
Table of Contents

Overview

Introduction	1.1
Course Outline	1.2

Weekly Notes

Notes: Week 00 - GitHub and markdown	2.1
Notes: Week 01 - Terminal, Python, Jupyter Notebook	2.2
Notes: Week 02 - Python as a powerful calculator: basics and arithmetics	2.3
Notes: Week 03 - Control flow	2.4
Notes: Week 04 - Data Structure	2.5
Notes: Week 05 - Serialization: File, CSV, JSON	2.6
Notes: Week 06 - API	2.7
Notes: Week 07 - Get semi-structured data - Web scraping	2.8
Notes: Week 08 - Advanced scraping: anti-crawler, browser emulation and other nitty gritty	2.9
Notes: Week 09 - Work with table: data cleaning and pre-processing	2.10
Notes: Week 10 - Work with table: 1D analysis and 2D analysis	2.11
Notes: Week 11 - Present findings: data visualization and reproducible report	2.12
Notes: Week 12 - Text data	2.13
Notes: Week 13 - Datetime and Time Series	2.14
Notes: Week 14 - Network data	2.15
Notes: Week 15 - Geographical data	2.16
Notes: Week 16 - High Dimensional Data	2.17
Notes: Week 17 - Machine learning primer: clustering, classification, regression	2.18
Notes: Week 18 - Python Engineering and Data Engineering	2.19

General FAQs

Course Admin	3.1
Grading Scheme	3.2
Guide for Contributor	3.3
Projects Guideline	3.4
Setup Python Environment on Windows and MAC	3.5
Shell	3.6
Python Language Basics	3.7
Python 2 v.s. Python 3	3.8

Dataprep	3.9
Pro Tips	3.10
Resources	3.11
Guide for contributor	3.12
GitHub	3.13
HTML	3.14
Encoding	3.15
pip	3.16
Computational Thinking	3.17
File I/O	3.18
Leetcode	3.19

Python Modules FAQs

module: geopy	4.1
module: requests	4.2
module: csv	4.3
module: BeautifulSoup	4.4
module: jupyter	4.5
module: pandas	4.6
module: seaborn	4.7
module: matplotlib	4.8
module: lxml	4.9
module: python-twitter	4.10
module: datetime	4.11
module: selenium	4.12
module: wordcloud	4.13
module: geopandas	4.14

Other Uncategorized FAQs

Other Frequently Asked Questions	5.1
----------------------------------	-----

Python and Data Analysis for Communication Students

- [Python and Data Analysis for Communication Students](#)
 - [Overview](#)
 - [Weekly Notes](#)
 - [FAQ Catalogue](#)
 - [General FAQs](#)
 - [Python Modules FAQs](#)
 - [Other Uncategorized FAQs](#)
 - [License](#)

This is an open source text book for communication students to learn Python and data skills. The purpose of this course is to motivate the students to become a [T-shape talent](#) in communications field. The course involves intensive training of Python and quest in solving practical problems. This open source book collects all the materials related with lab exercises covering basic Python, data scraping, table manipulation and data mining. You can get started by reading [Week 00](#) note. This note requires no background in programming or technology. It walks you through our learning environment so you know how to find learning materials and seek for help effectively.

Overview

- [Introduction](#)
- [Course Outline](#)
- [Spring 2018 offering of the course](#)
- [Fall 2018 offering of the course](#)

Weekly Notes

- [Notes: Week 00 - GitHub and markdown](#)
- [Notes: Week 01 - Terminal, Python, Jupyter Notebook](#)
- [Notes: Week 02 - Python as a powerful calculator: basics and arithmetics](#)
- [Notes: Week 03 - Control flow](#)
- [Notes: Week 04 - Data Structure](#)
- [Notes: Week 05 - Serialization: File, CSV, JSON](#)
- [Notes: Week 06 - API](#)
- [Notes: Week 07 - Get semi-structured data - Web scraping](#)
- [Notes: Week 08 - Advanced scraping: anti-crawler, browser emulation and other nitty gritty](#)
- [Notes: Week 09 - Work with table: data cleaning and pre-processing](#)
- [Notes: Week 10 - Work with table: 1D analysis and 2D analysis](#)
- [Notes: Week 11 - Present findings: data visualization and reproducible report](#)
- [Notes: Week 12 - Text data](#)
- [Notes: Week 13 - Datetime and Time Series](#)
- [Notes: Week 14 - Network data](#)
- [Notes: Week 15 - Geographical data](#)
- [Notes: Week 16 - High Dimensional Data](#)
- [Notes: Week 17 - Machine learning primer: clustering, classification, regression](#)
- [Notes: Week 18 - Python Engineering and Data Engineering](#)

FAQ Catalogue

You can search through our whole repo, including all the notes and FAQs using the built-in GitHub search function. For example, you can [search for "encoding"](#).

General FAQs

- [Course Admin](#)
- [Grading Scheme](#)
- [Guide for Contributor](#)
- [Projects Guideline](#)
- [Setup Python Environment on Windows and MAC](#)
- [Shell](#)
- [Python Language Basics](#)
- [Python 2 v.s. Python 3](#)
- [Dataprep](#)
- [Pro Tips](#)
- [Resources](#)
- [Guide for contributor](#)
- [GitHub](#)
- [HTML](#)
- [Encoding](#)
- [pip](#)
- [Computational Thinking](#)
- [File I/O](#)
- [Leetcode](#)

Python Modules FAQs

- [module: geopy](#)
- [module: requests](#)
- [module: csv](#)
- [module: BeautifulSoup](#)
- [module: jupyter](#)
- [module: pandas](#)
- [module: seaborn](#)
- [module: matplotlib](#)
- [module: lxml](#)
- [module: python-twitter](#)
- [module: datetime](#)
- [module: selenium](#)
- [module: wordcloud](#)
- [module: geopandas](#)

Other Uncategorized FAQs

- [Other Frequently Asked Questions](#)
-

License

CC-BY-NC-ND

[List of contributors](#)

Course Outline

- [Course Outline](#course-outline) - [Week 0 - GitHub](#week-0---github) - [Week 1 - Hands-on the Terminal](#week-1---hands-on-the-terminal) - [Week 2 - Use Python as a daily tool](#week-2---use-python-as-a-daily-tool) - [Week 3 - Python for anything](#week-3---python-foranything) - [Week 4 - JSON and API](#week-4---json-and-api) - [Week 5 - Web Scraping Basics](#week-5---web-scraping-basics) - [Week 6 - Advanced Web Scraping](#week-6---advanced-web-scraping) - [Week 7 - Table manipulation and 1-D analysis](#week-7---table-manipulation-and-1-d-analysis) - [Week 8 - Visualisation, presentation and reproducible reports](#week-8---visualisation-presentation-and-reproducible-reports) - [Week 9 - Text analysis](#week-9---text-analysis) - [Week 10 - Time series](#week-10---time-series) - [Week 11 - Graph theory and social network analysis](#week-11---graph-theory-and-social-network-analysis) - [Week 12 - 2D analysis](#week-12---2d-analysis) - [Week 13 - High-dimensional analysis](#week-13---high-dimensional-analysis) - [Week 14 - Clustering](#week-14---clustering) - [Week 15 - Classification](#week-15---classification) - [Week 16 - Regression](#week-16---regression) - [Week 17 - Recommender System](#week-17---recommender-system) - [Open topics](#open-topics)

Week 0 - GitHub

Objective

- Can use GitHub for resource hosting, project management and discussion forum.
- Can use GitHub Desktop to sync local repos with remote repos.
- Can use `gh-pages` to host static web pages as one's portfolio.

Week 1 - Hands-on the Terminal

Objective:

- Able to navigate file system in Terminal, using shell
- Create the first python script and execute it

MAC:

- `cmd+space` to open Spotlight; search “Terminal” to open terminal

Shell commands:

- `cd` to switch working folder
 - Path separated by `/`
 - Special paths: `..`, `...`, `-`, `~`
- `ls` to list files/ folders in current folder
- `pwd` to check current working folder
- `ls / pwd` is your friend; type often to make sure where you are
- `touch` to create empty new file; `mkdir` to create new directory
- `python` to execute python scripts (usually in `.py` but not necessary)
- Format of shell commands:
 - `<command-name> <arg1> <arg2>.. (space separated arguments)`

Challenge:

1. Write a Python script to output "Good evening" in the Terminal.

References:

- Terminal and shell commands (Chinese)
- Appendix A of "Learn Python the hard way"

Week 2 - Use Python as a daily tool

Objective:

- Can use Python as a daily tool -- at least a powerful calculator
- Become comfortable with Python interpreter -- the REPL pattern (Read-Evaluate-Print Loop)
- Can use `help` to get inline documentation on new modules and functions

Python language introduction:

- Variables and assignment
- Basic data types: `int`, `float`, `str`, `bool`
- Arithmetic:
 - `+`, `-`, `*`, `/`, `//`, `%`, `**`
 - `math`, `numpy` (may need `pip`)
- Use functions and modules:
 - `import` (and `import ... from ...`)
 - `.` notation to reference to member variable/ method
 - `()` notation to call function
- Common modules and functions
 - `str.*` functions
 - String templating 1: `str.format`
 - String templating 2: `format_str % (var1, var2, ...)`
 - `random`
 - `numpy`, `scipy`

Challenge:

1. Build a mortgage calculator - given principal `P`, interest rate `r` and load period `n`, calculate the amortised monthly payment `A`
2. Calculate the `area` of a circle given its radius `r`
3. Given the length of hypotenuse of a right triangle, calculate the length of its legs. You may want to get values like $\sin(\frac{\pi}{6})$ via `numpy.pi` and `numpy.sin`
4. Generate 10 random numbers. (it is OK to run your script 10 times)

References:

- Chapter 1, 2, 3 of [official Python 2 tutorial](#)
- Python format string: <https://pyformat.info/>

Week 3 - Python for anything

Objective:

- Master the composite data type `[]` and `{}` in Python
- Master the control logics in Python, especially `if` and `for`
- Further understand the different roles of text editor and interpreter. Be comfortable writing batch codes in `.py` file and execute in Shell environment.
- [O] Understand Python engineering

Python language:

- `help`
- `bool` and comparisions
 - `str` comparison and `int` comparison
- Composite data types: `list` `[]`, `dict` `{}`
- Control flow:
 - `for`, `while`
 - `if`
 - `try..except`
- Function, class, module:
 - `def`
 - `class`
 - `*.py`; `from`, `import`

Workflow:

- Python interpreter
- pip: `pip3` for `python3`
 - `--user` option in shared computer

Challenge:

1. Distances among cities:
 - i. Calculate the "straight line" distance on earth surface from several source cities to Hong Kong. The source cities: New York, Vancouver, Stockholm, Buenos Aires, Perth. For each source city, print one line containing the name of the city and distance. "Great-circle distance" is the academic name you use to search for the formula.
 - ii. Use `list` and `for` loop to handle multiple cities
 - iii. Use function to increase the reusability
2. Divide HW1 groups randomly: (case contribution)
 - i. Get the list of student IDs from the lecturer
 - ii. Generate the grouping randomly
3. Solve the "media business model" calculator.

References:

- Chapter 4, 5, 6 of [official Python 3 tutorial](#)

Week 4 - JSON and API

Objective:

- Learn to use Jupyter notebook. All demos following this week will be conducted in Jupyter notebook.
- Understand API/ JSON and can retrieve data from online databases (twitter, GitHub, weibo, douban, ...)
- Understand basic file format like `json` and `csv`.
 - Be able to comfortably navigate through compound structures like `{}` and `[]`.
 - Be able to comfortably use (multiple layer of) for-loop to re-format data.
 - Be able to use serialisers to handle input/ output to files.

The brief of Application Programming Interface (API):

- Operate in client-and-server mode.
- Client does not have to download the full volume of data from server. Only use the data on demand.
- Server can handle intensive computations that not available by client.
- Server can send updated data upon request.

Modules:

- Handle HTTP request/ response: `requests`
- Serialiser: `json (.loads, .dumps)` and `csv`

Challenges:

- Taiwan had an earthquake in early Feb. Let's discuss this issue:
 - Search for the earthquake instances around Taiwan in recent 100 years and analyse the occurrences of earthquakes. You can refer to the same database used [here](#). Checkout the [API description](#). The `count` and `query` API are useful.
 - Search on Twitter and collect user's discussions about this topic. See if there is any findings. You can approach from the human interface [here](#) (hard mode) or use [python-twitter](#) module (need to register developer and obtain API key).
- Retrieve and analyse the recent movie. Douban's API will be helpful here.
 - [API sample for Recent movies](#)
 - [API sample for movie details](#)
- Use Google Map API to retrieve geo-locations and canonical names: e.g. [Get the location of HKBU](#)
- Lookup real estate properties on HK gov open data portal. e.g. the [dataset page](#), the [API result](#)
- blockchain.info provides a set of [API](#) for one to retrieve information related with bitcoin transactions. Pick one wallet address, check its UTXO sets and sum up the values to get the total balance in this wallet.
- [A free cryptocurrency API](#) for you to retrieve and study historical exchange rates.
- Implement a basic version of first automated writer - QuakeBot from LA Times
 - Get data from USGS API
 - Print a story to the screen using string templating/ string interpolation
 - See [here](#) for an introduction of the bot. See [here](#) for an incident and think how to avoid it?

Exercise:

- Request certain API to acquire information
- Convert a JSON to CSV in Python
- Convert a CSV to JSON in Python

Further readings:

- Use `beautifulsoup` to [scrape](#) Twitter timeline content from [Wayback machine](#). A good example of investigative journalism, by William Lyon from [neo4j](#).

Week 5 - Web Scraping Basics

Objective:

- Understand the basics of HTML language, HTTP protocol, web server and Internet architecture
- Able to scrape static web pages and turn them into CSV files

Tools: ([Step-by-step reference](#))

- Virtualenv -- Create isolated environment to avoid projects clutter each other
- Jupyter notebook -- Web-based REPL; ready to distribute; all-in-one presentation

Modules:

- Handle HTTP request/ response: `requests`
- Parse web page: `xml`, `Beautiful Soup`, `HTMLParser`, `help(str)`
 - Useful string functions: `strip()`, `split()`, `find()`, `replace()`, `str[begin:end]`
- Serialiser: `csv`, `json`

Challenges: (save to `*.csv`)

- Use `lxml / bs4 requests`
 - Collect a table for the [NSFC/RGC joint research fund](#). A full table can be found [here](#). You are also welcome to collect data of [other funding schemes](#).
 - Collect all the faculty's information and make a contact book. [site](#).
 - Collect the movie list and their rating from [IMDB](#).
- Bonus:
 - Collect the tweets from a celebrity like [this post](#). You can search "python twitter" for many useful modules online.

References:

- Allison Parrish's [tutorial of scraper](#) in summer 2017.

Further reading:

- Study `urllib` as an alternative to `requests`
- Study Regular Expression and `re` library in Python
- See how [reproducibility is improved](#) with Jupyter notebook and other tools (not only Python).

Week 6 - Advanced Web Scraping

Objective:

- Bypass anti-crawler by modifying user-agent
- Handle glitches: encoding, pagination, ...
- Handle dynamic page with headless browser
- Handle login with headless browser
- Scrape social networks
- Case studies on different websites
- Further strengthen the list-of-dict data types; organise multi-layer loops/ item based parsing logics.

Cases:

- <https://github.com/hupili/python-for-data-and-media-communication/tree/master/scraping-examples>
- <https://github.com/data-projects-archive>

Week 7 - Table manipulation and 1-D analysis

Objective:

- Master the schema of "data-driven story telling": the crowd (pattern) and the outlier (anomaly)
- Can efficiently manipulate structured table formatted datasets
- Use `pandas` for basic calculation and plotting

Modules:

- `pandas`
- `seaborn`
- `matplotlib`

Statistics:

- `mean, media, percentile`
- `min, max`

- variance
- histogram
- sort
- central tendency and spread of data
- Scatter plot and correlation

Datasets to work on:

- [openrice.csv](#) contributed by group 1

References:

- First two chapters (i.e. before "3D") of the article [The Art of Effective Visualization of Multi-dimensional Data](#) by Dipanjan Sarkar
- [Exercise numpy](#) on ShiYanLou
- [Exercise pandas](#) on ShiYanLou

Additional notes:

- You need to finish [Dataprep](#) before analysis. That is, we start with structured data. Preparing the structured and cleaned data has no common schema. We have pointers in [Dataprep](#) for your own reading.

Week 8 - Visualisation, presentation and reproducible reports

Objective

- Understand the theory and common tricks of visualisation.
- Can plot charts using various visualisation libraries.
- Can plot maps.
- Understand the concept of "reproducibility" and can use GitHub repo plus Jupyter notebook to create such reports.

Libraries:

- [py-plotly](#)
- [pyecharts](#)

Week 9 - Text analysis

Objective:

- Further strengthen the proficiency of pandas: DataFrame and Series
- Learn to plot and adjust charts with [matplotlib](#)
- Master basic string operations
- Understand some major text mining models and be able to apply algorithm from 3rd party libraries.

Modules & topics:

- `str` - basic string processing
 - `.split()`, `in`, `.find()`
 - `%s` format string
 - `'.format()'` function
- `collections.Counter` for word frequency calculation
- `jieba` - the most widely used Chinese word segmentation package.

- (optional) `re` - Regular Expression (regex) is the swiss knife for text pattern matching.
- (optional) `nltk` - contains common routines for text analysis
- (optional) `gensim` - topic mining package. It also contains the `Word2Vec` routine.
- (optional) Sentiment analysis - construct classifier using `sklearn` or use an API like [text-processing](#). `TextBlob` is also useful and applied in [group 2's work](#).

Related cases:

- [Quartz's analysis](#) of New York Times's column of "Modern Love"
- Prof. Qian Gang's famous [analysis of texts in political communication](#).

References:

- Construct Naive Bayes based classifier for sentiment analysis. [Read here](#)

Datasets to work on:

- [NBC Russian Troll on Twitter dataset](#) -- The 200,000 deleted Twitter messages posted by Russian's troll accounts. Around 50M, in CSV format.
- [Hillary Clinton email archive from WikiLeaks](#) There are the plain text and parsed data but you may need to run a scraper to get the data first.

Week 10 - Time series

- Understand the principle of timestamp and datetime format
- Master basic computation on datetime values
- Understand periodical analysis (daily, weekly, monthly, seasonal, etc)
- Can handle timezone conversion

Modules:

- `datetime`
- `dtparser`
- `pandas`
 - `basic visualisation` `.plot`
 - zoom in/ out: `.resample` , `.aggregate`
- `seaborn`

References:

- timestamp usually come in unit of milliseconds (1/1000) of a second. [An example](#) to parse this timestamp format into `datetime` format.

Datasets:

- [NBC Russian Troll on Twitter dataset](#) (used last week)
- [Twitter Data of the Donald & Ivanka Trump analysis](#) -- reproduce the charts.

Week 11 - Graph theory and social network analysis

Objective:

- Understand the basics of graph theory
- Understand most common applications in social network analysis
- Can conduct graph analysis and visualisation in `networkx`

Graph metrics and algorithms:

- Shortest path
- Graph profiling: diameter, degree distribution, clustering coefficient
- Centrality: degree, PageRank, betweenness, closeness, ...
- Community detection

Challenges:

- Generate the Zachary's Karate Club data: https://en.wikipedia.org/wiki/Zachary's_karate_club .
- [SNAP dataset](#)
- [Cosponsorship Network Data](#)
- Analyse the [Les Misérables' graph data](#).

References:

- [數據新聞：政商網絡系列（下）（文：陳電鋸）](#) -- articulation via centrality
- [Clustering Game of Thrones](#) -- application of community detection
- 大家都叫我老楊, [推特上有多少「新五毛」？](#). The analysis is done in R but the dataset and topic is interesting to look at.
- Some books for further reading: <http://www.socilab.com/#books>

Week 12 - 2D analysis

Objective

- Understand correlation and can calculate correlation
 - Can articulate correlation and causality
-

Following are advanced topics for your own reading. We do not discuss those topics due to lack of regular class hours.

Week 13 - High-dimensional analysis

Objective:

- Understand correlation and causality. Can conduct visual (explorative) analysis of correlation
- Can interpret common statistic quantities
- Dimensionality reduction

Challenge:

1. Explore the HK Legco voting records

Modules:

- `sklearn`
 - `decomposition.PCA`
- `seaborn`
- (optional) `scipy.statsmodel`

References:

- [HK Legco 2012 - 2016 dataset from Initium Media, 2016](#)

- [HK Legco voting analysis](#) with PCA, an early version, 2014.

Week 14 - Clustering

Week 15 - Classification

Week 16 - Regression

Week 17 - Recommender System

Open topics

Those topics may be discussed if there is plenty Q/A time left in certain week. Or, you are welcome to explore those topics via group project.

- Cloud (AWS)
- Deep learning

Week 00: Introduction and Preparation

- [Week 00: Introduction and Preparation](#week-00-introduction-and-preparation) - [Foreword](#foreword) - [Course structure](#course-structure) - [Introduction & Objectives](#introduction--objectives) - [Getting-started on GitHub](#getting-started-on-github) - [Understand markdown](#understand-markdown) - [Headings](#headings) - [Listing](#listing) - [Emphasis](#emphasis) - [URL](#url) - [Image](#image) - [Inline code snippet](#inline-code-snippet) - [Code block & Syntax Highlighting](#code-block--syntax-highlighting) - [Quote](#quote) - [Use GitHub issue tracker as a discussion forum](#use-github-issue-tracker-as-a-discussion-forum) - [Learn other's code from commit history](#learn-others-code-from-commit-history) - [Preview a Jupyter notebook hosted on GitHub with Nbviewer](#preview-a-jupyter-notebook-hosted-on-github-with-nbviewer) - [Jupyter Notebook](#jupyter-notebook) - [Nbviewer](#nbviewer) - [Why do we need to preview Jupyter notebook on Nbviewer?](#why-do-we-need-to-preview-jupyter-notebook-on-nbviewer) - [Other uses of GitHub for public discussions](#other-uses-of-github-for-public-discussions) - [GitHub Desktop](#github-desktop) - [Install GitHub Desktop](#install-github-desktop) - [Use GitHub Desktop to synchronize codes between local repository and GitHub hosted repository](#use-github-desktop-to-synchronize-codes-between-local-repository-and-github-hosted-repository) - [Create your first repository](#create-your-first-repository) - [Create a file in this repository](#create-a-file-in-this-repository) - [Publish your repository to GitHub](#publish-your-repository-to-github) - [Re-edit your file and synchronize codes between two ends](#re-edit-your-file-and-synchronize-codes-between-two-ends) - [GitHub Pages](#github-pages) - [Publish your first webpage on gh-pages](#publish-your-first-webpage-on-gh-pages) - [Step 1. Create a repository](#step-1-create-a-repository) - [Step 2. Clone the repository](#step-2-clone-the-repository) - [Step 3. Create an index file](#step-3-create-an-index-file) - [Step 4. Commit & publish](#step-4-commit--publish) - [Bonus: Add sub-path to your GitHub hosted domain](#bonus-add-sub-path-to-your-github-hosted-domain) - [Bonus: Bind a custom domain name](#bonus-bind-a-custom-domain-name) - [Step 1. Purchase domain](#step-1-purchase-domain) - [Step 2. Point domain name to GitHub](#step-2-point-domain-name-to-github) - [Step 3. Point GitHub repo to domain name](#step-3-point-github-repo-to-domain-name) - [The principle further explained](#the-principle-further-explained) - [Bonus: Collaboration on GitHub](#bonus-collaboration-on-github) - [How to collaborate with your teammates](#How-to-collaborate-with-teammates) - [Step 1. Invite the collaborators](#step-1-Invite-the-collaborators) - [Step 2. Sent invite link](#step-2-Sent-invite-link) - [The workflow Fork repo, modify code and send pull Request](#the-workflow-fork-repo-modify-code-and-send-pull-request) - [Fork repo](#fork-repo) - [Modify code](#modify-code) - [Pull Request](#pull-request) - [A first journey into the open source world](#a-first-journey-into-the-open-source-world) - [Common licenses](#common-licenses) - [Ethics and Code of conduct](#ethics-and-code-of-conduct) - [Exercises and Challenges](#exercises-and-challenges) - [Browse past student projects](#browse-past-student-projects) - [Raise a question](#raise-a-question) - [Participate in a discussion thread](#participate-in-a-discussion-thread) - [Bonus: conduct a survey of related reading materials on GitHub](#bonus-conduct-a-survey-of-related-reading-materials-on-github) - [Bonus: fork - modify - Pull Request (PR)](#bonus-fork---modify---pull-request-pr) - [References and further reading](#references-and-further-reading)

Foreword

The course COMM7780/JOUR7280 Big Data for Media and Communication is set up for master students in the school of communication, Hong Kong Baptist University. The purpose of this course is to motivate the students to become a **T-shape talent** in communications field. The course involves intensive training of Python and quest in solving practical problems. This open book collects all the materials related with lab exercises covering Python basics, data scraping, table manipulation and data mining. Students also apply their duly learned knowledge to write data-driven reports on a regular basis.

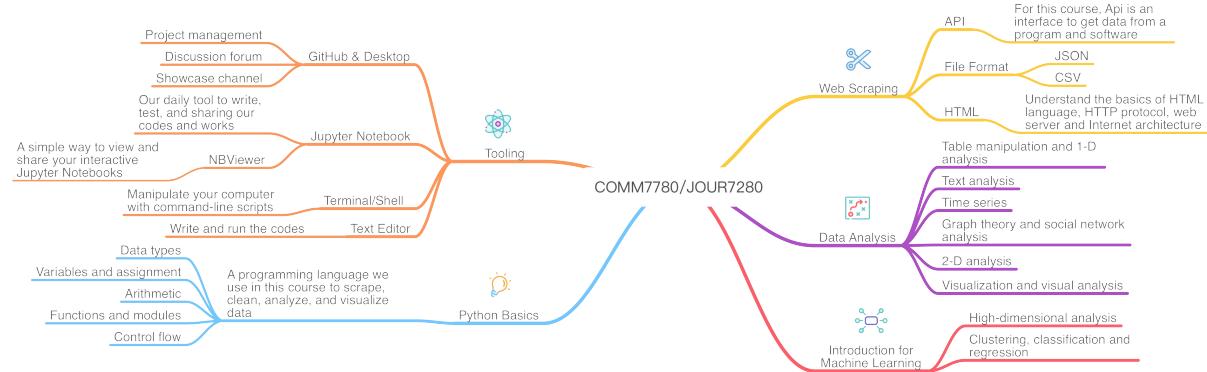
The final projects of past students can be found in this archive GitHub organization: <https://github.com/data-projects-archive/>

This course is tailor made for communication background students and imposes no prerequisites. However, one need to get prepared to the following challenges:

1. Become a native GitHub resident -- join discussions, hand-in homeworks and collaborate projects, all on [GitHub](#).
2. Expect intensive coding exercises in Python. Every class is about half lecture and half lab exercise. You will also have many exercises after the class.
3. Identify and solve practical problems in one's own domain, that can be tackled by a limited set of duly learned data analytics knowledge. Every week, one group will work on a real case using that week's knowledge.

Course structure

Figure last updated: 20180803



Here is a brief map about what we are going to learn in the first semester. For more details, please refer to our [course outline](#).

Introduction & Objectives

This chapter covers "GitHub literacy" and walks the readers through the basic steps towards an open source learning experience. Following are the concrete objectives:

- Can use GitHub for resource hosting, project management and discussion forum.
- Can use GitHub Desktop to sync local repository with remote repository.
- Can use `gh-pages` to host static web pages as one's portfolio.

Chapter 0 is intended for anyone who wants to enroll in this course to study before the class. Besides learning GitHub, we also expect the readers to "tune in" our open book. You should feel conformable navigate around our Openbook, identify developed/ working-in-progress files, contribute ideas/ codes, and actively participate in the issue tracker.

Note that sections marked "**Bonus:**" are optional materials. You can still follow the rest of the course without learning those sections.

Getting-started on GitHub

You can sign up in [GitHub](#). Pick a nice name, then you can start your wonderful journey conquering the galaxy of data and codes. There are many useful learning materials and meaningful projects on the platform waiting for you to open it, many interesting stories to discover.

Understand markdown

Markdown is a lightweight and easy-to-use syntax for styling all forms of writing on the GitHub platform.

-- Here is the explanation by GitHub.

Markdown is an easy-to-read and easy-to-write markup language. Computer programming language usually needs to be compiled or interpreted or rendered in order to get readable result from the source code. Markdown syntax is designed in a way that makes it readable in both rendered (HTML) format or source code format. The conventional extension for a markdown file is `.md` and the content is just plain text.

As a beginner, one needs to master those syntax:

- Headings
- Listing, ordered or unordered
- Emphasis
- URL
- Image
- Inline code snippet
- Code block & Syntax Highlighting
- Quote

Headings

```
# H1  
## H2  
##### H6
```

Listing

Unordered:

```
* Item 1  
* Item 2  
  * Item 2a  
  * Item 2b
```

Ordered:

```
1. Item 1  
2. Item 2  
3. Item 3  
  1. Item 3a  
  2. Item 3b
```

Emphasis

```
Strong emphasis, aka bold, with **asterisks** or __underscores__.
```

URL

```
[text](link) For example: [GitHub](http://github.com)
```

Image

```
![Alt Text](url/file_path) For example:  
![Course Structure](assets/course-structure.png)
```

Inline code snippet

```
`In line code here`
```

Code block & Syntax Highlighting

1. Use ```` in the beginning and end of the code to build the code block.
2. Add tag after the beginning ```` to highlighting the code, like python, text, md, javascript...

```
```javascript
var s = "JavaScript syntax highlighting";
alert(s);
```

```
```python
s = "Python syntax highlighting"
print s
```

```
No language indicated, so no syntax highlighting.
But let's throw in a <b>tag</b>.
``
```

```
#### Quote
```

```
```text
> Quotes here.
```

Here are some readings to get you started with markdown:

- See the markdown source code of [the current chapter](#) on GitHub.
- Go to the issue tracker (introduced in next section) and "edit" any message. See how markdown works in the discussion thread.
- The official document of [Mastering Markdown](#) from GitHub.
- Another good introduction of use of markdown [Markdown Syntax](#) (in Chinese)

Bonus: Insert a internal link between different sections in different chapters

Example: I want to insert a link in chapter 2 `array` section which connect to one of the example of chapter 3.

```
[chapter 2](/notes-week-02.md#python-has-two-basic-modes-script-and-interactive)
```

- `[chapter 2]` is the text you want to insert a link.
- `/notes-week-02.md` is the file path you want to connect to.
- `python-has-two-basic-modes-script-and-interactive` is the header of specific section you want to connect to. You can get this by lower all the letters of the header and connect them with `-`, and do not forget to use `#` ahead of the header.

## Use GitHub issue tracker as a discussion forum

The GitHub issue has a lot of features, but overall it looks like a lightweight collaboration system. Assignee or project manager can pull requests for new contributors and set a to-do list for contributors and teammates. You can use issue to ask questions, discuss with your team, label the `issue` you encounter, and collaborate with others, which greatly

advances the managing of a Project. You can click [here](#) to participate in the issues channel of our Gitbook to discuss with us. Feel free to leave your comments or questions so that we can help each other and learn from each other.

**Example 1:** Set a to-do list for your teammates, once they finished the quest you pull, they can just tick to show the progress.

 hupili commented 7 days ago • edited by ChicoXYC ▾

Owner +  

<https://github.com/hupili/python-for-data-and-media-communication-gitbook/blob/master/notes-week-01.md>

Tick when resolved:

- <https://github.com/hupili/python-for-data-and-media-communication-gitbook/blob/master/notes-week-01.md#how-to-open-terminal-on-mac> . Add a screenshot to show spotlight search result.
- <https://github.com/hupili/python-for-data-and-media-communication-gitbook/blob/master/notes-week-01.md#2-shell-commands> . Add one screenshot of an entire Terminal to show multiple commands and their input/ output
- <https://github.com/hupili/python-for-data-and-media-communication-gitbook/blob/master/notes-week-01.md#2-shell-commands> . Use markdown "block code" notation to demo the input/ output of commands, **instead of** screenshots (images). This way looks better and also allows readers to copy and paste.
- <https://github.com/hupili/python-for-data-and-media-communication-gitbook/blob/master/notes-week-01.md#2-shell-commands> Please explain the components of the interactions in shell. That is, what is the \$ ? the username / path before \$ ? The user's input and the shell's output. Think of the questions that first time readers would ask. Use this screenshot, plus some annotations, to explain in details.
- Please turn them those pointers into URLs. "*(If it doesn't work, you can download some third party editors, such as sublime, visual studio code. You can edit .py file by these editors.)*"
- Rename the screenshot files to readable filename, i.e. a **slug-style** filename. Do not simply use the default "Screenshot ..." name. Or we will lose track in the future.

---

Use the following thread for discussion.

  hupili assigned ChicoXYC 7 days ago

**Example 2:** Discuss with your team members to track the working process.

The screenshot shows a GitHub commit history with three comments:

- ChicoXYC commented a day ago**: @bonnie-yu @hupili thanks for your feedback! I will modify soon.
- bonnie-yu commented a day ago • edited**: Should advise 1) the suggested python version, 2) how to use the command "python" to check python version, 3) method to make Python 2.7 and Python 3 coexist in the very beginning of chapter 1. So learners don't need to get back to these fundamentals in the half of learning.
- ChicoXYC commented 2 hours ago**: All problems discussed above has been modified, and add several supplements.
  - The relationship between text editor and terminal, and why we need to use text editor.
  - How to install python 3, the difference between python 2 & 3.
  - modify the materials includes first question.md and python2 vs 3.md .

## Learn other's code from commit history

<a href="#">notes-week-00.md</a>	Add Chapter 0 framework	7 days ago
<a href="#">notes-week-01.md</a>	Update notes-week-01.md	an hour ago
<a href="#">notes-week-02.md</a>	Update notes-week-02.md	7 days ago
<a href="#">notes-week-03.md</a>	Creates assets/Screen Shot 2018-03-06 at 1.10.13 AM.png	5 months ago
<a href="#">notes-week-04.md</a>	Updates notes-week-04.md	5 months ago
<a href="#">notes-week-05.md</a>	Updates notes-week-05.md	5 months ago
<a href="#">notes-week-06.md</a>	Creates assets/Screen Shot 2018-03-14 at 9.29.07 pm.png	4 months ago

You can see the latest update time and a brief summary of every piece of work in one's repository.

The screenshot shows a GitHub commit page for the file `notes-week-01.md`. The commit was made by `ChicoXYC` an hour ago. It has 1 parent commit `70eb8a3` and a commit hash `8f93cb3013483725a13c8cd709c0565affaaaf0ac`. The commit message is: "ChicoXYC committed an hour ago Verified". The commit details show 1 changed file with 1 addition and 1 deletion. The unified view of the diff shows the following changes:

```

@@ -117,7 +117,7 @@ Terminal is an interactive environment. The advantage of writing code inside is
117 117 We recommend two text editors, [sublime](https://www.sublimetext.com/) and [visual studio code](https://code.visualstudio.com/)(*click to download*). Then you can edit a .py file by these editors by double clicking the file. MAC will open "TextEdit" by default editor. You can set one of those two editors as default editor if necessary.
118 118
119 119 ### Install python 3
120 -Python is a popular programming language that is widely used by beginners and longtime developers alike. Meanwhile, its the language that we learn in this course to scrape, clean, analyze, and visualize data. And there are basically 2 main versions of python. Python 2 and 3. But in this course, we base our discussions and exercises on Python 3, and you can check out the [difference between python 2 and 3](/python-2-vs-python-3.md) and the instruction for [installation of python 3](/first-question.md).
120 +Python is a popular programming language that is widely used by beginners and longtime developers alike. Meanwhile, its the language that we learn in this course to scrape, clean, analyze, and visualize data. And there are basically 2 main versions of python. Python 2 and 3. But in this course, we base our discussions and exercises on Python 3, you can check out the [difference between python 2 and 3](/python-2-vs-python-3.md) and the instruction for [installation of python 3](/first-question.md) in related materials in our gitbook.
121 121
122 122 * Modify the `ex1.py` file by a text editor.
123 123

```

By clicking one of the files, and its commits, you can see the details of the improvements and changes they recently made, and you can learn from their work.

Red means deletion and green means addition. This notation makes it easy for you to track the difference and learn what others are doing.

## Preview a Jupyter notebook hosted on GitHub with Nbviewer

### Jupyter Notebook

[Jupyter Notebook](#) is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. In our course, Jupyter notebook will be our daily tool to write, test, and sharing our codes and works. It's very useful for us to learn and make **reproducible works**. The good advantage of jupyter notebook includes:

- The Notebook has support for over 40 programming languages, including Python, R, Julia, and Scala.
- Notebooks can be shared with others using email, Dropbox, GitHub and the Jupyter Notebook Viewer.
- Your code can produce rich, interactive output: HTML, images, videos, LaTeX, and custom MIME types.

You can check out our [tutorial](#) for how to install Jupyter environment and use Jupyter notebook, though, it's still in improvement stage.

### Nbviewer

[Nbviewer](#) creates a simple way to view and share Jupyter Notebooks. You just need to copy the link of one Jupyter notebook and paste in Nbviewer. Click [here](#) to know more about Nbviewer,to see what formats they support to present notebooks to the user.

Here is an example, you can find one Jupyter notebook from github, and copy the link, paste into the Nbviewer.

[Nbviewer](#) creates a simple way to view and share Jupyter Notebooks. You just need to copy the link of one Jupyter notebook and paste in Nbviewer. Click [here](#) to know more about Nbviewer, to see what formats they support to present notebooks to the user.

Here is an example, you can find one Jupyter notebook from github, and copy the link, paste into the Nbviewer.

copy the jupyter notebook's link

Branch: master ▾

HKBU-BIG-DATA-MEDIA / Final Project - Airplane crash / A Brief report Airplane crash worldwide.ipynb

ChicoXYC Add files via upload aad3a32 on 1 May

1 contributor

2.43 MB

In [37]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import plotly
import plotly.plotly as py
import plotly.graph_objs as go
import matplotlib.pyplot as plt
from datetime import date, timedelta, datetime

Data = pd.read_csv('Airplane_Crashes_and_Fatalities_Since_1908.csv')
```

Download History

jupyter  
nbviewer

# nbviewer

A simple way to share Jupyter Notebooks

Enter the location of a Jupyter Notebook to have it rendered here:

URL | GitHub username | GitHub username/repo | Gist ID

Go!

## Why do we need to preview Jupyter notebook on Nbviewer?

Usually, we can directly preview a Python notebook on GitHub. However, GitHub prohibits Javascript execution for security reasons. If you have interactive chart, e.g. `echart`, `plotly`, those will not render on GitHub. Nbviewer supports javascript and it is the first free online tool to preview Python notebook, so we recommend it. For concrete examples of dynamic charts, here is an [example](#) from students' last year. The last interactive map `flight path` can not be shown directly in GitHub, that's why we need Nbviewer.

## Other uses of GitHub for public discussions

GitHub's versioned storage and its adoption of markdown as primary syntax makes it very easy to maintain content on GitHub. Other users can also participate in the discussion via GitHub issue tracker. There were some practices of using GitHub to discuss public events. Following are some pointers for your reference:

# GitHub Desktop

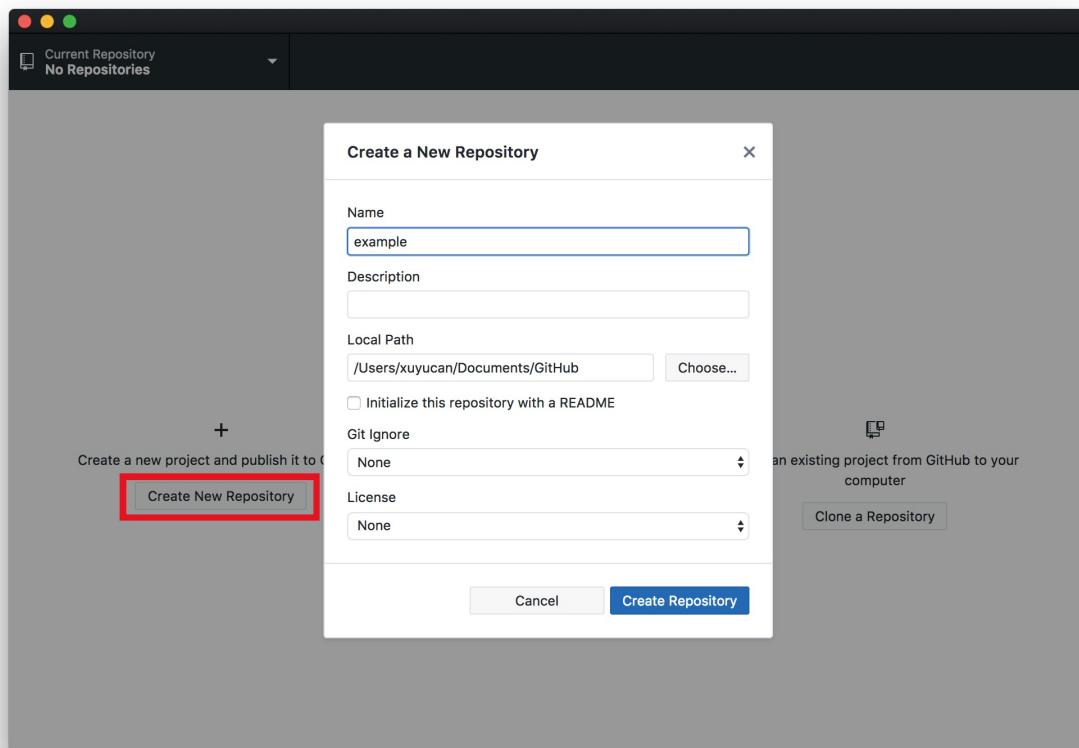
## Install GitHub Desktop

You can download from [here](#), and install it like you installed other apps before.

## Use GitHub Desktop to synchronize codes between local repository and GitHub hosted repository

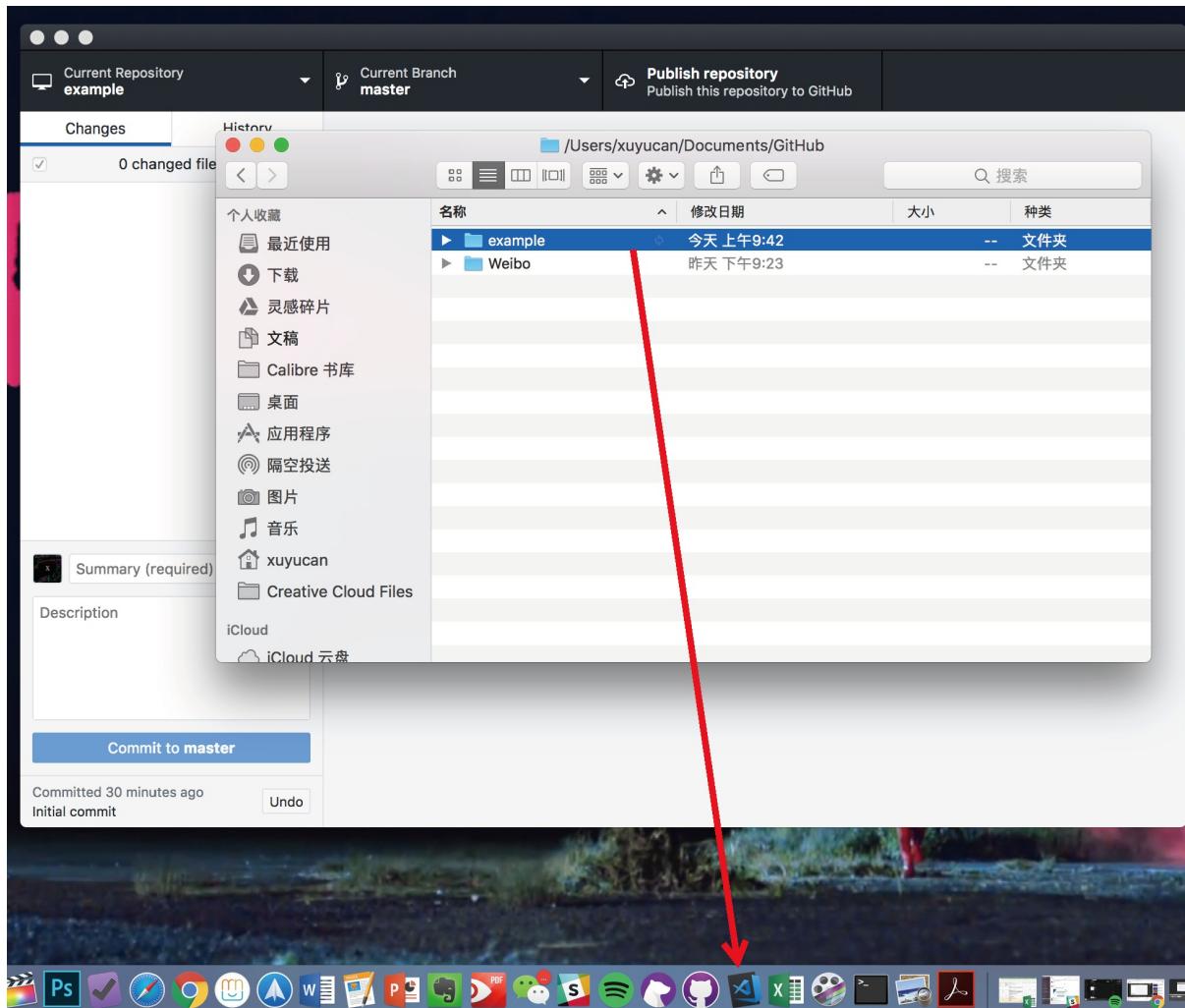
Talking about this function, GitHub is like a cloud disk, which is similar to Google Drive, OneDrive. You can synchronize your codes and files between local and GitHub website. It's useful not just others can see your recent updates, but also improve the efficiency during the collaboration with others. If one of your teammates commit changes, you can synchronize by GitHub desktop and keep the same stage with them.

## Create your first repository

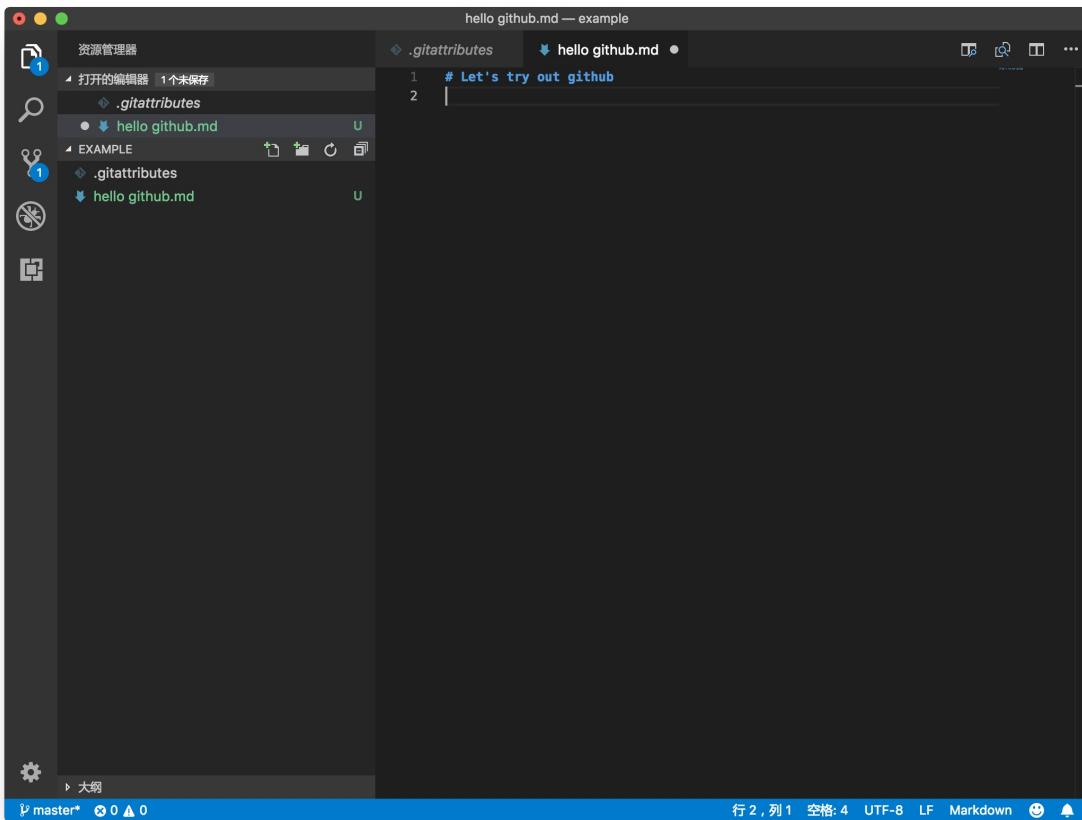


After you log in GitHub desktop, click `create new repository`, give a name you like and choose the local path you want(but keep in mind where they are).

## Create a file in this repository

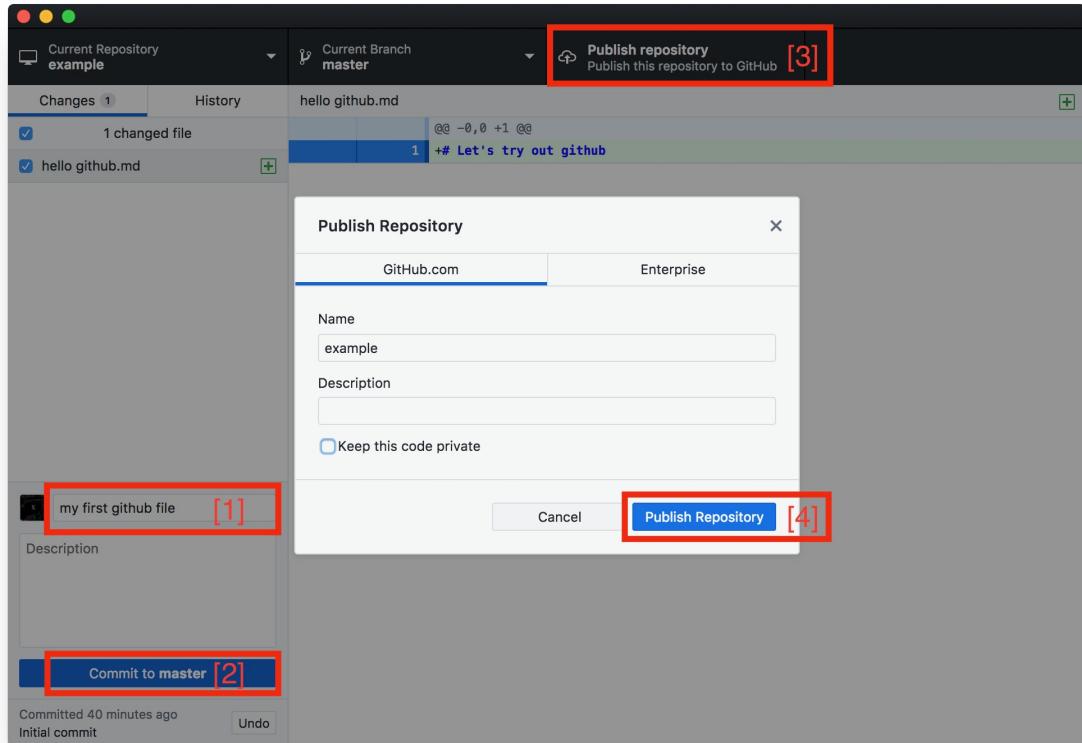


Find the repository/folder you just created, you can click [open this repository in finder](#) to know its file path. After you find the folder, drag the repository/folder into the text editor, in this case, I use *visual studio code*.



Under the example repository, create your first file `hello github.md`, write an H1 line `Let's try out github`, and save it.

## Publish your repository to GitHub



After creating your first file, open GitHub desktop, you can see the changes you did before. Give a description of this changes in [1], then commit to master in [2],(this step is like that you confirm the changes). After that, you can click [3] and [4] to publish your repo.

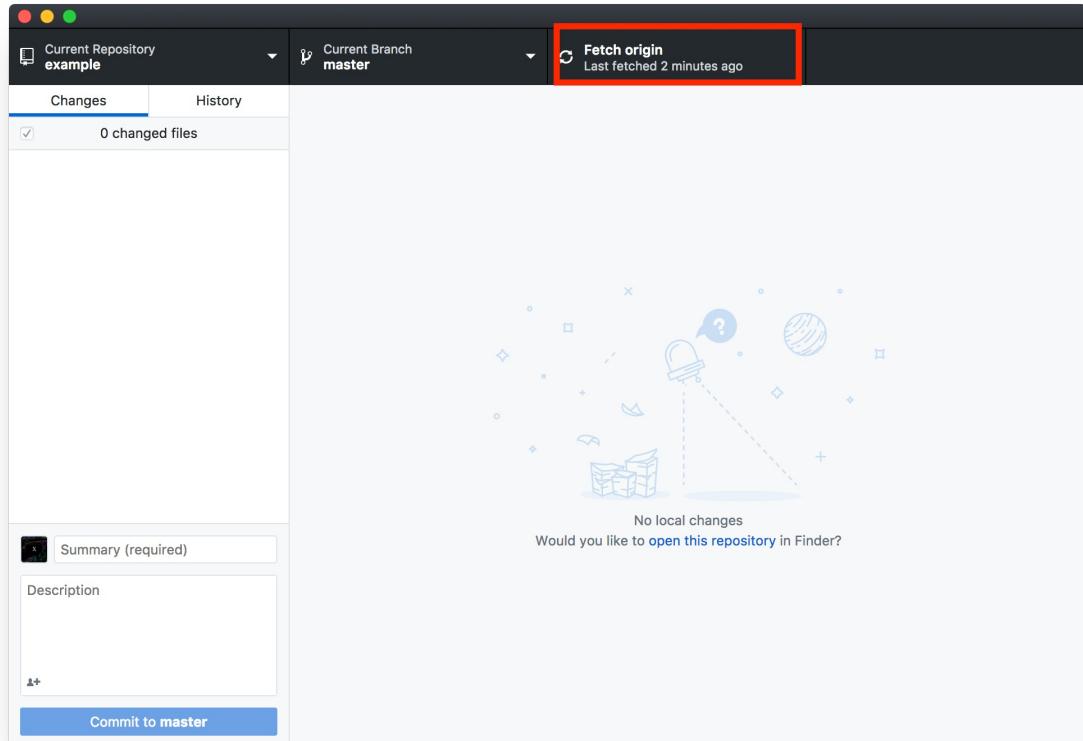
The screenshot shows the GitHub homepage. At the top, there's a banner with the text "Learn Git and GitHub without any code!" and a message about creating a repository, starting a branch, writing comments, and opening a pull request. Below the banner are two buttons: "Read the guide" (in green) and "Start a project". A notification box in the top-left corner informs users that new Terms of Service and Privacy Statement are in effect. The main content area has three sections: "Repositories" (listing repos like hupili/python-for-dat..., ChicoXYC/HKBU-BIG-DA..., etc.), "Recent activity" (listing a recent assignment to Chapter 0 first version content), and "All activity" (listing a starred event for gka starring lipis/flag-icon-css). The repository ChicoXYC/example is highlighted with a red box.

Open the GitHub website, find your new repo, click the repo and check out the files in this repo, whether it keeps the same pace with your local file.

## Re-edit your file and synchronize codes between two ends

You can re-edited the files and codes both in GitHub website and in local text editor.

- If you edit in the website, after you save the file, click the `fetch origin`. Then the change you commit in the website will synchronize into your local repo.



- If you edit in a local text editor, it's pretty much the same. You give a description, commit to master, and then `fetch origin`. Then the change you commit in a local repo will synchronize into your GitHub site.

