
...
Teva Pharmaceutical Industries	Yitzhak Peterburg	Pharmaceuticals	Health Care
New China Life Insurance	Wan Feng	Insurance: Life, Health (stock)	Financials
Wm. Morrison Supermarkets	David T. Potts	Food and Drug Stores	Food & Drug Stores
TUI	Friedrich Joussen	Travel Services	Business Services
AutoNation	Michael J. Jackson	Specialty Retailers	Retailing

500 rows × 3 columns

7. Selecting Rows From a DataFrame by Label

```
| toyota = f500.loc['Toyota Motor']
toyota
```

rank		5
revenues		254694
revenue_change		7.7
profits		16899.3
assets		437575
profit_change		-12.3
ceo		Akio Toyoda
industry		Motor Vehicles and Parts
sector		Motor Vehicles & Parts
previous_rank		8
country		Japan
hq_location		Toyota, Japan
website		http://www.toyota-global.com
years_on_global_500_list		23
employees		364445
total_stockholder_equity		157210
Name:	Toyota Motor	dtype: object

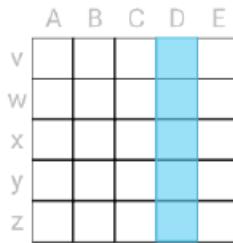
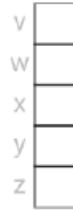
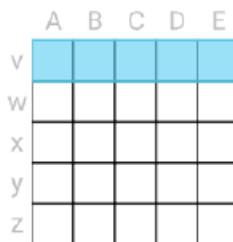
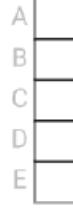
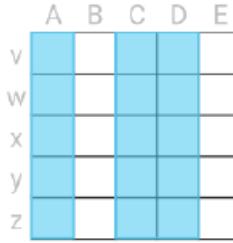
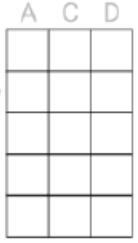
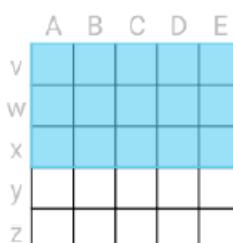
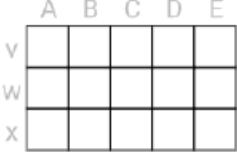
```
| drink_companies = f500.loc[['Anheuser-Busch InBev', 'Coca-Cola', 'Heineken Holding']]
drink_companies
```

	rank	revenues	revenue_change	profits	assets	profit_change	ceo	industry	sector	previous_rank	country	hq_location
Anheuser-Busch InBev	206	45905	5.3	1241.0	258381	-85.0	Carlos Brito	Beverages	Food, Beverages & Tobacco	211	Belgium	Leuven, Belgium
Coca-Cola	235	41863	-5.5	6527.0	87270	-11.2	James B. Quincey	Beverages	Food, Beverages & Tobacco	206	USA	Atlanta, GA
Heineken Holding	468	23044	-0.7	861.5	41469	-18.9	Jean-Francois van Boxmeer	Beverages	Food, Beverages & Tobacco	459	Netherlands	Amsterdam, Netherlands

```
| middle_companies = f500.loc['Tata Motors':'Nationwide', 'rank':'country']
middle_companies
```

	rank	revenues	revenue_change	profits	assets	profit_change	ceo	industry	sector	previous_rank	country
Tata Motors	247	40329	-4.2	1111.6	42162	-34.0	Guenter Butschek	Motor Vehicles and Parts	Motor Vehicles & Parts	226	India
Aluminum Corp. of China	248	40278	6.0	-282.5	75089	NaN	Yu Dehui	Metals	Materials	262	China
Mitsui	249	40275	1.6	2825.3	103231	NaN	Tatsuo Yasunaga	Trading	Wholesalers	245	Japan
Manulife Financial	250	40238	49.4	2209.7	537461	28.9	Donald A. Gulloien	Insurance: Life, Health (stock)	Financials	394	Canada
China Minsheng Banking	251	40234	-5.2	7201.6	848389	-1.8	Zheng Wanchun	Banks: Commercial and Savings	Financials	221	China
China Pacific Insurance (Group)	252	40193	2.2	1814.9	146873	-35.7	Huo Lianhong	Insurance: Life, Health (stock)	Financials	251	China
American Airlines Group	253	40180	-2.0	2676.0	51274	-64.8	W. Douglas Parker	Airlines	Transportation	236	USA
Nationwide	254	40074	-0.4	334.3	197790	-42.4	Stephen S. Rasmussen	Insurance: Property and Casualty (Mutual)	Financials	241	USA

8. Series vs DataFrames

Original Dataframe	Code	Result
	<code>single_col = df["D"]</code>	 <code>single_col</code> is a series object
	<code>single_row = df.loc["v"]</code>	 <code>single_row</code> is a series object
Original Dataframe	Code	Result
	<code>mult_cols = df[["A", "C", "D"]]</code>	 <code>mult_cols</code> is a dataframe object
	<code>mult_rows = df.loc[["v", "w", "x"]]</code>	 <code>mult_rows</code> is a dataframe object

9. Value Counts Method

Phương pháp này hiển thị từng giá trị khác null duy nhất trong một cột và số lượng của chúng theo thứ tự.

```
sectors = f500['sector']
print(type(sectors))
sectors_value_counts = sectors.value_counts()
sectors_value_counts
```

```
<class 'pandas.core.series.Series'>
```

```
Financials           118
Energy              80
Technology          44
Motor Vehicles & Parts 34
Wholesalers         28
Health Care          27
Food & Drug Stores   20
Transportation        19
Telecommunications    18
Retailing             17
Food, Beverages & Tobacco 16
Materials             16
Industrials            15
Aerospace & Defense    14
Engineering & Construction 13
Chemicals               7
Household Products      3
Media                  3
Hotels, Restaurants & Leisure 3
Business Services       3
Apparel                 2
Name: sector, dtype: int64
```

```
sectors_industries = f500[['sector', 'industry']]
print(type(sectors_industries))
si_value_counts = sectors_industries.value_counts()
si_value_counts
```

```
<class 'pandas.core.frame.DataFrame'>
```

sector	industry	
Financials	Banks: Commercial and Savings	51
Motor Vehicles & Parts	Motor Vehicles and Parts	34
Energy	Petroleum Refining	28
Financials	Insurance: Life, Health (stock)	24
Food & Drug Stores	Food and Drug Stores	20
Energy	Mining, Crude-Oil Production	18
Financials	Insurance: Property and Casualty (Stock)	18
Telecommunications	Telecommunications	18
Energy	Utilities	18
Wholesalers	Trading	15
Health Care	Pharmaceuticals	15
Aerospace & Defense	Aerospace and Defense	14
Engineering & Construction	Engineering, Construction	13
Technology	Electronics, Electrical Equip.	13
Energy	Energy	12
Materials	Metals	12
Retailing	Specialty Retailers	10

10. Selecting Items From a Series by Label

Tìm số lượng của từng giá trị duy nhất trong cột quốc gia cho toàn bộ khung dữ liệu f500:

```
countries = f500['country']
countries_counts = countries.value_counts()
countries_counts
```

```
USA          132
China        109
Japan         51
Germany       29
France         29
Britain        24
South Korea    15
Netherlands     14
Switzerland     14
Canada          11
Spain            9
Australia        7
Brazil            7
India             7
Italy              7
Taiwan            6
Russia            4
Ireland            4
Singapore          3
Sweden            3
Mexico            2
Malaysia           1
Thailand           1
Belgium            1
Norway             1
Luxembourg         1
Indonesia           1
Denmark             1
Saudi Arabia        1
Finland             1
Venezuela           1
Turkey              1
U.A.E              1
Israel              1
Name: country, dtype: int64
```

```
india = countries_counts['India']
india
```

```
7
```

```
north_america = countries_counts[['USA', 'Canada', 'Mexico']]
north_america
```

```
USA          132
Canada        11
Mexico          2
Name: country, dtype: int64
```

Exploring Data with Pandas Fundamentals

1. Introduction to the Data.....	2
2. Vectorized Operations	2
3. Series Data Exploration Methods	3
4. Series Describe Method.....	3
5. Method Chaining	5
6. Dataframe Exploration Methods	6
7. Dataframe Describe Method.....	7
8. Assignment with Pandas.....	7
9. Using Boolean Indexing with Pandas Objects.....	9
10. Using Boolean Arrays to Assign Values	10
11. Creating New Columns.....	11
12. Challenge: Top Performers by Country	12

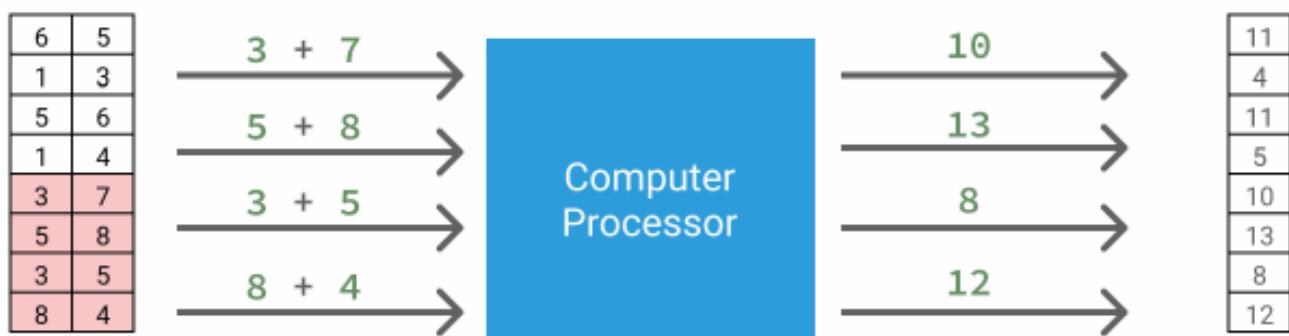
1. Introduction to the Data

```
f500 = pd.read_csv('f500.csv', index_col = 0)
f500_head = f500.head(10)
f500.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 500 entries, Walmart to AutoNation
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   rank              500 non-null    int64  
 1   revenues          500 non-null    int64  
 2   revenue_change    498 non-null    float64 
 3   profits           499 non-null    float64 
 4   assets            500 non-null    int64  
 5   profit_change    436 non-null    float64 
 6   ceo               500 non-null    object  
 7   industry          500 non-null    object  
 8   sector             500 non-null    object  
 9   previous_rank     500 non-null    int64  
 10  country           500 non-null    object  
 11  hq_location        500 non-null    object  
 12  website            500 non-null    object  
 13  years_on_global_500_list 500 non-null    int64  
 14  employees          500 non-null    int64  
 15  total_stockholder_equity 500 non-null    int64  
dtypes: float64(3), int64(7), object(6)
memory usage: 66.4+ KB
```

2. Vectorized Operations

Vì pandas được thiết kế để hoạt động giống như NumPy nên rất nhiều khái niệm và phương thức từ Numpy được hỗ trợ. Hãy nhớ lại rằng một trong những cách NumPy giúp làm việc với dữ liệu dễ dàng hơn là với các phép toán được vector hóa hoặc các phép toán được áp dụng cho nhiều điểm dữ liệu cùng một lúc



```
rank_change = f500['previous_rank'] - f500['rank']  
rank_change
```

```
company  
Walmart 0  
State Grid 0  
Sinopec Group 1  
China National Petroleum -1  
Toyota Motor 3  
...  
Teva Pharmaceutical Industries -496  
New China Life Insurance -70  
Wm. Morrison Supermarkets -61  
TUI -32  
AutoNation -500  
Length: 500, dtype: int64
```

3. Series Data Exploration Methods

```
rank_change_max = rank_change.max()  
rank_change_max
```

```
226
```

```
rank_change_min = rank_change.min()  
rank_change_min
```

```
-500
```

4. Series Describe Method

Phương pháp Series.describe(). Phương thức này cho chúng tôi biết có bao nhiêu giá trị khác null được chứa trong chuỗi, cùng với giá trị trung bình, giá trị nhỏ nhất, giá trị lớn nhất và các thống kê khác

```
assets = f500['assets']  
assets.describe()
```



```
count      5.000000e+02  
mean      2.436323e+05  
std       4.851937e+05  
min       3.717000e+03  
25%       3.658850e+04  
50%       7.326150e+04  
75%       1.805640e+05  
max       3.473238e+06  
Name: assets, dtype: float64
```

Sử dụng description() trên một cột chứa các giá trị không phải là số, chúng ta sẽ nhận được một số thống kê khác. Hãy xem xét một ví dụ:

```
country = f500['country']
country.describe()
```

```
count      500
unique     34
top       USA
freq      132
Name: country, dtype: object
```

```
rank = f500['rank']
rank_desc = rank.describe
rank_desc
```

```
<bound method NDFrame.describe of company
Walmart                  1
State Grid                2
Sinopec Group              3
China National Petroleum        4
Toyota Motor                 5
...
Teva Pharmaceutical Industries    496
New China Life Insurance        497
Wm. Morrison Supermarkets      498
TUI                         499
AutoNation                   500
Name: rank, Length: 500, dtype: int64>
```

```
countries_counts = f500['country'].value_counts()
countries_counts
```

USA	132
China	109
Japan	51
Germany	29
France	29
Britain	24
South Korea	15
Netherlands	14
Switzerland	14
Canada	11
Spain	9
Australia	7
Brazil	7
India	7
Italy	7
Taiwan	6
Russia	4
Ireland	4
Singapore	3
Sweden	3
Mexico	2
Malaysia	1
Thailand	1
Belgium	1
Norway	1
Luxembourg	1
Indonesia	1
Denmark	1
Saudi Arabia	1
Finland	1
Venezuela	1
Turkey	1
U.A.E	1
Israel	1

Name: country, dtype: int64

```
prev_rank = f500['previous_rank']
pre_rank_desc = prev_rank.describe()
pre_rank_desc
```

count	500.000000
mean	222.134000
std	146.941961
min	0.000000
25%	92.750000
50%	219.500000
75%	347.250000
max	500.000000

Name: previous_rank, dtype: float64

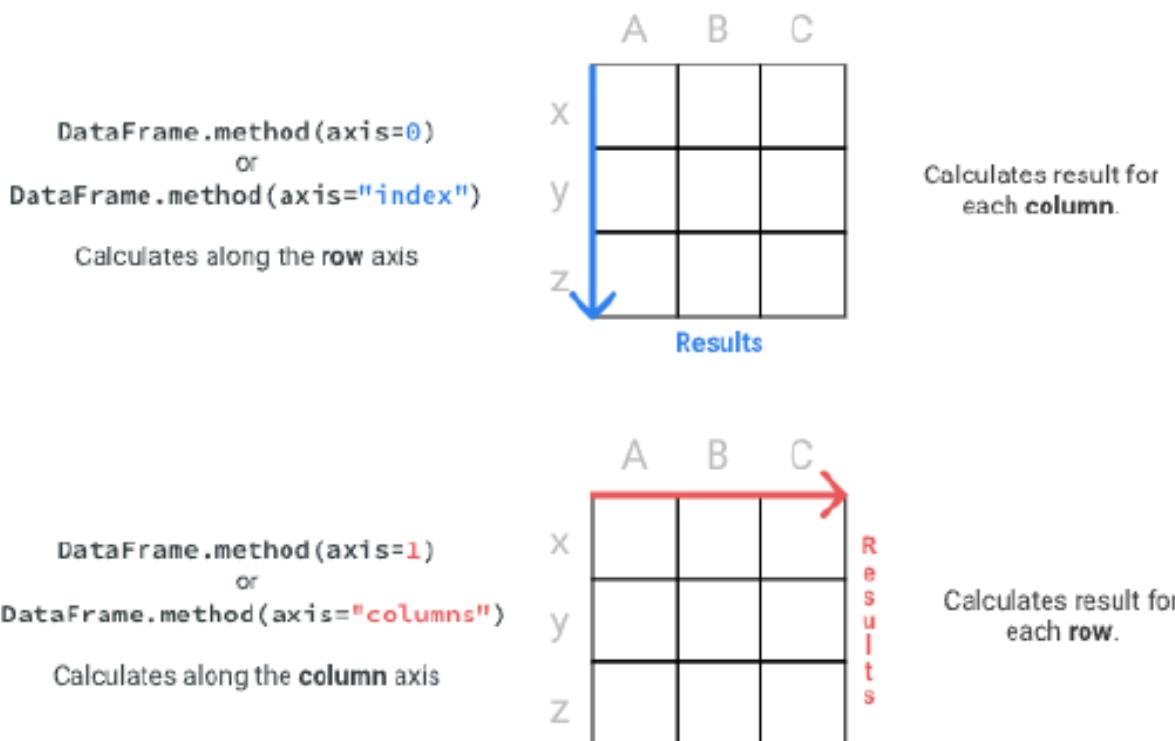
5. Method Chaining

```
zero_previous_rank: object = f500['previous_rank'].value_counts().loc[0]  
zero_previous_rank
```

33

6. Dataframe Exploration Methods

Không giống như các đối tác sê-ri của chúng, các phương thức khung dữ liệu yêu cầu tham số trực để chúng tôi biết trực nào cần tính toán. Mặc dù bạn có thể sử dụng các số nguyên để chỉ trực thứ nhất và trực thứ hai, nhưng các phương thức khung dữ liệu của pandas cũng chấp nhận các chuỗi "chỉ mục" và "cột" cho tham số trực:



```
medians = f500[['revenues', 'profits']].median(axis='index')  
medians
```

```
revenues    40236.0  
profits     1761.6  
dtype: float64
```

Chỉ bao gồm các cột float, int, boolean. Nếu Không, sẽ cố gắng sử dụng mọi thứ, sau đó chỉ sử dụng dữ liệu số. Không được triển khai cho Sê-ri.

```
max_f500 = f500.max(numeric_only=True)
max_f500
```

```
rank                  500.0
revenues             485873.0
revenue_change       442.3
profits              45687.0
assets               3473238.0
profit_change        8909.5
previous_rank        500.0
years_on_global_500_list    23.0
employees            2300000.0
total_stockholder_equity   301893.0
dtype: float64
```

7. Dataframe Describe Method

Theo mặc định, DataFrame.describe() sẽ chỉ trả lại số liệu thống kê cho các cột số. Nếu chúng tôi chỉ muốn lấy các cột đối tượng, chúng tôi cần sử dụng tham số include=['O'] :

```
f500.describe(include=['O'])
```

	ceo	industry	sector	country	hq_location	website
count	500	500	500	500	500	500
unique	500	58	21	34	235	500
top	C. Douglas McMillon	Banks: Commercial and Savings	Financials	USA	Beijing, China	http://www.walmart.com
freq	1	51	118	132	56	1

```
f500_desc = f500.describe()
f500_desc
```

	rank	revenues	revenue_change	profits	assets	profit_change
count	500.000000	500.000000	498.000000	499.000000	5.000000e+02	436.000000
mean	250.500000	55416.358000	4.538353	3055.203206	2.436323e+05	24.152752
std	144.481833	45725.478963	28.549067	5171.981071	4.851937e+05	437.509566
min	1.000000	21609.000000	-67.300000	-13038.000000	3.717000e+03	-793.700000
25%	125.750000	29003.000000	-5.900000	556.950000	3.658850e+04	-22.775000
50%	250.500000	40236.000000	0.550000	1761.600000	7.326150e+04	-0.350000
75%	375.250000	63926.750000	6.975000	3954.000000	1.805640e+05	17.700000
max	500.000000	485873.000000	442.300000	45687.000000	3.473238e+06	8909.500000

8. Assignment with Pandas

Chúng ta sẽ học cách thực hiện hai việc để có thể sửa các giá trị này:

- Thực hiện chuyển nhượng trong pandas.
- Sử dụng lập chỉ mục boolean trong pandas

```
top5_rank_revenue = f500[['rank', 'revenues']].head()
top5_rank_revenue
```

company	rank	revenues
Walmart	1	485873
State Grid	2	315199
Sinopec Group	3	267518
China National Petroleum	4	262573
Toyota Motor	5	254694

```
top5_rank_revenue['revenues'] = 0
top5_rank_revenue
```

company	rank	revenues
Walmart	1	0
State Grid	2	0
Sinopec Group	3	0
China National Petroleum	4	0
Toyota Motor	5	0

Bằng cách cung cấp nhãn cho cả hai trục, chúng tôi có thể gán chúng cho một giá trị duy nhất trong khung dữ liệu của mình.

```
top5_rank_revenue.loc["Sinopec Group", 'revenues'] = 999
top5_rank_revenue
```

company	rank	revenues
Walmart	1	0
State Grid	2	0
Sinopec Group	3	999
China National Petroleum	4	0
Toyota Motor	5	0

```
f500.loc['Dow Chemical', 'ceo'] = 'Jim Fitterling'
```

```
f500.loc['Dow Chemical', 'ceo']
```

```
'Jim Fitterling'
```

9. Using Boolean Indexing with Pandas Objects

Hãy kiểm tra xem mọi người có số yêu thích là 8. Đầu tiên, chúng tôi thực hiện phép toán boolean được vector hóa để tạo ra một chuỗi boolean:

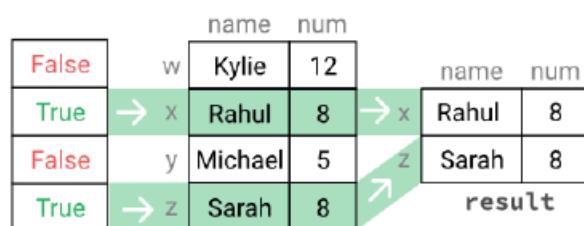
```
num_bool = df["num"] == 8
```

w	False
x	True
y	False
z	True

num_bool

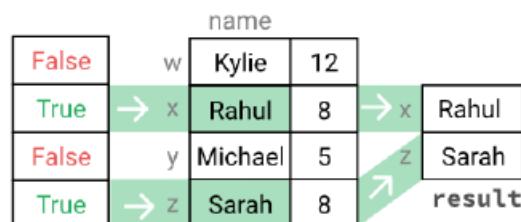
Chúng ta có thể sử dụng chuỗi đó để lập chỉ mục toàn bộ khung dữ liệu, để lại cho chúng tôi các hàng chỉ tương ứng với những người có Số Boolean yêu thích là 8:

```
result = df[num_bool]
```



Lưu ý rằng chúng tôi không sử dụng `loc[]`. Điều này là do các mảng boolean sử dụng cùng một phím tắt như các lát cắt để chọn lọc theo trực chỉ mục. Chúng ta cũng có thể sử dụng chuỗi Boolean để lập chỉ mục cho một cột của khung dữ liệu:

```
result = df.loc[num_bool, "name"]
```



```
| motor_bool = f500['industry'] == 'Motor Vehicles and Parts'  
motor_bool  
  
company  
Walmart False  
State Grid False  
Sinopec Group False  
China National Petroleum False  
Toyota Motor True  
...  
Teva Pharmaceutical Industries False  
New China Life Insurance False  
Wm. Morrison Supermarkets False  
TUI False  
AutoNation False  
Name: industry, Length: 500, dtype: bool
```

```
| motor_countries = f500.loc[motor_bool, 'country']  
motor_countries
```

```
company  
Toyota Motor Japan  
Volkswagen Germany  
Daimler Germany  
General Motors USA  
Ford Motor USA  
Honda Motor Japan  
SAIC Motor China  
Nissan Motor Japan  
BMW Group Germany  
Dongfeng Motor China  
Robert Bosch Germany  
Hyundai Motor South Korea  
China FAW Group China  
Beijing Automotive Group China  
Peugeot France  
Renault France  
Kia Motors South Korea  
Continental Germany  
Denso Japan  
Guangzhou Automobile Industry Group China  
Tata Motors India  
ZF Friedrichshafen Germany  
Jardine Matheson China  
Magna International Canada  
Volvo Sweden  
Hyundai Mobis South Korea  
Aisin Seiki Japan  
Zhejiang Geely Holding Group China  
Subaru Japan  
Bridgestone Japan  
Mazda Motor Japan  
Suzuki Motor Japan  
Sumitomo Electric Industries Japan  
Michelin France  
Name: country, dtype: object
```

10. Using Boolean Arrays to Assign Values

```
ampersand_bool = f500['sector'] == 'Motor Vehicles & Parts'  
f500.loc[ampersand_bool, 'sector'] = 'Motor Vehicles and Parts'
```

```
f500['sector']
```

```
company  
Walmart                               Retailing  
State Grid                            Energy  
Sinopec Group                         Energy  
China National Petroleum                Energy  
Toyota Motor                           Motor Vehicles and Parts  
...  
Teva Pharmaceutical Industries          Health Care  
New China Life Insurance               Financials  
Wm. Morrison Supermarkets            Food & Drug Stores  
TUI                                    Business Services  
AutoNation                            Retailing  
Name: sector, Length: 500, dtype: object
```

```
f500.loc[f500['sector']=='Motor Vehicles & Parts','sector']='Motor Vehicles and Parts'  
f500['sector']
```

```
company  
Walmart                               Retailing  
State Grid                            Energy  
Sinopec Group                         Energy  
China National Petroleum                Energy  
Toyota Motor                           Motor Vehicles and Parts  
...  
Teva Pharmaceutical Industries          Health Care  
New China Life Insurance               Financials  
Wm. Morrison Supermarkets            Food & Drug Stores  
TUI                                    Business Services  
AutoNation                            Retailing  
Name: sector, Length: 500, dtype: object
```

```
prev_rank_before = f500['previous_rank'].value_counts(dropna=False).head()  
prev_rank_before
```

```
NaN      33  
1.0      1  
302.0    1  
334.0    1  
325.0    1  
Name: previous_rank, dtype: int64
```

11. Creating New Columns

```
import numpy as np  
f500.loc[f500['previous_rank']==0, 'previous_rank']=np.nan  
prev_rank_after = f500['previous_rank'].value_counts(dropna=False).head()  
prev_rank_after
```

```
NaN      33  
1.0      1  
302.0    1  
334.0    1  
325.0    1  
Name: previous_rank, dtype: int64
```

```
| top5_rank_revenue['year_founded'] = 0  
top5_rank_revenue
```

company	rank	revenues	year_founded
Walmart	1	0	0
State Grid	2	0	0
Sinopec Group	3	999	0
China National Petroleum	4	0	0
Toyota Motor	5	0	0

```
f500['rank_change'] = f500['previous_rank'] - f500['rank']  
rank_change_desc = f500['rank_change'].describe()  
rank_change_desc
```

```
count    467.000000  
mean     -3.533191  
std      44.293603  
min     -199.000000  
25%    -21.000000  
50%    -2.000000  
75%    10.000000  
max     226.000000  
Name: rank_change, dtype: float64
```

12. Challenge: Top Performers by Country

```
top2_countries = f500['country'].value_counts().head(2)  
top2_countries
```

```
USA      132  
China    109  
Name: country, dtype: int64
```

```
industry_usa = f500['industry'][f500['country']=='USA'].value_counts().head(2)  
industry_usa
```

Banks: Commercial and Savings 8
Insurance: Property and Casualty (Stock) 7
Name: industry, dtype: int64

```
sector_china = f500['sector'][f500['country']=='China'].value_counts().head(3)  
sector_china
```

Financials 25
Energy 22
Wholesalers 9
Name: sector, dtype: int64

Exploring Data with Pandas Intermediate

1. Introduction	2
2. Reading CSV Files with Pandas.....	2
3. Using Iloc to Select by Integer Position	2
4. Using Iloc to Select by Integer Position Continued.....	4
5. Using Pandas Methods to Create Boolean Masks	5
6. Working with Integer Labels.....	6
7. Pandas Index Alignment.....	6
8. Using Boolean Operators	8
9. Using Boolean Operators Continued.....	9
10. Sorting Values	10
11. Using Loops with Pandas	11
12. Challenge: Calculating Return on Assets by Country	12

1. Introduction

```
f500 = pd.read_csv('f500.csv', index_col = 0)
f500.index.name = None
```

```
f500.loc[f500['previous_rank'] == 0, 'previous_rank'] = np.nan
f500_selection = f500[['rank', 'revenues', 'revenue_change']].head()
f500_selection
```

	rank	revenues	revenue_change
Walmart	1	485873	0.8
State Grid	2	315199	-4.4
Sinopec Group	3	267518	-9.1
China National Petroleum	4	262573	-12.3
Toyota Motor	5	254694	7.7

2. Reading CSV Files with Pandas

	rank	revenues	profits	country
Walmart	1	485873	13643.0	USA
State Grid	2	315199	9571.3	China
Sinopec Group	3	267518	1257.9	China
China Natural Petroleum	4	262573	1867.5	China
Toyota Motor	5	254694	16899.3	Japan

3. Using Iloc to Select by Integer Position

```
f500 = pd.read_csv('f500.csv')
f500[['company', 'rank', 'revenues']].head()
```

	company	rank	revenues
0	Walmart	1	485873
1	State Grid	2	315199
2	Sinopec Group	3	267518
3	China National Petroleum	4	262573
4	Toyota Motor	5	254694

	A	B	C
x			
y			
z			

```
df.loc["z", "A"]
```

*located at row with label z,
column with label A*

	A	B	C
x			
y			
z			

```
df.loc["y"]
```

located at row with label y

	A	B	C
x			
y			
z			

```
df.iloc[2, 0]
```

	A	B	C
x			
y			
z			

```
df.iloc[1]
```

```
fifth_row = f500.iloc[4]
fifth_row
```

```
company          Toyota Motor
rank                  5
revenues        254694
revenue_change      7.7
profits         16899.3
assets            437575
profit_change     -12.3
ceo                Akio Toyoda
industry       Motor Vehicles and Parts
sector           Motor Vehicles & Parts
previous_rank          8
country             Japan
hq_location        Toyota, Japan
website          http://www.toyota-global.com
years_on_global_500_list    23
employees        364445
total_stockholder_equity   157210
Name: 4, dtype: object
```

4. Using Iloc to Select by Integer Position Continued

```
company_value = f500.iloc[0,0]
company_value
```

```
'Walmart'
```

```
first_column = f500.iloc[:,0]
first_column
```

```
0                  Walmart
1                  State Grid
2                 Sinopec Group
3        China National Petroleum
4                  Toyota Motor
...
495  Teva Pharmaceutical Industries
496  New China Life Insurance
497  Wm. Morrison Supermarkets
498                  TUI
499          AutoNation
Name: company, Length: 500, dtype: object
```

```
second_to_sixth_rows = f500[1:5]
second_to_sixth_rows
```

	company	rank	revenues	revenue_change	profits	assets	profit_change	ceo	in
1	State Grid	2	315199	-4.4	9571.3	489838	-6.2	Kou Wei	I
2	Sinopec Group	3	267518	-9.1	1257.9	310726	-65.0	Wang Yupu	Pet R
3	China National Petroleum	4	262573	-12.3	1867.5	585619	-73.7	Zhang Jianhua	Pet R
4	Toyota Motor	5	254694	7.7	16899.3	437575	-12.3	Akio Toyoda	Van

```
first_three_rows = f500[:3]
first_three_rows
```

	company	rank	revenues	revenue_change	profits	assets	profit_change	ceo	
0	Walmart	1	485873	0.8	13643.0	198825	-7.2	C. Douglas McMillon	Me
1	State Grid	2	315199	-4.4	9571.3	489838	-6.2	Kou Wei	
2	Sinopec Group	3	267518	-9.1	1257.9	310726	-65.0	Wang Yupu	

```
first_seventh_row_slice = f500.iloc[[0, 6], :5]
first_seventh_row_slice
```

	company	rank	revenues	revenue_change	profits
0	Walmart	1	485873	0.8	13643.0
6	Royal Dutch Shell	7	240033	-11.8	4575.0

5. Using Pandas Methods to Create Boolean Masks

```
rev_is_null = f500['revenue_change'].isnull()  
rev_is_null.head()
```

```
0    False  
1    False  
2    False  
3    False  
4    False  
Name: revenue_change, dtype: bool
```

```
rev_change_null = f500[rev_is_null]  
rev_change_null[['company', 'country', 'sector']]
```

	company	country	sector
90	Uniper	Germany	Energy
180	Hewlett Packard Enterprise	USA	Technology

6. Working with Integer Labels

`df.iloc[1]`

iloc[1] uses the integer position of the row to select the second row

	A	B	C
0			
1			
2			

`df.loc[1]`

loc[1] uses the label of the row to select the row with an axis label of 1.

	A	B	C
0			
1			
2			

7. Pandas Index Alignment

```
previously_ranked = f500[f500['previous_rank'].notnull()]
previously_ranked
```

	company	rank	revenues	revenue_change	profits	assets	profit_change	
0	Walmart	1	485873	0.8	13643.0	198825	-7.2	Dou McM
1	State Grid	2	315199	-4.4	9571.3	489838	-6.2	Kou
2	Sinopec Group	3	267518	-9.1	1257.9	310726	-65.0	W Y
3	China National Petroleum	4	262573	-12.3	1867.5	585619	-73.7	Zh Jian
4	Toyota Motor	5	254694	7.7	16899.3	437575	-12.3	Toy
...
495	Teva Pharmaceutical Industries	496	21903	11.5	329.0	92890	-79.3	Yitz Peterl
496	New China Life Insurance	497	21796	-13.3	743.9	100609	-45.6	F F
497	Wm. Morrison Supermarkets	498	21741	-11.3	406.4	11630	20.4	Dav F
498	TUI	499	21655	-5.5	1151.7	16247	195.5	Fried Jous
499	AutoNation	500	21609	3.6	430.5	10060	-2.7	Mic Jack

500 rows × 17 columns

```
rank_change = previously_ranked['previous_rank'] - previously_ranked['rank']
print(rank_change.shape)
rank_change.tail(3)
```

```
(500,)
```

```
497    -61
498    -32
499   -500
dtype: int64
```

```

previously_ranked = f500[f500["previous_rank"].notnull()]
rank_change = previously_ranked[ "previous_rank"] - previously_ranked[ "rank"]
f500["rank_change"] = rank_change
f500["rank_change"]

```

```

0      0
1      0
2      1
3     -1
4      3
...
495   -496
496   -70
497   -61
498   -32
499   -500
Name: rank_change, Length: 500, dtype: int64

```

8. Using Boolean Operators

```

cols = ["company",
        "revenues",
        "country"]
f500_sel = f500[cols].head()

```

	company	revenues	country
0	Walmart	485873	USA
1	State Grid	315199	China
2	Sinopec Group	267518	China
3	China Nation...	262573	China
4	Toyota Motor	254694	Japan

f500_sel

```

over_265 = f500_sel["revenues"] > 265000
china = f500_sel["country"] == "China"

```

0	True	0	False
1	True	1	True
2	True	2	True
3	False	3	True
4	False	4	False

over_265

china

```
combined = over_265 & china
```

0	True	& 0	False	= 0	False
1	True	& 1	True	= 1	True
2	True	& 2	True	= 2	True
3	False	& 3	True	= 3	False
4	False	& 4	False	= 4	False

```
final_cols = ["company", "revenues"]
result = f500_sel.loc[combined, final_cols]
```

		company		revenues	country	
0	False	0	Walmart	485873	USA	
1	True	→ 1	State Grid	315199	China	
2	True	→ 2	Sinopec Group	267518	China	
3	False	3	China Nation...	262573	China	
4	False	4	Toyota Motor	254694	Japan	

combined

f500_sel

	company	revenues
1	State Grid	315199
2	Sinopec Group	267518

result

```
large_revenue = f500['revenues'] > 100000
negative_profits = f500['profits'] < 0
combined = large_revenue & negative_profits
big_rev_neg_profit = f500[combined]
big_rev_neg_profit
```

	company	rank	revenues	revenue_change	profits	assets	profit_change	ceo	industry	sector	previous_rank	country	hq_location	
32	Japan Post Holdings	33	122990		3.6	-267.4	2631385	-107.5	Masatsugu Nagato	Insurance: Life, Health (stock)	Financials	37	Japan	Tokyo, Japan
44	Chevron	45	107567		-18.0	-497.0	260078	-110.8	John S. Watson	Petroleum Refining	Energy	31	USA	San Ramon, CA

```
big_rev_neg_profit = f500[(f500['revenues'] > 100000) & (f500['profits'] < 0)]
big_rev_neg_profit
```

	company	rank	revenues	revenue_change	profits	assets	profit_change	ceo	industry	sector	previous_rank	country	hq_location	
32	Japan Post Holdings	33	122990		3.6	-267.4	2631385	-107.5	Masatsugu Nagato	Insurance: Life, Health (stock)	Financials	37	Japan	Tokyo, Japan
44	Chevron	45	107567		-18.0	-497.0	260078	-110.8	John S. Watson	Petroleum Refining	Energy	31	USA	San Ramon, CA

9. Using Boolean Operators Continued

```

filter_brazil_venezuela = (f500['country'] == 'Brazil') | (f500['country'] == 'Venezuela')
brazil_venezuela = f500[filter_brazil_venezuela]
brazil_venezuela

```

	company	rank	revenues	revenue_change	profits	assets	profit_change	ceo	industry	sector	previous_rank	country	hq_locati
74	Petrobras	75	81405	-16.3	-4838.0	246983	NaN	Pedro Pullen Parente	Petroleum Refining	Energy	58	Brazil	Rio Janeiro Bra
112	Itau Unibanco Holding	113	66876	21.4	6666.4	415972	-13.7	Candido Botelho Bracher	Banks: Commercial and Savings	Financials	159	Brazil	Sao Paulo Bra
150	Banco do Brasil	151	58093	-13.4	2013.8	426416	-52.3	Paulo Rogério Caffarelli	Banks: Commercial and Savings	Financials	115	Brazil	Brasil Bra
153	Banco Bradesco	154	57443	31.3	5127.9	366418	-5.7	Luiz Carlos Trabuco Cappi	Banks: Commercial and Savings	Financials	209	Brazil	Osasco Bra
190	JBS	191	48825	-0.1	107.7	31605	-92.3	Wesley Mendonça Batista	Food Production	Food, Beverages & Tobacco	185	Brazil	Sao Paulo Bra
369	Vale	370	29363	14.7	3982.0	99014	NaN	Fabio Schwartsman	Mining, Crude-Oil Production	Energy	417	Brazil	Rio Janeiro Bra
441	Mercantil Servicios Financieros	442	24403	50.3	2004.2	148659	-10.5	Gustavo J. Vollmer A.	Banks: Commercial and Savings	Financials	0	Venezuela	Caracas Venezuela
486	Ultrapar Holdings	487	22167	-2.3	447.5	7426	-0.8	Thilo Mannhardt	Energy	Energy	474	Brazil	Sao Paulo Bra

```

filter_tech_outside_usa = (f500['sector'] == 'Technology') & ~(f500['country'] == 'USA')
tech_outside_usa = f500[filter_tech_outside_usa].head()
tech_outside_usa

```

	company	rank	revenues	revenue_change	profits	assets	profit_change	ceo	industry	sector	previous_rank	country	hq_locati
14	Samsung Electronics	15	173957	-2.0	19316.5	217104	16.8	Oh-Hyun Kwon	Electronics, Electrical Equip.	Technology	13	South Korea	Seoul South K
26	Hon Hai Precision Industry	27	135129	-4.3	4608.8	80436	-0.4	Terry Gou	Electronics, Electrical Equip.	Technology	25	Taiwan	New Taipei City, Tai
70	Hitachi	71	84558	1.2	2134.3	86742	48.8	Toshiaki Higashihara	Electronics, Electrical Equip.	Technology	79	Japan	Tokyo Ja
82	Huawei Investment & Holding	83	78511	24.9	5579.4	63837	-5.0	Ren Zhengfei	Network and Other Communications Equipment	Technology	129	China	Shenzhen C
104	Sony	105	70170	3.9	676.4	158519	-45.1	Kazuo Hirai	Electronics, Electrical Equip.	Technology	113	Japan	Tokyo Ja

10. Sorting Values

```
selected_rows = f500[f500['country']=='China']
selected_rows
```

	company	rank	revenues	revenue_change	profits	assets	profit_change	ceo	industry	sector	previous_rank	country	hq_locatice
1	State Grid	2	315199	-4.4	9571.3	489838	-6.2	Kou Wei	Utilities	Energy	2	China	Beijin Chir
2	Sinopec Group	3	267518	-9.1	1257.9	310726	-65.0	Wang Yupu	Petroleum Refining	Energy	4	China	Beijin Chir
3	China National Petroleum	4	262573	-12.3	1867.5	585619	-73.7	Zhang Jianhua	Petroleum Refining	Energy	3	China	Beijin Chir
21	Industrial & Commercial Bank of China	22	147675	-11.7	41883.9	3473238	-5.0	Gu Shu	Banks: Commercial and Savings	Financials	15	China	Beijin Chir
23	China State Construction Engineering	24	144505	3.1	2492.9	201269	10.7	Guan Qing	Engineering, Construction	Engineering & Construction	27	China	Beijin Chir
...
487	Xiamen C&D	488	22145	6.6	280.2	21729	15.6	Huang Wenzhou	Trading	Wholesalers	0	China	Xiamen Chir
489	China General Technology	490	22113	-20.1	413.6	20860	-20.8	Xu Xianping	Engineering, Construction	Engineering & Construction	383	China	Beijin Chir
493	Xiamen ITG Holding Group	494	21930	34.3	35.6	12161	-25.1	Xu Xiaoxi	Trading	Wholesalers	0	China	Xiamen Chir
494	Xinjiang Guanghui Industry Investment	495	21919	31.1	251.8	31957	49.9	Shang Jiqiang	Trading	Wholesalers	0	China	Urumc Chir
496	New China Life Insurance	497	21796	-13.3	743.9	100609	-45.6	Wan Feng	Insurance: Life, Health (stock)	Financials	427	China	Beijin Chir

109 rows × 17 columns

```
sorted_rows = selected_rows.sort_values('employees')
sorted_rows[['company', 'country', 'employees']].head()
```

	company	country	employees
204	Noble Group	China	1000
458	Yango Financial Holding	China	10234
438	China National Aviation Fuel Group	China	11739
128	Tewoo Group	China	17353
182	Amer International Group	China	17852

```
selected_rows = f500[f500['country'] == 'Japan']
sorted_row = selected_rows.sort_values('employees', ascending = False)
top_japanese_employer = sorted_row.iloc[0]['company']
top_japanese_employer
```

'Toyota Motor'

11. Using Loops with Pandas

```

avg_rev_by_country = {}
countries = f500['country'].unique()
for c in countries:
    selected_rows = f500[f500['country'] == c]
    mean = selected_rows['revenues'].mean()
    avg_rev_by_country[c] = mean
avg_rev_by_country

```

```

{'USA': 64218.371212121216,
'China': 55397.880733944956,
'Japan': 53164.03921568627,
'Germany': 63915.0,
'Netherlands': 61708.92857142857,
'Britain': 51588.708333333336,
'South Korea': 49725.6,
'Switzerland': 51353.57142857143,
'France': 55231.793103448275,
'Taiwan': 46364.666666666664,
'Singapore': 54454.333333333336,
'Italy': 51899.57142857143,
'Russia': 65247.75,
'Spain': 40600.666666666664,
'Brazil': 52024.57142857143,
'Mexico': 54987.5,
'Luxembourg': 56791.0,
'India': 39993.0,
'Malaysia': 49479.0,
'Thailand': 48719.0,
'Australia': 33688.71428571428,
'Belgium': 45905.0,
'Norway': 45873.0,
'Canada': 31848.0,
'Ireland': 32819.5,
'Indonesia': 36487.0,
'Denmark': 35464.0,
'Saudi Arabia': 35421.0,
'Sweden': 27963.666666666668,
'Finland': 26113.0,
'Venezuela': 24403.0,
'Turkey': 23456.0,
'U.A.E': 22799.0,
'Israel': 21903.0}

```

```

top_employer_by_country = {}
countries = f500['country'].unique()

for c in countries:
    selected_rows = f500[f500['country'] == c]
    sorted_rows = selected_rows.sort_values('employees', ascending=False)
    top_employer = sorted_rows.iloc[0]
    employer_name = top_employer['company']
    top_employer_by_country[c] = employer_name

countries

```

array(['USA', 'China', 'Japan', 'Germany', 'Netherlands', 'Britain',
 'South Korea', 'Switzerland', 'France', 'Taiwan', 'Singapore',
 'Italy', 'Russia', 'Spain', 'Brazil', 'Mexico', 'Luxembourg',
 'India', 'Malaysia', 'Thailand', 'Australia', 'Belgium', 'Norway',
 'Canada', 'Ireland', 'Indonesia', 'Denmark', 'Saudi Arabia',
 'Sweden', 'Finland', 'Venezuela', 'Turkey', 'U.A.E', 'Israel'],
 dtype=object)

12. Challenge: Calculating Return on Assets by Country

```

f500['roa'] = f500['profits']/f500['assets']

top_roa_by_sector = {}
for sector in f500['sector'].unique():
    is_sector = f500['sector'] == sector
    sector_companies = f500.loc[is_sector]
    top_company = sector_companies.sort_values('roa', ascending=False).iloc[0]
    company_name = top_company['company']
    top_roa_by_sector[sector] = company_name

top_roa_by_sector
{'Retailing': 'H & M Hennes & Mauritz',
 'Energy': 'National Grid',
 'Motor Vehicles & Parts': 'Subaru',
 'Financials': 'Berkshire Hathaway',
 'Technology': 'Accenture',
 'Wholesalers': 'McKesson',
 'Health Care': 'Gilead Sciences',
 'Telecommunications': 'KDDI',
 'Engineering & Construction': 'Pacific Construction Group',
 'Industrials': '3M',
 'Food & Drug Stores': 'Publix Super Markets',
 'Aerospace & Defense': 'Lockheed Martin',
 'Food, Beverages & Tobacco': 'Philip Morris International',
 'Household Products': 'Unilever',
 'Transportation': 'Delta Air Lines',
 'Materials': 'CRH',
 'Chemicals': 'LyondellBasell Industries',
 'Media': 'Disney',
 'Apparel': 'Nike',
 'Hotels, Restaurants & Leisure': 'McDonald's',
 'Business Services': 'Adecco Group'}

```

Data Cleaning Basics

1.	Reading CSV Files with Encodings.....	2
2.	Cleaning Column Names	2
3.	Cleaning Column Names Continued	4
4.	Converting String Columns to Numeric	4
5.	Converting Columns to Numeric Dtypes.....	5
6.	Renaming Columns	6
7.	Extracting Values From Strings.....	6
8.	Correcting Bad Values.....	8
9.	Dropping Missing Values	9
10.	Filling Missing Values.....	10
11.	Challenge: Clean a String Column	10

1. Reading CSV Files with Encodings

```
laptops = pd.read_csv("laptops.csv")
```

```
-----
          Traceback (most recent call last)
pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._convert_tokens()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._convert_with_dtype()

pandas/_libs/parsers.pyx in pandas._libs.parsers.TextReader._string_convert()

pandas/_libs/parsers.pyx in pandas._libs.parsers._string_box_utf8()

UnicodeDecodeError: 'utf-8' codec can't decode byte 0xe9 in position 4: invalid
continuation byte
```

2. Cleaning Column Names

```
laptops = pd.read_csv('laptops.csv', encoding='Latin-1')
```

```
laptops
```

	Manufacturer	Model Name	Category	Screen Size	Screen	CPU	RAM	Storage	GPU	Operating System
0	Apple	MacBook Pro	Ultrabook	13.3"	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8GB	128GB SSD	Intel Iris Plus Graphics 640	macOS
1	Apple	Macbook Air	Ultrabook	13.3"	1440x900	Intel Core i5 1.8GHz	8GB	128GB Flash Storage	Intel HD Graphics 6000	macOS
2	HP	250 G6	Notebook	15.6"	Full HD 1920x1080	Intel Core i5 2.5GHz	8GB	256GB SSD	Intel HD Graphics 620	No OS
3	Apple	MacBook Pro	Ultrabook	15.4"	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16GB	512GB SSD	AMD Radeon Pro 455	macOS
4	Apple	MacBook Pro	Ultrabook	13.3"	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8GB	256GB SSD	Intel Iris Plus Graphics 650	macOS
...
1298	Lenovo	Yoga 500-14ISK	2 in 1 Convertible	14.0"	IPS Panel Full HD / Touchscreen 1920x1080	Intel Core i7 6500U 2.5GHz	4GB	128GB SSD	Intel HD Graphics 520	Windows
1299	Lenovo	Yoga 900-13ISK	2 in 1 Convertible	13.3"	IPS Panel Quad HD+ / Touchscreen 3200x1800	Intel Core i7 6500U 2.5GHz	16GB	512GB SSD	Intel HD Graphics 520	Windows
1300	Lenovo	IdeaPad 100S-14IBR	Notebook	14.0"	1366x768	Intel Celeron Dual Core N3050 1.6GHz	2GB	64GB Flash Storage	Intel HD Graphics	Windows
1301	HP	15-AC110nv (i7-6500U/6GB/1TB/Radeon)	Notebook	15.6"	1366x768	Intel Core i7 6500U 2.5GHz	6GB	1TB HDD	AMD Radeon R5 M330	Windows
1302	Asus	X553SA-XX031T (N3050/4GB/500GB/W10)	Notebook	15.6"	1366x768	Intel Celeron Dual Core N3050 1.6GHz	4GB	500GB HDD	Intel HD Graphics	Windows

1303 rows × 13 columns

```
pd.DataFrame.info(laptops)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1303 entries, 0 to 1302
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Manufacturer    1303 non-null    object  
 1   Model Name      1303 non-null    object  
 2   Category         1303 non-null    object  
 3   Screen Size     1303 non-null    object  
 4   Screen           1303 non-null    object  
 5   CPU              1303 non-null    object  
 6   RAM              1303 non-null    object  
 7   Storage          1303 non-null    object  
 8   GPU              1303 non-null    object  
 9   Operating System 1303 non-null    object  
 10  Operating System Version 1133 non-null    object  
 11  Weight           1303 non-null    object  
 12  Price (Euros)   1303 non-null    object  
dtypes: object(13)
memory usage: 132.5+ KB
```

```
laptops.columns
```

```
Index(['Manufacturer', 'Model Name', 'Category', 'Screen Size', 'Screen',
       'CPU', 'RAM', 'Storage', 'GPU', 'Operating System',
       'Operating System Version', 'Weight', 'Price (Euros)'],
     dtype='object')
```

```
laptops_test= laptops.copy()
laptops_test.columns = ['A', 'B', 'C', 'D', 'E', 'F',
                       'G', 'H', 'I', 'J', 'K', 'L', 'M']
laptops_test.columns
```

```
Index(['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M'],
      dtype='object')
```

```
new_columns = []
for c in laptops.columns:
    clean_c = c.strip()
    new_columns.append(clean_c)
laptops.columns = new_columns
laptops.columns
```

```
Index(['Manufacturer', 'Model Name', 'Category', 'Screen Size', 'Screen',
       'CPU', 'RAM', 'Storage', 'GPU', 'Operating System',
       'Operating System Version', 'Weight', 'Price (Euros)'],
     dtype='object')
```

3. Cleaning Column Names Continued

```
def clean_col(col):
    col = col.strip()
    col = col.replace("Operating System", "os")
    col = col.replace("(", "")
    col = col.replace(")", "")
    col = col.lower()
    return col
new_columns = []
for c in laptops.columns:
    clean_c = clean_col(c)
    new_columns.append(clean_c)
laptops.columns = new_columns
laptops.columns
```

```
Index(['manufacturer', 'model name', 'category', 'screen size', 'screen',
       'cpu', 'ram', 'storage', 'gpu', 'operating system',
       'operating system version', 'weight', 'price euros'],
      dtype='object')
```

4. Converting String Columns to Numeric

```
laptops.iloc[:5, 2:5]
```

	category	screen size	screen
0	Ultrabook	13.3"	IPS Panel Retina Display 2560x1600
1	Ultrabook	13.3"	1440x900
2	Notebook	15.6"	Full HD 1920x1080
3	Ultrabook	15.4"	IPS Panel Retina Display 2880x1800
4	Ultrabook	13.3"	IPS Panel Retina Display 2560x1600

```
| laptops["screen size"].dtype
```

```
dtype('O')
```

```
laptops["screen size"].unique()
```

```
array(['13.3"', '15.6"', '15.4"', '14.0"', '12.0"', '11.6"', '17.3"',
       '10.1"', '13.5"', '12.5"', '13.0"', '18.4"', '13.9"', '12.3"',
       '17.0"', '15.0"', '14.1"', '11.3'], dtype=object)
```

```
unique_ram = laptops["ram"].unique()
unique_ram
```

```
array(['8GB', '16GB', '4GB', '2GB', '12GB', '6GB', '32GB', '24GB', '64G
B'],
      dtype=object)
```

```
laptops["screen size"] = laptops["screen size"].str.replace("'", '')  
laptops["screen size"].unique()
```

```
array(['13.3', '15.6', '15.4', '14.0', '12.0', '11.6', '17.3', '10.1',  
      '13.5', '12.5', '13.0', '18.4', '13.9', '12.3', '17.0', '15.0',  
      '14.1', '11.3'], dtype=object)
```

```
laptops["ram"] = laptops["ram"].str.replace('GB', '')  
unique_ram = laptops["ram"].unique()  
unique_ram
```

```
array(['8', '16', '4', '2', '12', '6', '32', '24', '64'], dtype=object)
```

5. Converting Columns to Numeric Dtypes

```
laptops["screen size"] = laptops["screen size"].astype(float)  
print(laptops["screen size"].dtype)  
laptops["screen size"].unique()
```

```
float64
```

```
array([13.3, 15.6, 15.4, 14. , 12. , 11.6, 17.3, 10.1, 13.5, 12.5, 13. ,  
      18.4, 13.9, 12.3, 17. , 15. , 14.1, 11.3])
```

```
laptops["ram"] = laptops["ram"].str.replace('GB', '')  
laptops["ram"] = laptops["ram"].astype(int)  
dtypes = laptops.dtypes  
dtypes
```

```
manufacturer          object  
model name            object  
category              object  
screen size           float64  
screen                object  
cpu                   object  
ram                  int32  
storage               object  
gpu                  object  
operating system     object  
operating system version object  
weight                object  
price euros           object  
dtype: object
```

6. Renaming Columns

```
laptops.rename({"screen size": "screen_size_inches"}, axis=1, inplace=True)
laptops.dtypes
```

```
manufacturer          object
model name           object
category             object
screen_size_inches   float64
screen               object
cpu                  object
ram                 int32
storage              object
gpu                 object
operating system    object
operating system version  object
weight              object
price euros         object
dtype: object
```

```
#laptops["ram"] = laptops["ram"].str.replace('GB', '').astype(int)
laptops.rename({"ram": "ram_gb"}, axis=1, inplace=True)
ram_gb_desc = laptops["ram_gb"].describe()
ram_gb_desc
```

```
count      1303.000000
mean       8.382195
std        5.084665
min        2.000000
25%       4.000000
50%       8.000000
75%       8.000000
max       64.000000
Name: ram_gb, dtype: float64
```

7. Extracting Values From Strings

```
laptops['gpu'].head()
```

```
0    Intel Iris Plus Graphics 640
1    Intel HD Graphics 6000
2    Intel HD Graphics 620
3    AMD Radeon Pro 455
4    Intel Iris Plus Graphics 650
Name: gpu, dtype: object
```

```
laptops['gpu'].head().str.split().str[0]
```

```
0    Intel
1    Intel
2    Intel
3    AMD
4    Intel
Name: gpu, dtype: object
```

```
laptops['cpu'].head()
```

```
0      Intel Core i5 2.3GHz
1      Intel Core i5 1.8GHz
2  Intel Core i5 7200U 2.5GHz
3      Intel Core i7 2.7GHz
4      Intel Core i5 3.1GHz
Name: cpu, dtype: object
```

```
laptops["gpu_manufacturer"] = (laptops["gpu"].str.split().str[0])
laptops["cpu_manufacturer"] = (laptops["cpu"].str.split().str[0])
cpu_manufacturer_counts = laptops["cpu_manufacturer"].value_counts()
```

```
laptops["gpu_manufacturer"]
```

```
0      Intel
1      Intel
2      Intel
3      AMD
4      Intel
...
1298    Intel
1299    Intel
1300    Intel
1301      AMD
1302    Intel
Name: gpu_manufacturer, Length: 1303, dtype: object
```

```
laptops["cpu_manufacturer"]
```

```
0      Intel
1      Intel
2      Intel
3      Intel
4      Intel
...
1298    Intel
1299    Intel
1300    Intel
1301    Intel
1302    Intel
Name: cpu_manufacturer, Length: 1303, dtype: object
```

```
cpu_manufacturer_counts
```

```
Intel      1240
AMD        62
Samsung     1
Name: cpu_manufacturer, dtype: int64
```

8. Correcting Bad Values

```
laptops["operating system"].value_counts()  
  
Windows      1125  
No OS        66  
Linux         62  
Chrome OS     27  
macOS         13  
Mac OS        8  
Android       2  
Name: operating system, dtype: int64  
  
mapping_dict = {  
    'Android': 'Android',  
    'Chrome OS': 'Chrome OS',  
    'Linux': 'Linux',  
    'Mac OS': 'macOS',  
    'No OS': 'No OS',  
    'Windows': 'Windows',  
    'macOS': 'macOS' }  
laptops["operating system"] = laptops["operating system"].map(mapping_dict)  
laptops["operating system"]  
  
0      macOS  
1      macOS  
2      No OS  
3      macOS  
4      macOS  
...  
1298    Windows  
1299    Windows  
1300    Windows  
1301    Windows  
1302    Windows  
Name: operating system, Length: 1303, dtype: object
```

9. Dropping Missing Values

```
laptops_no_null_rows = laptops.dropna(axis=0)
laptops_no_null_rows
```

	manufacturer	model name	category	screen_size_inches	screen	
5	Acer	Aspire 3	Notebook	15.6	1366x768	C 1
6	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	C 2
8	Asus	ZenBook UX430UN	Ultrabook	14.0	Full HD 1920x1080	C 1
9	Acer	Swift 3	Ultrabook	14.0	IPS Panel Full HD 1920x1080	C 1
13	Dell	Inspiron 3567	Notebook	15.6	Full HD 1920x1080	C 1
...
1298	Lenovo	Yoga 500-14ISK	2 in 1 Convertible	14.0	IPS Panel Full HD / Touchscreen 1920x1080	C 2
1299	Lenovo	Yoga 900-13ISK	2 in 1 Convertible	13.3	IPS Panel Quad HD+ / Touchscreen 3200x1800	C 2
1300	Lenovo	IdeaPad 100S-14IBR	Notebook	14.0	1366x768	C 1

```
laptops_no_null_cols = laptops.dropna(axis=1)
laptops_no_null_cols
```

	manufacturer	model name	category	screen_size_inches	screen	
0	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	C 2
1	Apple	Macbook Air	Ultrabook	13.3	1440x900	C 1
2	HP	250 G6	Notebook	15.6	Full HD 1920x1080	C 2
3	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	C 2
4	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	C 3
...
1298	Lenovo	Yoga 500-14ISK	2 in 1 Convertible	14.0	IPS Panel Full HD / Touchscreen 1920x1080	C 2
1299	Lenovo	Yoga 900-13ISK	2 in 1 Convertible	13.3	IPS Panel Quad HD+ / Touchscreen 3200x1800	C 2
1300	Lenovo	IdeaPad 100S-14IBR	Notebook	14.0	1366x768	C 1

10. Filling Missing Values

```
value_counts_before = laptops.loc[laptops["operating system version"].isnull(), "operating system"].value_counts()  
value_counts_before
```

```
No OS      66  
Linux     62  
Chrome OS 27  
macOS     13  
Android    2  
Name: operating system, dtype: int64
```

```
| laptops.loc[laptops["operating system"] == "macOS", "operating system version"] = "X"  
| laptops.loc[laptops["operating system"] == "No OS", "operating system version"] = "Version Unknown"  
| value_counts_after = laptops.loc[laptops["operating system version"].isnull(), "operating system"].value_counts()  
| value_counts_after
```

```
Linux      62  
Chrome OS  27  
macOS     13  
Android    2  
Name: operating system, dtype: int64
```

11. Challenge: Clean a String Column

```
laptops["weight"] = laptops["weight"].str.replace("kgs","");
laptops["weight"] = laptops["weight"].str.replace("kg","");
laptops["weight"] = laptops["weight"].astype(float)  
laptops.rename({"weight": "weight_kg"}, axis=1, inplace=True)  
laptops.to_csv('laptops_cleaned.csv', index=False)
```

```
laptops_cleaned = pd.read_csv('laptops_cleaned.csv', encoding='Latin-1')  
laptops_cleaned
```

	manufacturer	model name	category	screen_size_inches	screen	cpu	ram_gb	storage	gpu	operating system	operating system version	weight
0	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 2.3GHz	8	128GB SSD	Intel Iris Plus Graphics 640	macOS	NaN	1
1	Apple	Macbook Air	Ultrabook	13.3	1440x900	Intel Core i5 1.8GHz	8	128GB Flash Storage	Intel HD Graphics 6000	macOS	NaN	1
2	HP	250 G6	Notebook	15.6	Full HD 1920x1080	Intel Core i5 7200U 2.5GHz	8	256GB SSD	Intel HD Graphics 620	No OS	Version Unknown	1
3	Apple	MacBook Pro	Ultrabook	15.4	IPS Panel Retina Display 2880x1800	Intel Core i7 2.7GHz	16	512GB SSD	AMD Radeon Pro 455	macOS	NaN	1
4	Apple	MacBook Pro	Ultrabook	13.3	IPS Panel Retina Display 2560x1600	Intel Core i5 3.1GHz	8	256GB SSD	Intel Iris Plus Graphics 650	macOS	NaN	1
...
1298	Lenovo	Yoga 500-14ISK	2 in 1 Convertible	14.0	IPS Panel Full HD / Touchscreen 1920x1080	Intel Core i7 6500U 2.5GHz	4	128GB SSD	Intel HD Graphics 520	Windows	10	1
1299	Lenovo	Yoga 900-13ISK	2 in 1 Convertible	13.3	IPS Panel Quad HD+ / Touchscreen 3200x1800	Intel Core i7 6500U 2.5GHz	16	512GB SSD	Intel HD Graphics 520	Windows	10	1

LINE GRAPHS AND TIME SERIES

1. Introduction.....	2
2. Graphs.....	3
3. Line Graphs.....	4
4. Matplotlib.....	6
5. Customizing a Graph	7
6. WHO Time Series Data	9
7. Types of Growth	10
8. Types of Change	15
9. Comparing Line Graphs.....	16

1. Introduction

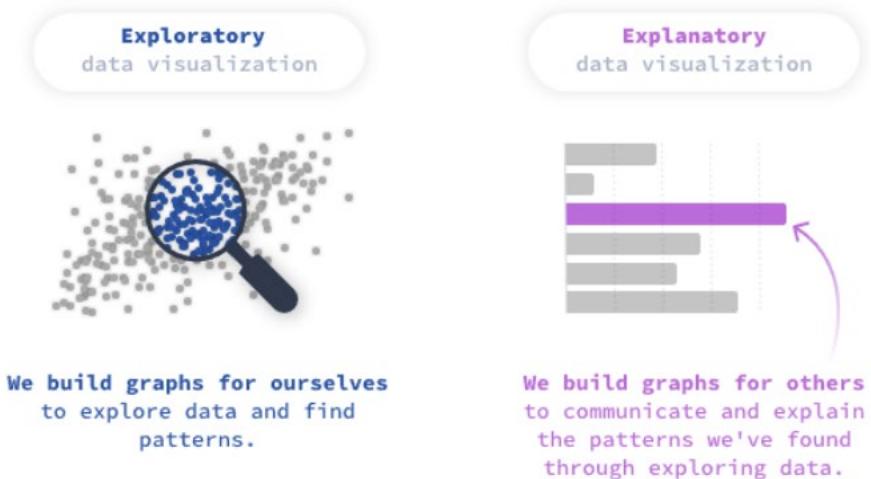
Thông thường, chúng ta khám phá dữ liệu bằng 2 cách sau:

- Statistical methods (measuring averages, measuring variability, etc.)
- Data visualization (transforming data into a visual form)

Một trong những mục tiêu trọng tâm của data visualization là giúp chúng ta khám phá data.

Và cái mục tiêu trọng tâm khác là giúp chúng ta phân tích và giải thích kết quả mà chúng ta tìm được thông qua khám phá data. Ta có 2 kiểu data visualization:

- Exploratory data visualization: ta xây dựng đồ thị cho bản thân ta để tìm hiểu data và tìm kiếm mô hình.
- Explanatory data visualization: ta xây dựng đồ thị cho những người khác để cùng bàn luận và giải thích mô hình ta nhận ra thông qua khám phá data.



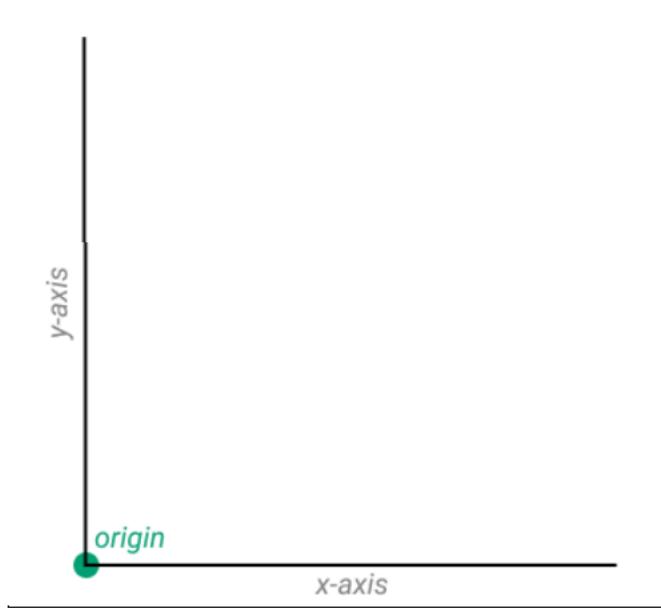
Ta sẽ được học theo trình tự sau:

- How to visualize time series data with line plots.
- What are correlations and how to visualize them with scatter plots.
- How to visualize frequency distributions with bar plots and histograms.

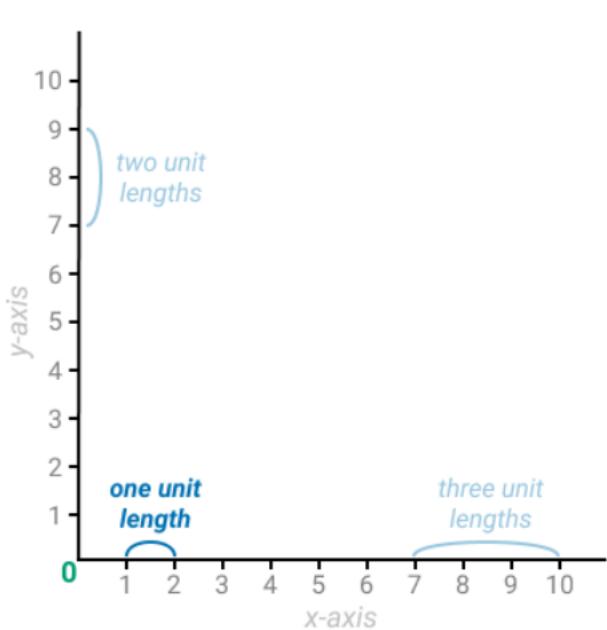
- d. How to speed up our exploratory data visualization workflow with the pandas library.
 - e. How to visualize multiple variables using Seaborn's relational plots.
 - f.
2. Graphs

Trước khi ta đến với Matplotlib và bắt đầu khám phá dataset, ta sẽ giới thiệu sơ qua graphs – nó là cái gì và sao để xây dựng nó một cách toán học.

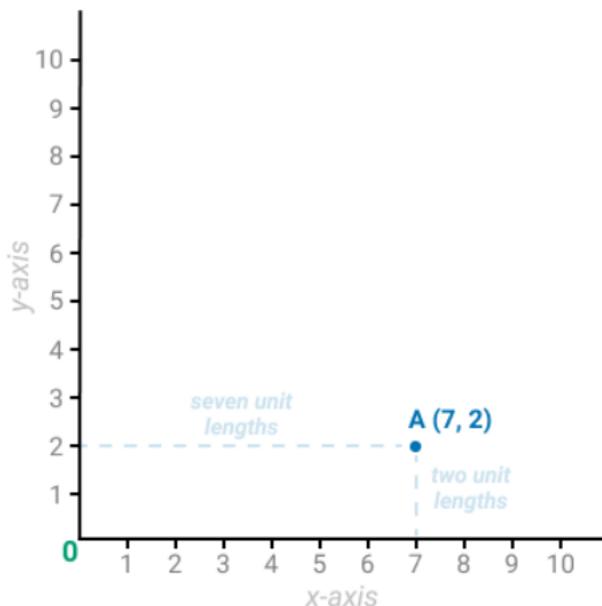
Ta có thể tạo 1 graph bằng cách vẽ 2 đường thẳng tại góc phải của mỗi đường. Mỗi đường được gọi là axis – đường ngang ở dưới là x-axis và đường dọc bên trái là y-axis. Điểm giao của 2 đường gọi là origin.



Mỗi cột cái độ chia riêng.



Độ chia giúp chúng ta định vị khá chính xác bất kỳ điểm nào trên graph. Điểm A(7,2) với lần lượt x-axis là 7 và y-axis là 2.



3. Line Graphs

Dưới đây là bảng số liệu cho thấy số người mắc COVID 19 khắp thế giới trong vòng 7 tháng đầu năm 2020.

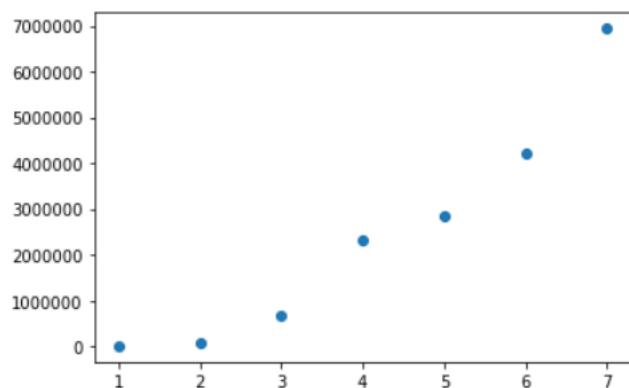
Month Number	New Reported Cases
1	9926
2	76246
3	681488
4	2336640
5	2835147
6	4226655
7	6942042

Source: <https://covid19.who.int/>

Mỗi hàng cho thấy một cặp dữ liệu kết nối với nhau :

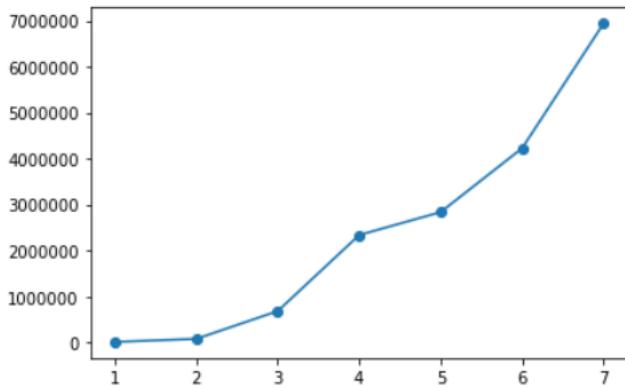
- a. Thứ tự tháng
- b. Số ca mắc trong tháng tương ứng.

Khi ta có các cặp dữ liệu, ta có thể định vị chúng trên graph bằng cách sử dụng 2 số liệu như 2 điểm.



Số ca mắc ứng với mỗi tháng.

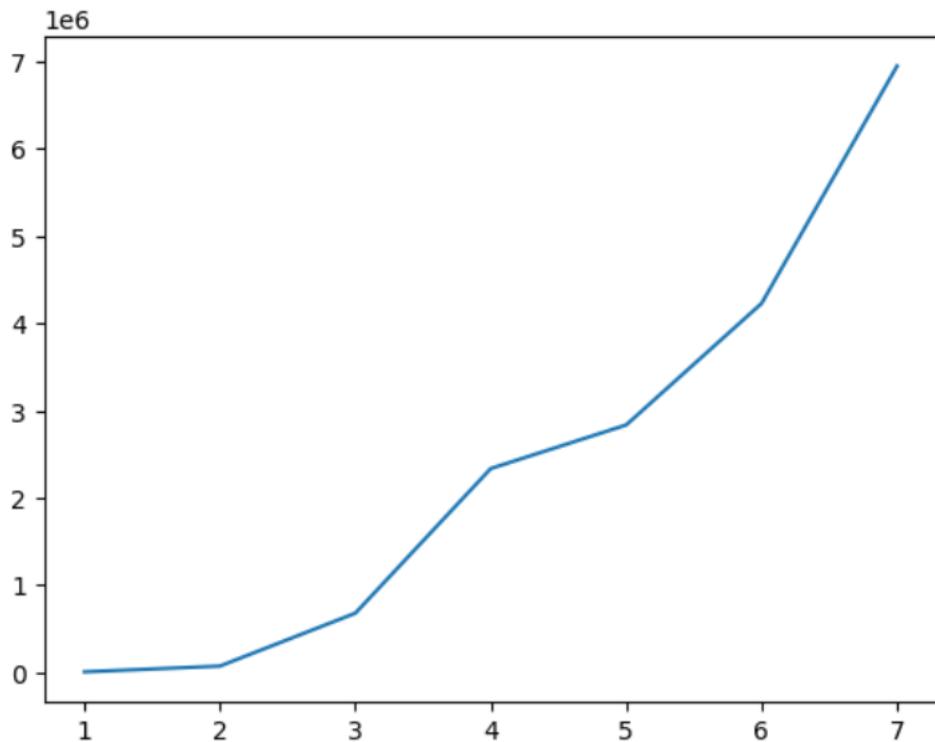
Khi ta vẽ đồ thị về cái gì đó thay đổi qua thời gian, ta nối tất cả các điểm với 1 đường.



4. Matplotlib

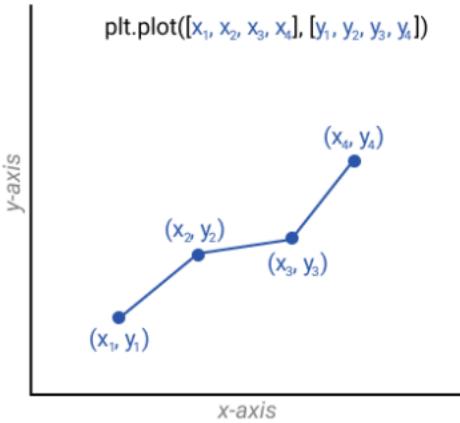
Ta có thể vẽ graph bằng Matplotlib, một thư viện Python được thiết kế chuyên để tạo ra visualizations.

```
month_number = [1, 2, 3, 4, 5, 6, 7]
new_cases = [9926, 76246, 681488, 2336640
             ,2835147, 4226655, 6942042]
plt.plot(month_number, new_cases)
plt.show()
```



Plt.plot() có chức năng tạo ra 1 line graph từ 2 mảng dữ liệu có cùng độ lớn.

Trong plt.plot(month_number, new_case), month_number đứng trước, sau đó là new_case. Mảng được cho trước là x-axis và mảng thứ hai là y-axis.



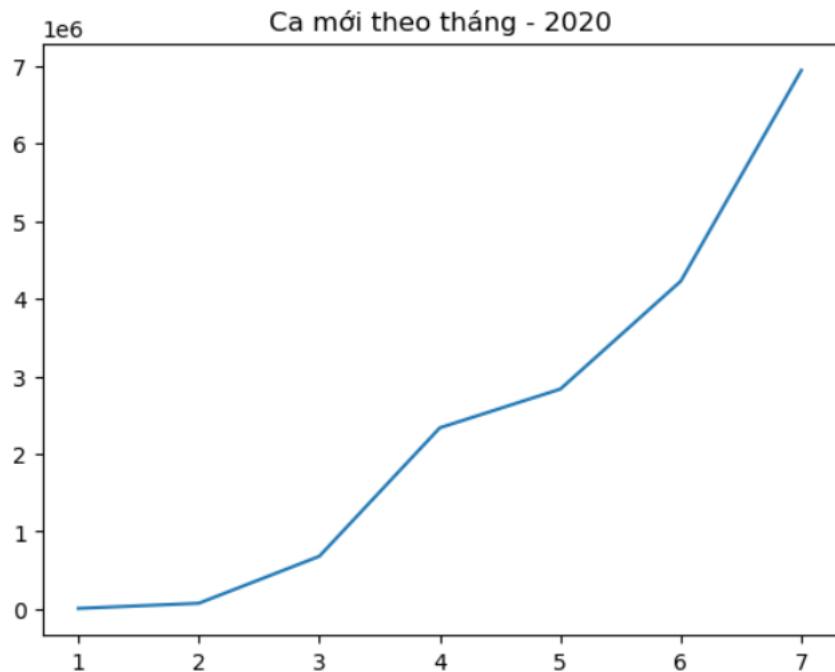
5. Customizing a Graph

Trên cùng bên trái của graph, ta có thể thấy “le6”, nếu ta muốn loại bỏ dòng này, ta có thể dùng plt.ticklabel_format(axis,style).