## GitHub Pages

GitHub Pages are websites for you and your projects. It helps you turn your GitHub repository into elegant websites to showcase your portfolio, your projects, documentation or anything you want to share with the world. There is no need to set up the database and configure servers. It's the most direct path to create websites for you and your projects. You can visit [GitHub Pages](#) to learn more.

### Publish your first webpage on gh-pages

#### Step 1. Create a repository

Head over to GitHub and create a new repository named `username.github.io`, where username is your username on GitHub.

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner                          Repository name

 ChicoXYC  

Great repository names are short and memorable. Need inspiration? How about **effective-octo-tribble**.

Description (optional)

 **Public**  
Anyone can see this repository. You choose who can commit.

 **Private**  
You choose who can see and commit to this repository.

**Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** ▾      Add a license: **None** ▾      ⓘ

**Create repository**

*Note: If the first part of the repository doesn't exactly match your username, it won't work, so make sure to get it right.*

## Step 2. Clone the repository

There are basically two ways to clone the repository. Use terminal with codes or GitHub desktop. Since we just talk about the GitHub desktop. So we use this way to check out whether we are already familiar with it.

Click the "Set up in Desktop" button. When the GitHub desktop app opens, save the project.

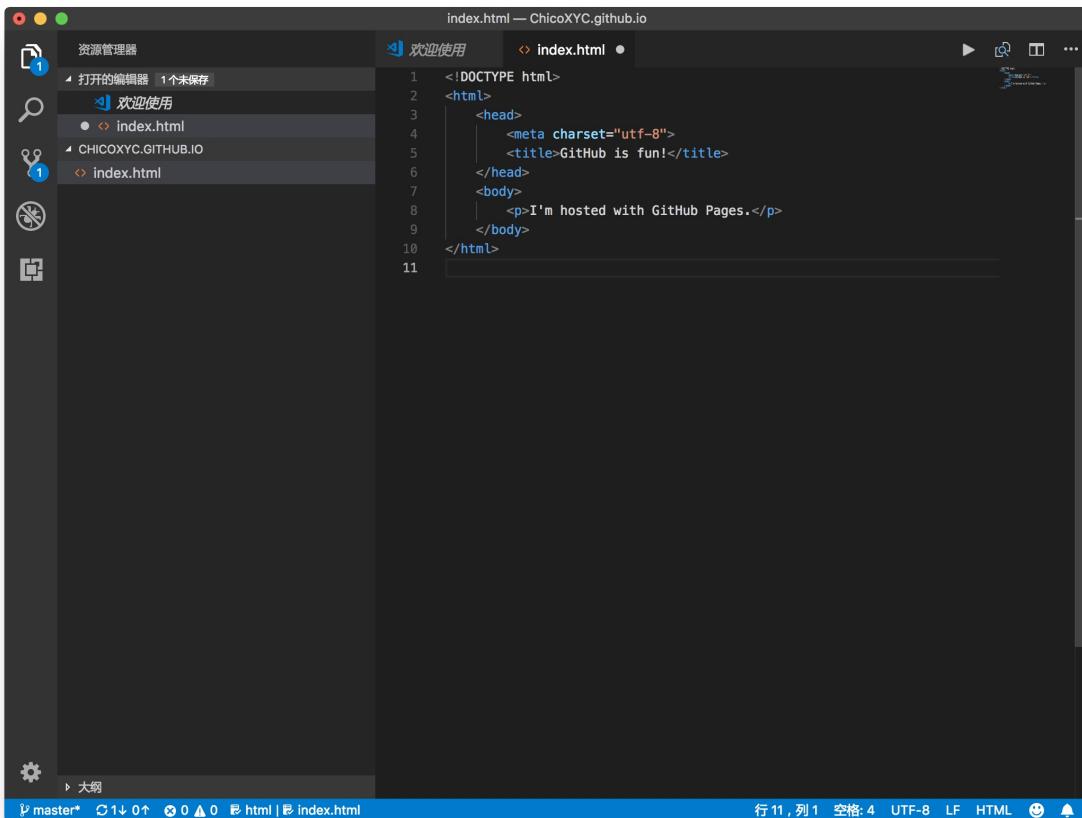
### Quick setup — if you've done this kind of thing before

 **Set up in Desktop**    or    **HTTPS**    **SSH**    `git@github.com:ChicoXYC/ChicoXYC.github.io.git`

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

## Step 3. Create an index file

Basically, `index.html` is the default file served by the web server. So it is equivalent to visit `example.com` and `example.com/index.html`. Naming your file as `index.html` can lead to this more concise notation in browser's address bar and in communication campaigns -- the naming in the world of web is usually the shorter the better. More explanations are [here](#).

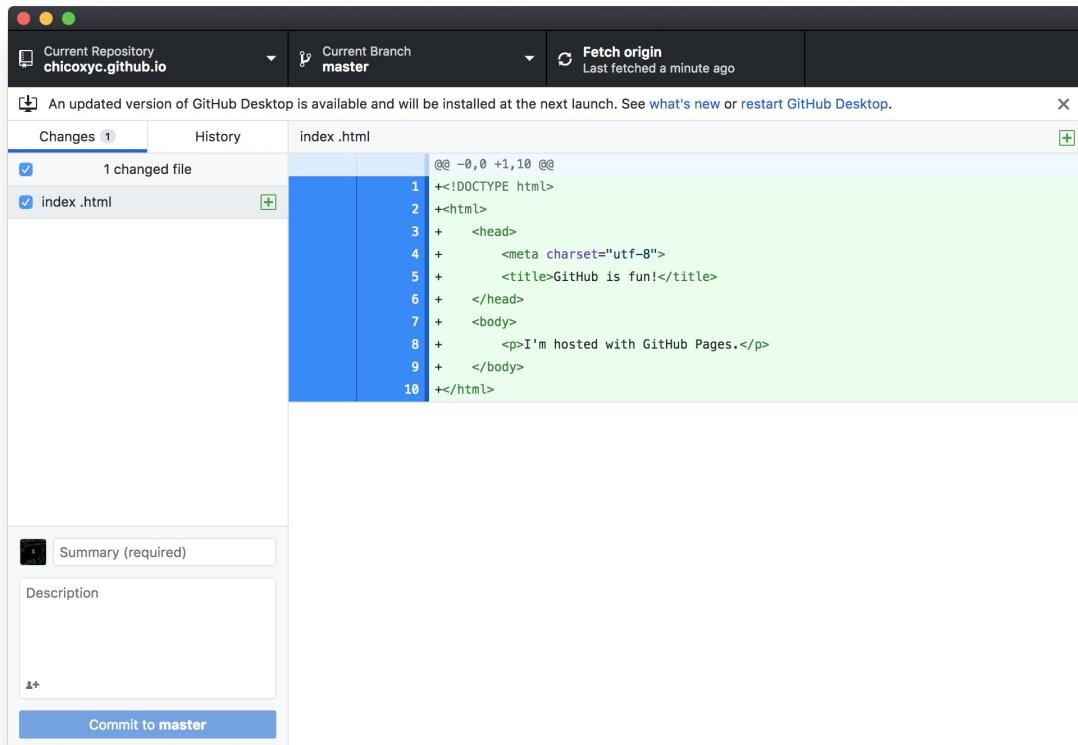


Grab your favorite text editor and add an index.html file to your project. You can copy this example in your html.

```
<!DOCTYPE html>
<html>
 <head>
 <meta charset="utf-8">
 <title>GitHub is fun!</title>
 </head>
 <body>
 <p>I'm hosted with GitHub Pages.</p>
 </body>
</html>
```

#### Step 4. Commit & publish

Change to your GitHub desktop, commit your changes, and press the publish/push button.



Then you can go to your webpage with <https://username.github.io>. Change username to yours and see what is happening. During further study, you will use GitHub pages to do more, to share and show anything you want with the world. For example, you can put your own CV, your portfolio, your essay of learning Python etc into html. For example, I build a simple introduction/CV for myself, you can check out here: <https://chicoxyc.github.io> .

## Bonus: Add sub-path to your GitHub hosted domain

By default, every user gets a secondary domain name at `username.github.io` where `username` is your GitHub username and `username.github.io` is also the repo name. It is of practical interest to add a sub-path to your domain for many reasons:

1. Test
2. Hand-in homework
3. Work on projects

Suppose you already have everything in a local folder called `homework1` . There are two ways to publish this folder at the URL <https://username.github.io/homework1> :

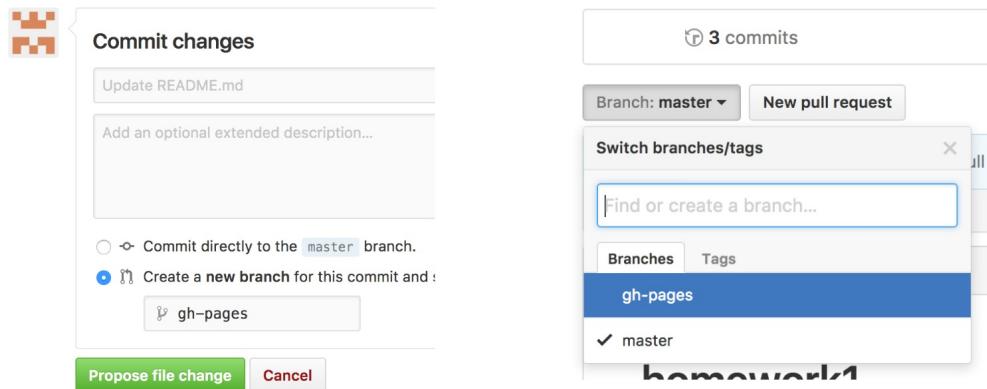
1. Host via the main repo:
  - o Upload `homework1` to the root of repo `username.github.io` directly.
2. Host via a separate repo:
  - o Create a new repo called `homework1` ;
  - o Create a branch called `gh-pages` ;
  - o Sync (upload) the content of your local folder `homework1` to the repo `homework1` on the branch `gh-pages` .

We recommend the first way to host basic materials about yourself and the second way to host projects. The second way exercises the principle of [Separation of Concerns](#).

In short, if your username is `xxx`, then the following repositories will be hosted as web sites on corresponding paths:

Repo	Count	Branch	Website URL
<code>XXX.github.io</code>	1	master, gh-pages	<a href="http://XXX.github.io">http://XXX.github.io</a>
<code>&lt;repo-name&gt;</code>	No limit	gh-pages	<a href="http://XXX.github.io/&lt;repo-name&gt;">http://XXX.github.io/&lt;repo-name&gt;</a>

Here is one way you can create a new branch on GitHub web UI:



### Step 1:

Edit any file. Before you commit changes, select “Create a new branch...” and use the name “gh-pages”

### Step 2:

Next time when you want to modify this repo, select the “gh-pages” branch from the drop down first, before clicking “edit” or “create new file”

**NOTE:** The content is served from `gh-pages` branch by default. You can [change this default setting](#).

## Bonus: Bind a custom domain name

See it in action first:

- The live website: <http://datavis.studio/>
- The GitHub repo: <https://github.com/hkbujour2106/hkbujour2106.github.io>

### Step 1. Purchase domain

- name.com is one example
- You can choose other providers, e.g. namecheap.com , godaddy.com , ...
- For “.hk” domain name,  
<https://www.hkdnr.hk/>

The screenshot shows the name.com website interface. At the top, there are navigation links: Domains, Websites, Hosting & SSL, Email, Support, and Account. A search bar contains the text "datavis.studio". Below the search bar, it says "Your domain is available! SALE!" followed by "datavis.studio". To the right, the price is listed as "\$9.99" with an "Add to Cart" button. In the center, there's a table showing the items: "datavis.studio" (1 year, \$9.99) and "Whois Privacy" (\$4.99). To the right, an "Order Summary" box shows a total of "\$14.98" with a "Next Step: Payment" button highlighted with a red oval.

Your order is complete!

This screenshot shows the "Order Summary" page after a purchase. It lists the items: "Registration" for "datavis.studio" at \$9.99 and "Whois Privacy" for "datavis.studio" at \$4.99. The total amount is \$14.98. At the bottom, there are links: "Go to My Dashboard", "Manage Payment Profile", "View Invoice", and "Domain Details".

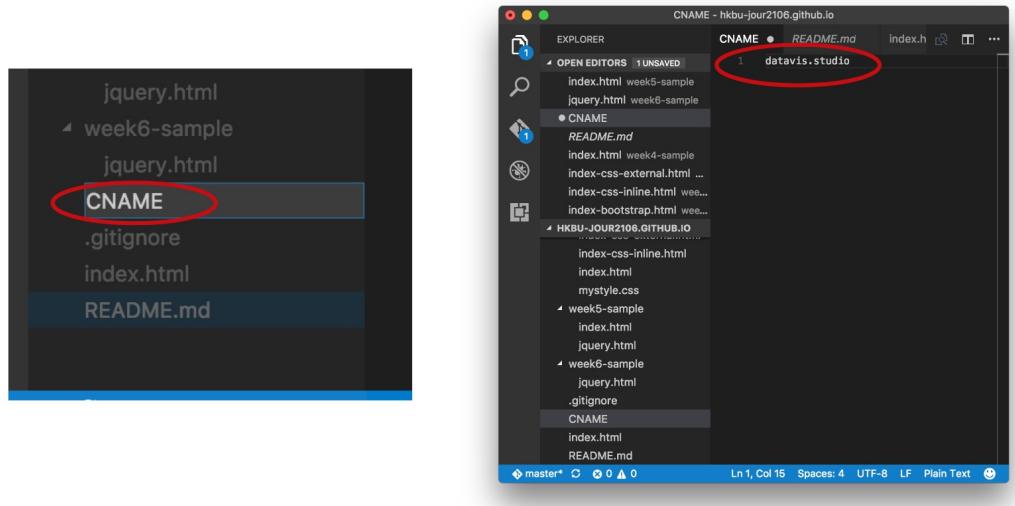
## Step 2. Point domain name to GitHub

- Add an “A record” of answer “151.101.64.133”
  - Note that there are many IP addresses that also work.
  - If you know command line (future topic), try “ping hkbu-jour2106.github.io” to get a sample address

The screenshot shows the "Edit DNS Records" page for "datavis.studio". A table lists existing records: one A record with Host "datavis.studio" and Answer "151.101.64.133", and another A record with Host ".datavis.studio" and Answer "151.101.64.133". A new row is being added, with the Type set to "A", the Host set to "datavis.studio", and the Answer set to "151.101.64.133". The "Actions" column includes "Edit" and "Delete" buttons for the existing records, and an "Add Record" button for the new row.

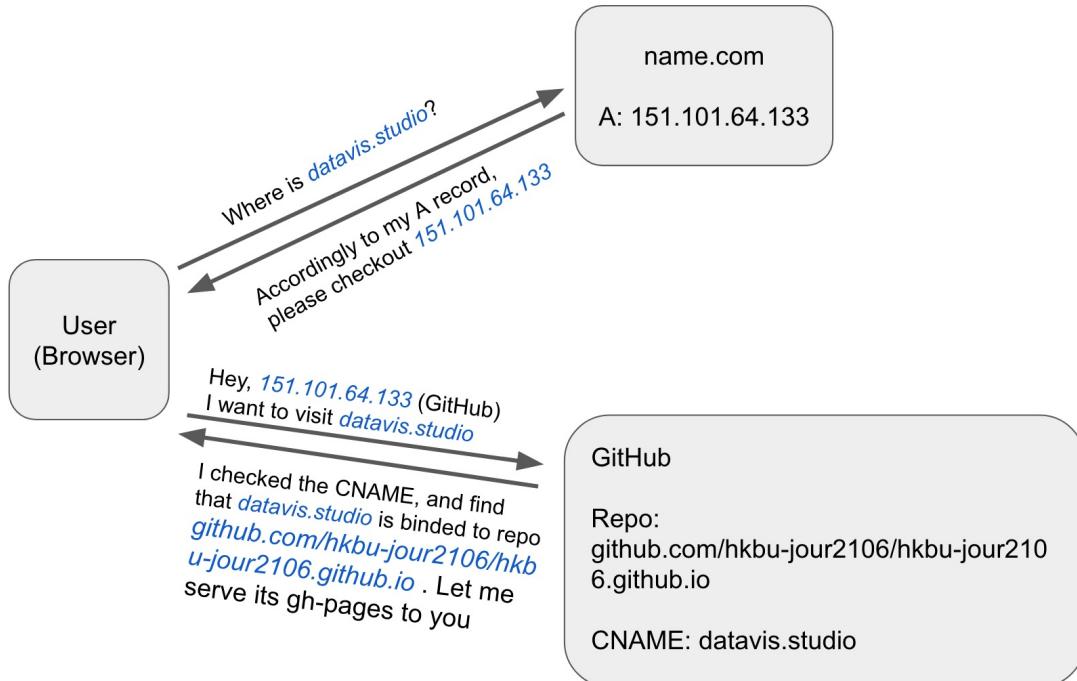
## Step 3. Point GitHub repo to domain name

- Create a files named “CNAME”
- In this file, put one line of your newly purchased domain name



Checkout the working CNAME

## The principle further explained



Checkout [this article](#) for how DNS works.

## Bonus: Collaboration on GitHub

### How to collaborate with teammates

#### Step1: Invite the collaborators

Find settings in repository and input the mail-address to add collaborators

A screenshot of a GitHub repository settings page. At the top, there are several tabs: Code, Issues (27), Pull requests (0), Projects (1), Wiki, Insights, and Settings. The Settings tab is highlighted with a red box. On the left, a sidebar has tabs for Options, Collaborators (which is selected and highlighted with a red border), Branches, Webhooks, Integrations & services, and Deploy keys. The main area is titled 'Collaborators' and shows one user 'Chico\_X' with the name 'ChicoXYC'. Below this is a search bar with placeholder text 'Search by username, full name or email address' and a note: 'You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.' At the bottom right of this section is a button labeled 'Add collaborator' with a red arrow pointing towards it.

## Step2: Sent invite link

After the collaborators accept the invitation, they are able to edit in the repository. If they didn't notice the invitation in the mail, you can send the invite link, like the below picture.

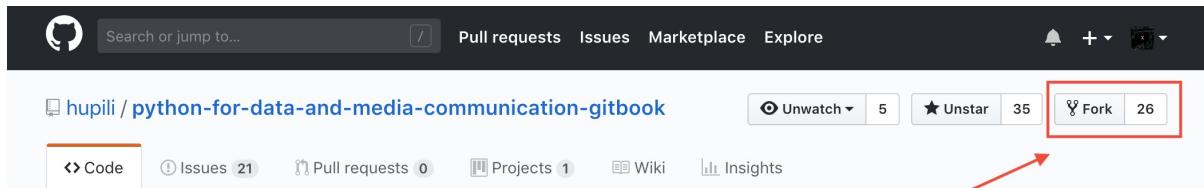
A screenshot of the GitHub repository 'Collaborators' page. It lists three users with pending invitations: 'Awaiting zacharyzeng's response', 'Josh Awaiting coolxu8888's response', and 'GAO Chao Awaiting FLYSTEPHEN's response'. Each user entry includes a 'Copy invite link' button (the third one is highlighted with a red box) and a 'Cancel invite' button. Below the user list is a search bar with placeholder text 'Search by username, full name or email address' and a note: 'You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.' At the bottom right is an 'Add collaborator' button.

## The workflow Fork repo, modify code and send pull Request

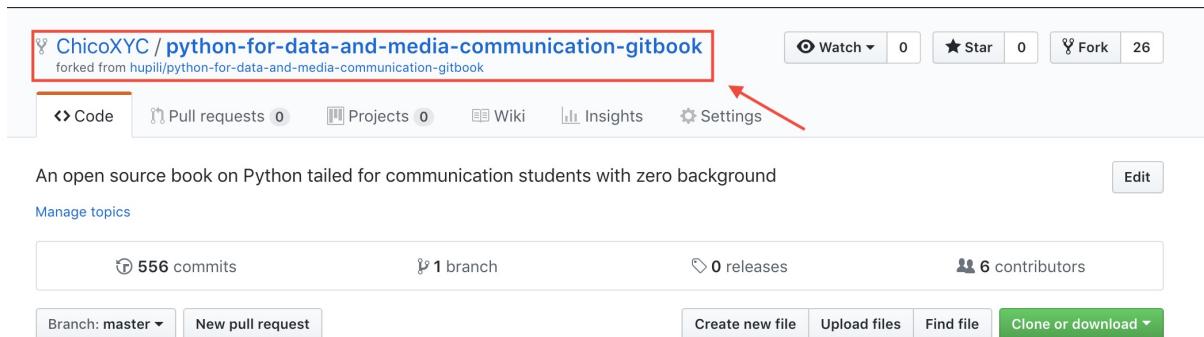
### Fork repo

Simply speaking, `fork` means to get a `copy` of a repo to your own account so that you can have authority to modify the codes, change files.

For example, one can fork a copy of our weekly notes [here](#):



After you forked, visit to your profile, you can find there is an copy of the our openbook in your repos. You can also find the information of the upstream repository.



## Modify code

After you get the copy of one repo, you can clone to the local, and the following are all we've already learned, modify the codes by text editor or add your Jupyter notebooks, commit your changes, push it to the remote.

## Pull Request

As we discussed, the forked repo is only a copy of the original, and in a collaborative projects, there might be many different forked repos, every part of the team modifies their codes and files separately. So how to merge those separate changes in to a complete one? `pull requests` is to solve the problem. After we pushed our changes to our repos. We can click `pull request`, give the summary of the request and commit. Basically, it tells the original repo owners that we want to merge our modified codes/files to the original repo. GitHub will automatically check whether your branch has conflicts with the original branch or not. If there is no conflict, the original owner can decide whether they accept your changes or not, if do, they will merge your changes into the original one. That's basically how we collaborate in GitHub.

ChicoXYC / [python-for-data-and-media-communication-gitbook](#)  
forked from [hupili/python-for-data-and-media-communication-gitbook](#)

[Watch](#) 0    [Star](#) 0    [Fork](#) 27

[Code](#)    [Pull requests 0](#)    [Projects 0](#)    [Wiki](#)    [Insights](#)    [Settings](#)

[Filters](#) ▾        [Labels](#)    [Milestones](#)

[New pull request](#)





Welcome to Pull Requests!

Pull requests help you collaborate on code with other people. As pull requests are created, they'll appear here in a searchable and filterable list. To get started, you should [create a pull request](#).

## Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also compare across forks.

others / original one to your repo

base fork: hupili/python-for-data-and... ▾ base: master ▾ ← head fork: ChicoXYC/python-for-data-... ▾ compare: master ▾

✓ Able to merge. These branches can be automatically merged.

→ Add to merge These branches can be automatically merged.

[click here](#)

 [Create pull request](#)

Discuss and review the changes in this comparison with others. [?](#)

---

 [1 commit](#)

 [1 file changed](#)

 [0 commit comments](#)

 [1 contributor](#)

---

 [Commits on Oct 10, 2018](#)

 [ChicoXYC](#)

[pull request test](#) ...

the information about the commit changes that you want to pull request

 [Verified](#)

a9a1bc

Showing 1 changed file with 1 addition and 0 deletions.

Unified Split

1 test

View   

```
... ... @@ -0,0 +1 @@
1 + this file is used for testing
```

base fork: [hupili/python-for-data-and-ml](#) · base: [master](#) · head fork: [ChicoXYC/python-for-data-...](#) · compare: [master](#)

✓ Able to merge. These branches can be automatically merged.

pull request test

Write Preview

AA B i “ <> ↲

create a new file

give the title and summary,  
then click

Attach files by dragging & dropping, [selecting them](#), or pasting from the clipboard.

Allow edits from maintainers. [Learn more](#)

Create pull request

Reviewers  
No reviews

Assignees  
No one—assign yourself

Labels  
None yet

Projects  
None yet

Milestone  
No milestone

The owner will decide whether accept your requests or not, if they merge the requests, the repo you want to merge will be updated.

The screenshot shows a GitHub pull request page. At the top, there are tabs for Conversation (0), Commits (1), Checks (0), and Files changed (1). Below the tabs, a comment from user ChicoXYC is shown, stating "create a new file". To the right of the comment are buttons for Collaborator, emoji, and more options. Below the comment, the pull request itself is visible, titled "pull request test". It includes a "Verified" badge and a commit hash "a9a1bd4". A note below the PR says "Add more commits by pushing to the master branch on ChicoXYC/python-for-data-and-media-communication-gitbook." At the bottom of the PR view, there is a large green button labeled "Merge pull request" with a checkmark icon. To the right of the button, it says "You can also open this in GitHub Desktop or view command line instructions."

**NOTE:** The Pull Request is not an one-off effort. If your modification is sub optimal, the maintainer of the project may request further revisions. The discussion thread of a PR is used for this purpose. GitHub's PR system is very expressive. You can also comment on a specific line of a specific version (commit) if you want. When there is new version, the initiator does not have to create a new PR. As soon as the initiator pushes new commits to the original source branch, the updates will appear in the PR thread. See this workflow in action from [this example](#).

## A first journey into the open source world

GitHub adopts a "sharing first" principle in its community. Whatever content you create *publicly* on GitHub is available for other GitHub users to view and reuse *within GitHub's functionality*. Some respectful and cautious users may not feel comfortable to work on derivatives of your work in other channels. You can grant them such permissions via some common open source license.

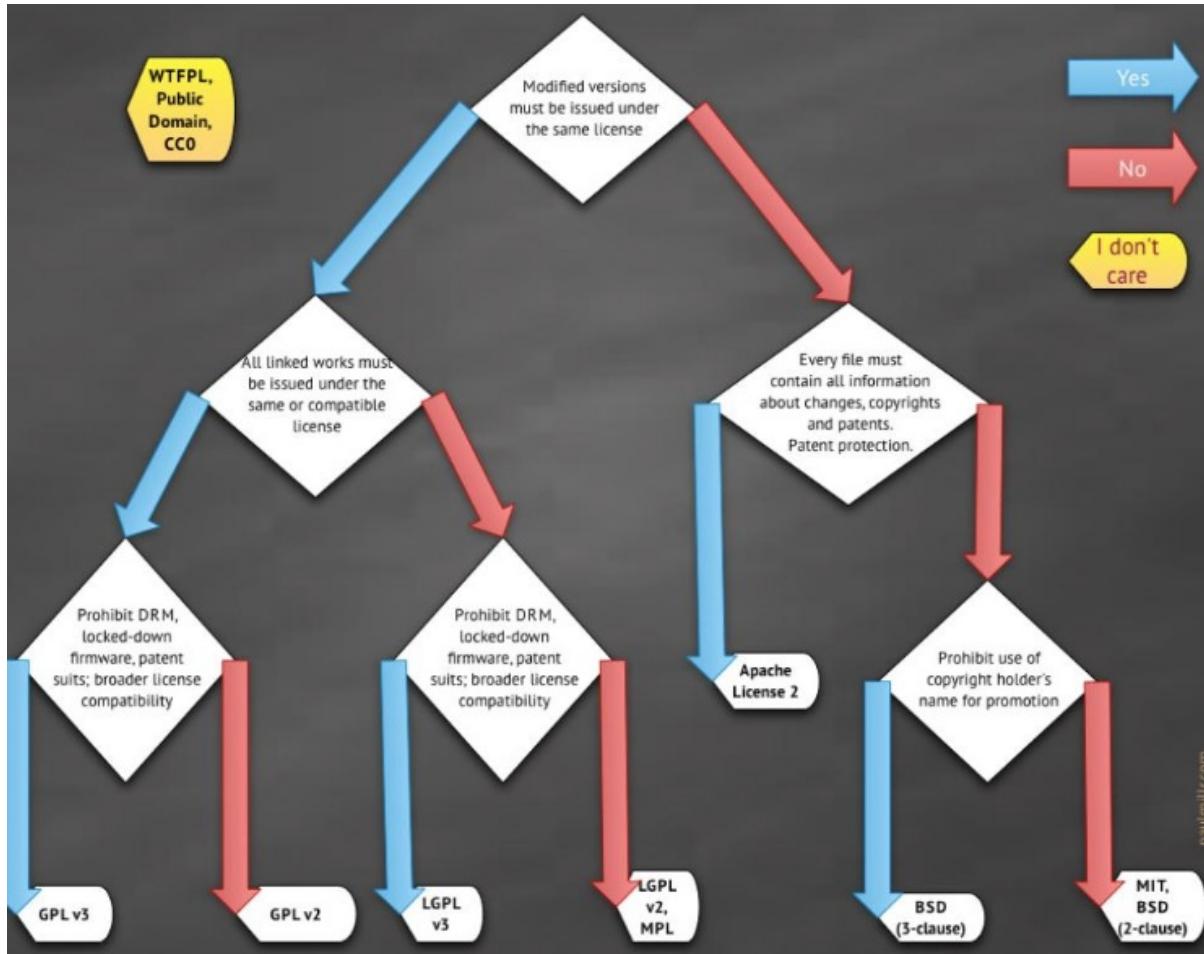
### Common licenses

The short take-away for:

- GPL -- You can use the content freely, but your derivative work based on a GPL licensed project is required to be open source.
- MIT -- Do whatever you want. You are obligated to provide attribution with your code or binary (e.g. say "this project uses code that is MIT licensed" -- with a copy of the license and copyright of the author of the open source code).
- Creative Commons -- There are several levels. CC0 is least restricted. It is basically "Public Domain". You can use other levels of CC or combinations of CC keywords to specify the usage rights. For example, this repo is licensed under "CC-BY-NC-ND", meaning:
  - BY: attribute to the author when you use
  - NC: non-commercial use only
  - ND: no derivatives -- you can share but can only share in its original format.

For homeworks, we usually use CC 4.0 license, please refer [here](#) for explanation.

Of course, the non granted rights are always negotiable with the authors who own the content. The spirit of open source is to cater to the greater good, while protect the authorship. There is a flow chart for you to decide what license you should choose.



## Ethics and Code of conduct

GitHub adopts a "**sharing first**" principle. In most cases, you can feel free to fork others project and drive the works within GitHub's eco-system. Even when the license is restrictive, you can still do so for study and research purpose, based on the commonly adopted "**fair use**" principle, under the condition that original authors are proper acknowledged.

When you talk with people on GitHub, try to keep the message precise and concise. This is to save everyone's time. Besides, please refrain from irrelevant discussions or meaningless messages. [This article](#) from PingWest pointed out to some common mis-conducts on GitHub.

## Exercises and Challenges

### Browse past student projects

- [Data Projects Archive](#) is the archive for our student data projects from various related courses. Try to navigate those repositories (repo) and:
  - Find the presentation slides and final report (if available).
  - Find the code and data.
  - Check the commit history to roughly understand how the project evolves.

- Challenge: before learning any programming or data analysis/ visualization skills, can you figure out the major steps of a data project by exploring Git?

## Raise a question

Go to the [issue tracker of this repo](#) and create a new issue to ask for a question.

## Participate in a discussion thread

Find a meaningful discussion thread on issue tracker and join the discussion. Sometimes, students might share their solution if they find alternative ways than the ones given in our text book. Here is [one example thread](#).

## Bonus: conduct a survey of related reading materials on GitHub

Try out GitHub's search function. See if you can find other open courses, books, tutorials, sample codes, that are related with this course. Please note down your findings. A short description and information about difficulty/ relevance will help your future reference.

## Bonus: fork - modify - Pull Request (PR)

Following up the previous challenge, you may want to suggest new entries into our reading material list on [reading-materials.md](#). Please operate in the following steps:

- Fork the current repo
- Modify the forked version (under your own name). You can do this all with in browser. Just enter the editor from GitHub website.
- After you save (commit) the works, you can send a Pull Request (PR) to the current repo. Someone from our team will review the changes and merge/ close the PR.

## References and further reading

- [Jupyter Notebooks from S2018 students](#). You can check out some projects and their notebooks to get more familiar with Jupyter Notebook.
- [GitHub official guide](#). You can basically learn everything about GitHub in it's guide.
- GitHub and GitHub Desktop are just two graphical ways to operate the powerful version control system, "Git". Interested readers can further study the command line from this online resource: <https://try.github.io/>

---

If you have any questions, or seek for help troubleshooting, please [create an issue here](#)

# Week 01: Terminal, Python, Jupyter Notebook

- [Week 01: Terminal, Python, Jupyter Notebook](#week-01-terminal-python-jupyter-notebook) - [Objective of this week](#objective-of-this-week) - [About terminal on Mac](#about-terminal-on-mac) - [What is terminal on Mac? ](#what-is-terminal-on-mac) - [What is the function of the terminal on Mac? ](#what-is-the-function-of-the-terminal-on-mac) - [Why we need to use it? ](#why-we-need-to-use-it) - [How to open terminal on MAC? ](#how-to-open-terminal-on-mac) - [Shell commands](#shell-commands) - [Directory: Where you are](#directory-where-you-are) - [Create/delete/rename files and folders](#create-delete-rename-files-and-folders) - [Get inline help in the command line](#get-inline-help-in-the-command-line) - [The `man` command](#the-man-command) - [Basic usage](#basic-usage) - [Bonus: Command return and command output](#bonus-command-return-and-command-output) - [Edit and execute python file](#edit-and-execute-python-file) - [Text editor](#text-editor) - [Install python 3](#install-python-3) - [Modify the `ex1.py` file by a text editor](#modify-the-ex1py-file-by-a-text-editor) - [Execute .py file](#execute-py-file) - [Familiar with python interactive mode](#familiar-with-python-interactive-mode) - [Python interpreter](#python-interpreter) - [Invoking the Interpreter](#invoking-the-interpreter) - [Two basic modes: script and interactive](#two-basic-modes-script-and-interactive) - [Execute an existing script interactively](#execute-an-existing-script-interactively) - [Learn to use Jupyter Notebook](#learn-to-use-jupyter-notebook) - [Virtual environment](#virtual-environment) - [Setup virtualenv and install Jupyter Notebook](#setup-virtualenv-and-install-jupyter-notebook) - [Basic usage](#basic-usage-1) - [Exercises and Challenges](#exercises-and-challenges) - [References and Further Readings](#references-and-further-readings)

In this very first chapter, you will start a journey, swimming in the ocean of codes and data. During the following months, you may experience a staggering start, enjoyable progress or even deeply frustration. You have to step out your comfort zone, learning from each other and conquer the overwhelming information world with your persistence and intelligence. If you have the determination to accept this challenge, you will see a brand new yourself at the end of the course.

## Objective of this week

- Setup the Python environment. Please read [here](#).
- Learn what is terminal. Be able to navigate file system in Terminal using the shell.
- Create the first python script and execute it.
- Learn the interactive mode of Python interpreter -- convenient for rapid experimentation.
- Learn the Jupyter notebook -- our major working platform in following weeks.

## About terminal on Mac

### What is terminal on Mac?

Basically, Terminal is a shell which receives/sends input and output for command-line program.

### What is the function of the terminal on Mac?

The function of the terminal is very powerful, and all the basic operations of the system can be completed in terminal, such as modifying file permissions, hiding/displaying files, and so on.

### Why we need to use it?

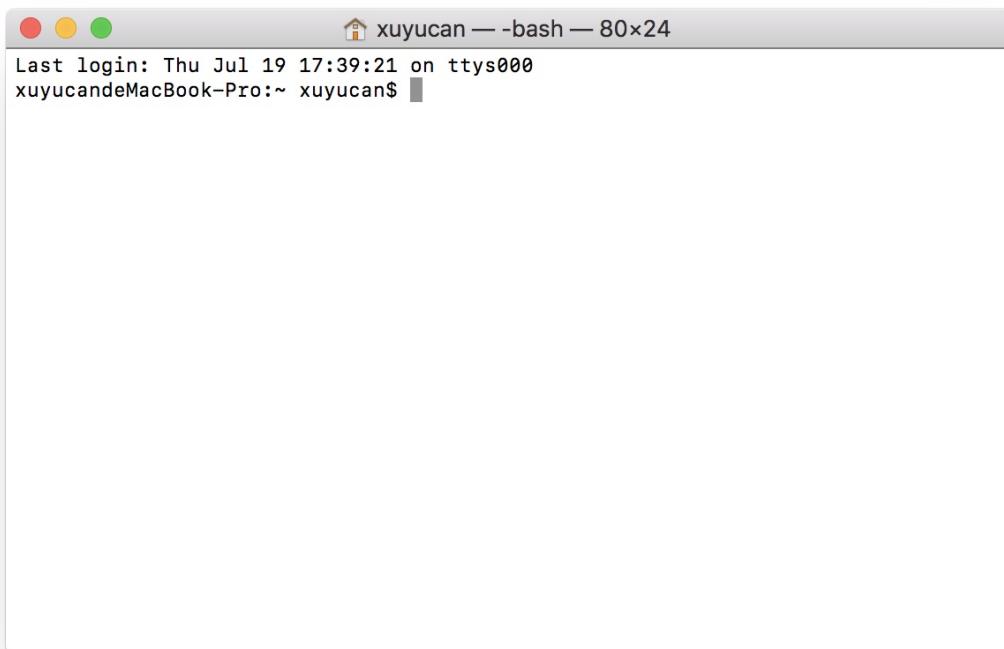
- Many features of computers are not available in the graphical interface, only through the command line.
- Work can be more efficient with command-line scripts.

## How to open terminal on MAC?

- press **command+space** to open spotlight



- search "terminal" to open terminal.



When you open terminal, you can see these two lines. The first line represents the last time you log in. And the second line **xuyucandeMacbook-pro** shows your computer model. **~xuyucan** is your account/username. What you need to focus is notation `$`. Its the sign that you can input some commands, and after you click the return, the computer will send back the results in next line.

## Shell commands

Following are some elementary commands you should know in terminal.

```
Last login: Fri Jul 20 14:26:02 on ttys000
[xuyucandeMacBook-Pro:~ xuyucan$ ls
Applications Library chromedriver
Calibre 书库 Movies pandas
Creative Cloud Files Music python-twitter
Desktop Pictures venv
Documents Public
Downloads PycharmProjects
[xuyucandeMacBook-Pro:~ xuyucan$ cd desktop
[xuyucandeMacBook-Pro:desktop xuyucan$ pwd
/Users/xuyucan/desktop
[xuyucandeMacBook-Pro:desktop xuyucan$ ls
Enjoy+ Gift Registration.pdf
Flying in the sky, a report of air crash – The Data & News Society.docx
python
python-twitter
ye.jpg
屏幕快照 2018-07-20 下午 2.37.46.png
[xuyucandeMacBook-Pro:desktop xuyucan$ cd python
[xuyucandeMacBook-Pro:python xuyucan$ ls
17426316.py hello.py scraibe 2.py
2_2.py homework.py scraibe.py
Case1_Advanced.py homework2.py sight.py
Case1_Fundamental.py imbd.py taiwan-comments.py
H1.py list.py taiwan_earthquake.csv
comments.csv list2.py taiwanearthquake.py
d.py movie.csv test.py
data.csv movie.numbers title.py
douban1.py nmb.py weibo.py
finaltest.py populartedtalks.csv y.py
first.py quiz.py
[xuyucandeMacBook-Pro:python xuyucan$ cd ..
xuyucandeMacBook-Pro:desktop xuyucan$]
```

## Directory: Where you are

Please type often `pwd` and `ls` to know where you are. And use `cd` to the change the location.

- `ls` means listing, showing the files in current folder. e.g.:

```
~ xuyucan$ ls
Applications Library chromedriver
Calibre 书库 Movies pandas
Creative Cloud Files Music python-twitter
Desktop Pictures venv
Documents Public
Downloads PycharmProjects
```

- `cd xx` means change directory to xx, or change to xx folder, and you can add the location after `cd`.
- `pwd` means print what directory, or show where you are e.g.:

```
~ xuyucan$ cd Desktop
desktop xuyucan$ pwd
/Users/xuyucan/Desktop
```

e.g.2: try to `cd` to a certain folder in your desktop. As for me, I have created a folder named python in my desktop within several .py files, and I `cd` to this folder and list the file in this folder.

```
desktop xuyucan$ cd python
python xuyucan$ ls
17426316.py hello.py scrabe 2.py
2_2.py homework.py scrabe.py
Case1_Advanced.py homework2.py sight.py
Case1_Fundamental.py imedb.py taiwan-comments.py
H1.py list.py taiwan_earthquake.csv
comments.csv list2.py taiwanearthquake.py
```

- Besides following `cd` with an explicit path, one can use those special notation as a shortcut to change-directory to some common locations:
  - `cd ~` or `cd` to return to home.
  - `cd .` to go to current directory.
  - `cd ..` to return back to the upper directory.

```
python xuyucan$ cd ..
desktop xuyucan$ cd ~
~ xuyucan$
```

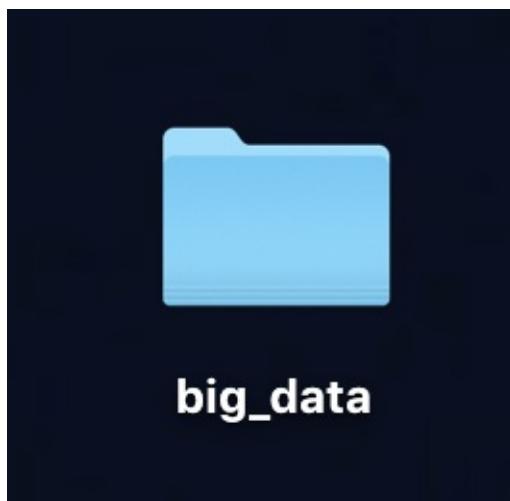
## Create/delete/rename files and folders

Space separates the arguments and commands. So be careful. You can `ls` to check the new file or folder. (*Make sure it exists.*)

- `touch` means to create a file
- `rm` means to delete a file
- `mkdir` means to create a folder
- `rmdir` means to delete a folder
- `mv` means to rename

e.g.: At first, cd to desktop and create a new folder `big_data`

```
~ xuyucan$ cd desktop
desktop xuyucan$ mkdir big_data
```



you can see the new folder in your desktop then cd to `big_data` folder, create a new python file `ex1.py`, rename it as `exercise.py`, delete this file, and delete the `big_data` directory. During the process, you can check out whether the file/folder changed.

```
desktop xuyucan$ cd big_data
big_data xuyucan$ touch ex1.py
big_data xuyucan$ mv ex1.py exercise.py
big_data xuyucan$ rm exercise.py
big_data xuyucan$ cd ..
desktop xuyucan$ rmdir big_data
```

## Get inline help in the command line

### The `man` command

`man` - format and display the on-line manual pages, its helpful to get help and explanation from the official manual. It can be used to **display manual pages**, **search for occurrences of specific text**, and other useful functions.

If you specify section, man only looks in that section of the manual. name is normally the name of the manual page, which is typically the name of a command, function, or file.

### Basic usage

Enter the manual page of any shell command:

```
man {command-name}
```

Frequent hotkeys in manual page window:

- Use `j` and `k` to move the page down/ up by one line.
- Use `ctrl+d` and `ctrl+u` to move down/ up by one-half screen.
- Use `q` to exit (return to shell prompt).

Following are several common use and examples:

- Display the usage of commands, like `ls`

```
$ man ls

LS(1) BSD General Commands Manual LS(1)

NAME
ls -- list directory contents

SYNOPSIS
ls [-ABCFGHLOPRSTUW@abcdefghijklmnopqrstuvwxyz1] [file ...]

DESCRIPTION
For each operand that names a file of a type other than directory, ls
displays its name as well as any requested, associated information. For
each operand that names a file of type directory, ls displays the names
of files contained within that directory, as well as any requested, asso-
ciated information.

If no operands are given, the contents of the current directory are dis-
played. If more than one operand is given, non-directory operands are
displayed first; directory and non-directory operands are sorted sepa-
rately and in lexicographical order.

The following options are available:

-@ Display extended attribute keys and sizes in long (-l) output.

-1 (The numeric digit ``one''). Force output to be one entry per
line. This is the default when output is not to a terminal.
```

...

- List all chapters and their file path: `-aw`

```
$ man -aw ls
/usr/share/man/man1/ls.1
$ man -aw printf
/usr/share/man/man1/printf.1
/usr/share/man/man3/printf.3
```

- Display the certain section of `printf`: `man + int + printf`

```
$ man 3 printf

PRINTF(3) BSD Library Functions Manual PRINTF(3)

NAME
 printf, fprintf, sprintf, snprintf, asprintf, dprintf, vprintf, vfprintf,
 vsprintf, vsnprintf, vasprintf, vdprintf -- formatted output conversion

LIBRARY
 Standard C Library (libc, -lc)

SYNOPSIS
 #include <stdio.h>

 int
 printf(const char * restrict format, ...);

 int
 fprintf(FILE * restrict stream, const char * restrict format, ...);
 ...
```

- Display all the section of `printf`: `-a`

```
man -a printf
```

- Search online manuals that contain the keyword: `-k`

```
man -k printf
```

- Search for files, whose name contain specified keywords: `-f`

```
man -k print
```

For more functions, you can type `man man` on terminal to see more.

## Bonus: Command return and command output

- Return value: it is a convention for UNIX-like system/ program to return a value upon completion of execution. The return value indicates whether the program executes as expected. Usually, the return value is `0`, meaning the execution is successful. If the return value is non-zero, it means an error occurred and you should go check the error code with the manual. One can use this command to check the return value of *the very last* command  
`echo $? .`
- Output: a command/ program can output information for the user. There are two output streams:
  - `stdout` -- "standard output" -- This is usually useful data for further processing, e.g. as input to next

command, for the users to comprehend. Later of this section, you will see the first Python coding using `print()`. This `print()` basically writes texts to `stdout`.

- o `stderr` -- "standard error" -- This includes error information that can help the user to debug. By default, when you operate in a MAC Terminal, `stderr` and `stdout` are written to the same stream, so you can not distinguish them by eyesight. Interested readers can check [this discussion](#) to see how to divert the two streams.

## Edit and execute python file

### Text editor

Terminal is an interactive environment. The advantage of writing code inside is that you can get the result instantly, but the weakness is that you can't save it. When you want to run it again, you have to tap it again. This is why we need a text editor. In actual programming development, we always use a text editor to write the code and save it as a file, so that the computer can run repeatedly. We recommend two text editors, [sublime](#) and [visual studio code](#)([click to download](#)). Then you can edit a .py file by these editors by double clicking the file. MAC will open "TextEdit" by default editor. You can set one of those two editors as default editor if necessary.

### Install python 3

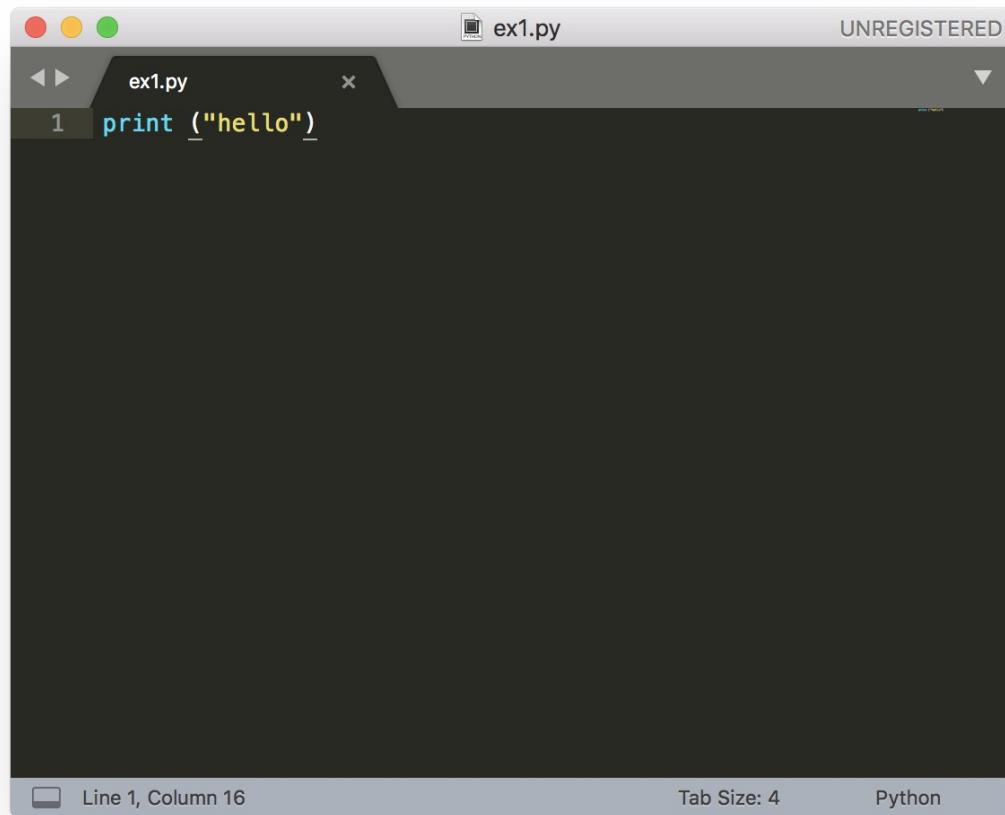
Python is a popular programming language that is widely used by beginners and longtime developers alike. Meanwhile, its the language that we learn in this course to scrape, clean, analyze, and visualize data. And there are basically 2 main versions of python. Python 2 and 3. You can check the version using following command on your terminal:

```
python --version
```

In this course, we base our discussions and exercises on Python 3, you can check out the [difference between python 2 and 3](#) and the instruction for [installation of python 3](#) in related materials in our gitbook (*if you have already set up the python 3, just ignore it*).

### Modify the `ex1.py` file by a text editor

`print` is a python language, which means print, or show the things that written in the files in the terminal. eg: `print ("hello")` on sublime to print the string hello.



A screenshot of a code editor window titled "ex1.py". The window has three tabs at the top: "ex1.py", "x", and "UNREGISTERED". The main editor area contains the following Python code:

```
print ("hello")
```

The status bar at the bottom shows "Line 1, Column 16" and "Tab Size: 4". The language mode is set to "Python".

Press **Command+s** to save the file as "ex1.py" on desktop.

## Execute .py file

python ex1.py on terminal to execute the file.

```
desktop xuyucan$ python ex1.py
hello
```

If the output is "2.x", you will need to try `python3`. For example, when you execute Python script, you need to type `python3 myscript.py` when our book uses `python myscript.py`.

## Familiar with python interactive mode

### Python interpreter

An interpreter is a program that reads and executes code. This includes source code, pre-compiled code, and scripts. Basically, the Python interpreter is the application that runs your python script.

By default, Python source files are treated as encoded in UTF-8. But the standard library only uses ASCII characters for identifiers, a convention that any portable code should follow. To display all these characters properly, python interpreter will recognize that the file is UTF-8, and support all the characters in the file.

What the interpreter does in a nutshell:

1. Read the script line by line and converts that script into python byte code.
2. The interpreter then executes the file instruction by instruction, it is at this stage errors are created if your code generates such errors.

## Invoking the Interpreter

Typing the command `python` or `python3` on your terminal. After that, you will see `>>>` notation which indicates you that you have already entered the interactive mode and the interpreter is waiting for your input. For instance:

```
$ python3
>>> hello
hello
>>> 1 + 2
3
>>> a = 0
```

Type `control + d`, or use `quit()` function to the interpreter.

## Two basic modes: script and interactive

1. The `script mode` is the normal mode where the scripted and finished `.py` files are run in the Python interpreter.
2. The `interactive mode` is a command line shell which gives immediate feedback for each statement.

Differences between two modes:

- A `.py` file can only be executed in script mode, using `python3 + filename.py` to run the file.
- In interactive mode, you can only enter one line and execute one line each time, while in script mode, you can execute all the code in the file at once by running the `.py` file directly.
- The interactive mode is primarily used to debug the code and testing.

## Execute an existing script interactively

Sometimes, you have an existing script, maybe from past works or from others. You want to execute this script first but stays in the Python interpreter after that. In this way, the state of the interpreter, e.g. all the variables, will be fully preserved for your further exploration. One can use the `-i` option. The command line pattern is as follows:

```
python -i myscript.py
```

## Learn to use Jupyter Notebook

Jupyter notebook is originally called "IPython notebook" (interactive Python notebook), thus having the `.ipynb` suffix/extended name of the the Jupyter notebook file.

It provides a web-based interface for you to interactively test and build Python codes. It is well suited for a bottom-up approach when building larger projects.

## Virtual environment

You will hear the term "environment" a lot of times when learning programming. It is a very broad term that refers to the context where the program is executed. The context can be time, operating system, current working folder, Python version, dependent module version, the status of system, the status of dependent components, ...

**TIP:** Two pieces of codes can act differently if the environments are different. When you find someone else's codes work but the same thing does not work at your side, it is a problem of "environment". Trouble shooting highly depends on experience and we will see a lot during the semester.

Python has a concept called virtual environment, "virtualenv" for short. You can use virtualenv to ensure the programs execute in the same environment. One common use case is to run Python2 and Python3 programs on the same computer. The system defaults to one of the major versions. However, you can use virtualenv to run some programs in Python2 and some programs in Python3. We also use virtualenv to ensure the dependent Python modules are the same, whose version is usually specified in `requirements.txt`.

There are two commands to setup virtualenv:

- `virtualenv` -- old executable usually used in Python2.
- `pyvenv` -- the default and recommended way of setting up virtualenv in Python3. The tools is shipped with Python3 installation.

## Setup virtualenv and install Jupyter Notebook

If it's the first time you use jupyter notebook, you need create a virtual environment first. The following are the usual path to setup jupyter environment. For users in CVA 517 LAB, please see [here](#).

Step 1: Create virtual environment

```
pyvenv venv
```

Step 2: Enter virtual environment

```
source venv/bin/activate
```

Step 3: Install Jupyter notebook

```
pip3 install jupyter
```

Step 4: Enter Jupyter notebook

```
jupyter notebook
```

For details, Please see to our [tutorial](#) of how to install and enter jupyter notebook. The following is what jupyter notebook will look like.

```

In [2]: # coding: utf-8

import requests
from bs4 import BeautifulSoup
import csv
r=requests.get('https://movie.douban.com/top250')
mypage= r.text

page_next=[]
url = 'https://movie.douban.com/top250'
for q in [-25,0,25,50,75,100,125,150,175,200,225]:
 page_number=str(q + 25)
 url_next=url+ "?start=" + page_number + '&filter='
 page_next.append(url_next)

cn_name = []
en_name = []
comment_number = []
rating = []

data = []

```

## Basic usage

1. click `new` to create a new python 3 notebook
2. write codes like you usually do in text editors, and press `shift + return` to run the code. It will return the results or errors under the cell.
3. use `! pip3 install module_name` to install modules in jupyter notebook.
4. in front of every cell, there is an `in [ ]` sign, the number in `[]` means the sequences of cells, and if there is `*` in `[]`, means that this cell is still running, you can either wait it finish or click `stop` under the `kernel` to exit from the running, pressing `i` twice will also do the trick.
5. cell. `run cell` run step by step. `run all above` to run and check the previous steps of coding.
6. kernel. `kernel` is a tool for interactive input and output all the things you did from the beginning. By clicking `restart`, you can give a variable another value.

## Exercises and Challenges

- Write a Python script to output "Good evening" in the Terminal.
- Use shell commands to re-organise the notes you take from this course. We understand that you can do this very quickly in GUI (e.g. in Finder of OS X). Please try the command line way and make it part of your daily life.

## References and Further Readings

- [Terminal and shell commands \(Chinese\)](#)
- [Appendix A of "Learn Python the hard way"](#) - Suggest all students make some self-study of this tutorial. Being comfortable in shell environment can make one efficient in programming.

If you have any questions, or seek for help troubleshooting, please [create an issue here](#)



## Week 02: Use Python as a daily tool

- [Week 02: Use Python as a daily tool](#week-02-use-python-as-a-daily-tool) - [Objective of this week](#objective-of-this-week) - [Variables and assignment](#variables-and-assignment) - [Basic data types](#basic-data-types) - [Integer: `int`](#integer-int) - [Floating point (real number): `float`](#floating-point-real-number-float) - [Boolean: `bool`](#boolean-bool) - [String: `str`](#string-str) - [Escape character in strings](#escape-character-in-strings) - [Arithmetic](#arithmetic) - [Basic rules](#basic-rules) - [Exercise 1: Simple calculation](#exercise-1-simple-calculation) - [Exercise 2: Calculate a mortgage](#exercise-2-calculate-a-mortgage) - [Modules, functions and packages](#modules-functions-and-packages) - [Functions](#functions) - [Modules](#modules) - [Packages](#packages) - [How to use modules](#how-to-use-modules) - [Step 1: pip install modules](#step-1-pip-install-modules) - [Step 2: import modules](#step-2-import-modules) - [How to find modules and packages we want](#how-to-find-modules-and-packages-we-want) - [How to call functions](#how-to-call-functions) - [`.` notation to reference to the member](#notation-to-reference-to-the-member) - [`()` notation to call function](#notation-to-call-function) - [Exercise 3: Calculate the area of a circle](#exercise-3-calculate-the-area-of-a-circle) - [Common modules and functions you should know in chapter 2](#common-modules-and-functions-you-should-know-in-chapter-2) - [Scipy & Numpy](#scipy--numpy) - [Basic functions: Arrays](#basic-functions-arrays) - [About index in data types](#about-index-in-data-types) - [String functions `(``str``)`)](#string-functions-str) - [Common functions](#common-functions) - [Python's classical percent-sign string interpolation](#pythons-classical-percent-sign-string-interpolation) - [Str.format()](#strformat) - [Random](#random) - [random.random()](#randomrandom) - [random.randint(a,b)](#randomrandintab) - [random.choice(seq)](#randomchoiceseq) - [random.choices(population, weights=None, \*, cum\_weights=None, k=1)](#randomchoicespopulation-weightsnone--cum\_weightsnone-k1) - [random.shuffle(list)](#randomshufflelist) - [random.sample(population, k)](#randomsamplepopulation-k) - [Exercises and Challenges](#exercises-and-challenges) - [Design and calculate a simple media business model](#design-and-calculate-a-simple-media-business-model) - [Calculate how many SD card you need to bring](#calculate-how-many-sd-card-you-need-to-bring) - [A simple model of media monitoring](#a-simple-model-of-media-monitoring) - [Mark Six lottery machine](#mark-six-lottery-machine) - [References](#references)

In the previous chapter, We introduced the basic knowledge about terminal on Mac and how to navigate file system in Terminal, using shell, creating the first python script and execute it... In this chapter, we want to focus on Python basics, including *variables*, *basic data types*, *arithmetic*, *functions* and several commonly used *modules*. After this chapter, you can use python as your daily tool, at least in form of a powerful calculator.

## Objective of this week

- Understand python basics
- Can use `help` to get inline documentation on new modules and functions
- Become comfortable with Python interpreter -- the REPL pattern (Read-Evaluate-Print Loop)
- Can use Python as a daily tool -- at least a powerful calculator

## Variables and assignment

Think of a variable as a name attached to a particular object. In Python, variables need not be declared or defined in advance. Therefore you should give it a definition or assign value to it. The equal sign `=` is used to assign values to variables.

Example 1:

```
>>> a = 1 + 2
>>> print(a)
```

"a" is variable, so you give it a definition that equals to 1+2, and print 'a', you will get 3 in terminal.

Example 2:

```
>>> fruit1 = 'apple'
>>> print(fruit1)
apple
```

It means you define variable "fruit1" as apple, so you will get apple in terminal.

**Note** You should use ` `` , or ` "" ` , to wrap `apple` , or the terminal will think apple is another variables. You can't assign variables to other variables.

Example 3:

```
>>> a = 'hello'
>>> b = 'world'
>>> c = a +' '+ b #the blank space in `` will be shown on the results
>>> print(c)
hello world
```

a, b, c are all variables, and you can do calculation like normal numbers.

## Basic data types

In fact, there are so many different kinds of data you can use to define the variables. Different data types are totally different. Following are some basic data types.

### Integer: `int`

`int` means integer, like 7, 8, 9. We deal with integers in our work for most of the time.

Example 4:

```
>>> a = int(7.9)
>>> print(a)
7
```

`int()` is a function, which converts the number in `()` to an integer.

### Floating point (real number): `float`

`float` means a number with decimal, like 2.1, 3.8, 5.6 and so on. It was initially called "float" because of the way computer handles very big/ very small real numbers is by [Floating point arithmetics](#).

Example 5:

```
>>> b = float(7)
>>> print(b)
7.0
```

`float()` means you converts the number/integer in `()` to a decimal.

You will find most of our daily arithmetic operations apply to `int` and `float` , like `+` , `-` , `*` , `/` . We will show you basic arithmetics after explaining basic data types.

## Boolean: `bool`

The `bool()` method converts a value to Boolean (True or False), using the standard truth testing procedure.

It's not mandatory to pass a value to `bool()`. If you do not pass a value, `bool()` returns False. In general use, `bool()` takes a single parameter value.

The following values are considered `false` in Python:

- None
- False
- Zero of any numeric type. For example, 0, 0.0, 0j
- Empty sequence. For example, (), [], "".
- Empty mapping. For example, {}
- Objects of Classes which has `bool()` or `len()` method which returns 0 or False

All other values except these values are considered `true`.

Example 6:

```
>>> test = []
>>> print(test, 'is', bool(test))
[] is False
>>> test = [0]
>>> print(test, 'is', bool(test))
[0] is True
>>> test = 0.0
>>> print(test, 'is', bool(test))
0.0 is False
>>> test = None
>>> print(test, 'is', bool(test))
None is False
>>> test = True
>>> print(test, 'is', bool(test))
True is True
>>> test = 'Loving you'
>>> print(test, 'is', bool(test))
'Loving you' is True
```

You can apply **boolean logic** on bool type of values using operators like `and`, `or` and `not`. We will explain more in [Chapter 3](#) when we discuss program flow management via conditional branches and loops.

## String: `str`

- `str` means string, a sequence of characters, like quiet, asdf, HK\_NY and so on.
- Strings can be created by enclosing characters inside a single quote or double quotes, like `' '`, `" "`. Even triple quotes can be used in Python but generally used to represent multi-line strings and doc-strings.

**Note:** what if there is a `" "` in your line, how do you print this string line?

Example 7:

Please print Xiao Ming says " I don't feel well today. ".

```
>>> print("Xiao Ming says \"I don't feel well today\"")
Xiao Ming says "I don't feel well today"
```

What does `\\"` means?, let's talk about this more.

## Escape character in strings

An **escape character** is "a character which invokes an alternative interpretation on subsequent characters in a character sequence".

Python has its own rules and grammar. Like if you need to use special character `''` in a string, you should use escape character. Because python regards `''` as sign of a string, if you use `''` inside of the string, it will cause invalid syntax error. Usually, we uses the \ (backslash) as an escape character for. The following are the commonly used examples.

- `\'` means single quote
- `\\"` means double quote
- `\\\` means backslash
- `\n` means new line
- `\r` means carriage return
- `\t` means tab
- `\b` means backspace
- `\f` means form feed
- `\v` means vertical tab

Example 8:

```
>>> print("I don't feel well \ntoday")
I don't feel well
today
```

## Arithmetic

### Basic rules

Operator	Description	Example
<code>+</code> Addition	Adds values on either side of the operator.	<code>a + b = 30</code>
<code>-</code> Subtraction	Subtracts right hand operand from left hand operand.	<code>a - b = -10</code>
<code>*</code> Multiplication	Multiplies values on either side of the operator	<code>a * b = 200</code>
<code>/</code> Division	Divides left hand operand by right hand operand	<code>b / a = 2</code>
<code>%</code> Modulus	Divides left hand operand by right hand operand and returns remainder	<code>b % a = 0</code>
<code>**</code> Exponent	Performs exponential (power) calculation on operators	<code>a**3 = 1000, a=10</code>
<code>//</code> Floor Division	The division of operands where the result is the quotient in which the digits after the decimal point are removed.	<code>9//2 = 4, 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0</code>

### Exercise 1: Simple calculation

```
>>> a = 10//3
>>> b = 10%3
>>> c = 10**3
>>> print("a=",a, "b=",b, "c=",c)
a= 3 b= 1 c= 1000
```

## Exercise 2: Calculate a mortgage

Question: Calculate mortgage based on the following formula: (Assign "r" "P" "n" specific numbers by yourself.)

$$A = P \cdot \frac{r(1+r)^n}{(1+r)^n - 1}$$

*formula from wikipedia*, you can check out what each variable represents.

```
>>> r = 0.05
>>> n = 20
>>> P = 5000000
>>> A = r*P*(1+r)**n/((1+r)**n-1)
>>> print("A=",A)
A= 401212.935953
```

## Modules, functions and packages

### Functions

Function is a fundamental building block in Python that includes certain codes that can be re-used. A function takes input in form of "arguments" and gives corresponding output. For example, `print()` is a built-in function, which takes input of a sequence of arguments, representing values to be print. `print()` consumes those arguments, turn them into `str` format, and write the output onto your screen. `int()` is another built-in function, which takes an arbitrary value as input and output a corresponding integer.

Note the "function call" notation, i.e. `()` following the name of the function. When Python interpreter sees this notation, it will enter the body of the function and execute the codes there. When you find yourself doing something repeated in Python, you should consider to wrap those codes into a function, and use function calls with proper parameters at the place you want to use it. We will discuss this more on how to write and how to call one's own function in [Chapter 3](#).

### Modules

Module is a higher level building block in Python that includes certain *function* and can be reused. Some examples are like [numpy](#), [scipy](#) and [geopy](#). Those modules extend basic Python functions, so that you can easily finish complex tasks, like compute the matrix multiplication and get distance between two cities. Otherwise, you need to write hundreds of lines of codes, in order to get it done.

### Packages

A package is a collection of Python modules. It can be single `.py` file or a folder structure of `.py` files. Without worrying about the internal layouts, the way of using package is similar of using module. We will use the two terms interchangeably future discussions.

## How to use modules

### Step 1: pip install modules

There're some preparations you need to do before you import or use modules. You need to install firstly. You can either install from the official website of the package or use other third party tools, like `pip`, which we recommend.

Pip is a function of the Python Packaging Authority ([PyPA](#)), which is a working group that maintains many of the relevant projects in Python packaging. `pip` is already installed if you are using Python 2  $\geq 2.7.9$  or Python 3  $\geq 3.4$ . You can type `python --version` to check out current version. If you are the older versions, please check out [here](#) to install and upgrade `pip`.

**Note** If you use python 3, please use `pip3` to install.

- To install the latest version of "SomeProject": type `pip3 install 'SomeProject'` in your terminal.
- To install a specific version: `pip3 install 'SomeProject==1.4'`
- Upgrading packages: `pip3 install --upgrade SomeProject`
- To install packages that are isolated to the current user: `pip3 install --user SomeProject`. In school computer lab, you should use this method because you don't have authority to install in whole computer but just your account, so that the computer will keep your record.

### Step 2: import modules

In python, we use `import` statements to call a certain modules or functions. `import` means import a module from the library/package you download or install from the internet. You can import modules by writing `import + module name` on terminal. And there are many modules out there, commonly used modules like [numpy](#) *click to check out more details*.

There are 3 ways to import a module, usually we use the first method, but you will learn the last two method in the later stage.

- `import` module
- `from module.xx.xx import xx`
- `from module.xx.xx import xx as rename`

For example, `import numpy` is to import a module called numpy. (Generally, "import" is always written on the top to indicate readers what kind of module and functions you will use.) After you import the module, you can use `help(module)` to check out their documentations

Example 10:

```
>>> import numpy
>>> help(numpy)
Help on package numpy:

NAME
 numpy

DESCRIPTION
 NumPy
 =====

 Provides
 1. An array object of arbitrary homogeneous items
```

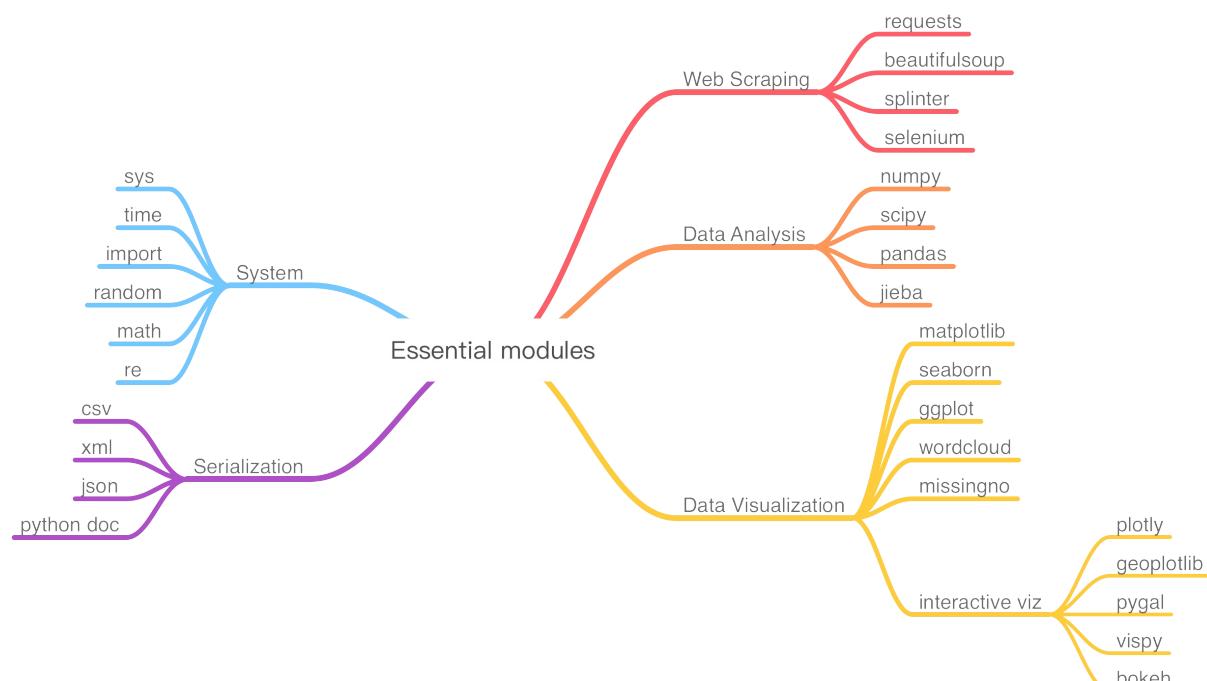
- 2. Fast mathematical operations over arrays
- 3. Linear Algebra, Fourier Transforms, Random Number Generation

How to use the documentation

...

After you learn what you want in their documentations, you can press `control + z` to exit.

## How to find modules and packages we want



Basically, those are the modules that we might use in our daily study. You can get to know more modules by googling your demand or create a new `issue` in [here](#) to ask for help.

## How to call functions

### . notation to reference to the member

A member of a module can be a function or a variable. As you already know from the previous content that there are many functions in a module. We use `module name . member name` to reference to one member in a module. If you don't know what are the members in this module, you can use `help(module)` to load help documentation, or use `dir(module)` to checkout the available members.

Example 11: Get the constant Pi

```
>>> import numpy
>>> print(numpy.pi)
3.14159265359
```

This example means you choose a function called "pi" from the module `numpy`. You will get pi as 3.1415927.

Therefore, whenever you want to call a function, you should check out which modules contain this function, then you use `.` notation to reference to the function.

### ( ) notation to call function

After we reference to one function, usually we have to input the parameters to call the function. we use ('parameters') after the function to call the function, after executing this command, you can get the results you want.

Example 12: Use `sin` and `pi` function from `numpy` module

```
>>> import numpy
>>> print(numpy.sin(numpy.pi/6))
0.5
```

```
import numpy
print(numpy.sin(numpy.pi/6))
 ↓ ↓ ↓
 module function parameter
```

### Exercise 3: Calculate the area of a circle

Q: Calculate the area of a circle, and assign specific radius by yourself.

Example:

```
>>> import numpy
>>> r = 5
>>> area = r**2 * numpy.pi
>>> print("area=", area)
area= 78.5398163397
```

## Common modules and functions you should know in chapter 2

### Scipy & Numpy

[SciPy](#) (pronounced “Sigh Pie”) is a Python-based ecosystem of open-source software for mathematics, science, and engineering. Which contains some most wildly used packages including `NumPy`, `Matplotlib`, `pandas`. [NumPy](#) is the fundamental package for scientific computing with Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

### Basic functions: Arrays

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of non-negative integers. The number of dimensions is the `rank` of the array; the `shape` of an array is a tuple of integers giving the size of the array along each dimension.

We can initialize numpy arrays from nested Python lists, and access elements using square brackets [] :

```
>>> import numpy as np
>>> a = np.array([1, 2, 3]) # Create a rank 1 array
>>> print(type(a))
numpy.ndarray
>>> print(a.shape) # how many elements
(3,)
>>> print(a[0], a[1], a[2]) # index elements
```

```

1 2 3
>>> a[0] = 5 # Change an element of the array
>>> print(a)
[5, 2, 3]
>>> b = np.array([[1,2,3],[4,5,6]]) # Create a rank 2 array
>>> print(b.shape)
(2, 3) #the first number means how many elements in this array, in this case, this array has two elements - two lists. The second number means how many sub-elements in each elements. In this case, there are 3 values in each list. So, it's like a layer nesting.
>>> print(b[0, 0], b[0, 1], b[1, 0])
1 2 4 #you can access values by their index. Similarly, the first number is to index elements in this array, the second number is to index the sub-elements in each elements.

```

**Note:** The text following by `#` is not the code, we call it comment. We can use `#` in ahead of text in the code blocks to comment out those text, which gives instruction to the system that this line is no need to execute. It's the explanations or instructions of the codes that tell others what you are doing so that they can quickly understand. Or even when you look back the codes later, you can quickly recall what you were doing. It's a good habit especially when you collaborate with others.

You can directly add `#` ahead of the codes/text to comment out them. If you want to comment out multiple lines. You can select those lines and type `command+/`. And type `command+/-` again to un-comment them.

## About index in data types

Simply, `Index` is like the position of one element in the whole list or object, which used to access this element. The index of first element is `0` and increases by integer. The following are examples may help you better understand this.

```

String 'H e l l o'
Index [0] [1] [2] [3] [4]
ch_list ['a', 'b', 'c', 'd', 'e']
Index [0] [1] [2] [3] [4]

```

In `ch_list ['a', 'b', 'c', 'd', 'e']`, `ch_list[0]='a'`, 0 is the index of 'a' in this list, similarly use in array. We will use more in chapter 3 so that you can get more familiar with this issue.

In addition, you can check out more array functions in scipy's [tutorial](#).

## String functions ( `str.*` )

### Common functions

Python has a built-in string class named "str" with many handy features, which allow us to easily make modifications to strings in Python. Here are some of the most common string methods:

Function	Description
<code>str.lower()</code>	Returns the lowercase or uppercase version of the string
<code>str.strip()</code>	Returns a string with whitespace removed from the start and end
<code>str.find('other')</code>	Searches for the given other string within s, and returns the first index where it begins or -1 if not found
<code>str.replace('old', 'new')</code>	Returns a string where all occurrences of 'old' have been replaced by 'new'
<code>str.split('delimiter')</code>	Returns a list of substrings separated by the given delimiter. <code>'a,b,c'.split(',')</code> -> <code>['a', 'b', 'c']</code> . As a convenient special case, <code>str.split</code> (with no arguments) splits on all whitespace chars. <code>'a b c'.split()</code> -> <code>['a', 'b', 'c']</code>

str.join(list)	Joins the elements in the given list together using the string as the delimiter. The list elements will be joined by sequences. e.g. '---'.join(['a', 'b', 'c']) -> a---b---c
----------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Example 13:

```
>>> test1 = 'PYTHON'
>>> test1.lower()
'python'
>>> test2 = '\n python is fun \n'
>>> test2.strip()
'python is fun'
>>> test3 = 'python loves,\\'you\''
>>> test3.find('you')
14 #returns the first character where 'you' begins
>>> test4 = 'python loves,\\'you\''
>>> test4.replace('you', 'me')
"python loves,'me'"
>>> test5 = 'python loves you'
>>> test5.split()
['python', 'loves', 'you']
>>> test6 = 'python loves you, do you like it'
>>> test6.split(',')
['python loves you', ' do you like it']
```

## Python's classical percent-sign string interpolation

Before `str.format()` was introduced, using `%` to do **simple string interpolation** can be very easily. Use it if the order of your arguments is not likely to change and you only have very few elements you want to concatenate.

Example 14:

```
>>> print("%s %s" % ('Hello', 'World'))
Hello World
```

In above example, `s%` means its a string. We used two `%s` string format specifier to tell Python where to substitute the value, and using 'Hello' and 'World' to replace those two strings. `%` between two parts basically means a command to call the replace actions.

If we want to make multiple substitutions in a single string, and as the `%` operator only takes one argument, we need to wrap the right-hand side in a tuple as shown in the example below.

Example 15:

```
>>> name = 'world'
>>> program = 'python'
>>> print('Hello %s! This is %.%(name,program)')
Hello world! This is python.
```

Also, if we want to replace the integer, use `d%`.

```
>>> print 'one is %d'%1
one is 1
```

## Str.format()

`str.format()` perform as a string formatting operation, which provides the ability to do complex variable substitutions and value formatting, and great flexibility over the output of the string in a way that is easier to read, write and maintain than just using plain old concatenation.

### Syntax and common functions:

- Accessing arguments by position.

```
>>> '{0}, {1}, {2}'.format('a', 'b', 'c')
'a, b, c'
>>> '{}, {}, {}'.format('a', 'b', 'c') # this method works only with the same number of {} and parameters
>>> '{2}, {1}, {0}'.format('a', 'b', 'c')
'c, b, a'
>>> '{2}, {1}, {0}'.format(*'abc') # unpacking argument sequence
'c, b, a'
>>> '{0}{1}{0}'.format('abra', 'cad') # arguments' indices can be repeated
'abracadabra'
```

Note: `.format(*'abc')` is equivalent to `.format(*['a', 'b', 'c'])`, and further equivalent to `.format('a', 'b', 'c')`. This is called argument unpack. You can revisit this example after learning the next chapter about compound structures.

- Accessing arguments by name.

```
>>> '{name},{age}'.format(age=18, name='xyc')
'xyc,18'
```

- Accessing arguments' attributes.

```
>>> person = {"name": "xyc", "age": "18"}
>>> print("person_name: {name}, person_age: {age}".format(**person))
person_name: xyc, person_age: 18
```

- Accessing arguments' items by locations.

```
>>> coord = (3, 5)
>>> 'X: {0[0]}; Y: {0[1]}'.format(coord)
'X: 3; Y: 5'
```

- Replacing %s and %r. It's like % way, please see to Examples 14.

```
>>> "Hello: {!r}; This is: {!s}".format('World', 'Python')
"Hello: 'World'; This is: Python"
```

- Named placeholders

`.format()` also accepts keyword arguments. One can replaces string with arguments passed in function.

```
>>> data = {'first': 'Hello', 'last': 'World!'}
>>> '{first} {last}'.format(first='Hello', last='World!')
'Hello World!'
```

- Aligning the text and specifying a width

```
>>> '{:<30}'.format('left aligned')
'left aligned'
>>> '{:>30}'.format('right aligned')
' right aligned'
>>> '{:^30}'.format('centered')
' centered '
>>> '{:^^30}'.format('centered') # use '*' as a fill char
'*****centered*****'
```

- Replacing %+f, %-f, and % f and specifying a sign

```
>>> '{:+f}; {:+f}'.format(3.14, -3.14) # show it always
'+3.140000; -3.140000'
>>> '{: f}; {: f}'.format(3.14, -3.14) # show a space for positive numbers
' 3.140000; -3.140000'
>>> '{:-f}; {:-f}'.format(3.14, -3.14) # show only the minus -- same as '{:f}; {:f}'
'3.140000; -3.140000'
```

- Truncating long strings to a specific number of characters.

```
>>> '{:.4}'.format('telephone')
'tele'
```

- Format numbers

- 'd' - Decimal Integer. Outputs the number in base 10, and number ahead of `d` means the specific width you want to keep.

```
>>> '{:d}'.format(42)
'42'
>>> '{:4d}'.format(42)
' 42'
```

- 'f' - Fixed-point notation. Displays the number as a fixed-point number. The default precision is 6, and number ahead of `f` means how many decimals you want to keep.

```
'''python
| | |
':{:06.2f}'.format(3.141592653589793) '003.14' '{:06.4f}'.format(3.141592653589793)
'3.1416'
```

For more `string` operations, you can check out in [python docs](#).

## Random

This module implements pseudo-random number generators for various distributions. There are many useful and simple functions, like `random.shuffle()`, `random.random()`, `random.sample()`. You can check out their [documentation](#) to learn the details.

### `random.random()`

Return the next random floating point number in the range [0.0, 1.0).

```
>>> import random
>>> random.random()
0.9897798657708502 #random floating number in [0.0,1.0]
```

### `random.randint(a,b)`

Return a random integer N and a <= N <= b. Alias for `randrange(a, b+1)`.

```
>>> import random
>>> random.randint(2,7)
5 #one random number >=2 and <=7
```

## random.choice(seq)

Return a random element from the non-empty sequence seq. If seq is empty, raises IndexError.

```
>>> import random
>>> number_list=[1,2,3,4,5,6,7,8,9,10]
>>> random.choice(number_list)
6 #one random number of the list
```

## random.choices(population, weights=None, \*, cum\_weights=None, k=1)

Return a random sub-seq from the non-empty sequence seq.

```
>>> import random
>>> number_list=[1,2,3,4,5,6,7,8,9,10]
>>> random.choices(number_list,k=3)
[5, 6, 8] #return k random number of the list
```

## random.shuffle(list)

Return a list that randomly sort the sequence of all elements.

```
>>> import random
>>> number_list=[1,2,3,4,5,6,7,8,9,10]
>>> random.shuffle(number_list)
>>> print(number_list)
[7, 1, 2, 8, 6, 3, 4, 10, 9, 5] #randomly sort
```

## random.sample(population, k)

Return a k length list of unique elements chosen from the population sequence or set. Used for random sampling without replacement.

```
>>> import random
>>> number_list=[1,2,3,4,5,6,7,8,9,10]
>>> random.sample(number_list,4)
[5, 10, 6, 1] #random sample 4 elements
```

# Exercises and Challenges

## Design and calculate a simple media business model

A group of HKBU students decided to found up a news website. Basically, their business model is to provide the content to their consumers, and earn money by a combination of subscription fee and advertising fee.

The following are the cost and revenue component in their business plan: (per month)

- Major cost:
  - Content cost 70000 dollars
  - Other cost like labor cost is 30000 dollars
  - Server cost: 50000
- Revenue:
  - 10% of website monthly visitors are expected to become the subscribers
  - Subscription fee is 15 dollar/subscriber a month

- o Ad revenue = `0.8 dollar/visitor` a month

Please build a calculator to estimate their revenues, given the number of monthly visitors as an variable `visitor`. Show the net income when `visitor` equals `40000`, `60000` and `80000`, respectively.

Functions you need to use:

`input()` function. Please refer to [here](#) for it's documentation.

Further challenges:

- Explore how the net income changes when the number of visitors changes
- Explore how the parameters of content cost, per user subscription fee and per user Ad revenue influences net income.
- (optional) Assume a more practical situation. The company invest `50000` in IT infrastructure as fixed assets. The infrastructure can support daily office use and support up to `50000` monthly visitors. When the number of monthly visitors becomes larger than `50000`, the company need to switch to cloud services, in order by computing powers on demand. The excess amount incurs a cost of `0.001` dollar per user.

**SPOILER:** The reference solution and variations can be found in [Calculate Marketing Objective for Your Media Startup](#), a blog post from our past students. Note that it uses more complex logics like `if` and `for`, which will be detailed in [Chapter 3](#).

## Calculate how many SD card you need to bring

Suppose you are working on a documentary. You know how many days of footage you are going to shoot. Since there is no Internet connection in the destination, you have to store the footages first. Please calculate how many SD cards you want to bring with you.

## A simple model of media monitoring

You are assigned the task of media monitoring. Your supervisor is interested in whether certain keywords appear in one article. In this exercise, we make a simple monitoring service. Suppose you already downloaded [this article](#) somehow (see [notes-week-07.md](#) and [notes-week-08.md](#)). The content is stored in variable like this: (called "heredoc")

```
article = """
The chief executive of Chinese online retailer JD.com, Liu Qiangdong, was briefly arrested in the US on accusations of criminal sexual conduct.

Mr Liu, one of China's richest people, was arrested in Minneapolis shortly before midnight on Friday and released on Saturday afternoon.

JD.com said Mr Liu, also known as Richard Liu, was falsely accused. Police say the investigation is open.

JD.com, also known as Jingdong, has alliances with Tencent and Walmart.

..."""
```

Tasks:

- Try to use string function to identify if certain keywords appears in the above article
- What if you want to monitor other articles?

## Mark Six lottery machine

Mark Six (六合彩) is a famous game in Hong Kong. In this quiz, please write a lottery machine to simulate the Mark Six game. Requirements:

- Create a repo called "mark-six", which includes a script called "lucky.py"
- When the user executes "python3 lucky.py", it outputs the lottery result.
- Create a "README.md" file to give an introduction of this project. You can include background information, program design rationale, pointers to your references, and sample output (from multiple executions).

TIP: the module `random` helps here. You can revisit this exercise in week-03 to make improvements.

## References

- [Chapter 1, 2, 3 of official Python 2 tutorial](#), About introduction to Python.
  - [Python doc modules](#) About modules, functions and packages.
  - [Python format string](#). They introduce some most common use-cases covered by the old and new style string formatting API with practical examples.
- 

If you have any questions, or seek for help troubleshooting, please [create an issue here](#)

# Week 03: Control Flow

- [Week 03: Control Flow](#week-03-control-flow) - [Objective](#objective) - [Use "Help" more to learn by yourself](#use-help-more-to-learn-by-yourself) - [Bool and comparisons](#bool-and-comparisons) - [Logic operators](#logic-operators) - [Comparison operators](#comparison-operators) - [Str comparison](#str-comparison) - [Int comparison](#int-comparison) - [Control flows](#control-flows) - [Execute code selectively: If-else Statement](#execute-code-selectively-if-else-statement) - [Repeat similar operations: loop](#repeat-similar-operations-loop) - [For loop](#for-loop) - [Use for statement to pick up values](#use-for-statement-to-pick-up-values) - [Use for loop to calculate](#use-for-loop-to-calculate) - [While loop](#while-loop) - [Difference between for loops and while loops](#difference-between-for-loops-and-while-loops) - [Break and Continue statement](#break-and-continue-statement) - [Break statement](#break-statement) - [Continue statement](#continue-statement) - [Integrated example: for and if](#integrated-example-for-and-if) - [Bonus: alternatives and more efficient calculation](#bonus-alternatives-and-more-efficient-calculation) - [Function](#function) - [Function definition (def)](#function-definition-def) - [Bonus: Scope of variables in function](#bonus-scope-of-variables-in-function) - [Errors and Exceptions](#errors-and-exceptions) - [Try and except](#try-and-except) - [Raising errors](#raising-errors) - [Exception error types](#exception-error-types) - [Common coding patterns](#common-coding-patterns) - [Representing a dataset](#representing-a-dataset) - [Handle repeated works](#handle-repeated-works) - [Infinite loop (daemon)](#infinite-loop-daemon) - [REPL](#repl) - [if..else; OR try..except](#ifelse-or-tryexcept) - [Multiple loop](#multiple-loop) - [Try sub-tasks and regroup success/ fail cases](#try-sub-tasks-and-regroup-success-fail-cases) - [Test and batch execution for repeated tasks](#test-and-batch-execution-for-repeated-tasks) - [Exercises and Challenges](#exercises-and-challenges) - [Generate detailed mortgage schedule](#generate-detailed-mortgage-schedule) - [Hint on table formatting](#hint-on-table-formatting) - [Hint on math formula](#hint-on-math-formula) - [Automatic writer for financial report](#automatic-writer-for-financial-report) - [Calculate credit and expense instalment](#calculate-credit-and-expense-instalment) - [References](#references)

Previously, We learn Python basics including data types, arithmetic, functions and several commonly used modules, with which help, you can build a calculator to do some simple case analysis. This week, we will step further to learn composite data types, the function and method of how to use them. Meanwhile, we will touch hands on the basic control flows to better understand the logic behind the coding works. After that, you can create some functions by your own so that you can use Python to do more.

## Objective

- Master the composite data type [] and {} in Python
- Master the control logics in Python, especially if and for
- Further understand the different roles of text editor and interpreter. Be comfortable writing batch codes in .py file and execute in Shell environment.
- Bonus: Understand Python engineering

## Use "Help" more to learn by yourself

In the Terminal of your computer, use `help` for any instruction for using Python functions.

### Note

- Type 'q' to quit help;
- Type 'j' to scroll down;
- Type 'k' to scroll up.

Example 1: If you want to know how to use 'numpy'. Type `help(numpy)` to learn more, you can get the information as follow:

```

>>> import(numpy)
>>> help(numpy)
Help on package numpy:

NAME
 numpy

FILE
 /Library/Python/2.7/site-packages/numpy-override(numpy/__init__.py

DESCRIPTION
 NumPy
 =====

 Provides
 1. An array object of arbitrary homogeneous items
 2. Fast mathematical operations over arrays
 3. Linear Algebra, Fourier Transforms, Random Number Generation

 How to use the documentation
 ...

```

## Bool and comparisons

### Logic operators

The logical operators in Python (`and`, `or`, `not`) are often used in the `if`, `if...else`, and `if...elif` statements. They enable you to make multiple comparisons inside a single statement, such as to determine whether a value is within a certain range.

Operators	What it means	What it looks like
and	True if both are true	x and y
or	True if at least one is true	x or y
not	True only if false	not x

Example 2:

```

>>> print((6 > 5) and (2 < 4)) # Its true when both expressions are True
True
>>> print((8 == 8) or (6 != 6)) # Its true when one expression is True
True
>>> print(not(3 <= 1)) # Its true when the original expression is False
True

```

### Comparison operators

In programming, comparison operators are used to compare values and evaluate down to a single Boolean value of either True or False. The following are the common comparison operators:

Operators	Meaning
<code>==</code>	Equal to
<code>!=</code>	Not equal to
<code>&lt;</code>	Less than
<code>&gt;</code>	Greater than

<code>&lt;=</code>	Less than or equal to
<code>&gt;=</code>	Greater than or equal to

## Str comparison

Strings can also be used with Boolean operators. They are case-sensitive. And you can use `str.()` functions to convert to upper- or lower-case letters.

Example 3:

```
>>> Name1 = 'YUCAN'
>>> Name2 = 'yucan'
>>> Name3 = Name2.upper()
>>> print("Name1 == Name2: ", Name1 == Name2)
('Name1 == Name2: ', False)
>>> print("Name1 == Name3: ", Name1 == Name3)
('Name1 == Name3: ', True)
```

## Int comparison

Example 4:

```
>>> x = 4
>>> y = 6

>>> print("x == y:", x == y)
x == y: False
>>> print("x != y:", x != y)
x != y: True
>>> print("x < y:", x < y)
x < y: True
>>> print("x > y:", x > y)
x > y: False
>>> print("x <= y:", x <= y)
x <= y: True
>>> print("x >= y:", x >= y)
x >= y: False
```

## Control flows

A control flow is a block of programming that analyses variables and chooses a direction in which to go based on given parameters. In python, all codes and statements are faithfully executed in exact top-down order. But what if you want to change the flow?

For example, you want the program to take some decisions and do different things depending on different situations, such as printing 'True' or 'False' depending on the different comparison and test?

Under such circumstances, using control flow statements will help you manipulate data better. There are several control flow statements we will learn in this chapter.

**NOTE:** In `control flows` chapter, we will encounter a lot of `indentations` when handling `if/for/while/def/class`. It is hard and inconvenient to type in Python shell, so please write down the codes in text editor and save it as a `.py` file, so that you can just execute the file once to get the answer. If you forget how to do this, please refer to [chapter 2](#).

## Execute code selectively: If-else Statement

`if...else` statement is used to conditionally execute a statement or a block of statements. Conditions can be true or false, execute one thing when the condition is true, something else when the condition is false.

```
if ...: #close with an ':'
 print(sth) #indented
elif ...: # elif = else if
 print(sth)
else:
 print(sth)
```

**Note:** All function definitions or condition comparisons should end with a `:`, and all the content in those functions and conditions need to be indented. you can just indent with clicking `tab`.

Take the case that we talk about chapter 2 as an example. (The full version of the case is [here](#))

Example 14: We want to know how much is the cost with the number of users we have. When the number is less than 50,000, the cost will be 10,000. If the number is not less than 50,000, then  $\text{cost} = 10000 + 0.1 \times (\text{number\_of\_users} - 50000)$ . The actual number of users we have now is 100,000. The if-else statement will be as fellow:

```
number_of_users = 100000
if number_of_users <= 50000:
 cost = 10000
else: # number_of_users > 50000
 cost = 10000 + 0.1 * (number_of_users - 50000)
print(cost)
```

Output:

```
15000.0
```

Example 15: If there is two charge plans when the number of users is more than 50,000.

1. when  $50,000 \leq \text{the number of user} \leq 100,000$ , the  $\text{cost} = 10000 + 0.1 \times (\text{number\_of\_users} - 50,000)$ ;
2. when the  $\text{number of user} \geq 100,000$ , the  $\text{cost} = 10,000 + 0.1 \times (100,000 - 50,000) + 0.2 \times (\text{number\_of\_users} - 100,000)$ .
3. The actual number of users we have now is 120,000.

The if-else statement will be as follows:

```
number_of_users = 120000
if number_of_users <= 50000:
 cost = 10000
elif number_of_users <= 100000: # 500000 <= number_of_users <= 100000
 cost = 10000 + 0.1 * (number_of_users - 50000)
else: # number_of_users > 100000
 cost = 10000 + 0.1 * (number_of_users - 50000) + 0.2 * (number_of_users - 100000)
print(cost)
```

Output:

```
21000.0
```

## Repeat similar operations: loop

### For loop

For loop(For Statement) has the ability to iterate over the items of any sequence, such as a list or a string.

### Syntax

```
for x in y: #close with an ':'
 print(sth) #indented
 if ...: # you can insert `if` in `for` loop
 print(sth)
```

### Use for statement to pick up values

Example 18: List every integer from 1 to 10.

```
for i in range(1,11):
 print(i)
```

Output:

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

**Note:** The `range()` function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default). `(1,11)` means values from 1 to 10 (not including 11)

Example 19: Square of every integer from 1 to 10.

```
for i in range(1,11):
 print(i**2) #or i*i
```

Output:

```
1, 4, 9, 16, 25, 36, 49, 64, 81, 100
```

### Use for loop to calculate

Example 20: Calculate the summation from 1 to 100.

```
total = 0
for i in range(1, 101): # numbers which are >=1 and <101
 total = total + i
print(total)
```

Output:

```
5050
```

## While loop

We can execute a set of statements in while loop as long as a condition is true.

### Syntax

```
while ...: #close with an ':'
 print(sth) #indented
 if ...: # you can insert `if` in `while`
 print(sth)
```

Example 16:

```
i = 1
while i < 6:
 i = i + 1
 print(i)
```

Output:

```
2
3
4
5
6
```

Example 17:

```
i = 1
while i < 6:
 print(i)
 i = i + 1
 if i == 3:
 break #we will talk this later
```

Output:

```
1
2
```

## Difference between for loops and while loops

1. While Loops allow you put a condition in it, like `while i<10`, and it will stop when the condition no longer being meet( `i >= 10`). you can also substitute in a boolean(true/false) for 10 as well as many other types of variables.
2. For Loops allow you to run through the loop many times you'd like it to run through the problem such as `for i in range(0,100)`, this will continually increase `i` until that condition returns false(`>100`), you can replace 10 with other numbers and variables, like `for name in name_list`, means that you want to loop the whole `name_list` to run through the problem. And it will quit once the condition is no longer being met.
3. Generally speaking, if you want to use loop to do conditional comparison, `while` loops is work for you, if you want to loop every elements of a whole list, `for` is better.

## Break and Continue statement

### Break statement

Stop the loop even if the while condition is true.

Example 22:

```
i = 1
while i < 9:
 print(i)
 if i == 5:
 break
 i = i + 1
```

Output:

```
1
2
3
4
5 #stop the loop
```

## Continue statement

Stop the current iteration, skip certain value, and continue with the next.

Example 23:

```
i = 1
while i < 9:
 i = i + 1
 if i == 5:
 continue
 print(i)
```

Output:

```
2
3
4 #number 5 is missing, while the loop continues
6
7
8
9
```

## Integrated example: for and if

Example 21: Calculate the break-even point of the simple business model.

Like the [example](#) we used before. Find that break-even point of subscribed users to make profit.

```
coding: utf-8

Fixed_Cost = 30000
Content_Cost = 70000

member_ff = 15
convert_rate = 0.1
ad_revenue_each_person = 1

num = float(input('please input your estimate number of subscribers:')) #input a estimated number
for i in range(0,int(num)):
 if i < 50000:
 Total_Cost = Fixed_Cost + Content_Cost
 else:
```

```

 Total_Cost = Fixed_Cost + Content_Cost + 0.1 * (i - 50000)

 Revenue = (1*i) + (0.1*15*i)
 Net_Income = Revenue - Total_Cost

 if Net_Income >= 0:
 print('subscribers= ',i)
 break

 if Net_Income < 0:
 print('Net_Income=', Net_Income) #the max value return by your in

```

Output:

```
subscribers= 40000
```

## Bonus: alternatives and more efficient calculation

Here is a challenge upon last example: Try to scale up the parameters by a uniform factor and test the efficiency of your program. For example, let `Fixed_Cost = 3000000` and `Content_Cost = 7000000`, the resulting number of subscribers will scale up by the same factor. However, it takes much more time for your program to run because the loop executes for more iterations. Can your program scale up 1000x, or 1000000x? If it takes very long to get the result, how do you think you can improve?

The basic idea is to "jump" somehow, instead of increasing the number of subscribers by only `1` every time. You can jump by `100` or `1000` depending on the problem scale. A more efficient solution is the [bisection method](#). See the discussions from our students on [Issue #34](#).

## Function

### Function definition (`def`)

A function is a block of code which only runs when it is called. You can call this function by passing parameters into a function. Then the function can return data as a result.

```

def function_name(parameters): #define function
 control flow #use control flow
 return #return to default or specified value

function_name("parameter1") #call the function

```

#### How `def` works

- Keyword `def` (means definite) marks the start of function header.
- A function name to uniquely identify it.
- Parameters (arguments) are optional, through which we pass values to a function.
- A colon `:` to mark the end of function header.
- Describe what the function does.
- Statements must have same indentation level (usually click `tab` on your keyboard to indent).
- An optional `return` statement to return a value from the function.
- when you find you are using a function again and again. you can use "def statement" to duplicate the logic.
- Type "Tab" to move the section rightwards. Type "tab"+"Shift" to move the section leftwards.

Example 24: based on the example we talk about above. Build a def function to calculate the profits when you give different number of users.

1. when  $50,000 \leq \text{the number of user} \leq 100,000$ , the cost=  $10000 + 0.1 \times (\text{number\_of\_users} - 50,000)$ ;
2. when the number of user  $\geq 100,000$ , the cost=  $10,000 + 0.1 \times (100,000 - 50,000) + 0.2 * (\text{number\_of\_users} - 100,000)$ .

```
def calculate_profit(number_of_users):
 if number_of_users < 50000:
 cost = 10000
 elif number_of_users <= 100000: # 500000 <= number_of_users <= 100000
 cost=10000+0.1*(number_of_users-50000)
 else: # number_of_users > 100000
 cost = 10000 + 0.1*(number_of_users-50000) + 0.2 * (number_of_users - 100000)
 revenue = 0.1 * number_of_users
 profit = revenue - cost
 return profit

print(calculate_profit(100))
```

Output:

```
-9990.0
```

The advantage of using "def" is that you can recall the function again and again. Just change the parameter. For example:

```
print(calculate_profit(100))
print(calculate_profit(1000))
print(calculate_profit(10000))
print(calculate_profit(100000))
```

Output:

```
-9990.0
-9900.0
-9000.0
-5000.0
```

## Bonus: Scope of variables in function

Try the following test program:

```
a = 1
print('(outside) a=', a)

def change():
 a = 2
 print('(in change) a=', a)
change()

print('(outside) a=', a)
```

The result is:

```
%python3 test.py
(outside) a= 1
(in change) a= 2
(outside) a= 1
```

One can see that the "same" variable `a` has different value inside and outside a function. The operation inside a function does not affect the outer variable `a`. This is a matter of "scope". Every variable, function, or more generally "token"/ "symbol" in Python has its scope of effectiveness. The inner function definition of `a` masks the definition outside `a`. In order to make the operation inside one function able to operate the variables outside, one can use `global` and `non-local` keywords. Please see more details from [this blog post](#). However, using global variable is not recommended because it makes the program hard to maintain when the size becomes large. There are many ways to work around. In the above example, we can use simply pass `a` into the function via argument list and use return statement to return the changed value:

```
a = 1
print('(outside) a=', a)

def increase(a):
 a = a + 1
 print('(increase) a=', a)
 return a

a = increase(a)
print('(outside) a=', a)

a = increase(a)
print('(outside) a=', a)
```

We change the simple assignment to addition to show the result of multiple execution. The output is:

```
%python test.py
(outside) a= 1
(increase) a= 2
(outside) a= 2
(increase) a= 3
(outside) a= 3
```

## Errors and Exceptions

### Try and except

For most errors, the Python interpreter will issue an exception. In fact, in many cases, we need to control the code that may generate exceptions. In python, error and exception handling allows us to continue our program if an exception occurs.

The try statement works as follows.

1. the try statement between the try and except (keywords) is executed.
2. If no exception occurs, the except clause is skipped and execution of the try statement is finished.
3. If an exception occurs during execution of the try clause, and its type matches the exception named after the except keyword, the except clause will be executed, **and the execution continues after the try statement**.
4. If an exception occurs which does not match the exception named in the except clause, it is passed on to outer try statements; if no handler is found, it is an unhandled exception and execution stops.

Example 25:

```
try:
 print("Hello World")
except:
 print("This is an error message!")
```

Example 26:

```
while True:
 try:
 x = int(input("Please enter a number: ")) #input a int and non-int to test
 break
 except ValueError:
 print("Oops! That was no valid number. Try again...")
```

## Raising errors

The `raise` statement allows you to force a specified exception to occur. It can be used in following `try` and `if`.

Example 27:

```
x = 5
if x < 10:
 raise ValueError('x should not be less than 10!')
```

## Exception error types

There are several common exception errors:

- `IOError` . If the file cannot be opened
- `ImportError` . If python cannot find the module
- `ValueError` . Raised when a operation or function receives an argument that has the right type but an inappropriate value
- `KeyboardInterrupt` . Raised when the user hits the interrupt key (normally Control-C or Delete)

Note that "Exception" is in fact `class` in Python. Python has many built-in exceptions that are grouped in a hierarchy known as inheritance/ derivation. The benefit is that, the outmost layer of codes can catch a broader exception while inner layer of codes can raise a narrower/ more specific exception at the same time. For the full list of built-in exceptions, please see the official documentation on [exceptions](#).

## Common coding patterns

### Representing a dataset

A usual dataset can be think of as a table composed of rows and columns. One row is one **data point**/ sample/ response/ observation/ record/ entry/ object. One column is one **feature**/ variable/ property/ dimension. You may see different terms from different literature and problem domain. In our discussion the **strong** term will be used but other terms are also used interchangeably sometimes, especially when discussing with external references.

There are several common ways to represent a dataset:

- List-of-list/ 2D array/ Matrix

```
dataset = [
 [f_11, f_12, ... f_1m], # 1st data point
 [f_21, f_22, ... f_2m], # 2nd data point
 ...
 [f_n1, f_n2, ... f_nm] # nth data point
]
```

One advantage of this notation is its compact representation of a lot of data. Another advantage is its natural fit into scientific computation, especially machine learning routines, because many such routines rely on a matrix structure. The disadvantage is the lack of "variable names", or "headers" in a table representation. You need to maintain such information outside this matrix structure.

- List-of-dict/ list of records/ records

```
dataset = [
 {
 "feature1": value_11,
 "feature2": value_12,
 ...
 "featurem": value_1m,
 }, # 1st data point
 {
 "feature1": value_21,
 "feature2": value_22,
 ...
 "featurem": value_2m,
 }, # 2nd data point
 ...
 {
 "feature1": value_n1,
 "feature2": value_n2,
 ...
 "featurem": value_nm,
 }, # nth data point
]
```

One can readily see the advantage of this representation compared with the previous one is its high readability. Although Python's grammar is flexible, there are certain meaning implied by list and dict. For a **list**, the elements should be of the same nature. Or say, they are the same category of objects. You can put "apple" and "orange" in the same list if the list intends to hold a series of fruits. A counter-example, which looks unnatural, is `["apple", "orange", 2018, "Sept", "University"]`. As to a **dict**, the keys are usually of different nature. One key usually describes certain aspect of an element; All the keys when put together completely describe one element. Once you understand the nature of list and dict, you feel the second way of dataset representation straightforward: Every element of outer layer list is one data point; Every key of inner dict is one variable; The value of variable `v` and data point `i` can be referenced by `dataset[i]["v"]`.

This way of data representation is most commonly found in many data APIs (See chapter 4). The advantage is that we do not need another structure to store table header. However, this is also the disadvantage -- the table header/ dict keys are repeated as many times as number of data points, making data transfer rather inefficient.

- Dict-of-list/ series/ column-first representation

```
dataset = {
 "feature1": [value_11, value_12, ... value_1m],
 "feature2": [value_21, value_22, ... value_2m],
 ...
 "featurem": [value_n1, value_n2, ... value_nm],
}
```

This representation gives a good trade-off between the previous two representations. The format also has a famous name called "column-based representation", which is common in numeric computation software/ statistics software like MATLAB and R. You will also find this representation an easy fit into visualisation libraries. For example, `matplotlib.pyplot.plot(dataset["feature1"], dataset["feature2"])` works smoothly.

- Dict-of-dict

This format is seen less frequently but one has no problem understand it. Note that `dict` can be thought of a powerful version of `list`, by assigning index to keys and elements to values. You will find the syntax to refer the same element from original list and this new dict the same. The minor difference is that the keys of `dict` is not ordered but the indices of a `list` are ordered. Without bothering with details, one can convert list into dict with this shortcut: `dict(zip(range(len(mylist)), mylist))`.

## Handle repeated works

As a beginner, the best approach to problem solving is "bottom-up". We will show you a lot of this approach during lectures and class demos. You need to pay attention how the instructor solves a problem, instead of focusing on the final code block that solves the problem. That is, **process is more important than result**. In this quasi text book, we note a common pattern here.

You will meet a lot of repeated works in programming. For example, downloading all the PDFs from a website, add the scores of selected students by 1, change the format of a whole dataset. Saving repeat manual work is one core advantage of programming. However, before you try to touch all the input elements, you need to solve the case for one element. Make sure you test different scenarios and find it work before proceed. Then you can wrap the logics into a function and invoke following pattern.

```
def select(element):
 if condition: # change "condition" to True if you want to select every element
 return True
 else:
 return False

def handle(element):
 # Do something with element
 result = ...
 return result

input_list = [...] # Put the repeated problems into this list
output_list = [] # output list is empty at the beginning

for element in input_list:
 if select(element):
 result = handle(element)
 output_list.append(result)
```

`for` loop is your best friend to handle repeated works where the problem size is predefined.

You can further structure your code in this way.

```
for element in input_list:
 if not select(element):
 continue
 result = handle(element)
 output_list.append(result)
```

**QUIZ:** What are the pros and cons of those two different writing styles?

## Infinite loop (daemon)

Sometimes, you want your program to work infinitely: wait for certain input; take action and wait again. This is a common pattern in system design. For example, a web server constantly takes "HTTP request" from client software (e.g. web browser) and send "HTTP response" to the client. You don't want the server to stop just after serving one HTTP request. Similarly, imaging you are going to write a Python script that monitors No. 8 Typhoon signal and post a

Twitter when [HKO](#) issues the warning (Ah! "Artificial Intelligence"! Robot!). You don't want the program to exit after posting one message. Instead you want to program to keep working until you terminate it/ interrupt it. A common pattern is as follows.

```
import time
i = 0
while True:
 print('Entering a new round of loop')

 # Do something useful here
 print('Working hard! This is the %d-th time' % i)
 ...

 print('Leaving a new round of loop')
 i = i + 1 # Count the number iterations
 time.sleep(1000) # to avoid too frequent poll of external resources
```

Once you execute this script, a lot of messages will be output to the screen non-stop. You can terminate the script, or technical "interrupt" the script by pressing `control+c` if you are in a Terminal/ shell environment. Always keep this hotkey in your mind. It is life saving sometime when you go deeper into the journey. For example, your web scraper may have bug and consumes extremely large amount of network data when you execute the script. You don't have to wait till the end of scraping. Just use `control+c` to end the program. Note:

- It is a convention for shell environment, so you can also terminate other Unix/ Linux commands in this way.
- You can also terminate a for-loop using the same method.
- You can also terminate a function call that does not respond after a while using the same method.

## REPL

Read-Evaluation-Print-Loop (REPL) is a common pattern similar to the daemon pattern above. Actually, you have already see several times such pattern in action. One is the system shell (inside Terminal), another is the Python shell (Python interactive environment). The shell environment allows you to continuously input commands; It then evaluate your input to calculate the result; then it goes back to get another line of input, a.k.a. "loop". A REPL in Python looks like this:

```
while True:
 user_input = input('Prompt message:')
 if user_input == 'exit':
 break
 else:
 output = evaluate(user_input)
 print(output)
```

## if..else; OR try..except

Consider a common problem in programming: given number of students as `n` and number of books as `m`, calculate how many books each student can get. The answer seems straightforward:  $m / n$ .

However, there is a glitch: `n` can be `0` and the mathematical meaning of division is undefined in this case. We need to determine whether `n` is equal to `0` before the calculation. If so, we simply output a warning message like "can not divide books among 0 students". Here are two ways to implement this:

if..else pattern:

```
if n > 0:
 result = m / n
 print(result)
else:
```

```
print("can not divide books among 0 students")
```

try..except pattern:

```
try:
 result = m / n
 print(result)
except ZeroDivisionError:
 print("can not divide books among 0 students")
```

Some advocates of `try..except` in Python community tend to abuse `try` in all places where `if` was used. The code is usually shorter with the help of `try..except` structure, especially when you are in a large project with many layers of function calls. The downside is that, it is difficult to find which `except` finally handles the error, because 1) `except` selects certain `Exception` it is interested in; 2) `except` can further `raise` the `Exception` if it can not handle it. In order to make the error handling explicit, you resort to the `if..else` pattern but tend to write longer ("verbose" in programming jargon) codes, especially when there are many layers of function calls.

## Multiple loop

Suppose you are a matchmaker and want every PR firm to get in touch with every journalist. Following constructs are given:

```
def touch(PR, journalist):
 # this function simply let a single `PR` firm get in touch with a single `journalist`
 pass

PR_list = [
 #...
]

journalist_list = [
 #...
]
```

Now you can use a multiple loop to call each possible pair of combinations:

```
for p in PR_list:
 for j in journalist_list:
 touch(p, j)
```

Python is flexible. Codes are organised into "blocks" seen as a chunk of codes with the same indentation level. We already see expressions that creates code blocks like `for`, `if`, `def` and `with`. Blocks can be nested into larger blocks so that one can built more complex logics.