Điều tiếp theo là sử dụng hàm plt.title() để thêm tiêu đề cho graph.

```
import matplotlib.pyplot as plt

month_number = [1, 2, 3, 4, 5, 6, 7]
new_cases = [9926, 76246, 681488, 2336640
             ,2835147, 4226655, 6942042]
plt.plot(month_number, new_cases)
plt.title("Ca mới theo tháng - 2020")
plt.show()
```



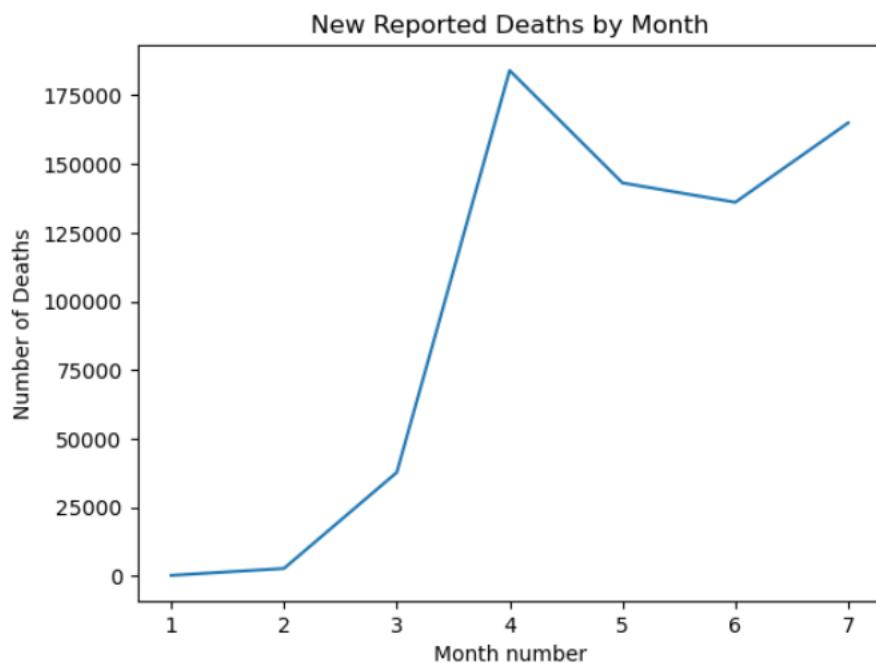
Để thêm label cho 2 cột x,y ta dùng plt.xlabel() và plt.ylabel().

```

import matplotlib.pyplot as plt

month_number = [1, 2, 3, 4, 5, 6, 7]
new_deaths = [213, 2729, 37718, 184064, 143119, 136073, 165003]
plt.plot(month_number, new_deaths)
plt.title("New Reported Deaths by Month")
plt.xlabel("Month number")
plt.ylabel("Number of Deaths")
plt.show()

```



6. WHO Time Series Data

```

import pandas as pd
who_time_series = pd.read_csv('WHO_time_series.csv')
print(who_time_series.head())

```

	Date_reported	Country	New_cases	Cumulative_cases	New_deaths
0	2020-01-04	China	1	1	0
1	2020-01-05	China	0	1	0
2	2020-01-06	China	3	4	0
3	2020-01-07	China	0	4	0
4	2020-01-08	China	0	4	0

```

import pandas as pd
who_time_series = pd.read_csv('WHO_time_series.csv')
who_time_series['Date_reported'] = pd.to_datetime(who_time_series['Date_reported'])
print(who_time_series.head(5))
print(who_time_series.tail(5))
print(who_time_series.info())

      Date_reported Country  New_cases  Cumulative_cases  New_deaths \
0   2020-01-04    China        1              1            0
1   2020-01-05    China        0              1            0
2   2020-01-06    China        3              4            0
3   2020-01-07    China        0              4            0
4   2020-01-08    China        0              4            0

      Cumulative_deaths
0                      0
1                      0
2                      0
3                      0
4                      0

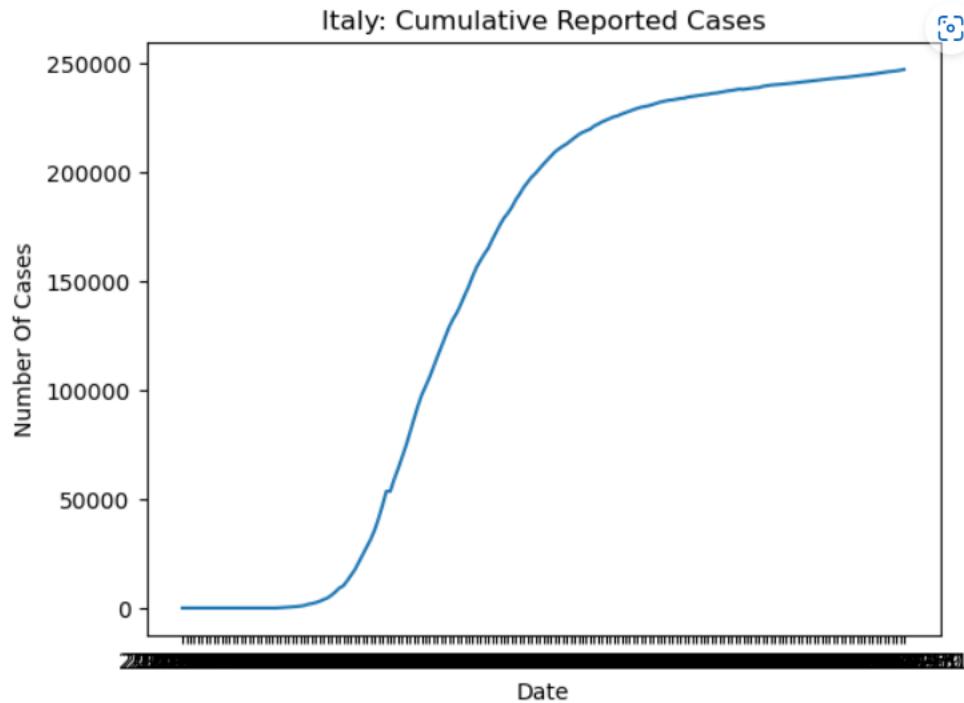
      Date_reported          Country  New_cases \
31871  2020-07-31       Panama      1046
31872  2020-07-31  Timor-Leste        0
31873  2020-07-31  Guatemala     1221
31874  2020-07-31  Saint Vincent and the Grenadines        0
31875  2020-07-31  Democratic Republic of the Congo       79

      Cumulative_cases  New_deaths  Cumulative_deaths
31871           63269        25            1374
31872             24         0              0
31873           48826        32            1867
31874             52         0              0
31875           9009         2            214
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 31876 entries, 0 to 31875
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype  
---  -- 
 0   Date_reported    31876 non-null   datetime64[ns]
 1   Country          31876 non-null   object  
 2   New_cases         31876 non-null   int64  
 3   Cumulative_cases 31876 non-null   int64  
 4   New_deaths        31876 non-null   int64  
 5   Cumulative_deaths 31876 non-null   int64  
dtypes: datetime64[ns](1), int64(4), object(1)
memory usage: 1.5+ MB
None

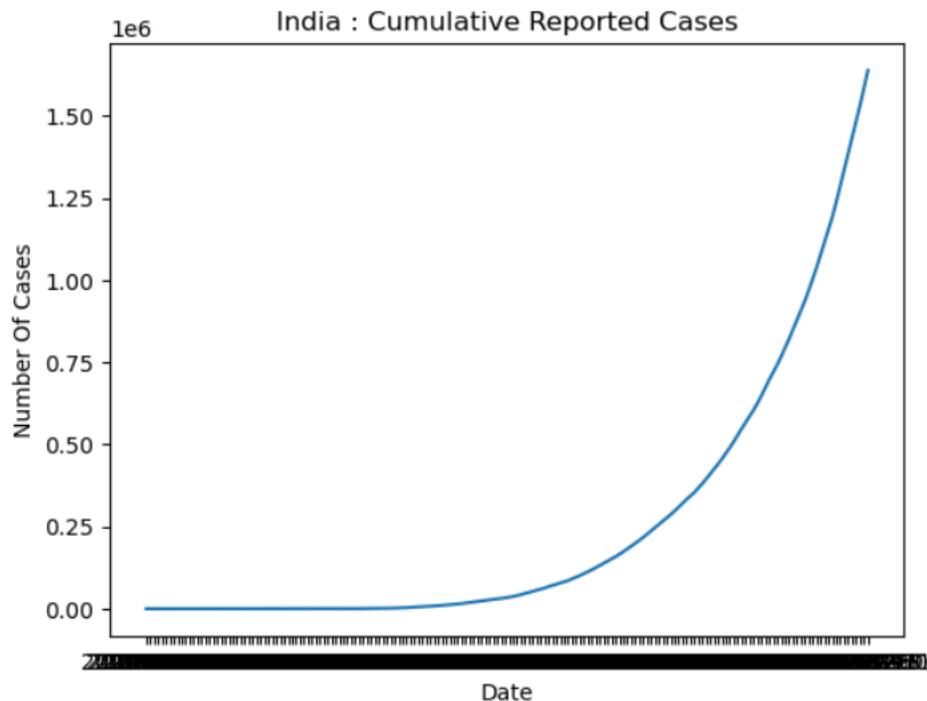
```

7. Types of Growth

```
import pandas as pd
who_time_series = pd.read_csv('WHO_time_series.csv')
italy = who_time_series[who_time_series['Country'] == 'Italy']
plt.plot(italy['Date_reported'], italy['Cumulative_cases'])
plt.title('Italy: Cumulative Reported Cases')
plt.xlabel('Date')
plt.ylabel('Number Of Cases')
plt.show()
```



```
import pandas as pd
who_time_series = pd.read_csv('WHO_time_series.csv')
india = who_time_series[who_time_series['Country'] == 'India']
plt.plot(india['Date_reported'], india['Cumulative_cases'])
plt.title('India : Cumulative Reported Cases')
plt.xlabel('Date')
plt.ylabel('Number Of Cases')
plt.show()
```



```

import pandas as pd
who_time_series = pd.read_csv('WHO_time_series.csv')
poland = who_time_series[who_time_series['Country'] == 'Poland']
plt.plot(poland['Date_reported'], poland['Cumulative_cases'])
plt.title('Poland: Cumulative Reported Cases')
plt.xlabel('Date')
plt.ylabel('Number Of Cases')
plt.show()

```



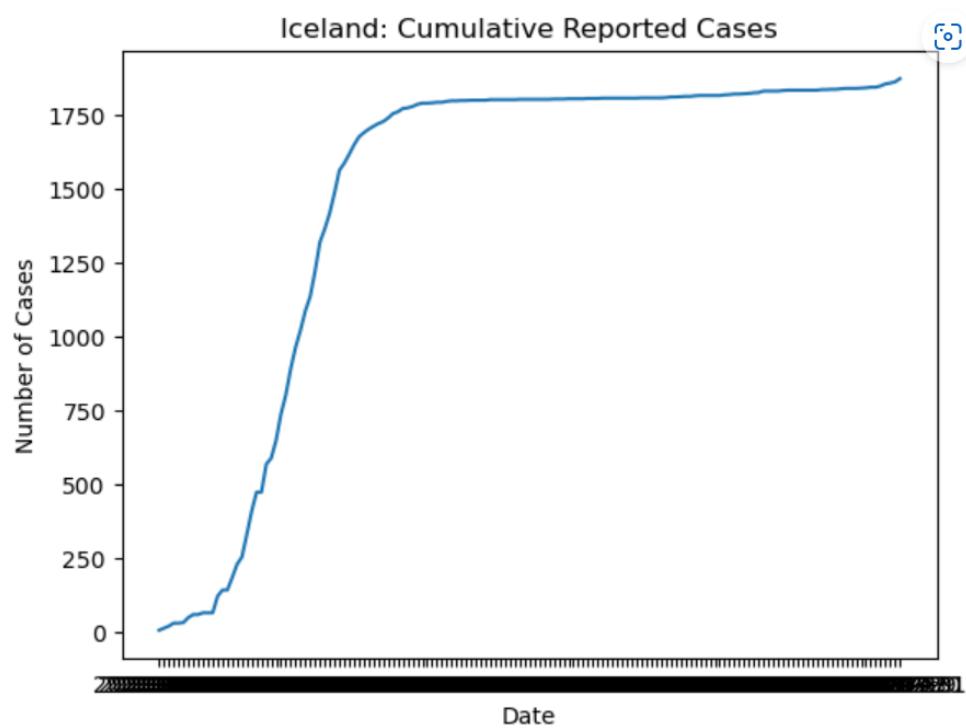
```

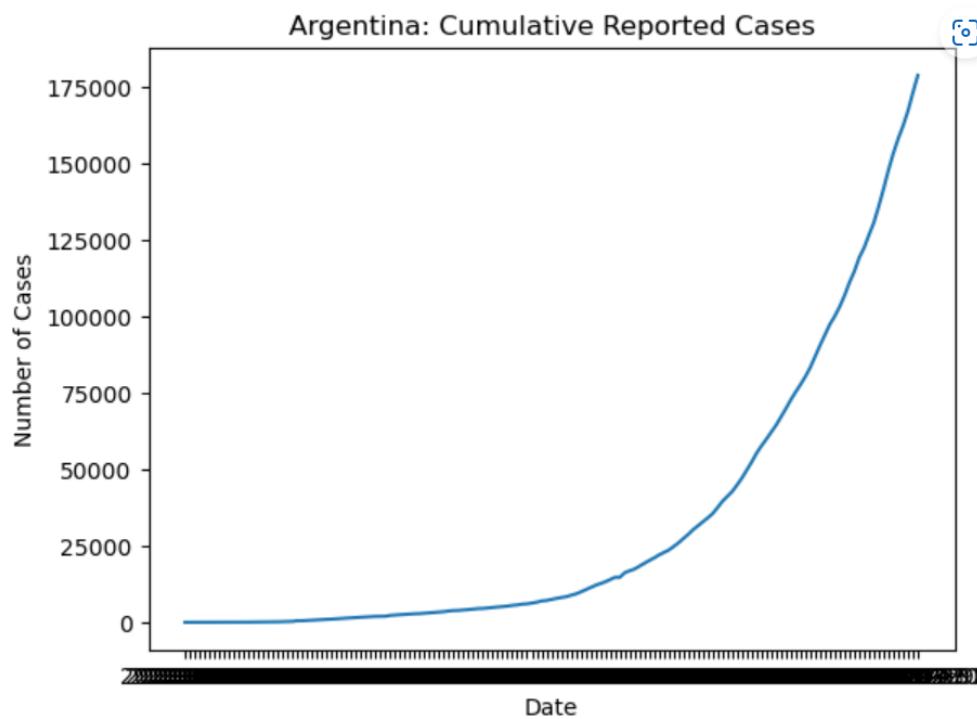
def plot_cumulative_cases(country_name):
    country = who_time_series[who_time_series['Country'] == country_name]
    plt.plot(country['Date_reported'], country['Cumulative_cases'])
    plt.title('{}: Cumulative Reported Cases'.format(country_name))
    plt.xlabel('Date')
    plt.ylabel('Number of Cases')
    plt.show()

plot_cumulative_cases('Brazil')
plot_cumulative_cases('Iceland')
plot_cumulative_cases('Argentina')

brazil = 'exponential'
iceland = 'logarithmic'
argentina = 'exponential'

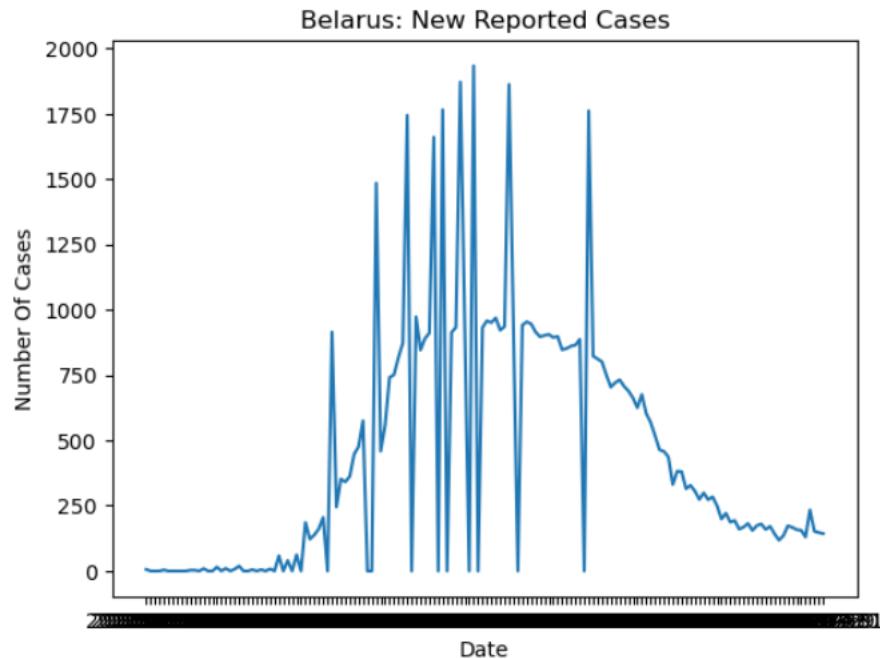
```





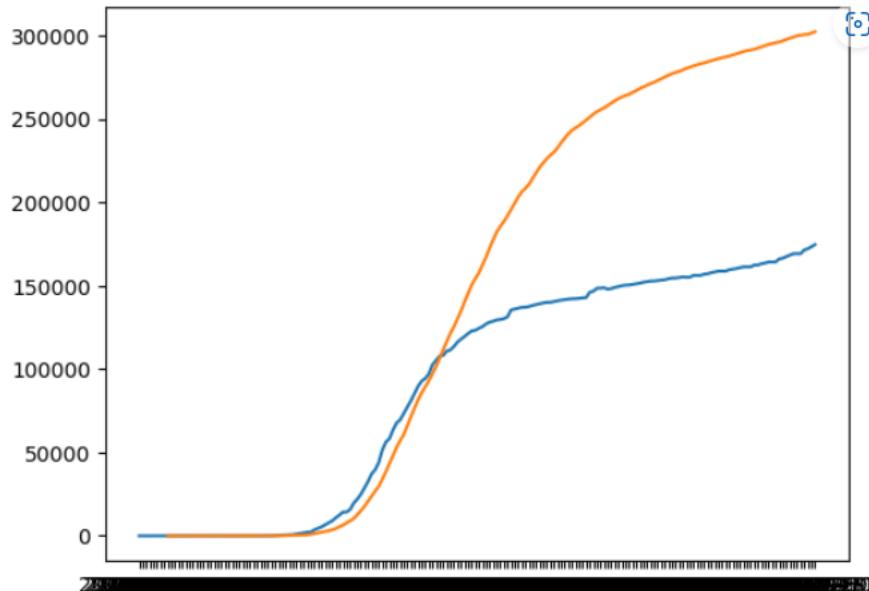
8. Types of Change

```
import pandas as pd
who_time_series = pd.read_csv('WHO_time_series.csv')
belarus = who_time_series[who_time_series['Country'] == 'Belarus']
plt.plot(belarus['Date_reported'], belarus['New_cases'])
plt.title('Belarus: New Reported Cases')
plt.xlabel('Date')
plt.ylabel('Number Of Cases')
plt.show()
```

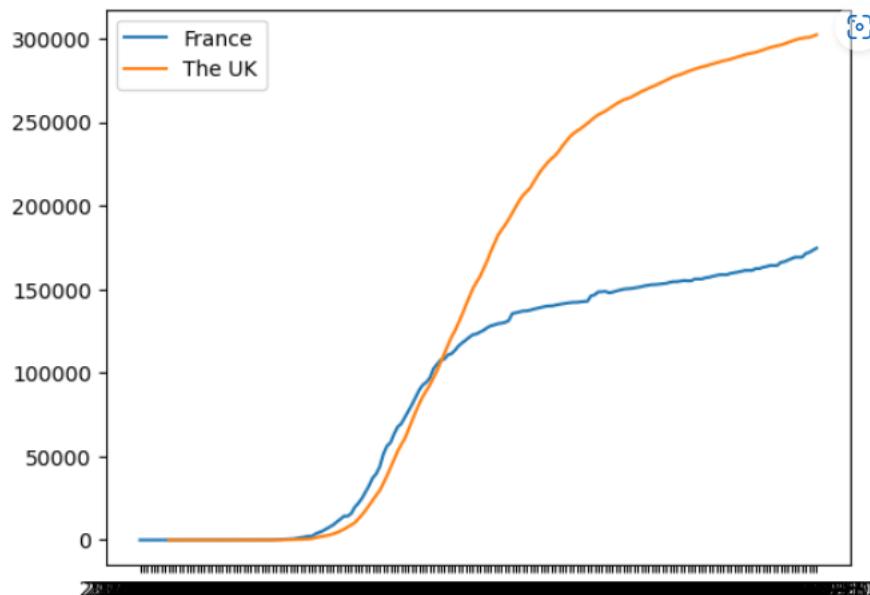


9. Comparing Line Graphs

```
who_time_series = pd.read_csv('WHO_time_series.csv')
france = who_time_series[who_time_series['Country'] == 'France']
uk = who_time_series[who_time_series['Country'] == 'The United Kingdom']
plt.plot(france['Date_reported'], france['Cumulative_cases'])
plt.plot(uk['Date_reported'], uk['Cumulative_cases'])
plt.show()
```



```
plt.plot(france['Date_reported'], france['Cumulative_cases'], label='France')
plt.plot(uk['Date_reported'], uk['Cumulative_cases'], label='The UK')
plt.legend()
plt.show()
```

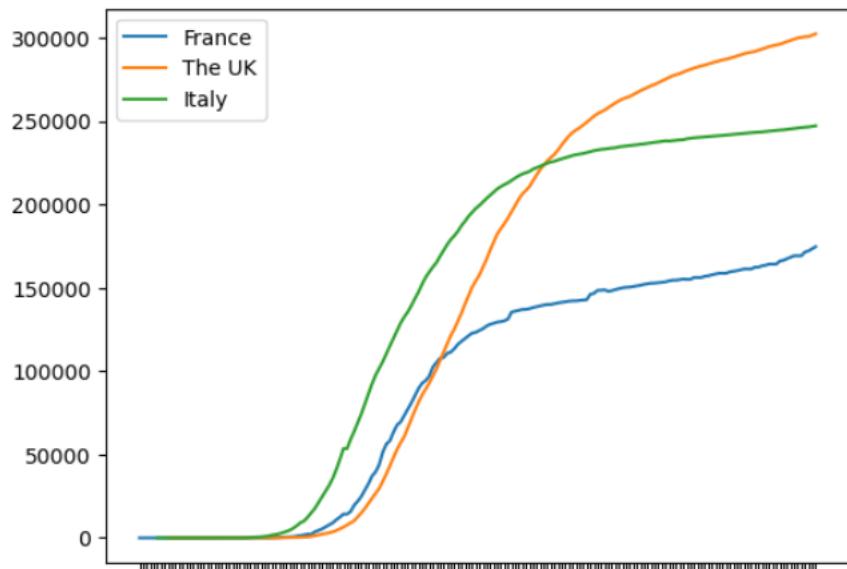


```

france = who_time_series[who_time_series['Country'] == 'France']
uk = who_time_series[who_time_series['Country'] == 'The United Kingdom']
italy = who_time_series[who_time_series['Country'] == 'Italy']
plt.plot(france['Date_reported'], france['Cumulative_cases'], label='France')
plt.plot(uk['Date_reported'], uk['Cumulative_cases'], label='The UK')
plt.plot(italy['Date_reported'], italy['Cumulative_cases'], label='Italy')
plt.legend()
plt.show()

#greatest_july - 'The UK'
#lowest_july - 'France'
#increase_march - 'Italy'

```



Scatter Plots and Correlations

1. Bike Sharing Time Series	2
2. Exploring Data	2
3. Seasonal Trends.....	4
4. Scatter Plots.....	5
5. Correlation.....	7
6. Pearson Correlation Coefficient	8
7. Measuring Pearson's R	8
8. Categorical Columns	10

1. Bike Sharing Time Series

Dùng pandas đọc dataset số lượng xe đạp được thuê

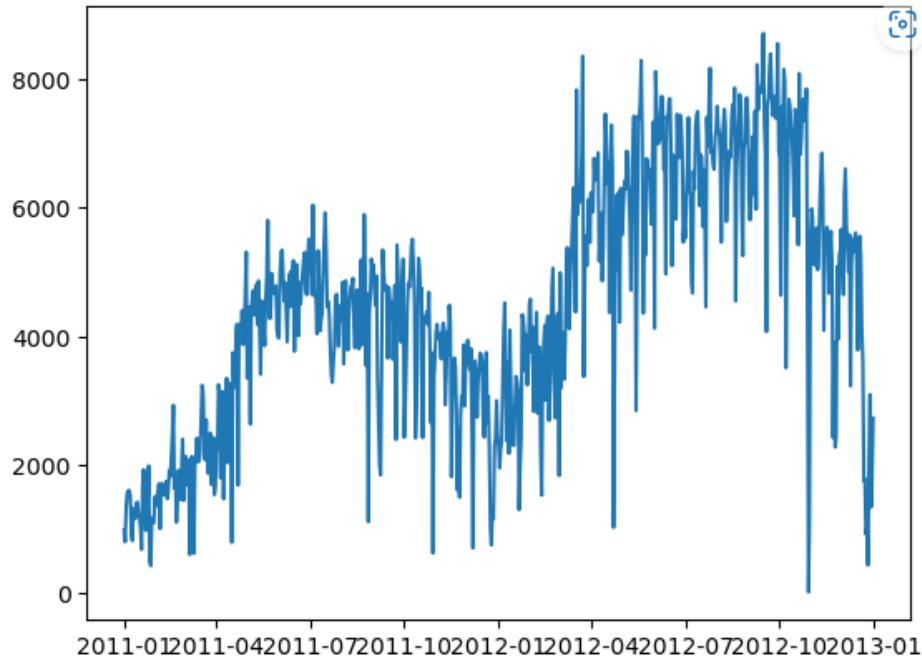
```
import pandas as pd
bike_sharing = pd.read_csv('day.csv')
bike_sharing.head()
bike_sharing.tail()
bike_sharing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype  
---  --  
 0   instant     731 non-null    int64  
 1   dteday      731 non-null    object 
 2   season      731 non-null    int64  
 3   yr          731 non-null    int64  
 4   mnth        731 non-null    int64  
 5   holiday     731 non-null    int64  
 6   weekday     731 non-null    int64  
 7   workingday  731 non-null    int64  
 8   weathersit  731 non-null    int64  
 9   temp         731 non-null    float64 
 10  atemp        731 non-null    float64 
 11  hum          731 non-null    float64 
 12  windspeed   731 non-null    float64 
 13  casual       731 non-null    int64  
 14  registered   731 non-null    int64  
 15  cnt          731 non-null    int64  
dtypes: float64(4), int64(11), object(1)
memory usage: 91.5+ KB
```

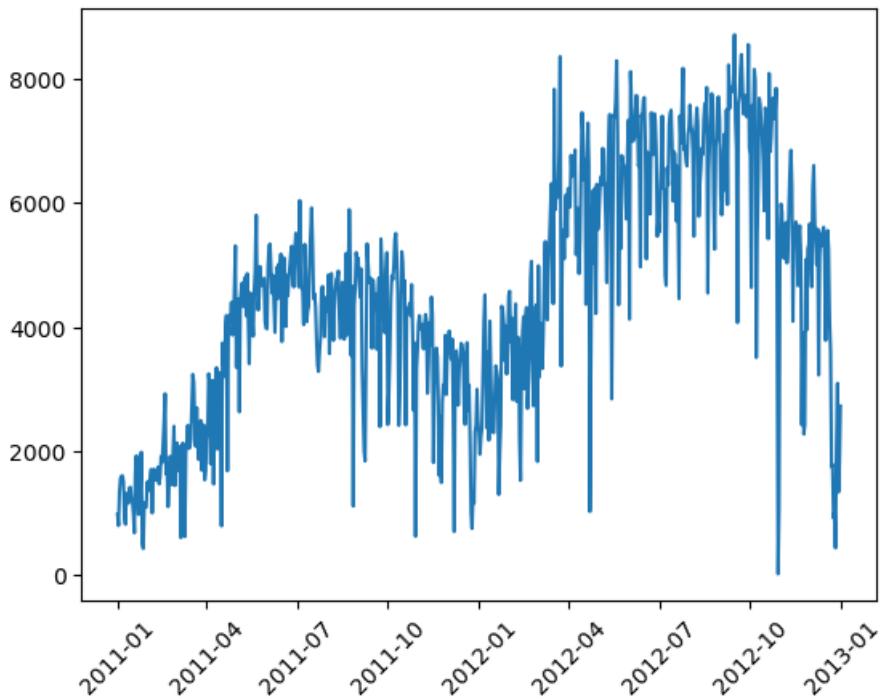
2. Exploring Data

Visualization dữ liệu vừa rồi

```
import matplotlib.pyplot as plt  
plt.plot(bike_sharing['dteday'], bike_sharing['cnt'])  
plt.show()
```



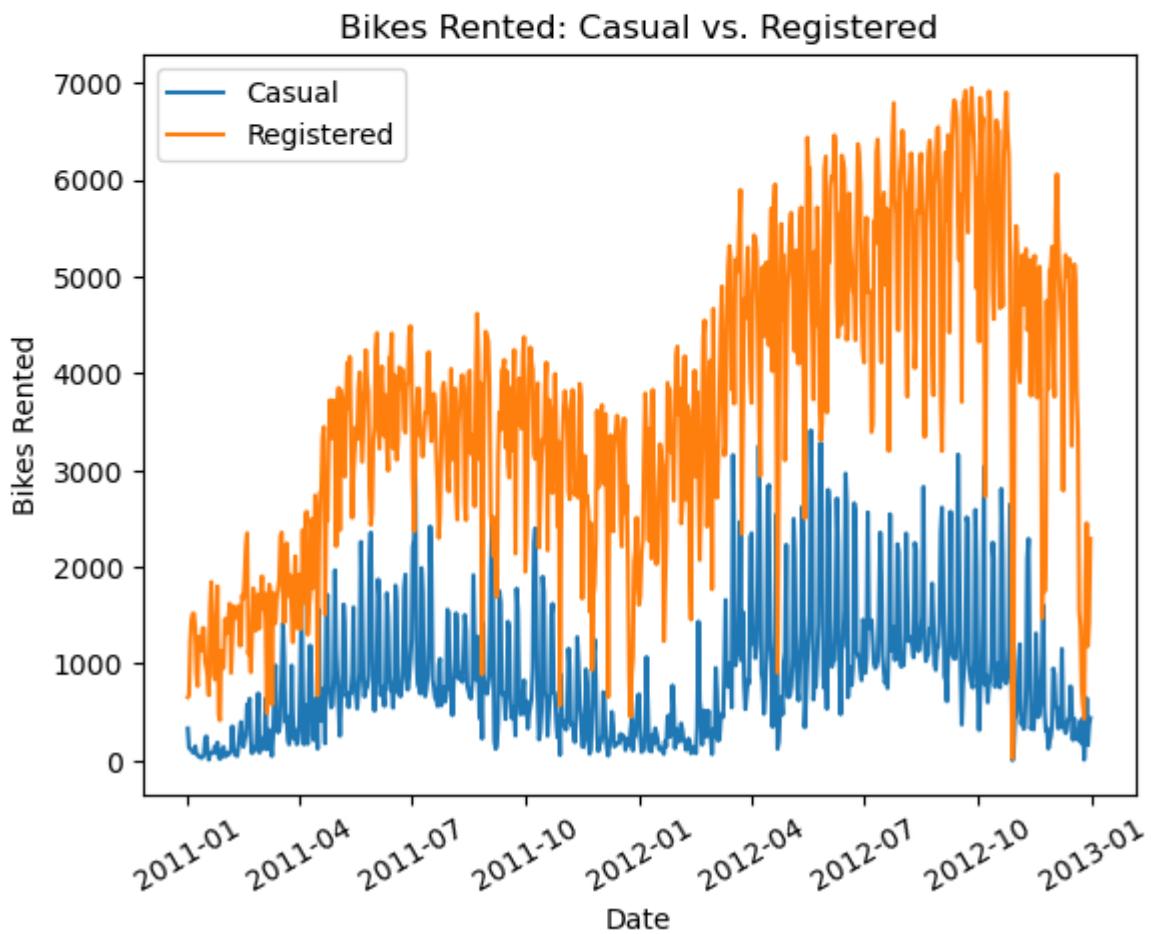
```
import matplotlib.pyplot as plt  
plt.plot(bike_sharing['dteday'], bike_sharing['cnt'])  
plt.xticks(rotation=45)  
plt.show()
```



```

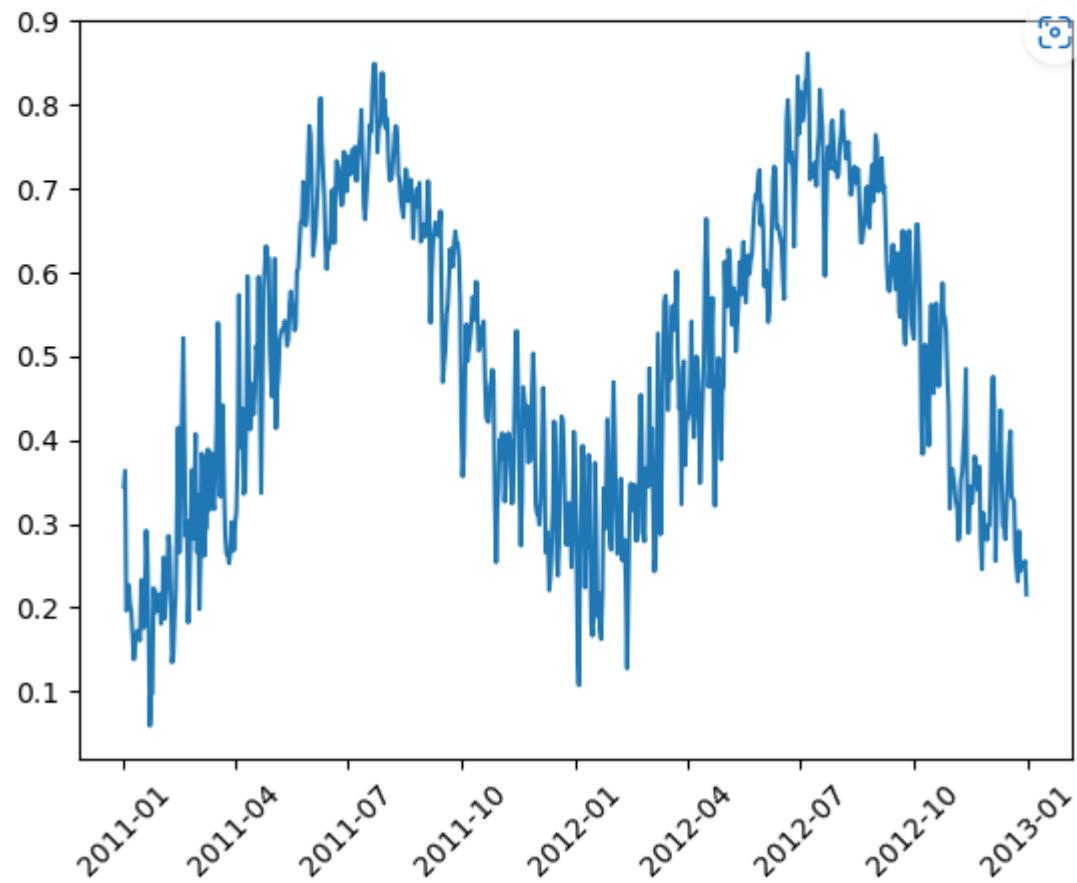
import pandas as pd
import matplotlib.pyplot as plt
bike_sharing = pd.read_csv('day.csv')
bike_sharing['dteday'] = pd.to_datetime(bike_sharing['dteday'])
plt.plot(bike_sharing['dteday'], bike_sharing['casual'],
label='Casual')
plt.plot(bike_sharing['dteday'], bike_sharing['registered'],
label='Registered')
plt.xticks(rotation=30)
plt.ylabel('Bikes Rented')
plt.xlabel('Date')
plt.title('Bikes Rented: Casual vs. Registered')
plt.legend()
plt.show()

```



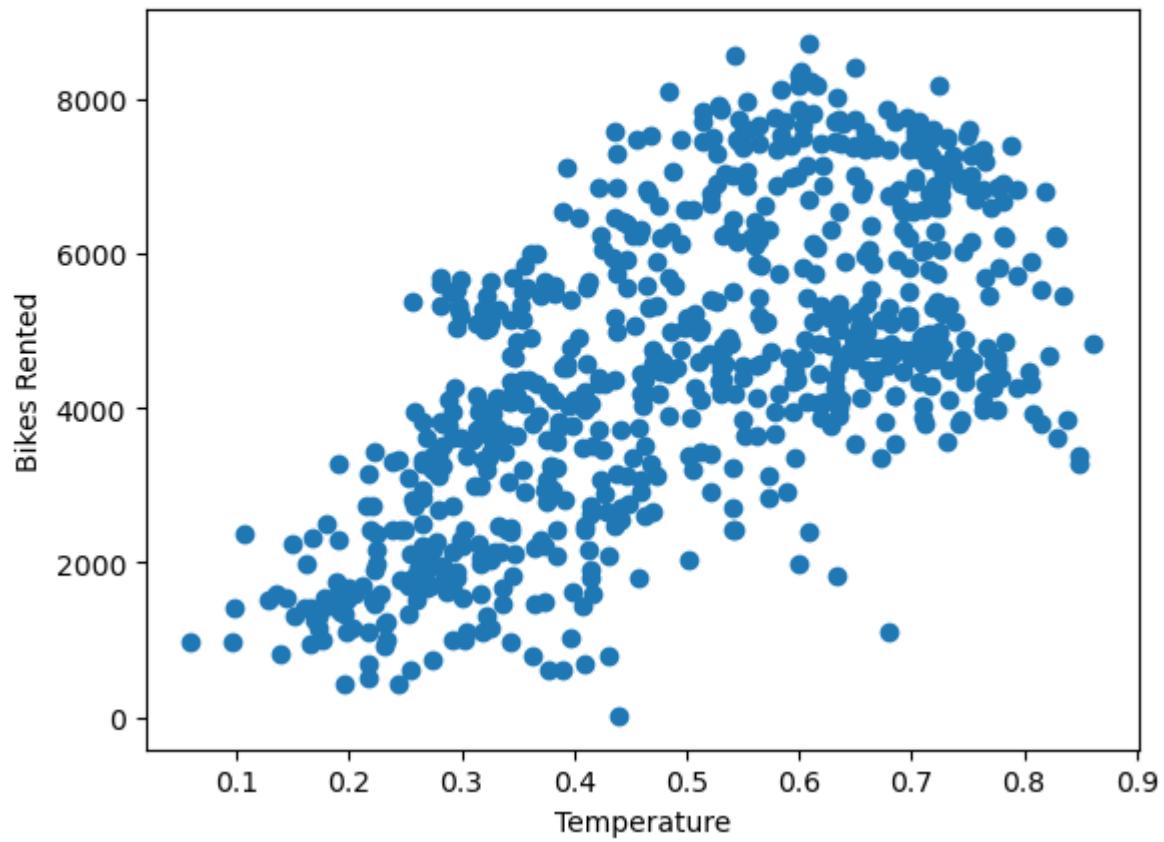
3. Seasonal Trends

```
import pandas as pd
import matplotlib.pyplot as plt
bike_sharing = pd.read_csv('day.csv')
bike_sharing['dteday'] = pd.to_datetime(bike_sharing['dteday'])
plt.plot(bike_sharing['dteday'], bike_sharing['temp'])
plt.xticks(rotation=45)
plt.show()
```

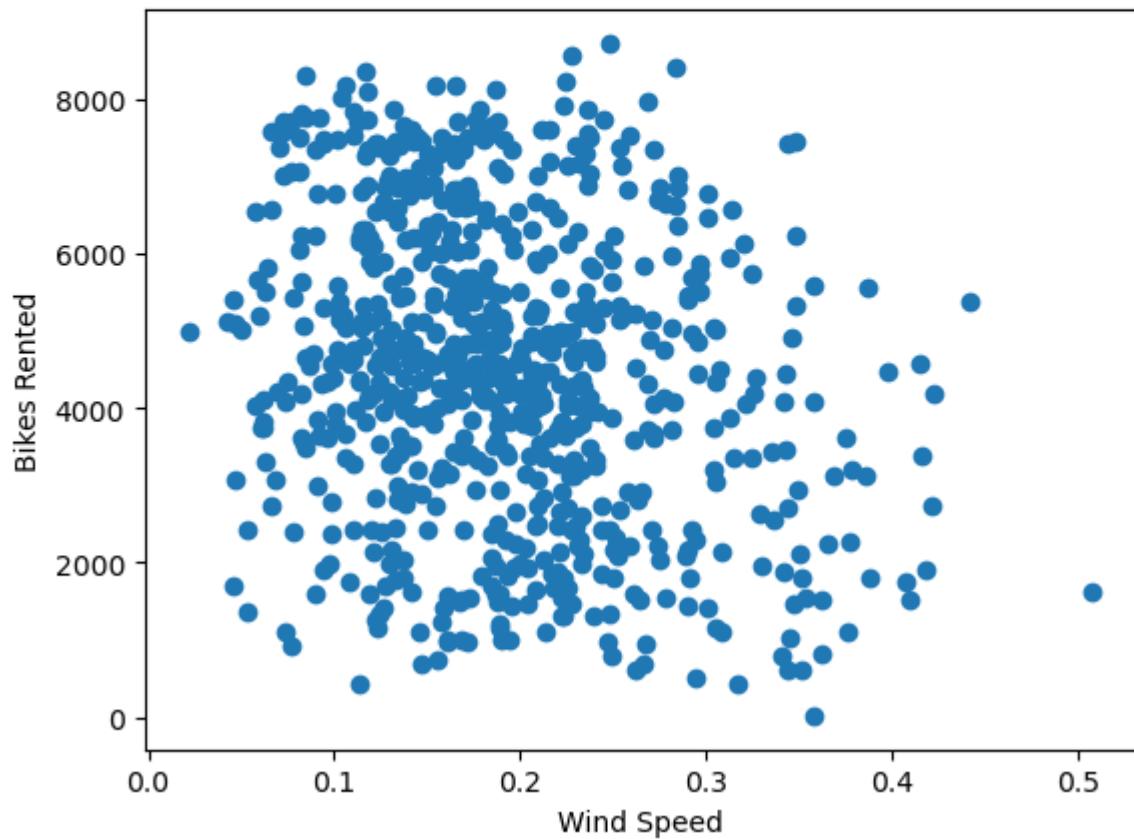


4. Scatter Plots

```
plt.scatter(bike_sharing['temp'], bike_sharing['cnt'])  
plt.xlabel('Temperature')  
plt.ylabel('Bikes Rented')  
plt.show()
```



```
import pandas as pd
import matplotlib.pyplot as plt
bike_sharing = pd.read_csv('day.csv')
bike_sharing['dteday'] = pd.to_datetime(bike_sharing['dteday'])
plt.scatter(bike_sharing['windspeed'], bike_sharing['cnt'])
plt.ylabel('Bikes Rented')
plt.xlabel('Wind Speed')
plt.show()
```

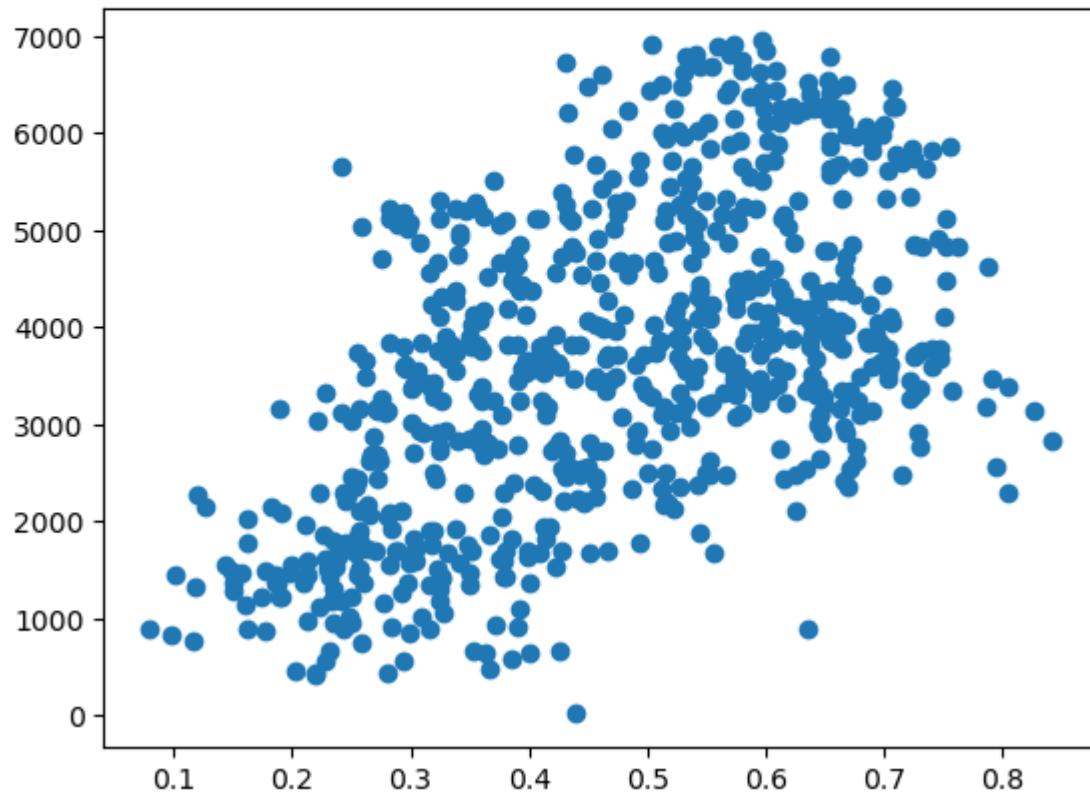


5. Correlation

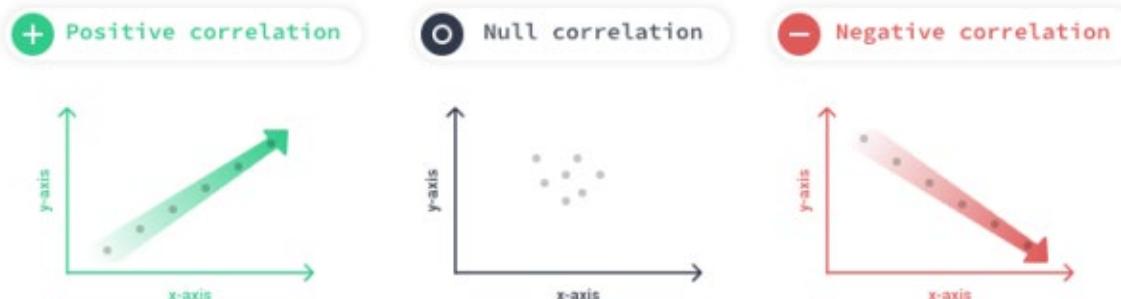
```

import pandas as pd
import matplotlib.pyplot as plt
bike_sharing = pd.read_csv('day.csv')
bike_sharing['dteday'] = pd.to_datetime(bike_sharing['dteday'])
plt.scatter(bike_sharing['atemp'], bike_sharing['registered'])
plt.show()
correlation = 'positive'

```



6. Pearson Correlation Coefficient



7. Measuring Pearson's R

```
bike_sharing['temp'].corr(bike_sharing['cnt'])
```

```
0.6274940090334918
```

```
bike_sharing['windspeed'].corr(bike_sharing['cnt'])
```

```
-0.23454499742167
```

```
#bike_sharing['temp'].corr(bike_sharing['cnt'])  
bike_sharing['cnt'].corr(bike_sharing['temp'])
```

```
0.6274940090334918
```

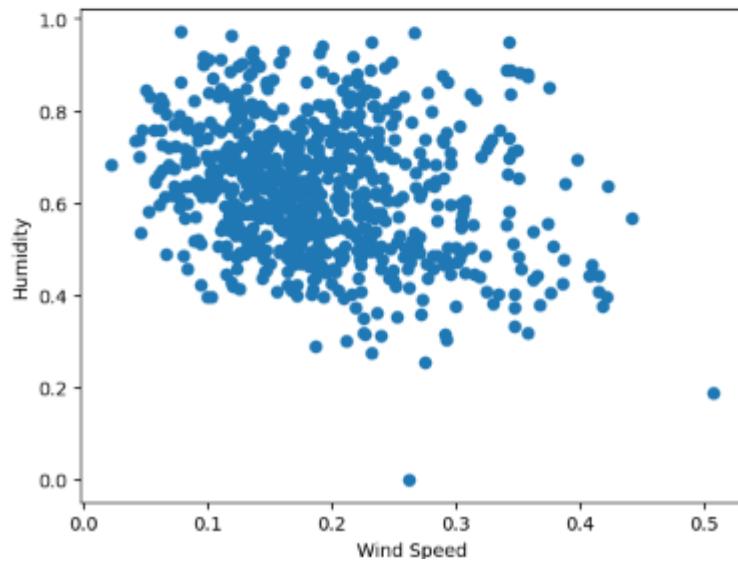
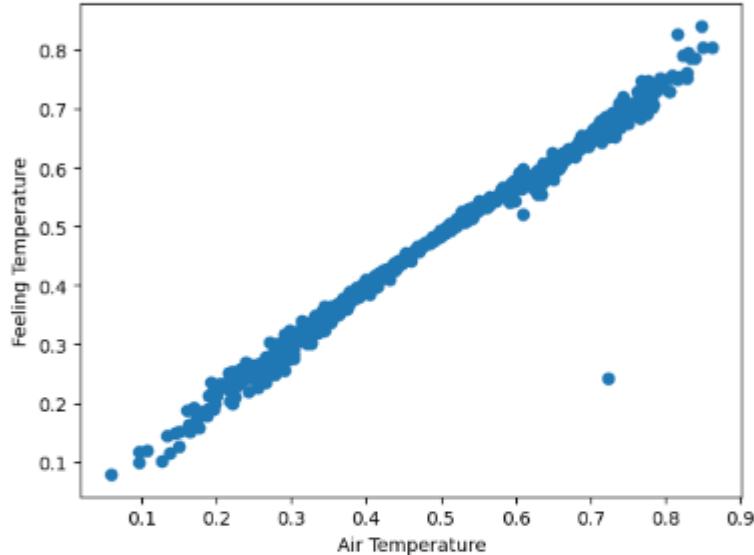
```
#bike_sharing['windspeed'].corr(bike_sharing['cnt'])  
bike_sharing['cnt'].corr(bike_sharing['windspeed'])
```

```
-0.23454499742167
```

```

import pandas as pd
import matplotlib.pyplot as plt
bike_sharing = pd.read_csv('day.csv')
bike_sharing['dteday'] = pd.to_datetime(bike_sharing['dteday'])
temp_atemp_corr = bike_sharing['temp'].corr(bike_sharing['atemp'])
wind_hum_corr = bike_sharing['windspeed'].corr(bike_sharing['hum'])
plt.scatter(bike_sharing['temp'], bike_sharing['atemp'])
plt.xlabel('Air Temperature')
plt.ylabel('Feeling Temperature')
plt.show()
plt.scatter(bike_sharing['windspeed'], bike_sharing['hum'])
plt.xlabel('Wind Speed')
plt.ylabel('Humidity')
plt.show()

```



8. Categorical Columns

```
bike_sharing['workingday'].value_counts()
```

```

1    500
0    231
Name: workingday, dtype: int64

```

```
bike_sharing.corr()['workingday'][['casual','registered']]
```

```
casual      -0.518044  
registered   0.303907  
Name: workingday, dtype: float64
```

```
bike_sharing['workingday'].replace({1:0, 0:1}, inplace=True)  
bike_sharing.corr()['workingday'][['casual','registered']]
```

```
casual      0.518044  
registered  -0.303907  
Name: workingday, dtype: float64
```

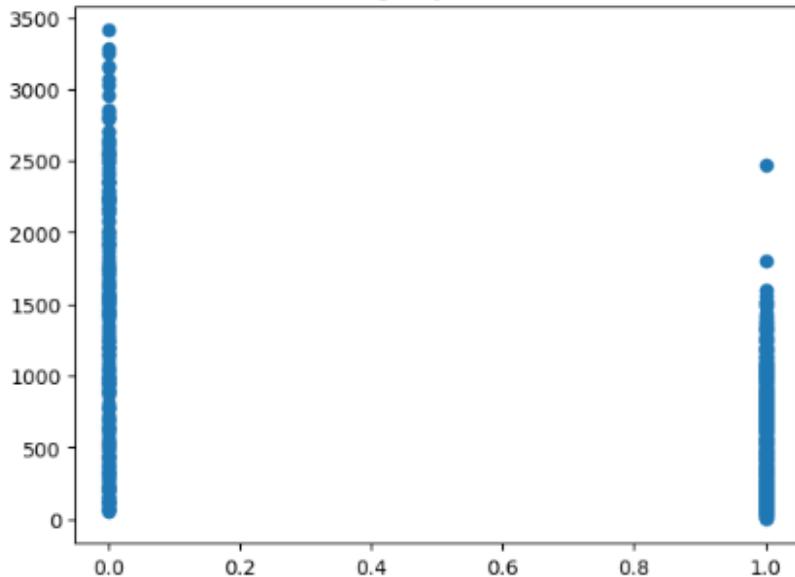
Bar Plots, Histograms, and Distributions

1. Introduction	2
2. Bar Plots	2
3. Customizing Bar Plots	4
4. Frequency Tables.....	7
5. Grouped Frequency Tables.....	9
6. Histograms.....	11
7. The Normal Distribution.....	13
8. The Uniform Distribution.....	13
9. Skewed Distributions.....	14

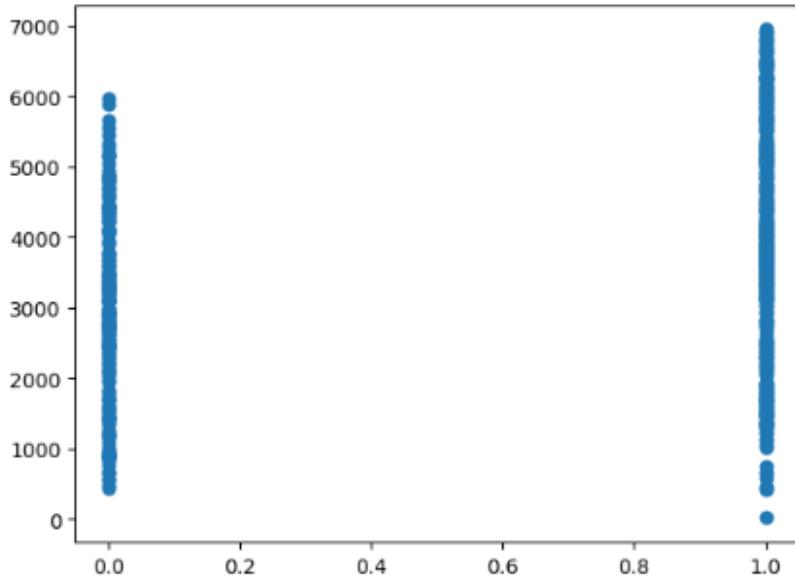
1. Introduction

```
import pandas as pd
import matplotlib.pyplot as plt
bike_sharing = pd.read_csv('day.csv')
bike_sharing['dteday'] = pd.to_datetime(bike_sharing['dteday'])
plt.scatter(bike_sharing['workingday'], bike_sharing['casual'])
plt.title('Working Day Vs. Casual')
plt.show()
plt.scatter(bike_sharing['workingday'], bike_sharing['registered'])
plt.title('Working Day Vs. Registered')
plt.show()
```

Working Day Vs. Casual

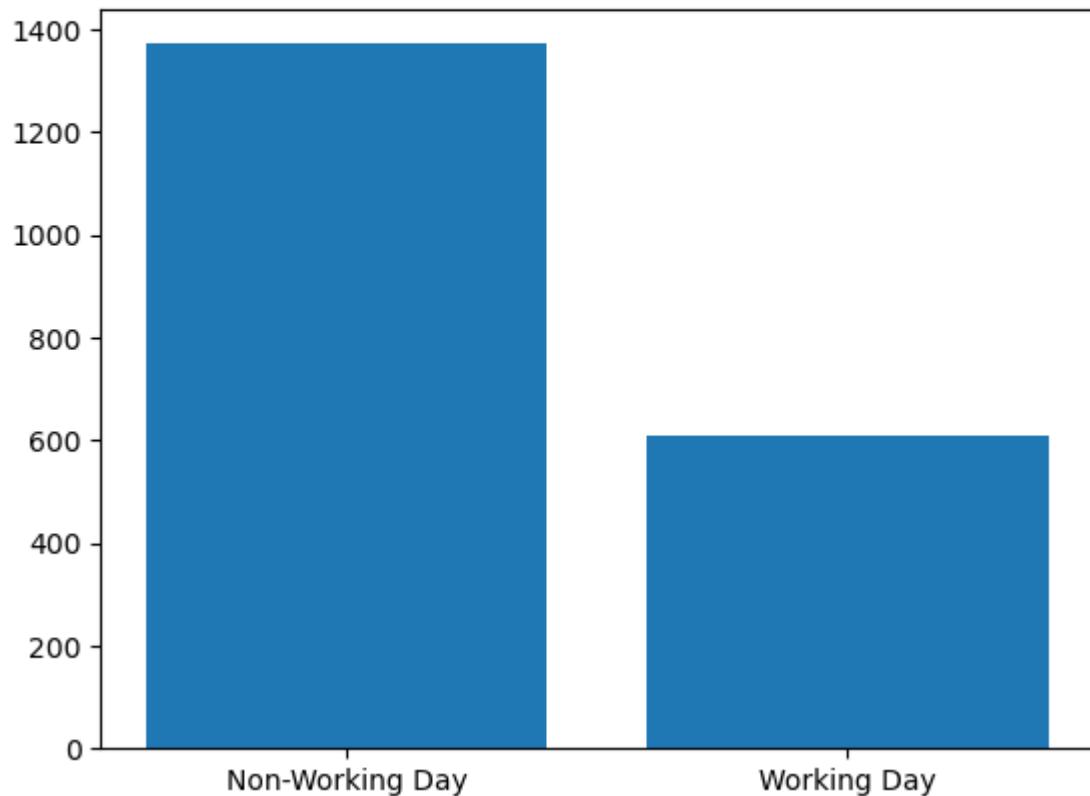


Working Day Vs. Registered

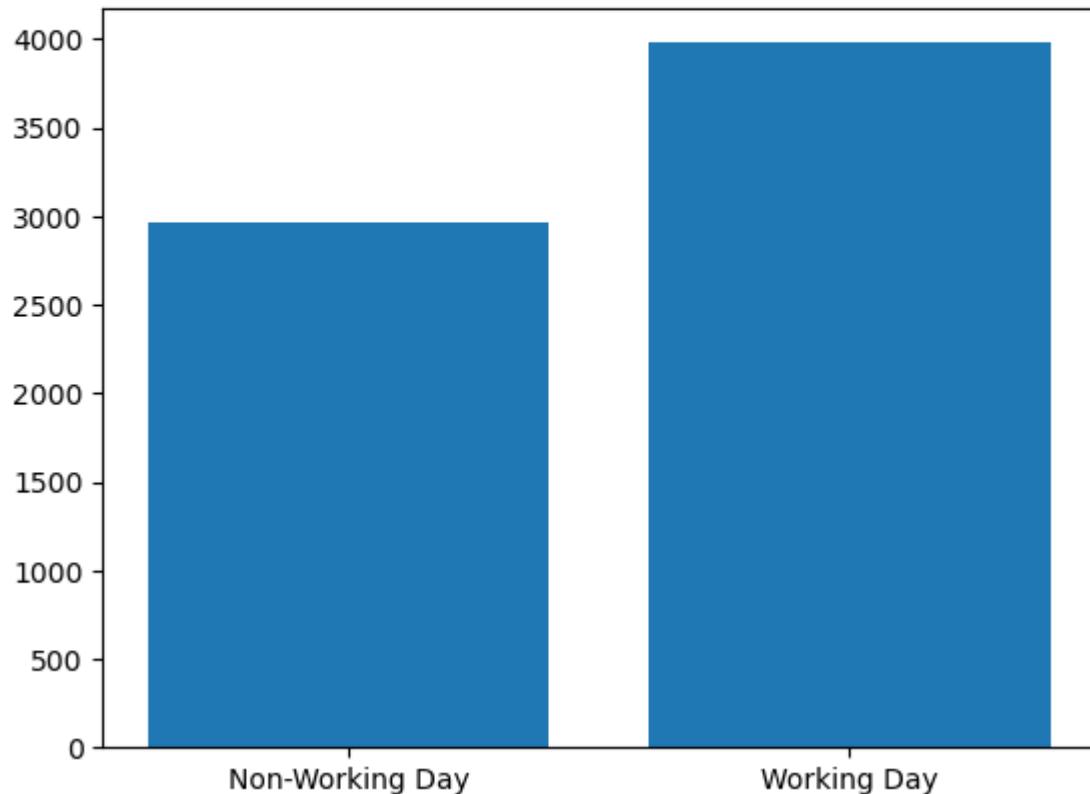


2. Bar Plots

```
import matplotlib.pyplot as plt
working_days = ['Non-Working Day', 'Working Day']
casual_avg = [1371, 607]
plt.bar(working_days, casual_avg)
plt.show()
```



```
import matplotlib.pyplot as plt
working_days = ['Non-Working Day', 'Working Day']
registered_avg = [2959, 3978]
plt.bar(working_days, registered_avg)
plt.show()
```

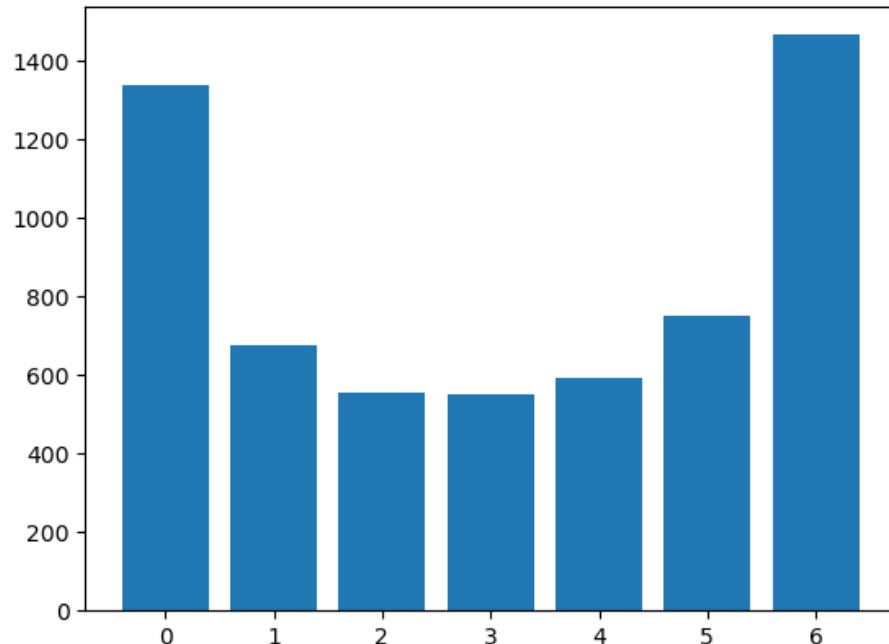


3. Customizing Bar Plots

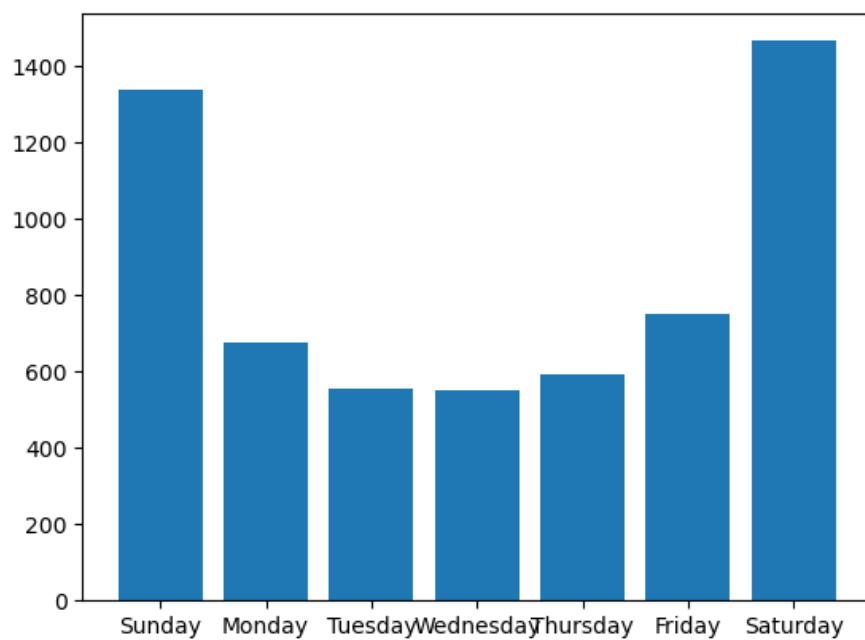
```
bike_sharing['weekday'].value_counts().sort_index()
```

```
0    105
1    105
2    104
3    104
4    104
5    104
6    105
Name: weekday, dtype: int64
```

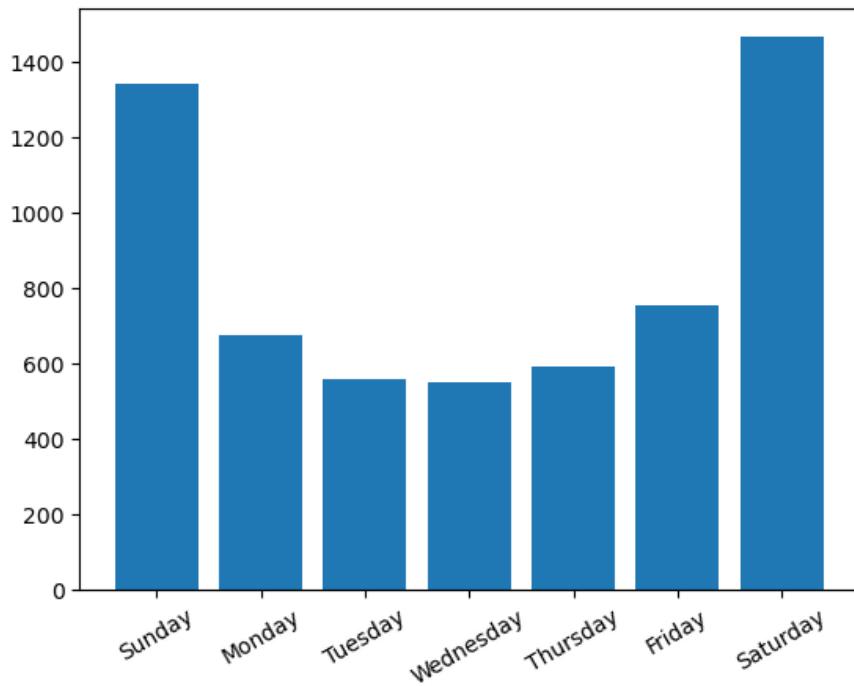
```
weekday_averages = bike_sharing.groupby('weekday').mean()[['casual', 'registered']].reset_index()
plt.bar(weekday_averages['weekday'], weekday_averages['casual'])
plt.show()
```



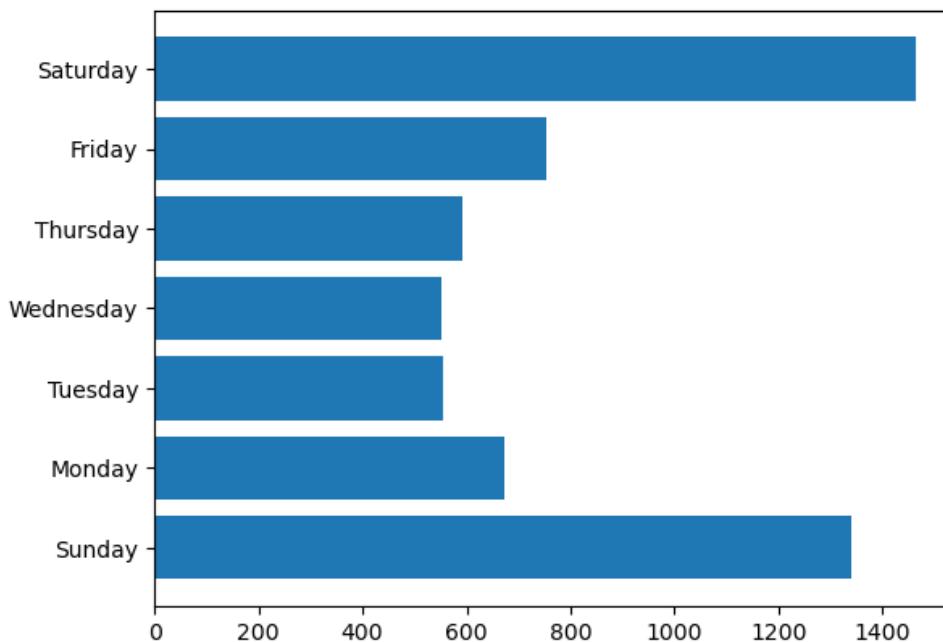
```
: weekday_averages = bike_sharing.groupby('weekday').mean()[['casual', 'registered']].reset_index()
plt.bar(weekday_averages['weekday'], weekday_averages['casual'])
plt.xticks(ticks=[0, 1, 2, 3, 4, 5, 6],
           labels=['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'])
plt.show()
```



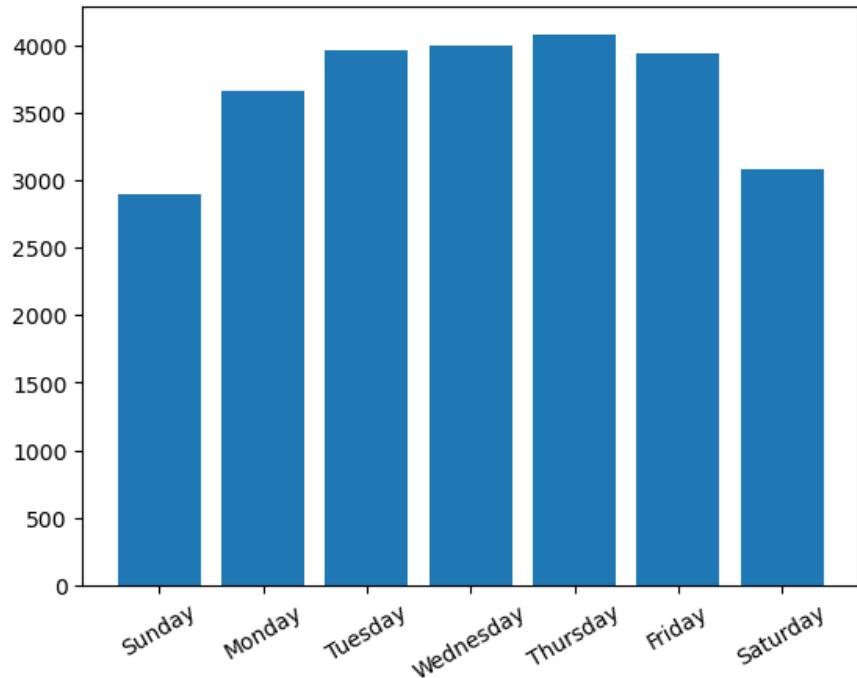
```
ay_averages = bike_sharing.groupby('weekday').mean()[['casual', 'registered']].reset_index()
ar(weekday_averages['weekday'], weekday_averages['casual'])
ticks(ticks=[0, 1, 2, 3, 4, 5, 6],
      labels=['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'], rotation=30)
how()
```



```
: weekday_averages = bike_sharing.groupby('weekday').mean()[['casual', 'registered']].reset_index()
plt.barh(weekday_averages['weekday'], weekday_averages['casual'])
plt.yticks(ticks=[0, 1, 2, 3, 4, 5, 6],
           labels=['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'])
plt.show()
```

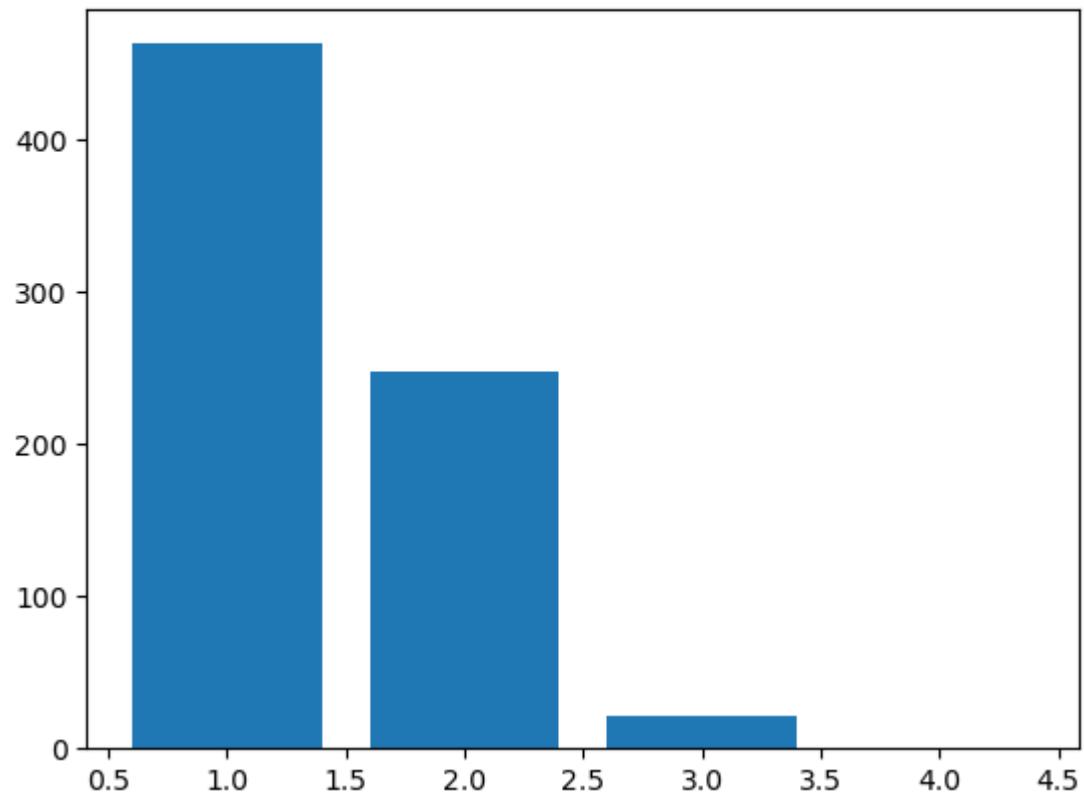


```
import pandas as pd
import matplotlib.pyplot as plt
bike_sharing = pd.read_csv('day.csv')
bike_sharing['dteday'] = pd.to_datetime(bike_sharing['dteday'])
weekday_averages = bike_sharing.groupby('weekday').mean()[['casual', 'registered']].reset_index()
plt.bar(weekday_averages['weekday'], weekday_averages['registered'])
plt.xticks(ticks=[0, 1, 2, 3, 4, 5, 6],
           labels=['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'], rotation=45)
plt.show()
```



4. Frequency Tables

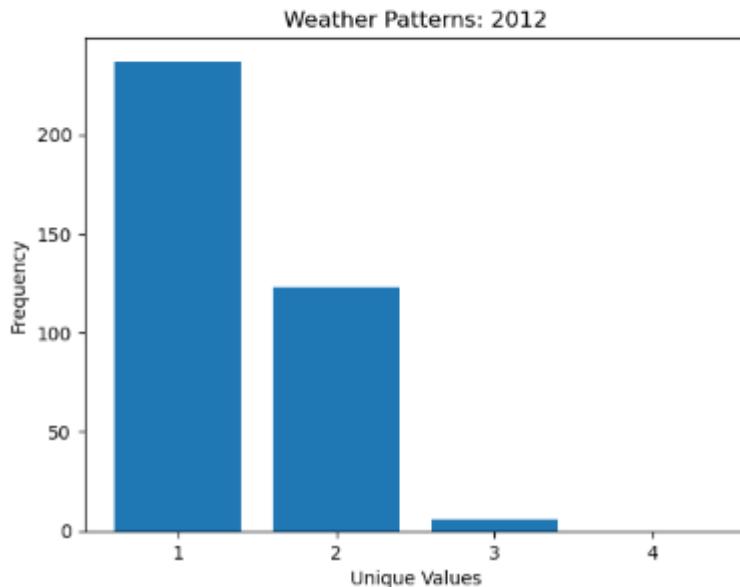
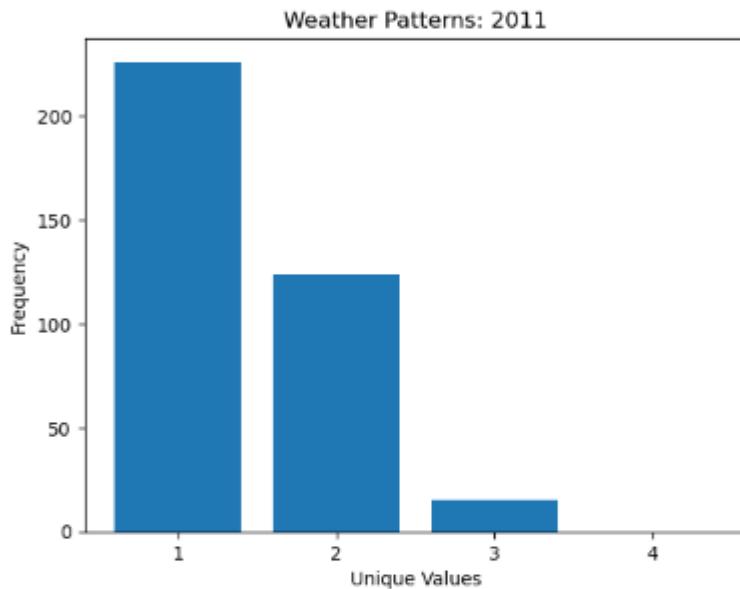
```
weather_types = [1, 2, 3, 4]
frequencies = [463, 247, 21, 0]
plt.bar(weather_types, frequencies)
plt.show()
```



```

import matplotlib.pyplot as plt
unique_values = [1, 2, 3, 4]
weather_2011 = [226, 124, 15, 0]
weather_2012 = [237, 123, 6, 0]
plt.bar(unique_values, weather_2011)
plt.xticks(ticks=[1,2,3,4])
plt.title('Weather Patterns: 2011')
plt.ylabel('Frequency')
plt.xlabel('Unique Values')
plt.show()
plt.bar(unique_values, weather_2012)
plt.xticks(ticks=[1,2,3,4])
plt.title('Weather Patterns: 2012')
plt.ylabel('Frequency')
plt.xlabel('Unique Values')
plt.show()

```



5. Grouped Frequency Tables

```
bike_sharing['cnt'].value_counts()
```

```
5409    2  
2424    2  
5698    2  
4459    2  
5119    2  
..  
5046    1  
4713    1  
4763    1  
4785    1  
2729    1  
Name: cnt, Length: 696, dtype: int64
```

```
bike_sharing['cnt'].value_counts(bins=10)
```

```
(4368.0, 5237.2]    137  
(3498.8, 4368.0]   122  
(5237.2, 6106.4]   81  
(6975.6, 7844.8]   79  
(6106.4, 6975.6]   76  
(2629.6, 3498.8]   73  
(1760.4, 2629.6]   71  
(891.2, 1760.4]    62  
(7844.8, 8714.0]   17  
(13.307, 891.2]    13  
Name: cnt, dtype: int64
```

```
bike_sharing['cnt'].value_counts(bins=10).sort_index()
```

```
(13.307, 891.2]    13  
(891.2, 1760.4]    62  
(1760.4, 2629.6]   71  
(2629.6, 3498.8]   73  
(3498.8, 4368.0]   122  
(4368.0, 5237.2]   137  
(5237.2, 6106.4]   81  
(6106.4, 6975.6]   76  
(6975.6, 7844.8]   79  
(7844.8, 8714.0]   17  
Name: cnt, dtype: int64
```

```
import pandas as pd
import matplotlib.pyplot as plt
bike_sharing = pd.read_csv('day.csv')
bike_sharing['dteday'] = pd.to_datetime(bike_sharing['dteday'])
registered_freq = bike_sharing['registered'].value_counts(bins=10).sort_index()
casual_freq = bike_sharing['casual'].value_counts(bins=10).sort_index()
registered_freq
#casual_freq
```

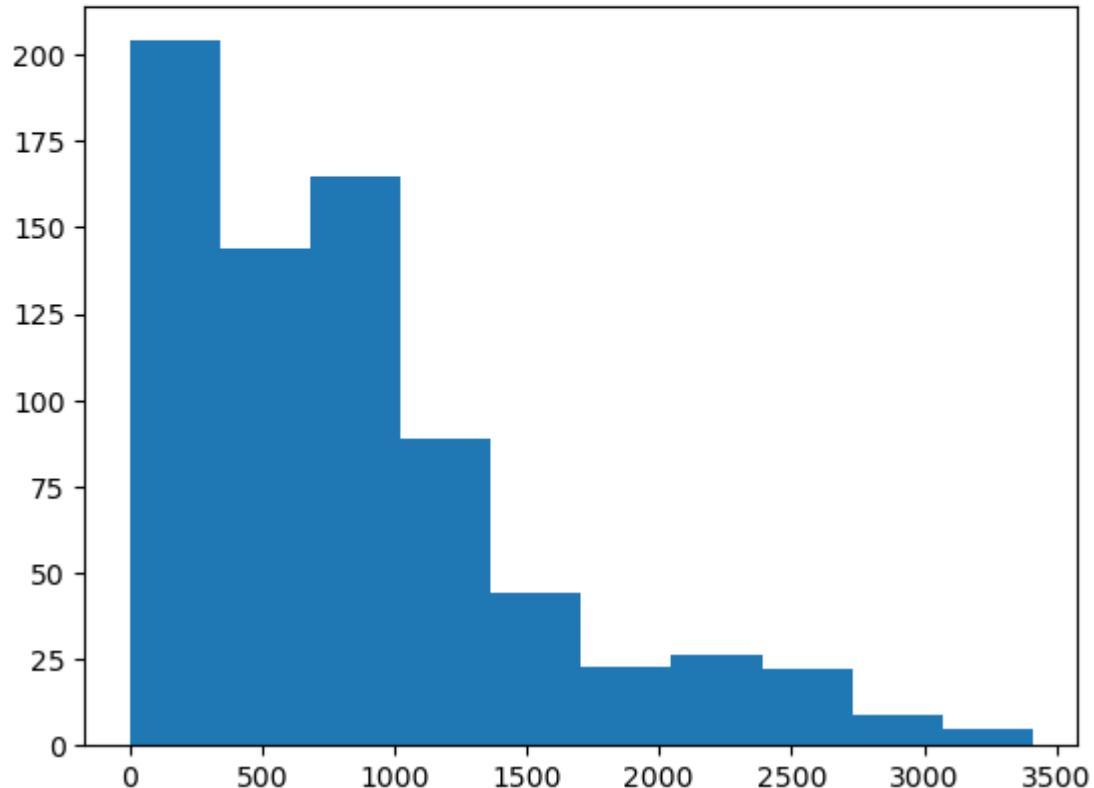
```
(13.073, 712.6]      14
(712.6, 1405.2]     44
(1405.2, 2097.8]    79
(2097.8, 2790.4]    80
(2790.4, 3483.0]   113
(3483.0, 4175.6]   144
(4175.6, 4868.2]   85
(4868.2, 5560.8]   71
(5560.8, 6253.4]   57
(6253.4, 6946.0]   44
Name: registered, dtype: int64
```

```
import pandas as pd
import matplotlib.pyplot as plt
bike_sharing = pd.read_csv('day.csv')
bike_sharing['dteday'] = pd.to_datetime(bike_sharing['dteday'])
registered_freq = bike_sharing['registered'].value_counts(bins=10).sort_index()
casual_freq = bike_sharing['casual'].value_counts(bins=10).sort_index()
#registered_freq
casual_freq
```

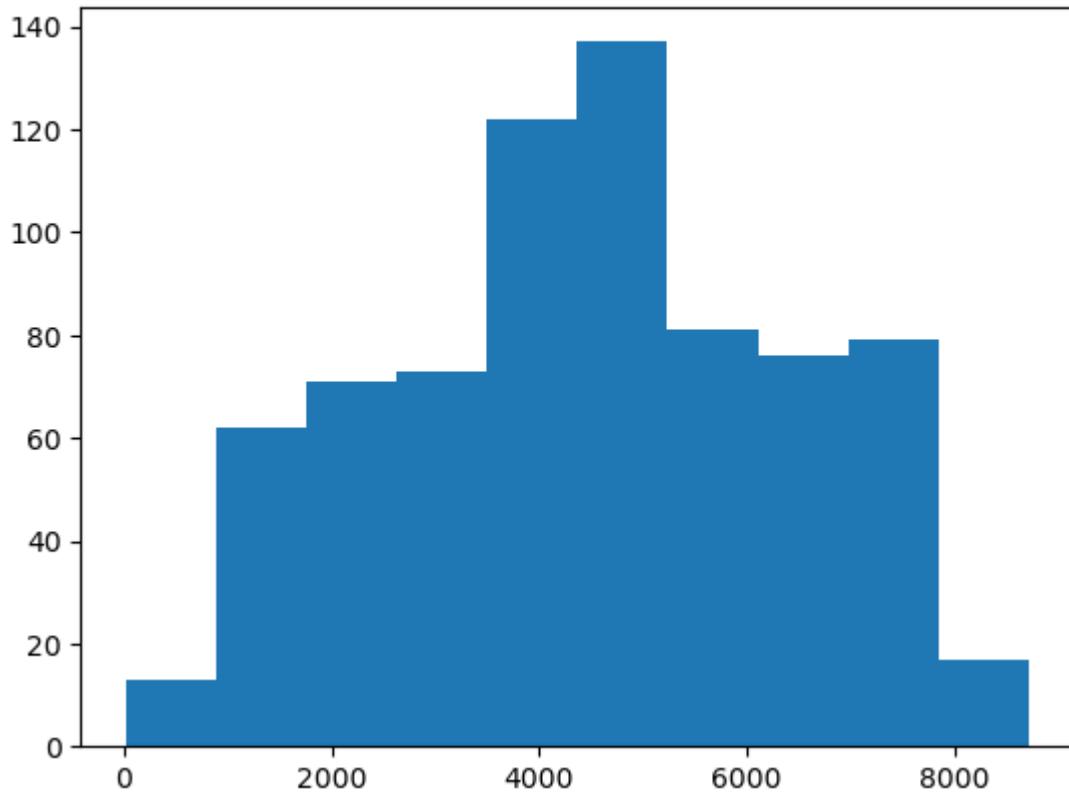
```
(-1.408999999999998, 342.8]    204
(342.8, 683.6]                  144
(683.6, 1024.4]                 165
(1024.4, 1365.2]                89
(1365.2, 1706.0]                44
(1706.0, 2046.8]                23
(2046.8, 2387.6]                26
(2387.6, 2728.4]                22
(2728.4, 3069.2]                9
(3069.2, 3410.0]                5
Name: casual, dtype: int64
```

6. Histograms

```
import pandas as pd
import matplotlib.pyplot as plt
bike_sharing = pd.read_csv('day.csv')
bike_sharing['dteday'] = pd.to_datetime(bike_sharing['dteday'])
plt.hist(bike_sharing['casual'])
plt.show()
```



```
plt.hist(bike_sharing['cnt'])  
plt.show()
```



7. The Normal Distribution

Histogram cho ta thấy sự phân bố của giá trị, nếu hình dáng là nó đối xứng thì ta gọi nó là sự phân bố đối xứng.