## Try sub-tasks and regroup success/ fail cases

When you further dive into the programming journey, you will find codes easily raise errors. You can use `try...except` to catch the errors and treat them separately. A common pattern is to divide your task into many smaller sub-tasks, `try` each one and assemble the results. For example, when you scrape a list of webpages, some may succeed and some may fail. You don't want the whole program to crash upon one fail case. Following structure may help:

```
tasks = [] # store all the components

success = []
fail = []
```

```
for task in tasks:
 try:
 # do something with task
 result = do_something_with(task)
 success.append((task, result))
 except Exception as e:
 fail.append((task, e))

for (task, result) in success:
 # maybe you want to assemble the `result`s from successful tasks
 pass

for (task, e) in fail:
 # analyse and handle the failed tasks
 pass
```

In our web scraping example, `tasks` can be a list of URLs to scrape and `do_something_with` is the single page scraping function that takes a URL as input and return the scraped data items if successful. How to implement this `do_something_with` function is the topic of [notes-week-07.md](#) but it is good to learn the basic idea here.

## Test and batch execution for repeated tasks

Suppose you can handle all the tasks by:

```
...
for t in tasks:
 o = handle(t)
...
...
```

The size of `tasks` may be very large which takes days or months to run. Sometimes, you want to do some pilot testing before setting it at full scale. Or, you risk running into errors after some heavy computation. In Python, we can use the list slicing syntax to cut down the task size and do not break the overall structure of the code. For example, if we want to test for the first 10 entries, we can:

```
for t in tasks[:10]:
 o = handle(t)
```

When we finish testing, we can comment out the list slicing part:

```
for t in tasks: #[:10]:
 o = handle(t)
```

Suppose you have 10 machines and 1000 tasks in `tasks`. You can copy the same code to all the machines and slice `tasks[i * 100: (i + 1) * 100]` for the i-th machine (i starts from 0).

## Exercises and Challenges

### Generate detailed mortgage schedule

In [notes-week-02.md](#), we made a mortgage calculator using the amortised formula from Wikipedia. Now that we have the monthly instalment number, we can further generate the mortgage schedule that a regular user is highly interested. A mortgage schedule is basically a table in which one row represents the status of a month. One row contains the following information: `month` , `instalment` , `interest in instalment` , `principal in instalment` , `outstanding principal after this instalment` . The first 12 months are as follows. Try to print this table.

Month	Instalment	Interest	Principal	Outstanding
001	65995.57	41666.67	24328.91	9975671.09
002	65995.57	41565.30	24430.28	9951240.82
003	65995.57	41463.50	24532.07	9926708.74
004	65995.57	41361.29	24634.29	9902074.46
005	65995.57	41258.64	24736.93	9877337.53
006	65995.57	41155.57	24840.00	9852497.53
007	65995.57	41052.07	24943.50	9827554.02
008	65995.57	40948.14	25047.43	9802506.59
009	65995.57	40843.78	25151.80	9777354.80
010	65995.57	40738.98	25256.60	9752098.20
011	65995.57	40633.74	25361.83	9726736.37
012	65995.57	40528.07	25467.51	9701268.86
...				

## Hint on table formatting

**Hint:** Use the floating point formatter to limit numbers to two digits. Use `{:03d}` to pad with extra zeros. In order for the table to look aligned, you may want to use `\t` as the delimiter. Actually, `\t` means `table` and is used to align texts to next table marker.

The initial part of the program with parameter settings are as follows:

```
r = 0.05 / 12
n = 20 * 12
P = 10000000
A = P * (r * (1 + r) ** n) / ((1 + r) ** n - 1)
print(A)
outstanding = P
m = 0
```

## Hint on math formula

**Hint:** The key of this exercise is about calculating the schedule **all in Python**. Given week02's knowledge, you can calculate and print out the result line by line. After week03, you will be able to use a loop to repeat the procedure and saves some codes. Following is an example of how to calculate the mortgage status of first month.

```
Suppose we already have r, n, P, A
value of A is "instalment" in our natural language

Initially
outstanding = P

After the first month
interest_in_instalment = r * outstanding
principal_in_instalment = A - r * outstanding
outstanding = outstanding - principal_in_instalment

Now try to print in our expected format
print(...)

After the second month
Try to continue the process
```

## Automatic writer for financial report

Financial report usually possesses some apparent patterns. We can analyse the patterns and use string templating to perform automatic writing. Actually most of the "AI" based "robotic journalist" is no complex than string templating. After this chapter, you can even use conditional branches to plugin corresponding text. For example, when the value

goes up, you say "increases by" and when the value goes down, you say "decreases". People who write this kind of template is called "meta reporter".

You can approach in following ways:

1. Use regular string templating. The `str.format` function can be useful, especially its keyword argument/dictionary form.
2. Use `jinja2`. It provides a more expressive templating environment. You can even write branching and looping logics in its template. `jinja2` is commonly used in web development but is not limited to that. With the help of `jinja2`, we can separate the concern: 1) frontend developer, or say "meta reporter", focuses on the presentation and language, i.e. how to embed the variables into the right place of the template; 2) backend developer, or say data supplier, focuses on data ingestion and data analysis. The bridge of the two worlds is a spreadsheet.

## Calculate credit and expense instalment

One reference blog post: [https://mp.weixin.qq.com/s/IB\\_b-4HnhHdVdRiLz5lPuw](https://mp.weixin.qq.com/s/IB_b-4HnhHdVdRiLz5lPuw)

Key points:

- Basic arithmetic in Python.
- Use `random` related library to simulate uncertainty.
- Use `for` loop to calculate month by month.

## References

- Chapter 4, 5, 6, 8, 9 of official Python 3 [tutorial](#)
  - Another [tutorial](#) with many examples (Chinese)
- 

If you have any questions, or seek for help troubleshooting, please [create an issue here](#)

# Week 04: Data Structure

- [Week 04: Data Structure](#week-04-data-structure) - [Composite data types (collection)](#composite-data-types-collection) - [List []](#list-) - [Common features of list](#common-features-of-list) - [List functions](#list-functions) - [List methods](#list-methods) - [in operator on list](#in-operator-on-list) - [List slicing](#list-slicing) - [Dict {}](#dict-) - [Common features of dict](#common-features-of-dict) - [Dict functions](#dict-functions) - [Dict methods](#dict-methods) - [in operator in dict](#in-operator-in-dict) - [Tuple ()](#tuple-) - [Bonus: Copy collection objects](#bonus-copy-collection-objects) - [Bonus: Class and objects](#bonus-class-and-objects) - [Create a class, the init() function](#create-a-class-the-init-function) - [Bonus: Class inheritance](#bonus-class-inheritance) - [Exercises and Challenges](#exercises-and-challenges) - [Divide HW1 groups randomly: (case contribution)](#divide-hw1-groups-randomly-case-contribution) - [Hint for group assignment challenge](#hint-for-group-assignment-challenge) - [Simple word frequency](#simple-word-frequency) - [Bonus: pretty print in ordered format](#bonus-pretty-print-in-ordered-format) - [Bonus: Use the Counter class](#bonus-use-the-counter-class) - [Convert Hong Kong district names into a convenient data structure](#convert-hong-kong-district-names-into-a-convenient-data-structure) - [Conversion between simplified Chinese and traditional Chinese](#conversion-between-simplified-chinese-and-traditional-chinese)

## Composite data types (collection)

Here are some common composite data types:

Type	Values	Python Implementation
Array	Mutable object containing other values	list or []
Union	Contains values that can be multiple types	dict or {}
Record	Immutable object containing other values	tuple or ()

### List []

List is an ordered sequence of items. It is one of the most used datatype in Python and is very flexible. All the items in a list do not need to be of the same type.

Declaring a list is pretty simple. Items separated by commas are enclosed within brackets `[]`.

Example 5:

```
>>> a = [0, 6.6, 'python']
```

### Common features of list

1. Each element in the sequence is ordered and can be indexed. The first index is 0 and the second index is 1
2. Lists can be added and multiplied
3. One can add or remove elements into list
4. One can check whether one elements is in the list
5. One can slice the list

**Note:** We talk about `index` in chapter 2, for those who forget how it works, please refer to [here](#)

Example 6:

```
>>> test_list = ['Hello', 'Python', 2018, 814]
```

```
>>> test_list2 = [1, 2, 3, 4, 5, 6, 7]
>>> print ("test_list[0]: ", test_list[0]) #index[0] in test_list
test_list[0]: Hello
>>> print ("test_list2[1:5]: ", test_list2[1:5]) #index[1] to index[5] but does not include index5 value.
test_list2[1:5]: [2, 3, 4, 5]
>>> test_list + test_list2
['Hello', 'Python', 2018, 814, 1, 2, 3, 4, 5, 6, 7]
>>> test_list*2 #duplicate
['Hello', 'Python', 2018, 814, 'Hello', 'Python', 2018, 814]
>>> 2018 in test_list #check whether 2018 is in test_list
true
```

## List functions

Functions	Description
len(list)	Number of list elements
max(list)	The maximum value in the list
min(list)	The minimum value in the list element

Example 7:

```
>>> test_list2 = [1, 2, 3, 4, 5, 6, 7]
>>> len(test_list2)
7
>>> max(test_list2)
7
>>> min(test_list2)
1
```

## List methods

Methods	Description
append()	Adds an element at the end of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Example 8: All examples are corresponding to the list methods stated above.

```
>>> test_list = ['Hello', 'Python', 2018, 814]
>>> test_list.append(2049) #append() takes exactly one argument
>>> print(test_list)
['Hello', 'Python', 2018, 814, 2049]
```

```
>>> test_list2 = [23, 2018, 814, 2049, 2018]
```

```
>>> test_list2.count(2018) #count numbers of 2018
2
```

```
>>> test_list.extend(test_list2) #add test_list2 into test_list
>>> print("Extended List : ", test_list)
Extended List : ['Hello', 'Python', 2018, 814, 23, 2018, 814, 2049, 2018]
```

```
>>> print("Index for python : ", test_list.index('Python'))
1 #index value 'Python'
>>> print("Index for 2018 : ", test_list.index(2018)) #find the first position of the indexed value
2
```

```
>>> test_list = ['Hello', 'Python', 2018, 814, 23, 2018, 814, 2049, 2018]
>>> test_list.insert(3, 2009) #list.insert(index, object), insert value 2009 in the index3
>>> print("New List : ", test_list)
New List : ['Hello', 'Python', 2018, 2009, 814, 23, 2018, 814, 2049, 2018]
```

```
>>> test_list = ['Hello', 'Python', 2018, 2009, 814, 23, 2018, 814, 2049, 2018]
>>> test_list.pop(2) #delete index2 value and return this value
>>> print('List now : ',test_list)
List now : ['Hello', 'Python', 2009, 814, 23, 2018, 814, 2049, 2018]
>>> test_list.remove('Hello') #remove certain value
>>> print('List now : ',test_list)
List now : ['Python', 2009, 814, 23, 2018, 814, 2049, 2018]
```

```
>>> test_list = ['Python', 2009, 814, 23, 2018, 814, 2049, 2018]
>>> test_list.reverse() #reverse list
>>> print('reverse list : ',test_list)
reverse test_list : [2018, 2049, 814, 2018, 23, 814, 2009, 'Python']
```

```
>>> vowels = ['e', 'a', 'u', 'o', 'i']
>>> vowels.sort() #reverse = True(descending), reverse = False(ascending, if no parameters in(), they will return default value, ascending)
>>> vowels
['a', 'e', 'i', 'o', 'u']
>>> vowels = ['e', 'a', 'u', 'o', 'i']
>>> print('vowels ascending : ',vowels) # sort by ascending
vowels ascending : ['a', 'e', 'i', 'o', 'u']
>>> vowels.sort(reverse = True)
>>> print('vowels descending : ',vowels) # sort by descending
vowels descending : ['u', 'o', 'i', 'e', 'a']
```

## in operator on list

`in` operator in list is useful for checking if there is a member in the list or a collection. For example:

```
>>> my_list = ['chico', 419, 'Ri', 52, 6]
>>> 'Ri' in my_list
True
>>> 55 in my_list
False
```

## List slicing

In short, We use the colon `:` to slice the list. The following are the common usages:

```
a is a list
a[3:10] # items start from index3 to index10
a[3:] # items start from index3 to the end
a[:10] # items starts from the beginning to index10
a[:] # a copy of the whole list
a[-1] # last item in the list
a[-2:] # last two items in the list
a[:-2] # everything except the last two items
```

Apart from increasing or decreasing by integer 1, we can also add step in list slicing.

Syntax:

```
sliceable_list[start:stop:step]
```

The start and stop is already explained in the above example. For step - the amount by which the index increases, defaults to 1. If it's negative, you're slicing over the iterable in reverse. For example:

```
>>> r=[1,2,3,4,5,6]
>>>r[::-2] #iterate whole list, increased by step 2
[1, 3, 5]
>>> r[2::-2] #iterate from index2 to the end, increased by step 2
[3, 5]
>>> r[::-2] #iterate in reverse, decreased by step 2
[6, 4, 2]
```

## Dict {}

A dictionary is a collection which is disordered, changeable and indexed. In Python dictionaries are written with curly brackets {}, and they have keys and values, like `d = {key1 : value1, key2 : value2}`.

Example 9:

```
>>> my_dict = {
... "apple": "green",
... "banana": "yellow",
... "cherry": "red"
... }
>>> print(my_dict)
{'apple': 'green', 'banana': 'yellow', 'cherry': 'red'}
```

## Common features of dict

1. One can Access the values in the dict by `key`
2. Change dictionary by adding/deleting/updating key and values

Example 10:

```
>>> person_dict = {'Chico': 24, 'Ivy': 20, 'Ri': 29}
>>> print('Chico : ',person_dict['Chico']) #access values
Chico : 24
>>> person_dict['Ri'] = 19
>>> print('Ri : ',person_dict['Ri'])
Ri : 19
>>> person_dict['Frank'] = 31 #update value
>>> print('Frank : ',person_dict['Frank'])
Frank : 31
>>> del person_dict['Ivy'] #delete key
>>> print('New_dict : ',person_dict)
New_dict : {'Chico': 24, 'Ri': 19, 'Frank': 31}
```

## Dict functions

Functions	Description
len(dict)	Number of dict elements, which is the total number of keys
str(dict)	Output dictionary as a string

Example 11:

```
>>> person_dict = {'Chico': 24, 'Ri': 19, 'Frank': 31}
>>> len(person_dict)
3
>>> print("To String : %s" % str(person_dict))
To String : {'Chico': 24, 'Ri': 19, 'Frank': 31} #it's a string, not the same as original dict
```

## Dict methods

Methods	Description
fromkeys()	creates dictionary from given sequence
get()	Returns value of the key, default=None
items()	Returns view of dictionary's (key, value) pair
keys()	Returns view object of all keys
contains(key)	Return bool value by checking whether the key is in dict
pop()	Returns & removes element having given key
values()	Returns view of all values in dictionary
update()	Updates the Dictionary

Example 12: All examples are corresponding to the list methods stated above.

```
>>> seq = ['Chico', 'Ivy', 'Ri']
>>> p_dict = dict.fromkeys(seq) #get/create keys from the list
>>> print("New_dict : %s" % str(p_dict))
New_dict : {'Chico': None, 'Ivy': None, 'Ri': None}
>>> p_dict = dict.fromkeys(seq, 'A+') #give all keys value A+
>>> print("New_dict : %s" % str(p_dict))
New_dict : {'Chico': 'A+', 'Ivy': 'A+', 'Ri': 'A+'}
```

```
>>> p_dict = {'Name':'Chico','Gender':'Male','Age':'23'}
>>> print("Age : %s" % p_dict.get('Age')) #get key value
Age : 23
>>> print("Gender : %s" % p_dict.get('Gender'))
Gender : Male #if you get a wrong key, it will return None
```

```
>>> p_dict = {'Name':'Chico','Gender':'Male','Age':'23'}
>>> print("dict_values : %s" % p_dict.items()) #view dict's items
dict_values : dict_items([('Name', 'Chico'), ('Gender', 'Male'), ('Age', '23')]) #return a tuple
```

```
>>> p_dict = {'Name':'Chico','Gender':'Male','Age':'23'}
>>> print("dict_keys : %s" % p_dict.keys()) #view all keys
```

```

dict_keys : dict_keys(['Name', 'Gender', 'Age'])
>>> p_dict = {'Name':'Chico','Gender':'Male','Age':'23'}
>>> print("has_key : %s" % p_dict.__contains__('Age')) #two '_', check out keys
has_key : True
>>> print("has_key : %s" % p_dict.__contains__('School'))
has_key : False
>>> p_dict = {'Name':'Chico','Gender':'Male','Age':'23'}
>>> pop_value = p_dict.pop('Gender') #drop out key
>>> print(pop_value)
Male
>>> print(p_dict)
{'Name': 'Chico', 'Age': '23'}

```

```

>>> p_dict = {'Name': 'Chico', 'Age': '23'}
>>> print("Value : %s" % p_dict.values()) #get all values
Value : dict_values(['Chico', '23'])
>>> my_dict = {'Name': 'Chico', 'Age': '23'}
>>> new_dict = {'Gender':'Male'}

```

```

>>> my_dict.update(new_dict) #update new_dict in my_dict
>>> print('new_dict : %s' % my_dict)
new_dict : {'Name': 'Chico', 'Age': '23', 'Gender': 'Male'}

```

## in operator in dict

Likewise, one can use `in` operator to check whether there is certain key in the dict. For example:

```

>>> my_dict = {'Name': 'Chico', 'Gender': 'Male', 'Age': 23}
>>> 23 in my_dict
True

```

In dict, we can use `in` operator to do more, for example, we can build a dict to calculate the words frequency of an article(s), and store the value in the dict. You can refer to [this challenge](#) for further information.

## Tuple ()

A tuple is similar to a list, except that the elements of the tuple cannot be modified. The function and method of a tuple is similar to list, therefore we are not discussing more.

Example 13:

```

>>> tup = (1, 2, 3, 4, 5)
>>> print("tup[0]: ", tup[0])
tup[0]: 1

```

## Bonus: Copy collection objects

The assignment from one object to another object is essentially an assignment of "pointer". You can understand it as a reference to the object. This design is for efficiency purpose. Some non-intuitive behaviour to new learners may arise due to this design.

We first try the simple object (data type):

```

>>> a = 1
>>> b = a # The key line of assignment
>>> a
1

```

```
>>> b
1
>>> b = 'fffff'
>>> b
'fffff'
>>> a # a is not changed
1
```

Now let's test the `list` collection type:

```
>>> a = [1, 2, 3]
>>> b = a # The key line of assignment
>>> b
[1, 2, 3]
>>> b[2] = 'ffff'
>>> b # b is changed as expected
[1, 2, 'ffff']
>>> a # a is also changed
[1, 2, 'ffff']
```

The solution is to use `copy.deepcopy` to create a copy, not just a reference.

```
>>> import copy
>>> a = [1, 2, 3]
>>> a
[1, 2, 3]
>>> b = copy.deepcopy(a)
>>> b
[1, 2, 3]
>>> b[2] = 'ffff'
>>> b
[1, 2, 'ffff']
>>> a
[1, 2, 3]
```

## Bonus: Class and objects

Class is an abstraction that describes certain objects with the same properties and methods. It defines the properties and methods that are common to every object in the collection. An object is an instance of a class. The process creating an object from a class is called "instantiation" and is usually invoked by the "construct function" of a class. In Python, this function is called `__init__()`. The higher level concept is called "[Object Oriented Programming](#)", which appears in nearly all modern programming languages. Think of it as a way to model our real world. We will discuss OOP a bit later. This section is a quick peek into the basics -- `class` and `object`.

### Create a class, the `init()` function

All classes have a function called `__init__()`, which is always executed when the class is being initiated. Therefore the `__init__()` function is used to assign values to object properties, or other operations that are necessary to do when the object is being created.

Example 28: Create a animal class.

```
class Animal(): #class + class name to give a statement
 def __init__(self, name): #self refer to class itself, its default.
 self.name = name #all objects in this class has name

a = Animal("dog") #give a new object, an animal named dog
print(a.name)
```

Output:

```
dog
```

Example 29: Create a person class and give new object

```
class Person(): #build a person class
 def __init__(self, name, age): #those objects has name and age
 self.name, self.age = name, age
 def __str__(self): # def a certain return
 return 'My name is {self.name}, and I\'m {self.age} years old'.format(self=self) #review the format().function
 str(Person('xyc', 18)) #call the function by passing parameters in the function
```

Output:

```
My name is xyc, and I'm 18 years old
```

**Note:** In the example above, `return 'My name is {self.name}, and I'm {self.age} years old'.format(self=self)`

is equal to

```
return 'My name is {0}, and I'm {1} years old'.format(self.name, self.age)
```

So, what does this(`self=self`) means?

```
def __str__(self):
 return 'My name is {self.name}, and I\'m {self.age} years old'.format(self=self)
```

The first `self` in `self=self` refers to what the `format` function will change in your string. For example, it could also be

```
def __str__(self):
 return 'My name is {obj.name}, and I\'m {obj.age} years old'.format(obj=self)
```

The right-hand side `self` in `self=self` refers to the actual value that will be input. In this case, the object passed as argument. In other words, it could be written as

```
def __str__(obj):
 return 'My name is {self.name}, and I\'m {self.age} years old'.format(self=obj)
```

### Now why to use `self`?

It is just a convention used in python programming. Python passes to its instance methods automatically an object that is a pointer to itself. Can also check [this question](#) for further info.

Example 30:

```
class Account:
 def __init__(self, number, name):
 self.number = number
 self.name = name
 self.balance = 0

 def deposit(self, amount):
 if amount <= 0:
 raise ValueError('must be positive')
 self.balance += amount
```

```

def withdraw(self, amount):
 if amount <= self.balance:
 self.balance -= amount
 else:
 raise RuntimeError('balance not enough')

acct1 = Account('123-456-789', 'Chico') #open an account
acct1.deposit(500)
acct1.withdraw(100)
print(acct1.balance)

```

Output:

```
400
```

## Bonus: Class inheritance

"Class" is more than a group of common features, i.e. member variable and member functions. The real power of class comes when you get into the OOP world. The first step is to understand [class inheritance](#).

Now that you have the keyword, please try to search online to understand the syntax and grammar. Then follow some concrete examples to see how class inheritance can be used in real project. Our class does not emphasize OOP, so details are omitted here. Our core objective is to master the basics of Python and use it to collect, analyse and visualise data, in order to tell good stories. Most of the programs you will need to write during the class look "flat", i.e. we do not expect many layers of modules/ functions/ classes. However, when you build a large project, the OOP design patterns can make one more efficient, less error prone and easier to collaborate.

## Exercises and Challenges

### Divide HW1 groups randomly: (case contribution)

1. You can get the students' IDs from the file [chapter3-exercise-student-list.csv](#) and cases from the file [chapter3-exercise-case-list.csv](#).
2. Generate the grouping randomly, each team has 5 students and need be randomly distributed one case from 10 over all.

Sample input (part of your initial `.py` script):

```

student_list =[
 18421111,
 18421112,
 18421113,
 18421114,
 18421115,
 18421116,
 18421117,
 18421118,
 18421119,
 18421120,
 18421121,
 18421122,
 ...
 18421160
]
case_list =[
 'case1 - build a calculator to evaluate your business model',
 'case2 - build a automatic earthquake robot to broadcast the new earthquake',
 'case3 - evaluate social media performance of a luxury brand',

```

```
'case4 - study movie blockbuster \'Dying to Survive\'',
'case5 - invest your money like the Internet giant, Tencent',
'case6 - where are the 200,000 inferior vaccines flowing?',
'case7 - study classics, Who control the discourse power in \'Dream of the Red Chamber\'',
'case8 - research about Didi-driver crimes in China',
'case9 - \'Me too\' analysis',
'case10 - what is hip-hop in china?'
]

Write your code here
```

Sample output ( `print` to the screen):

```
Group 1
Student ID ID1
Student ID ID2
Student ID ID3
Student ID ID4
Student ID ID5
Assigned case n
=====
Group 2
...
```

## Hint for group assignment challenge

Following hints can help you think the algorithm but you do not have to use all the hints at the same time:

- Consider a multiple loop
- `random.shuffle()` OR `random.choice()` can be useful
- This is essentially a "mapping problem" and one powerful data structure designed for this type of problem is `dict`. You can use case as key and `list` of students as value.
- Always watch out for boundary conditions in programming: does your code still work when the number of students can not be divided by number of cases? Say 10 students, 3 cases.

## Simple word frequency

Word frequency is a common routine used in text analysis. Given a piece of English text, you can perform word frequency analysis in following steps:

1. Use `str.split()` to get a `list` of blank space separated words.
2. Use a `dict()` in this structure: `{word: frequency}`, where key is the word and value is an integer.
3. Loop over the `list` obtained in step 1 and use the accumulator in step 2 to count the frequencies.

In the end, the `dict` is our answer.

**HINT:** Before you accumulate in a dict like `ans[key] += 1`, you may want to check if the `key` actually exists in the `dict` by `key in ans` or `ans.contains(key)`.

**QUIZ:** Does this procedure work with Chinese? If so, please give the code or pseudo code. If not, in which step it fails?

## Bonus: pretty print in ordered format

The default print of `dict` has unreadable special chars like `{, }, : .`. Can you print the result in a ascii table like what we did in the mortgage schedule? Please rank the table from highest frequency to lowest frequency.

## Bonus: Use the Counter class

```
>>> from collections import Counter
>>> c = Counter()
>>> c
Counter()
>>> type(c)
<class 'collections.Counter'>
>>> isinstance(c, dict)
True
```

## Convert Hong Kong district names into a convenient data structure

Suppose we are building the profile information for Hong Kong's 18 districts. The data can be found on [wikipedia](#). It is in HTML's table format (or available as mediawiki notation). Both format is not easy for program to further process.

Before we learn scraping in week-05 and week-06, we can still do some simple processing here. We copy and paste the table into our source code to generate a tab-separated-values (TSV). With a bit string processing, we can process this raw string of data into list-of-dict structure or dict-of-dict structure.

You can start with the following code snippet:

```
a = '''
Central and Western 中西區 244,600 12.44 19,983.92 Hong Kong Island
Eastern 東區 574,500 18.56 31,217.67 Hong Kong Island
Southern 南區 269,200 38.85 6,962.68 Hong Kong Island
Wan Chai 灣仔區 150,900 9.83 15,300.10 Hong Kong Island
Sham Shui Po 深水埗區 390,600 9.35 41,529.41 Kowloon
Kowloon City 九龍城區 405,400 10.02 40,194.70 Kowloon
Kwun Tong 觀塘區 641,100 11.27 56,779.05 Kowloon
Wong Tai Sin 黃大仙區 426,200 9.30 45,645.16 Kowloon
Yau Tsim Mong 油尖旺區 318,100 6.99 44,864.09 Kowloon
Islands 離島區 146,900 175.12 825.14 New Territories
Kwai Tsing 葵青區 507,100 23.34 21,503.86 New Territories
North 北區 310,800 136.61 2,220.19 New Territories
Sai Kung 西貢區 448,600 129.65 3,460.08 New Territories
Sha Tin 沙田區 648,200 68.71 9,433.85 New Territories
Tai Po 大埔區 307,100 136.15 2,220.35 New Territories
Tsuen Wan 荃灣區 303,600 61.71 4,887.38 New Territories
Tuen Mun 屯門區 495,900 82.89 5,889.38 New Territories
Yuen Long 元朗區 607,200 138.46 4,297.99 New Territories
'''
```

Please try to workout a variable `d` for our future lookup. Given the english name as `name`, we can get the district's Chinese name via `d[name]['Chinese']` and population by `d[name]['Population']`.

## Conversion between simplified Chinese and traditional Chinese

Write a function:

- Input: `str` -- simplified Chinese
- Output `str` -- traditional Chinese

You can use a `dict` to maintain the mapping and use `for` loop to process every part of the paragraph.

**TIP:** You can not build a complete converter, but you can still try and build part of it.

# Week 05: Serialization - File, CSV, JSON

- [Week 05: Serialization - File, CSV, JSON](#week-05-serialization--file-csv-json) - [Objective](#objective) - [File operation](#file-operation) - [Write a file](#write-a-file) - [Read a file](#read-a-file) - [Append ()](#append-) - [CSV](#csv) - [csv.reader](#csvreader) - [csv.writer](#csvwriter) - [Write row](#write-row) - [Write rows](#write-rows) - [Exercise](#exercise) - [JSON](#json) - [JSON functions](#json-functions) - [json.dumps: Serialize object to a JSON formatted str](#jsondumps-serialize-object-to-a-json-formatted-str) - [json.loads: Deserialize JSON str to a Python object](#jsonloads-deserialize-json-str-to-a-python-object) - [json.load & json.dump](#jsonload--json-dump) - [Exercises and Challenges](#exercises-and-challenges) - [References](#references)

Previously, we learned composite data types, and the basic control flows. This week, we will learn how to get data, convert data and save data. Before we step into scraping from the HTML, we will introduce a more convenient method - API. From this week on, you are gradually entering the door of "big data", the API allows you to get more types and a larger bass of data at the same time. After this week, you can use API method to request data from open data websites and social media platforms. And even create an automatic writing robot on Twitter.

## Objective

- Learn to use Jupyter notebook. All demos following this week will be conducted in Jupyter notebook.
- Understand API/ JSON and can retrieve data from online databases (twitter, GitHub, weibo, douban, ...)
- Understand basic file format like JSON and CSV.
  - Be able to comfortably navigate through compound structures like {} and [].
  - Be able to comfortably use (multiple layer of) for-loop to re-format data.
  - Be able to use serialisers to handle input/ output to files.

The brief of Application Programming Interface (API):

Operate in client-and-server mode. Client does not have to download the full volume of data from server. Only use the data on demand. Server can handle intensive computations that not available by client. Server can send updated data upon request.

## File operation

### Write a file

Step 1: Create the file

```
f = open("test.txt", "w")
```

- You can write different kind of files by changing file name, like `name.txt`, `name.json`, `name.json`. But different file has different ways to write and read.
- 'w' means write.

Step 2: Write content

```
f.write('python tutorial')
f.close()
```

- `f.close()` to close the writing, this will close the instance of the file `test.txt` stored.

## Read a file

You also need to open the file first and then read the content.

### Syntax:

```
f = open("test.txt", "r+")
contents=f.read() #read all
content =f.read(6) #read first 6 characters
```

- 'r' means read. 'r+' means it will read from the beginning, if you want to print certain part of the string, you should use this method.

## Append ()

The append function is used to append to the file instead of overwriting it. To append to an existing file, simply open the file in append mode ("a")

### Syntax:

```
h = open("Hello.txt", "a")
write("Hello World again")
h.close
```

## CSV

The CSV (Comma Separated Values) format is the most common import and export format for spreadsheets and databases, just like `xlsx` file, which is used for storing your data and make it easy to read and write. You don't need to pip3 install csv. Just `import csv` before using it.

The CSV has two basic functions: `reader` and `writer`. Objects read and write.

### `csv.reader`

Return a reader object which will iterate over lines in the given CSV file. Each row read from the CSV file is returned as a list of strings. No automatic data type conversion is performed.

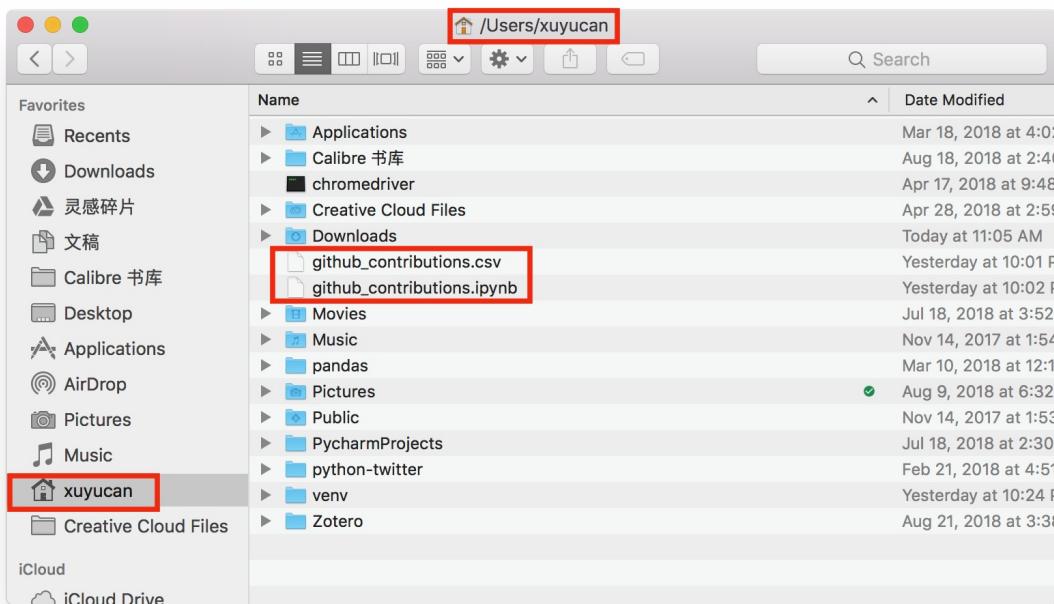
Suppose we have a CSV file, `name_list.csv`, you can download it [here](#). The content is as follows:

name	id	gender	location	phone
Chico	1742	M	KLN	3344
Ri	1743	F	LOS	5168
Ivy	1655	F	MK	7323

Example 1: How to read this CSV?

```
import csv
with open('chapter4-example-name_list.csv','r') as f: # open CSV
 rows = csv.reader(f) # read CSV
 for row in rows: #loop every row
 print(row)
```

Note: If you download the csv, you should copy the csv file in the folder where your venv folder are. Usually, it's in the user path. All the files you write and read should in this folder.



Output:

```
['name', 'id', 'gender', 'location', 'phone']
['Chico', '1742', 'M', 'KLN', '3344']
['Ri', '1743', 'F', 'LOS', '5168']
['Ivy', '1655', 'F', 'MK', '7323']
```

#### Note:

- with open(...) as f means you give f a definition, which stands for opening the file. In the example, f can be changed by any words you like. It just means you rename the step of the opening. It's equal to f = open('chapter4-example-name\_list.csv', mode='r').
- open() means open the file. If there is no such file, it will create a new one. If there is a existing one, writer function will clear all the previous content and then write the new content.

## csv.writer

Return a writer object responsible for converting the user's data into delimited strings. CSV file can be any object with a write() method.

Example 2: How to write a CSV file?

## Write row

```
import csv
with open('name.csv', 'w') as f:
 mywriter=csv.writer(f) #writer is the function to write something
 mywriter.writerow(['Chico', 'Male']) #you can just use writer.writerow()
 mywriter.writerow(['Ri', 'Female']) #write another row
```

Output:

	A	B
1	Chico	Male
2	Ri	Female

Note:

- `w` means write. By the way, if you want to read the file, you can input `r`, representing "read".
- `csv.writer()` means to write something in the name.csv file.
- `writerow()` means write one row and then another row. The input should be list type. `writerows()` means they will write row after row until loop all the elements from a list.
- arguments in `writerow()` should be a list, because `csv` function treat a row as a list, therefore you should use `[]` to wrap up the argument.

## Write rows

Method 1:

```
import csv
mylist=[['KLN',1742,3344],['Los',1743,5168]]
with open('location.csv','w') as f:
 mywriter=csv.writer(f)
 mywriter.writerows(mylist) #if there are sub-list in the list
```

Output:

	A	B	C
1	KLN	1742	3344
2	Los	1743	5168

Method 2:

```
import csv
number_list = [11,22,33,44,55,66]
with open('number.csv','w',) as f:
 mywriter=csv.writer(f)
 mywriter.writerows([[number] for number in number_list]) # equal to a for loop
```

Output:

	A
1	11
2	22
3	33
4	44
5	55
6	66

**Method 3:**

```
student_list = ['Chico', 'Ri', 'Ken', 'Aaliyah', 'Voodoo']
with open('student.csv', 'w') as f:
 writer=csv.writer(f)
 for i in range(0,len(student_list)):
 writer.writerow(student_list[i])
```

Output:

	A	B	C	D	E	F	G
1	C	h	i	c	o		
2	R	i					
3	K	e	n				
4	A	a	l	i	y	a	h
5	V	o	o	d	o	o	

**Note:** I guess you already find what's wrong here. `writerows()` means write all the rows in one time. It writes every item of the list into a row. Like we said, csv.writerow function treat a row as a list, for which it will regard the first row 'Chico' as a list of items with 5 characters, or 5 strings. Then it is put in 5 cells. So how to avoid spilt characters of a string into different cells?

Try change `writer=csv.writer(f)` to `writer=csv.writer(f, delimiter=' ')`, see what will happen.

**Exercise**

1. Write row ['hello','python'] in A1 in the CSV file

```
import csv
with open('hello.csv','w') as f:
 mywriter=csv.writer(f)
```

```
mywriter.writerow(['hello','python']) #writerow('hello','python') won't work, you should put it in a list.
Every item of the list will be write in a cell of the table.
```

1. Use writerows to write [['spam','1'],['22','333'],['OK','Good']] in the CSV file

```
import csv
with open('test.csv','w') as f:
 mywriter=csv.writer(f)
 mywriter.writerows([['spam','1'],['22','333'],['OK','Good']])
```

	A	B	C
1	spam	1	
2	22	333	OK
3	Good		

There are 3 items, or 3 lists, therefore it will output 3 rows. And inside each row, there are different items, So it will write each item in one cell.

## JSON

JSON (JavaScript Object Notation) is a lightweight data interchange format inspired by JavaScript object literal syntax.

- JSON is a syntax for storing and exchanging data.
- JSON is text, written with JavaScript object notation.

### What does JSON looks like?

Example 3:

```
{
 "firstName": "Jane",
 "lastName": "Doe",
 "hobbies": ["running", "sky diving", "singing"],
 "age": 35,
 "children": [
 {
 "firstName": "Alice",
 "age": 6
 },
 {
 "firstName": "Bob",
 "age": 8
 }
]
}
```

### Why to use JSON?

The advantages of JSON:

1. The data size is small. Compared with XML(another) file type to store and exchange data, JSON is small in size

and faster in passing.

2. The transmission speed is fast. JSON is much faster than XML.
3. Data format is simple, easy to read and write, and the format is compressed.
4. Easy to use with python, JSON is a form of `k-v` structure.

In simple terms, JSON is a dict, which has keys, each key corresponds to a value. The middle is separated by `:`, the outermost is surrounded by `{}`, and the different key-value pairs are separated by `,`. Example like this

```
{"key1": "value1", "key2": "value2", "key3": "value3"}
```

If there is a case where a Key corresponds to multiple values, use `[]` to include all the corresponding values.

```
{"key1": ["v11", "v12", "v13"], "key2": "v22"}
```

Now you can go back to have a look of example 3, test yourself whether you have understand the JSON structure.

## JSON functions

### `json.dumps`: Serialize object to a JSON formatted str

#### Syntax

```
json.dumps(obj, skipkeys=False, ensure_ascii=True, check_circular=True, allow_nan=True, cls=None, indent=None,
separators=None, encoding="utf-8", default=None, sort_keys=False, **kw)
```

Don't be panic, we do not have to use those all parameters. Basically, you need to know those two.

1. If `sort_keys` is true (default: False), then the output of dictionaries will be sorted by key.
2. If `ensure_ascii` is true (the default), the output is guaranteed to have all incoming non-ASCII characters escaped like Chinese. If `ensure_ascii` is false, these characters will be output as-is.

Example 4: Convert Python object data into JSON string

```
import json

data = {
 'name' : 'ACME',
 'shares' : 100,
 'price' : 542.23
}
json_str = json.dumps(data)
json_str
```

Output:

```
'{"name": "ACME", "shares": 100, "price": 542.23}'
```

If you want to save the JSON string to a file, so that others can re-use this file, you can use `open()` with the `.write()` function to achieve that.

```
with open('json_data.json', "w") as f:
 f.write(json_str)
 f.close()
```

**TIP:** In Jupyter notebook, you can write shell commands after `! . cat` is essentially a shell command that reads the content of a file and output to the screen. It is a common way to check if the output (to a file) is intended. In [notes-week-01.md](#), we have learned some useful commands like `cd`, `pwd` and `ls`. Those can all be used here. Also recall how we install new Python modules in a Jupyter notebook: `!pip install <package-name>`.

```
!cat json_data.json
```

```
{"name": "ACME", "shares": 100, "price": 542.23}
```

## json.loads: Deserialize JSON str to a Python object

### Syntax

```
json.loads(s, *, encoding=None, cls=None, object_hook=None, parse_float=None, parse_int=None, parse_constant=None, object_pairs_hook=None, **kw)
```

Example 5: Parse JSON string to Python internal data structure.

```
json_str = """
{
 "dataset": {
 "train": { "type": "mnist", "data_set": "train", "layout_x": "tensor" },
 "test": { "type": "mnist", "data_set": "test", "layout_x": "tensor" }
 }
}

import json
result = json.loads(json_str)
```

**TIP:** when you write a large chunk of data in Python, this "multi-line string", "block string literal", or "verbatim" is very useful. It helps you to reserve all the format like indentations in the data. This feature appears in nearly all programming languages and the official name is HEREDOC.

In the above example, you can `.read()` the `json_str` from a file instead of using HEREDOC. Try it yourself.

Output: the content (string representation) of `result`

```
{"dataset": {"test": {"data_set": "test",
 "layout_x": "tensor",
 "type": "mnist"},
 "train": {"data_set": "train", "layout_x": "tensor", "type": "mnist"}}}
```

## json.load & json.dump

Actually, if we want to convert between json file with python object. We can directly use `json.load` & `json.dump`. The difference between `loads` and `load` OR `dumps` and `dump` is that you can get the string by using `-s` method. And sometimes, we need those strings to do other things instead of justing writing into files.

Example 6: Converting between JSON file and Python object

```
stu = {
 "age": 20,
 "score": 88,
 "name": "Bob"
}
```

```
with open('stu.json', 'w') as f:
 json.dump(stu, f) #converting Python object to JSON file
```

Open the stu.json file, it will output like:

```
{"age": 20, "score": 88, "name": "Bob"}
```

```
with open('stu.json', 'r') as f:
 data = json.load(f) #converting JSON file to Python object
data
```

Output:

```
{"age": 20, "score": 88, "name": "Bob"}
```

## Exercises and Challenges

TODO

## References

- Python official doc about [json](#)

---

If you have any questions, or seek for help troubleshooting, please [create an issue here](#)

# Week 06: API

- [Week 06: API](#week-06-api) - [Requests](#requests) - [Make a Request](#make-a-request) - [Response Content](#response-content) - [API](#api) - [Why use api](#why-use-api) - [Use API via HTTP request/ response](#use-api-via-http-request-response) - [Test API without using Python](#test-api-without-using-python) - [Use API via function calls to other modules/ packages](#use-api-via-function-calls-to-other-modules-packages) - [Bonus: Your first automatic writing robot on Twitter](#bonus-your-first-automatic-writing-robot-on-twitter) - [Exercises and Challenges](#exercises-and-challenges) - [Newyork Times API](#newyork-times-api) - [Douban API](#douban-api) - [Yahoo Finance API](#yahoo-finance-api) - [GitHub API](#github-api) - [Twitter API](#twitter-api) - [Wechat API](#wechat-api) - [Google Map API](#google-map-api) - [Bonus: Real Estate property in Hong Kong (via government open data portal API)](#bonus-real-estate-property-in-hong-kong-via-government-open-data-portal-api) - [Bonus: Blockchain - chain data and exchange data](#bonus-blockchain---chain-data-and-exchange-data) - [Bonus: Automatic earthquake writer](#bonus-automatic-earthquake-writer) - [Bonus: MailGun API](#bonus-mailgun-api)

## Requests

Requests are the primary module we use to crawl website and get the content.

### Make a Request

Making a request is easy. First of all, please use `pip` to install requests before import. Then import the `requests` module. Usually, we use `requests.get` to make a request for data.

```
import requests
r = requests.get('http://www.imdb.com/list/ls058982125/') #2017 Movie List
```

Now, we have a Response object called `r`. We can get all the information we need from this object, information about the 2017 movie list.

### Response Content

After we get the response object `r`, we need to read the content of the server's response. There are three ways to read contents.

```
import requests
r = requests.get('http://www.imdb.com/list/ls058982125/') #pass url in ()
r.text
r.json
r.content
```

If you want to return text data, use `r.text` .(Use mostly)

If you want to return JSON, use `r.json`

If you want to return pictures and files, use `r.content` , it will return the binary data.

## API

Application Program Interfaces, or APIs, are commonly used to retrieve data from remote websites. Sites like Twitter, Douban and even government websites all offer certain data through their APIs. To use an API, you make a request to a remote web server, and retrieve the data you need.

## Why use api

Compared with other ways like download directly, APIs are useful in the following cases:

- The data is changing quickly. It doesn't need you to re-download it every minute - this will take a lot of bandwidth, and be pretty slow, the data we get through api is always up to date (unless you specify).
- Usually, the data sets return from APIs are usually can be directly convert to CSV or JSON, which is more structural and clean, you don't need to spend a lot of time to clean and find the data compared with request from `.html`.
- You can filter the data you want instead of download a large of data set.

Generally, different websites have different APIs methods to request the data. For example, the APIs of [USGS](#) and [Douban](#) is a url, and you can change some parameters in the url to get the different data. While, the twitter api is a set of tokens and keys, you can call different functions to get different data.

## Use API via HTTP request/ response

So, in the following example, we'll be querying a simple API to retrieve data about the last 100 years' earthquakes that happen in Taiwan.

Step 1: Find it's API, and read it's documentation

API link : <https://earthquake.usgs.gov/fdsnws/event/1/>

Different organizations and websites have their own rules of using API. For this earthquake API, you cannot just request all data from this original API link, you need to specify which region, what period of time you want. It's like declare what content/data you want to request. Then you can just pass those parameters following the original API link to request those data, you can click the above API link see their examples to learn more.

Step 2: Set arguments and functions we want use

Functions:

- `query?`
- `count?`

Arguments:

- Start time: 1918-08-24
- End time: 2018-08-24
- Min latitude: 21.890
- Min longitude: 119.300
- Max latitude: 25.320
- Max longitude: 122.030

You can get 4 location parameters from [worldatlas](#)

Step 3: Create the API We want

```
import requests
import csv

api_url = 'https://earthquake.usgs.gov/fdsnws/event/1/'
api_method = 'query?'
```

```

api_method_2 = 'count?'
api_format = 'format=geojson'
api_starttime = '1918-02-21'
api_endtime = '2018-02-21' #if you want to return data up to now, just to omit the endtime
api_minlatitude = '21.890'
api_minlongitude = '119.300'
api_maxlatitude = '25.320'
api_maxlongitude = '122.030'

query_url ='{}{1}{2}&starttime={3}&minlatitude={4}&maxlatitude={5}&minlongitude={6}&maxlongitude={7}'.format(api_url,api_method,api_format,api_starttime,api_minlatitude,api_maxlatitude,api_minlongitude,api_maxlongitude)
#prepare for calling query function

count_url ='{}{1}{2}&starttime={3}&minlatitude={4}&maxlatitude={5}&minlongitude={6}&maxlongitude={7}'.format(api_url,api_method_2,api_format,api_starttime,api_minlatitude,api_maxlatitude,api_minlongitude,api_maxlongitude)
#prepare for calling count function

```

#### Step 4: Requests content

If you just want to know how many earthquakes happen in the past 100 years, just use `count_url`, it will return how many data we get.

```

response = requests.get(count_url)
data = response.json()
data # equals to print(data)

```

Note: **#in Jupyter Notebook you can omit 'print()' by directly using its name to print something**

Output:

```
{"count": 3939, "maxAllowed": 20000}
```

If you just want to get all data and extract key information, just use `query_url`.

```

response = requests.get(query_url)
data = response.json() #response JSON
data # print all data

```

#### Step 5: Select the key values we want: mag, place and time, and write it in the csv file

You can check out the returned JSON, The outermost layer is a dict, and you will find all key information we want is in the key `features`, meanwhile, there are so many features. So firstly, we should extract all features by using `data['features']`, it will return a list of all features. And in the list, the information of every earthquake is in an dict. So we can further use keys to access its values.

```

with open('TTT.csv', 'w') as f:
 writer = csv.writer(f)
 header = ['place', 'mag', 'time']
 writer.writerow(header)

 for i in range(0, len(data['features'])):
 writer.writerow([data['features'][i]['properties']['place'], data['features'][i]['properties']['mag'], data['features'][i]['properties']['time']])

 f.close()

 # data['features'][i]['properties']['mag'] means to find all magnitude of the earthquake
 # data['features'][i]['properties']['place'] # find all location
 # data['features'][i]['properties']['time'] # find all time

```

Output:

place	mag	time
<b>4km NNE of Buli, Taiwan</b>	4.6	1534551317630
<b>3km NW of Buli, Taiwan</b>	4.5	1534500581110
<b>83km SE of Taitung City, Taiwan</b>	4.2	1533954258660
<b>34km ENE of Hengchun, Taiwan</b>	4.1	1533442171700
<b>7km ESE of Douliu, Taiwan</b>	4.2	1533217098370
<b>28km NNE of Taitung, Taiwan</b>	3.4	1531514555940
<b>40km NNE of Taitung City, Taiwan</b>	4.6	1531507120600
<b>7km SE of Jiayi Shi, Taiwan</b>	2.7	1531400455280
<b>13km NNE of Tainan, Taiwan</b>	2.6	1531343136010
<b>12km W of Yujing, Taiwan</b>	4.3	1531312911840
<b>43km NNW of Taitung, Taiwan</b>	4.3	1531231102740
<b>36km NNE of Taitung City, Taiwan</b>	4.1	1530648616320
<b>49km ENE of Yujing, Taiwan</b>	4.6	1530609657510
<b>4km SW of Jiayi Shi, Taiwan</b>	4.2	1530516369720

Quiz: You can see that the time is not what we want. Can you convert them to UTC time(Universal Time Coordinated).

## Test API without using Python

Note that `requests.get(url)` basically sends an `GET` HTTP request to the server. You can achieve this without using Python. Your web browser, like Google Chrome, sends a lot of `GET` requests every day to the websites you visit. You can use `print(url)` to get the assembled final URL and copy-and-paste this URL into your browser address bar, to test the response from the server. Since most API returns JSON data structure, you can use the [JSONView](#) extension in Google Chrome to get a better view and easier explore the response.

```
{
 type: "FeatureCollection",
 metadata: {
 generated: 1539593010000,
 url: "https://earthquake.usgs.gov/fdsnws/event/1/query?format=geojson&s...",
 title: "USGS Earthquakes",
 status: 200,
 api: "1.5.8",
 count: 12
 },
 features: [
 {
 type: "Feature",
 properties: {
 mag: 4.4,
 place: "7km NNE of Buli, Taiwan",
 time: 1537214897950,
 updated: 1539285835419,
 tz: 480,
 url: "https://earthquake.usgs.gov/earthquakes/eventpage/us2000hg3z",
 detail: "https://earthquake.usgs.gov/fdsnws/event/1/query?eventid=us2000hg3z&format=json",
 felt: 1,
 cdi: 3.1,
 mmi: null,
 alert: null,
 status: "reviewed",
 tsunami: 0,
 sig: 298,
 net: "us",
 code: "2000hg3z",
 ids: "us2000hg3z",
 sources: "us",
 types: "dyfi,geoserve,origin,phase-data",
 nst: null,
 dmin: 0.239,
 rms: 0.93,
 gap: 80,
 magType: "mb",
 type: "earthquake",
 ...
 }
 }
]
}
```

## Use API via function calls to other modules/ packages

Another API method that is worthy to introduce is using the keys and token they give to you, through which you can use the function they build to get the data. But a shortcoming about this method is there is very limit the amount of data you can get. The organizations with a large database are very strict about this. In this example, we will also use Taiwan example, to scrape users tweets under the keyword of 'Taiwan earthquake'.

Step 1: Check out `python-twitter` and install it

`Python twitter` is a library which provides a pure Python interface for the Twitter API. It works with Python 2.7+ and Python 3.

As usual, we use Jupyter Notebook to do exercise and pip way to install

```
!pip install python-twitter
```

### Step 2: Getting your application tokens

In order to use the `python-twitter` API client, you first need to acquire a set of application tokens. These will be your `consumer_key` and `consumer_secret`, which get passed to `twitter.Api()` when starting your application. Please refer to the [documentation](#) and apply your own key.

```
import twitter
import csv
api = twitter.Api(consumer_key) #please pass your own api key
```

### Step 3: Find the correct function to call

There are many functions to accomplish different demands. If you want to get status from one given twitter ID, `api.GetUserTimeline` is the right one. As for our example, we should use `api.GetSearch` to get the comments. Then change the parameters we want. For more functions, please see [here](#).

```

results = api.GetSearch(term='taiwan earthquake',
 raw_query=None,
 geocode=None,
 since_id=None,
 max_id=None,
 until='2018-08-28',
 since='2018-01-01',
 count=100, #maximum you can get
 lang='en',
 locale=None,
 result_type='mixed',
 include_entities=None,
 return_json=False)
len(results) #check how many statues you get
type(results) # check the data type

```

Step 4: Take out the value we want and write it to CSV file

```

with open('earthquake_comments.csv', 'w') as f:
 writer = csv.writer(f)
 header = ['Names', 'IDs', 'Time', 'Comments']
 writer.writerow(header)
 for i in range(0, len(results)):
 writer.writerow([results[i].user.screen_name,
 results[i].id,
 results[i].created_at,
 results[i].text])

```

Output:

earthquake_comments			
Names	IDs	Time	Comments
Anaweb0VLP	1034175018635350016	Mon Aug 27 20:25:13 +0000 2018	RT @Xiani_PCh: @ForeignPolicy And this is Kofi Annan refusing humanitarian aid to #Taiwan after a devastating earthquake <a href="https://t.co/lNfQj...">https://t.co/lNfQj...</a>
Xiani_PCh	1034076116745576448	Mon Aug 27 13:52:13 +0000 2018	@ForeignPolicy And this is Kofi Annan refusing humanitarian aid to #Taiwan after a devastating earthquake <a href="https://t.co/lNfQjMUJgqq">https://t.co/lNfQjMUJgqq</a>
ChinaKennedy	1033782658252169216	Sun Aug 26 18:26:07 +0000 2018	ROC former PM Dr. Zhao Xuan Liu (Chemistry, Toronto Univ, Canada) spent August 7, 2009 with his Dad on Chinese Fath... <a href="https://t.co/DsG11vD8g">https://t.co/DsG11vD8g</a>
ARDAmichael	1033719617850494976	Sun Aug 26 14:15:37 +0000 2018	So its my first time experiencing earthquake in my life and it happened here in Tainan, Taiwan 😊
TRB_	1033705845597458434	Sun Aug 26 13:20:53 +0000 2018	Just felt a small earthquake #taiwan
MagnumMisserium	1033469936008683526	Sat Aug 25 21:43:28 +0000 2018	RT @mathewstoller: 2. Lynn noticed that an earthquake in Taiwan took down a whole bunch of seemingly unrelated factories halfway across th...
laughtherpesok	1033410707184738304	Sat Aug 25 17:48:07 +0000 2018	RT @mathewstoller: 2. Lynn noticed that an earthquake in Taiwan took down a whole bunch of seemingly unrelated factories halfway across th...
Sheptacular	1033405231957245952	Sat Aug 25 17:26:21 +0000 2018	RT @mathewstoller: 2. Lynn noticed that an earthquake in Taiwan took down a whole bunch of seemingly unrelated factories halfway across th...
Jack0Spades	1033399446636298241	Sat Aug 25 17:03:22 +0000 2018	RT @mathewstoller: 2. Lynn noticed that an earthquake in Taiwan took down a whole bunch of seemingly unrelated factories halfway across th...
ChristoPierson	1033395794190188544	Sat Aug 25 16:48:51 +0000 2018	RT @mathewstoller: 2. Lynn noticed that an earthquake in Taiwan took down a whole bunch of seemingly unrelated factories halfway across th...
joke2power	1033387515359375104	Sat Aug 25 16:15:57 +0000 2018	RT @mathewstoller: 2. Lynn noticed that an earthquake in Taiwan took down a whole bunch of seemingly unrelated factories halfway across th...
HillTranquility	1033382468185931776	Sat Aug 25 15:55:54 +0000 2018	RT @mathewstoller: 2. Lynn noticed that an earthquake in Taiwan took down a whole bunch of seemingly unrelated factories halfway across th...
mathewstoller	1033381924511903745	Sat Aug 25 15:53:44 +0000 2018	2. Lynn noticed that an earthquake in Taiwan took down a whole bunch of seemingly unrelated factories halfway acros... <a href="https://t.co/O2sohydzae">https://t.co/O2sohydzae</a>
golpol0	1033356597064593410	Sat Aug 25 14:13:06 +0000 2018	Taiwan earthquake leaves several dead as rescuers desperately search for survivors <a href="https://t.co/c244xebPBQ">https://t.co/c244xebPBQ</a>
shiniiy	1033320228061503488	Sat Aug 25 11:48:35 +0000 2018	Earthquake in Lombok Bali, floods in Kerala, and now more floods in Taiwan. What's happening?
			Is Mother Earth unl... <a href="https://t.co/wSdjnis7OSR">https://t.co/wSdjnis7OSR</a>
republicbuzz	1033231447576829952	Sat Aug 25 05:55:48 +0000 2018	Search and rescue efforts at teetering apartment building hampered by strong aftershocks <a href="https://t.co/Op3N4xJWtH">https://t.co/Op3N4xJWtH</a> <a href="https://t.co/7TW7kZjyPQ">https://t.co/7TW7kZjyPQ</a>
AZCyberpro	1033147053117857792	Sat Aug 25 00:20:27 +0000 2018	Deadly earthquake strikes Taiwan; shelters opened <a href="https://t.co/r6DGUfllyP">https://t.co/r6DGUfllyP</a>
JasonKuo18	1032841445403635713	Fri Aug 24 04:06:04 +0000 2018	RT @lingwen: On behalf of the people of #Taiwan, I want to extend my condolences to all those affected by the #earthquake in #Japan this mo...
translatorbali	1032744405810069504	Thu Aug 23 21:40:28 +0000 2018	RT @MOfA_Taiwan: Our thoughts & prayers are with those affected by the earthquake that struck the island of #Lombok in #Indonesia. #Taiwan...
wendiwendi	1032439660042219520	Thu Aug 23 01:29:31 +0000 2018	Magnitude 4.1 earthquake jolts SE Taiwan   Taiwan News <a href="https://t.co/67h8fgppdk">https://t.co/67h8fgppdk</a>
I_Say_What	1032314190873030656	Wed Aug 22 17:10:57 +0000 2018	RT @babu_vardaan: BIG ONE SOON #EARTHQUAKE STRONG FREQUENCY RECEIVED

## Bonus: Your first automatic writing robot on Twitter

Although our focus in this chapter is to get data from different sources via API, some of the APIs are more than that. For example, you can use the `python-twitter` module to post a status to Twitter. Now that you already know how to template string (`%` and `str.format`) and how to post a Twitter status, you can combine them as your first *automatic writing robot!*

Here are some examples of such robots:

- A robot that summarises Initium Media articles and post status upon new release. You can find [source code](#), or

see it [in action](#).

- [EQBOT](#) gets earthquake updates in an interested region and post to Twitter when there is a new earthquake. The bot is reproduced by our TA as [@Chico.xyc](#), whose source codes are in this [notebook](#).

## Exercises and Challenges

### Newyork Times API

<https://developer.nytimes.com/>

### Douban API

Retrieve and analyse the recent movie. Douban's API will be helpful here:

- [API sample for Recent movies](#)
- [API sample for movie details](#)

Try to store the recent movie data into a CSV for future analysis. One example header could be:

```
title,director,year,cast,description,region
```

Reference use cases:

- [Hu Zizhe, COMM7780, S2018](#)

### Yahoo Finance API

Yahoo Finance provides rich set of stocks API. You can get company information and quote information from the API. The API is originally designed for their internal use. People have done a lot of analysis on it and reverse engineered the parameters so you can easily play around it. Find some example use from [this notebook](#).

### GitHub API

GitHub provides very convenient/ developer friendly access to its public database via [its API](#). For example, one can enumerate the people who have starred one repo by calling (HTTP request) the following API endpoint:

```
https://api.github.com/repos/{username}/{reponame}/stargazers
```

For example, click [here](#) to see the API response that includes all users who have starred our open book repo.

### Twitter API

Twitter provides official API to access its data and allows one to programmably post status updates on the platform. We have introduced the API and a Python package in previous sections. You can search for a certain keywords and get related tweets using this API.

Reference use cases:

- [XU Yucan, COMM7780, S2018](#)

### Wechat API

Wechat does not provide public API but its own web App has an internal HTTP API. People have done good analysis on it and built very convenient packages to retrieve data and post messages. [wechat-egonet](#) is a script based on the library called `itchat`. You can use it to get your friend list and chatrooms information.

Reference use cases:

- **pwords**, 2018, 个人微信群组关系网络图 (ego-network) [Link](#)

## Google Map API

Retrieve a JSON response from Google Map API. Here is an example: [Get the location of HKBU](#).

Once you know how to use `requests` and `json` to get the interested coordinates data, you can revisit the [city distance challenge](#) from last chapter. Then you have a fully automated solution.

**NOTE:** The above HTTP API does not work starting from Aug 2018. Instead, you can use the Google geo coder from `geopy` library. You need to apply for an API key first. Following sample code can help you.

```
from geopy.geocoders import GoogleV3
geolocator = GoogleV3(api_key='Put your Google Map API Key here')
location = geolocator.geocode("慈雲山毓華里18號慈華商場地下2-3號")
location.point
```

## Bonus: Real Estate property in Hong Kong (via government open data portal API)

Lookup real estate properties on HK gov open data portal, e.g. the [dataset page](#). Retrieve the data that `scp_mktc` contains '九龍' from year 2000 up to now. API result is like [this](#).

## Bonus: Blockchain - chain data and exchange data

[blockchain.info](#) provides a set of [API](#) for one to retrieve information related with bitcoin transactions. Pick one wallet address, check its UTXO sets and sum up the values to get the total balance in this wallet.

A [free cryptocurrency API](#) for you to retrieve and study historical exchange rates. We are interested in bitcoin price. Try to get the exchange rate day-by-day of `BTC/USDT` pair in recent years and store them in a file.

## Bonus: Automatic earthquake writer

- Implement a basic version of first automated writer - QuakeBot from LA Times.
  - Get real-time data of earthquakes in `America` from USGS API
  - Print a story to the screen include place, time, magnitude, using template string / string interpolation
  - See [here](#) for an introduction of the bot. See [here](#) for an incident and think how to avoid it?

Reference use cases:

- XU Yucan, [COMM7780/JOUR7280](#), F2018

## Bonus: MailGun API

Try to send automatic emails using [MailGun](#). Imagine you want to maintain the relationship with thousands of customers. You have the name list with email addresses. You want to write an intimate email so that every mail looks like manually tailored for the recipient. However, it is very time consuming to do this manually. You come up with an

idea based on string templating and MailGun API. MailGun is a commonly used service for receiving email and sending email. You can use string templating to format the customised message and then use MailGun to send them to the recipients.

An example of using MailGun can be found in [this repo](#).

# Week 07: Get semi-structured data - Web scraping

- [Week 07: Get semi-structured data - Web scraping](#week-07-get-semi-structured-data--web-scraping) - [A word on unified environment](#a-word-on-unified-environment) - [virtualenv, Python3 and Python2](#virtualenv-python3-and-python2) - [Jupyter notebook](#jupyter-notebook) - [Knowledge about HTML](#knowledge-about-html) - [Chrome DevTools](#chrome-devtools) - [How to use Chrome DevTools](#how-to-use-chrome-devtools) - [Frontend three: HTML, JS, and CSS](#frontend-three-html-js-and-css) - [HTML](#html) - [Scaper](#scraper) - [Basic logic](#basic-logic) - [New module: BeautifulSoup](#new-module-beautifulsoup) - [Use BeautifulSoup parser](#use-beautifulsoup-parser) - [Find data: find() and find\_all()](#find-data-find-and-find\_all) - [Get data](#get-data) - [Get title](#get-title) - [Get date](#get-date) - [Get author](#get-author) - [Method 1: Failed because of lack of specificity](#method-1-failed-because-of-lack-of-specificity) - [Method 2: Best Current Practice (BCP) -- multiple layers of element lookup (find)](#method-2-best-current-practice-bcp---multiple-layers-of-element-lookup-find) - [Method 3: Parse authors by splitting a larger text](#method-3-parse-authors-by-splitting-a-larger-text) - [Get tags](#get-tags) - [Scrape all articles of one page](#scrape-all-articles-of-one-page) - [Bonus: Scrape all articles features of all pages](#bonus-scrape-all-articles-features-of-all-pages) - [Scraper pattern](#scraper-pattern) - [Data structure](#data-structure) - [Item-first v.s. attribute first](#item-first-vs-attribute-first) - [Deal with missing data in scraping](#deal-with-missing-data-in-scraping) - [Bonus: Scrape by text processing and regular expression](#bonus-scrape-by-text-processing-and-regular-expression) - [Bonus: Crawler](#bonus-crawler) - [Workflow of a search engine like Google](#workflow-of-a-search-engine-like-google) - [Crawler is more than scraper](#crawler-is-more-than-scraper) - [Crawler is not necessary in most of your cases](#crawler-is-not-necessary-in-most-of-your-cases) - [scrapy](#scrapy) - [scrapy-cluster](#scrapy-cluster) - [Exercises and Challenges](#exercises-and-challenges) - [Scrape github users' contribution frequency](#scrape-github-users-contribution-frequency) - [Further challenge1: more users](#further-challenge1-more-users) - [Further challenge2: detailed activities](#further-challenge2-detailed-activities) - [Scrape the faculty list](#scrape-the-faculty-list) - [Scrape Haunted House in Hong Kong](#scrape-haunted-house-in-hong-kong) - [Scrape Hacker News](#scrape-hacker-news) - [Scrape Juejin](#scrape-juejin) - [Bonus: Automatic trending topic detection and posting](#bonus-automatic-trending-topic-detection-and-posting) - [Scrape Douban Top 250](#scrape-douban-top-250) - [Scrape IMDB Top 250](#scrape-imdb-top-250) - [Scrape opening journalist positions in the world](#scrape-opening-journalist-positions-in-the-world) - [Related Readings](#related-readings)

## A word on unified environment

### virtualenv, Python3 and Python2

Python3 is the de facto standard in Python now (year 2018). The community spent [around 10 years](#) efforts before the widespread of Python3. The standard and interpreter core are out in the market for a very long time but many useful libraries were waiting to support Python3. People have maintained a [wall of shame](#), which is later changed to "wall of superpowers", to track the progress of moving to Python3. It is all because Python3 is not a backward compatible upgrade to Python2. Past codes will break when we upgrade from Python2 to Python3.

**TIP:** If you start a new project now, use Python3, [Python3](#), [Python3](#).

However, in some places, especially older systems/ environments, you still have Python2, or dependencies on Python2. To make things uncluttered, please setup virtualenv and your Jupyter notebook in virtualenv.

### Jupyter notebook

Windows users and Mac OS X users suffer different environment setup problems in chapter 1-3. Starting from chapter 4, all our works will be based on virtualenv and Jupyter notebook. So everyone will see the same input/ output in the future. -- A big cheers! -- you have already passed through the most difficult part of the whole course.

You can refer to [FAQ on Jupyter](#) for more information.

# Knowledge about HTML

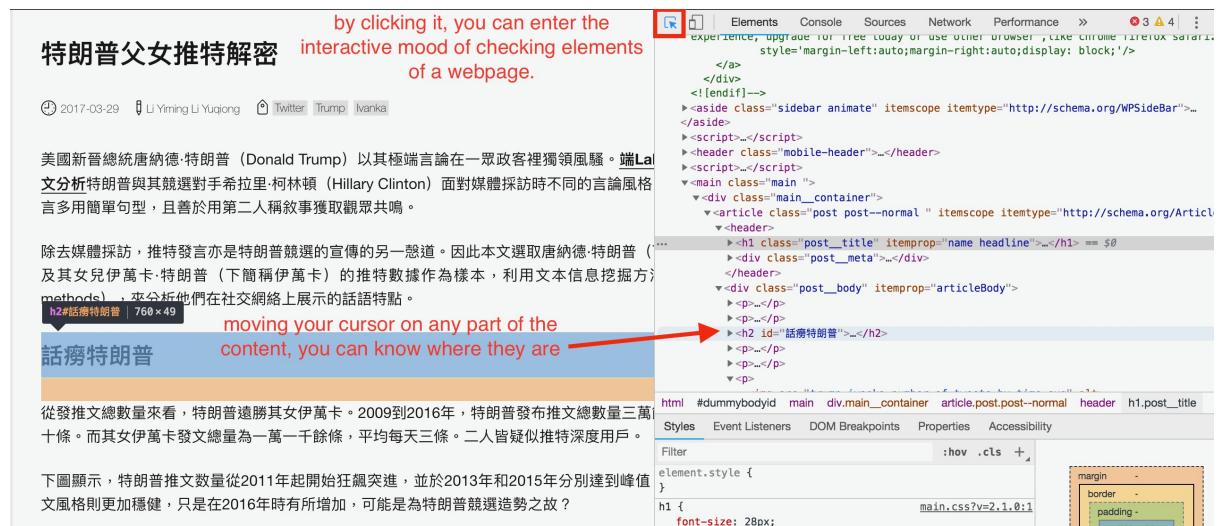
## Chrome DevTools

Chrome DevTools is a set of web developer tools built directly into the Google Chrome browser. For us, Chrome DevTools can help us better learn the basics of viewing and even changing a page's code, with what we can understand the structure of a webpage better, how a website store the data, present the information, and most importantly, how we locate/find those information we want and retrieve them into structural data to process further analysis.

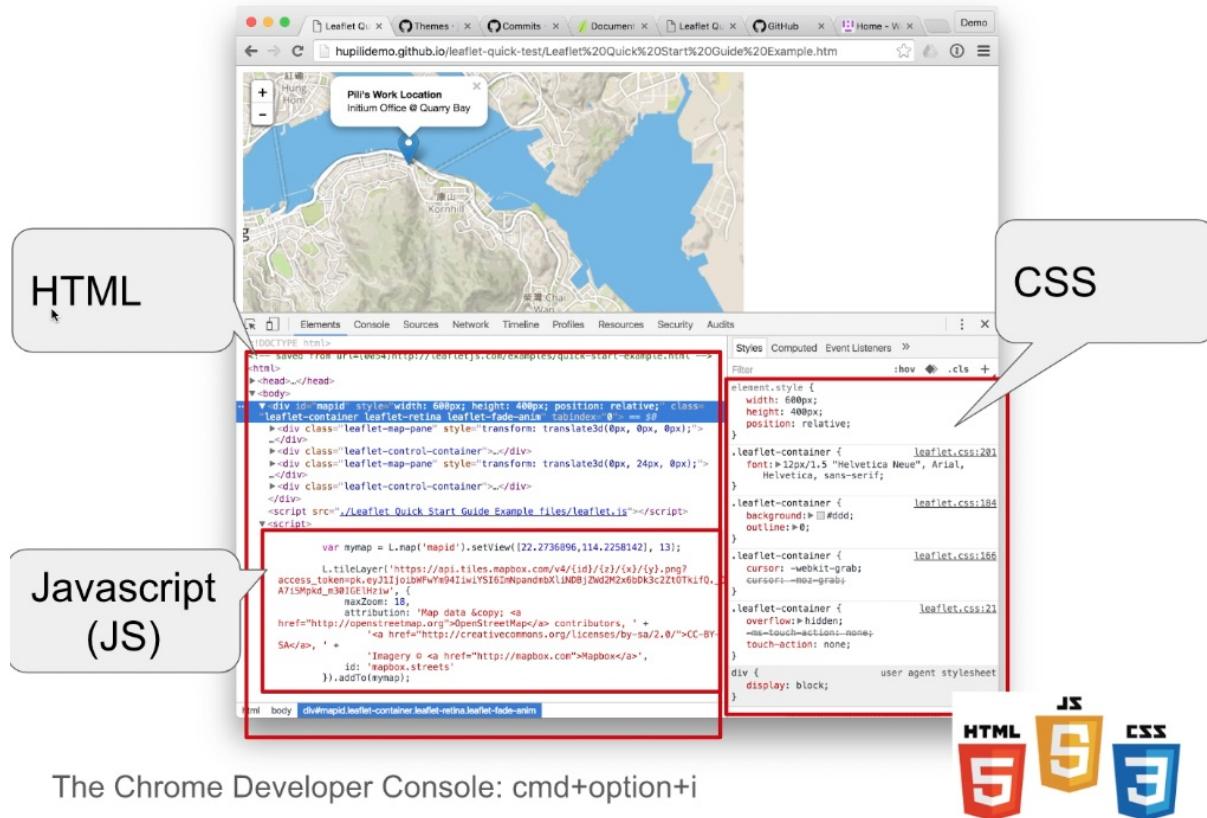
## How to use Chrome DevTools

1. Firstly, It is suggested to use 'Chrome' as our browser.
  2. In Chrome, `option+command+i` to open the Chrome DevTools, a.k.a Chrome developer console.
  3. Click the upper left corner of the console, you can select an element in the webpage to inspect it. You will see its source code by moving your cursor on to it.

Eg: Check out the structure of a webpage, a project about tweets of Trump <https://initiumlab.com/blog/20170329-trump-and-ivanka/>. For example, moving your cursor to check out every `h2` headline.



## Frontend three: HTML, JS, and CSS



The Chrome Developer Console: cmd+option+ cmd+option+

- HTML is a machine language of web page. Writing something in HTML means to create a web page. It is a structure of diverse tags. Those tags are in pairs, with open tag and closing tag that wrap up content we want to present. Like `<p> content </p>`.
- CSS stands for Cascading Style Sheets. CSS describes how HTML elements are to be displayed on screen, paper, or in other media.
- JavaScript is the programming language of HTML and the Web, which is mainly used for image manipulation, form validation, and dynamic changes of content.

## HTML

HTML is a "declarative language", whereas Python is "imperative language" (in short, loose terms). The key difference here is that declarative language does not instruct the machine how to solve a problem/ present result in a step-by-step manner. Instead, it tells the machine what the desired output is and it is subject to the machine how to generate the output in its own ways.

The whole web lives on HTML so you can find numerous free online resources for further study of HTML, e.g. [here](#). Our major objective here is not to teach one to write web pages (frontend development). We emphasize on understanding the HTML pages and parsing useful **structured** data out of the pages. Here are the core concepts on HTML:

- It is a language based on tags. Common tags can be `p`, `h1`, `h2`, `h3`, `ul`, `ol`, `li`, `img`, `a`, ...
- Tags come in paired and nested manners:
  - Paired -- if you see `<p>`, there must be a `</p>` later on. The document looks like

```

<p>
 ... content is here ...
</p>

```

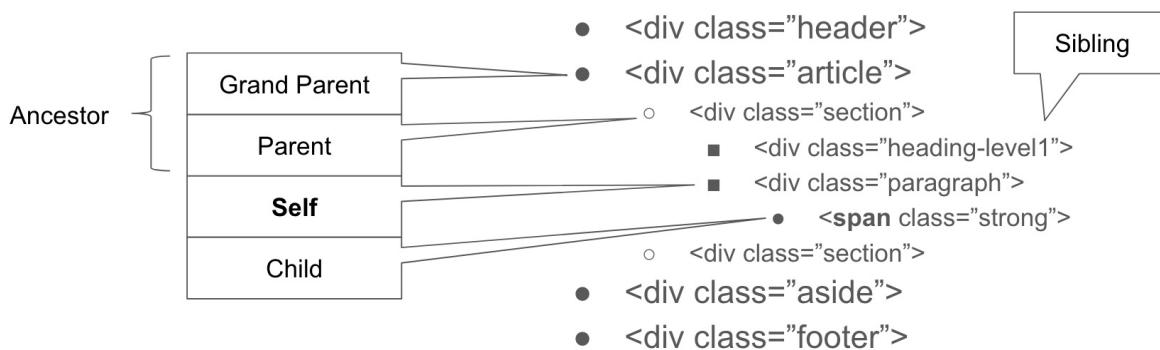
- Nested -- imagine a list of shopping items on the page, and each item is an image link, that leads you to detailed page when clicked. The structure looks like

```



```

In this way, one can build a complex structure of pages. The inner tags are called "children" of the out tags. One can picture in mind that the HTML document is organised in **tree** like structure, where `<html></html>` is the "root" and every other tag is a "branch" who has child tag and parent tag. One may note that `img` tag does not come in pairs in above example. That is because you can not place any content inside `img`. It is a "leaf" and we don't enclose further tags inside it. Towards this end, we can give a shorthand omitting the closing tag `</img>`. It was once a good practice to make the tags come in paired format. Since the inception of HTML5, it was suggested to leave "leaf" tags the way it naturally should be -- i.e. no pairing closing tag and no content inside.



(an illustration of HTML tree structure)

For the ease of discussion, we also call "HTML tag" as "HTML element" or "HTML node" interchangeably.

## Scraper

### Basic logic

Before we try to get data, here is the logic we should know. Basically, when we scrape a website, firstly we need to know the website.

1. Does the website provide its own API to get the data?
2. If no, we will scrape the data from its html.
3. All data or information is stored in the html tags. Tag names are settled by the website creators, which always appears as pairs. So, all we need to do is to find the tags that contains our required data. For example, the article titles are usually in `h1`, and texts are usually in `p`. You can find those tags by using Chrome DevTools, which we talked about this at the beginning of the chapter.
4. After we find the data and the tags, we write code to get them (using `control flows`), clean them (`manipulating strings`, `strip()`, `replace()` ...), and store them into files (`csv`, `JSON`).

### New module: BeautifulSoup

`bs4` is the abbreviation of BeautifulSoup4. Beautiful Soup is a Python library for pulling data out of HTML and XML files.

In other words, BeautifulSoup parses the HTML content you request into structural data so that we can easily find the element we want.

## Install BeautifulSoup

```
!pip3 install --user bs4
```

## Import the module

```
from bs4 import BeautifulSoup
```

## Use BeautifulSoup parser

BeautifulSoup parser can convert the results we request into structural data so that we can easily find the data we want.

Eg: <https://initiumlab.com/blog/20170329-trump-and-ivanka/>

```
import requests
from bs4 import BeautifulSoup
r = requests.get('https://initiumlab.com/blog/20170329-trump-and-ivanka/')
#print(r) you will get <Response [200]> means request successful
html_str = r.text #get the content of the request
```

- Store the web as `r`, `get()` means try to get response of that web page, passing url string in to `()`.
  - `text` means to show the text of the web page.

Output: This is before the parsing step, you can see that they are like a mess.

Output: After parsing, you can see that the data is more structural, and we can further get/find the data by using control flows and manipulating of `if` and `for`.

```
In [7]: data
Out[7]:
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8"/>
<link href="https://assets-cdn.github.com" rel="dns-prefetch"/>
<link href="https://avatars0.githubusercontent.com" rel="dns-prefetch"/>
<link href="https://avatars1.githubusercontent.com" rel="dns-prefetch"/>
<link href="https://avatars2.githubusercontent.com" rel="dns-prefetch"/>
<link href="https://avatars3.githubusercontent.com" rel="dns-prefetch"/>
<link href="https://github-cloud.s3.amazonaws.com" rel="dns-prefetch"/>
<link href="https://user-images.githubusercontent.com/" rel="dns-prefetch"/>
<link crossorigin="anonymous" href="https://assets-cdn.github.com/assets/frameworks-95aff0b550d3fe338b645a4deebdc1b.css" integrity="sha512-Z0JAr9+DkI1NjGVdZr3GivARUgJtA0o2eHlTv7Ou2gshR5awWVf8QGs11Ns9ZxQLEs+G5/SuARmvpOLMzulw==" media="all" rel="stylesheet">
<link crossorigin="anonymous" href="https://assets-cdn.github.com/assets/github-1b92fc57a93ac751f875b024fc0646d2.css" integrity="sha512-eeb0wTQqP0vhEgOFWGwR3lZg7k8NoNU3aJm5fyWM7vaS1m1ZgpLIlXr0J82rnHRY1pkDXlw034JzTj9YfJQ==" media="all" rel="stylesheet">
```

## Find data: find() and find\_all()

- `find` to find what we want, and output the first item. Like if there are 10 h1, they will return the first one.
- `find_all` return a list of all the values we want. Like if there are 10 h1, they will return a list that contain all those h1. **A list means that we can use for loop to further filter.**

Example:

`html_doc` is as following:

```
html_doc = """
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title">The Dormouse's story</p>

<p class="story">Once upon a time there were three little sisters; and their names were
Elsie,
Lacie and
Tillie;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
"""

from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc, 'html.parser')
my_a = soup.find('a') #find a
my_a
```

Output:

```
Elsie

my_a = soup.find_all('a') #find all a
my_a
```

Output:

```
[Elsie,
 Lacie,
 Tillie]

my_a = soup.find('a', attrs={'id': 'link3'}) #find link3
```

Note: You can see that in tag a, there are some attributes, like class, id. Those attributes are used to distinguish this tag from other similar tags, especially when there are many tags in the html page. So, if you want to locate or find sth. more precisely. You can find those attributes specifically by writing it as

```
soup.find('tag_name', attrs={'attributes': 'values'})
```

## Output:

```
Tillie
```

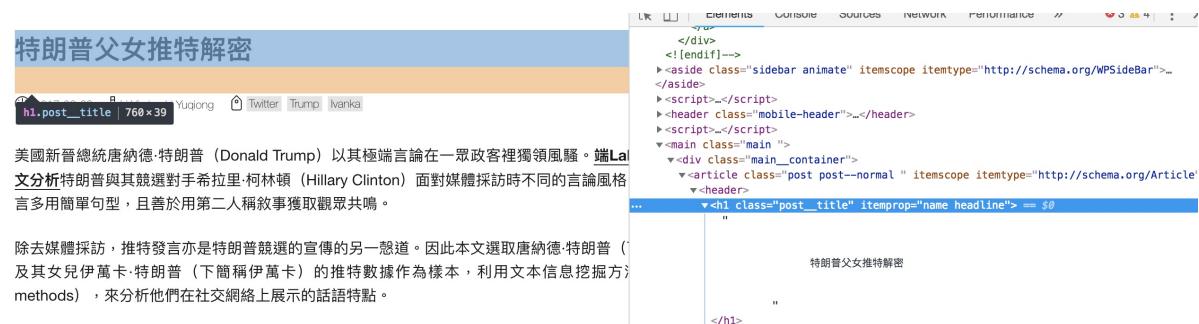
Basically, parser and find functions are the most used of `BeautifulSoup` library for us, if you want to know more functions and syntax of using it, please check out [here](#).

## Get data

## Get title

Open the chrome devtools, by moving the mouse on the headline, you can find title is in:

**<h1 class="post\_title" itemprop="name headline"> 特朗普父女推特解密</h1>**



```
import requests
import csv
from bs4 import BeautifulSoup
r = requests.get('https://initiumlab.com/blog/20170329-trump-and-ivanka/')
html_str = r.text
data = BeautifulSoup(html_str,"html.parser")
my_h1 = data.find('h1') # we use tag and attributes to extract the data we want. Type(my_
my_h1` is bs4.element.Tag
my_title = my_h1.text #turn bs4.element.Tag into pure text
my_title.strip() # remove the character specified at the beginning and end of the string
```

Output: You can learn the logic and function of each step.

```
In [6]: myhl=data.find('h1')

In [11]: myhl

Out[11]: <h1 class="post__title" itemprop="name headline">

 特朗普父女推特解密

</h1>

In [12]: myhl.text

Out[12]: '\n \n \n 特朗普父女推特

In [13]: myhl.text.strip()

Out[13]: '特朗普父女推特解密'
```

### Note:-

- `strip()` means delete the meaningless character at the beginning and end of the string.
- `HTML/bs4_tag.text` means turn `bs4.element.Tag` into pure text.
- You can `help(str.strip)` to see the usage of `strip`.
- `type(sth)` is to print what is the format of `sth`. It's useful because you should know what's the data it return to further extract the value we want. Like, if it is a list, we should first use index to access its value. Similarly, if it is a dict, we should use keys to access its value.

## Get date

In the same way we use to find the title, we can find that time tag as follows:

```
<time itemprop="dateCreated" datetime="2017-03-29T....." content="2017-03-29">
 2017-03-29
</time>
```

Extract time value:

```
my_date = data.find('time').text.strip()
```

Output:

```
In [17]: mytime=data.find('time')

In [18]: mytime
Out[18]: <time content="2017-03-29" datetime="2017-03-29T18:00:00+08:00" itemprop="dateCreated">
 2017-03-29
</time>

In [19]: mytime.text.strip()
Out[19]: '2017-03-29'
```

## Get author

### 特朗普父女推特解密



美國新晉總統唐納德·特朗普（Donald Trump）以其極端言論在一眾政客裡獨領風騷。[端](#)[文](#)[分析](#)特朗普與其競選對手希拉里·柯林頓（Hillary Clinton）面對媒體採訪時不同的言論風格，言多用簡單句型，且善於用第二人稱敘事獲取觀眾共鳴。

除去媒體採訪，推特發言亦是特朗普競選的宣傳的另一懸道。因此本文選取唐納德·特朗普（及其女兒伊萬卡·特朗普（下簡稱伊萬卡）的推特數據作為樣本，利用文本信息挖掘方



```
> <h1 class="post__title" itemprop="name headline">.
> <div class="post__meta">
> <table>
> <tbody>
> <tr class="post__time">
> <td>....</td>
> <td>....</td>
> </tr>
> <tr class="post__authors">
> <td>
> Li Yiming == $0
> Li Yuqiong
> </td>
> </tr>
> <tr class="post__tags">....</tr>
> </tbody>
> </table>
> </div>
> <div class="post__content">....</div>
> <div class="post__comment">....</div>
> <div class="post__share">....</div>
```

You can find that authors are in the span, so could we just use `find.span` to get the authors?

### Method 1: Failed because of lack of specificity

```
my_authors = data.find('span')
```

Output:

```
首頁
```

It is not what we want, the reasonable guess is that there are many 'span'. So check how many span there, and find the difference between those tags. `command+f` to open the search bar in console, and input 'span'. You can see, there are more than 2 'span'. Therefore we should find all the span.

```
my_spans = data.find_all('span')
my_spans
```

Output:

```
[首頁,
 文章,
 專題,
 活動,
 職位,
 團隊,
 訂閱,
 Li Yiming,
 Li Yuqiong,

 ...
香港北角英皇道 663 號泓富產業千禧廣場 1907 室]
```

You can see that `find_all` returns a list and there are so many spans, and the authors are also in two of those spans. So how we get them? We use `index` to access them.

```
authors = []
author_1 = my_span[7].text
author_2 = my_span[8].text
authors.append(author_1) #append them into a list
authors.append(author_2)
```

**Note:** Why do we just use `authors = my_span.find[7:9].text` to find all authors? Because `find[7:9]` or `find_all` return a list of elements, however, if you use `().text` function in a list of elements, it will raise error, the list has no attribute 'text', which means it cannot be converted to text directly in this way.

**TIP:** "specificity" issue is quite common in writing scraper. There are usually many ways, and usually easy, to find the element(s) we are interested in. One needs to work hard to ensure, the scraper does not pollute the result by other elements that we are **not interested** in.

### Method 2: Best Current Practice (BCP) -- multiple layers of element lookup (find)

```
<header>
 <h1 class="post_title" itemprop="name headline">...</h1>
 <div class="post_meta">
 <table>
 <tbody>
 <tr class="post_time">...</tr>
 <tr class="post_authors">
 <td>...</td>
 <td>
 Li Yiming
 Li Yuqiong == $0
 </td>
 </tr>
 <tr class="post_tags">...</tr>
 </tbody>
 </table>
 </div>
</header>
```

- The logic here is if we can not specify one elements in the inner circle, we spread out to find the differentiate tag that only the element has.
- In the HTML, we can find that authors upper tag is 'td'. But there are too many td. And it is difficult to be specific. So, we spread out.
- Outer the `td` is the `tr` tag with a class named `post_authors`, you can find its the special tag only used to wrap authors, so try to locate and extract author names by tag: `tr`. Once we find the `tr`, we wan further find `td`, then find all `span` in `td`, that's what we want.

```
my_authors = []
post_authors = data.find('tr', attrs={'class':'post__authors'})
#pay attention to its syntax, find('tag_name,attributes={'key':'value'})
my_td = post_authors.find_all('td')
my_span = my_td[1].find_all('span')
for span in my_span:
 my_authors.append(span.text)
my_authors
```

Output:

```
['Li Yiming', 'Li Yuqiong']
```

### Method 3: Parse authors by splitting a larger text

```
my_authors = data.find('tr', attrs={'class':'post__authors'}).text.strip().replace('\n', ',') #after .text, we go
t the author names with several characters, we can further use strip and replace to omit those meaning less cha
racters.
```

Output:

```
In [26]: my_authors = data.find('tr', attrs={'class':'post__authors'})
```

```
In [31]: my_authors.text.strip()
```

```
Out[31]: 'Li Yiming\nLi Yuqiong'
```

```
In [32]: my_authors.text.strip().replace('\n', ',')
```

```
Out[32]: 'Li Yiming,Li Yuqiong'
```

- Syntax: `find('tag_name,attributes={'key':'value'})`
- `attrs = attributes`. It contains more detailed information about about HTML tags, which helps to locate and identify the values better.
- `replace('a', 'b')` means replace a as b. You can see that even after `strip()`, there is a `\n` in lines, in such circumstances, we can use replace to get off those characters.

**TIP:** Please compare this method to the previous method. The best practice is to refrain from text processing if possible, especially in the upstream (earlier stage) of data processing pipeline. Method 3 looks simpler by a glance but Method 2 is more stable, especially in the long run, when more people join force to maintain one set of codes. The major drawback of splitting text is that the delimiter (`\n` here) may also appear as part of the text content. It is unlikely the case in our current example but have caused trouble to many students in other scenarios.

## Get tags

```
my_tags = data.find('tr', attrs={'class':'post__tags'}).text.strip().replace(' ', '').replace('\n\n\n\n\n', '|')
#you can find that there are several blanks and escape characters in return my_tags. We can use replace to get
off those meaningless characters.
```

## Scrape all articles of one page

If you want to scrape more articles, you will find there are some repeatable work and logic, so it's better for us to define a function to scrape more articles. Aside of this, all we need to do is find all articles url so that we can use a `for loop` to scrape more articles, right?

Step 1: Based on what we do in the above example, we can define a function like the following.

```
def scrape_one_article(article_url):
 r = requests.get(article_url).text
 data = BeautifulSoup(r, "html.parser")
 my_title = data.find('h1').text.strip()
 my_date = data.find('time').text.strip()
 my_authors = data.find('tr', attrs={'class':'post__authors'}).text.strip().replace('\n', ',')
 my_tags = data.find('tr', attrs={'class':'post__tags'}).text.strip().replace(' ', '|').replace('\n\n\n\n\n', '|')
)
 my_url = article_url
 return my_title, my_authors, my_date, my_tags, my_url
```

Step 2: Get all the urls of one page. You can click '文章' to get into the articles page.

The screenshot shows a blog post titled '特朗普父女推特解密'. The sidebar on the left has a navigation menu with '文章' highlighted in red. The main content area includes the author information (Li Yiming Li Yuqiong), publication date (2017-03-29), and social sharing icons. The text discusses Donald Trump's speech patterns compared to Hillary Clinton during interviews.

The screenshot shows another blog post titled '「消失的檔案」：香港開放數據大檢討'. The sidebar on the left has a navigation menu with '文章' highlighted in red. The main content area includes the author information (Ren Xinya), publication date (2017-04-07), and social sharing icons. The text discusses the state of data openness in Hong Kong.

You can see that there are many articles in this page. So how can we scrape all the articles features including authors, dates, headlines? Firstly, we should get all urls of those articles. Therefore, we define another function to get those urls.

```
def scrape_articles_urls_of_one_page(article_page_url): #scrape_articles_urls_of_one_page
 article_urls = []
 r = requests.get(article_page_url).text
 data = BeautifulSoup(r, "html.parser")
 my_urls = data.find_all('a', attrs={'class': 'post__title-link js-read-more'}) #find the links

 #quiz1: if you read the code of this page, most students will try to find ('a', attrs={'class': 'post__title-link'}) first and failed. Do you know why?

 #quiz2: you will find that url can be extracted by my_url['href'], the results will be like this: '../blog/20160908-taipei-power-usage/', but the real one should be like this 'http://initiumlab.com/blog/20160908-taipei-power-usage/',
 #so who do we format those links we want?

 for my_url in my_urls:
 url = '{0}{1}'.format('http://initiumlab.com', my_url['href'][2:]) #format urls
 #print(url)
 article_urls.append(url)

 return article_urls
scrape_articles_urls_of_one_page('http://initiumlab.com/blog/')
```

Output:

```
['http://initiumlab.com/blog/20170407-open-data-hk/',
 'http://initiumlab.com/blog/20170401-data-news/',
 'http://initiumlab.com/blog/20170329-trump-and-ivanka/',
 'http://initiumlab.com/blog/20170324-hk-odd/',
 'http://initiumlab.com/blog/20170315-news-tool/',
 'http://initiumlab.com/blog/20170312-soma-post/',
 'http://initiumlab.com/blog/20170222-new-media/',
 'http://initiumlab.com/blog/20170113-Sharing-With-Friends-Versus-Strangers/',
 'http://initiumlab.com/blog/20161229-Facebook-App-Download-Conversion/',
 'http://initiumlab.com/blog/20160908-taipei-power-usage/']
```

After we get all the articles urls of one page, you can call the `scrape_one_article(article_url)` function to scrape all the features of this page.

```
articles = []
article_urls = scrape_articles_urls_of_one_page('http://initiumlab.com/blog/') #scrape_articles_urls_of_one_page1
for article_url in article_urls:
 articles.append(scrape_one_article(article_url))

with open('articles.csv', 'w', newline='') as f:
 writer = csv.writer(f)
 header = ['titles', 'Authors', 'Dates']
 writer.writerow(header)
 writer.writerows(articles)
```

Output:

articles				
Titles	Authors	Dates	Tags	Urls
「消失的檔案」：香港開放數據大檢討	Ren Xinya	2017-04-07	HongKong OpenData	<a href="http://initiumlab.com/blog/20170407-open-data-hk/">http://initiumlab.com/blog/20170407-open-data-hk/</a>
數據新聞的精選 Feed 列表	Chia ni Liu	2017-04-01	DataJournalism DataVisualisation RSS Feed	<a href="http://initiumlab.com/blog/20170401-data-news/">http://initiumlab.com/blog/20170401-data-news/</a>
特朗普父女推特解密	Li Yiming,Li Yuqiong	2017-03-29	Twitter Trump Ivanka	<a href="http://initiumlab.com/blog/20170329-trump-and-iwanka/">http://initiumlab.com/blog/20170329-trump-and-iwanka/</a>
2017 香港開放數據黑客松成果紀錄	Chia ni Liu	2017-03-24	Hackathon OpenData	<a href="http://initiumlab.com/blog/20170324-hk-odd/">http://initiumlab.com/blog/20170324-hk-odd/</a>
編輯必備！4個助你挖掘價值訊息的選題工具	Hu Pili, Ren Xinya	2017-03-15	NewsTool Editor TopicSelection	<a href="http://initiumlab.com/blog/20170315-news-tool/">http://initiumlab.com/blog/20170315-news-tool/</a>
哥大資料視覺化大師 Soma 在浸會大學的密技分享	Chia ni Liu	2017-03-12	Design DataJournalism JonathanSoma	<a href="http://initiumlab.com/blog/20170312-soma-post/">http://initiumlab.com/blog/20170312-soma-post/</a>
新出媒體知多少？	Ren Xinya	2017-02-22	newmedia	<a href="http://initiumlab.com/blog/20170222-new-media/">http://initiumlab.com/blog/20170222-new-media/</a>
俗話說「好事不出門、壞事傳千里」，真的是這樣嗎？	Ren Xinya	2017-01-13	InterpersonalCloseness Word-of-mouthValence SocialMedia	<a href="http://initiumlab.com/blog/20170113-Sharing-With-Friends-Versus-Strangers/">http://initiumlab.com/blog/20170113-Sharing-With-Friends-Versus-Strangers/</a>
Facebook App Download Conversion	Initium Lab	2016-12-29	lang-English DataAnalytics GrowthHacking	<a href="http://initiumlab.com/blog/20161229-Facebook-App-Download-Conversion/">http://initiumlab.com/blog/20161229-Facebook-App-Download-Conversion/</a>
省電不差我一個嗎？台北市一級發布區用電量比較	Tianchung Lin	2016-09-08	powerusage Taipei	<a href="http://initiumlab.com/blog/20160908-taipei-power-usage/">http://initiumlab.com/blog/20160908-taipei-power-usage/</a>

## Bonus: Scrape all articles features of all pages

Since we scrape one page of articles, can I scrape all articles of all pages? Of course! we just come from 0 to 1, next step is from 1 to n. But there are some difficulties on the way which might be a little bit difficult for us, but definitely we can solve this.

Potential challenges:

- Scrape articles urls from different pages. Because the format of articles urls changes in different pages, plus the articles urls are not directly what we want, which need us further construct those urls.
- Format all pages urls.
- Get straight/understanding with 3 layers structure.

```
import requests
from bs4 import BeautifulSoup
import csv

def scrape_one_article(article_url): #scrape one articles features, which we've already done this
 r = requests.get(article_url).text
 data = BeautifulSoup(r,"html.parser")
 my_title = data.find('h1').text.strip()
 my_date = data.find('time').text.strip()
 my_authors = data.find('tr', attrs={'class':'post__authors'}).text.strip().replace('\n',' ')
 my_tags = data.find('tr', attrs={'class':'post__tags'}).text.strip().replace(' ','').replace('\n\n\n\n\n','')
)
 my_url = article_url
 return my_title,my_authors,my_date,my_tags,my_url

def scrape_articles_urls_of_one_page(article_page_url): #scrape_articles_urls_of_one_page, its a little bit different from demo above because in the following pages(2-7), the articles' urls are different...
 article_urls = []
 r = requests.get(article_page_url).text
 data = BeautifulSoup(r,"html.parser")
 my_urls = data.find_all('a', attrs={'class':'post__title-link js-read-more'})
 for my_url in my_urls:
 url ='{0}blog{1}'.format('http://initiumlab.com/',my_url['href'].split('/blog')[-1]) #format urls.
 # Fail try 1 : use slice to cut off .../...
 # Fail try 2 : use blog instead of /blog to split. There are blog in the headline
 #print(url)
 article_urls.append(url)
 return article_urls

def scrape_all_pages(url):
 articles=[]
 for i in range(1,8): #format all pages urls
 if i == 1:
 page_url = url
 else:
 page_url = '{url_initial}page/{number}/'.format(url_initial = url,number=i)
 #print(page_url)

 article_urls = scrape_articles_urls_of_one_page(page_url)
 for article_url in article_urls:
 articles.append(scrape_one_article(article_url))
```

```

return(articles)

with open('initiumlab_articles.csv', 'w', newline='') as f:
 all_articles = scrape_all_pages('http://initiumlab.com/blog/')
 writer = csv.writer(f)
 header = ['Titles', 'Authors', 'Dates', 'Tags', 'Url']
 writer.writerow(header)
 writer.writerows(all_articles)

```

Output will be like the following picture, and you can also find the csv file [here](#).

initiumlab_articles				
Titles	Authors	Dates	Tags	Url
「消失的檔案」：香港開放數據大檢討	Ren Xinya	2017-04-07	HongKong OpenData	<a href="http://initiumlab.com/blog/20170407-open-data-hk/">http://initiumlab.com/blog/20170407-open-data-hk/</a>
數據新聞的精選 Feed 表單	Chia ni Liu	2017-04-01	Data.Journalism DataVisualisation RSS Feed	<a href="http://initiumlab.com/blog/20170401-data-news/">http://initiumlab.com/blog/20170401-data-news/</a>
特朗普父女推特解密	Li Yiming,Li Yuqiong	2017-03-29	Twitter Trump Jinlka	<a href="http://initiumlab.com/blog/20170329-trump-and-ivanka/">http://initiumlab.com/blog/20170329-trump-and-ivanka/</a>
2017 香港開放數據黑名單成績紀錄	Chia ni Liu	2017-03-24	Hackathon OpenData	<a href="http://initiumlab.com/blog/20170324-hk-odd/">http://initiumlab.com/blog/20170324-hk-odd/</a>
編輯必備！4 個助你挖掘資訊訊息的選題工具	Hu Pili, Ren Xinya	2017-03-15	NewsTool Editor TopicSelection	<a href="http://initiumlab.com/blog/20170315-news-tool/">http://initiumlab.com/blog/20170315-news-tool/</a>
葛大資訊視覺化大師 Soma 在漫書會的密技分享	Chia ni Liu	2017-03-12	Design Data.Journalism JonathanSoma	<a href="http://initiumlab.com/blog/20170312-soma-post/">http://initiumlab.com/blog/20170312-soma-post/</a>
新出媒體知多少？	Ren Xinya	2017-02-22	newmedia	<a href="http://initiumlab.com/blog/20170222-new-media/">http://initiumlab.com/blog/20170222-new-media/</a>
俗話說「好事不出門、壞事傳千里」，真的是這樣嗎？	Ren Xinya	2017-01-13	InterpersonalCloseness Word-of-mouthValence SocialMedia	<a href="http://initiumlab.com/blog/20170113-Sharing-With-Friends-Versus-Strangers/">http://initiumlab.com/blog/20170113-Sharing-With-Friends-Versus-Strangers/</a>
Facebook App Download Conversion	Initium Lab	2016-12-29	lang-English DataAnalytics GrowthHacking	<a href="http://initiumlab.com/blog/20161229-Facebook-App-Download-Conversion/">http://initiumlab.com/blog/20161229-Facebook-App-Download-Conversion/</a>
省電不讓我一個曉？台北市一級節能布區用電量比較	Tianchung Lin	2016-09-19	powerusage Taipei	<a href="http://initiumlab.com/blog/20160908-taipei-power-usage/">http://initiumlab.com/blog/20160908-taipei-power-usage/</a>
五強圍剿：香港立法會選舉地區直選議席分配是否合理？	黃凌輝	2016-09-03	香港立法會 議席分配	<a href="http://initiumlab.com/blog/20160903-lego-seats-analysis/">http://initiumlab.com/blog/20160903-lego-seats-analysis/</a>
過去四年，立法會議員都討論了什麼？	Initium Lab	2016-09-01	香港立法會 PCA 議案分析	<a href="http://initiumlab.com/blog/20160901-lego-5-motion-analysis/">http://initiumlab.com/blog/20160901-lego-5-motion-analysis/</a>
立法會選舉的年代之謎：數據指我們不一定投建制	Tom Tsang	2016-08-24	香港立法會 HongKong Election	<a href="http://initiumlab.com/blog/20160824-lego-voters-young-vs-old/">http://initiumlab.com/blog/20160824-lego-voters-young-vs-old/</a>
「腦海術」，佔領英美課題？——關於中式數學教育的幾個事實	Cheng Yixiang	2016-08-15	MathOlympics Education	<a href="http://initiumlab.com/blog/20160815-math-olympics/">http://initiumlab.com/blog/20160815-math-olympics/</a>
開放陸客赴台，五年來改變了什麼？	趙安平	2016-08-12	lang-Chinese Taiwan>MainlandStudent	<a href="http://initiumlab.com/blog/20160812-tw-mainland-student/">http://initiumlab.com/blog/20160812-tw-mainland-student/</a>
2張圖看香港過去50年漲價最多是哪個商品	Andy Lin	2016-08-10	HongKong ConsumerPriceIndex CPI	<a href="http://initiumlab.com/blog/20160810-hk-consumer-price/">http://initiumlab.com/blog/20160810-hk-consumer-price/</a>
數字看「開放陸生赴台就讀」的五年	趙安平	2016-08-08	lang-Chinese Taiwan>MainlandStudent	<a href="http://initiumlab.com/blog/20160808-tw-mainland-student/">http://initiumlab.com/blog/20160808-tw-mainland-student/</a>
香港房價比台北貴多少？3張圖看兩座城市的居住條件	Andy Lin	2016-08-03	RjEconomics Housing	<a href="http://initiumlab.com/blog/20160803-hk-house-price/">http://initiumlab.com/blog/20160803-hk-house-price/</a>
一張地圖帶你「搵錢」：香港年輕選民	Vincent Lau	2016-07-31	香港立法會 HongKong Election	<a href="http://initiumlab.com/blog/20160731-lego-young-electors/">http://initiumlab.com/blog/20160731-lego-young-electors/</a>
23萬投票紀錄 回顧第三屆香港立法會	Initium Lab	2016-07-30	香港立法會 投票分析 PCA	<a href="http://initiumlab.com/blog/20160730-Voting-Preference-Analysis-for-Hong-Kong-Legislative-Council-2012-2016/">http://initiumlab.com/blog/20160730-Voting-Preference-Analysis-for-Hong-Kong-Legislative-Council-2012-2016/</a>
Adopt MediaWiki As Our Internal Knowledge Management System	Chunlinling Lyu	2016-07-30	Tech lang-English MediaWiki KnowledgeManagement	<a href="http://initiumlab.com/blog/20160730-Tech-lang-English-MediaWiki-wiki-knowledge-management-system/">http://initiumlab.com/blog/20160730-Tech-lang-English-MediaWiki-wiki-knowledge-management-system/</a>
奧運冠軍的後奥运生活	沈雲峰	2016-07-28	Olympics Sports RunnersSystem ChineseAthletes	<a href="http://initiumlab.com/blog/20160728-chinese-olympics-champions/">http://initiumlab.com/blog/20160728-chinese-olympics-champions/</a>
香港房價年齡地圖：帶你看3萬樓私人大廈的建築年份	Andy Lin	2016-07-25	Economics HouseAge HongKong	<a href="http://initiumlab.com/blog/20160725-hk-house-age/">http://initiumlab.com/blog/20160725-hk-house-age/</a>
射電望遠鏡你知道多少	黃杜焜	2016-07-24	one-pc infographics science	<a href="http://initiumlab.com/blog/20160724-radio-telescope/">http://initiumlab.com/blog/20160724-radio-telescope/</a>
台灣房價分析：找出最近四年房價最慢的地方	Andy Lin	2016-07-18	Economics HousePrice Taiwan	<a href="http://initiumlab.com/blog/20160718-taiwan-house-price/">http://initiumlab.com/blog/20160718-taiwan-house-price/</a>
職業運動員究竟能拿多少薪水？	Cheng Yixiang	2016-07-14	one-pc sports infographics	<a href="http://initiumlab.com/blog/20160714-top-player-salary/">http://initiumlab.com/blog/20160714-top-player-salary/</a>
世界主要貨幣十年走勢分析	Andy Lin	2016-07-14	RjCurrency Economics	<a href="http://initiumlab.com/blog/20160708-currency-exchange-rate-trend/">http://initiumlab.com/blog/20160708-currency-exchange-rate-trend/</a>
希拉里與特朗普的話語玄機	Cheng Yixiang	2016-07-04	TextMining Politics USA R	<a href="http://initiumlab.com/blog/20160704-trump-hillary-language/">http://initiumlab.com/blog/20160704-trump-hillary-language/</a>
Orlando Shooting: What did Americans say on Twitter?	Cheng Yixiang	2016-06-22	Tableau RjTwitter SentimentAnalysis	<a href="http://initiumlab.com/blog/20160622-orlando-gunsnot/">http://initiumlab.com/blog/20160622-orlando-gunsnot/</a>
從數據看全球海洋漁業二十年	Shen Shuyuan, Cheng Yixiang	2016-06-22	Data Hackathon Tableau	<a href="http://initiumlab.com/blog/20160622-global-fishing-20years/">http://initiumlab.com/blog/20160622-global-fishing-20years/</a>
A Small Tweak Doubles Ad CTR	Zeng Xi, Shen Shuyuan	2016-06-15	SideBar AdCTR Optimization	<a href="http://initiumlab.com/blog/20160615-sidebars-improvement/">http://initiumlab.com/blog/20160615-sidebars-improvement/</a>
統計的語言	Xia Mengyao, Cheng Yixiang	2016-06-13	BookReview Statistics	<a href="http://initiumlab.com/blog/20160613-how-to-lie-with-statistics-bookreview/">http://initiumlab.com/blog/20160613-how-to-lie-with-statistics-bookreview/</a>
「支援中西部地區招生協作計劃」令高等教育更加公平？	Cheng Yixiang	2016-06-08	Data CollegeEntranceExamination EducationFairness	<a href="http://initiumlab.com/blog/20160608-college-enrollment-plan/">http://initiumlab.com/blog/20160608-college-enrollment-plan/</a>
Map Visualisation for Panama Papers and Offshore Leaks in R	Charlie Chen, Chao Tianyi	2016-05-29	lang-English Map PanamaPapers	<a href="http://initiumlab.com/blog/20160529-panama-papers-on-map/">http://initiumlab.com/blog/20160529-panama-papers-on-map/</a>
Reflection on GitHub's New Pricing Model	Initium Lab	2016-05-28	lang-English GitHub CodeHosting	<a href="http://initiumlab.com/blog/20160528-github-pricing/">http://initiumlab.com/blog/20160528-github-pricing/</a>
Geek 如何對抗莆田系醫藥？	Xia Mengyao	2016-05-22	lang-Chinese GitHub CrowdSource	<a href="http://initiumlab.com/blog/20160522-putian-hospital/">http://initiumlab.com/blog/20160522-putian-hospital/</a>
Time Saving One-Liners for Journalists	Initium Lab	2016-04-08	lang-English Tricks	<a href="http://initiumlab.com/blog/20160408-Time-Saving-One-Liners-for-Journalists/">http://initiumlab.com/blog/20160408-Time-Saving-One-Liners-for-Journalists/</a>
帶一本舊書時光：香港 20 處讀書聖地	蘇麗芸, Initium Lab	2016-04-01	lang-Chinese Interactive	<a href="http://initiumlab.com/blog/20160401-blog-bookreading-place-in-hk/">http://initiumlab.com/blog/20160401-blog-bookreading-place-in-hk/</a>
你真的了解數據新聞嗎？——八大誤區，你中了幾招？	Xingchen Zhou	2016-03-16	lang-Chinese datajournalism	<a href="http://initiumlab.com/blog/20160316-eight-myths-about-data-journalism/">http://initiumlab.com/blog/20160316-eight-myths-about-data-journalism/</a>
Google Sheets 技巧總結	Chao Tianyi	2016-02-03	GoogleSheets	<a href="http://initiumlab.com/blog/20160203-google-sheets/">http://initiumlab.com/blog/20160203-google-sheets/</a>
Google 研究經費競爭本已激烈 許審制度再被批	張軒婷	2016-02-02	RGCFund	<a href="http://initiumlab.com/blog/20160202-rgc/">http://initiumlab.com/blog/20160202-rgc/</a>

## Scraper pattern

### Data structure

"list-of-dict" structure is preferred. We also organise our code in this way:

- First (outer) layer is `list` -- iterate the data items we are interested in.
- Second (inner) layer is `dict` -- extract the features/ properties of a single data item.

`list-of-list` is one alternative to store the data. The advantage is compact representation of data entries. Instead of having `{key1: value1, key2: value2, ...}`, we have `[value1, value2, ...]`. The (insignificant) disadvantage is missing "table headers", or "column names" which appeared as keys in the list-of-dict representation. One can maintain this information outside `dataset`.

The choice of data structure is closely related the workflow of your program. So, put it another words, it is a reflection of the thought process. Please checkout the [imdb.com scraper](#) for a complete example of this method.