Hầu hết các giá trị trong cột đó nằm ở giữa phạm vi.

Khi ta tiếp cận các điểm tối thiểu và tối đa của phạm vi, ta có ít hơn các giá trị đó. Để hiểu rõ hơn ý nghĩa của phân phối chuẩn, giả sử chúng ta đã vẽ một biểu đồ với số điểm mà học sinh đạt được trong bài kiểm tra toán. Nếu biểu đồ hiển thị một phân phối bình thường, thì điều này có nghĩa như sau: Phần lớn kết quả học tập của học sinh đều ở mức trung bình (nằm đâu đó ở giữa điểm tối thiểu và điểm tối đa). Khi ta đạt đến điểm tối thiểu và tối đa, ta ngày càng có ít học sinh hơn.

8. The Uniform Distribution

Một phân phối đối xứng khác mà chúng ta có thể thấy trong thực tế là đồng phục

phân phối - các giá trị được phân phối đồng đều. Các thanh có chiều cao bằng nhau

vì các khoảng có tần số bằng nhau.

9. Skewed Distributions

Nếu đuôi chỉ về bên phải, thì phân phối bị lệch phải (phân phối của cột ngẫu nhiên bị lệch phải). Nếu đuôi chỉ về bên trái, thì phân phối được cho là lệch trái.

Pandas Visualizations and Grid Charts

1. Traffic Congestions in Sao Paulo	2
2. Traffic Behavior Dataset	2
3. Slowness in Traffic.....	2
4. Pandas Visualization Methods	3
5. Frequency of Incidents	5
6. Correlations with Traffic Slowness	6
7. Traffic Slowness Over 20%.....	9
8. How Traffic Slowness Change.....	10
9. Comparing Graphs	11

1. Traffic Congestions in Sao Paulo

Phân tích dữ liệu, visualization bằng pandas nhanh hơn.

So sánh tốt hơn giữa 2 đồ thị bằng grid charts.

2. Traffic Behavior Dataset

```
import pandas as pd
traffic = pd.read_csv('traffic_sao_paulo.csv', sep=';')
traffic.head()
traffic.tail()
traffic.info()
```

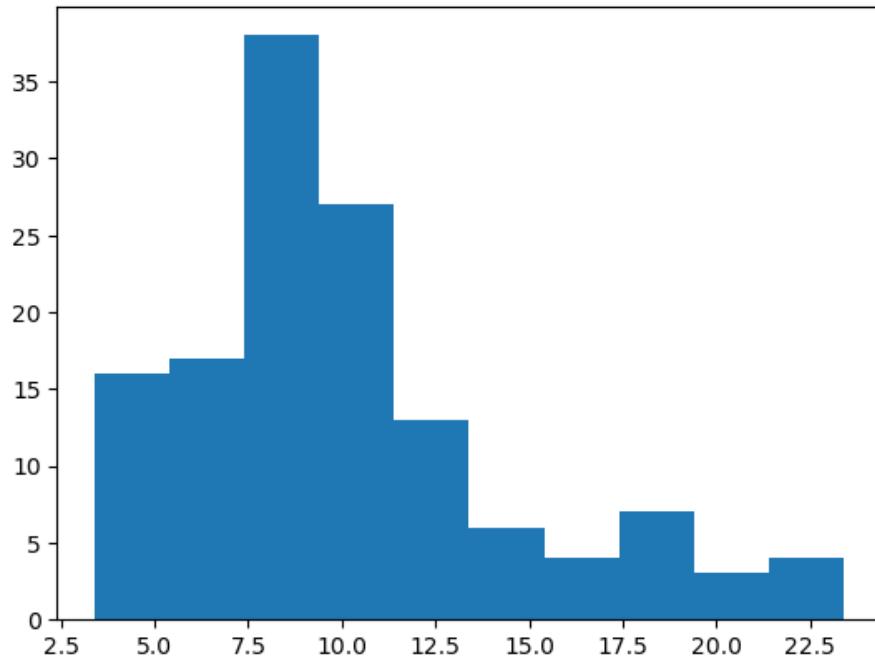
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 135 entries, 0 to 134
Data columns (total 18 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Hour (Coded)    135 non-null    int64  
 1   Immobilized bus 135 non-null    int64  
 2   Broken Truck     135 non-null    int64  
 3   Vehicle excess   135 non-null    int64  
 4   Accident victim 135 non-null    int64  
 5   Running over     135 non-null    int64  
 6   Fire vehicles    135 non-null    int64  
 7   Occurrence involving freight 135 non-null    int64  
 8   Incident involving dangerous freight 135 non-null    int64  
 9   Lack of electricity 135 non-null    int64  
 10  Fire             135 non-null    int64  
 11  Point of flooding 135 non-null    int64  
 12  Manifestations   135 non-null    int64  
 13  Defect in the network of trolleybuses 135 non-null    int64  
 14  Tree on the road   135 non-null    int64  
 15  Semaphore off      135 non-null    int64  
 16  Intermittent Semaphore 135 non-null    int64  
 17  Slowness in traffic (%) 135 non-null    object 
dtypes: int64(17), object(1)
memory usage: 19.1+ KB
```

3. Slowness in Traffic

```
traffic['Slowness in traffic (%)'].describe()
```

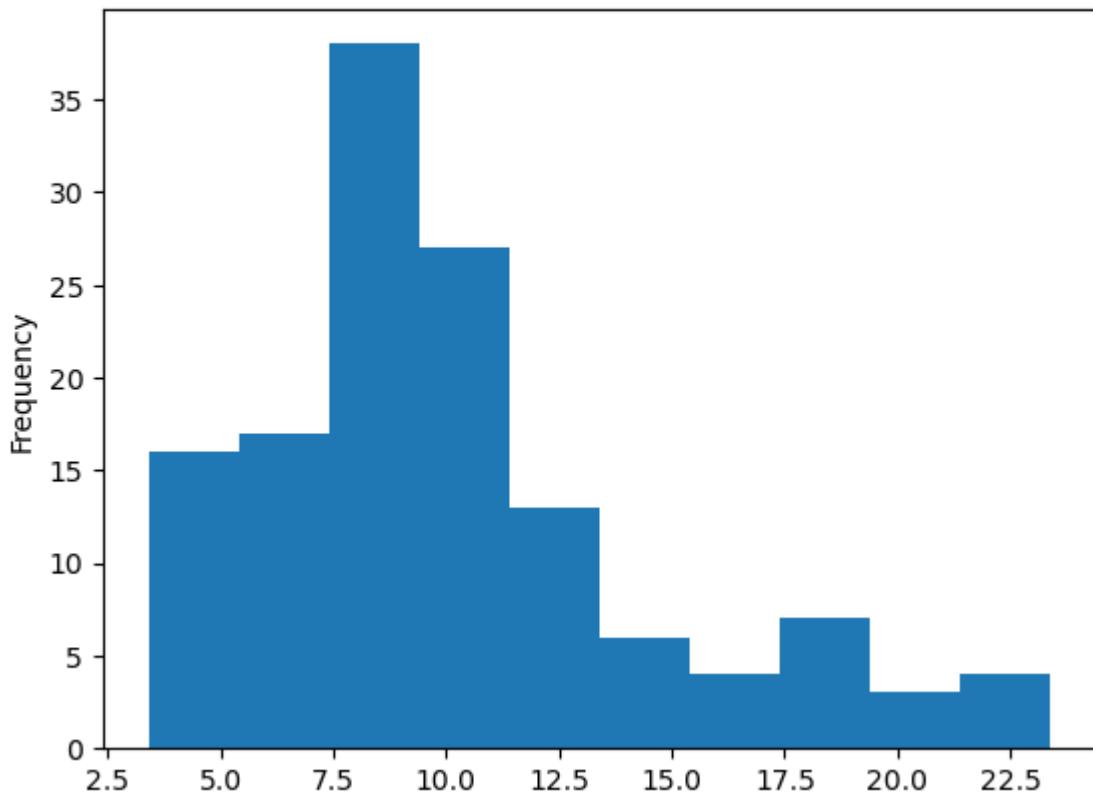
```
count    135
unique   83
top      10,3
freq     4
Name: Slowness in traffic (%), dtype: object
```

```
import pandas as pd
traffic = pd.read_csv('traffic_sao_paulo.csv', sep=';')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)'].str.replace(',', '.')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)'].astype(float)
import matplotlib.pyplot as plt
plt.hist(traffic['Slowness in traffic (%)'])
plt.show()
```



4. Pandas Visualization Methods

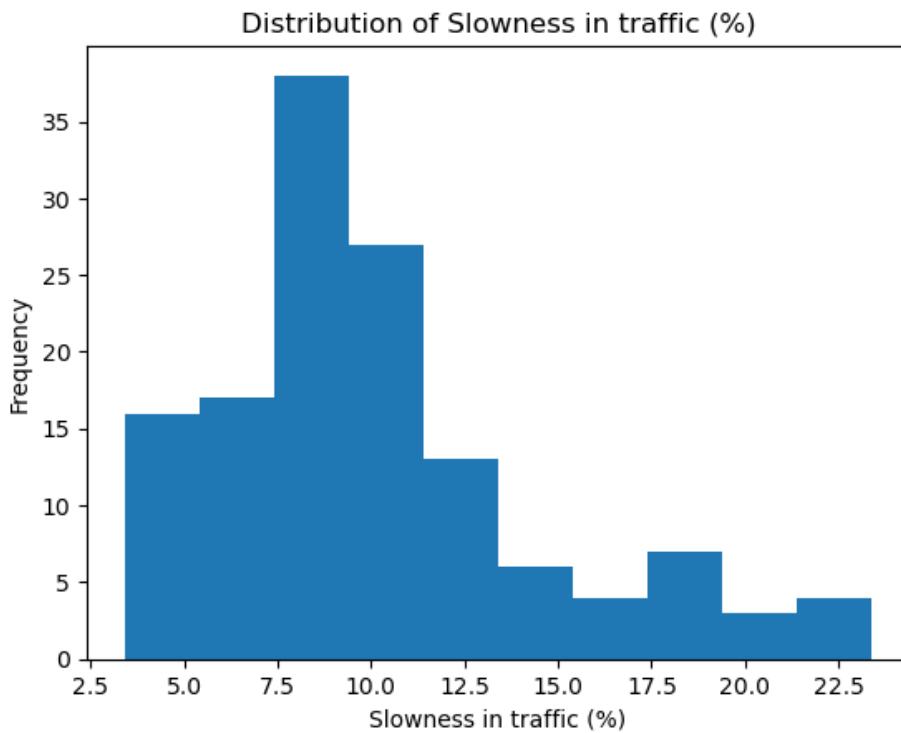
```
traffic['Slowness in traffic (%)'].plot.hist()  
plt.show()
```



```

import matplotlib.pyplot as plt
import pandas as pd
traffic = pd.read_csv('traffic_sao_paulo.csv', sep=';')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)'].str.replace(',', '.')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)'].astype(float)
traffic['Slowness in traffic (%)'].plot.hist()
plt.title('Distribution of Slowness in traffic (%)')
plt.xlabel('Slowness in traffic (%)')
plt.show()

```



5. Frequency of Incidents

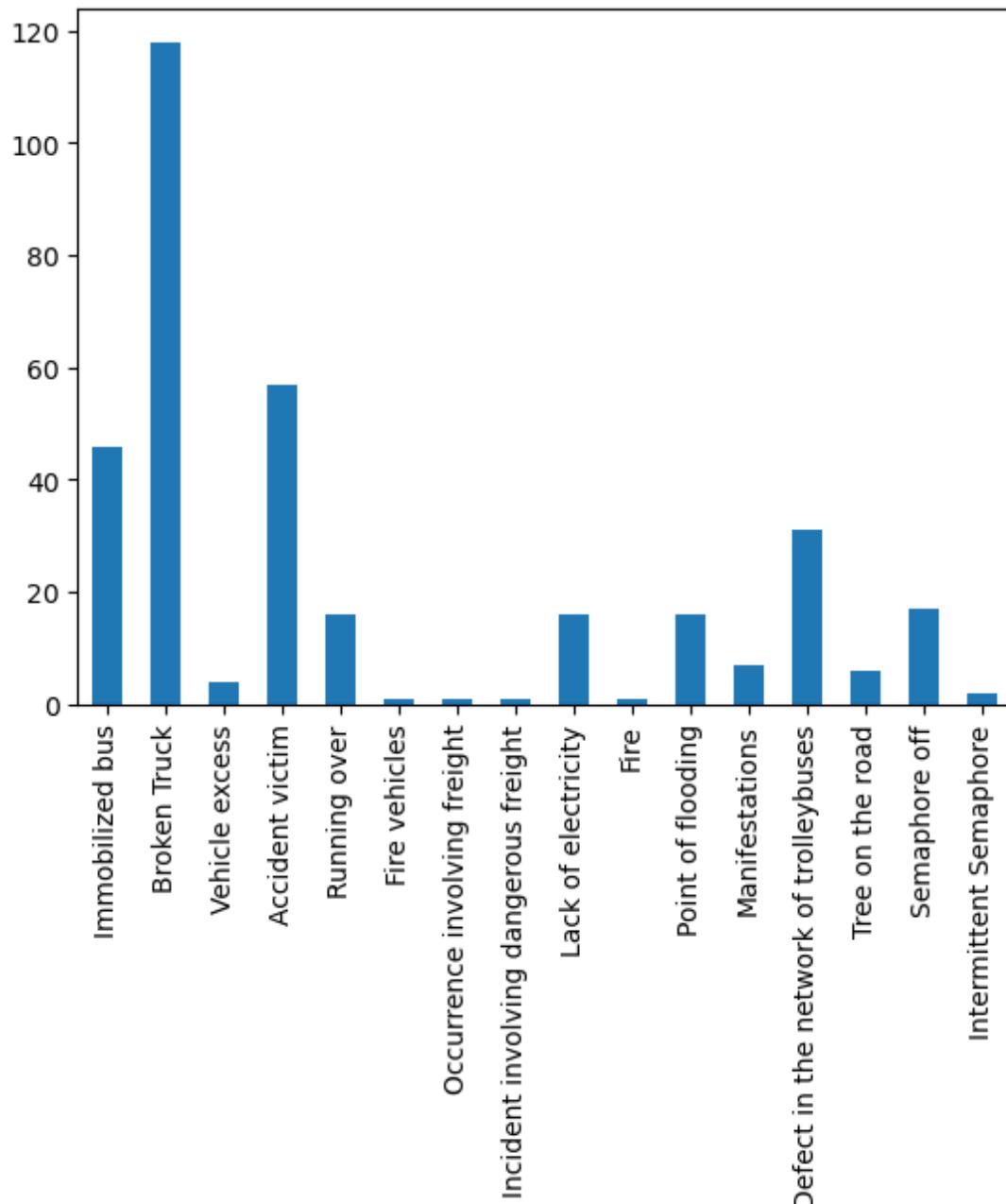
```

incidents = traffic.drop(['Hour (Coded)', 'Slowness in traffic (%)'], axis=1)
incidents.sum()

```

Immobilized bus	46
Broken Truck	118
Vehicle excess	4
Accident victim	57
Running over	16
Fire vehicles	1
Occurrence involving freight	1
Incident involving dangerous freight	1
Lack of electricity	16
Fire	1
Point of flooding	16
Manifestations	7
Defect in the network of trolleybuses	31
Tree on the road	6
Semaphore off	17
Intermittent Semaphore	2
dtype: int64	

```
type(incidents.sum())
incidents.sum().plot.bar()
plt.show()
```



6. Correlations with Traffic Slowness

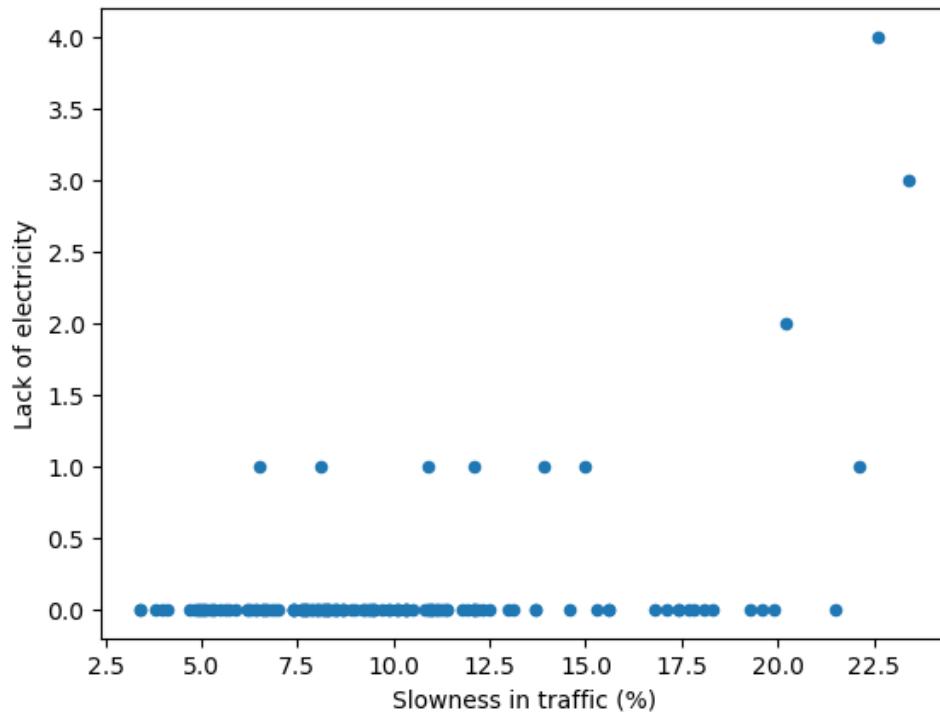
```
: traffic.corr()['Slowness in traffic (%)']

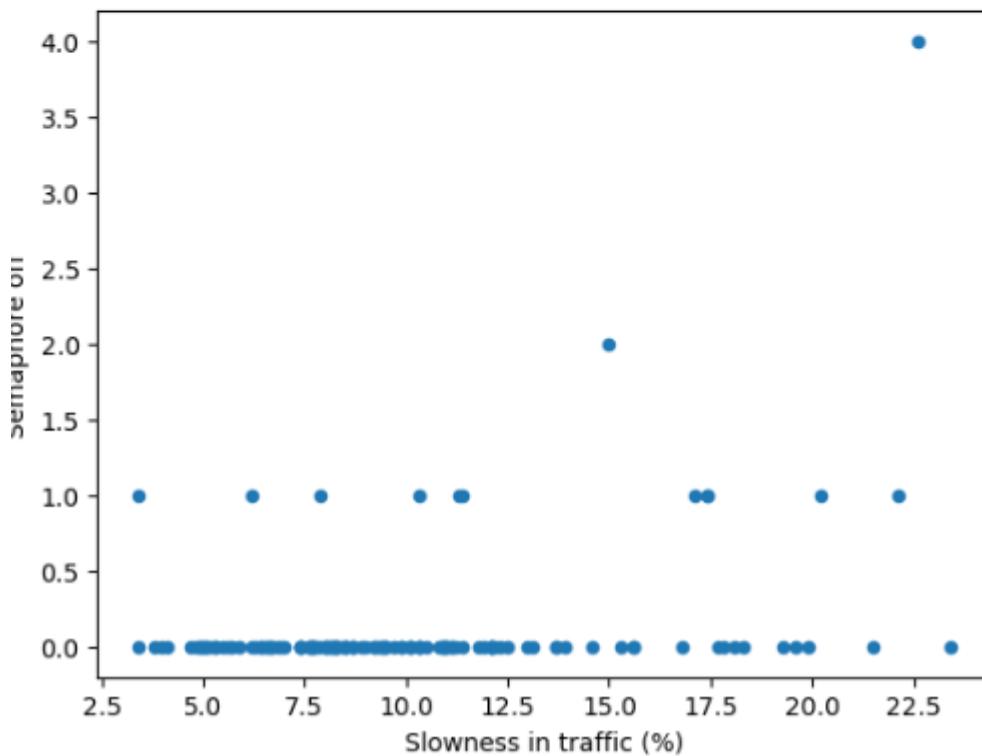
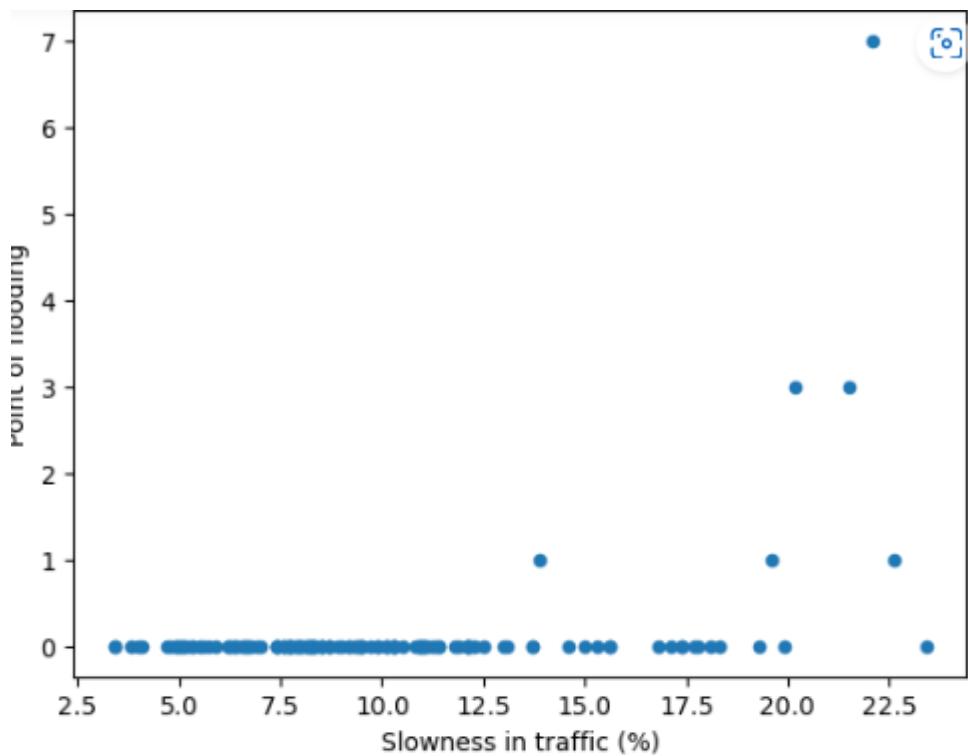
Hour (Coded)          0.729962
Immobilized bus       0.101143
Broken Truck          0.131998
Vehicle excess        -0.045297
Accident victim       0.121730
Running over          -0.001133
Fire vehicles         0.134103
Occurrence involving freight  0.026791
Incident involving dangerous freight 0.000957
Lack of electricity   0.436569
Fire                  -0.046737
Point of flooding     0.420016
Manifestations        0.066377
Defect in the network of trolleybuses -0.147035
Tree on the road       -0.098489
Semaphore off          0.347242
Intermittent Semaphore -0.119942
Slowness in traffic (%) 1.000000
Name: Slowness in traffic (%), dtype: float64
```

```

import pandas as pd
import matplotlib.pyplot as plt
traffic = pd.read_csv('traffic_sao_paulo.csv', sep=';')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)'].str.replace(',', '.')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)'].astype(float)
traffic.plot.scatter(x='Slowness in traffic (%)',
y='Lack of electricity')
plt.show()
traffic.plot.scatter(x='Slowness in traffic (%)',
y='Point of flooding')
plt.show()
traffic.plot.scatter(x='Slowness in traffic (%)',
y='Semaphore off')
plt.show()

```



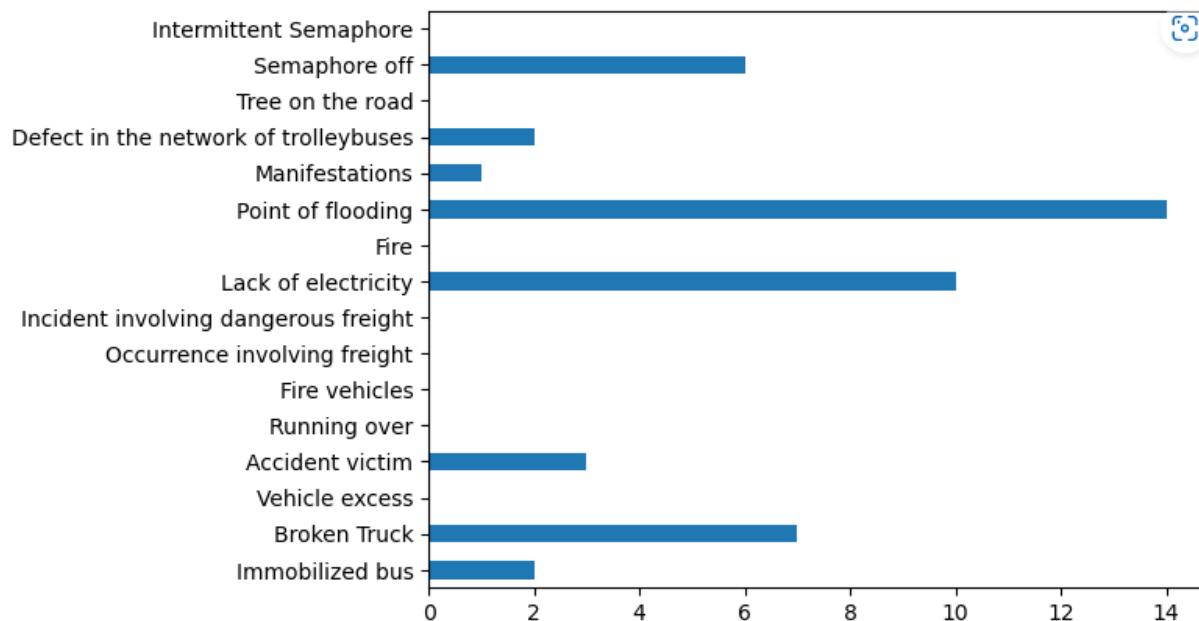


7. Traffic Slowness Over 20%

```

import pandas as pd
import matplotlib.pyplot as plt
traffic = pd.read_csv('traffic_sao_paulo.csv', sep=';')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)'].str.replace(',', '.')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)'].astype(float)
slowness_20_or_more = traffic[traffic['Slowness in traffic (%)'] >= 20]
slowness_20_or_more = slowness_20_or_more.drop(['Slowness in traffic (%)',
'Hour (Coded)'], axis=1)
incident_frequencies = slowness_20_or_more.sum()
incident_frequencies.plot.barh()
plt.show()

```

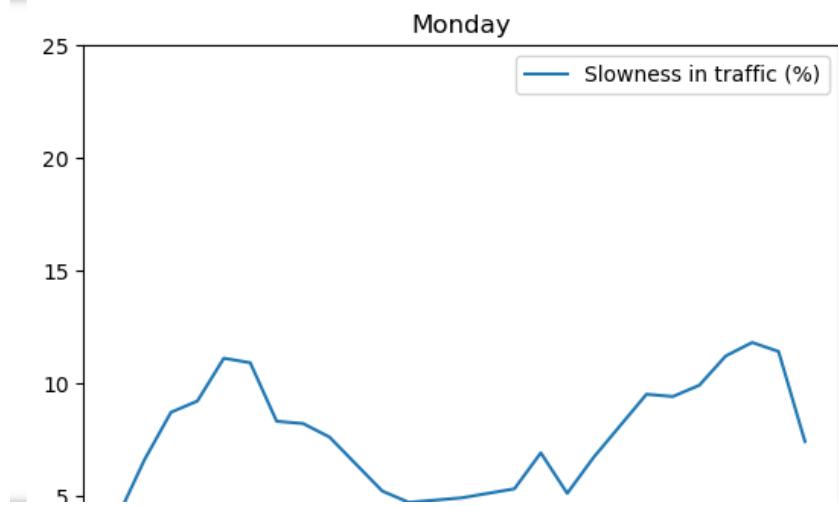


8. How Traffic Slowness Change

```

import pandas as pd
import matplotlib.pyplot as plt
traffic = pd.read_csv('traffic_sao_paulo.csv', sep=';')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)].str.replace(',', '.')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)].astype(float)
days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
traffic_per_day = {}
for i, day in zip(range(0, 135, 27), days):
    each_day_traffic = traffic[i:i+27]
    traffic_per_day[day] = each_day_traffic
for day in days:
    traffic_per_day[day].plot.line(x='Hour (Coded)',
y='Slowness in traffic (%)')
plt.title(day)
plt.ylim([0, 25])
plt.show()

```

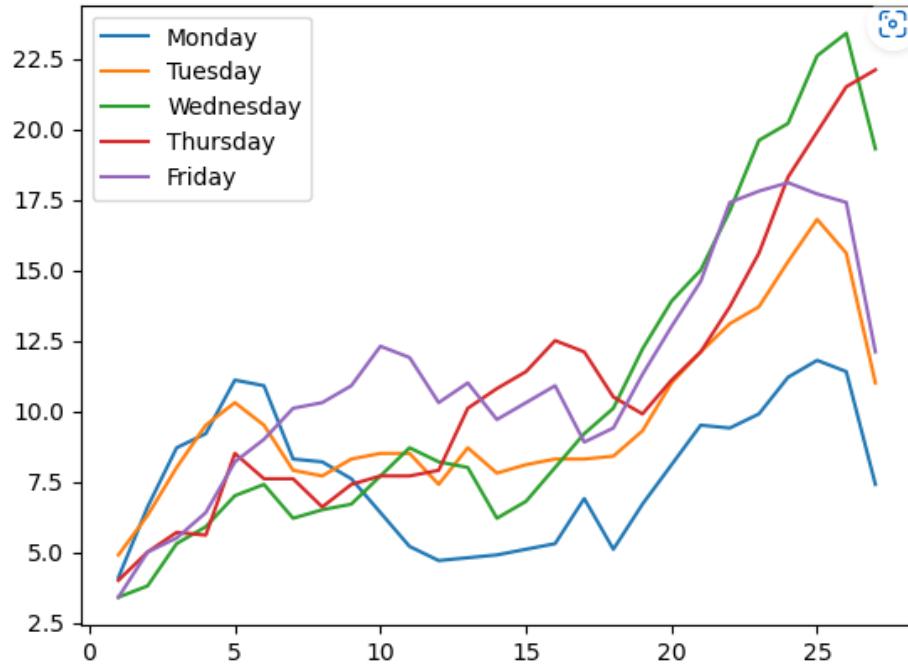


9. Comparing Graphs

```

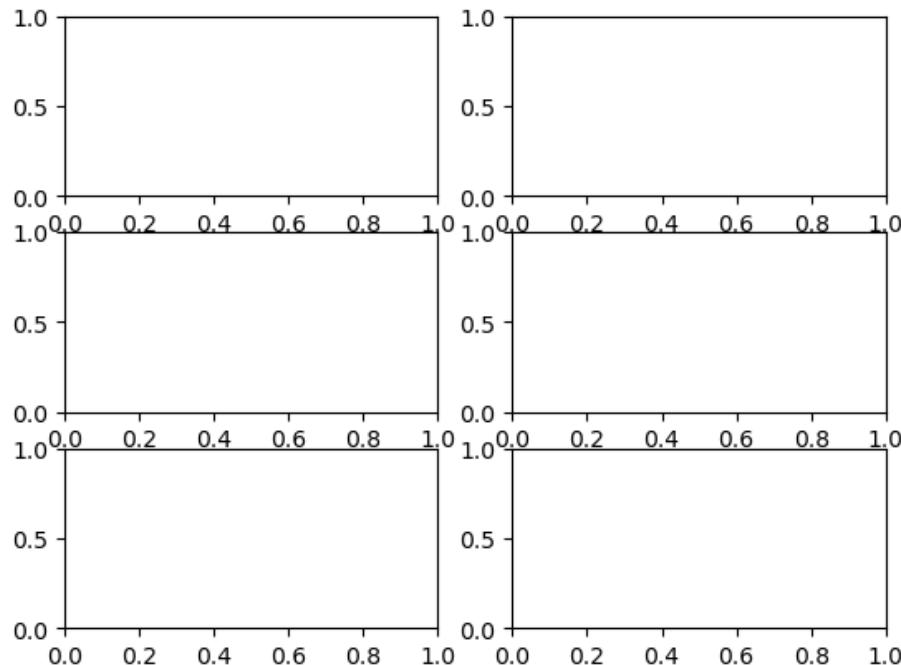
import matplotlib.pyplot as plt
traffic = pd.read_csv('traffic_sao_paulo.csv', sep=';')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)'].str.replace(',', '.')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)'].astype(float)
days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
traffic_per_day = {}
for i, day in zip(range(0, 135, 27), days):
    each_day_traffic = traffic[i:i+27]
    traffic_per_day[day] = each_day_traffic
for day in days:
    plt.plot(traffic_per_day[day]['Hour (Coded)'],
            traffic_per_day[day]['Slowness in traffic (%)'],
            label=day)
plt.legend()
plt.show()

```



10. Grid Charts

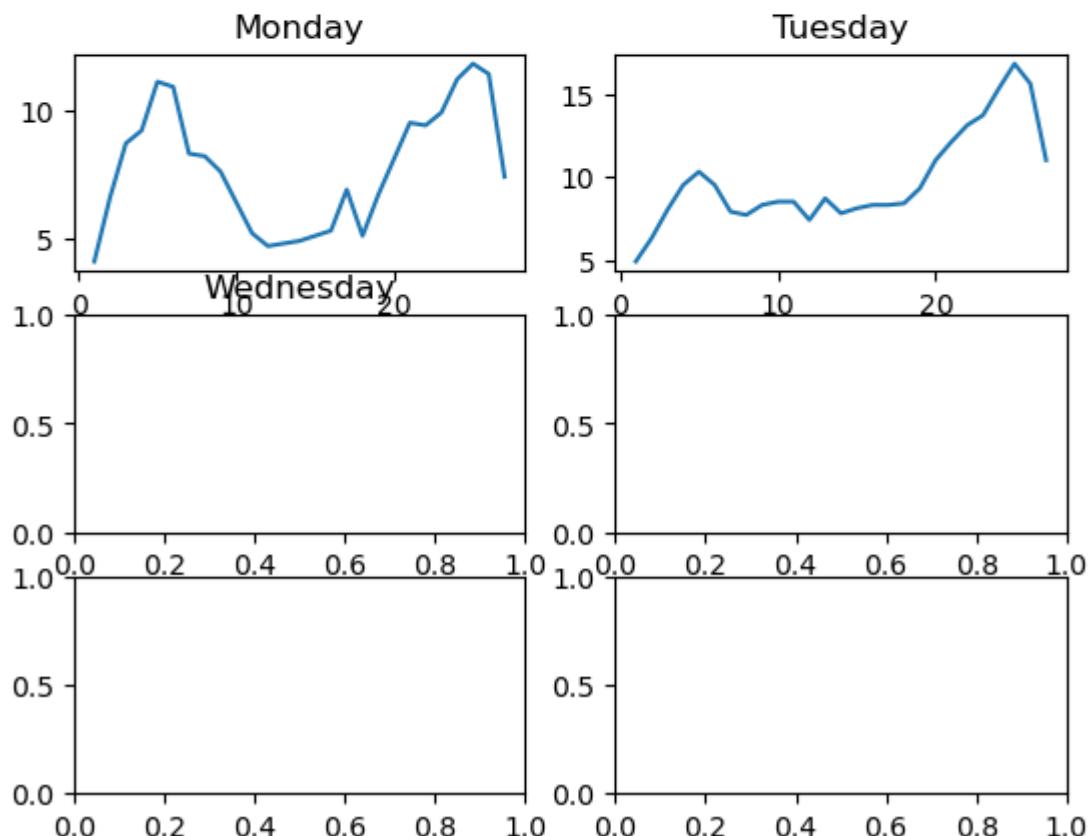
```
import pandas as pd
import matplotlib.pyplot as plt
traffic = pd.read_csv('traffic_sao_paulo.csv', sep=';')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)'].str.replace(',', '.')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)'].astype(float)
plt.figure()
plt.subplot(3, 2, 1)
plt.subplot(3, 2, 2)
plt.subplot(3, 2, 6)
plt.subplot(3, 2, 3)
plt.subplot(3, 2, 4)
plt.subplot(3, 2, 5)
plt.show()
```



```

# The first subplot
plt.subplot(3, 2, 1)
plt.plot(traffic_per_day['Monday']['Hour (Coded)'],
         traffic_per_day['Monday']['Slowness in traffic (%)'])
plt.title('Monday')
# The second subplot
plt.subplot(3, 2, 2)
plt.plot(traffic_per_day['Tuesday']['Hour (Coded)'],
         traffic_per_day['Tuesday']['Slowness in traffic (%)'])
plt.title('Tuesday')
# The third subplot
plt.subplot(3, 2, 3)
plt.title('Wednesday')
# The rest of the subplots
plt.subplot(3, 2, 4)
plt.subplot(3, 2, 5)
plt.subplot(3, 2, 6)
plt.show()

```

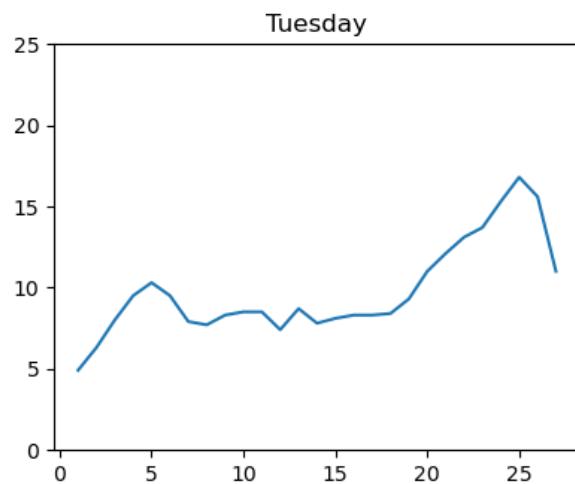
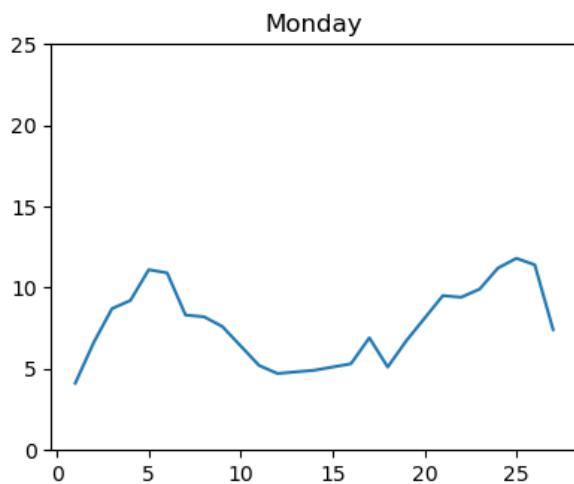


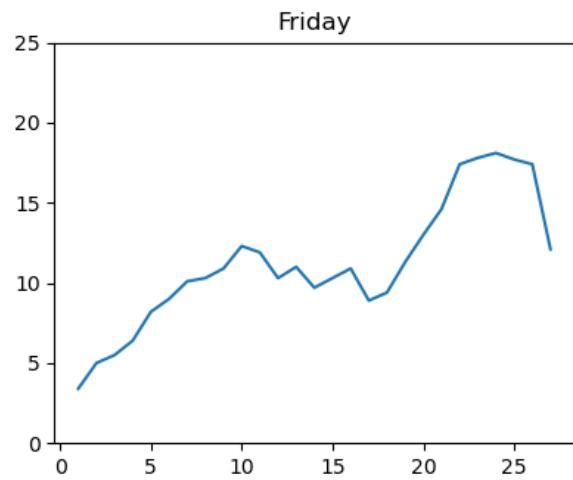
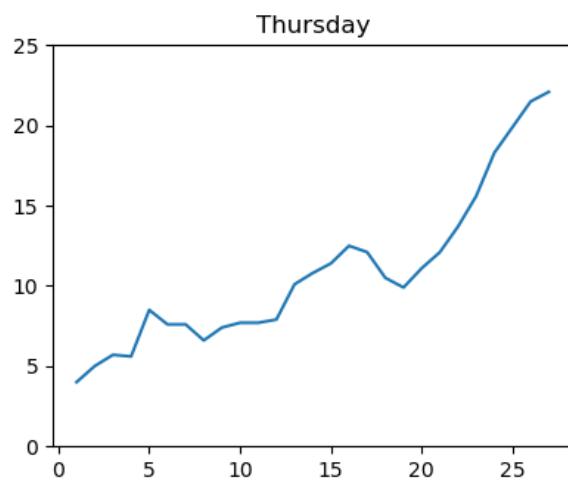
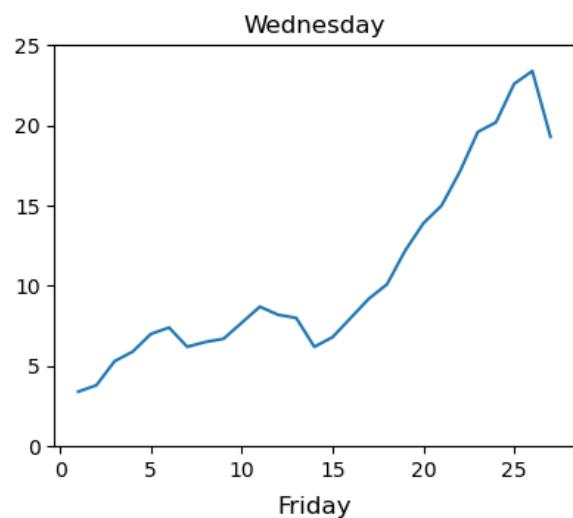
```

import pandas as pd
import matplotlib.pyplot as plt
traffic = pd.read_csv('traffic_sao_paulo.csv', sep=';')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)'].str.replace(',', '.')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)'].astype(float)
days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
traffic_per_day = {}
for i, day in zip(range(0, 135, 27), days):
    each_day_traffic = traffic[i:i+27]
    traffic_per_day[day] = each_day_traffic
plt.figure(figsize=(10,12))
for i, day in zip(range(1,6), days):
    plt.subplot(3, 2, i)
    plt.plot(traffic_per_day[day]['Hour (Coded)'],
            traffic_per_day[day]['Slowness in traffic (%)'])
    plt.title(day)
    plt.ylim([0,25])

plt.show()

```



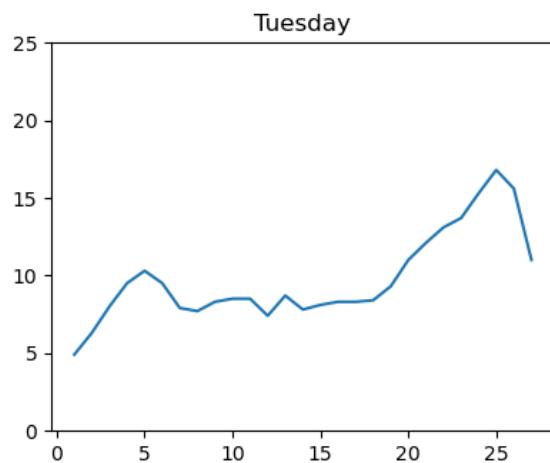
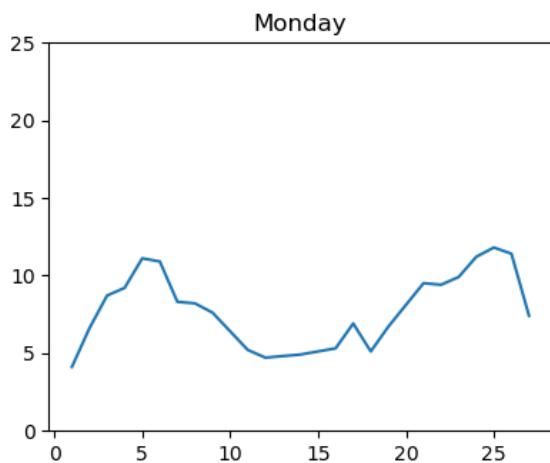


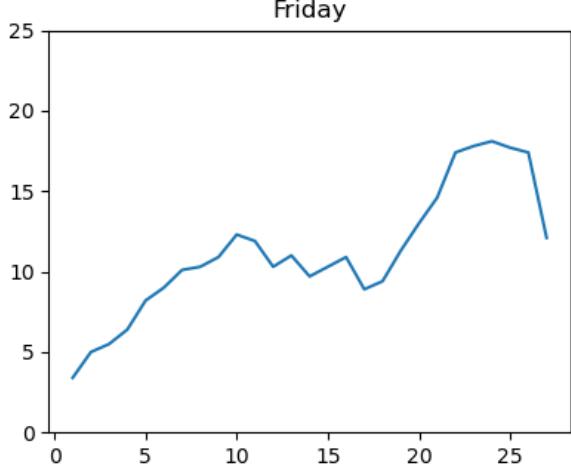
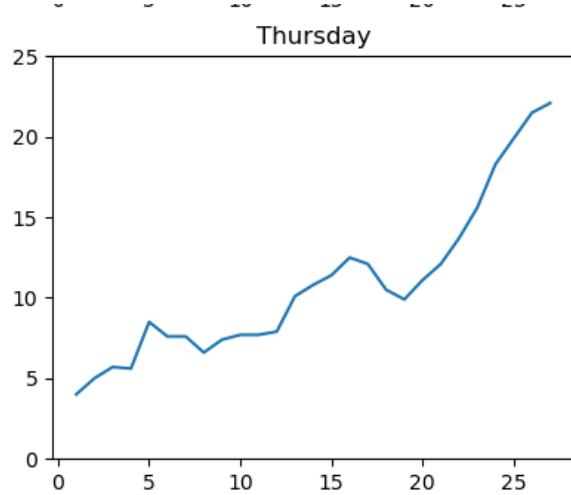
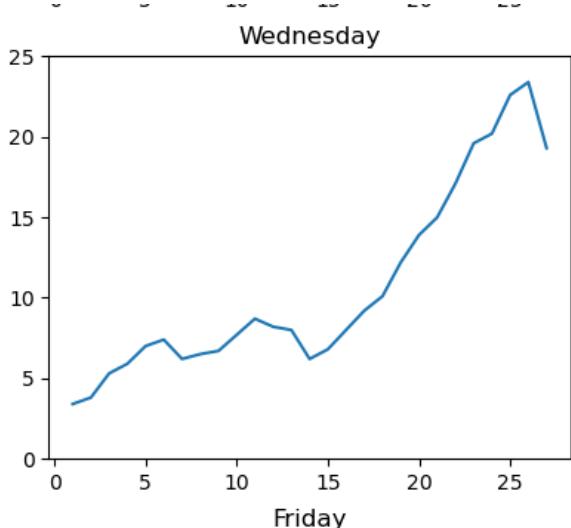
```

import pandas as pd
import matplotlib.pyplot as plt
traffic = pd.read_csv('traffic_sao_paulo.csv', sep=';')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)'].str.replace(',', '.')
traffic['Slowness in traffic (%)'] = traffic['Slowness in traffic (%)'].astype(float)
days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
traffic_per_day = {}
for i, day in zip(range(0, 135, 27), days):
    each_day_traffic = traffic[i:i+27]
    traffic_per_day[day] = each_day_traffic

plt.figure(figsize=(10,12))
for i, day in zip(range(1,6), days):
    plt.subplot(3, 2, i)
    plt.plot(traffic_per_day[day]['Hour (Coded)'],
            traffic_per_day[day]['Slowness in traffic (%)'])
    plt.title(day)
    plt.ylim([0,25])

```





CONTENTS

01. Download webpages with Requests	2
02. Manipulate images with Pillow.....	4
03. Parse HTML with BeautifulSoup	10
04. Make movies with Matplotlib	14
05. Serve webpages with flask.....	16
06. Image-editor challenge	21
07. Image-editor solution.....	24

01. Download webpages with Requests

- Requests: <https://requests.readthedocs.io/en/master/>
- *SDO, a Decade of Sun*: <https://www.youtube.com/watch?v=l3QQQu7QLoM>
- NASA Solar Dynamics Observatory: <https://sdo.gsfc.nasa.gov>

```
import requests
import IPython

request = requests.get('https://sdo.gsfc.nasa.gov/assets/img/browse/2013/09/03/20130903_002924_1024_0171.jpg')

request
<Response [200]>

type(request)
requests.models.Response

request.ok
True
```

- HTTP status codes: https://en.wikipedia.org/wiki/List_of_HTTP_status_codes

```
request.status_code
200

request.url
'https://sdo.gsfc.nasa.gov/assets/img/browse/2013/09/03/20130903_002924_1024_0171.jpg'

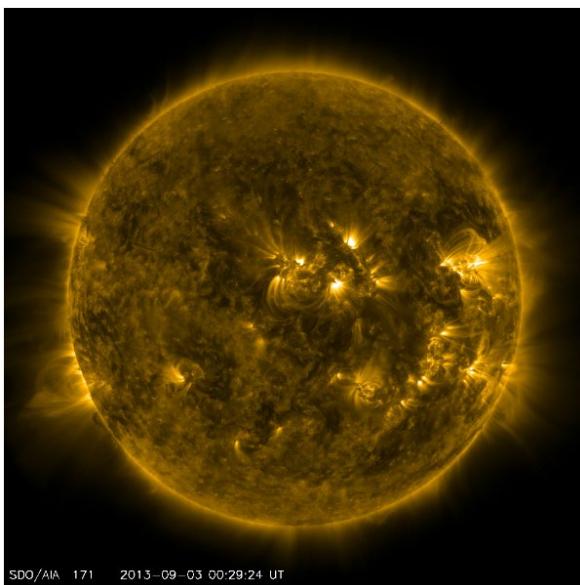
request.elapsed
datetime.timedelta(seconds=2, microseconds=294909)

request.headers
{'Date': 'Thu, 24 Nov 2022 13:06:57 GMT', 'Server': 'Apache', 'X-Frame-Options': 'SAMEORIGIN', 'Strict-Transport-Security': 'max-age=31536000; includeSubDomains;', 'Last-Modified': 'Thu, 28 Jan 2016 18:32:01 GMT', 'ETag': '"2915c-52a69226c2a40"', 'Accept-Ranges': 'bytes', 'Content-Length': '168284', 'Keep-Alive': 'timeout=5, max=100', 'Connection': 'Keep-Alive', 'Content-Type': 'image/jpeg'}
```

```
# don't run this, it will print out gibberish!
request.text
```

request.content

```
IPython.display.Image(data=request.content)
```



```
# Search wikipedia for SDO:  
# https://en.wikipedia.org/w/index.php?search=SDO&sort=relevance  
  
wikipedia = requests.get('https://en.wikipedia.org/w/index.php',  
                         params = {'search': 'SDO', 'sort': 'relevance'})  
  
wikipedia.ok  
True  
  
wikipedia.text
```

<!DOCTYPE html><html class="client-nojs" lang="en" dir="ltr"><head><meta charset="UTF-8"/><title>SDO - Wikipedia</title><script>document.documentElement.className="client-js";RLCONF={"wgBreakFrames":false,"wgSeparatorTransformTable":["","",""], "wgDigitTransformTable":["","",""], "wgDefaultDateFormat":"dmy", "wgMonthNames":["", "January", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December"], "wgRequestId": "66dc690c-a621-40fc-9989-ffb9a96447a3b", "wgCSPNonce":false, "wgCanonicalNamespace": "", "wgCanonicalSpecialPageName":false, "wgNamespaceNumber":0, "wgPageName": "SDO", "wgTitle": "SDO", "wgCurRevisionId":995286672, "wgRevisionId":995286672, "wgArticleId":2504260, "wgIsArticle":true, "wgIsRedirect":false, "wgAction": "view", "wgUserName":null, "wgUserGroups":["*"], "wgCategories": ["Disambiguation pages with short descriptions", "Short description is different from Wikidata", "All article disambiguation pages", "All disambiguation pages", "Disambiguation pages"], "wgPageContentLanguage": "en", "wgPageContentModel": "wikitext", "wgRelevantPageName": "SDO", "wgRelevantArticleId":2504260, \n"wgIsProbablyEditable":true, "wgRelevantPageIsProbablyEditable":true, "wgRestrictionEdit":[], "wgRestrictionMove":[], "wgFlaggedRevsParams": {"tags": {"status": {"levels": 1}}}, "wgVisualEditor": {"pageLanguageCode": "en", "pageLanguageDir": "ltr", "pageVariantFallbacks": "en"}, "wgMFDisplayWikibaseDescriptions": {"search": true, "watchlist": true, "tagline": false, "nearby": true}, "wgWMESchemaEditAttemptStepOversample": false, "wgWMEPageLength": 800, "wgNoticeProject": "wikipedia", "wgVector2022PreviewPages": [], "wgMediaViewerOnClick": true, "wgMediaViewerEnabledByDefault": true, "wgPopupsFlags": 10, "wgULSCurrentAutonym": "English", "wgEditsSubmitButtonLabelPublish": true, "wgCentralAuthMobileDomain": false, "wgULSPosition": "interlanguage", "wgULSIsCompactLinksEnabled": true, "wgWikibaseItemId": "Q349152", "GEHomepageSuggestedEditsEnableTopics": true, "wgGETopicsMatchModeEnabled": false, "wgGetStructuredTaskRejectionReasonTextInputEnabled": false}; RLSTATE={"ext.globalCssJs.user.styles": "ready", "site.styles": "ready", "user.styles": "ready", \n"ext.globalCssJs.user": "ready", "user": "read", "user.options": "loading", "skins.vector.styles.legacy": "ready", "ext.visualEditor.desktopArticleTarget.noscript": "read"}</script>

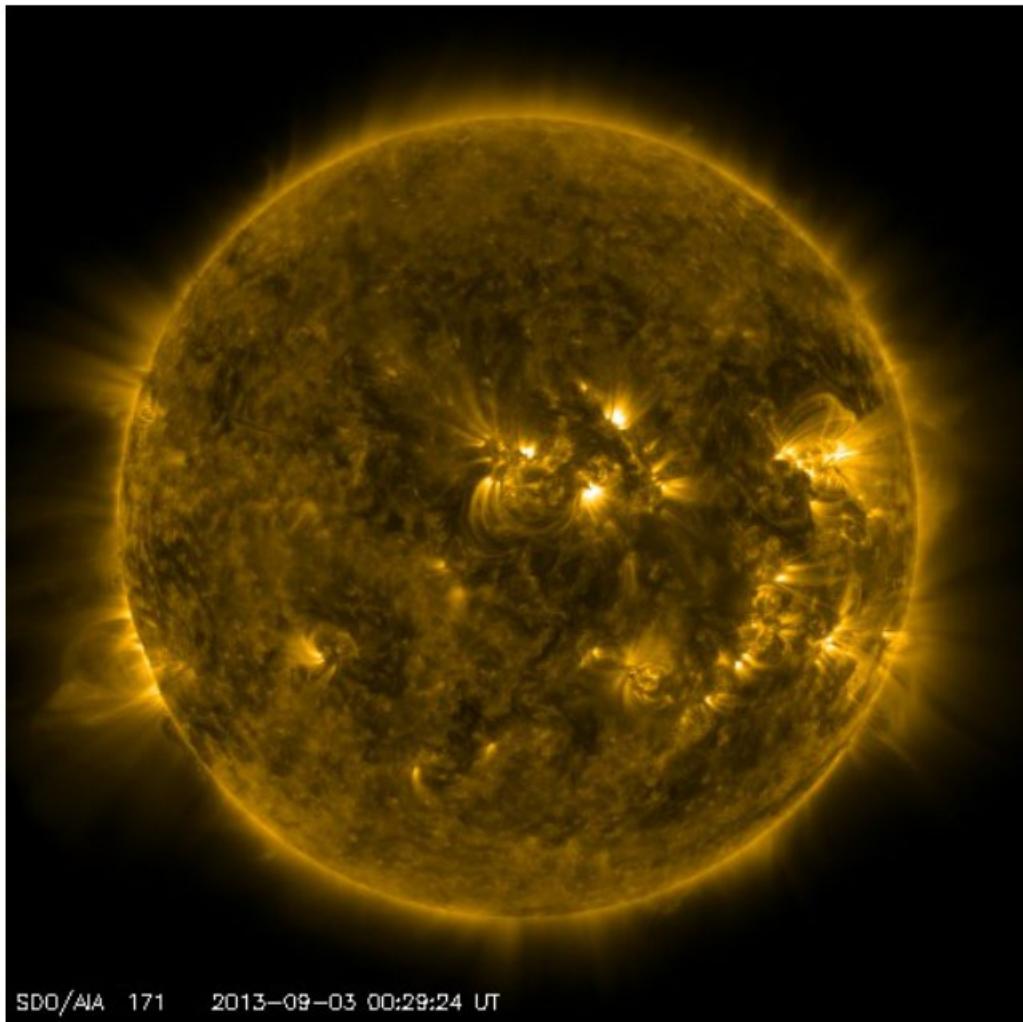
02. Manipulate images with Pillow

- Pillow: <https://pillow.readthedocs.io>

```
# pillow - previously known as Python Imaging Library, keeps the same Python name  
  
import PIL  
import PIL.Image  
import PIL.ImageOps  
import PIL.ImageEnhance  
import PIL.ImageFilter  
  
import requests  
import io  
  
# cached/20130903_002924_512_0171.jpg  
# cached/20130903_082900_512_0171.jpg
```

```
img1 = PIL.Image.open('cached/20130903_002924_512_0171.jpg')
```

```
img1
```

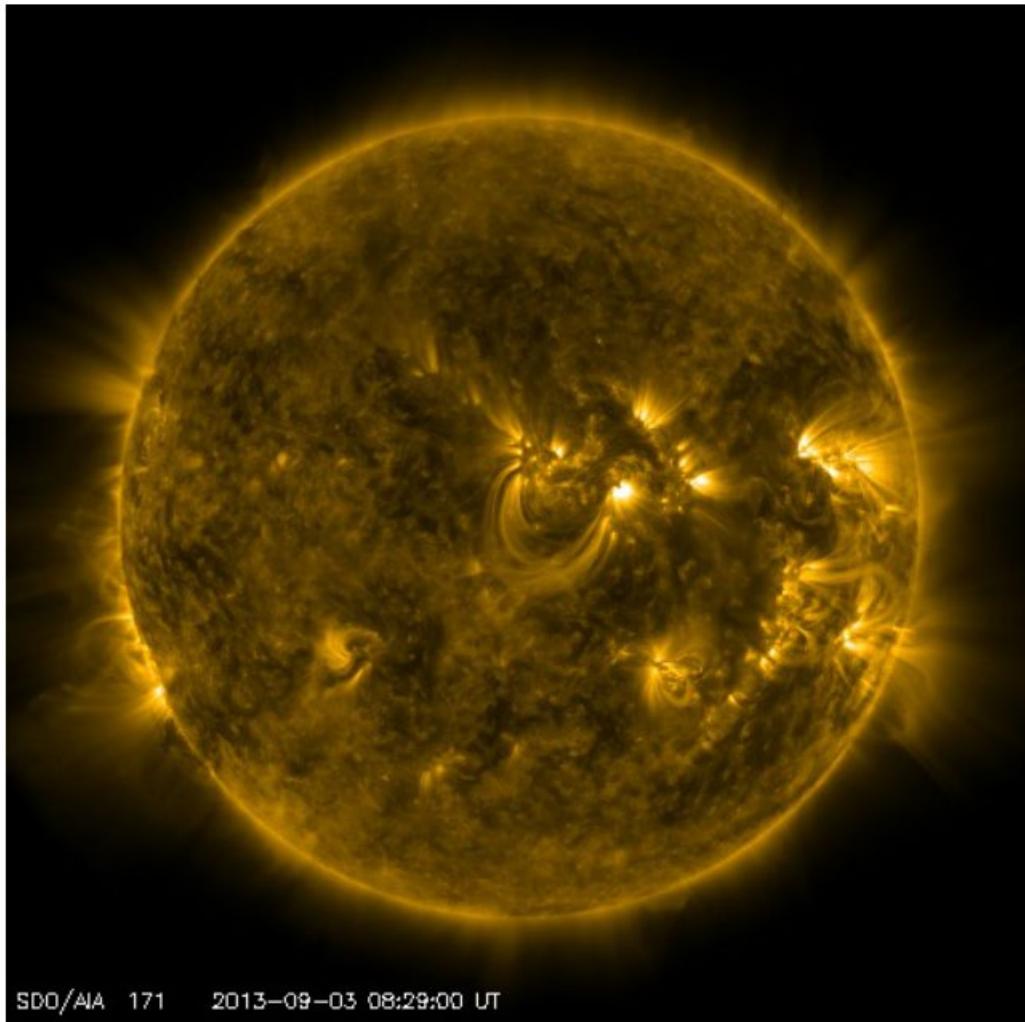


```
type(img1)
```

```
PIL.JpegImagePlugin.JpegImageFile
```

```
req = requests.get('https://sdo.gsfc.nasa.gov/assets/img/browse/2013/09/03  
img2 = PIL.Image.open(io.BytesIO(req.content))
```

```
img2
```

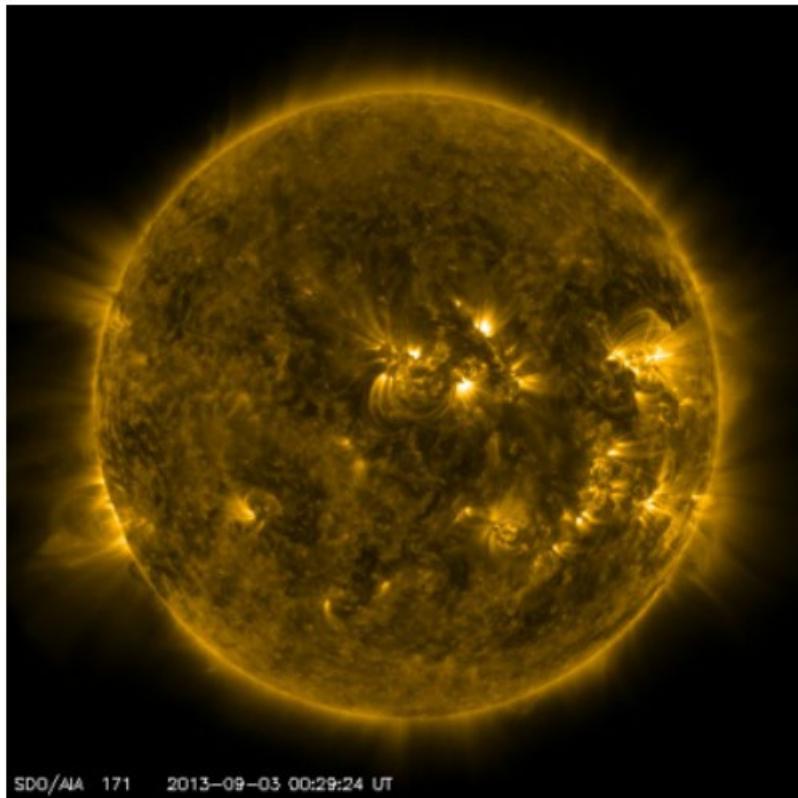


```
img1.size, img1.mode
```

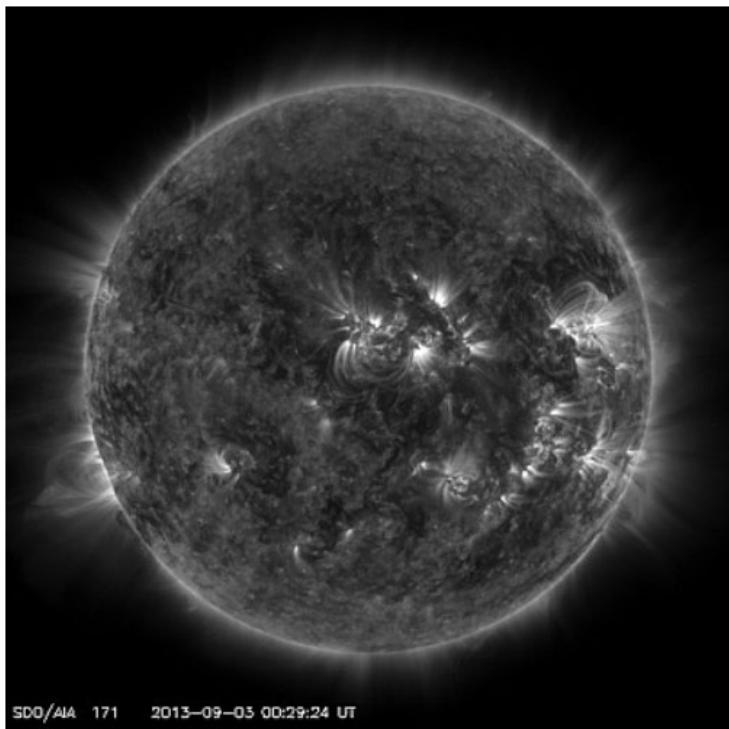
```
((512, 512), 'RGB')
```

```
img1_resized = img1.resize((400,400))  
img2_resized = img2.resize((400,400))
```

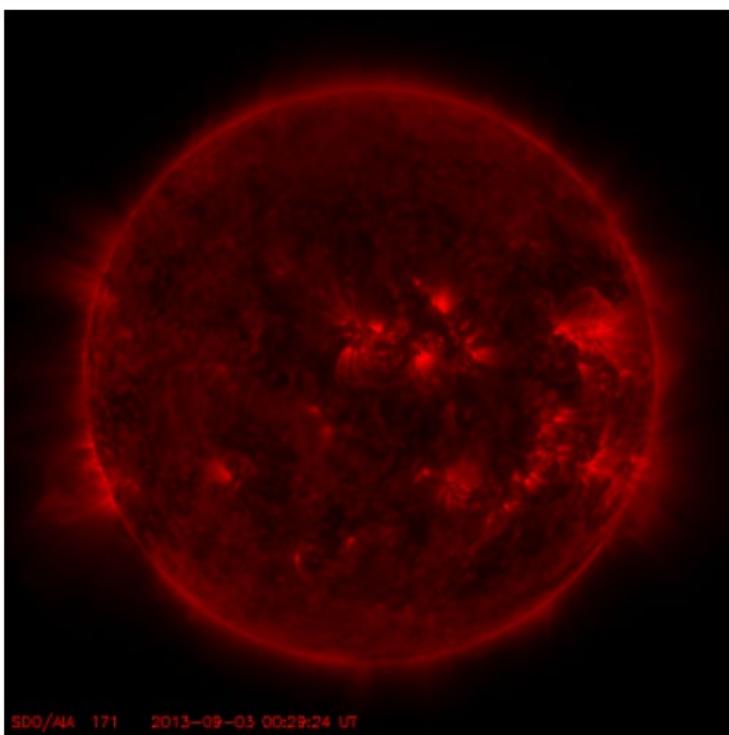
```
img1_resized
```



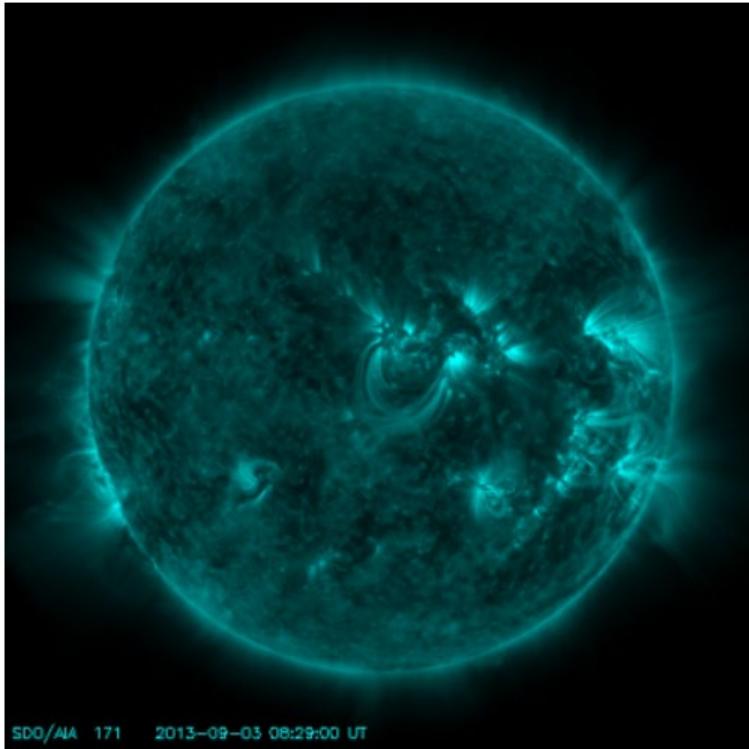
```
img1_resized.convert('L').filter(PIL.ImageFilter.SHARPEN)
```



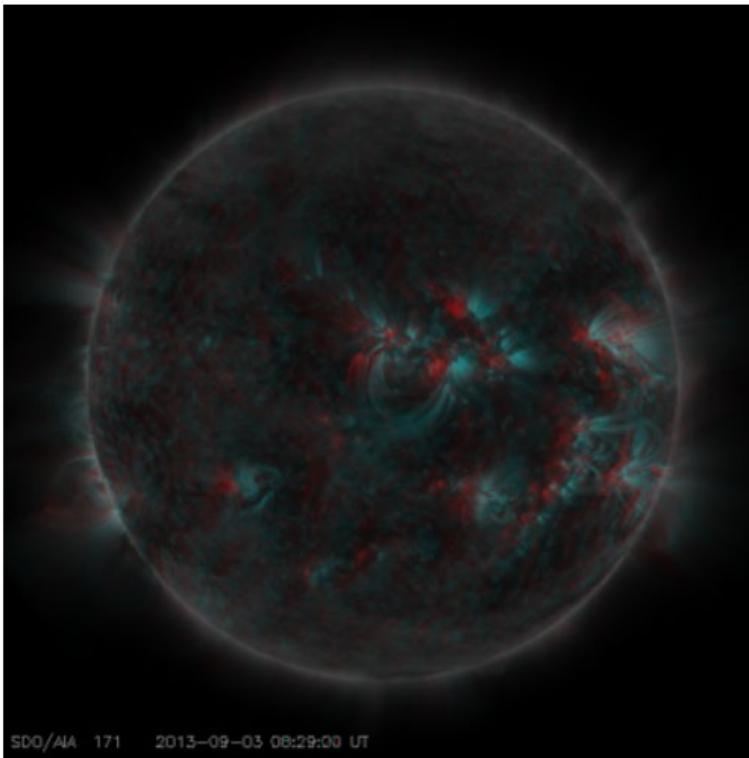
```
img1_red = PIL.ImageOps.colorize(img1_resized.convert('L'), (0,0,0), (255,0,0))  
img1_red
```



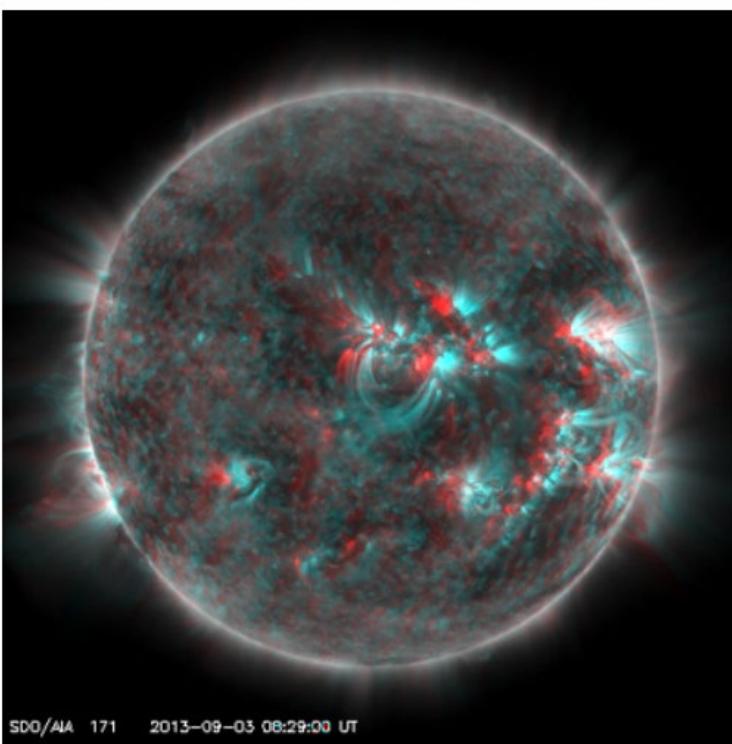
```
img2_cyan = PIL.ImageOps.colorize(img2_resized.convert('L'), (0,0,0), (0,255,255))
img2_cyan
```



```
blend = PIL.Image.blend(img1_red, img2_cyan, 0.5)
blend
```



```
PIL.ImageEnhance.Brightness(blend).enhance(2.5)
```



03. Parse HTML with BeautifulSoup

- SDO image index for June 6, 2012: <https://sdo.gsfc.nasa.gov/assets/img/browse/2012/06/06>

```
print(open('cached/index_20120606.html', 'r').read())

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
  <head>
    <title>Index of /assets/img/browse/2012/06/06</title>
  </head>
  <body>
<h1>Index of /assets/img/browse/2012/06/06</h1>
<pre>      <a href="?C=N;O=D">Name</a>          <a href="?C=M;O=A">Last modified</a>      <a href=?C=S;O=A">Size</a>  <a href="?C=D;O=A">Description</a><hr>      <a href="/assets/img/browse/2012/06/">Parent Directory</a>

      <a href="20120606_000008_1024_0193.jpg">20120606_000008_1024_0193.jpg</a> 2018-09-12 01:20 142K
      <a href="20120606_000008_2048_0193.jpg">20120606_000008_2048_0193.jpg</a> 2018-09-12 01:19 513K
      <a href="20120606_000008_4096_0193.jpg">20120606_000008_4096_0193.jpg</a> 2018-09-12 01:19 2.2M
      <a href="20120606_000008_512_0193.jpg">20120606_000008_512_0193.jpg</a> 2018-09-12 01:20 46K
      <a href="20120606_000009_1024_0304.jpg">20120606_000009_1024_0304.jpg</a> 2018-09-12 01:20 255K
      <a href="20120606_000009_2048_0304.jpg">20120606_000009_2048_0304.jpg</a> 2018-09-12 01:19 961K
      <a href="20120606_000009_4096_0304.jpg">20120606_000009_4096_0304.jpg</a> 2018-09-12 01:20 4.2M
      <a href="20120606_000009_512_0304.jpg">20120606_000009_512_0304.jpg</a> 2018-09-12 01:20 73K
      <a href="20120606_000011_1024_0131.jpg">20120606_000011_1024_0131.jpg</a> 2018-09-12 01:18 248K
      <a href="20120606_000011_2048_0131.jpg">20120606_000011_2048_0131.jpg</a> 2018-09-12 01:18 1.2M
```

- BeautifulSoup: <https://www.crummy.com/software/BeautifulSoup>