Also read the following section to compare the two different workflow

## Item-first v.s. attribute first

Item-first approach is adopted as best practice when you get started. Suppose we scrape the OpenRice website. Each item is a restaurant and fields include `title`, `like`, `location`, etc. Suppose we already have BeautifulSoup object in `mypage`. Following is the framework for item-first approach:

```
dataset = []
myitems = mypage.find_all(??)
for myitem in myitems:
 title = myitem.find(??).???
 like = myitem.find(??).???
 location = myitem.find(??).???
 ...
 # the variables can be None to represent missing data
 dataset.append([title, like, location, ...])
...
csv.writerows(dataset) # one line is enough for the output.
```

Attribute-first approach is not preferred, although sometimes the code seems simpler. We put the framework here for the completeness of discussion.

```
titles = []
likes = []
locations = []
...
for e in mypage.find_all(??):
 titles.append(e.??)
for e in mypage.find_all(??):
 likes.append(e.??)
for e in mypage.find_all(??):
 locations.append(e.??)
...
for i in range(len(titles)):
 csv.writerow(titles[i], likes[i], locations[i], ...)

Or try this more compact method:
csv.writerows(zip(titles, likes, locations, ...))
```

## Deal with missing data in scraping

One general guideline in data processing is to preserve as much original information as possible at the early stage of the pipeline. The downstream analysis programs can always decide how to deal with missing data. So instead of substituting the missing data with some "reasonable value", it is better to put `None` in that place. `None` will become empty cell in CSV and `null` in JSON to represent "empty". In later chapters, when we load the data via `pandas`, the data frame will put an `Nan` in that place, meaning "not a number". The `None`, `null`, `NaN`, or `empty`, are language specific way of treating missing data. No matter which way it adopts, letting the user know the data is missing is important.

To further understand this **missing data is data** philosophy, one can simply mind experiment the "average" operation. When the data entry is marked as missing, we just skip this data. However, if someone substituted this missing data in upstream with some reasonable value, say "0", the output will be smaller -- "0" does not contribute to the numerator but having a valid data entry here contributes 1 more to the denominator.

## Bonus: Scrape by text processing and regular expression

You may have noticed one way of scraping called "text processing". Common string functions in Python are like `strip()`, `split()`, `find()`, `replace()`, `str[begin:end]`. The advantage of text processing is its simplicity and you can write intuitive codes. The disadvantage of text processing is that it is error prone. Never the less, handling text is one important technique in data analysis pipeline.

Interested readers can further study [Regular Expression](#) (RegEx, regex, `re`) in Python. It is a powerful way for pattern matching and pattern substitution. The learning curve of regex is sharp so we omit the discussion in this chapter. We *might* revisit this concept and given an introduction in the text process chapter.

## Bonus: Crawler

### Workflow of a search engine like Google

A search engine mainly works in following way:

1. Crawl web pages from the Internet
2. Store those web pages in a distributed cluster
3. Build reverse index, which is essentially a mapping from the term (keyword) to web pages
4. Analyse the user query and use terms to recall candidate pages
5. Rank the candidate pages according to their relevance, using many features including term level, page level and user level ones.

### Crawler is more than scraper

A crawler is essentially a super module of scraper. When talking about "scraper", we mainly focus on retrieving and parsing a single document, be it an HTML, PDF, or image. Most of the time, we deal with HTML documents. "crawler" can follow the hyperlinks in a document, scrape documents pointed by those hyperlinks, and find new hyperlinks -- thus crawling.

Crawler is an essential building block for a search engine. Think of how Google and Baidu can reach the whole WWW-world without knowing where it is, or how large it is. It all starts by giving a set of "seed pages", and let the crawler expand the horizon by following the links on the pages.

### Crawler is not necessary in most of your cases

As a beginner of programmatic data collection, you often find crawler is non-necessary. The major reason is that in our use case, the "crawling zone" is bounded, namely there is a systematic way to specify where to crawl and how to crawl. In such scenario, you only need to focus on "scraper" part. Once you can handle one page, you can systematically generate other pages, or rules/ operation sequences to find other pages. Here are some examples of common generators:

- Find pages to scrape from a "hub page" -- e.g. find links to news articles from a list page, and then scrape each page from the list.
- Manipulate page id parameter in URL -- e.g. a forum/ a Wordpress blog site.
- Start from a seed page and continuously click "Next Page" -- e.g. search engine results. [notes-week-08.md](#) will explain in details how to emulate browser in a programmatic way.

## scrapy

[scrapy](#) is the most commonly used crawler framework in Python. Given this framework, you only need to write a `parse` function, which basically does two jobs:

1. Emit "data item" found in the current page

2. Emit "page item" that `scrapy` framework needs to follow.

Note the keyword `yield` when you try this framework. This is called "Generator" -- a common construct in most modern programming languages. You have already used generator for many times throughout this class. We don't mention it to avoid possible confusion. Interested readers can find a simple tutorial [here](#).

## scrapy-cluster

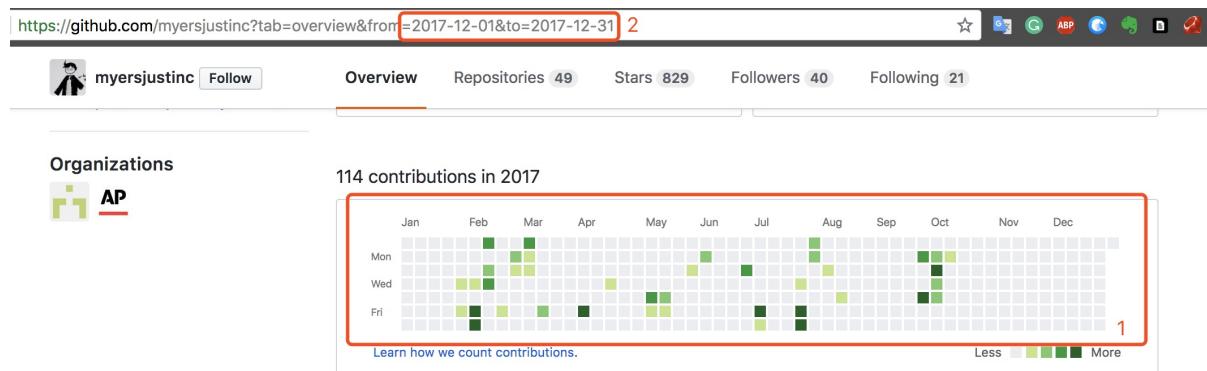
`scrapy-cluster` is a distributed crawling framework, that uses [Docker](#) container technology to easily and horizontally scale out with your task size. It is a super module of `scrapy`. The layering is as follows:

1. `parse()` function in `scrapy` -- This is essentially a "Scraper" -- single page, parsing
2. `scrapy` -- This is essentially **one** "Crawler" -- The emitted data items and page items are within one crawling topic.
3. `scrapy-cluster` -- This is essentially a (distributed) cluster of **multiple** Crawlers. Those crawlers can have different topics, priorities, scheduling options, etc.

## Exercises and Challenges

### Scrape github users' contribution frequency

Scrape contributions of [Justin Myers](#). We just need to know in different time, how many contributions he committed (1). You can change the url parameters to get the contributions of different time (2).



Please save the results into csv like the following.

1	Contribution	Date
2	0	2011-01-01
3	0	2011-01-02
4	0	2011-01-03
5	0	2011-01-04
6	0	2011-01-05

### Further challenge1: more users

Given a list of users, scrape all of their contribution frequency and store accordingly with the user identifier.

### Further challenge2: detailed activities

Below the contribution calendar, there is a list of detailed activities. Can you further scrape those activities? You may need to design a good table structure to store the data.

## Scrape the faculty list

Try to scrape as more fields as possible, e.g. name, introduction, contact, etc. Here are some potential scraping target for your choice:

- <http://www.jour.hkbu.edu.hk/faculty/>
- [http://www.comm.hkbu.edu.hk/comd-www/english/people/m\\_faculty\\_dept.htm](http://www.comm.hkbu.edu.hk/comd-www/english/people/m_faculty_dept.htm)
- [http://www.comm.hkbu.edu.hk/comd-www/english/people/m\\_faculty\\_dept\\_academy\\_film.htm](http://www.comm.hkbu.edu.hk/comd-www/english/people/m_faculty_dept_academy_film.htm)
- [http://www.comm.hkbu.edu.hk/comd-www/english/people/m\\_faculty\\_dept\\_communication\\_studies.htm](http://www.comm.hkbu.edu.hk/comd-www/english/people/m_faculty_dept_communication_studies.htm)
- [http://www.comm.hkbu.edu.hk/comd-www/english/people/m\\_faculty\\_dept\\_journalism.htm](http://www.comm.hkbu.edu.hk/comd-www/english/people/m_faculty_dept_journalism.htm)

Note, scraping techniques demoed in this chapter may not be enough. You may need to do some text processing (`str` functions).

Note, for the 2-5 URLs, you may suffer from encoding issue because the server side configuration has a problem. You can use the `r.encoding` to specify the right encoding. Following is a sample:

```
r = requests.get('http://www.comm.hkbu.edu.hk/comd-www/english/people/m_faculty_dept.htm')
r.encoding = 'utf-8' # Try what happens without this line
mypage = BeautifulSoup(r.text)
mypage.find('td', {'class': 'personNameArea'}).text
```

## Scrape Haunted House in Hong Kong

<https://www.squarefoot.com.hk/haunted/> – a list of haunted houses in Hong Kong. A group of DJ students last term used [scrapinghub.com](#) to crawl the data. Now you can write Python codes with fine control. There are two more databases for you to cross-check <https://news.gohome.com.hk/tc/category/haunted-house/haunted-house-article/> and <http://www.hkea.com.hk/UHousesServ>

## Scrape Hacker News

Hacker News is the world number 1 technology news crowd gathering service: <https://news.ycombinator.com/>. One can sense the trend from those articles shared by mostly guru users. You can get post title, link, points (a.k.a. "likes") from the website. More introduction of this news source and related services can be found [here](#).

## Scrape Juejin

<https://juejin.im/> is the Chinese counterpart of Hacker News. One can check the difference of topic popularity between HN and juejin, in past 5-10 years, and answer the question: is China switching from a copycat role to a leading role in the technology world? There might be a difficulty when comparing topics because the two sites use different languages.

## Bonus: Automatic trending topic detection and posting

You can setup a robot to scrape HackerNews continuously. Once a topic gets enough points, that may be a trending one in the technology sphere. As an automated journalist, you may want to immediately post this to a Twitter account. For example, [this bot](#) tweets a Hacker News story once it reaches 50 points. There are other thresholds in this family.

## Scrape Douban Top 250

<https://movie.douban.com/top250> . Maybe you already know a convenient way using API, this is still good exercise. Is there any difference between the data you scraped from web and the data you retrieved via API?

## Scrape IMDB Top 250

<https://www.imdb.com/chart/top> . Can you get as many fields as you can? Can you find any difference between the two lists? What are the implications of the differences?

## Scrape opening journalist positions in the world

The following online job list can be scraped with the knowledge in this chapter:

- <https://careers.journalists.org/jobs/10753217/graphics-journalist>
- <https://www.indeed.com/q-Data-Journalism-Internship-jobs.html>
- <https://hk.jobsdb.com/hk/search-jobs/data-journalist/1>
- [https://www.glassdoor.co.uk/Job/data-journalist-jobs-SRCH\\_K00,15.htm](https://www.glassdoor.co.uk/Job/data-journalist-jobs-SRCH_K00,15.htm)
- <https://www.linkedin.com/jobs/search/?keywords=data%20journalism&location=%E5%85%A8%E7%90%83&locationId=OTHERS.worldwide>

**Note:** LinkedIn is dynamic and requires [selenium](#) in Chapter 6 to run. You need to work around the sign in page.

(Feel free to add to the list when you find new ones)

## Related Readings

Here are some scrapers and the output dataset from our past students, you can learn some tricks and search for inspirations of your own project:

- [HK Carpark price data](#)
  - [Qidian](#)
  - [CTrip scenic point data](#)
  - This [blog post](#) has some past scraping ideas. Some ideas are beyond this chapter. You can ask for an evaluation before start.
- 

If you have any questions, or seek for help troubleshooting, please [create an issue here](#)

## Week 08: Advanced scraping - anti-crawler, browser emulation and other nitty gritty

- [Week 08: Advanced scraping - anti-crawler, browser emulation and other nitty gritty](#week-08-advanced-scraping--anti-crawler-browser-emulation-and-other-nitty-gritty) - [Objective](#objective) - [Anti-crawling](#anti-crawling) - [User agent](#user-agent) - [Bonus: Test HTTP requests](#bonus-test-http-requests) - [Rate throttling](#rate-throttling) - [Hide numeric incremental IDs](#hide-numeric-incremental-ids) - [Hide key information using special fonts](#hide-key-information-using-special-fonts) - [Bonus: Stateful page transition](#bonus-stateful-page-transition) - [Bonus: client authentication](#bonus-client-authentication) - [Common issues](#common-issues) - [Encoding](#encoding) - [Network delay and jitter](#network-delay-and-jitter) - [Network interruption](#network-interruption) - [Firewall](#firewall) - [Browser rendering delay](#browser-rendering-delay) - [Browser emulation](#browser-emulation) - [Why use Browser Emulation](#why-use-browser-emulation) - [Limitation](#limitation) - [Selenium](#selenium) - [Downloading Python bindings for Selenium](#downloading-python-bindings-for-selenium) - [Drivers](#drivers) - [Navigating](#navigating) - [Locating Elements](#locating-elements) - [Find\_element(s)\_by\_css\_selector](#find\_elements\_by\_css\_selector) - [Locating elements by attribute](#locating-elements-by-attribute) - [Locating elements with multiple class name](#locating-elements-with-multiple-class-name) - [Locating Child Element](#locating-child-element) - [Scroll down certain element](#scroll-down-certain-element) - [Example: CNN articles scraping](#example-cnn-articles-scraping) - [Fundamental: One page](#fundamental-one-page) - [Advanced: All pages](#advanced-all-pages) - [Splinter](#splinter) - [Finding elements](#finding-elements) - [Fundamental version: one page](#fundamental-version-one-page) - [Advanced version: all pages](#advanced-version-all-pages) - [Bonus: Twitter example with browser emulation](#bonus-twitter-example-with-browser-emulation) - [Analyse Network Traces](#analyse-network-traces) - [Bonus: Crawl mobile Apps](#bonus-crawl-mobile-apps) - [Packet analysis](#packet-analysis) - [Example: Kwai (kuaishou)](#example-kwai-kuaishou) - [App de-compilation](#app-de-compilation) - [App emulation](#app-emulation) - [Bonus: Other quick scraping/ crawling tricks](#bonus-other-quick-scraping-crawling-tricks) - [Exercises and Challenges](#exercises-and-challenges) - [In-bound marketing and SEO auditing](#in-bound-marketing-and-seo-auditing) - [Crawl the legal case of China](#crawl-the-legal-case-of-china) - [Bonus: Crawl Weibo data and discover KOL](#bonus-crawl-weibo-data-and-discover-kol) - [Bonus: Cheat an online voting system](#bonus-cheat-an-online-voting-system) - [Related Readings](#related-readings)

In this chapter, we will learn **advanced scraping**, which is scraping dynamic loading pages or some pages that need us interactively involved, the ones that we need emulate browsers to navigate and find elements.

As you may have a clue due to its name, this chapter will be more demanding than previous chapters. We need spend more time to learn how to use those two complete new libraries - `selenium` and `splinter`. Which are similar but with a little bit difference. At the same time, we need learn more about `Frontend three`: HTML, JS, and CSS. And how to locate and find elements from them.

After this chapter, I believe, we can apply what we learn to the most of `usual scraping cases`, most of websites, social media etc...Which will paves the way for further data analysis stage (interested students can talk to me or refer to chapter 7 for learning advanced).

## Objective

- Bypass anti-crawler by modifying user-agent
- Handle glitches: encoding, pagination, ...
- Handle dynamic page with headless browser
- Handle login with headless browser
- Scrape social networks
- Case studies on different websites

- Further strengthen the list-of-dict data types; organise multi-layer loops/ item based parsing logics.

## Anti-crawling

### User agent

The simplest way to prevent crawler access it to limit by user agent. "User agent" can be thought of synonym for "web browser". When you surf the Internet with a normal web browser, the server will know whether you use Chrome, Firefix, IE, or other browsers. Your browser give this information to the web server by a field called `user-agent` in the HTTP request headers. Similarly, `requests` is a like a web browser, for Python code, not for human. It also gives the `user-agent` to the web server and the default value is like `python-requests/*`. In this way, the server knows that the client is Python requests module, not a regular human user. One can by-pass this limit by modifying the user-agent string.

```
r = requests.get(url,
 headers = {
 'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_3) AppleWebKit/537.36 (KHTML, like Gecko)
o) Chrome/64.0.3282.167 Safari/537.36'
 }
)
```

Full code and demo can be found in [this notebook](#).

### Bonus: Test HTTP requests

<https://nghttp2.org/httpbin> is a useful service to test HTTP requests. This service basically echos the content or certain parameters from your HTTP request. You can get better idea of what your tools send to the server via this service.

Example: Check the user-agent of shell command `curl` :

```
%curl https://nghttp2.org/httpbin/user-agent
>{"user-agent":"curl/7.54.0"}
```

Example: Check the default user-agent of `requests` :

```
>>> r = requests.get('https://nghttp2.org/httpbin/user-agent')
>>> r.text
'{"user-agent":"python-requests/2.19.1"}\n'
>>> r = requests.get('https://nghttp2.org/httpbin/user-agent', headers={'user-agent': 'See, I modified the user
agent!!!'})
>>> r.text
'{"user-agent":"See, I modified the user agent!!"}\n'
```

### Rate throttling

- Limit by IP
- Limit by cookie/ access token
- Limit by API quota per a unit time, usually implemented with a [leaky bucket algorithm](#)

### Hide numeric incremental IDs

Scrapers usually return a list of objects. Sometimes the list can be enumerated given certain IDs. One common case is the use of `page=xxx` parameter in the URL. You can increment the page number and assemble valid URLs. Some carefully designed web service will try to hide this kind of incremental IDs, in order to prevent other's crawler

accessing this information so easily. Nevertheless, you can analyse page structure in depth and find a way. The principle is that: as long as the user can see it, there is no way to ultimately hide it from a robot. The only thing website builder can do is to make the crawling less straightforward.

## Hide key information using special fonts

qidian.com hides key information using special fonts. When normal user visit the webpage, those non-printable characters are rendered in a normal way because they load a special font. However, when you check the Chrome Developer Console, or try to get the values of the string in Python, the number field appears to be non-printable characters. The way to work around is to analyse the font file and make a decoding logic yourself. Find discussion on [#85](#).

## Bonus: Stateful page transition

Most early websites are designed in a *stateless* manner. That is, the order how you visit those pages does not make a difference. For example, you can visit article 1 and jump to article 2.

On the contrary, some modern websites can be *stateful*. One common example is the requirement of login. You need to first visit the login page, before sending username/ password to the server. You need to have successfully sent the username/ password in order to access certain restricted resources.

Imagine a social network for another *stateful* example. As a regular user, you must first visit your own homepage to access friend list. Then you can access the profile page of a friend, after which you can visit a specific post.

Websites can record the user activity log and enforce stateful transition. However, this design also consumes a lot of resources and is not commonly preferred. In some mission critical sites, like banks, you may find stateful enforcement by certain token injected from the server side onto the current page.

Again, nothing can be completely hidden if a regular user can access it. More efforts are needed.

## Bonus: client authentication

Checking user-agent is the first step to identify whether a client is a valid or not. This step is generally referred as "client authentication". The server checks if the request is sent from a legitimate client. Examples are like [client side signature](#). However, the web is an open world. You can be assured that people can hide nothing if the data is already in your browser. Even if the web developers apply complex computational logics to conduct client authentication, the client side authenticator code is available in your browser as Javascript. With some cryptography knowledge, one may be able to reverse engineer it.

However, cracking the system is not the purpose. Our objective is to get data. Instead of cracking and translating the logics into Python script, we had better re-use the Javascript scripts as-is. Or further more, we can run browser emulator to naturally trigger those logics, which may be a more direct solution. Browser emulator is one major topic introduced in this chapter.

# Common issues

## Encoding

Example 1: See [51job.com example](#)

Example 2:

```
r = requests.get('http://www.comm.hkbu.edu.hk/comd-www/english/people/m_faculty_dept.htm')
r.encoding = 'utf-8'
```

```
mypage = BeautifulSoup(r.text)
mypage.find('td', {'class': 'personNameArea'}).text
```

## Network delay and jitter

- Delay refers to the elapsed time used for the browser to receive the complete HTTP response since the first byte. Your code needs to consider potential delays especially under different network condition. Or the data you are trying to access may not be ready when your code tries to process. This is not a major problem when we use `requests` in last chapter, because `requests` is "blocking". That is, the whole HTTP response will be received before Python further executes the code. However, it may arise as a major problem in this chapter when we use browser emulator.
- Jitter refers to the unstable/ unsteady delay. Sometimes the delay is large and some time it is small. You will find your usually working codes go wrong in some rare scenarios.

To handle delay and jitter, the common strategy is to wait and test. You can use `time.sleep` to pause for some time before proceed. If there is a way to test the finishing condition, you had better test before move. For example, use `.find_element_by_xxx` to check if the intended element is already loaded. If not, wait further.

## Network interruption

Network can be interrupted in many ways, like sudden loss of wifi signal. When the network is interrupted, you may get partial data or corrupted data. Make sure to guard your parser codes with `try...except` block, handle the errors and print detailed log for further trouble shooting.

## Firewall

If you are behind firewalls, which is common in campus and enterprise networks, some automatic HTTP requests may be flagged and further stopped. There is no direct solution to this. When in doubt, try one alternative network (wifi, wired, 4G, ..) to see if things work.

## Browser rendering delay

When you use browser emulator, you also need to know that it takes time to render the whole page. You do not feel that because computer is very fast. Most of the loading and rendering could have happened in minisecond level. However, when you use automatic programs to browse and click, things become different. Your own program may be so fast that the dependent element/ data has not loaded when you try to access it.

For example, `StaleElementReferenceException` and `IndexError` are quite common when using `selenium`. The error sometimes disappears when you execute the same script with the same parameter again. Or it is better to add some `time.sleep` between critical operations. For example, you want to wait the browser to load new content after triggering a click event on a button.

Related issues:

- Booking.com #68

## Browser emulation

Primarily, Browser emulation or browser automation is used for automating web applications for testing purpose. Like when you build your web application, you want to simulate how many users your server can handle, and how the users act when they look into your website, how they open page, click, navigate and read the the page content.

But browser emulation is certainly not limited to just that. For our course, we mainly use it to manipulate the browser, to interactively communicate with the website, locate the information and get the data we want.

## Why use Browser Emulation

1. Some of complicated website can't be directly scraped by static method. For example, some elements, especially page turning buttons/links in the webpage have embedded javascript codes, which need users certain actions to further loading the content.
2. Browser Emulation way can handle some complicated scraping work like ones that need you login.
3. Some webpages have strictly rules for anti-scraping. However, in browser emulation, we simulate users' behaviors, which is more camouflaged and not easy to discover, meaning that the limits is smaller than static scraping like `request`.

In our course, we mainly introduce two libraries - `Selenium` and `Splinter` for browser emulation and dynamic scraping. Those packages are wildly used in this field. And their documentations are easy to read.

## Limitation

Each time, it need to load all the content of the webpage, the crawling speed is slow, therefore not suitable for scraping cases with a large load of data.

## Selenium

Selenium is a set of different software tools, each with a different approach to supporting browser automation. These tools are highly flexible, allowing many options for locating and manipulating elements within a browser, the key tool we will use is Selenium Python bindings.

Selenium Python bindings provides a simple API to write functional/acceptance tests using Selenium WebDriver. Through Selenium Python API you can access all functionalities of Selenium WebDriver in an intuitive way.

You can visit [here](#) to learn how to use those functions by yourself. In the following example, we will use CNN articles scraping case to elaborate the basic functions of it, and how to scrape a webpage that need our interaction.

## Downloading Python bindings for Selenium

```
!pip install selenium #in Jupyter Notebook
```

```
from selenium import webdriver #import
```

## Drivers

Selenium requires a driver to interface with the chosen browser. Chrome, for example, requires Chromedriver, which needs to be installed before the below examples can be run.

you can download different drivers for supported browsers in the following links:

Supported Browsers	Download Links
Chrome	<a href="https://sites.google.com/a/chromium.org/chromedriver/downloads">https://sites.google.com/a/chromium.org/chromedriver/downloads</a>
Firefox	<a href="https://github.com/mozilla/geckodriver/releases">https://github.com/mozilla/geckodriver/releases</a>
Safari	<a href="https://webkit.org/blog/6900/webdriver-support-in-safari-10/">https://webkit.org/blog/6900/webdriver-support-in-safari-10/</a>

For windows users: please refer to [here](#) for instruction of download.

After you download the webdriver, we can use the following command to initiate the webdriver.

```
browser = webdriver.Chrome() #default to initiate webdriver, you can assign it with driver or browser or other things you like.
```

**Note:** Make sure it's in your PATH. e. g. place it in `/usr/bin` or `/usr/local/bin`. If it's not in the PATH, when you initiate the Chromedriver, it will raise error `Message: 'chromedriver' executable needs to be in PATH.` If raise this error, you can solve this problem by putting dependency/chromedriver you download into PATH. The PATH here is the current working folder, where your Jupyter Notebook is running. You can check out where it is by the following command.

```
!echo $PATH #you will get a set of paths and the first one is what we want. In the following example, it will be as:
#/Library/Frameworks/Python.framework/Versions/3.6/bin
!ls #add the first path returned from last step to list all files in the folder
!open #add the first path returned from first step to open the path and put the chromedriver into the path
```

```
: !echo $PATH
/Library/Frameworks/Python.framework/Versions/3.6/bin:/anaconda3/bin:/Library/Frameworks/Python.framework/Versions/3.6/bin:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin
```

```
the first path /Library/Frameworks/Python.framework/Versions/3.6/bin is where you are working now
!ls /Library/Frameworks/Python.framework/Versions/3.6/bin #list files in this folder
```

```
2to3 jupyter-bundlerextension pycodestyle
2to3-3.6 jupyter-console pydoc3
check-manifest jupyter-kernel pydoc3.6
codecov jupyter-kernelspec pygmentsize
coverage jupyter-migrate pytest
coverage-3.6 jupyter-nbconvert python3
coverage3 jupyter-nbextension python3-32
coveralls jupyter-notebook python3-config
dukpy jupyter-qtconsole python3.6
dukpy-install jupyter-run python3.6-32
easy_install jupyter-serverextension python3.6-config
easy_install-3.6 jupyter-troubleshoot python3.6m
idle3 jupyter-trust python3.6m-config
idle3.6 naked pyvenv
iptest pbr pyvenv-3.6
iptest3 pip tox
ipython pip3 tox-quickstart
ipython3 pip3.6 virtualenv
jsonschema pj wordcloud_cli.py
jupyter py.test you-get
```

```
: !open /Library/Frameworks/Python.framework/Versions/3.6/bin #after open it, place chromedriver in it.
```

## Navigating

You can do a lot of interactive things with the webpage with help of the selenium, like navigating to a link, searching, scrolling, clicking etc. In the following example, we will demo the basic usage of navigating.

```
from selenium import webdriver
browser = webdriver.Chrome() #initiate webdriver
browser.get('http://google.com/') #visit to google page
element = browser.find_element_by_name("q") #Find the search box
element.send_keys("github python for data and media communication gitbook") #search our openbook
element.submit() #submit search action
you will find the webpage will automatically return the results you search
open_book = browser.find_element_by_css_selector('.g')
link = open_book.find_element_by_tag_name('a') #find our tutorial
you can also find by the link text. link = browser.find_element_by_partial_link_text('GitHub - hupili')
link.click() #click the link, enter our tutorial
browser.execute_script("window.scrollTo(0,1200);") #scroll in the page, window.scrollTo(x,y), x means horizontal
l, y means vertical
notes_links = browser.find_element_by_link_text('notes-week-08.md') #find link of notes 6
```

```
notes_links.click() #click into notes 6
#browser.close()
```

## Locating Elements

There are many ways to locate the elements. It's similar to the usage in `requests` method, just a simple `find...` sentence but more diverse.

Selenium provides the following methods to locate elements in a page:

- `find_element(s)_by_id`
- `find_element(s)_by_name`
- `find_element(s)_by_xpath`
- `find_element(s)_by_link_text`
- `find_element(s)_by_partial_link_text`
- `find_element(s)_by_tag_name`
- `find_element(s)_by_class_name`
- `find_element(s)_by_css_selector`

For instruction of the syntax, you can refer this [documentation](#). In our notes, we mainly use `find_element(s)_by_css_selector` method, due to its easy expression and rich matchability.

## Find\_element(s)\_by\_css\_selector

### Locating elements by attribute

Eg:

```
<div id="summaryList_mixed" class="summaryList" style="display: block;"></div>
```

```
css = element_name[<attribute_name>='<value>']
```

1. Select id. Use `#` notation to select the id:

```
css="div#summaryList_mixed" or "#summaryList_mixed"
```

1. Select class. Use the `.` notation to select the class:

```
css="div.summaryList" or just css=".summaryList"
```

1. Select multiple attributes:

```
css="div[class='summaryList'] [style='display:block']"
```

### Locating elements with multiple class name

When using `.className` notation, every class needs a prefix `.` : `.className1.className2.className3` (no blanks between those class names if they are used to attribute one element)

For example: for the following case:

```
<i class=".sr_item sr_item_new sr_item_default sr_property_block sr_flex_layout" >
```

The css will be like this:

```
css='.sr_item.sr_item_new.sr_item_default.sr_property_block.sr_flex_layout'
```

For detail cases, please refer [here](#)

### Locating Child Element

Eg:

```
<div id="summaryList_mixed" class="summaryList" style="display: block;">
 <div class="summaryBlock"></div>
 <div class="summaryBlock"></div>
 <div class="summaryBlock"></div>
 <div class="summaryBlock"></div>
</div>
```

1. Locate all children

```
css="div#summaryList_mixed .summaryBlock"
```

1. Locate the certain one with "nth-of-type". The first one is "nth-of-type(1)", and the last one is "last-child"

```
css="div#summaryList_mixed .summaryBlock:nth-of-type(2)"
```

For more explanations and examples about css selector, here is a good [documentation](#) you can refer to.

### Scroll down certain element

In the navigation or scraping, we may need to click the button to turn pages. And the buttons locate differently in different website. How we locate those buttons? Scroll down to the element may help you accomplish this. We use the [cnn example](#) to demo here, and test how to turn pages via browser emulation.

```
from selenium import webdriver
import time
browser = webdriver.Chrome()
url = 'https://money.cnn.com/search/index.html?sortBy=date&primaryType=mixed&search=Search&query=trade%20war'
browser.get(url)
next_button = browser.find_element_by_css_selector('#mixedpagination ul.pagingLinks li.ends.next span a') #get
the element's location
next_button.location
loc = next_button.location
browser.execute_script("window.scrollTo({x}, {y});".format(**loc)) #scroll to the element
next_button.click()
```

Apart from directly scroll to the elements. There are two scrolling usages you may need to know.

```
#method 1
browser.execute_script('window.scrollBy(x,y)') # x is horizontal, y is vertical

#method 2
browser.execute_script('window.scrollTo(0, document.body.scrollHeight);') #scroll to the page bottom

#method 3
browser.execute_script('window.scrollTo(0, document.body.scrollHeight/1.5);') #you can divide numbers after the
page height

#method 4
```

```
element = browser.find_element_by_class_name("pn-next")#locate the element
browser.execute_script("return arguments[0].scrollIntoView();", element) #scroll to view the element
```

## Example: CNN articles scraping

The following is the link of results returned by keyword searching of `trade war`. We can scrape those articles title, time and url for further studying. The reason why we need use `selenium` is because the page turning links are embedded javascript codes, which cannot be extracted and use directly in `requests` way. To solve that, we need to interact with the page, and do browser emulation.

<https://money.cnn.com/search/index.html?sortBy=date&primaryType=mixed&search=Search&query=trade%20war>

### Fundamental: One page

```
!pip3 install selenium # if you installed before, just ignore
from selenium import webdriver

browser = webdriver.Chrome()
browser.get('http://money.cnn.com/search/index.html?sortBy=date&primaryType=mixed&search=Search&query=trade%20war')

articles = []
for session in browser.find_elements_by_css_selector('#summaryList_mixed .summaryBlock'): #find all articles wrapped in the path of class='summaryBlock' under the id='summaryList_mixed'
 article = {}
 h = session.find_element_by_css_selector(".cnnHeadline a")
 article['headline'] = h.text #find headline block
 article['url'] = h.get_attribute('href') #get url attributes from headline block
 article['date'] = session.find_element_by_css_selector("span.cnnDateStamp").text #find date
 articles.append(article)
articles
```

Output:

```
articles
[{'date': 'Sep 13, 2018',
 'headline': "There's no way to avert the next financial crisis, warn former regulators from 2008 meltdown",
 'url': 'http://money.cnn.com/2018/09/12/news/economy/financial-crisis-anniversary-bernanke/index.html'},
 {'date': 'Sep 12, 2018',
 'headline': "America is now the world's largest oil producer",
 'url': 'http://money.cnn.com/2018/09/12/investing/us-oil-production-russia-saudi-arabia/index.html'},
 {'date': 'Sep 11, 2018',
 'headline': 'Is the tech stock rally coming to an end?',
 'url': 'http://money.cnn.com/2018/09/11/technology/tech-stocks-fang/index.html'},
 {'date': 'Sep 11, 2018',
 'headline': 'Apple is still struggling to sell iPhones in the world's hottest markets',
 'url': 'http://money.cnn.com/2018/09/11/technology/apple-iphone-2018-india/index.html'},
 {'date': 'Sep 11, 2018',
 'headline': "China's troubles push Hong Kong stocks into a bear market",
 'url': 'http://money.cnn.com/2018/09/11/investing/hang-seng-bear-market/index.html'},
 {'date': 'Sep 10, 2018',
 'headline': 'The long-running CBS show is over for its longtime leading man',
 'url': 'http://money.cnn.com/2018/09/10/media/les-moonves-bill-carter/index.html'},
 {'date': 'Sep 10, 2018',
 'headline': "China's exports are slowing. The trade war will make things worse",
 'url': 'http://money.cnn.com/2018/09/10/news/economy/china-exports-trade-war/index.html'},
 {'date': 'Sep 10, 2018',
 'headline': "Moonves speaks after CBS breakup, says he's 'deeply saddened'",
 'url': 'http://money.cnn.com/2018/09/10/media/reliable-sources-09-09-18/index.html'},
 {'date': 'Sep 10, 2018',
 'headline': 'How Jack Ma went from English teacher to tech billionaire',
 'url': 'http://money.cnn.com/2018/09/09/technology/business/jack-ma-alibaba-bio/index.html'},
 {'date': 'Sep 09, 2018',
 'headline': 'Les Moonves is out at CBS after harassment allegations, corporate battle',
 'url': 'http://money.cnn.com/2018/09/09/media/les-moonves-cbs/index.html'}]
```

### Advanced: All pages

```
from selenium import webdriver
import time #mainly use its time sleep function

def get_articles_from_browser(b):
 articles = []
 for session in browser.find_elements_by_css_selector('#summaryList_mixed .summaryBlock'): #find all article
s wrapped in the path of class='summaryBlock' under the id='summaryList_mixed'
 article = {}
 h = session.find_element_by_css_selector(".cnnHeadline a")
 article['headline'] = h.text #find headline block
 article['url'] = h.get_attribute('href') #get url attributes from headline block
 article['date'] = session.find_element_by_css_selector("span.cnnDateStamp").text #find date
 articles.append(article)

 return articles

url = 'http://money.cnn.com/search/index.html?sortBy=date&primaryType=mixed&search=Search&query=trade%20war'
browser = webdriver.Chrome()
browser.get(url)
time.sleep(2) #sleep 2 second for each call action, if it's too frequently with no sleep time, its has high opp
ortunity to be banned from the website.

all_page_articles = []
for i in range(10):
 time.sleep(0.5)
 try:
 new_articles = get_articles_from_browser(browser)
 all_page_articles.extend(new_articles)
 browser.execute_script('window.scrollTo(0, document.body.scrollHeight/1.5);')#test several numbers to c
hoose a suitable one
 next_page = browser.find_element_by_link_text('Next')
 next_page.click()
 except Exception as e:
 print(e)
 print('Error on page %s' % i)

import pandas as pd #spoiler. pandas is the key module in the next chapter, you can check out chapter 7 for fur
ther information.
df = pd.DataFrame(all_page_articles) #convert articles into dataframe
df
```

Output:

	<b>date</b>	<b>headline</b>	<b>url</b>
0	Sep 11, 2018	China's troubles push Hong Kong stocks into a ...	<a href="http://money.cnn.com/2018/09/11/investing/hang...">http://money.cnn.com/2018/09/11/investing/hang...</a>
1	Sep 11, 2018	Hong Kong bear market; Japan's chip deal; Alib...	<a href="http://money.cnn.com/2018/09/11/investing/prem...">http://money.cnn.com/2018/09/11/investing/prem...</a>
2	Sep 10, 2018	The long-running CBS show is over for its long...	<a href="http://money.cnn.com/2018/09/10/media/les-moon...">http://money.cnn.com/2018/09/10/media/les-moon...</a>
3	Sep 10, 2018	China's exports are slowing. The trade war wil...	<a href="http://money.cnn.com/2018/09/10/news/economy/c...">http://money.cnn.com/2018/09/10/news/economy/c...</a>
4	Sep 10, 2018	Moonves speaks after CBS shakeup, says he's 'd...	<a href="http://money.cnn.com/2018/09/10/media/reliable...">http://money.cnn.com/2018/09/10/media/reliable...</a>
5	Sep 10, 2018	How Jack Ma went from English teacher to tech ...	<a href="http://money.cnn.com/2018/09/09/technology/bus...">http://money.cnn.com/2018/09/09/technology/bus...</a>
6	Sep 09, 2018	Les Moonves is out at CBS after harassment all...	<a href="http://money.cnn.com/2018/09/09/media/les-moon...">http://money.cnn.com/2018/09/09/media/les-moon...</a>
7	Sep 07, 2018	US jobs report; British Airways hack; Bitcoin ...	<a href="http://money.cnn.com/2018/09/07/investing/prem...">http://money.cnn.com/2018/09/07/investing/prem...</a>
8	Sep 06, 2018	Hong Kong now has more super-rich people than ...	<a href="http://money.cnn.com/2018/09/06/investing/worl...">http://money.cnn.com/2018/09/06/investing/worl...</a>
9	Sep 05, 2018	Why Wall Street shouldn't worry about the trad...	<a href="http://money.cnn.com/2018/09/05/investing/mark...">http://money.cnn.com/2018/09/05/investing/mark...</a>
10	Sep 05, 2018	Emerging markets look sick. Will they infect W...	<a href="http://money.cnn.com/2018/09/05/investing/stoc...">http://money.cnn.com/2018/09/05/investing/stoc...</a>
11	Sep 05, 2018	Analyst: Trade war is big headline, not big pr...	<a href="https://money.cnn.com/video/news/economy/2018/...">https://money.cnn.com/video/news/economy/2018/...</a>
12	Sep 05, 2018	Big Tech on Capitol Hill; Emerging markets fea...	<a href="http://money.cnn.com/2018/09/05/investing/prem...">http://money.cnn.com/2018/09/05/investing/prem...</a>
13	Sep 05, 2018	Inside Craig Melvin's promotion on NBC's "Toda...	<a href="http://money.cnn.com/2018/09/04/media/craig-me...">http://money.cnn.com/2018/09/04/media/craig-me...</a>
14	Sep 05, 2018	Summer's over. Now investors have to get ready...	<a href="http://money.cnn.com/2018/09/04/investing/mark...">http://money.cnn.com/2018/09/04/investing/mark...</a>
15	Sep 04, 2018	Facebook's former security chief: US elections...	<a href="http://money.cnn.com/2018/09/04/technology/us-...">http://money.cnn.com/2018/09/04/technology/us-...</a>
16	Sep 04, 2018	A swine fever outbreak is the latest threat to...	<a href="http://money.cnn.com/2018/09/04/news/economy/a...">http://money.cnn.com/2018/09/04/news/economy/a...</a>
17	Sep 03, 2018	The trade war is deepening the gloom at Chines...	<a href="http://money.cnn.com/2018/09/03/news/economy/c...">http://money.cnn.com/2018/09/03/news/economy/c...</a>
18	Sep 02, 2018	Does Trump have a point about Google?	<a href="https://money.cnn.com/video/news/2018/09/02/do...">https://money.cnn.com/video/news/2018/09/02/do...</a>

## Splinter

Splinter achieves pretty much the same results as Selenium does, though there might be a little difference in syntax. In the following, we will also use `splinter` method to demo the cnn example, you can compare it with `selenium` method, and choose one you like to practice more.

```
!pip3 install splinter
from splinter import Browser
import time
url = 'http://money.cnn.com/search/index.html?sortBy=date&primaryType=mixed&search=Search&query=trade%20war'
browser = Browser('chrome')
browser.visit(url)
time.sleep(2)
```

## Finding elements

Splinter provides 6 methods for finding elements in the page, one for each selector type: `css` , `xpath` , `tag` , `name` , `id` , `value` , `text` . Each of these methods returns a list with the found elements. And you can use `index` to access each of them in the list. This method is different from `selenium` which provides with finding single element and list of elements. All in all, those two methods are very alike. You can check out [here](#) for Splinter doc about finding elements. In this case, we mainly use `find_by_css(css_selector)` method.

### Fundamental version: one page

```
!pip3 install splinter
from splinter import Browser
import time
url = 'http://money.cnn.com/search/index.html?sortBy=date&primaryType=mixed&search=Search&query=trade%20war'
```

```

browser = Browser('chrome')
browser.visit(url)
time.sleep(2)

articles = []
for block in browser.find_by_css('#summaryList_mixed .summaryBlock'):
 article = {}
 h = block.find_by_css('.cnnHeadline a')
 article['headline'] = h.text
 article['url'] = h['href']
 article['date'] = block.find_by_css('span.cnnDateStamp').text
 articles.append(article)

articles

```

Output:

```

articles
[{'date': 'Sep 13, 2018',
 'headline': 'The trade war is already hurting US companies in China',
 'url': 'http://money.cnn.com/2018/09/13/news/economy/amcham-china-trade-war/index.html'},
 {'date': 'Sep 13, 2018',
 'headline': "There's no way to avert the next financial crisis, warn former regulators from 2008 meltdown",
 'url': 'http://money.cnn.com/2018/09/12/news/economy/financial-crisis-anniversary-bernanke/index.html'},
 {'date': 'Sep 12, 2018',
 'headline': "America is now the world's largest oil producer",
 'url': 'http://money.cnn.com/2018/09/12/investing/us-oil-production-russia-saudi-arabia/index.html'},
 {'date': 'Sep 11, 2018',
 'headline': 'Is the tech stock rally coming to an end?',
 'url': 'http://money.cnn.com/2018/09/11/technology/tech-stocks-fang/index.html'},
 {'date': 'Sep 11, 2018',
 'headline': "Apple is still struggling to sell iPhones in the world's hottest markets",
 'url': 'http://money.cnn.com/2018/09/11/technology/apple-iphone-2018-india/index.html'},
 {'date': 'Sep 11, 2018',
 'headline': "China's troubles push Hong Kong stocks into a bear market",
 'url': 'http://money.cnn.com/2018/09/11/investing/hang-seng-bear-market/index.html'},
 {'date': 'Sep 10, 2018',
 'headline': 'The long-running CBS show is over for its longtime leading man',
 'url': 'http://money.cnn.com/2018/09/10/media/les-moonves-bill-carter/index.html'},
 {'date': 'Sep 10, 2018',
 'headline': "China's exports are slowing. The trade war will make things worse",
 'url': 'http://money.cnn.com/2018/09/10/news/economy/china-exports-trade-war/index.html'},
 {'date': 'Sep 10, 2018',
 'headline': "Moonves speaks after CBS breakup, says he's 'deeply saddened'",
 'url': 'http://money.cnn.com/2018/09/10/media/reliable-sources-09-09-18/index.html'},
 {'date': 'Sep 10, 2018',
 'headline': 'How Jack Ma went from English teacher to tech billionaire',
 'url': 'http://money.cnn.com/2018/09/09/technology/business/jack-ma-alibaba-bio/index.html'}]

```

How to find its css? When you open chrome devtools, you can find the css in the style console by corresponding to the elements in the webpage.

## Results for 'trade war'

View: All Results (9333) | Articles (9046) | Videos (287) Sort by: Date

[The trade war is already hurting US companies in China](#)

Nearly two-thirds of US firms who responded to a survey by American chambers of commerce in China reported that the waves of new tariffs have harmed their business.

Sep 13, 2018

[There's no way to avert the next financial crisis, warn former regulators from 2008 meltdown](#)

A decade after the financial crisis, there's no telling when the next calamity will strike, according to three regulators who helped steer the country through the meltdown a decade ago.

Sep 13, 2018

[America is now the world's largest oil producer](#)

For the first time since 1973, the United States is the world's largest producer of crude oil.

Sep 12, 2018

Remove any dependency to the following hidden dropdown from the JS

```

-->
<div class="cnnSourceCat" style="display:none;"></div>
<div class="clearFloat"></div>
<div class="cnnSearchResults"></div>
<div id="modquoteinfo" class="mod-quoteinfo" style="display: none;"></div>
<div id="cnnSearchSpotlight" class="summaryList"></div>
<!-- end of .cnnSearchResults -->
<div id="summaryList_mixed" class="summaryList" style="display: block;">
 <div class="summaryBlock">
 <div class="cnnHeadline">
 == $0
 </div>
 </div>

```

Styles Event Listeners DOM Breakpoints Properties Accessibility

Filter :hov .cls +

```

element.style {
}
.cnnHeadline a { cnnmoney.search-min.css:12
 ✓ line-height: 8px;
 ✓ padding: 0;
 ✓ margin: 0;
}
.cnnHeadline a { cnnmoney.search-min.css:2
 color: #151515;
}

```

margin - border - padding - auto x auto - - -

**Advanced version: all pages**

```

url = 'http://money.cnn.com/search/index.html?sortBy=date&primaryType=mixed&search=Search&query=trade%20war'

def get_articles_from_browser(b):
 articles = []
 for block in b.find_by_css('#summaryList_mixed .summaryBlock'):
 article = {}
 h = block.find_by_css('.cnnHeadline a')
 article['headline'] = h.text
 article['url'] = h['href']
 article['date'] = block.find_by_css('span.cnnDateStamp').text
 articles.append(article)
 return articles

Launch the initial page#
browser = Browser('chrome')
browser.visit(url)
time.sleep(2)

all_page_articles = []
for i in range(50): #scrape 50 pages
 time.sleep(0.5)
 try:
 new_articles = get_articles_from_browser(browser)
 all_page_articles.extend(new_articles)
 browser.execute_script('window.scrollTo(0, document.body.scrollHeight/1.5);') #scroll down
 next_buttons = browser.find_by_css('.pagingLinks li.ends.next')
 next_buttons[0].click() #splinter find_by return a list, therefore we need use index0 to access the next button.
 except Exception as e:
 print(e)
 print('Error on page %s' % i)

import pandas as pd
df = pd.DataFrame(all_page_articles)
df

```

Output: There will be 500 rows.

	<b>date</b>	<b>headline</b>	<b>url</b>
0	Sep 13, 2018	The trade war is already hurting US companies ...	http://money.cnn.com/2018/09/13/news/economy/a...
1	Sep 13, 2018	There's no way to avert the next financial cri...	http://money.cnn.com/2018/09/12/news/economy/f...
2	Sep 12, 2018	America is now the world's largest oil producer	http://money.cnn.com/2018/09/12/investing/us-o...
3	Sep 11, 2018	Is the tech stock rally coming to an end?	http://money.cnn.com/2018/09/11/technology/tec...
4	Sep 11, 2018	Apple is still struggling to sell iPhones in t...	http://money.cnn.com/2018/09/11/technology/app...
5	Sep 11, 2018	China's troubles push Hong Kong stocks into a ...	http://money.cnn.com/2018/09/11/investing/hang...
6	Sep 10, 2018	The long-running CBS show is over for its long...	http://money.cnn.com/2018/09/10/media/les-moon...
7	Sep 10, 2018	China's exports are slowing. The trade war wil...	http://money.cnn.com/2018/09/10/news/economy/c...
8	Sep 10, 2018	Moonves speaks after CBS breakup, says he's 'd...	http://money.cnn.com/2018/09/10/media/reliable...
9	Sep 10, 2018	How Jack Ma went from English teacher to tech ...	http://money.cnn.com/2018/09/09/technology/bus...
10	Sep 09, 2018	Les Moonves is out at CBS after harassment all...	http://money.cnn.com/2018/09/09/media/les-moon...
11	Sep 07, 2018	US jobs report; British Airways hack; Bitcoin ...	http://money.cnn.com/2018/09/07/investing/prem...
12	Sep 06, 2018	Hong Kong now has more super-rich people than ...	http://money.cnn.com/2018/09/06/investing/wor...
13	Sep 05, 2018	Why Wall Street shouldn't worry about the trad...	http://money.cnn.com/2018/09/05/investing/mark...

## Bonus: Twitter example with browser emulation

After we can handle browser emulation to find and extract data from a dynamic loading webpage, we can further apply this method to crawl some data from social media platforms, like the recent hot topic discussed right now on Twitter, to further analyze people's comments and opinions about certain events. The following are some pointers that may be useful for you to manipulate browser emulation with Twitter:

1. First you need to Simulate the login process
2. Do some navigating, searching, scrolling action to load more contents and tweets you want
3. Extract tweets by different finding elements method

Here are the common issues when scraping those social media platform:

1. Here is strong limitation to the data you can get. For example, after the certain point, the browser window and not be scrolled and there is only a `back to top` button at the bottom of the page.
2. The scraping results from webpage end may be different from the mobile end. For example, <https://twitter.com/> and <https://mobile.twitter.com/home>. Because Twitter has different regulations to different platforms.

Here is the [sample codes](#) we did with selenium browser emulation. We scrape tweets by keyword searching `Mangkhut`, the typhon that stroke Hong Kong and nearby region in 2018-09-16.

## Analyse Network Traces

Open "Google Chrome Developer Console" by `command+option+i`. Check out the "Network" tab and use "XHR" filter to find potential packages. We are usually interested in `json` files or `xml` files sent over the line. Most dynamic web pages have predictable / enumerable internal API -- use HTTP request to get certain `json` / `xml` files.

Some websites render HTML at the backend and send them to the frontend in a dynamic way. You find the URL in address bar stays the same but the content is changed. You can also find the HTML files and their **real URL** via developer console. One such example is [xiachufang.com scraper](#). Another one [here](#) scraped Centaline Property's buildings on sale and schools around the buildings use the same way.

Another common case is "infinite scroll" design. When a page adopts an infinite scroll design, asynchronous data loading is inevitable. When you encounter those cases, network trace analysis may give more concise solution. There are usually `xhr` interfaces. You don't even need dynamic crawling (browser simulation). One example is [mafengwo.com's user history](#).

## Bonus: Crawl mobile Apps

With the explosion of mobile Apps, more and more data is shifted from the open web to mobile platform. The design principle of web and mobile are very different. When Tim Berners Lee initially designed the WWW, it was intended to be an open standard that every one can connect to. That is why, once the web server is up, you can use Chrome to access it while other users may use Firefox or even Python `requests`. There are many tools to emulate browser activities, so you can programmably do the same thing as if a regular user is surfing the Internet. Compare with the open web, mobile world is a closed eco system. It often requires heavy duty packet analysis, App decompilation, or App emulation, in order to get data behind the mobile Apps. "Packet analysis" is most close to our course and is elaborated below.

### Packet analysis

This section is very similar to earlier [Analyse Network Traces](#). The only difference is that we analyse the mobile App packet here.

No matter how mysterious a mobile App seems to be, it has to talk to a server in order to get updated information. You can be assured that everything you see from your smart phone screen comes from either of the two channels:

1. Embedded in the phone, i.e. in the operating system, or in the App when you initially install
2. Loaded via the Internet upon certain user operation, e.g. App launch, swipe left, touch, ...

Channel 1 is the topic of next section. Channel 2 is what we are going to tackle. The idea is to insert a sniffer between the App and the backend server. In this way, whatever conversation the App has with the server will pass the sniffer first. The sniffer is also called "man-in-the-middle (MITM)", and a famous attack is named after this. You may have also heard the term "proxy", which intercepts your original network packet, modify it somehow, and then send the packet to the destination. One can use proxy to bypass Internet censorship or use proxy to hide the original sender's address. Our key tool is a MITM proxy. Here are two common choices

- "Charles proxy" -- Its GUI is very convenient for further packet analysis. It also has iOS and MAC clients. The software is not free though.
- `mitmproxy` -- You can install it via `pip`. It is free and open source. It provides command line interface to help intercept and dump packets. Recent version also provides web interface to browse the sniffed packets.

## Example: Kwai (kuaishou)

[Kuaishou](#) is a popular video sharing platform originated from China. We analyse its international version, [kwai](#), and scrape the top players data.

We'll omit the configuration of Charles Proxy on iOS and MAC, because there are numerous resources online and the interfaces are always changing. Once you finish configuration, do the following steps:

- Start sniffing in Charles Proxy on iOS.
- Open Kwai App.
- Browse like a normal user. Note that the packet sniffer can only intercept the conversations that happened. So you want to trigger more actions.
- Quit Kwai App.
- Send sniffed packet traces to MAC for further analysis.

There is no direct formula for packet analysis. We usually observe the request/ response sequence by time. For example, if you "pull down" to refresh the video list at 10th second, then the relevant packets are very likely to be sent around 10th second. You can find that data is obtained from an endpoint called `http://api.kwai.com/`. Specifically, the App sends HTTP requests to `http://api.kwai.com/rest/n/feed/hot` in order to obtain a list of hot videos. In our previous scraper examples, HTTP request is usually sent using the `GET` method. In the case of Kwai, `POST` is used.

A complete `POST` request is composed of three parts:

- headers -- send in HTTP protocol; users can not see
- params -- usually appears as `?a=3&b=5` in browser bar; `a` and `b` here are called parameters
- data -- the `POST` body; this is the main content to be consumed by the web server; based on this content, the server give corresponding response.