```
import bs4

# If the `lxml` parser is not available with `BeautifulSoup`,
# it can be installed separately (e.g., `conda install lxml`)

soup = bs4.BeautifulSoup(open('cached/index_20120606.html','r'), 'lxml')

type(soup)
bs4.BeautifulSoup

soup.head
<head>
<title>Index of /assets/img/browse/2012/06/06</title>
</head>

soup.title
<title>Index of /assets/img/browse/2012/06/06</title>

type(soup.title)
bs4.element.Tag

soup.head.name, soup.head.contents
('head', ['\n', <title>Index of /assets/img/browse/2012/06/06</title>, '\n'])

soup.title.string
'Index of /assets/img/browse/2012/06/06'

soup.a.attrs
{'href': '?C=N;O=D'}

soup.a['href']
'?C=N;O=D'

soup.head.contents
['\n', <title>Index of /assets/img/browse/2012/06/06</title>, '\n']

soup.head.children
<list_iterator at 0x2d3d57f6670>

list(soup.head.children)
['\n', <title>Index of /assets/img/browse/2012/06/06</title>, '\n']
```

```
soup.title.parent  
<head>  
<title>Index of /assets/img/browse/2012/06/06</title>  
</head>
```

```
soup.title.parents  
<generator object PageElement.parents at 0x000002D3D72FE740>
```

```
soup.pre.find_all('a')  
[<a href="?C=N;O=D">Name</a>,  
<a href="?C=M;O=A">Last modified</a>,  
<a href="?C=S;O=A">Size</a>,  
<a href="?C=D;O=A">Description</a>,  
<a href="/assets/img/browse/2012/06/">Parent Directory</a>,  
<a href="20120606_000008_1024_0193.jpg">20120606_000008_1024_0193.jpg</a>,  
<a href="20120606_000008_2048_0193.jpg">20120606_000008_2048_0193.jpg</a>,  
<a href="20120606_000008_4096_0193.jpg">20120606_000008_4096_0193.jpg</a>,  
<a href="20120606_000008_512_0193.jpg">20120606_000008_512_0193.jpg</a>,  
<a href="20120606_000009_1024_0304.jpg">20120606_000009_1024_0304.jpg</a>,  
<a href="20120606_000009_2048_0304.jpg">20120606_000009_2048_0304.jpg</a>,  
<a href="20120606_000009_4096_0304.jpg">20120606_000009_4096_0304.jpg</a>,  
<a href="20120606_000009_512_0304.jpg">20120606_000009_512_0304.jpg</a>,  
<a href="20120606_000011_1024_0131.jpg">20120606_000011_1024_0131.jpg</a>,  
<a href="20120606_000011_2048_0131.jpg">20120606_000011_2048_0131.jpg</a>,  
<a href="20120606_000011_4096_0131.jpg">20120606_000011_4096_0131.jpg</a>,  
<a href="20120606_000011_512_0131.jpg">20120606_000011_512_0131.jpg</a>,  
<a href="20120606_000013_1024_0171.jpg">20120606_000013_1024_0171.jpg</a>,  
<a href="20120606_000013_2048_0171.jpg">20120606_000013_2048_0171.jpg</a>,
```

```
[link.string for link in soup.pre.find_all('a') if link.string.endswith('512_0171.jpg')]

['20120606_000013_512_0171.jpg',
 '20120606_001437_512_0171.jpg',
 '20120606_003037_512_0171.jpg',
 '20120606_004501_512_0171.jpg',
 '20120606_005925_512_0171.jpg',
 '20120606_011525_512_0171.jpg',
 '20120606_012949_512_0171.jpg',
 '20120606_014549_512_0171.jpg',
 '20120606_020013_512_0171.jpg',
 '20120606_021437_512_0171.jpg',
 '20120606_022901_512_0171.jpg',
 '20120606_024501_512_0171.jpg',
 '20120606_030101_512_0171.jpg',
 '20120606_031525_512_0171.jpg',
 '20120606_032949_512_0171.jpg',
 '20120606_034413_512_0171.jpg',
 '20120606_040013_512_0171.jpg',
 '20120606_041437_512_0171.jpg',
 '20120606_043037_512_0171.jpg',
 '20120606_044501_512_0171.jpg',
 '20120606_045925_512_0171.jpg',
 '20120606_051525_512_0171.jpg',
 '20120606_052949_512_0171.jpg',
 '20120606_054237_512_0171.jpg',
 '20120606_060037_512_0171.jpg',
 '20120606_061525_512_0171.jpg',
 '20120606_063025_512_0171.jpg',
 '20120606_064513_512_0171.jpg',
 '20120606_065937_512_0171.jpg',
 '20120606_071549_512_0171.jpg',
 '20120606_073037_512_0171.jpg',
```

04. Make movies with Matplotlib

```
import matplotlib
import matplotlib.pyplot as pp
import matplotlib.animation as anim

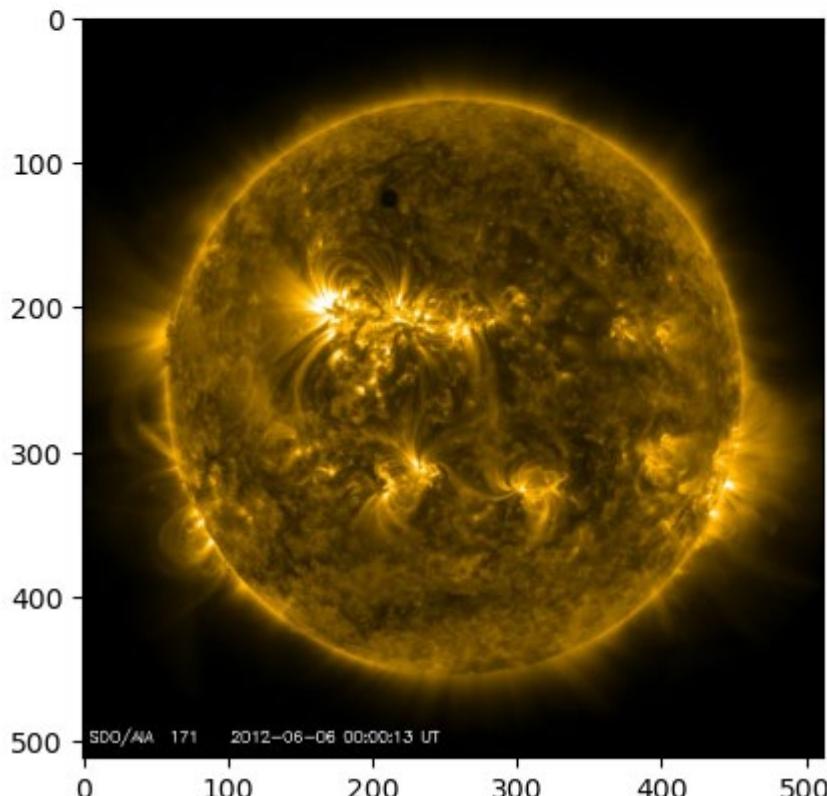
import PIL
import PIL.Image

import bs4

frame = PIL.Image.open('frames/20120606_000013_512_0171.jpg', 'r')

pp.imshow(frame)

<matplotlib.image.AxesImage at 0x1e6266b1fd0>
```

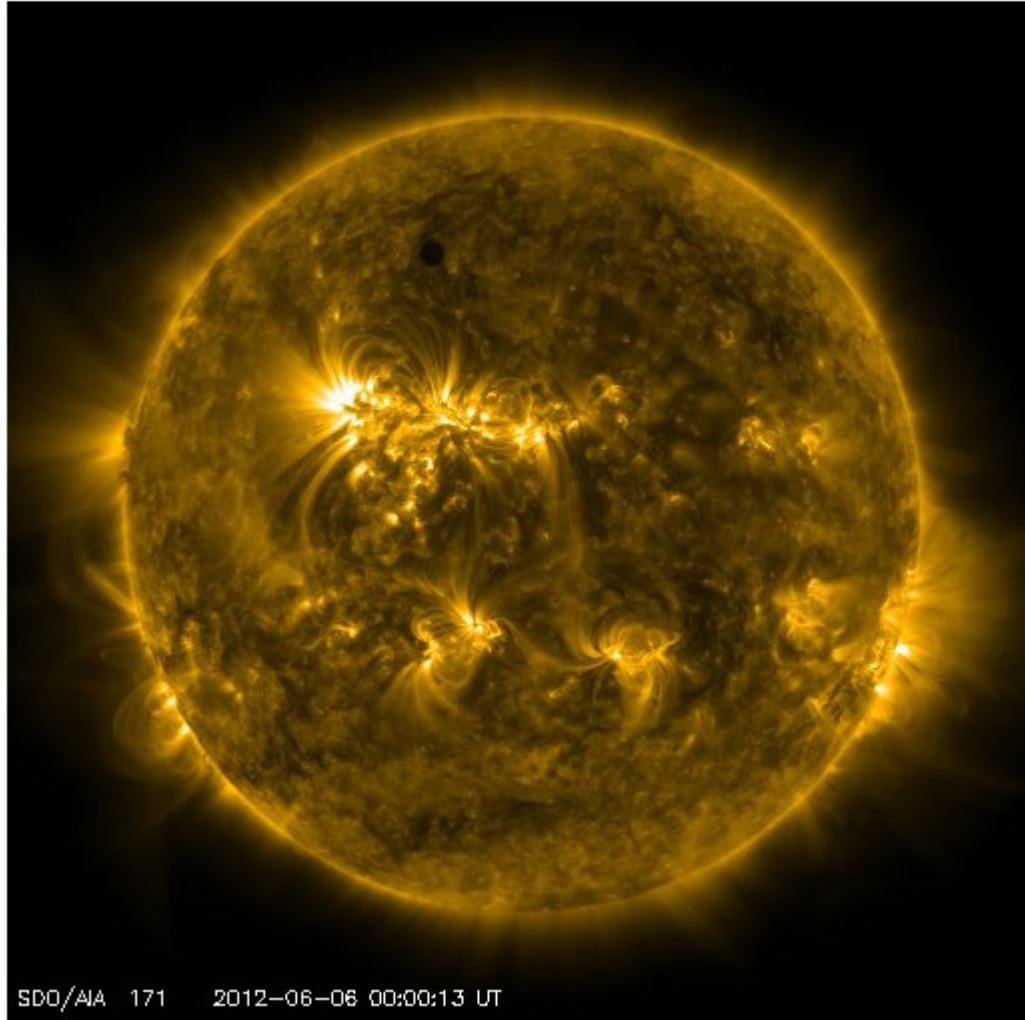


```
figure = pp.figure(figsize=(512/72, 512/72), dpi=72)

axes = pp.Axes(figure, [0., 0., 1., 1.])
axes.set_axis_off()
figure.add_axes(axes)

pp.imshow(frame)

<matplotlib.image.AxesImage at 0x1e626eb21f0>
```



05. Serve webpages with flask

```
import io
import base64

import requests
import bs4

import PIL.Image
import PIL.ImageOps
import PIL.ImageEnhance

def getstereo(date):
    # get SDO index for date, find all 512x512 171-angstrom images
    indexreq = requests.get(f'https://sdo.gsfc.nasa.gov/assets/img/browse/{date}')
    soup = bs4.BeautifulSoup(indexreq.content, 'lxml')
    filenames = [link.string for link in soup.pre.find_all('a')
                 if link.string.endswith('512_0171.jpg')]

    # get two images across the day
    images = []
    # note integer division in case there's an odd number
    for filename in [filenames[0], filenames[len(filenames)//2]]:
        imagereq = requests.get(f'https://sdo.gsfc.nasa.gov/assets/img/browse/{date}/{filename}')
        images.append(PIL.Image.open(io.BytesIO(imagereq.content)))

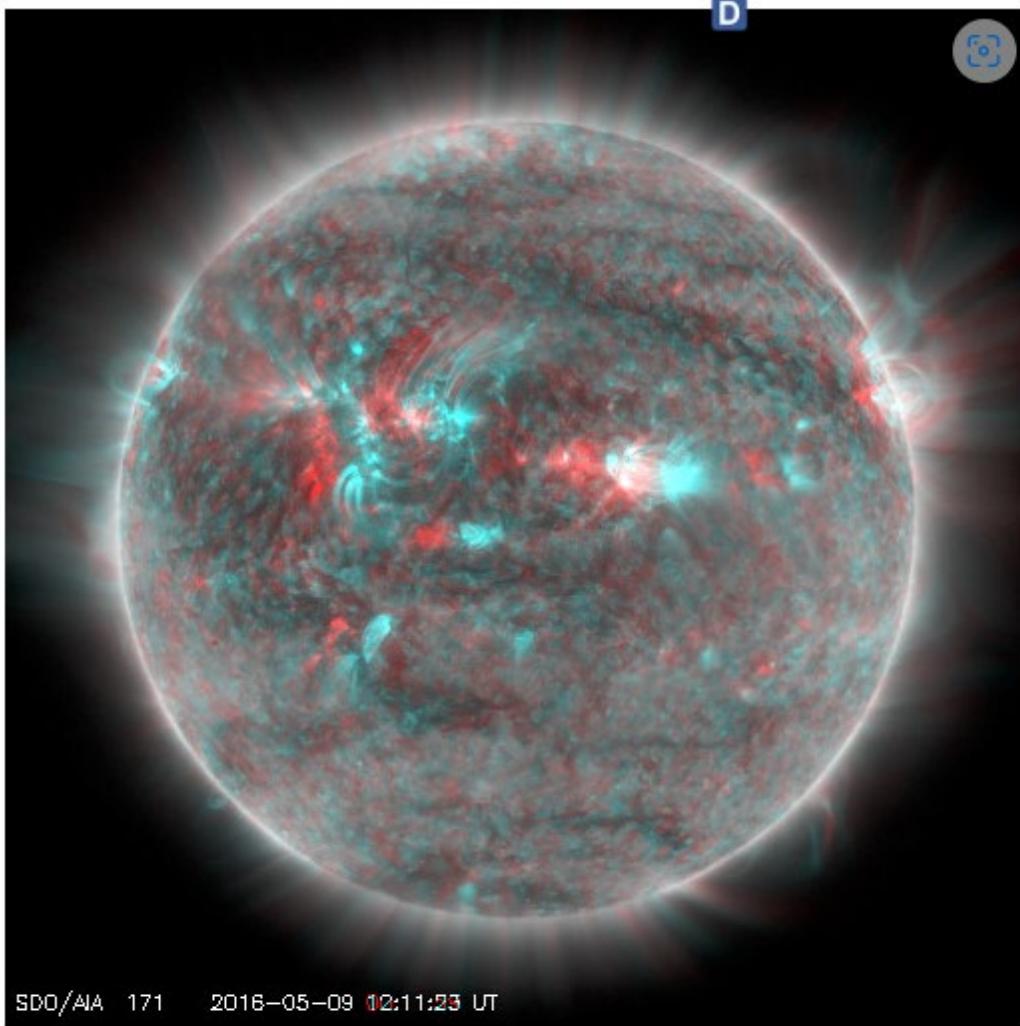
    # grayscale + colorize the two images
    red = PIL.ImageOps.colorize(images[0].convert('L'), (0,0,0), (255,0,0))
    cyan = PIL.ImageOps.colorize(images[1].convert('L'), (0,0,0), (0,255,255))

    # blend and adjust brightness
    blend = PIL.Image.blend(red, cyan, 0.5)
    final = PIL.ImageEnhance.Brightness(blend).enhance(2.5)

    return final

img = getstereo('2016/05/09')

img
```



- `getstereo.py` contains this code.
- Use `getstereo.getcached()` if `getstereo.getstereo()` fails because of web problems

```
%%file server.py

import flask

app = flask.Flask(__name__)

@app.route('/')
def hello_world():
    return flask.Response('<html><body><p>Hello, world!</p></body></html>')

app.run(host='0.0.0.0')
```

Writing server.py

```
%%file server.py

import flask

import getstereo

app = flask.Flask(__name__)

@app.route('/<year>/<month>/<day>')
def getimage(year, month, day):
    stereo = getstereo.getstereo(f'{year}/{month}/{day}')
    return flask.Response("Got it!")

app.run(host='0.0.0.0')
```

Overwriting server.py

```
buffer = io.BytesIO()
img.save(buffer, format='PNG')
```

```
buffer.getvalue()[:100]
```

b'\x89PNG\r\n\x1a\n\x00\x00\x00\x00rIHDR\x00\x00\x02\x00\x00\x00\x02\x00\x08\x02\x00\x00\x00{\x1aC\xad\x00\x01\x00\x00IDATx\x9c\xec\xfdm\x8b,I\xae-\x0c\xea\x91F\xb8\xe3d\x92t\xd3\xdc\xa6\x99\x87\x81\xf9\xff\xff\xfh'\xbe\x1c\xe6r\xb8\x87\xa2\x8a\$\'\x02\xc3\x0c!\'\x1f\x96In\xee\x91\xbb\xba\xba0\xf7\xe9\xaa\xae'm

```

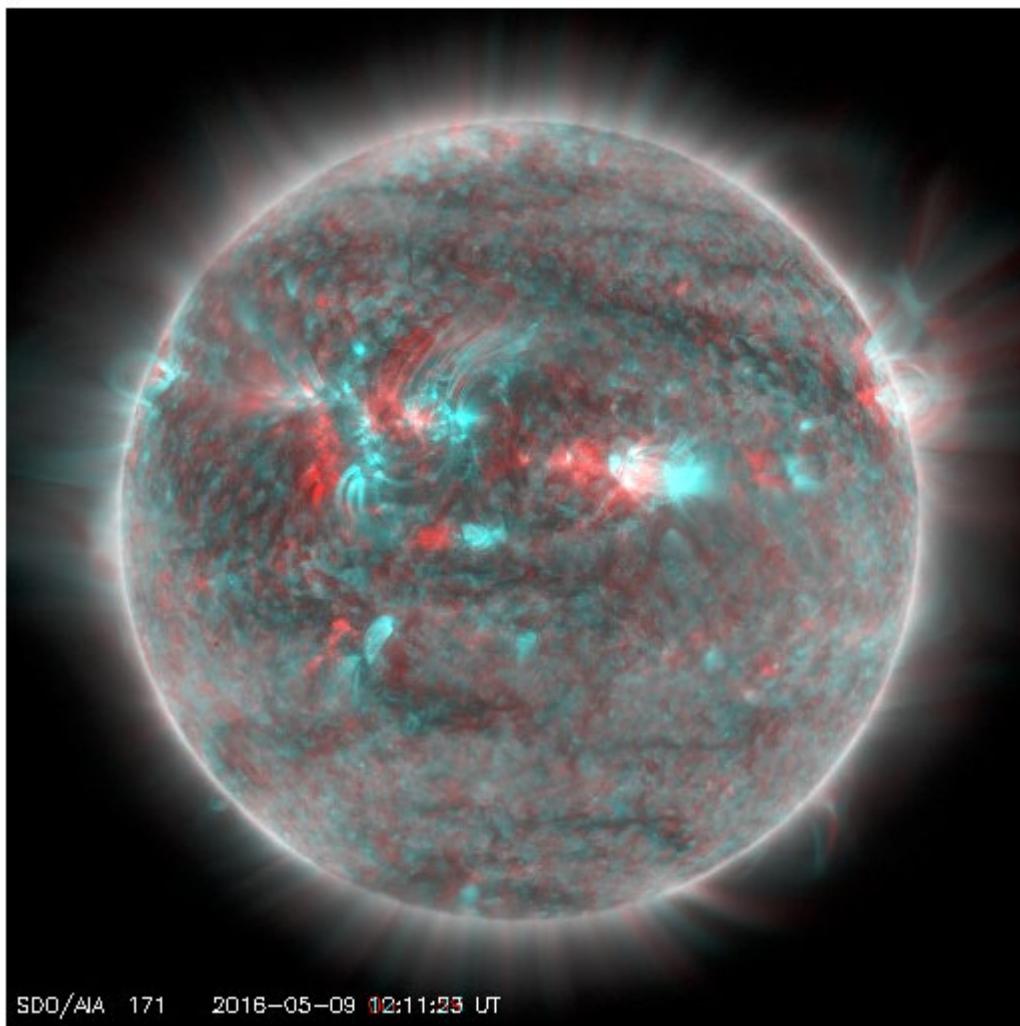
base64.b64encode(buffer.getvalue())[:100]

b'iVBORw0KGgoAAAANSUhEUgAAAgAAAAIACAIAAAAB7GkOtAAEAAE1EQVR4nOz9bYssSa4tD0qRRrjjZJJ009ymmmYeB+f//aGC+HOZ
y'

b64image = base64.b64encode(buffer.getvalue()).decode('ascii')

f''


')
```



```
%%file server.py

import flask

import getstereo

app = flask.Flask(__name__)

@app.route('/<year>/<month>/<day>')
def getimage(year, month, day):
    stereo = getstereo.getstereo(f'{year}/{month}/{day}')
    inline = getstereo.makebase64(stereo)

    return flask.Response(f'<html><body>{inline}</body></html>')

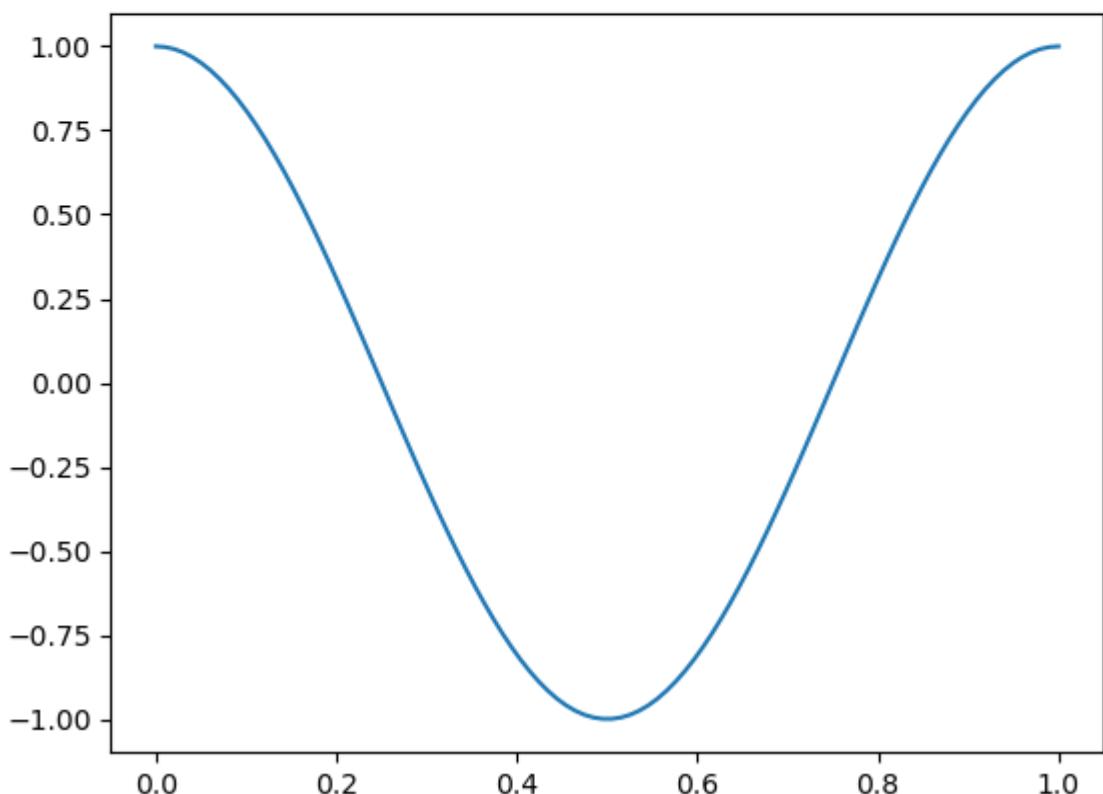
app.run(host='0.0.0.0')
```

Overwriting server.py

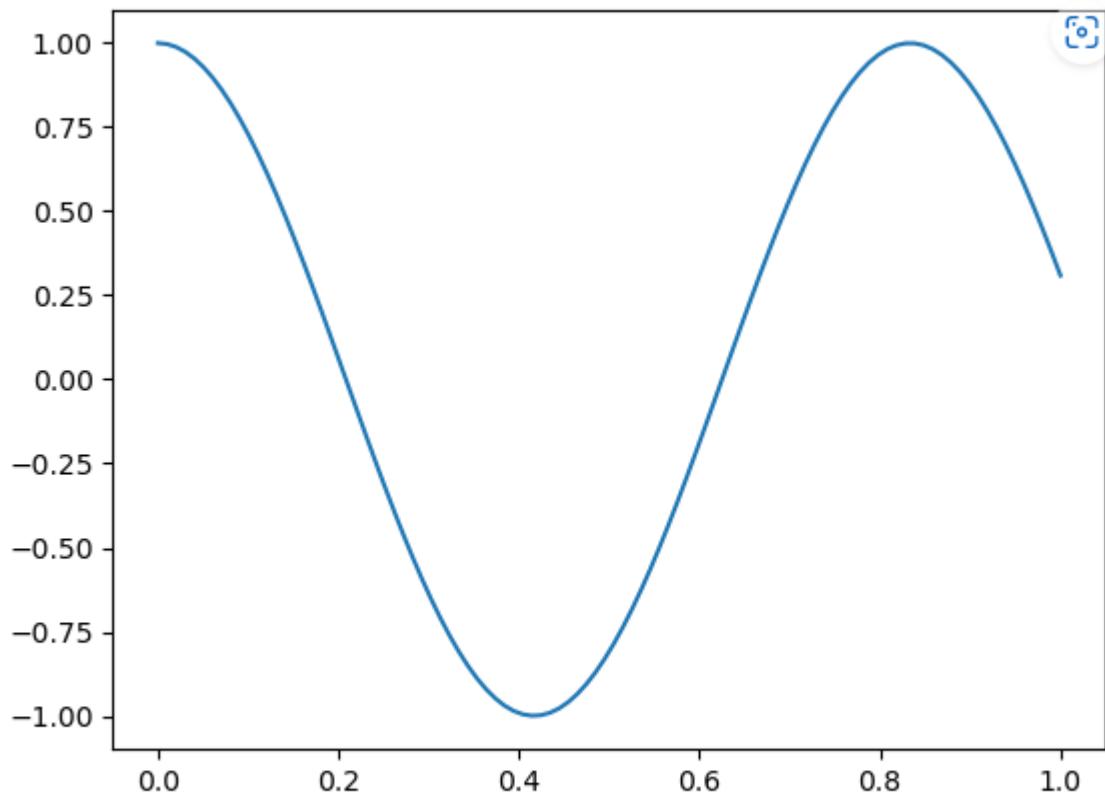
06. Image-editor challenge

```
import math  
  
import numpy as np  
import matplotlib.pyplot as pp  
  
x = np.linspace(0, 1, 100)  
  
def plotcos(f):  
    pp.plot(x, np.cos(2*math.pi*f*x))
```

```
plotcos(1)
```



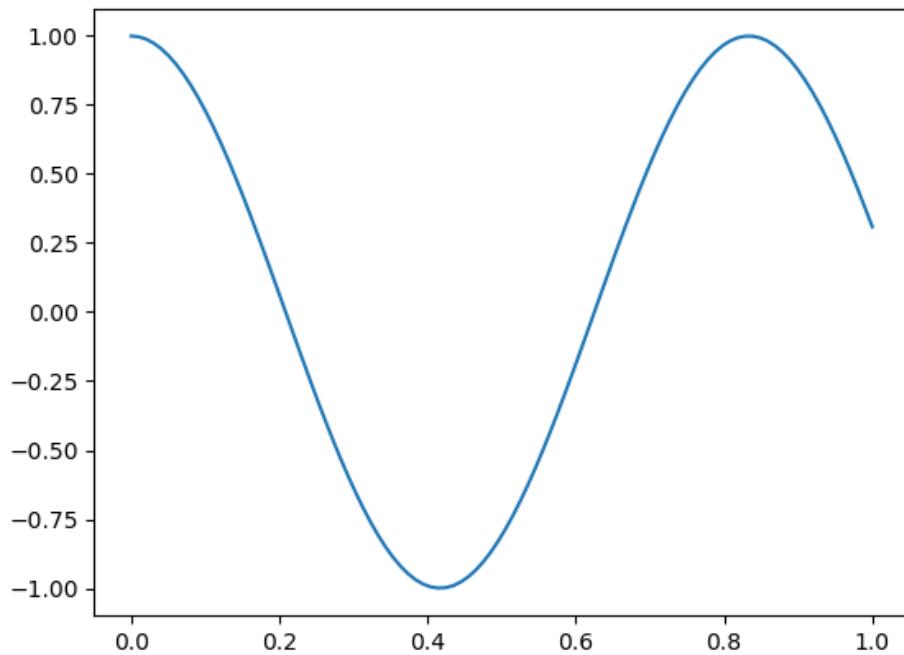
```
plotcos(1.2)
```



- ipywidgets: <https://ipywidgets.readthedocs.io>

```
import ipywidgets  
  
f_control = ipywidgets.FloatSlider(value=1, min=0.5, max=2.0, step=0.05, continuous_update=False)  
  
ipywidgets.interact(plotcos, f=f_control)
```

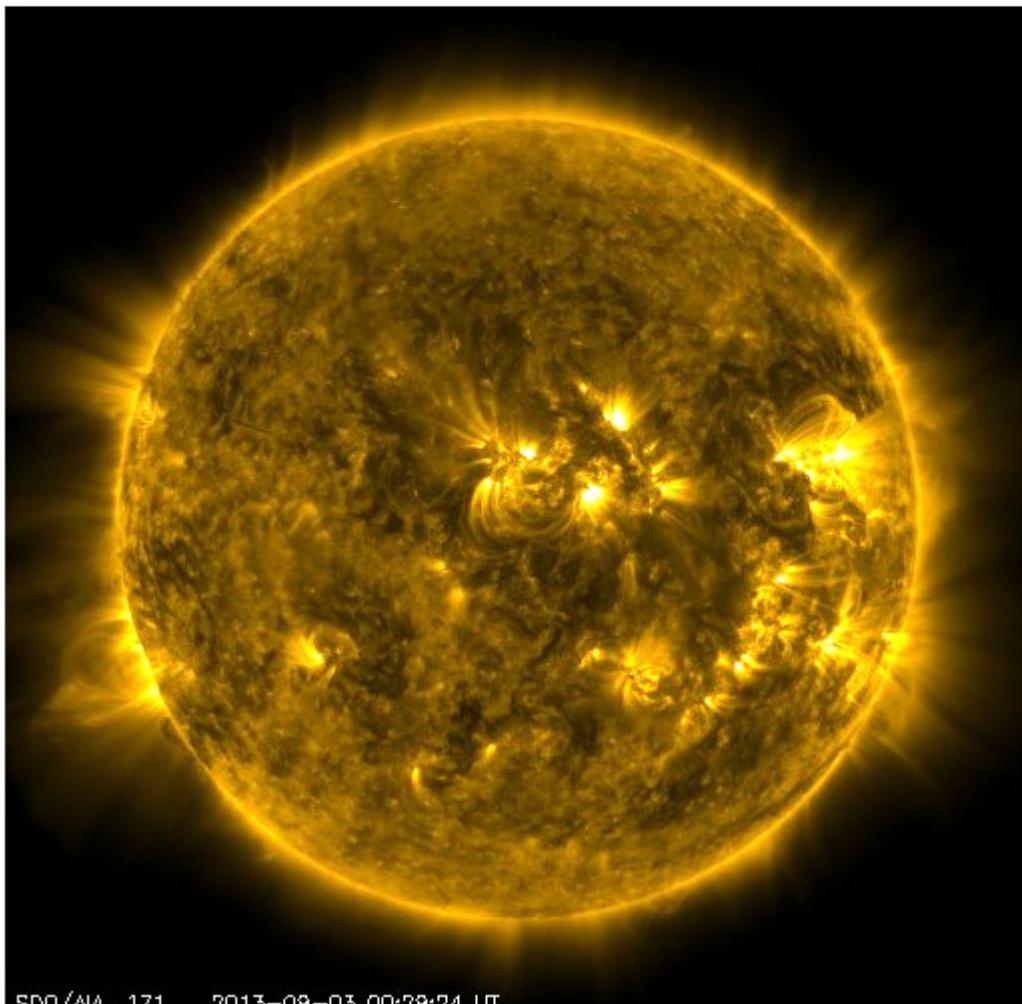
f  1.20



```
<function __main__.plotcos(f)>
```

07. Image-editor solution

```
import PIL  
import PIL.Image  
import PIL.ImageOps  
import PIL.ImageEnhance  
import PIL.ImageFilter  
  
import ipywidgets  
  
img = PIL.Image.open('cached/20130903_002924_512_0171.jpg')  
  
PIL.ImageEnhance.Brightness(img).enhance(1.5)
```



```
def adjust(color, contrast, brightness):
    return PIL.ImageEnhance.Color(
        PIL.ImageEnhance.Contrast(
            PIL.ImageEnhance.Brightness(img).enhance(brightness)
        ).enhance(contrast)
    ).enhance(color)

color_control = ipywidgets.FloatSlider(min=0.1, max=4, step=0.05, value=1, continuous_update=False)
contrast_control = ipywidgets.FloatSlider(min=0.1, max=4, step=0.05, value=1, continuous_update=False)
brightness_control = ipywidgets.FloatSlider(min=0.1, max=4, step=0.05, value=1, continuous_update=False)

ipywidgets.interact(adjust, color=color_control,
                    contrast=contrast_control,
                    brightness=brightness_control)
```

```
ipywidgets.interact(adjust, color=color_control,  
                    contrast=contrast_control,  
                    brightness=brightness_control)
```



```
<function __main__.adjust(color, contrast, brightness)>
```

Báo cáo cuối kỳ

1. Giới thiệu đề tài	2
2. Lấy dữ liệu từ API	3
3. Xử lý và phân tích dữ liệu	5
3.1. Xử lý dữ liệu	5
3.2. Phân tích dữ liệu	8
4. Kết quả	12

1. Giới thiệu đề tài

Ngày nay, dữ liệu thời tiết khá là cần thiết cho cuộc sống của con người, việc thống kê và xử lý chúng còn ít rắc rối, lý do em chọn đề tài này vì muốn người xem có thể dễ dàng xem và hiểu từ dữ liệu được trực quan hóa.

- Tên đề tài: Weather Statistics In Any City
- Ngôn ngữ dùng để xử lý: Python
- Nguồn dữ liệu: API

Một số khái niệm.

API là các phương thức, giao thức kết nối với các thư viện và ứng dụng khác. Nó là viết tắt của Application Programming Interface – giao diện lập trình ứng dụng. API cung cấp khả năng cung cấp khả năng truy xuất đến một tập các hàm hay dùng. Và từ đó có thể trao đổi dữ liệu giữa các ứng dụng.

API hiện nay đều tuân thủ theo tiêu chuẩn REST và HTTP, tạo sự thân thiện dễ sử dụng với nhà phát triển. Giúp người dùng dễ dàng truy cập, dễ hiểu hơn. Web API hiện đại dùng cho các đối tượng cụ thể, chẳng hạn như mobile developer với document, version khác nhau.

API key: Đây là loại code (string) được truyền tải bởi các chương trình máy tính gọi là API để xác định chương trình, nhà phát triển hoặc người dùng nó tới trang web. Các API key được sử dụng với mục đích nhằm giới hạn, kiểm soát sử dụng API. Chẳng hạn như ngăn chặn sự việc lạm dụng API.

- Phương thức hoạt động của web API

Đầu tiên là xây dựng URL API để bên thứ ba có thể gửi request dữ liệu đến máy chủ cung cấp nội dung, dịch vụ thông qua giao thức HTTP hoặc HTTPS.

Tại web server cung cấp nội dung, các ứng dụng nguồn sẽ thực hiện kiểm tra xác thực nếu có và tìm đến tài nguyên thích hợp để tạo nội dung trả về kết quả.

Server trả về kết quả theo định dạng JSON hoặc XML thông qua giao thức HTTP/HTTPS.

Tại nơi yêu cầu ban đầu là ứng dụng web hoặc ứng dụng di động , dữ liệu JSON/XML sẽ được parse để lấy data. Sau khi có được data thì thực hiện tiếp các hoạt động như lưu dữ liệu xuống Cơ sở dữ liệu, hiển thị dữ liệu...

- Những ưu điểm đáng chú ý của API

API được sử dụng trên hầu hết những ứng dụng của desktop, ứng dụng mobile và các ứng dụng website.

Nó linh hoạt với các định dạng dữ liệu khác nhau khi trả về client.

Với API, mọi người có thể nhanh chóng xây dựng HTTP service khiến công việc lập trình trở nên đơn giản hơn.

Nó sử dụng mã nguồn mở, có chức năng RESTful đầy đủ. Nhờ vậy, có thể sử dụng bởi bất kỳ client nào hỗ trợ Json, XML quen thuộc như trước.

Có thể giao tiếp 2 chiều, được xác nhận trong các giao dịch khác nhau. Từ đó đảm bảo có được độ tin cậy cao.

API có khả năng hỗ trợ đầy đủ các thành phần MVC như Unit Test, Model Binder, Controller, Action,...

- Nhược điểm của API

Dù có nhiều ưu điểm, API vẫn còn khá mới. Những người dùng chưa đánh giá được nhiều về nhược điểm của nó. Tuy nhiên, bạn có thể nhìn thấy những nhược điểm dưới đây:

Website API chưa hoàn toàn là RESTful service như thông thường. Nó mới chỉ hỗ trợ mặc định GET, POST mà thôi.

Để dùng được hiệu quả, mọi người cần có kiến thức chuyên sâu, có kinh nghiệm backend tốt nếu không sẽ khó lòng tận dụng triệt để những lợi thế, tính năng mà API có.

Phát triển, nâng cấp hay vận hành API là một quá trình lâu dài và khó khăn. Thậm chí, tiêu tốn khá nhiều chi phí của người vận hành.

Nếu hệ thống bị tấn công trong khi chủ sở hữu chưa giới hạn điều kiện kỹ, việc bảo mật sẽ rất khó khăn.

2. Lấy dữ liệu từ API

Truy cập trang [Historical weather API - OpenWeatherMap](#) và lấy link API về.

API call

```
https://history.openweathermap.org/data/2.5/history/city?lat={lat}&lon={lon}&type=hour&start={start}&end={end}&appid={API key}
```



```
https://history.openweathermap.org/data/2.5/history/city?lat={lat}&lon={lon}&type=hour&start={start}&cnt={cnt}&appid={API key}
```



Sử dụng ngôn ngữ lập trình Python và một số thư viện sau để hỗ trợ cho đồ án này.

```
import requests, json
import pandas as pd
import numpy as np
import time
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

Dùng thư viện Time để lấy thời gian thực, khoảng thời gian tra cứu lên đến 7 ngày. Mốc thời gian tùy người tra cứu thay đổi ở dạng Unix time. Ở đây em sử dụng để lấy khoảng thời gian từ 1 tuần trước đến thực tại.