Charles Proxy's MAC software can help you to convert one HTTP request into the Python language, with the above three parts filled -- that is, give you the Python code that can **replay** one request. The variable configurations are as follows, with certain fields masked to preserve privacy:

```
headers = {
 'Host': 'api.kwai.com',
 ...
 'Accept': 'application/json',
 'User-Agent': 'kwai-ios',
 'Accept-Language': 'en-HK;q=1, zh-HK;q=0.9, zh-Hans-HK;q=0.8',
}
```

```

params = (
 ('appver', '5.7.3.494'),
 ...
 ('c', 'a'),
 ('ver', '5.7'),
 ('sys', 'ios11.4'),
 ('mod', 'iPhone10,3'),
 ...
)

data = [
 ...
 ('coldStart', 'true'),
 ('count', '20'),
 ('country_code', 'hk'),
 ('id', '13'),
 ('language', 'en-HK;q=1, zh-HK;q=0.9, zh-Hans-HK;q=0.8'),
 ('pv', 'false'),
 ('refreshTimes', '0'),
 ('sig', ...),
 ('source', '1'),
 ('type', '7'),
]

```

Here's the request operation and its outcome:

```

In [3]: response = requests.post('http://api.kwai.com/rest/n/feed/hot', headers=headers, params=params, data=data)
df = pd.DataFrame(response.json()['feeds'])

In [5]: df[['view_count', 'comment_count', 'like_count', 'caption', 'kwaid', 'user_name', 'user_sex', 'timestamp',]]
Out[5]:
 view_count comment_count like_count caption kwaid user_name user_sex timestamp
0 4309018 0 254393 3D水墨大片《上合之合》 Kwai_Positive_Energy 快手正能量 M 1528341685218
1 435265 952 7939 #穿在身上的非遗# 2018“锦绣中华——中国非物质文化遗产服饰秀”之白鹭为霜 NaN 光明网 M 1528281401483
2 1094241 2865 20876 2018首对大熊猫龙凤胎6月5日早上在成都大熊猫繁育研究基地诞生了! 🎉 [胜利] 你觉得熊猫宝... NaN 人民网 M 1528277614426
3 491762 3112 24136 #高考加油#\n高考, 你从来不是一个人在战斗\n\n当你在为高考努力奋斗时, 是否忽略了身... NaN 靠谱青年 U 1528285262624
4 1252430 3447 41799 还好跑得快 吓死宝宝了\n\n @小蚊子(O7199076) xiaozhizhi0 陈思婷 F 1528288566766
5 927683 973 13326 不会跳蒙古舞。搞笑版。乱蹦跶的。要把我笑死 grx473371609 白川茉莉 F 1528201413086
6 1576546 2038 36726 累死了😂\n双击评论666 @允大哥(0939855539) yaqian1314520 小牙签 F 1527335519652
7 1276101 2322 26261 秒化.... Zy1314528 小宇糖果君 M 1528283993829
8 962724 1461 12779 这个动作可以加强脊柱侧弯的能力, 瘦腰, 开肩, 美化背部。左右侧练习。每组保持10-20个呼吸。 NaN 瑜伽生活的我 F 1528205965235
9 5449202 8316 86213 第一时间你们联想到了啥子, 给个爱心吧\n\n V5888866666 艺术背景墙小伟 M 1527060092852
10 1642969 3125 15131 ... zhao24577546 阿狸小厨房 F 1528285523225

```

Note the `pd.DataFrame` is a `pandas` object, which will be explained in [notes-week-09.md](#).

## App de-compilation

This means to "crack" the App. You need to first get the installation package of the App, analyse its structure, decompile it, and understand how this App talk with a server from its source code.

Sophisticated App will embed certain cryptography routine in the App and authenticate itself with the server. Even if you successfully analysed the network packet, it is hard for you to come up with the correct authentication parameters. In order to understand how this authentication process is conducted, you may want to reverse engineer the App.

Further discussion is omitted here because this part takes years of computer science background, especially in information security domain.

## App emulation

[Appium](#) is a frequently used automatic testing tool. You can use this tool to emulate user operations on mobile Apps and scrape the data from the screen. For Android users, [Auto.js](#) is a convenient library that relies on accessibility features and does not require root access.

Actually, `selenium`, we introduced earlier in this chapter, was initially also an automatic testing tool for the web frontend. Then it became a bridge between the programmable user and web browser driver, which was used in a lot scraping works. When you find yourself stuck with data access because of non-human behaviour (e.g. anti-crawling), you can try to search the keywords "emulation" or "auto testing", and can usually get some pointers to useful tools.

## Bonus: Other quick scraping/ crawling tricks

In our class, we show you the very basic steps of scraping so that you know how things happen in a sequential way. However, you don't have write codes for everything from scratch in real practice. People already made numerous tools and libraries that can help you do certain tasks quickly. Here are some examples related with course (Shell/ Python) for those who are interested:

- Type `wget -r {url}`, where `{url}` is the URL of the website you want to crawl. After running this command, you can find all the web pages and their dependent resources are on your computer. You can fine tune the parameters to limit crawling scope, like number of hops or types of files. Use `man wget` to find out more.
- There are many shell commands which can be combined to perform efficient text processing. [This article](#) shows how one can combine a few Shell commands to quickly download the Shakespeare works.
- [This repo](#), originally a workshop given on PyConHK in 2015, shows you some handy tools and libraries in Python that allow one to scrape more with less codes. For example, you can use `readability` to extract the main body of an HTML page, without bothering with its page structure. For readers with frontend development background, `pyquery` is a handy library to allow you write jQuery like selectors to access HTML elements. `scrapy` is a machine learning based library that can learn the labelled crawling target and generate corresponding rules; The user only needs to tell `scrapy` what to crawl, instead of how to crawl.
- [Data Science at the Command Line](#) by Jeroen Janssens is a comprehensive and duly updated reference book for command line tools for data science. Its [Obtaining data](#) is a good further reading for those who are interested in more efficient data collection in Linux shell environment.

## Exercises and Challenges

### In-bound marketing and SEO auditing

Search Engine Optimization (SEO) is one common technique a digital marketer needs to master. Suppose you have led a team to conduct the optimization. Now it is time to audit the optimization result. One of the key function is to build scraper which can:

- Input 1 is a search query, i.e. some keywords
- Input 2 is a set of URLs from your own website
- Output the ranks of each URL in the search result list

### Crawl the legal case of China

<http://wenshu.court.gov.cn> collects the legal cases in China. It supports advanced search options. One can emulate browser to download relevant documents on a certain area. Please try:

- Give a keyword as input.
- Download the documents of the first page, e.g. `.docx` files, onto local disk.

- Organise an index of those documents into a `csv` which may include "title", "court", "date", "document-path", and other fields if you deem useful.

## Bonus: Crawl Weibo data and discover KOL

Key Opinion Leader (KOL) is the goto person for targeted massive marketing. As a marketing specialist, you want to identify the KOLs in a certain area so that your team can reach out to them effectively. Before learning sophisticated graph mining algorithms, one can do the follow challenge to get some preliminary result:

- Given an industry domain, identify `keywords`
- For every `keyword` in `keywords`, scrape the search of related micro blogs.
- Every piece of microblog may have following data structure:

```
microblog = {
 'username': 'DATA HERE',
 'datetime': 'DATA HERE',
 'text': 'DATA HERE',
 'num_like': 'DATA HERE',
 'num_comment': 'DATA HERE',
 'num_share': 'DATA HERE'
}
```

- A simple algorithm to find KOL is to count `num_like`, `num_comment`, `num_share` for each `username`.

## Bonus: Cheat an online voting system

Some people host competitions online and calculate the leaderboard based on web traffic, like page visits, number of clicks of "upvote" and so on. The system is very easy to cheat if it does not adopt CAPTCHA system. After this week, you can use Python to emulate user behaviour and cheat those systems. Here are some examples for your reference:

- Increase Youtube page views by refreshing browser page [code](#)

Please find another system/ another parameter from the system, which you can cheat using similar tricks.

## Related Readings

- Dynamic loading and crawling example: [libguides example](#)
- Social media crawling example: [Scrape a luxury brand with keyword in Weibo](#)
- Dynamic page crawling, with a matter of parsing page content: [Timeout](#)
- Dynamic crawling a static page with a matter of pagination: [Amazon Books](#)
- A high school student writes Selenium Chinese documents [SELENIUM 的中文文档](#)

---

If you have any questions, or seek for help troubleshooting, please [create an issue here](#)

# Week 09: Work with table - data cleaning and pre-processing

- [Week 09: Work with table - data cleaning and pre-processing](#week-09-work-with-table---data-cleaning-and-pre-processing) - [Objective](#objective) - [Preparation](#preparation) - [Python environment](#python-environment) - ['pandas` introduction](#pandas-introduction) - [Pandas Series](#pandas-series) - [Convert between list/dict with series](#convert-between-listdict-with-series) - [list<-->series](#list--series) - [dict<-->series](#dict--series) - [Get quick stats of series](#get-quick-stats-of-series) - [Series.sort\_values](#seriesort\_values) - [Series.sum](#seriessum) - [Slice series](#slice-series) - [Reference to elements in series](#reference-to-elements-in-series) - [Benefits of using series](#benefits-of-using-series) - [Pandas Dataframe](#pandas-dataframe) - [Data Loading](#data-loading) - [Load table (DataFrame) from local csv file](#load-table-dataframe-from-local-csv-file) - [Load table (DataFrame) from a URL](#load-table-dataframe-from-a-url) - [Select data](#select-data) - [Select columns with []](#select-columns-with-) - [Select rows with .loc and .iloc](#select-rows-with-loc-and-iloc) - [Basic statistics](#basic-statistics) - [DataFrame.describe()](#dataframedescribe) - [Count values of series](#count-values-of-series) - [Plot a simple chart: histogram](#plot-a-simple-chart-histogram) - [Sorting: DataFrame.sort\_values](#sorting-dataframesort\_values) - [Transformation: DataFrame.apply and Series.apply](#transformation-dataframeapply-and-seriesapply) - [lambda: anonymous function](#lambda-anonymous-function) - [Filtering](#filtering) - [Filter by numeric range](#filter-by-numeric-range) - [Filter by exact value](#filter-by-exact-value) - [Filter by more than two conditions](#filter-by-more-than-two-conditions) - [Save data](#save-data) - [to\_csv](#to\_csv) - [to\_dict](#to\_dict) - [to\_json](#to\_json) - [Bonus: from Python pandas and Javascript visualization](#bonus-from-python-pandas-and-javascript-visualization) - [Dataprep](#dataprep) - [Cleaning](#cleaning) - [Transformation](#transformation) - [Extraction](#extraction) - [References](#references)

This week, we will step from data collecting process into analyzing process. We will learn a new module called `pandas`, with which, we can do tasks like data cleaning, pre-processing, filtering, and even simple visualizations. In this week, we will only cover some basic usage of `pandas`, advanced data analyzing and data manipulating will be touched in the following weeks, stay tuned.

## Objective

- Master the schema of data pre-processing: cleaning, transforming, extracting
- Can efficiently manipulate structured table formatted datasets
- Use `pandas` for basic calculation and plotting

Modules:

- `pandas`

Datasets to work on:

- `openrice_sample.csv`

In this chapter, we will not cover the specific scraping demo but basic usage of `pandas`. Interested students can refer to [here](#) for scraping process.

---

## Preparation

### Python environment

Please install libraries/dependencies in your virtual environment:

```
pip install pandas, requests, csv
if you've already installed, just ignore
```

If you install in Jupyter notebook, you can prefix the command with `!` in order to execute those commands in a Jupyter notebook cell.

## pandas introduction

Pandas is an open source library providing easy-to-use data structures and data analysis tools for the Python programming language, enabling you to carry out your entire data analysis workflow in Python without having to switch to a more domain specific language like R. For easy and light weighed data analysis, pandas is our best choice.

There are two basic data structures:

### Pandas Series

A series is a one-dimensional object that can hold any data type such as integers, floats and strings. Simply, series is like a single column of a DataFrame.

Example 1:

```
import pandas as pd
x = pd.Series([1,0,2,8])
x
0 1
1 0
2 2
3 8
dtype: int64
```

The first axis is referred to as the index. Also, we can define indexes for the data by our own way.

```
x = pd.Series([1,0,2,8], index=['a', 'b', 'c', 'd'])
x
a 1
b 0
c 2
d 8
dtype: int64
```

### Convert between list/dict with series

#### list<-->series

Example:

```
#import pandas as pd
l = ['火鍋', '海鮮', '甜品/糖水', '壽司/刺身', '日式放題', '烤肉', '薄餅']
list_series = pd.Series(l)
list_series
```

```
0 火鍋
1 海鮮
```

```
2 甜品/糖水
3 壽司/刺身
4 日式放題
5 烤肉
6 薄餅
dtype: object
```

Convert series back to list

```
list_series.tolist()
```

### dict<-->series

Converting between dict and series is pretty much the same like converting between list and series.

```
word_dict = {'火鍋':39, '海鮮':28, '甜品/糖水':24, '壽司/刺身':14, '日式放題':11, '烤肉':10, '薄餅':9}
dict_series = pd.Series(word_dict)
```

```
火鍋 39
海鮮 28
甜品/糖水 24
壽司/刺身 14
日式放題 11
烤肉 10
薄餅 9
dtype: int64
```

Convert series back to dict

```
dict_series.to_dict()
```

## Get quick stats of series

### Series.sort\_values

Sorting values in ascending or descending order.

```
words_dict = {'火鍋':39, '壽司/刺身':14, '甜品/糖水':24, '日式放題':11, '薄餅':9, '烤肉':10, '海鮮':28}
dict_series = pd.Series(words_dict)
dict_series.sort_values(ascending=False)
```

Output:

```
火鍋 39
海鮮 28
甜品/糖水 24
壽司/刺身 14
日式放題 11
烤肉 10
薄餅 9
dtype: int64
```

### Series.sum

Use the above dict\_series as an example:

```
dict_series.sum()
```

Output:

```
135
```

## Slice series

Slicing series is the same as slicing list, just give the index interval can work. Also take the above example:

```
dict_series[:4]
```

Output:

```
火鍋 39
壽司/刺身 14
甜品/糖水 24
日式放題 11
dtype: int64
```

## Reference to elements in series

Reference to elements in series is pretty much the same as doing it in dict. each element in series is like the key in dict.

```
dict_series['火鍋']
```

Output:

```
39
```

## Benefits of using series

Series has the advantage from both `dict` and `list`. One can use a key to reference to the elements, which leads a `dict` like look and feel. However, for a regular `dict`, keys do not have certain ordering. Python only guarantees `dict.keys()`, `dict.values()` and `dict.items()` adopts the same ordering but how they are ordered is not specified. `pandas.Series` can preserve the order of elements, which gives a `list` like look and feel.

## Pandas Dataframe

A DataFrame is a two dimensional object that can have columns with different types,dictionaries, lists, series etc... Dataframe is the primary pandas data structure.

Example 3: [2017 Hong Kong population](#). There are several series, one can merge them to a dataframe.

```
land_area = pd.Series({
 'Hong Kong Island': 79.92,
 'Kowloon': 46.94,
 'New Territories and Islands': 954.69
})
mid_year_population = pd.Series({
 'Hong Kong Island': 1248.5,
 'Kowloon': 2256.1,
 'New Territories and Islands': 3886
})
population_density = pd.Series({}
```

```

 'Hong Kong Island': 15620,
 'Kowloon': 48060,
 'New Territories and Islands': 4070,
 })
hongkong_population_distribution = pd.DataFrame({
 'Land Area (sq. km)': land_area,
 'Mid- year Population (''000)': mid_year_population,
 'Population Density (Persons per sq. km)': population_density
})
hongkong_population_distribution

```

Output:

	Land Area (sq. km)	Mid- year Population ('000)	Population Density (Persons per sq. km)
Hong Kong Island	79.92	1248.5	15620
Kowloon	46.94	2256.1	48060
New Territories and Islands	954.69	3886.0	4070

## Data Loading

### Load table (DataFrame) from local csv file

First of all, you need to download the csv file from [here](#). For how to download the file, you can refer to [here](#).

Put csv file into the same folder with Jupyter notebook. You can type `!pwd` to check out where it is and put the file in this path.

```

import pandas
pandas.read_csv('openrice_sample.csv')

```

The output will be as below:

In [3]:	import pandas pandas.read_csv('openrice_sample.csv')	Out[3]:	LAB EAT Restaurant & Bar	尖沙咀金巴利道87-89號僑豐大廈地下1-2號舖	\$201-400	西式	海鮮	436	(565 食評)	50744	網上訂座可享75折優惠
		0	Shine	尖沙咀北京道12A號太子集團中心6樓	\$201-400	西式	酒	693.0	(777 食評)	21136	送 25里數 / 30積分
		1	Yadllie Plate	旺角西洋菜街1號兆萬中心11樓	\$101-200	韓國 菜	韓式 炸雞	487.0	(630 食評)	27681	早鳥及消夜時段訂座 可享全單八折優惠
		2	漁獲浜燒 Toretore Hamayaki	銅鑼灣軒尼詩道525號澳門逸園中心18樓	\$201-400	日本 菜	日本 菜	503.0	(524 食評)	19647	送 25里數 / 30積分
		3	心之食堂 Love Cafe	銅鑼灣耀華街3號百樂中心12樓1203室	\$201-400	日本 菜	壽司/ 刺身	490.0	(545 食評)	6580	全單85折
		4	The Grill Room	銅鑼灣駱克道459-461號The L. Square 5樓	\$201-400	西式	扒房	508.0	(662 食評)	32342	網上訂座八折優惠-商務二人或四人套餐
		5	Sky726	旺角彌敦道724-726號25樓	\$201-400	法國 菜	甜品/ 糖水	369.0	(496 食評)	38563	自選餐牌主菜買一送一優惠 或 甜品8折優惠： 飲品一律8折

If there is no header in the csv file. We can use `Pandas` as below to add proper headers for a form.

```

df = pandas.read_csv('openrice_sample.csv', header=None, names=['name', 'location', 'price', 'country', 'type', 'likes', 'review', 'bookmark', 'discount_info'])
`df` is short for "dataframe", which is usually used as return value in pandas.

```

### Load table (DataFrame) from a URL

We can also load CSV from GitHub directly with the help of `requests` and `io.StringIO`. `io` module is used for dealing with various types of I/O (input/output). Due to the requirement that `pandas.read_csv` function needs a `file-like object` as the argument, we need to use `io.StringIO` to write and store those string, returned from the request, temporarily. Then we can read the csv in pandas. Interested students can find more info about `io` module in [here](#).

**Note:** We should use `raw data url` from github page instead of preview url.

```
import pandas as pd
import io
import requests
url="https://raw.githubusercontent.com/hupili/python-for-data-and-media-communication/master/scraping-examples/openrice/openrice_sample.csv"
s=requests.get(url).content
df=pd.read_csv(io.StringIO(s.decode('utf-8')),header=None, names=['name', 'location','price','country','type','likes','review','bookmark','discount_info'])
```

## Select data

First of all, every time we load a csv in Jupyter, always print the first several rows to have a overview what's look like. It's useful and necessary because sometimes the dataset can be really large, if you print the whole table, your browser might get crushed. We can use following command to check out the records here.

```
df.head() #displaying first 5 rows by default, you can pass the number in () to show more/less rows.
```

the output will be as blow:

		name	likes	location	price	country	style	review	bookmark	discount_info
0	LAB EAT Restaurant & Bar	436	尖沙咀金巴利道87-89號僑豐大廈地下1-2號舖	\$201-400	西式	海鮮	(565 食評)	50664		網上訂座可享75折優惠
1	Shine	692	尖沙咀北京道12A號太子集團中心6樓	\$201-400	西式	酒	(776 食評)	21069		送 25里數 / 30積分
2	Espuma	610	尖沙咀厚福街8號H8 2樓	\$101-200	西班牙菜	甜品/糖水	(949 食評)	40374		送 25里數 / 30積分
3	The Captain's House	511	尖沙咀厚福街8號H8 18樓	\$201-400	西式	海鮮	(634 食評)	33139		送 25里數 / 30積分
4	Yadlie Plate	487	旺角西洋菜街1號兆萬中心11樓	\$101-200	韓國菜	韓式炸雞	(630 食評)	27542		早鳥及消夜時段訂座 可享全單八折優惠
5	Day and Nite by Master Kama	462	旺角山東街50號1-2樓	\$101-200	日本菜	海鮮	(595 食評)	28151		送 25里數 / 30積分
6	漁獲浜燒 Toretore Hamayaki	503	銅鑼灣軒尼詩道525號澳門逸園中心18樓	\$201-400	日本菜	日本菜	(524 食評)	19622		送 25里數 / 30積分
7	The Grill Room	508	銅鑼灣駱克道459-461號The L. Square 5樓	\$201-400	西式	扒房	(662 食評)	32287	訂座驚喜星期六及日: 套餐\$238起-全日酒精飲品 Happy Hour 價錢	
8	心之食堂 Love Cafe	487	銅鑼灣耀華街3號百樂中心12樓1203室	\$201-400	日本菜	壽司/刺身	(542 食評)	6551		送 25里數 / 30積分
9	Sky726	367	旺角彌敦道724-726號25樓	\$201-400	法國菜	甜品/糖水	(494 食評)	38441	自選餐牌主菜買一送一優惠 或 甜品8折優惠; 飲品一律8折	

There are several ways to select data. We only focus on the most used ones. `[]`, `.loc` and `.iloc`. Collectively, they are called the indexers.

### Select columns with []

`[]` method is mainly used for selecting single column and multiple columns. Basically, `[]` method treat the dataframe as a dict, using a key to refer to certain value. If you want to select one column of data, just simply put the name of the column in-between the brackets. For example, you want all the restaurant locations. You can type:

```
df['location']
```

Then the output will be as below:

```
In [79]: df['location']
```

```
Out[79]: 0 尖沙咀金巴利道87-89號僑豐大廈地下1-2號舖
1 尖沙咀北京道12A號太子集團中心6樓
2 旺角西洋菜街1號兆萬中心11樓
3 銅鑼灣軒尼詩道525號澳門逸園中心18樓
4 銅鑼灣耀華街3號百樂中心12樓1203室
5 銅鑼灣駱克道459-461號The L. Square 5樓
6 旺角彌敦道724-726號25樓
7 尖沙咀彌敦道132號美麗華廣場一期2樓 207 號舖
8 尖沙咀北京道65-69號環球商業大廈2樓205-06室
9 銅鑼灣開平道1號Cubus 21樓
10 尖沙咀廣東道3-27號海港城海連大廈3樓LCX 34號舖
11 佐敦長樂街23-27號德威大廈地下3號舖
12 尖沙咀棉登徑11號地舖
13 旺角彌敦道726號18樓全層
14 油麻地彌敦道483-485號2樓
15 銅鑼灣軒尼詩道525號澳門逸園中心17樓
16 尖沙咀松山道15-17號地下D舖
17 旺角通菜街1號威達商業大廈地下E, F & H號舖
18 旺角砵蘭街106號地下
19 旺角彌敦街33號朗廷軒地下B舖
20 銅鑼灣波斯富街99號利舞臺廣場18樓A號舖
```

You can find that the data type of the results returned is changed. Using `type(df['location'])` to check out what it is. If you want to keep the it with the `dataframe`, you can write the above code as this: `df[['location']]`.

To select multiple columns of the data, you can pass it a list of column names.

```
df[['name', 'location', 'likes']]
```

The output will be like this:

```
In [12]: df[['name', 'location', 'likes']]
```

```
Out[12]:
```

	name	location	likes
0	LAB EAT Restaurant & Bar	尖沙咀金巴利道87-89號僑豐大廈地下1-2號舖	436
1	Shine	尖沙咀北京道12A號太子集團中心6樓	692
2	Espuma	尖沙咀厚福街8號H8 2樓	610
3	The Captain's House	尖沙咀厚福街8號H8 18樓	511
4	Yadllie Plate	旺角西洋菜街1號兆萬中心11樓	487
5	Day and Nite by Master Kama	旺角山東街50號1-2樓	462
6	漁獲浜燒 Toretore Hamayaki	銅鑼灣軒尼詩道525號澳門逸園中心18樓	503
7	The Grill Room	銅鑼灣駱克道459-461號The L. Square 5樓	508
8	心之食堂 Love Cafe	銅鑼灣耀華街3號百樂中心12樓1203室	487
9	Sky726	旺角彌敦道724-726號25樓	367

### Select rows with .loc and .iloc

The `.loc` indexer will return a single row as a Series when given a single row `label`, and return DataFrame if multiple rows are selected.

Example:

```
df = pd.DataFrame([[170, 60], [180, 82], [175, 70]],
 index=['Ri', 'Frank', 'Tyler'],
 columns=['height', 'weight'])
df
```

	height	weight
Ri	170	60
Frank	180	82
Tyler	175	70

```
df.loc['Frank'] #select one row
```

```
height 180
weight 82
Name: Frank, dtype: int64
```

```
df.loc[['Frank', 'Tyler']]
```

	height	weight
Frank	180	82
Tyler	175	70

Similarly, there is another function `.iloc`, which is purely integer-location based indexing for selection by position.

Using the `openrice.csv` as an example:

```
#read csv first, make sure the header is right, you can refer to previous content in this chapter.
df.iloc[5]
```

name	Day and Nite by Master Kama
likes	462
location	旺角山東街50號1-2樓
price	\$101-200
country	日本菜
style	海鮮
review	(595 食評)
bookmark	28151
discount_info	送 25里數 / 30積分
Name:	5, dtype: object

```
df.iloc[[5, 10, 15]]
```

In [15]: `df.iloc[5,10,15]`

Out[15]:

	name	likes	location	price	country	style	review	bookmark	discount_info
5	Day and Nite by Master Kama	462	旺角山東街50號1-2樓	\$101-200	日本菜	海鮮	(595 食評)	28151	送 25里數 / 30積分
10	Burgeroom	437	尖沙咀彌敦道132號美麗華廣場一期2樓 207 號舖	\$51-100	美國菜	漢堡包	(511 食評)	13781	送 25里數 / 30積分
15	Espuma	557	中環威靈頓街2-8號威靈頓廣場M88 17樓	\$101-200	西班牙菜	甜品/糖水	(690 食評)	20889	送 25里數 / 30積分

```
df.iloc[10:20]
```

In [17]: `df.iloc[10:15]`

Out[17]:

	name	likes	location	price	country	style	review	bookmark	discount_info
10	Burgeroom	437	尖沙咀彌敦道132號美麗華廣場一期2樓 207 號舖	\$51-100	美國菜	漢堡包	(511 食評)	13781	送 25里數 / 30積分
11	樂園 Happyland	572	尖沙咀北京道65-69號環球商業大廈2樓205-06室	\$101-200	西式	樓上cafe	(635 食評)	10078	送 25里數 / 30積分
12	極尚大喜屋日本料理 Deluxe Daikya Japanese Restaurant	384	尖沙咀加連威老道2-6號愛實商業大廈後座1樓102號舖	\$201-400	日本菜	日式放題	(507 食評)	48390	特選时段 85 折優惠！該優惠不得與其他優惠共用，適用於晚市19:30前離座及21:15後入...
13	The Hwaduk	450	銅鑼灣開平道1號Cubus 21樓	\$101-200	韓國菜	韓式炸雞	(498 食評)	16523	送 25里數 / 30積分
14	Cafe Paradise	383	太子界限街23號地下C舖	\$51-100	意大利菜	甜品/糖水	(473 食評)	25254	送 25里數 / 30積分

## Basic statistics

### DataFrame.describe()

Descriptive or summary statistics in python – pandas, can be obtained by using describe function. `describe()` function gives you a summary about the dataframe or certain series with the `mean`, `count`, `std` and `freq` values etc. Only the column with pure numbers can be described if you don't specify the column.

Example:

```
df.describe()
 likes bookmark
count 244.000000 244.000000
mean 249.094262 12596.356557
std 112.075938 8567.647276
min 66.000000 1430.000000
25% 165.000000 6741.750000
50% 215.500000 10755.500000
75% 309.500000 15464.500000
max 692.000000 52023.000000
```

```
df['style'].describe()
count 244
unique 47
top 火鍋
freq 39
Name: style, dtype: object
```

### Count values of series

After we can select one column or row, we can do further calculation or analysis. One common usage is to count different values in one column.

Example

```
df['country'].value_counts()
```

```
西式 47
日本菜 45
意大利菜 26
韓國菜 20
粵菜 (廣東) 18
港式 18
多國菜 17
泰國菜 10
台灣菜 10
西班牙菜 10
...
Name: country, dtype: int64
```

```
df['style'].value_counts()
```

```
火鍋 39
海鮮 28
甜品/糖水 24
壽司/刺身 14
日式放題 11
烤肉 10
自助餐 9
```

```

薄餅 9
咖啡店 8
All Day Breakfast 7
...
Name: style, dtype: int64

```

`value_counts()` function gives you a hint for further filter and data processing. For example, after you know the 火锅 is the most popular food type. We can do a filter that select all the restaurants in 火锅 and cross analysis it with likes, prices etc., which we will cover later in this chapter.

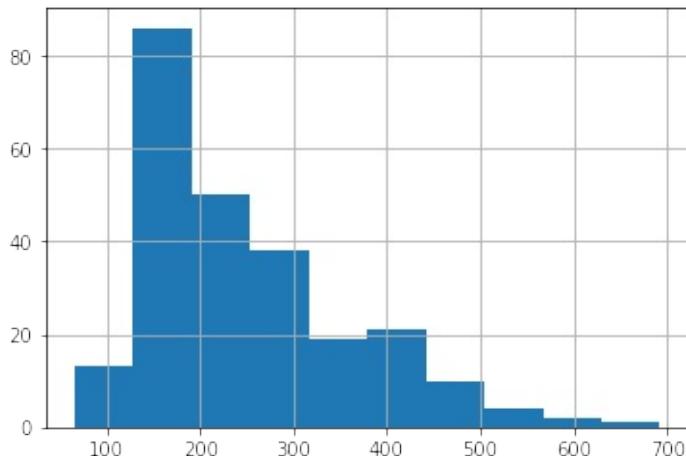
### Plot a simple chart: histogram

After we got the results from our analysis, the key point is to visualize them so that we can have a better understanding of the results and get insights from them. By using `.hist()` function, We can plot a simple histogram to show the distribution and trend.

Example:

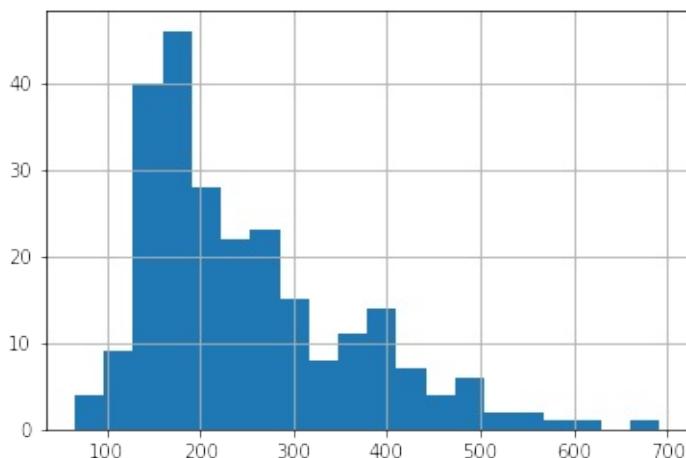
```
%matplotlib inline #this line is to solve the problem that histogram doesn't reveal.
df['likes'].hist()
```

and you can get a distribution like below:



You can change shape of the charts by changing the bins(basically, one bin means one column)

```
df['likes'].hist(bins=20)
```



## Sorting: DataFrame.sort\_values

Sort values in dataframe is similar to which in series. You can sort by different columns like the example:

```
df.sort_values(by='likes', ascending=False)
```

**Note:** In the `sort_values` function, we can only use `ascending` method, `descending` will not work here, which is designed by pandas documentation. But we can use `ascending=False` to meet `descending` needs.

What's more, in the multiple columns dataset, one may need to filter or sort values by multiple columns, we can use following method to accomplish this:

```
#the first false corresponding to the first column
df.sort_values(['likes','bookmark'], ascending=[False, False])
#you can also write as following, 0 means false, 1 means true
#df.sort_values(['likes','bookmark'], ascending=[0, 0])
```

		name	location	price	country	type	likes	review	bookmark	discount_info
1		Shine	尖沙咀北京道12A號太子集團中心6樓	\$201-400	西式	酒	693.0	(777 食評)	21136	送 25里數 / 30積分
8		樂園 Happyland	尖沙咀北京道65-69號環球商業大廈2樓205-06室	\$101-200	西式	樓上cafe	574.0	(637 食評)	10097	送 25里數 / 30積分
26		神戶屋日韓放題 Kobe Fusion Restaurant	旺角西洋菜街1號兆萬中心8樓	\$201-400	韓國菜	壽司/刺身	549.0	(675 食評)	16806	送 25里數 / 30積分
5		The Grill Room	銅鑼灣駱克道459-461號The L. Square 5樓	\$201-400	西式	扒房	508.0	(662 食評)	32342	網上訂座八折優惠-商務二人或四人套餐
3		漁獲浜燒 Toretore Hamayaki	銅鑼灣軒尼詩道525號澳門逸園中心18樓	\$201-400	日本菜	日本菜	503.0	(524 食評)	19647	送 25里數 / 30積分
4		心之食堂 Love Cafe	銅鑼灣耀華街3號百樂中心12樓1203室	\$201-400	日本菜	壽司/刺身	490.0	(545 食評)	6580	全單85折

## Transformation: DataFrame.apply and Series.apply

In the process of analyzing, we will encounter a lot of data cleaning issues, like the missing values, `NoneType` values or other type of values that have side effects, which need us to build different functions to handle them or convert them. For example, the price in the `openrice` is a range of number which cannot be compared directly. We need firstly clean the data and convert price range to numeric values.

If you need to compare price which is a interval. You need to pay special attention on numbers. Otherwise, Python recognize '\$101-200' < '\$51-100' because Python only compare the first number in sequence of each interval.

You need to convert each interval string into numbers, which means you need to choose a number to represent each interval to do comparison. Here, we use `mapping` function:

```
mapping = {
 '$101-200': 200,
 '$201-400': 400,
 '$51-100': 100,
 '$401-800': 800,
 '$50以下': 25
}
```

Now, build a function to handle those data.

```
original_string = '$60以下'
mapping.get(original_string, 0)
if those string is in the mapping dict, it will return the paired value, otherwise, it will return the value you set, in this case, is the second parameter 0.
```

```
def cleaning(e):
 return mapping.get(e, 0)
cleaning('$50以下')
```

**Note:** In the mapping function, it's not like the traditional dict - key & value like.

```
mapping = {'A': 'B'}
mapping.get('A')
```

Basically, it tells that we use B to replace A. For vast sum of data, we can use apply function to do cleaning. For example, if you have a series of data need to be mapping:

```
mapping =
{'A':'B',
'C':'D',
...,
'X':'Y'}
def cleaning(e):
 return mapping.get(e, 0)
pandas.series.apply(cleaning)
```

```
In [45]: original_string = '$60以下'
mapping.get(original_string, 0)
Out[45]: 0

In [48]: def cleaning(e):
 return mapping.get(e, 0)
cleaning('50以下')
Out[48]: 25
```

Then we can use `apply` function to do cleaning for the whole column. Basically, `<data>.apply(<function_name>)` means that pass those data one by one to call the function, which is equal to a `for` loop processing data.

```
df['price'].apply(cleaning) #clean the whole column

In [49]: df['price'].apply(cleaning)
Out[49]: 0 300
1 300
2 150
3 300
4 150
5 150
6 300
7 300
8 300
9 300
10 75
11 150
12 300
13 150
14 75
15 150
```

### lambda: anonymous function

lambda also called as anonymous function, which doesn't have a specific name, only need one line to declare function. The usual syntax is as follows:

```
lambda arguments : expression
```

There can be multiple arguments, and expression is the results it returns.

For example:

```
c = lambda x,y : x**2 + 4*y
```

```
c(4,2)
#24
```

x,y are arguments, and after the : is the results. This is equal to:

```
def f(x,y):
 return x**2 + 4*y
f(4,2)
```

The advantages of `lambda`:

- Using lambda omit the process of defining functions, which make the code more light.
- We can save time by using lambda without thinking about naming the function
- The power of lambda is better shown when you use them as an anonymous function inside another function.

For example: Build a filter lambda function inside a function

```
Program to filter out only the even items from a list

my_list = [1, 5, 4, 6, 8, 11, 3, 12, 9, 27, 15, 14, 17]
new_list = list(filter(lambda x: (x%3 == 0) , my_list))
new_list
Output: [6, 3, 12, 9, 27, 15]
```

## Filtering

In the above example, we learned how to access to the columns, raws and multiple data in the dataframe. For further analysis, we need to do some filtering work to help us better finding the insights. For example, how to select the most expensive restaurants and the least likeable restaurants? Is there any correlation between likes and types?...

For filtering, in pandas, we use the `df[ conditions ]` method. This is basically means, we first use `condition` to filter the data, and put it into data frame. For example, if you want to know how many restaurants having over 500 likes, and what are those restaurants.

```
df['likes'] > 500 #this is a condition filtering restaurants that received more than 500 likes.
```

```
In [80]: df['likes'] > 500
Out[80]: 0 False
 1 True
 2 False
 3 True
 4 False
 5 True
 6 False
 7 False
 8 True
 9 False
 10 False
 11 False
 12 False
 13 False
 14 False
 15 False
```

The results returned is a series true or false, true means meeting the condition, while false is not.

To get the information of those filtered restaurants, we use `df[ ]` outside of the `condition`.

```
df[df['likes'] > 500]
```

		name	location	price	country	style	likes	review	bookmark	discount_info
1		Shine	尖沙咀北京道12A號太子集團中心6樓	\$201-400	西式	酒	693.0	(777 食評)	21136	送 25里數 / 30積分
3		漁獲浜燒 Toretore Hamayaki	銅鑼灣軒尼詩道525號澳門逸園中心18樓	\$201-400	日本菜	日本菜	503.0	(524 食評)	19647	送 25里數 / 30積分
5		The Grill Room	銅鑼灣駱克道459-461號The L. Square 5樓	\$201-400	西式	扒房	508.0	(662 食評)	32342	網上訂座八折優惠-商務二人或四人套餐
8		樂園 Happyland	尖沙咀北京道65-69號環球商業大廈2樓 205-06室	\$101-200	西式	樓上 cafe	574.0	(637 食評)	10097	送 25里數 / 30積分
26		神戶屋日韓放題 Kobe Fusion Restaurant	旺角西洋菜街1號兆萬中心8樓	\$201-400	韓國菜	壽司/刺身	549.0	(675 食評)	16806	送 25里數 / 30積分

There are many ways to do filtering, the following are the common methods that are wildly used.

### Filter by numeric range

If there is multiple conditions in filtering, we need to use `()` to include each condition. For example, filter the `bookmark` between 50000-60000.

```
df[(df['bookmark'] < 60000) & (df['bookmark']>50000)]
```

		name	location	price	country	style	likes	review	bookmark	discount_info
0	LAB EAT Restaurant & Bar	尖沙咀金巴利道87-89號僑豐大廈地下1-2號舖	\$201-400	西式	海鮮	436.0	(565 食評)	50744	網上訂座可享75折優惠	
41	The Place	旺角上海街555號香港康得思酒店L樓層	\$401-800	多國菜	海鮮	239.0	(481 食評)	52153	全單75折	
236	The Place	旺角上海街555號香港康得思酒店L樓層	\$401-800	多國菜	海鮮	239.0	(481 食評)	52154	全單75折	
368	LAB EAT Restaurant & Bar	尖沙咀金巴利道87-89號僑豐大廈地下1-2號舖	\$201-400	西式	海鮮	436.0	(565 食評)	50744	網上訂座可享75折優惠	

### Filter by exact value

You can filter the exact value by passing the specific numbers or strings. For example, we can select all Spanish restaurants.

```
[df['country'] == '西班牙菜'] #use two == for comparison
```

the output will be:

		name	location	price	country	style	likes	review	bookmark	discount_info
65		ROJO	天后炮台山木星街3號地下B舖	\$201-400	西班牙 菜	酒	230.0	(263 食評)	5696	餐廳優惠月: 網上訂座以\$88享Tapas 套餐
76		La Postre	銅鑼灣銅鑼灣耀華街3-9號百樂 中心19樓	\$201-400	西班牙 菜	甜品/糖 水	192.0	(219 食評)	9858	全單9折
98		La Postre	上環永樂街50號昌盛大廈3樓	\$201-400	西班牙 菜	甜品/糖 水	178.0	(216 食評)	12546	全單9折
107	Sabor Spanish Touch Private Kitchen	上環永樂街1-3號世茂大廈6樓 601室	\$201-400	西班牙 菜	私房菜	237.0	(259 食評)	8540	送 25里數 / 30積分	
128	Ei Cerdò	荃灣大堵街77號地舖	\$201-400	西班牙 菜	西班牙 菜	155.0	(186 食評)	5199	餐廳優惠月: 網上訂座以低至\$68享Tapas 套餐	
133	OM Tapas	尖沙咀棉登徑1號地舖	\$201-400	西班牙 菜	酒	130.0	(155 食評)	7285	餐廳優惠月: 網上訂座星期日至四晚市時段可以 \$88享Tapas 套餐	
162	La Postre	尖沙咀厚福街8號H8地下	\$201-400	西班牙 菜	甜品/糖 水	129.0	(161 食評)	6364		全單9折

### Filter by more than two conditions

As we discuss before, we can filter multiple conditions by using () to include each condition. For example, we can select some cost-effective restaurants for a friends gathering. Like, the `seafood restaurants` with price range in `$100-200` and `likes` more than 300 .

```
df[(df['price'] == '$101-200') & (df['style'] == '海鮮') & (df['likes'] > 300)]
```

```
In [55]: df[(df['price'] == '$101-200') & (df['style'] == '海鮮') & (df['likes'] > 300)]
```

```
Out[55]:
```

	name	location	price	country	style	likes	review	bookmark	discount_info
28	Mr. Crab by The Captain's House	灣仔莊士敦道74-78號一樓及地下C號舖	\$101-200	西式	海鮮	362.0 (404 食評)	11864	送25里數 / 30積分	
47	OPPA韓國燒肉店 OPPA Korean Restaurant	大埔舊墟直街4-20號美新大廈地下B2舖	\$101-200	韓國菜	海鮮	302.0 (342 食評)	6470	提供網上訂座	

For how to manipulate multiple conditions, You may need to review the bool comparison logic in the Chapter 3 - [logic operators](#)

## Save data

After you finish processing data in `pandas`, you may want to export it

- to store the dataset for later analysis.
- to convert it into a format that can be used by Frontend developers to make interactive web visualisations.

The common export formats are:

- `to_csv`
- `to_dict`
- to JSON format

### to\_csv

In pandas, dataframe to csv is very simple, just one line can work for this.

```
df.to_csv('full_filename', encoding='utf-8', index=False) #full_filename means you need to add the file format like openrice.csv .
```

You can pass other parameter if needed, for more information, you can check out [here](#) .

### to\_dict

There are different output formats with `to_dict` method, which depends on what parameter you pass in. By default, it will return dict like `{column -> {index -> value}}`. List like method `[{column -> value}, ... , {column -> value}]` is also wildly used. For example:

```
df.to_dict()
```

Output:

```
{'bookmark': {0: 50744,
 1: 21136,
 2: 27681,
 ...
 'country': {0: '西式',
 1: '西式',
 2: '韓國菜',
 ...
 'style': {0: '海鮮',
 1: '酒',
```

```
2: '韓式炸雞',
...}]}
```

```
df.to_dict('records')
```

Output:

```
[{'bookmark': 50744,
 'country': '西式',
 'discount_info': '網上訂座可享75折優惠',
 'likes': 436.0,
 'location': '尖沙咀金巴利道87-89號僑豐大廈地下1-2號舖',
 'name': 'LAB EAT Restaurant & Bar',
 'price': '$201-400',
 'review': '(565 食評)',
 'style': '海鮮'},
 ...,
 {'bookmark': 21136,
 'country': '西式',
 'discount_info': '送 25里數 / 30積分',
 'likes': 693.0,
 'location': '尖沙咀北京道12A號太子集團中心6樓',
 'name': 'Shine',
 'price': '$201-400',
 'review': '(777 食評)',
 'style': '酒'}]
```

You can check out [here](#) for more usage and methods.

### to\_json

`to_json` method will convert the dataframe to a JSON string, and its like combination of above two methods. The difference is that json has a `orient` parameter, which determine the output format, whether `list` like or `dict` like. Besides, for a file path or object. If not specified, the result is returned as a string.

```
df.to_json('full_file_name', orient='split', force_ascii=False) #then you can open the json file
```

```
df.to_json(orient='records', force_ascii=False) #you can change different orient method
```

Output:

```
[{"name": "LAB EAT Restaurant & Bar", "location": "尖沙咀金巴利道87-89號僑豐大廈地下1-2號舖", "price": "$201-400", "country": "西式", "style": "海鮮", "likes": 436.0, "review": "(565 食評)", "bookmark": 50744, "discount_info": "網上訂座可享75折優惠"}, ...
 {"name": "Shine", "location": "尖沙咀北京道12A號太子集團中心6樓", "price": "$201-400", "country": "西式", "style": "酒", "likes": 693.0, "review": "(777 食評)", "bookmark": 21136, "discount_info": "送 25里數 \\\ 30積分"}]
```

You can check out [here](#) for more usage and methods.

After we get the json string, we also can use `json.dumps` and `json.loads` to convert between Python object and json file.

## Bonus: from Python pandas and Javascript visualization

A very common workflow is to process data in Python and visualize data in Javascript. The last interactive chart in this [blog post](#) shows the political preference variation of HK legco members during the 2012-2016 term. The interactive chart is made by [echarts](#), a popular Javascript library for interactive visualisation from Baidu. However, the data collection and heavy duty data analysis are done in Python. We conducted the analysis in Python and export the `pandas.DataFrame` into a `JSON` format that can be consumed by echarts. You can checkout the [JSON file here](#).

The key takeaway is that data files like `json`, `csv` and `xml` are usually the bridge between frontend (e.g. Javascript) and backend (e.g. Python).

## Dataprep

You need to finish "Dataprep" before analysis. That is, we start with structured data. Preparing the structured and cleaned data has no common schema. A data scientist regularly spends most of the time in dataprep. We have pointers in [Dataprep](#) for your own reading, including some non-Python dataprep tools/ platforms. Now that we learned the basics of `pandas`, we can conduct the dataprep workflow all in Python.

Here is a polluted dataset from original openrice scraped data. Please try to combine pandas knowledge above and following guidelines to prepare a structured data that is good for further analysis.

## Cleaning

The format:

- Does this file use UTF-8 encoding? Checkout some common issues [about encoding](#). If not, are you able to convert it to the conventional UTF-8 encoding?
- Does the input data table has valid column names? If not, how do you know the meaning of each column?
- Does every row of the table have the same number of columns ("cells" more precisely)?
- Is every element in a single column of the data type? Say all integers or all strings. Do you see a string mixed into a column where you are supposed to see numbers?
- How many missing values are in this table? Do missing values affect your analysis result? Do you want to fill the missing values (e.g. `DataFrame.fillna()`) somehow; Or do you want to remove the rows/ columns which contain missing values?

The content:

- Check the variable distribution. Is there any special value that only appears once or a few times? Will it be a typo?
- Check the variable range. What is the common range of values in this variable? Is there any peculiar value?
- Check the string length. Is there a super long cell? It may be because parsing error during scraping stage. Some data may mix up.
- Check the missing values. Are there empty cells? What is the reasonable default value to fill in those empty cells?
- Check the above on a subset of data (filtering/ grouping). Does `50` looks like a regular price? Does `50` looks like a regular price within "seafood" category?
- Is the duplicate content? If there is no duplicate *entire rows*, is there duplicate rows in terms of a subset of the columns? Is this duplicate an error in the data? You may want to leverage some domain knowledge to further check.

As an exercise, you can download a CSV file with intentionally injected error [here](#). The notebook is for your reference.

## Transformation

- Convert text value to numeric value. e.g. convert price range text into a representative number that can be sorted.
- Merge multiple categories. e.g. merge 港式 and 潮州菜 into 中式 . Sometimes, less categories help analysis.

- You may need some domain knowledge and trials and errors, in order to find good method of grouping.
- Encoding, many times called "coding" in social science research, is the process of turning natural language data into numeric data. This is also a matter of domain knowledge but you can try the method here. e.g. in order to study the relationship between price range and the income level of that district, we can encode 18 districts into three income class `high` , `medium` and `low` .

## Extraction

We load the CSV data in one shot, because the current dataset is very small. In real practice, you may meet a large dataset, so the first step is usually data extraction. One can do extraction by certain rules, e.g. get the restaurants in a certain district, get the restaurants that are open in a certain time period. Or you are interested in the whole population but do not possess the appropriate computation power to handle this dataset. At this point, you may want to do sampling. `DataFrame.sample()` may be helpful here. When dealing with really large dataset, you may want to combine extraction by rule and extraction by sampling.

## References

- [Exercise numpy](#) on ShiYanLou
  - [Exercise pandas](#) on ShiYanLou
- 

If you have any questions, or seek for help troubleshooting, please [create an issue here](#)

# Week 10: Work with table - 1D analysis and 2D analysis

- [Week 10: Work with table - 1D analysis and 2D analysis](#week-10-work-with-table---1d-analysis-and-2d-analysis) - [Objective](#objective) - [More Arithmetics on DataFrame and Series](#more-arithmetics-on-dataframe-and-series) - [Exercise: The Berkeley admission synthesis dataset](#exercise-the-berkeley-admission-synthesis-dataset) - [Distribution](#distribution) - [Histogram](#histogram) - [Bonus: How histograms can be cheating](#bonus-how-histograms-can-be-cheating) - [Kernel Density Estimation (KDE)](#kernel-density-estimation-kde) - [Special points in distribution](#special-points-in-distribution) - [Bonus: Articulate central tendency and spread of data](#bonus-articulate-central-tendency-and-spread-of-data) - [Variance](#variance) - [Skewness](#skewness) - [Kurtosis](#kurtosis) - [The mode of data](#the-mode-of-data) - [Correlation](#correlation) - [Continuous: Scatter plot and correlation](#continuous-scatter-plot-and-correlation) - [Bonus: Better visualisation](#bonus-better-visualisation) - [Filter out in the charts](#filter-out-in-the-charts) - [A more primitive/ finer controlled way of plotting using matplotlib](#a-more-primitive-finer-controlled-way-of-plotting-using-matplotlib) - [Discrete: Cross-tab](#discrete-cross-tab) - [DataFrame.groupby](#dataframegroupby) - [pandas.pivot\_table](#pandaspivot\_table) - [Discretise multiple columns](#discretise-multiple-columns) - [pivot\_table to generate cross-tabs table](#pivot\_table-to-generate-cross-tabs-table) - [From correlation to causality](#from-correlation-to-causality) - [Bonus: (Statistical) Hypothesis testing](#bonus-statistical-hypothesis-testing) - [Reference](#reference)

## Objective

- Master the schema of "data-driven story telling": the crowd (pattern) and the outlier (anomaly)
- Can use `pandas`, `matplotlib` and `seaborn` to conduct 1D analysis and articulate on the statistics
- Can conduct 2D analysis by:
  - `pandas.pivot_table()` -- discrete distribution analysis (bin analysis)
  - `pandas.groupby().aggregate()` -- the SAC (splitting -- applying -- combining) pattern
  - `Series.corr()` -- calculate correlation
  - `DataFrame.plot()` or `matplotlib.pyplot.plot()` -- scatter plot to visualise 2D correlation

The dataset and case we use this week comes from a workshop called "descriptive analysis" conducted by Jenifer on GICJ2017 in South Africa. You can download data from:

- [www.jenster.com/nottingham.xlsx](http://www.jenster.com/nottingham.xlsx)
- [www.jenster.com/index.xlsx](http://www.jenster.com/index.xlsx)

New modules:

- **Matplotlib.** Matplotlib is a Python 2D plotting library and one of the most frequently used plotting modules in Python.
- **Seaborn.** Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

## More Arithmetics on DataFrame and Series

Series has many arithmetic functions. Although one can easily implement those functions with basic `list` and basic logics in Python, those provided by `Series` can save some time by writing for loops, because they are designed to work on "a series of numbers". Examples are like:

- `.sum()`

- `.min()`
- `.max()`
- `.quantile()`

Two `Series` of the same length can use conventional arithmetics and boolean operators to perform an **element-wise** operation. For example:

- `a + b` - result is a new `Series` whose values are the element-wise addition from `a` and `b`.
- `a == b` - result is a new `Series` whose values are element-wise `==` logical operation.

When one hand side of the equation is not a `Series`, but a "scalar", i.e. single number, this single number is used in all operations with all elements from another `Series`.

Many functions that are available for `Series` are also available for `DataFrame`. A `DataFrame` is in essence a collection of `Series`. To apply those functions that works on `Series` to `DataFrame`, we need to have certain ordering, which is given by a keyword parameter called `axis`:

- Operate along columns (`axis=0`)
- Operate along rows (`axis=1`)

**TIP:** Sometimes, one may be used to column operations or row operations, when writing his/ her computation logics. It is Ok to stick with one convention. When you need to operate along another axis, use the **transpose** version of the `DataFrame`, i.e. `DataFrame.T` (Use it like a member variable).

With the progress of data processing, the table at your hand is usually larger and larger. You may want to put the new result back to original table sometimes. There are mainly two ways:

- Adding a new column is easy. Just use `df['new-column'] = A valid Series`.
- Adding a new row needs some more work. Use `pandas.concat([df, row])`, where `df` is the original `DataFrame`; and `row` is the new `DataFrame` with one data point, whose columns are the same as `df`.

## Exercise: The Berkeley admission synthesis dataset

Calculate the by-department admission ratio and the whole-school admission ratio. Try to articulate whether there is gender discrimination or not.

The dataset can be downloaded [here](#).

For further reading, search for "Simpson's paradox".

Here is the [demo code](#).

## Distribution

For distribution, we can use some simple `pandas` statistics functions to get a overall picture of what's the data distribution like, from what we can get at least two analyzing directions:

1. Is there a clear trend or a pattern of the distribution?
2. Is there any abnormal or outliers that worthy noticing?

The following is the analyzing demo after we get a clean dataset, based on the example of *Cheating our children* case, and try to find insights.

## Histogram

```
import pandas as pd
from matplotlib import pyplot as plt
```

```

import seaborn as sns
import numpy as np

df = pd.read_excel('nottingham.xlsx')
#!pip install xlrd for supporting to read excel if error arisen.

df.describe() #get descriptive information

```

In [4]: `df.describe()`

Out[4]:

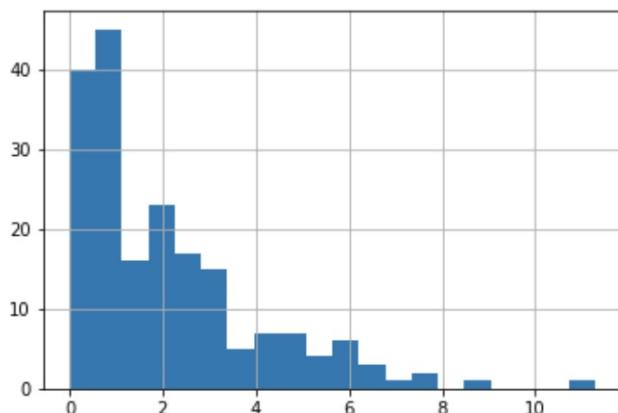
	RecType	TotPup	TotElig	AVG_ENG_MATH_SCORE_07	AVG_ENG_MATH_SCORE_0
<b>count</b>	207.000000	207.000000	205.000000	179.000000	182.000000
<b>mean</b>	1.057971	257.705314	37.097561	27.497207	27.51868
<b>std</b>	0.234255	175.026559	18.858613	2.031865	1.66702
<b>min</b>	1.000000	0.000000	2.000000	15.000000	22.90000
<b>25%</b>	1.000000	179.000000	25.000000	26.200000	26.30000
<b>50%</b>	1.000000	230.000000	34.000000	27.800000	27.40000
<b>75%</b>	1.000000	319.000000	50.000000	28.850000	28.80000
<b>max</b>	2.000000	2170.000000	119.000000	31.900000	31.20000

The columns:

- `AVG_ENG_MATH_SCORE_xx` : The average score for the particular class of year xx. For a class, this metric is the higher the better
- `P_ABSENT_PERSIST` : the absent ratio transformed somehow, the higher the worse.

This is multi dimensional data. We are interested in the relationship between those dimensions / variables. For example, the absent ratio.

```
%matplotlib inline #add this line before plotting charts
df['P_ABSENT_PERSIST'].hist(bins=20)
```



Quick Questions:

- What do you conclude from this histogram?
- What would you do next to mine the news?