ID thành phố có thể lấy từ file *city_list.json* hoặc tra cứu từ bảng Geoname id

```
now = int(time.time())      # Lấy realtime
lw = now - 604800           # Thời gian tra cứu (tối đa 7 ngày) theo định dạng Unix timestamps

cityid = 6167865            # Có thể dựa theo Geoname id hoặc tra cứu trong file hoặc bảng city bên trên

# Một số id mẫu
# New York 5128581, London 5056033, Paris 6455259,
# HCM 1566083, Hanoi 1581129, Sa Pa 1568043, Da Lat 1584071, ...
```

Dùng thư viện requests và json để nhận dữ liệu từ api với dạng file json. Như hình ta có thể thấy dữ liệu ta cần sử dụng nằm trong ‘List’

```
data = requests.get("https://history.openweathermap.org/data/2.5/history/city?id="+str(cityid)+"&type=hour&start="+str(lw)+"8")
data = json.loads(data.content)
data
```



Lưu file Json lại có tên là *data.json*.

```
data = data['list']
with open("data.json", "w") as outfile:
    json.dump(data, outfile)
```

3. Xử lý và phân tích dữ liệu

3.1. Xử lý dữ liệu

Đọc lại file json vừa thu được ở trên bằng pandas và biến nó thành 1 dataframe.

```
df = pd.read_json("data.json")
df = df.explode('weather') # Vì cột 'weather' là kiểu dữ liệu dict nhưng bị lồng bởi list nên ta cần bung ra
```

	dt	main	wind	clouds	weather	rain
0	1669824000	{'temp': 6.65, 'feels_like': 1.46, 'pressure': ...}	{'speed': 11.32, 'deg': 300, 'gust': 18.01}	{'all': 100}	{'id': 771, 'main': 'Squall', 'description': '...'}	NaN
1	1669827600	{'temp': 4.51, 'feels_like': -1.52, 'pressure': ...}	{'speed': 11.83, 'deg': 290, 'gust': 14.92}	{'all': 100}	{'id': 804, 'main': 'Clouds', 'description': '...'}	NaN
2	1669831200	{'temp': 4.04, 'feels_like': -2.15, 'pressure': ...}	{'speed': 11.83, 'deg': 280, 'gust': 15.43}	{'all': 75}	{'id': 803, 'main': 'Clouds', 'description': '...'}	NaN
3	1669834800	{'temp': 2.58, 'feels_like': -4.42, 'pressure': ...}	{'speed': 14.4, 'deg': 260, 'gust': 20.06}	{'all': 100}	{'id': 804, 'main': 'Clouds', 'description': '...'}	NaN
4	1669838400	{'temp': 1.67, 'feels_like': -3.45, 'pressure': ...}	{'speed': 6.26, 'deg': 271, 'gust': 8.05}	{'all': 95}	{'id': 804, 'main': 'Clouds', 'description': '...'}	NaN
...
163	1670410800	{'temp': 5, 'feels_like': 1.52, 'pressure': 10...}	{'speed': 4.63, 'deg': 320}	{'all': 100}	{'id': 701, 'main': 'Mist', 'description': 'mi...'}	NaN
164	1670414400	{'temp': 5.03, 'feels_like': 1.83, 'pressure': ...}	{'speed': 4.12, 'deg': 320}	{'all': 100}	{'id': 701, 'main': 'Mist', 'description': 'mi...'}	NaN
165	1670418000	{'temp': 5.03, 'feels_like': 1.56, 'pressure': ...}	{'speed': 4.63, 'deg': 340}	{'all': 100}	{'id': 701, 'main': 'Mist', 'description': 'mi...'}	NaN
166	1670421600	{'temp': 5.2, 'feels_like': 3.54, 'pressure': ...}	{'speed': 2.06, 'deg': 240}	{'all': 100}	{'id': 701, 'main': 'Mist', 'description': 'mi...'}	NaN
167	1670425200	{'temp': 5.82, 'feels_like': 3.81, 'pressure': ...}	{'speed': 2.5700000000000003, 'deg': 20}	{'all': 100}	{'id': 701, 'main': 'Mist', 'description': 'mi...'}	NaN

Vì dữ liệu bị lồng vô những nhóm chính là main, wind, clouds, weather và rain nên ta cần chuẩn hóa lại dữ liệu

```
main = pd.json_normalize(df['main'], max_level = 1)
main
```

	temp	feels_like	pressure	humidity	temp_min	temp_max
0	6.65	1.46	997	93	5.64	7.40
1	4.51	-1.52	997	91	2.97	6.83
2	4.04	-2.15	999	86	2.88	4.83
3	2.58	-4.42	1001	78	1.31	3.83
4	1.67	-3.45	1003	73	0.80	2.49
...
163	5.00	1.52	1019	93	4.17	5.40
164	5.03	1.83	1019	94	4.15	5.40
165	5.03	1.56	1019	94	4.17	5.40
166	5.20	3.54	1020	93	4.71	5.76
167	5.82	3.81	1020	91	4.73	6.40

Tương tự với những cột còn lại

```
wind = pd.json_normalize(df['wind'], max_level = 1)
clouds = pd.json_normalize(df['clouds'], max_level = 1)
weather = pd.json_normalize(df['weather'], max_level = 1)
```

Xóa những cột cũ sau khi đã chuẩn hóa xong và lưu vào 1 biến mới

Và vì thời gian API cung cấp ở dạng Unix time nên ta chuẩn hóa về dạng bình thường và ở múi giờ +7

```

df.drop(columns = ['main','wind','clouds','weather'], inplace = True)

df['dt'] = pd.to_datetime(df['dt'], unit='s')
df = df.set_index('dt')
df = df.tz_localize('UTC')
df = df.tz_convert('Asia/Ho_Chi_Minh')
df.reset_index(inplace = True)
df['dt']=df['dt'].dt.tz_localize(None)
df

```

	dt	rain
0	2022-11-30 23:00:00	NaN
1	2022-12-01 00:00:00	NaN
2	2022-12-01 01:00:00	NaN
3	2022-12-01 02:00:00	NaN
4	2022-12-01 03:00:00	NaN
...
163	2022-12-07 18:00:00	NaN
164	2022-12-07 19:00:00	NaN
165	2022-12-07 20:00:00	NaN
166	2022-12-07 21:00:00	NaN
167	2022-12-07 22:00:00	NaN

Sau khi chuẩn hóa hết cả các cột ta gộp chúng lại thành 1 dataframe hoàn chỉnh

```

df = pd.concat([df, main, wind, clouds, weather], axis =1 )
df

```

	dt	rain	temp	feels_like	pressure	humidity	temp_min	temp_max	speed	deg	gust	all	id	main	description	icon
0	2022-11-30 23:00:00	NaN	6.65	1.46	997	93	5.64	7.40	11.32	300	18.01	100	771	Squall	squalls	50d
1	2022-12-01 00:00:00	NaN	4.51	-1.52	997	91	2.97	6.83	11.83	290	14.92	100	804	Clouds	overcast clouds	04d
2	2022-12-01 01:00:00	NaN	4.04	-2.15	999	86	2.88	4.83	11.83	280	15.43	75	803	Clouds	broken clouds	04d
3	2022-12-01 02:00:00	NaN	2.58	-4.42	1001	78	1.31	3.83	14.40	260	20.06	100	804	Clouds	overcast clouds	04d
4	2022-12-01 03:00:00	NaN	1.67	-3.45	1003	73	0.80	2.49	6.26	271	8.05	95	804	Clouds	overcast clouds	04d
...
163	2022-12-07 18:00:00	NaN	5.00	1.52	1019	93	4.17	5.40	4.63	320	NaN	100	701	Mist	mist	50n
164	2022-12-07 19:00:00	NaN	5.03	1.83	1019	94	4.15	5.40	4.12	320	NaN	100	701	Mist	mist	50n
165	2022-12-07 20:00:00	NaN	5.03	1.56	1019	94	4.17	5.40	4.63	340	NaN	100	701	Mist	mist	50d
166	2022-12-07 21:00:00	NaN	5.20	3.54	1020	93	4.71	5.76	2.06	240	NaN	100	701	Mist	mist	50d
167	2022-12-07 22:00:00	NaN	5.82	3.81	1020	91	4.73	6.40	2.57	20	NaN	100	701	Mist	mist	50d

168 rows × 16 columns

Kiểm tra những cột có giá trị null

```
df.isnull().any()
```

dt	False
rain	True
temp	False
feels_like	False
pressure	False
humidity	False
temp_min	False
temp_max	False
speed	False
deg	False
gust	True
all	False
id	False
main	False
description	False
icon	False
dtype: bool	

Đặt cột thời gian là index và bỏ đi những cột không cần thiết cho project này

```
df = df.set_index('dt')
if 'rain' in df.columns:
    df.drop(columns = 'rain', inplace = True)
if 'snow' in df.columns:
    df.drop(columns = 'snow', inplace = True)
df.drop(columns = ['temp_min', 'temp_max', 'gust', 'id', 'icon'], inplace = True)
df
```

	temp	feels_like	pressure	humidity	speed	deg	all	main	description
dt									
2022-11-30 23:00:00	6.65	1.46	997	93	11.32	300	100	Squall	squalls
2022-12-01 00:00:00	4.51	-1.52	997	91	11.83	290	100	Clouds	overcast clouds
2022-12-01 01:00:00	4.04	-2.15	999	86	11.83	280	75	Clouds	broken clouds
2022-12-01 02:00:00	2.58	-4.42	1001	78	14.40	260	100	Clouds	overcast clouds
2022-12-01 03:00:00	1.67	-3.45	1003	73	6.26	271	95	Clouds	overcast clouds
...
2022-12-07 18:00:00	5.00	1.52	1019	93	4.63	320	100	Mist	mist
2022-12-07 19:00:00	5.03	1.83	1019	94	4.12	320	100	Mist	mist
2022-12-07 20:00:00	5.03	1.56	1019	94	4.63	340	100	Mist	mist
2022-12-07 21:00:00	5.20	3.54	1020	93	2.06	240	100	Mist	mist
2022-12-07 22:00:00	5.82	3.81	1020	91	2.57	20	100	Mist	mist

Đổi lại tên các cột dữ liệu cho dễ nhận biết để ra dataframe cuối cùng

```
df.rename(columns={"temp": "Temperature (°C)",
                  "feels_like": "Feels Like (°C)",
                  "pressure": "Atmospheric pressure (hPa)",
                  "humidity": "Humidity (%)",
                  "speed": "Wind speed (meter/sec)",
                  "deg": "Wind direction (°)",
                  "all": "Cloudiness (%)",
                  "main": "Weather parameters",
                  "description": "Weather descriptions"},
          inplace = True)
df.index.name = "Time"
df
```

Time	Temperature (°C)	Feels Like (°C)	Atmospheric pressure (hPa)	Humidity (%)	Wind speed (meter/sec)	Wind direction (°)	Cloudiness (%)	Weather parameters	Weather descriptions
2022-11-30 23:00:00	6.65	1.46	997	93	11.32	300	100	Squall	squalls
2022-12-01 00:00:00	4.51	-1.52	997	91	11.83	290	100	Clouds	overcast clouds
2022-12-01 01:00:00	4.04	-2.15	999	86	11.83	280	75	Clouds	broken clouds
2022-12-01 02:00:00	2.58	-4.42	1001	78	14.40	260	100	Clouds	overcast clouds
2022-12-01 03:00:00	1.67	-3.45	1003	73	6.26	271	95	Clouds	overcast clouds
...
2022-12-07 18:00:00	5.00	1.52	1019	93	4.63	320	100	Mist	mist
2022-12-07 19:00:00	5.03	1.83	1019	94	4.12	320	100	Mist	mist
2022-12-07 20:00:00	5.03	1.56	1019	94	4.63	340	100	Mist	mist
2022-12-07 21:00:00	5.20	3.54	1020	93	2.06	240	100	Mist	mist
2022-12-07 22:00:00	5.82	3.81	1020	91	2.57	20	100	Mist	mist

Tổng quan về dataframe vừa thu được

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 168 entries, 2022-11-30 23:00:00 to 2022-12-07 22:00:00
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Temperature (°C)    168 non-null   float64
 1   Feels Like (°C)     168 non-null   float64
 2   Atmospheric pressure (hPa) 168 non-null   int64  
 3   Humidity (%)        168 non-null   int64  
 4   Wind speed (meter/sec) 168 non-null   float64
 5   Wind direction (°)    168 non-null   int64  
 6   Cloudiness (%)       168 non-null   int64  
 7   Weather parameters   168 non-null   object  
 8   Weather descriptions 168 non-null   object  
dtypes: float64(3), int64(4), object(2)
memory usage: 13.1+ KB
```

Kích thước của df

```
df.shape
```

```
(168, 9)
```

3.2. Phân tích dữ liệu

Kiểm tra và nhận xét tổng quan về giá trị từng cột

```
df.describe(include=[np.number])
```

	Temperature (°C)	Feels Like (°C)	Atmospheric pressure (hPa)	Humidity (%)	Wind speed (meter/sec)	Wind direction (°)	Cloudiness (%)
count	168.000000	168.000000	168.000000	168.000000	168.000000	168.000000	168.000000
mean	2.479940	-0.939107	1016.857143	72.773810	4.459345	215.553571	80.250000
std	3.066114	4.113930	6.619049	11.025122	3.445857	72.764303	29.657072
min	-2.760000	-8.830000	997.000000	51.000000	0.450000	20.000000	4.000000
25%	-0.060000	-3.917500	1014.000000	65.000000	2.240000	160.000000	65.000000
50%	2.235000	-1.575000	1017.500000	71.000000	3.130000	240.000000	100.000000
75%	5.007500	2.882500	1021.000000	81.000000	5.360000	270.000000	100.000000
max	8.840000	7.200000	1029.000000	95.000000	17.490000	340.000000	100.000000

```
df.describe(include=[object])
```

	Weather parameters	Weather descriptions
count	168	168
unique	6	9
top	Clouds	overcast clouds
freq	150	98

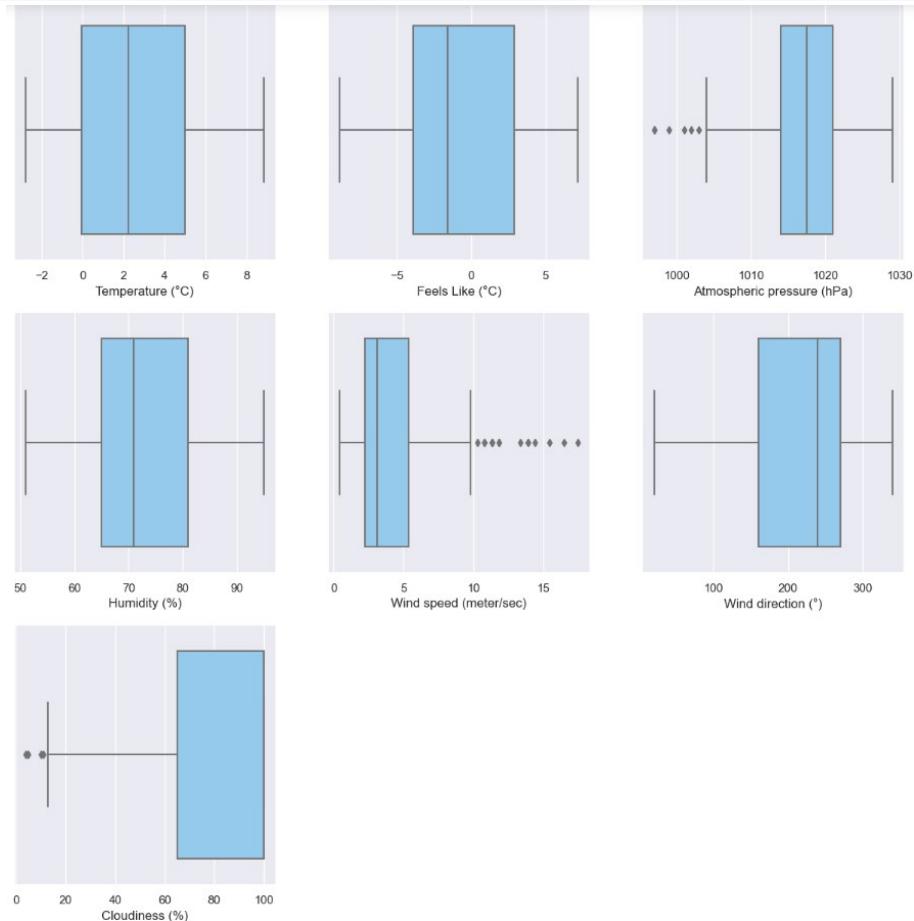
Kiểm tra dữ liệu ngoại lệ

```

Cols_num_list = Cols_category_list = df.select_dtypes(include = [np.number]).columns.to_list()
fig = plt.subplots(figsize=(15,15))
sp = 1
for i in Cols_num_list:
    sns.set(style="darkgrid")
    plt.subplot(3,3,sp)
    sns.boxplot(data = df,color = 'lightskyblue', x = i)
    sp = sp + 1

plt.show()

```



Lập bảng tương quan giữa các giá trị với nhau

```

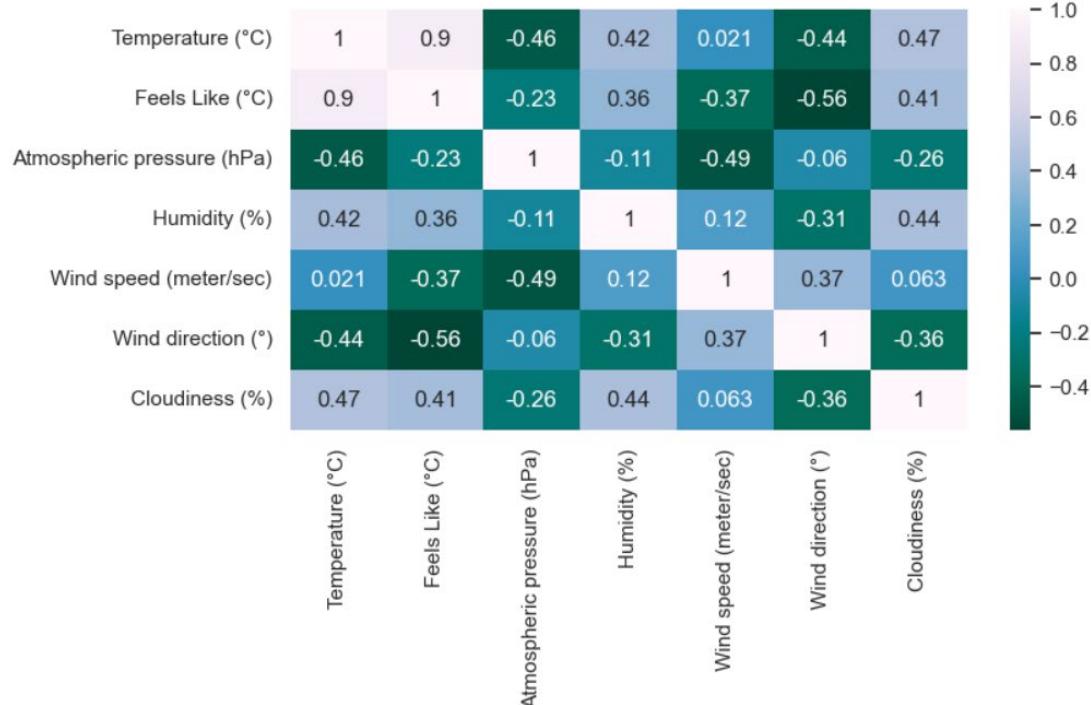
Correlation = df.corr()
Correlation

```

	Temperature (°C)	Feels Like (°C)	Atmospheric pressure (hPa)	Humidity (%)	Wind speed (meter/sec)	Wind direction (°)	Cloudiness (%)
Temperature (°C)	1.000000	0.903548	-0.458217	0.415069	0.021283	-0.442764	0.466027
Feels Like (°C)	0.903548	1.000000	-0.231478	0.356424	-0.373864	-0.564347	0.411667
Atmospheric pressure (hPa)	-0.458217	-0.231478	1.000000	-0.107609	-0.490114	-0.059512	-0.261635
Humidity (%)	0.415069	0.356424	-0.107609	1.000000	0.117110	-0.307501	0.436549
Wind speed (meter/sec)	0.021283	-0.373864	-0.490114	0.117110	1.000000	0.369925	0.063337
Wind direction (°)	-0.442764	-0.564347	-0.059512	-0.307501	0.369925	1.000000	-0.364986
Cloudiness (%)	0.466027	0.411667	-0.261635	0.436549	0.063337	-0.364986	1.000000

Tạo ma trận thẻ hiện sự tương quan

```
plt.figure(figsize = (8,4))
sns.heatmap(Correlation,mask = Correlation < -1,cmap="PuBuGn_r",annot=True, linewidths=0)
plt.show()
```



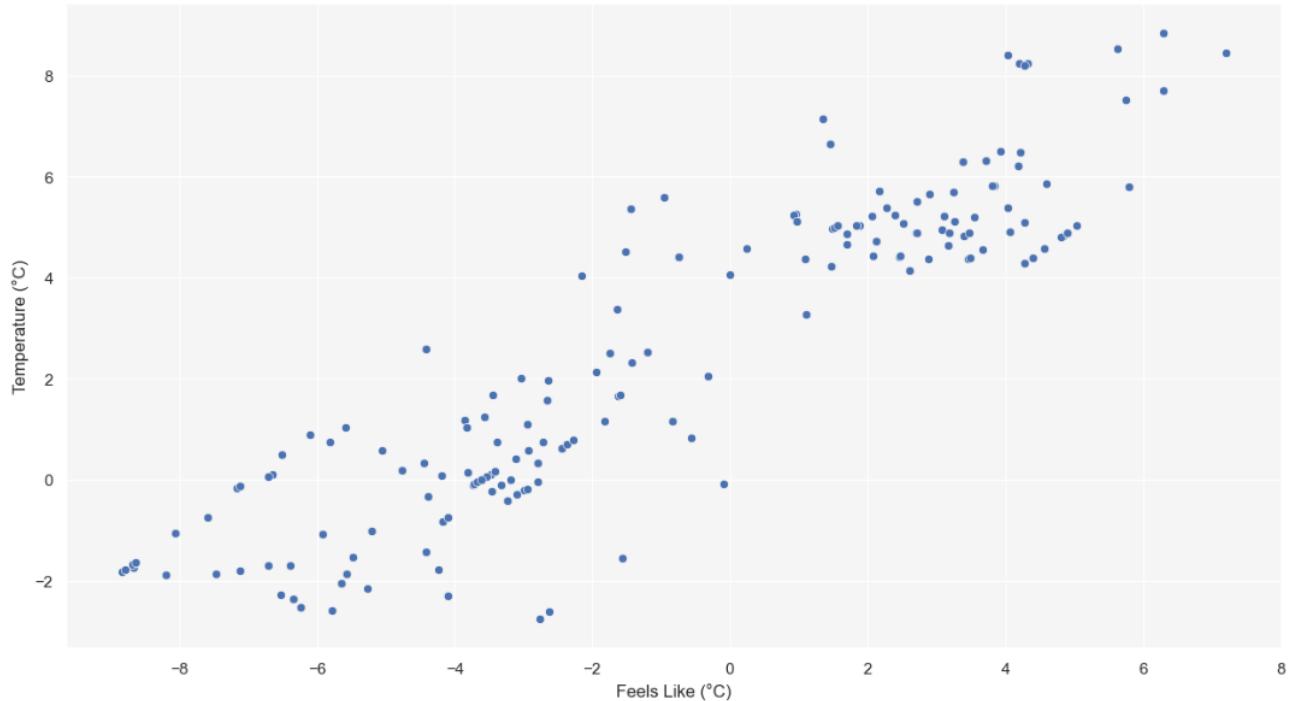
Sử dụng biểu đồ để thể hiện sự tương quan giữa vài cột dữ liệu chính

```

fig, axs = plt.subplots(figsize=(15,8))
axs.set_facecolor("whitesmoke")
area = Correlation['Feels Like (°C)']['Temperature (°C)']
sns.scatterplot( data=area, y=df['Temperature (°C)'], x=df['Feels Like (°C)'], legend="full")

plt.show()

```

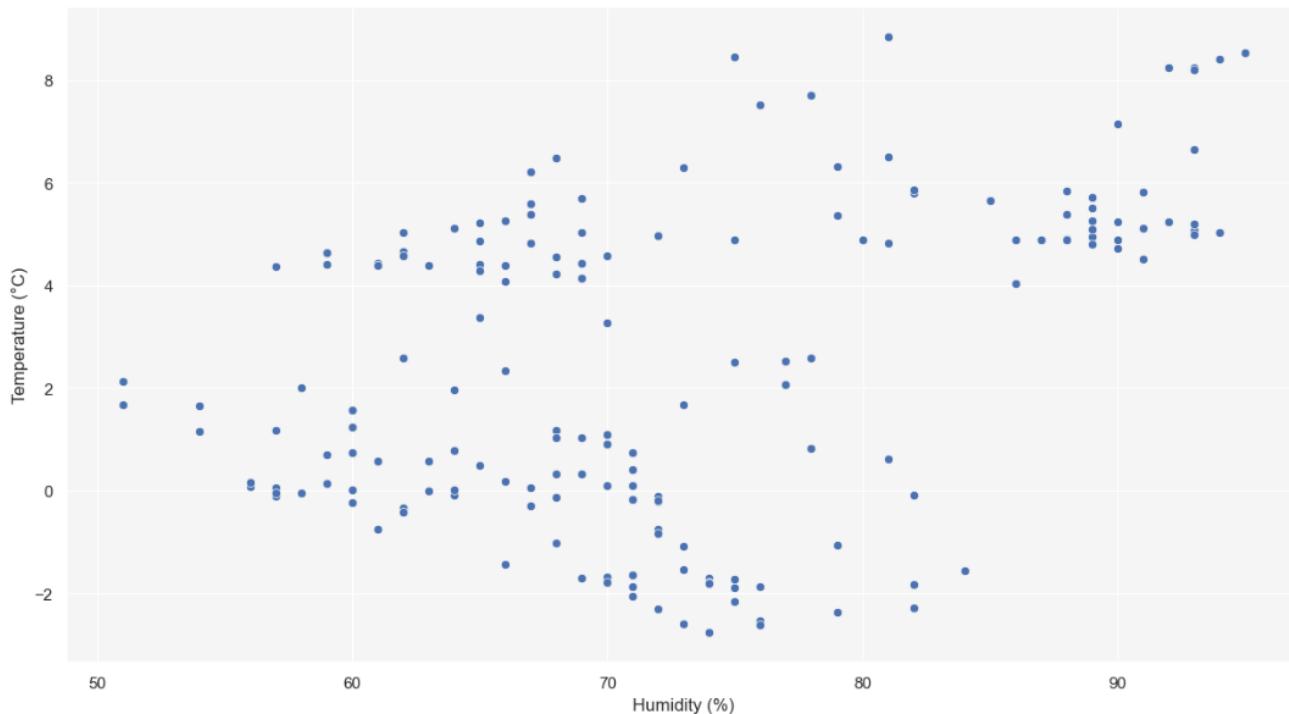


```

fig, axs = plt.subplots(figsize=(15,8))
axs.set_facecolor("whitesmoke")
area = Correlation['Feels Like (°C)']['Humidity (%)']
sns.scatterplot( data=area, y=df['Temperature (°C)'], x=df['Humidity (%)'], legend="full")

plt.show()

```



4. Kết quả

Biểu đồ thể hiện số liệu của nhiệt độ và cảm giác, độ ẩm

```
fig, ax2 = plt.subplots(figsize=(20,10))
plt.title('Temperature, Feeling and Humidity in lastweek', size = 30)
sns.set(style="white")
ax2.set_facecolor("whitesmoke")

ax2.bar(df.index, df['Humidity (%)'], width=0.035, alpha=1,color = 'lightskyblue',label='Humidity (%)')
ax2.grid(False) # turn off grid #
ax2.set_ylabel("%",size = 20)
ax2.set_xlabel('Days', size = 20)
ax2.legend(['Humidity'], loc="upper left", fontsize = 'large')

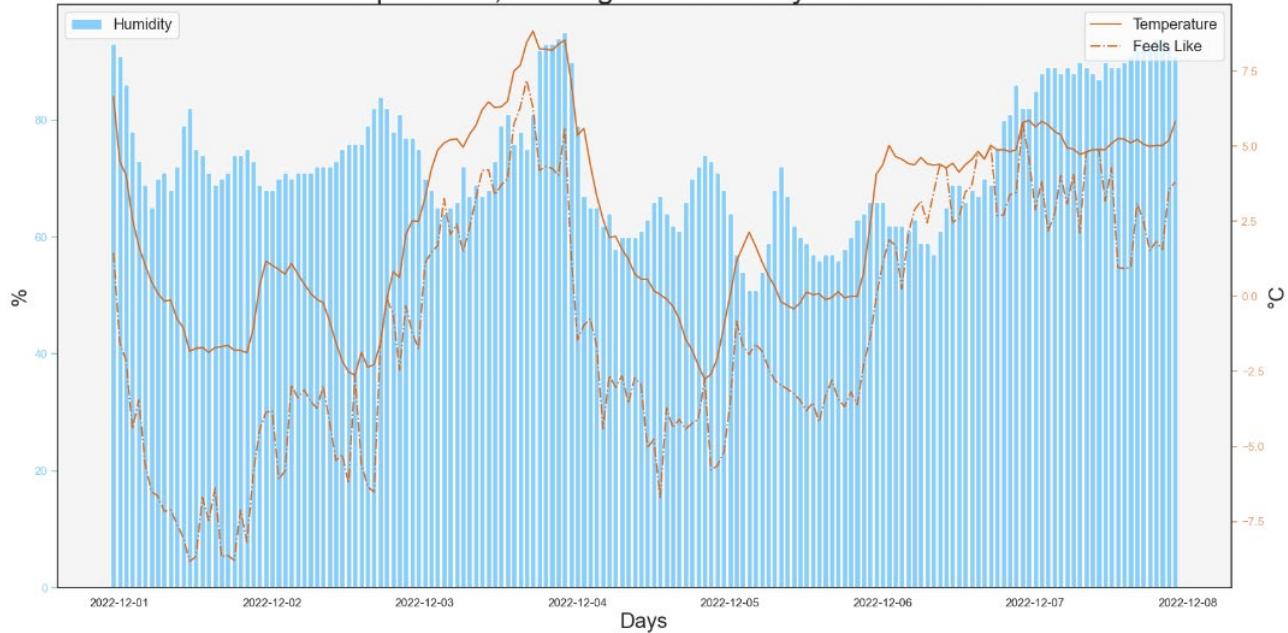
ax1 = ax2.twinx()

ax1.plot(df.index, df['Temperature (°C)'], alpha=1,color = 'chocolate',label='Temperature (°C)')
ax1.plot(df.index, df['Feels Like (°C)'],alpha=1,color = 'chocolate',linestyle = '-.',label='Feels Like (°C)')
ax1.set_ylabel("°C", size = 20)
ax1.set_xlabel('Days', size = 20)
ax1.legend(['Temperature','Feels Like'], loc="upper right", fontsize = 'large')

#specify axis colors
ax1.tick_params(axis='y', colors="#e19d72")
ax2.tick_params(axis='y', colors="#87cefa")

plt.show()
```

Temperature, Feeling and Humidity in lastweek



Đưa ra các khoảng thời gian chứa giá trị cao nhất và thấp nhất của nhiệt độ và độ ẩm

```
df['Temperature (°C)'].sort_values().head() df['Humidity (%)'].sort_values().head()
```

Time	Time
2022-12-04 20:00:00	-2.76
2022-12-02 13:00:00	-2.62
2022-12-04 21:00:00	-2.59
2022-12-02 12:00:00	-2.52
2022-12-02 15:00:00	-2.36

Name: Temperature (°C), dtype: float64

Time	Time
2022-12-05 03:00:00	51
2022-12-05 04:00:00	51
2022-12-05 02:00:00	54
2022-12-05 05:00:00	54
2022-12-05 14:00:00	56

Name: Humidity (%), dtype: int64

```
df['Temperature (°C)'].sort_values().tail() df['Humidity (%)'].sort_values().tail()
```

Time	Time
2022-12-03 18:00:00	8.25
2022-12-03 21:00:00	8.40
2022-12-03 16:00:00	8.46
2022-12-03 22:00:00	8.54
2022-12-03 17:00:00	8.84

Name: Temperature (°C), dtype: float64

Time	Time
2022-11-30 23:00:00	93
2022-12-03 21:00:00	94
2022-12-07 19:00:00	94
2022-12-07 20:00:00	94
2022-12-03 22:00:00	95

Name: Humidity (%), dtype: int64

Biểu đồ thể hiện áp suất không khí

```
fig = px.line(df, x=df.index, y=df['Atmospheric pressure (hPa)'], title="Atmospheric pressure in lastweek")
fig.show()
```

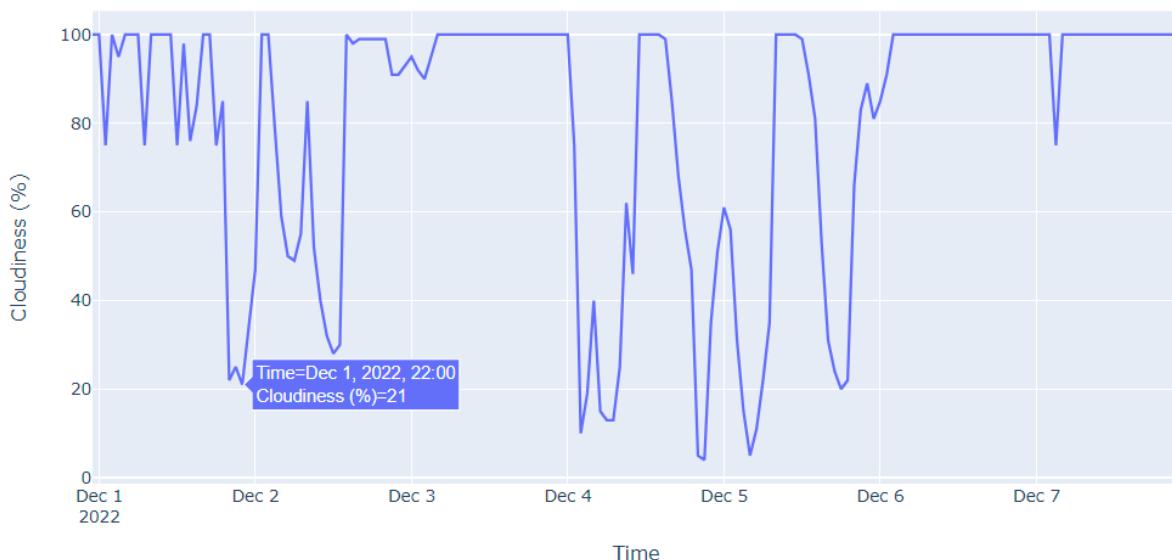


Biểu đồ thể hiện lượng bao phủ của mây

```
fig = px.line(df, x=df.index, y=df['Cloudiness (%)'], title="Cloudiness in lastweek")
fig.show()
```



Cloudiness in lastweek



Phân tích các loại thời tiết và biểu diễn chúng theo biểu đồ tròn

```
df['Weather parameters'].value_counts()
```

```
Clouds      150
Mist        10
Clear        4
Rain         2
Squall       1
Drizzle      1
Name: Weather parameters, dtype: int64
```

```
df['Weather descriptions'].value_counts()
```

```
overcast clouds      98
broken clouds        23
scattered clouds     17
few clouds          12
mist                 10
clear sky            4
light rain           2
squalls              1
light intensity drizzle  1
Name: Weather descriptions, dtype: int64
```

```

weather_descriptions = pd.DataFrame(df['Weather descriptions'].value_counts())
weather_descriptions.rename(columns={"Weather descriptions": "Counts"}, inplace = True)
weather_descriptions.index.name='Weather descriptions'
weather_descriptions['Precent'] = (weather_descriptions['Counts']/weather_descriptions['Counts'].sum())*100
weather_descriptions

```

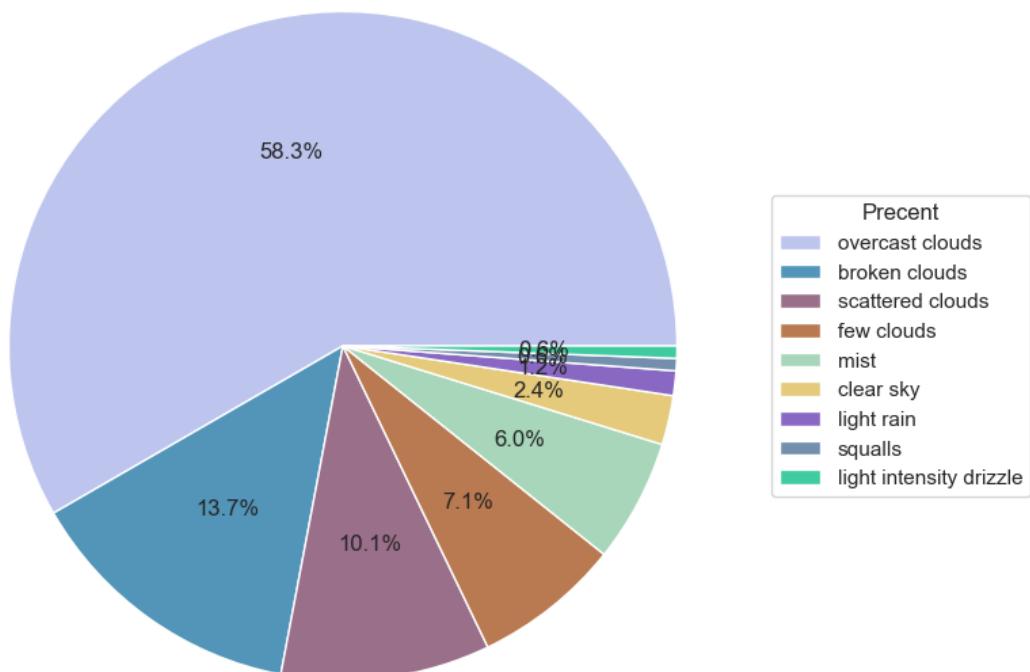
	Counts	Precent
Weather descriptions		
overcast clouds	98	58.333333
broken clouds	23	13.690476
scattered clouds	17	10.119048
few clouds	12	7.142857
mist	10	5.952381
clear sky	4	2.380952
light rain	2	1.190476
squalls	1	0.595238
light intensity drizzle	1	0.595238

```

fig,ax = plt.subplots(1, figsize=(8,8))
plt.title('Weather descriptions in lastweek', size = 15)
fig.subplots_adjust(wspace=0)
colors = ['#bec5ef', '#5296ba', '#9a6f8b', '#ba7a51',
          '#a8d6bb', '#e5ca7b', '#8868c3', '#738fad',
          '#40cba1', '#6e8484', '#b09d73', '#af71b0']
wedges, *_ = ax.pie(weather_descriptions['Precent'], wedgeprops = { 'linewidth' : 1, 'edgecolor' : 'white' }, colors=colors,
ax.legend(weather_descriptions.index, title="Precent", loc="center left", bbox_to_anchor=(1, 0, 0.5, 1))
plt.show()

```

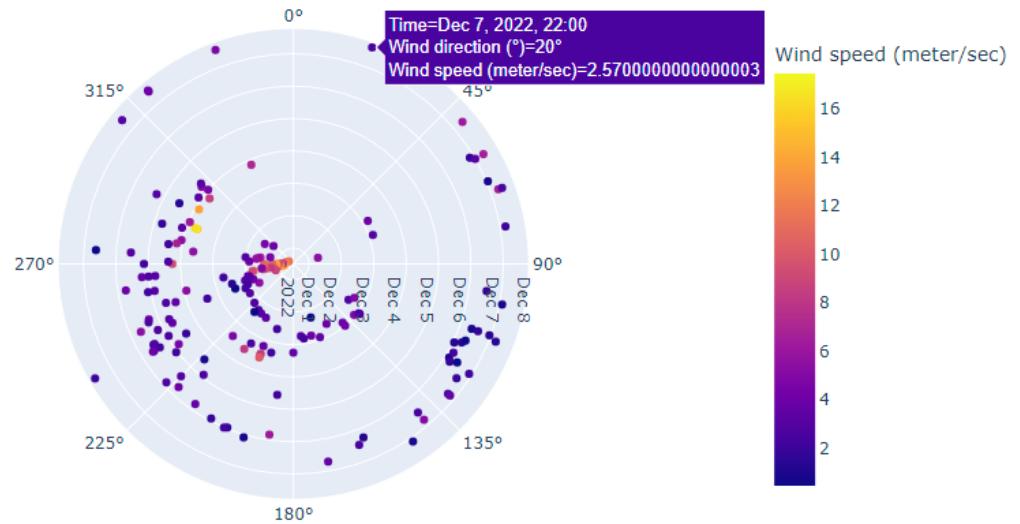
Weather descriptions in lastweek



Biểu đồ hướng và tốc độ gió

```
figsize=(20,20)
fig = px.scatter_polar(df, theta = df['Wind direction (°)'],r = df.index,
                      title='Wind Scatter',color="Wind speed (meter/sec)")
fig.show()
```

Wind Scatter



```
fig = px.bar(Wind_direction, y = 'Counts', x = Wind_direction.index)
fig.show()
```