It's clear that most of school has only 0 - 2 absent ratio, but 2 schools has more than 8 percent absent ratio. A question for us is why those two schools are abnormal and who are they? We can filter out to dig out more, and see if we can find anything interesting.

```
df[df['P_ABSENT_PERSIST'] > 8]
```

	RecType	Schoolme	Address1	Address2	Address3	Town	PostCode	TelNum	SchoolType	TotPup	TotElig	Avg_Eng_Math_Score_07	Avg_Eng_M
0	1	Claremont Primary and Nursery School	Claremont Road	Off Huckll Road		Nottingham	NG5 1BH	0115 9156870	CY	337	49.0		25.1
1	1	Windmill Primary & Nursery School	Sneinton Boulevard			Nottingham	NG2 4FZ	0115 9150195	CY	442	86.0		24.9

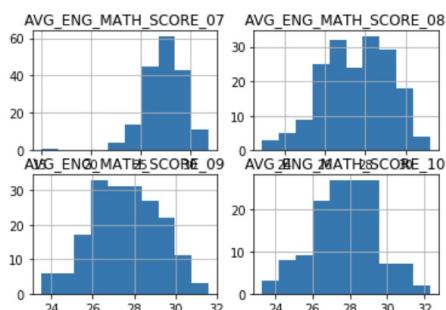
Also, one can check out the other columns and see if there anything abnormal or trend.

```
plt.subplot(2, 2, 1)
df['AVG_ENG_MATH_SCORE_07'].hist(bins=10)
plt.title('AVG_ENG_MATH_SCORE_07')

plt.subplot(2, 2, 2)
df['AVG_ENG_MATH_SCORE_08'].hist(bins=10)
plt.title('AVG_ENG_MATH_SCORE_08')

plt.subplot(2, 2, 3)
df['AVG_ENG_MATH_SCORE_09'].hist(bins=10)
plt.title('AVG_ENG_MATH_SCORE_09')

plt.subplot(2, 2, 4)
df['AVG_ENG_MATH_SCORE_10'].hist(bins=10)
plt.title('AVG_ENG_MATH_SCORE_10')
```



Same question: can you find anything notable from this chart? It that weird for the outlier whose score is around 15 in year 7? We can filter it out and invest later.

```
df[df['AVG_ENG_MATH_SCORE_07'] < 16]
```

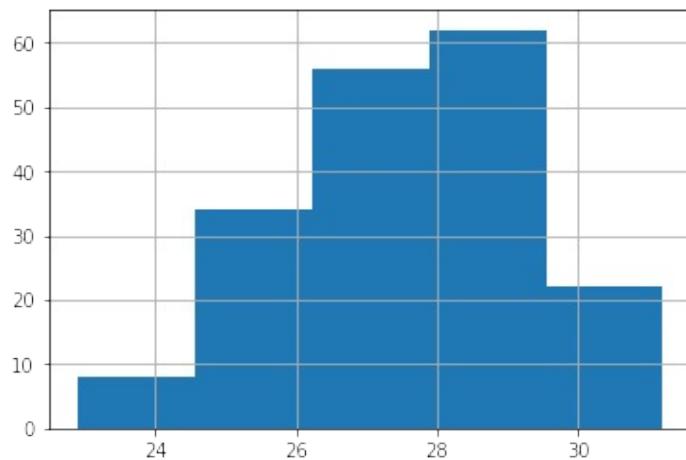
Out[6]:

	RecType	Schoolme	Address1	Address2	Address3	Town	PostCode	TelNum	SchoolType
199	2	Carlton Digby School	61 Digby Avenue	Mapperley		Nottingham	NG3 6DS	0115 9568289	

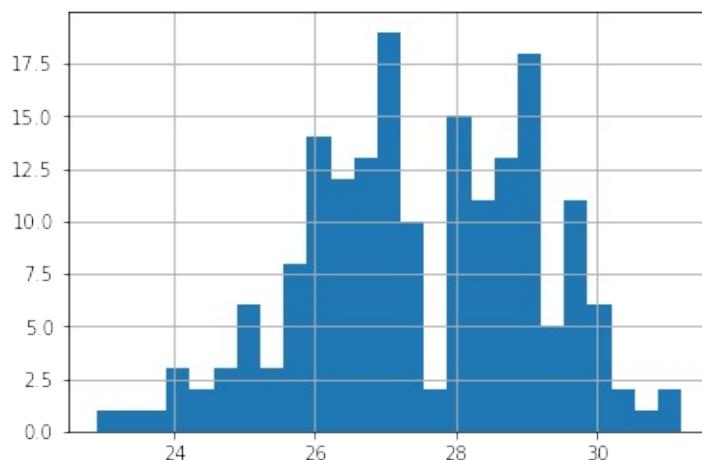
## Bonus: How histograms can be cheating

Try to adjust number of bins and bin boundaries to see what happens.

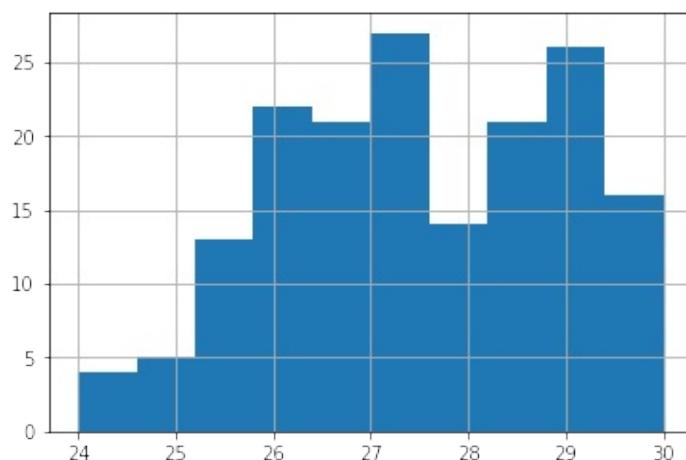
```
df['AVG_ENG_MATH_SCORE_08'].hist(bins=5) #bins=5
```



```
df['AVG_ENG_MATH_SCORE_08'].hist(bins=25) #bins = 25
```



```
df['AVG_ENG_MATH_SCORE_08'].hist(bins=10,range=(24,30)) #change data range
```



You can see that there is only one peak when the number of bins is 5, however, if you enlarges `bins`, you will get a more detailed information. There appears more peaks and a valley between. Besides, if you change the data range, the situation and conclusion here can be more diverse.

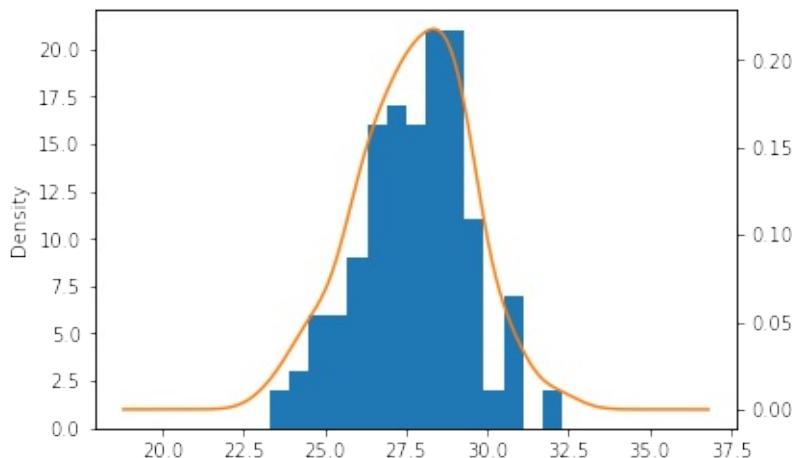
The key takeaway here is that different angle could lead to different stories, and it's all depend on which angle you choose to take in. What we can do is to learn to recognize the pattern here and not to be fooled by the charts.

## Kernel Density Estimation (KDE)

KDE is fundamentally similar to histogram. It takes every data point, interpolate the distribution using a continuous function (kernel), and then sum up those functions to generate the envelop of distribution. It has similar function when you articulate the distribution of data. The major advantage is that it does not suffer from the bin-segmentation issue as we see above in histogram.

For better data presentation and aesthetics purpose, people sometimes put histogram and KDE on the same chart, from which you can see the distribution pattern more clearly.

```
ax = df['AVG_ENG_MATH_SCORE_10'].hist(bins=15)
df['AVG_ENG_MATH_SCORE_10'].plot(kind='kde', ax=ax, secondary_y=True)
```



## Special points in distribution

- Mean

```
In [8]: df["AVG_ENG_MATH_SCORE_07"].mean()
Out[8]: 27.497206703910614

In [9]: df["AVG_ENG_MATH_SCORE_08"].mean()
Out[9]: 27.518681318681317

In [10]: df["AVG_ENG_MATH_SCORE_09"].mean()
Out[10]: 27.566844919786096

In [11]: df["AVG_ENG_MATH_SCORE_10"].mean()
Out[11]: 27.72086330935252
```

The statistical mean, gives a very good idea about the central tendency of the data being collected. What can we get from the mean? you can have a general idea that overall, students get better performance in higher grades, but is it normal?

- Max/ Min

```
In [10]: df["P_ABSENT_PERSIST"].max()
Out[10]: 11.3
```

```
In [11]: df[df["P_ABSENT_PERSIST"] == 11.3]
Out[11]:
 toolType TotPup TotElig AVG_ENG_MATH_SCORE_07 AVG_ENG_MATH_SCORE_08 AVG_ENG_MATH_SCORE_09 AVG_ENG_MATH_SCORE_10 P_ABSENT_PERSIST
 CY 337 49.0 25.1 26.0 26.2 26.4 11.3
```

The max and min shows the most extreme observations. If extreme values are real (not measurement errors), it becomes valuable to us, giving us a breakthrough point to dig out the reason, which we emphasize at the very beginning of this course - abnormal.

For this case, we can filter out the school with highest absent rate and see if there is anything interesting. Step further, we can filter out the school with high absent rate but high performance in score at the same time. This is also abnormal for us in theory which we can further check out.

```
In [14]: df[(df["P_ABSENT_PERSIST"] > df["P_ABSENT_PERSIST"].mean())
 & (df["AVG_ENG_MATH_SCORE_10"] > df["AVG_ENG_MATH_SCORE_10"].mean())]
Out[14]:
 toolType TotPup TotElig AVG_ENG_MATH_SCORE_07 AVG_ENG_MATH_SCORE_08 AVG_ENG_MATH_SCORE_09 AVG_ENG_MATH_SCORE_10 P_ABSENT_PERSIST
 CY 225 28.0 28.1 27.1 26.9 27.9 3.9
 CY 460 56.0 26.7 26.7 27.8 28.6 3.8
 CY 359 55.0 26.2 28.2 27.1 28.3 3.1
 CY 166 40.0 28.1 29.7 28.9 28.9 3.0
 VA 285 39.0 29.3 28.5 29.6 28.6 2.9
 CY 421 55.0 28.9 28.3 29.7 28.0 2.7
 CY 310 42.0 26.9 27.0 28.7 28.9 2.6
 VA 238 26.0 30.7 31.2 31.2 30.6 2.4
```

- Median

```
df["AVG_ENG_MATH_SCORE_10"].median()
```

Output:

```
27.8
```

Median provides a helpful measure of center of our dataset. But more often, we care more about the Percentile, like where are the majority of the data located.

- Percentile

Percentile is a given percentage of observations in a group of observations fall. For example, the 75th percentile is the value (or score) below which 75% of the observations may be found. 50th percentile is equal to median.

```
df["AVG_ENG_MATH_SCORE_10"].quantile(0.8) #90th percentile
```

29.62

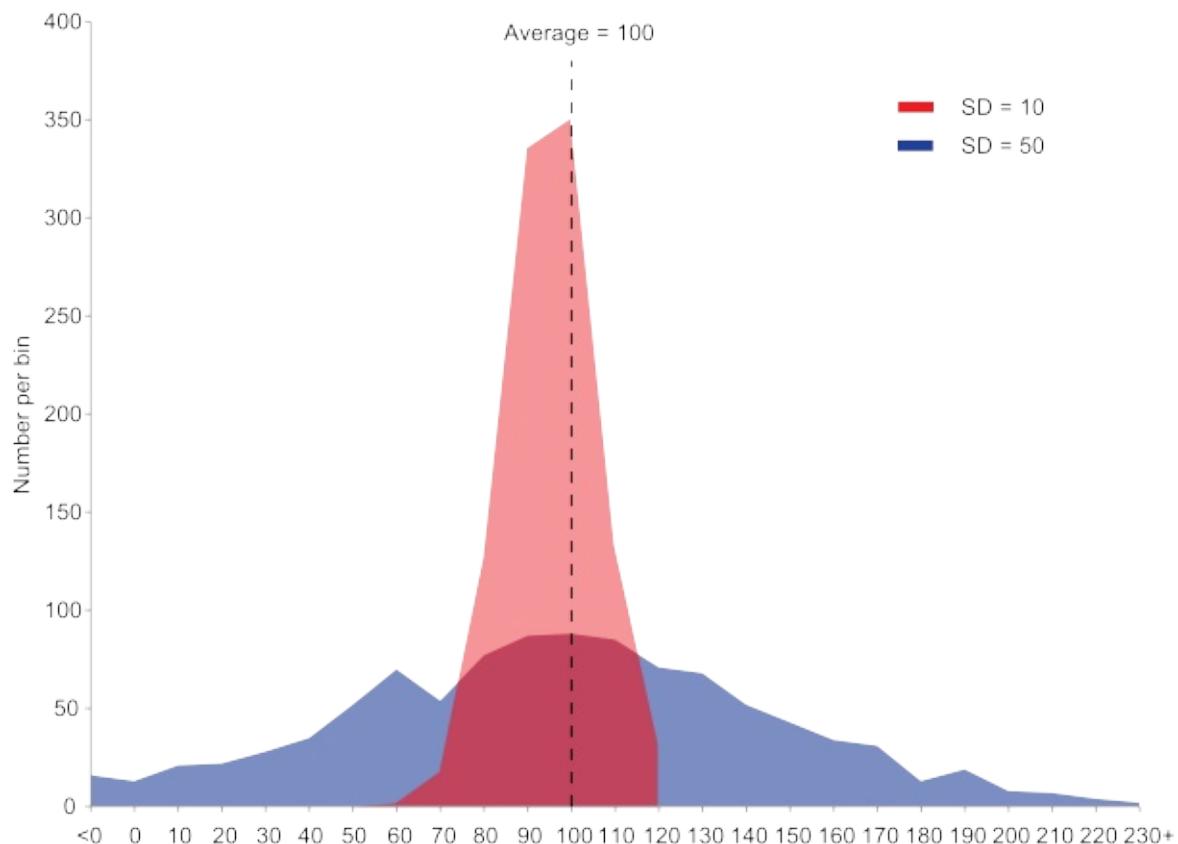
```
df["AVG_ENG_MATH_SCORE_10"].quantile(0.7) #70th percentile
```

28.759999999999998

## Bonus: Articulate central tendency and spread of data

### Variance

Variance is the expectation of the squared deviation of a random variable from its mean. Informally, it measures how far a set of (random) numbers are spread out from their average value. The smaller the variance, the sharper the distribution. The variance is the square of the standard deviation.



from Wikipedia

```
df[['AVG_ENG_MATH_SCORE_07', 'AVG_ENG_MATH_SCORE_08', 'AVG_ENG_MATH_SCORE_09', 'AVG_ENG_MATH_SCORE_10']].var(axis=0)
).sort_values(ascending=False)
```

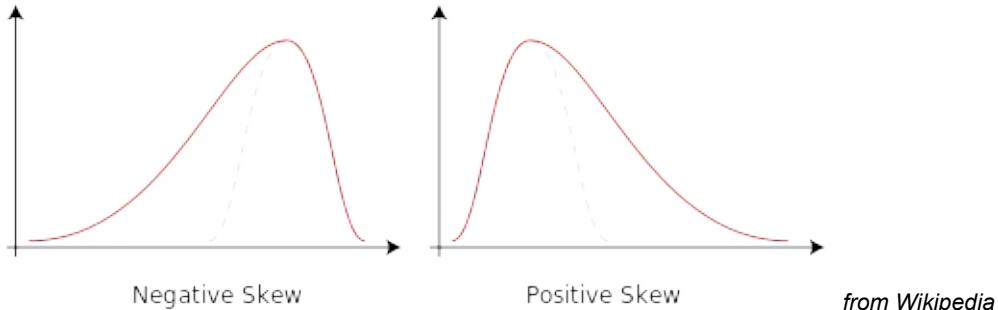
Output:

AVG_ENG_MATH_SCORE_07	4.128475
AVG_ENG_MATH_SCORE_10	2.983547
AVG_ENG_MATH_SCORE_08	2.778986
AVG_ENG_MATH_SCORE_09	2.694271

For this case, we can see that the score fluctuations of students in year 7 are the largest. And there is no clear pattern only considering this factor.

## Skewness

Skewness can help us recognize the general distribution pattern, whether it is feasible to treat it as a normal distribution. If  $Sk > 0$ , the larger the Sk value, the higher the degree of right deviation; If  $Sk < 0$ , the smaller the Sk value, the higher the degree of left deviation.



```
df[['AVG_ENG_MATH_SCORE_07', 'AVG_ENG_MATH_SCORE_08', 'AVG_ENG_MATH_SCORE_09', 'AVG_ENG_MATH_SCORE_10']].skew(axis=0).sort_values(ascending=False)
```

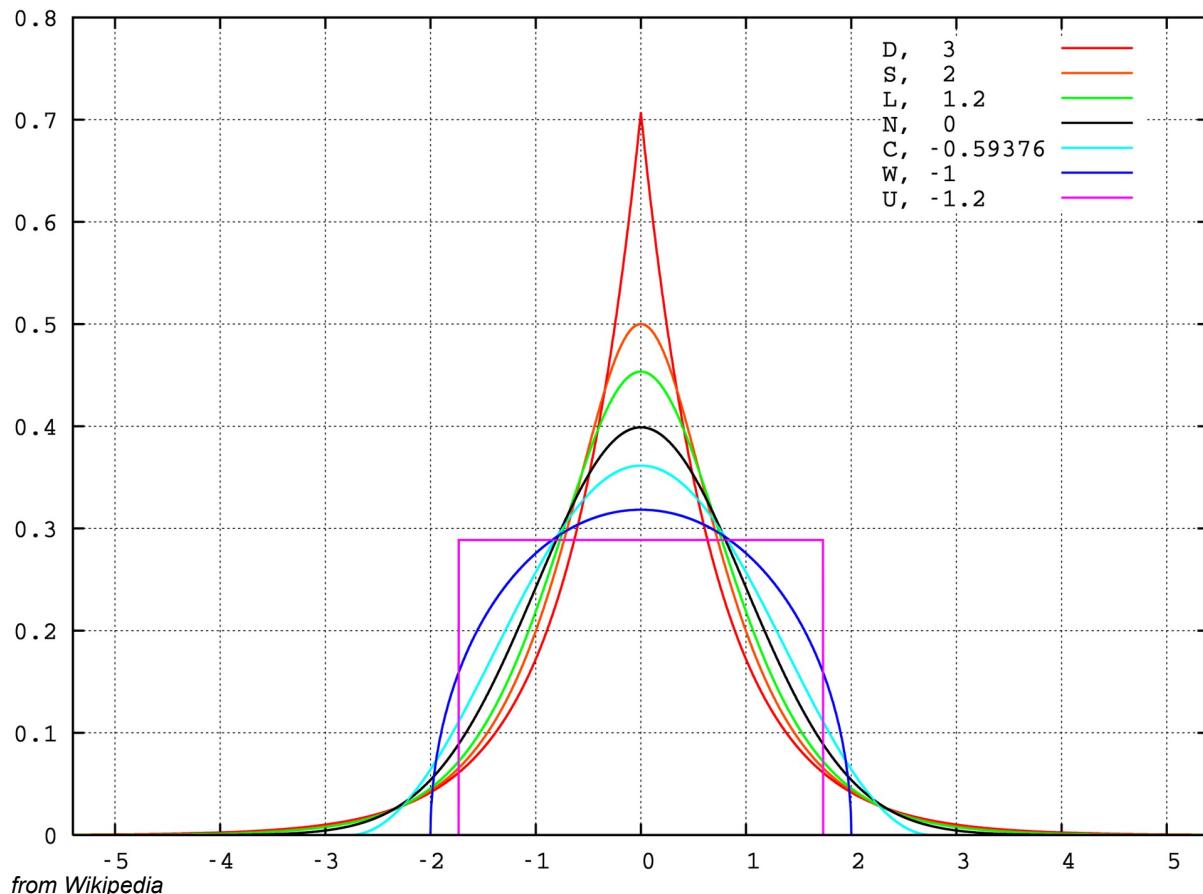
Output:

AVG_ENG_MATH_SCORE_09	-0.038742
AVG_ENG_MATH_SCORE_10	-0.152395
AVG_ENG_MATH_SCORE_08	-0.224148
AVG_ENG_MATH_SCORE_07	-1.352412

From the results, we can have a overview of that the scores distribution of those students is left deviated, and mass students is concentrated on the right of the figure. And there is no apparent pattern related to the years.

## Kurtosis

The kurtosis reflects the sharpness of the peak of the distribution. The greater the kurtosis, the sharper and steeper of the distribution peak. A high kurtosis means that the increase in variance is caused by an extreme values in the low frequency that is greater or less than the average.



```
df[['AVG_ENG_MATH_SCORE_07', 'AVG_ENG_MATH_SCORE_08', 'AVG_ENG_MATH_SCORE_09', 'AVG_ENG_MATH_SCORE_10']].kurtosis(axis=0).sort_values(ascending=False)
```

Output:

AVG_ENG_MATH_SCORE_07	6.934606
AVG_ENG_MATH_SCORE_10	-0.036167
AVG_ENG_MATH_SCORE_08	-0.428693
AVG_ENG_MATH_SCORE_09	-0.478333

From this statistic, we can know that students in year 7 has larger kurtosis, which means that the distribution peak is sharper, and there are more extreme value points in year 7.

## The mode of data

Get the value with maximum frequency. For example, filter out the grade with maximum frequency of each year students.

```
#get the data of all years
df_g = df[['AVG_ENG_MATH_SCORE_07', 'AVG_ENG_MATH_SCORE_08', 'AVG_ENG_MATH_SCORE_09', 'AVG_ENG_MATH_SCORE_10']]
#drop the rows with nan value, unless it will affect the results
df_g.dropna(inplace=True)
#df_g
df_g.mode()
```

	AVG_ENG_MATH_SCORE_07	AVG_ENG_MATH_SCORE_08	AVG_ENG_MATH_SCORE_09	AVG_ENG_MATH_SCORE_10
0	27.8	28.8	27.1	28.4
1	28.1	NaN	28.2	28.9
2	NaN	NaN	NaN	29.2

Note that there could be multiple values returned for the selected axis (when more than one item share the maximum frequency), which is the reason why a dataframe is returned. If you want to impute missing values with the mode in a dataframe df, you can just do this: `df_g.mode().iloc[0]`

Also, you can use `df_g['AVG_ENG_MATH_SCORE_08'].mode()` to get single series mode.

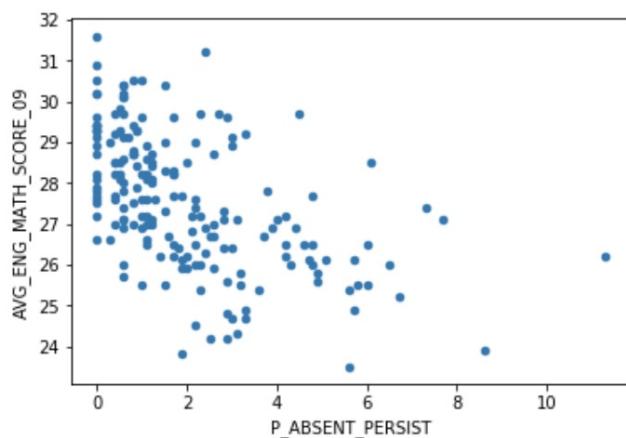
**NOTE:** The above is just for demonstration purpose. You usually do not checkout mode of real valued variables, because it is too likely for every instance to get different values. When the variable is highly discrete or is actually categorical, mode can tell us some stories.

## Correlation

### Continuous: Scatter plot and correlation

We can plot the correlation graph to display relationship between two variables and columns. For example, to figure out whether there is a correlation between absence and score.

```
df.plot('P_ABSENT_PERSIST', 'AVG_ENG_MATH_SCORE_09', kind='scatter')
#kind means the graph type
```



Generally, you can see that the higher the absence ratio, the lower the test score in general. But here are two questions:

- Is this relationship strong enough?
- What are the outliers?

To solve this problem, we need to use correlation functions to dig out more.

```
help(df['P_ABSENT_PERSIST'].corr)
```

Output:

```
Help on method corr in module pandas.core.series:
```

```

corr(other, method='pearson', min_periods=None) method of pandas.core.series.Series instance
 Compute correlation with `other` Series, excluding missing values

Parameters

other : Series
method : {'pearson', 'kendall', 'spearman'}
 * pearson : standard correlation coefficient
 * kendall : Kendall Tau correlation coefficient
 * spearman : Spearman rank correlation
min_periods : int, optional
 Minimum number of observations needed to have a valid result

Returns

correlation : float

```

From above you can see that, `corr` function is used to compute one series with other series. And there are 3 methods: `pearson`, `kendall`, `spearman`. we don't need necessarily to know how to calculate instead we need to what does it means and main differences. For example, `pearson` measure the degree of the relationship between linearly related variables, while Spearman rank correlation is a non-parametric test . For more details, You can refer [here](#)

```
df['P_ABSENT_PERSIST'].corr(df['AVG_ENG_MATH_SCORE_09'], method='pearson')
```

output:

```
-0.5205965225654683
```

#### Note:

- Pearson correlation is between [-1, 1]
- Values around 0 means no correlation/ weak correlation
- Values near 1 and -1 can be interpreted as strong (linear) correlation

Pearson correlation does not work very well with non-linear correlation or when the variables are not (jointly) normally distributed. It is also sensitive to outliers. Spearman rank correlation can help here. You can make a judgement whether there is a correlation between grades and absent rate.

```

df['P_ABSENT_PERSIST'].corr(df['AVG_ENG_MATH_SCORE_10'], method='spearman')
df['P_ABSENT_PERSIST'].corr(df['AVG_ENG_MATH_SCORE_09'], method='spearman')
df['P_ABSENT_PERSIST'].corr(df['AVG_ENG_MATH_SCORE_08'], method='spearman')
df['P_ABSENT_PERSIST'].corr(df['AVG_ENG_MATH_SCORE_07'], method='spearman')

```

#### The correlation for all grades

```

: df['P_ABSENT_PERSIST'].corr(df['AVG_ENG_MATH_SCORE_10'], method='spearman')
: -0.4977757689875953

: df['P_ABSENT_PERSIST'].corr(df['AVG_ENG_MATH_SCORE_09'], method='spearman')
: -0.5810765727681304

: df['P_ABSENT_PERSIST'].corr(df['AVG_ENG_MATH_SCORE_08'], method='spearman')
: -0.6002183459864541

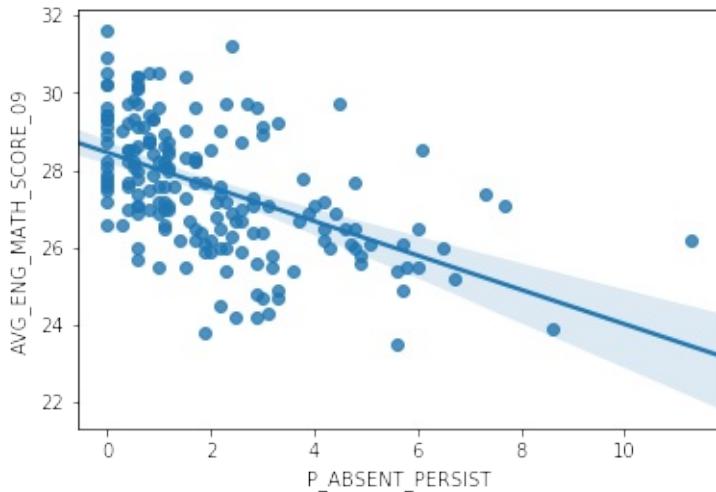
: df['P_ABSENT_PERSIST'].corr(df['AVG_ENG_MATH_SCORE_07'], method='spearman')
: -0.5038740642509405

```

## Bonus: Better visualisation

One can do more with the scatter graphs. We can draw the regression line in the charts, filter the outliers on the charts, adjust the transparency of the dots...

```
sns.regplot(df['P_ABSENT_PERSIST'], df['AVG_ENG_MATH_SCORE_09'])
```



```
np.polyfit(df['P_ABSENT_PERSIST'].fillna(0), df['AVG_ENG_MATH_SCORE_09'].fillna(0), 1)
```

**Quiz:** What does it look like if we plot above line? The return value of polyfit is Polynomial coefficients, highest power first.

**NOTE:** Try the codes without fillna and observe the error. Now it is time to do some cleaning.

```
na_selector = df['P_ABSENT_PERSIST'].isna()
na_selector |= df['AVG_ENG_MATH_SCORE_07'].isna()
na_selector |= df['AVG_ENG_MATH_SCORE_08'].isna()
na_selector |= df['AVG_ENG_MATH_SCORE_09'].isna()
na_selector |= df['AVG_ENG_MATH_SCORE_10'].isna()
len(df[na_selector])
len(df)
len(df[-na_selector])
df_cleaned = df[-na_selector]
np.polyfit(df_cleaned['P_ABSENT_PERSIST'].fillna(0),
 df_cleaned['AVG_ENG_MATH_SCORE_09'].fillna(0),
 1)
```

**Note:** For how to handle NA/NaN value in pandas, you can refer [here](#).

```

: na_selector = df['P_ABSENT_PERSIST'].isna()
: na_selector |= df['AVG_ENG_MATH_SCORE_07'].isna()
: na_selector |= df['AVG_ENG_MATH_SCORE_08'].isna()
: na_selector |= df['AVG_ENG_MATH_SCORE_09'].isna()
: na_selector |= df['AVG_ENG_MATH_SCORE_10'].isna()

: len(df[na_selector])
: 80

: len(df)
: 207

: len(df[~na_selector])
: 127

: df_cleaned = df[~na_selector]

: np.polyfit(df_cleaned['P_ABSENT_PERSIST'].fillna(0),
: df_cleaned['AVG_ENG_MATH_SCORE_09'].fillna(0),
: 1)

: array([-0.44654826, 28.54239409])

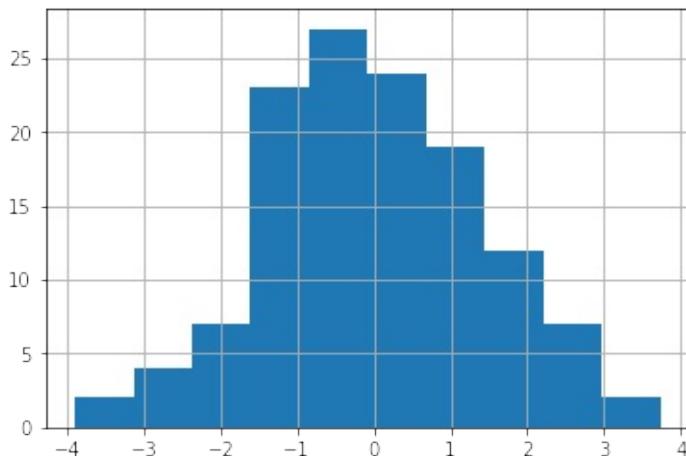
```

After we get the regression coefficient, we can estimate the grade09 and compare with the actual ones to see if there is a big difference.

```

absent = df_cleaned['P_ABSENT_PERSIST']
score_grade09 = df_cleaned['AVG_ENG_MATH_SCORE_09']
estimated_score_grade09 = 28.54239409 + (-0.44654826) * absent
(score_grade09 - estimated_score_grade09).hist()

```



We can filter out the school that have hugh different scores with the estimation.

```

df_cleaned[(score_grade09 - estimated_score_grade09) > 3]
df_cleaned[(score_grade09 - estimated_score_grade09) > 2.5]

```

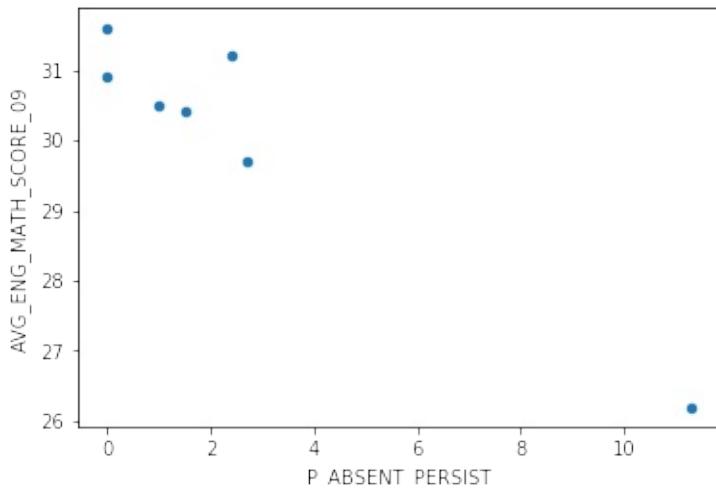
df_cleaned[(score_grade09 - estimated_score_grade09) > 3]							
toolType	TotPup	TotElig	Avg_Eng_Math_Score_07	Avg_Eng_Math_Score_08	Avg_Eng_Math_Score_09	Avg_Eng_Math_Score_10	P_Absent_Persist
VA	238	26.0	30.7	31.2	31.2	30.6	2.4
CY	123	13.0	31.4	28.0	31.6	30.9	0.0

df_cleaned[(score_grade09 - estimated_score_grade09) > 2.5]													
	RecType	Schoolme	Address1	Address2	Address3	Town	PostCode	TelNum	SchoolType	TotPup	TotElig	Avg_Eng_Math_Score_07	Avg_Eng_Math_Score_09
0	1	Claremont Primary and Nursery School	Claremont Road	Off Huckll Road		Nottingham	NG5 1BH	0115 9156870	CY	337	49.0	25.1	
62	1	English Martyrs' Catholic Primary	Bracken Road	Long Eaton		Nottingham	NG10 4DA	0115 9733209	VA	238	26.0	30.7	
94	1	Bleasby CofE Primary School	Station Road	Bleasby		Nottingham	NG14 7GD	01636 830203	VC	166	23.0	30.6	
182	1	Orston Primary School	Church Street	Orston		Nottingham	NG13 9NS	01949 850618	CY	123	13.0	31.4	

## Filter out in the charts

Get the threshold value for 95% percentile

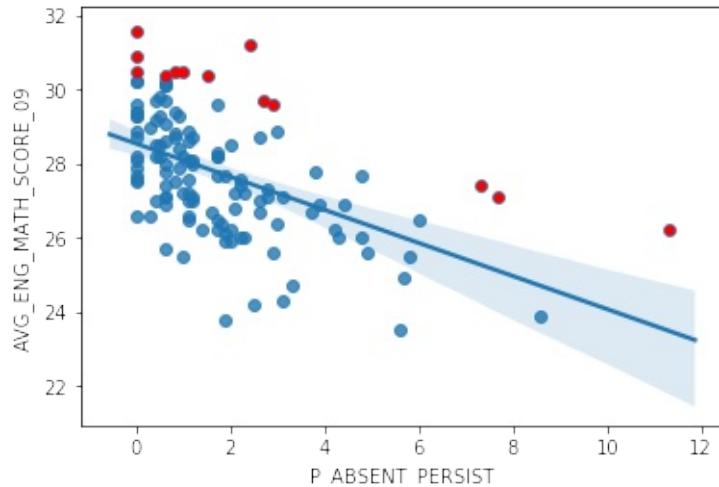
```
s = (score_grade09 - estimated_score_grade09)
s.quantile(0.95)
df_cleaned[s > s.quantile(0.95)].plot(
 x='P_ABSENT_PERSIST',
 y='AVG_ENG_MATH_SCORE_09',
 kind='scatter')
```



Adjust the quantile to include/ exclude suspicious schools.

```
ax = sns.regplot(
 df_cleaned['P_ABSENT_PERSIST'],
 df_cleaned['AVG_ENG_MATH_SCORE_09'],
)
df_cleaned[s > s.quantile(0.90)].plot(
 x='P_ABSENT_PERSIST',
 y='AVG_ENG_MATH_SCORE_09',
 kind='scatter',
```

```
color='red',
ax=ax)
```

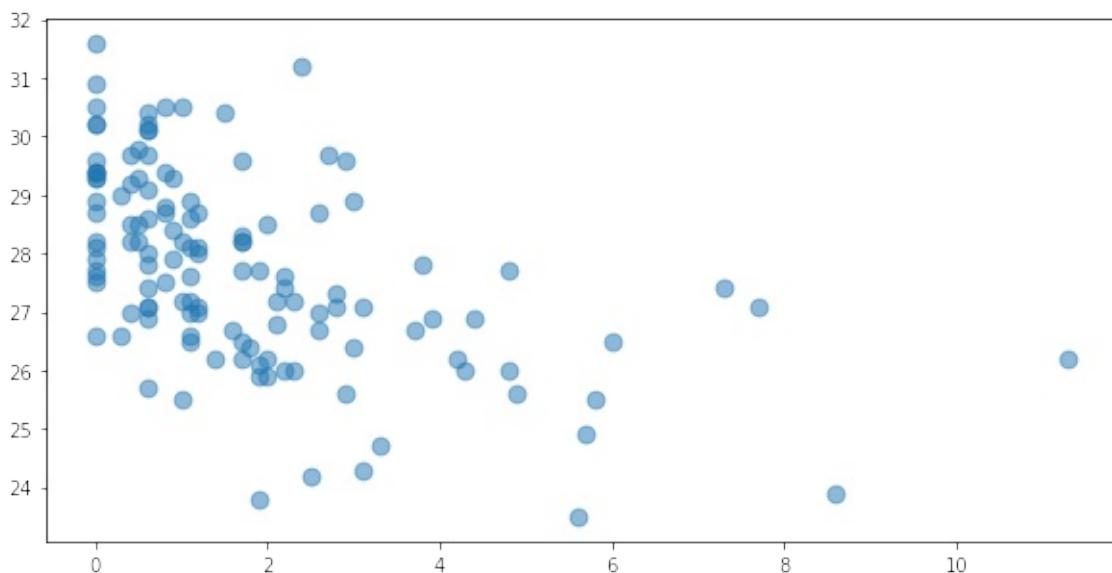


## A more primitive/ finer controlled way of plotting using matplotlib

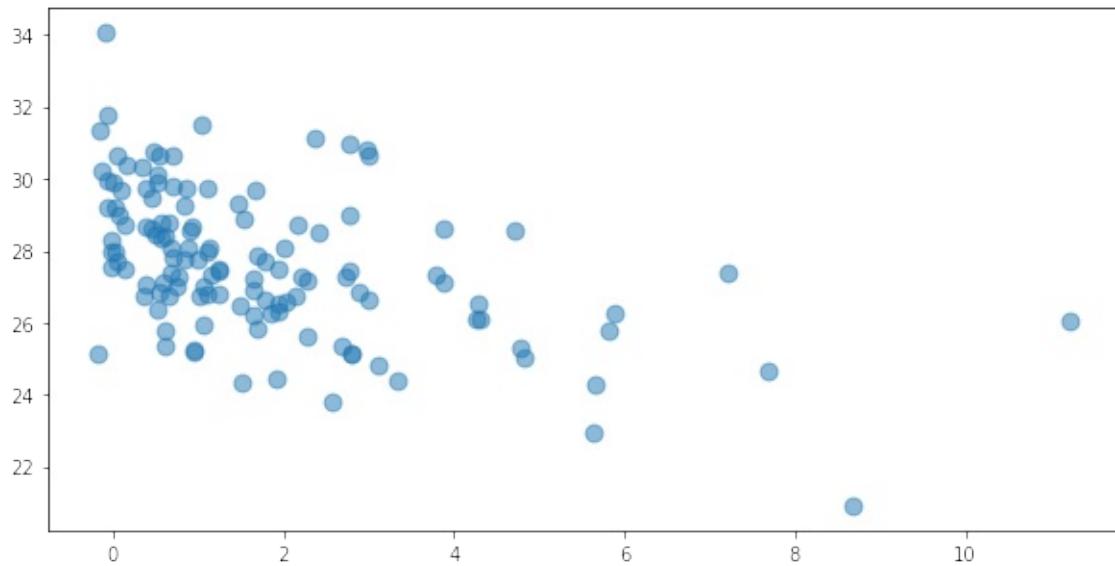
Two common tricks for better visuals:

- Add jitter to further scatter the data points
- Use transparency to help identify density

```
plt.figure(figsize=(10, 5))
plt.scatter(
 df_cleaned['P_ABSENT_PERSIST'],
 df_cleaned['AVG_ENG_MATH_SCORE_09'],
 s=80, alpha=0.5)
```



```
plt.figure(figsize=(10, 5))
plt.scatter(
 df_cleaned['P_ABSENT_PERSIST'] + np.random.normal(0, 0.1, len(df_cleaned)),
 df_cleaned['AVG_ENG_MATH_SCORE_09'] + np.random.normal(0, 1, len(df_cleaned)),
 s=80, alpha=0.5)
```



```

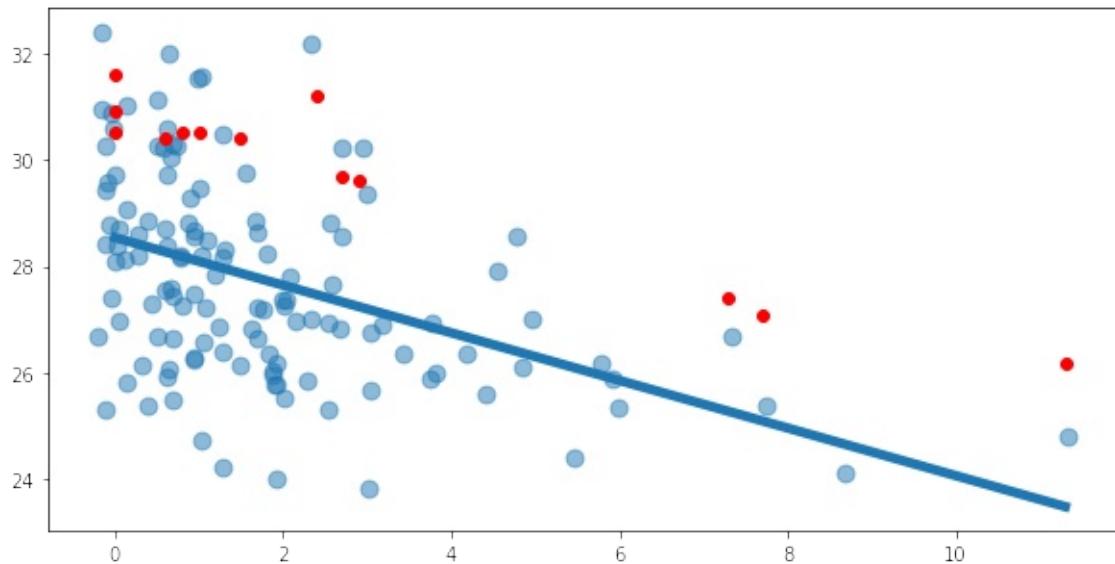
coeffs = np.polyfit(df_cleaned['P_ABSENT_PERSIST'].fillna(0),
 df_cleaned['AVG_ENG_MATH_SCORE_09'].fillna(0),
 1)
#coeffs
trendline_x = np.linspace(df_cleaned['P_ABSENT_PERSIST'].min(), df_cleaned['P_ABSENT_PERSIST'].max())
trendline_y = coeffs[0] * trendline_x + coeffs[1]
plt.figure(figsize=(10, 5))

plt.scatter(
 df_cleaned['P_ABSENT_PERSIST'] + np.random.normal(0, 0.1, len(df_cleaned)),
 df_cleaned['AVG_ENG_MATH_SCORE_09'] + np.random.normal(0, 1, len(df_cleaned)),
 s=80, alpha=0.5)

plt.plot(trendline_x, trendline_y, linewidth=5)

plt.scatter(
 df_cleaned[s > s.quantile(0.90)][['P_ABSENT_PERSIST']],
 df_cleaned[s > s.quantile(0.90)][['AVG_ENG_MATH_SCORE_09']],
 color='red'
)

```



Jitter is good to present data but you need to track how the data is jittered in order to align multiple plots. The complete version is followed.

```
plt.figure(figsize=(10, 5))

Plot main bubbles

x = df_cleaned['P_ABSENT_PERSIST']
x_jitter = x + np.random.normal(0, 0.05, len(df_cleaned))
y = df_cleaned['AVG_ENG_MATH_SCORE_09']
y_jitter = y + np.random.normal(0, 0.1, len(df_cleaned))

plt.scatter(
 x_jitter,
 y_jitter,
 s=80, alpha=0.5)

Fit the curve (a line) and plot trendline

coeffs = np.polyfit(x, y, 1)

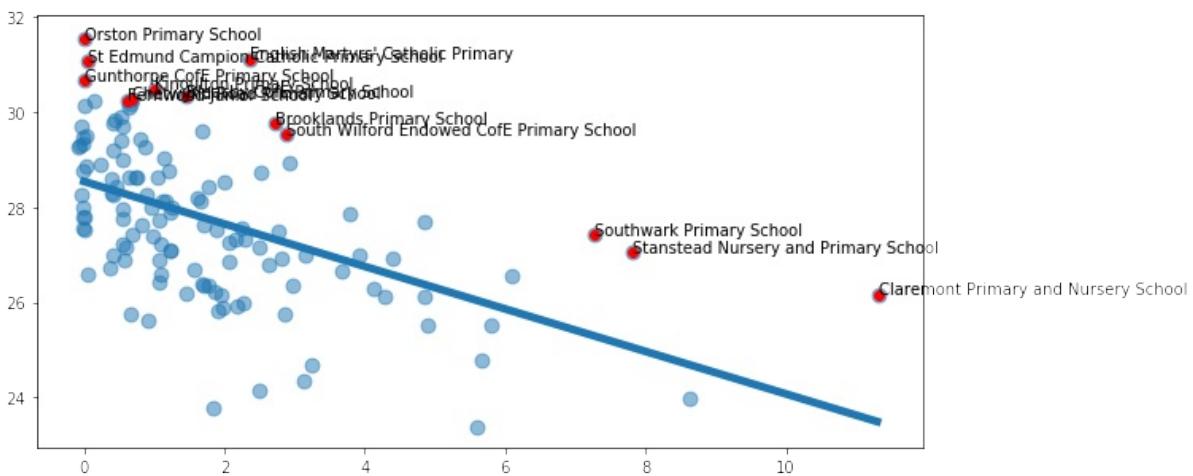
trendline_x = np.linspace(x.min(), x.max())
trendline_y = coeffs[0] * trendline_x + coeffs[1]

plt.plot(trendline_x, trendline_y, linewidth=5)

Identify suspicious schools, highlight and label texts

estimated_y = coeffs[0] * x + coeffs[1]
s = y - estimated_y

suspicious_x = x_jitter[s > s.quantile(0.90)].values
suspicious_y = y_jitter[s > s.quantile(0.90)].values
suspicious_t = df_cleaned[s > s.quantile(0.90)][['Schoolme']].values
plt.scatter(
 suspicious_x,
 suspicious_y,
 color='red'
)
for i in range(len(suspicious_t)):
 plt.text(suspicious_x[i], suspicious_y[i], suspicious_t[i])
```



## Discrete: Cross-tab

### DataFrame.groupby

Different groups with different absent rate may show different correlation with their scores. We can divide the absent rate with different groups to check out the situation here.

We name the absent rate <1 as hardworking group , absent rate 1<=rate<3 as middle group , and absent rate >3 as happy group .

```
def discretise(x):
 if x <= 1:
 x = '01_hard_working'
 elif x > 1 and x <= 3:
 x = '02_middle'
 elif x > 3:
 x = '03_happy'
 return x
f = ['mean','max','min','var','std']
df['group'] = df['P_ABSENT_PERSIST'].apply(discretise)
year9_agg = df.groupby('group')['AVG_ENG_MATH_SCORE_09'].agg(f)
year9_agg.sort_values(f,ascending=False)
```

		mean	max	min	var	std
	group					
<b>01_hard_working</b>	28.582667	31.6	25.5	1.634695	1.278552	
<b>02_middle</b>	27.241667	31.2	23.8	2.273451	1.507797	
<b>03_happy</b>	26.247500	29.7	23.5	1.644096	1.282223	

From the results, we can find the pattern that the more hardworking, the better performance students in their scores, which shows on the mean of their scores. However Middle group shows more diverse in this correlation, which can be explained by our experience. Because most of students are located in this area and the samples are more diverse. To the contrary, hardworking students get good grades and happy students get lower grades generally.

## pandas.pivot\_table

### Discretise multiple columns

```
def discretise_grade(g):
 if g >= 28:
 g = 'A'
 elif g > 26 and g <= 28:
 g = 'B'
 elif g <= 26:
 g = 'C'

 return g

discretise all all-year students scores
df['grade_07'] = df['AVG_ENG_MATH_SCORE_07'].apply(discretise_grade)
df['grade_08'] = df['AVG_ENG_MATH_SCORE_08'].apply(discretise_grade)
df['grade_09'] = df['AVG_ENG_MATH_SCORE_09'].apply(discretise_grade)
df['grade_10'] = df['AVG_ENG_MATH_SCORE_10'].apply(discretise_grade)
```

### pivot\_table to generate cross-tabs table

For example, to see whether higher score07 leads to higher score10?

```
df.pivot_table(index=['grade_07'], columns=['grade_10'], values='Schoolme', aggfunc='count')
```

grade_10	A	B	C
----------	---	---	---

grade_07			
----------	--	--	--

<b>A</b>	46	13	2
<b>B</b>	15	21	5
<b>C</b>	3	11	13

From this charts we can found out most students with grade A in year7 will still get good grades in year10. About  $46/61 = 72\%$ .

Q2: Does lower absent ratio leads to higher score08?

```
df.pivot_table(index=['group'], columns=['grade_08'], values='Schoolme', aggfunc='count')
```

grade_08	A	B	C
----------	---	---	---

group			
-------	--	--	--

<b>01_hard_working</b>	50	23	1
<b>02_middle</b>	27	31	11
<b>03_happy</b>	4	14	21

Generally, we can see that it matches to our speculation. Low-absent-rate group(hard\_working) got most A grade, about  $50/81 = 62\%$ , while only 5% of high-absent-rate group got A.

Q3: Does higher score07 and score08 leads to higher score10?

```
df.pivot_table(index=['grade_07', 'grade_08'], columns=['grade_10'], values='Schoolme', aggfunc='count')
```

	<b>grade_10</b>	<b>A</b>	<b>B</b>	<b>C</b>
<b>grade_07</b>	<b>grade_08</b>			
<b>A</b>	<b>A</b>	42.0	7.0	NaN
	<b>B</b>	4.0	4.0	1.0
	<b>C</b>	NaN	1.0	1.0
<b>B</b>	<b>A</b>	7.0	6.0	1.0
	<b>B</b>	8.0	15.0	3.0
	<b>C</b>	NaN	NaN	1.0
<b>C</b>	<b>A</b>	NaN	NaN	1.0
	<b>B</b>	NaN	6.0	5.0
	<b>C</b>	3.0	5.0	7.0

The results show highly correlation in this hypothesis. Students in year 7 and year 8 with grade higher than B, and at least one A, is more likely to get A in year 10. About  $53/70 = 76\%$ .

**Note:** From the Q2 question, we can know that there are 33 got C in year8, but from this chart, we can only got 18 C-students. What happened here? The reason here is that some of students' grade become NaN when they are in year 10. We can drop out all NaN values to check out the number whether the number is right.

```
df = df.dropna()
len(df[df['grade_08']=='C'])
18, which matches the wright answer
```

An interesting point here, the abnormal one is that 3 schools' students with grade C in year 7 and year 8 got A in year 10. What happened to those schools? We can filter out those schools.

```
df[(df['grade_07']=='C') & (df['grade_08']=='C') & (df['grade_10']=='A')]
```

	RecType	Schoolme	Address1	Address2	Address3	Town	PostCode	TelNum	SchoolType	TotPup
70	1	Carlton Central Junior School	Garden Avenue	Foxhill Road	Carlton	Nottingham	NG4 1QT	0115 9110402	CY	175
82	1	Bentinck Primary and Nursery School	Alfreton Road			Nottingham	NG7 4AA	0115 9151567	CY	268
90	1	Greenfields Community School	Orange Gardens	The Meadows	Nottingham	NG2 2JE	0115 9153762		CY	243

After we got those school names and address, next thing is to investigate on the stories behind the data.

## From correlation to causality

Seeking for causality is one of the constant pursuit of journalists. However, data and statistics can not help too much here. Correlation is an objective measure. No matter which correlation you use, as long as the mathematical formula is defined, you can get an exact number. However, causality is subjective, which can not be calculated, but can be reasoned/ articulated/ discussed. Suppose we already find the correlation between A and B, there are two directions to consider when discussing causality:

1. Whether event A happens before event B? If not, A can not be the cause of B (in a causal world/ regular world)
2. Does the domain knowledge/ physical process restrict A to be the cause of B? e.g. observing correlation intelligent parents  $\leftrightarrow$  intelligent children, we know parents should come first.

The discussion causality is hard to be thorough. That is why, as journalist, once you get the facts (data analysis/ investigation/ desktop research/ ...), it is a common practice to consult experts for comments and insights.

## Bonus: (Statistical) Hypothesis testing

Hypothesis testing is a common statistical tool used in social research domain. Suppose we have observations in  $x$ , the general process is as follows:

- Establish "null hypothesis" as  $H_0$ , which states that the observation is purely due to chance.
  - Calculate the likelihood that we have such observations under null hypothesis, i.e.  $P\{x \mid H_0\}$ . This is called "p-value".
  - If p-value is small, we reject  $H_0$  because  $x$  is not likely to happen given that condition. In other words, it is "statistically significant that  $x$  is not happening purely due to chance" -- in short,
- lower p-value  $\rightarrow$  more significant  $\rightarrow$  "more convincing"

"Think stats" by Allen has a whole Chapter 7 on hypothesis testing. We omit the detailed discussion here. The purpose of this book is to help new learners to acquire essential Python and data analytics/ visualisation skills so that they can articulate the result by "common sense"/ "data sense". Articulating in a rigorous statistical language is not a requirement here.

In the literatures, people usually put some alternative hypotheses aside  $H_0$ . By reject  $H_0$ , they conclude  $H_1$  or  $H_2$ , ... This is not always correct though. The short message that "lower p-value --> more significant" is a bit misleading. The precise version needs two more elaborations:

- The alternative hypotheses need to be stated in a mutually exclusive and collectively exhaustive way, together with  $H_0$ . Sometimes, people state the alternative not exactly the complement of  $H_0$ .
- Even if when  $H_0$  is rejected, we can not draw the conclusion on your research question directly. The rejection happens under the assumed model  $M$ . Under  $M$ ,  $x$  is unlikely to happen due to chance. There must be something (some cause). However, this do not give us information on how likely  $M$  itself is correct.

The short take-away is: Use hypothesis with caution. When in doubt, just do a full reporting on all relevant experiments, tests, results. Leave it to the readers to draw their own conclusions.

## Reference

- Downey, A. B. (2014). Think stats: exploratory data analysis. O'Reilly Media, Inc. Retrieved from [https://the-eye.eu/public/Books/IT%20Various/think\\_stats.pdf](https://the-eye.eu/public/Books/IT%20Various/think_stats.pdf)

# Week 11: Present findings - data visualization and reproducible report

- [Week 11: Present findings - data visualization and reproducible report](#week-11-present-findings---data-visualization-and-reproducible-report) - [Objective](#objective) - [Data Visualization Libraries](#data-visualization-libraries) - [matplotlib](#matplotlib) - [Quickstart for matplotlib](#quickstart-for-matplotlib) - [How to order the keys of bar chart](#how-to-order-the-keys-of-bar-chart) - [How to add annotation in matplotlib](#how-to-add-annotation-in-matplotlib) - [How to plot multiple chart in one input/ output cell](#how-to-plot-multiple-chart-in-one-input-output-cell) - [seaborn](#seaborn) - [Quickstart for seaborn](#quickstart-for-seaborn) - [Plot bar-charts and other charts](#plot-bar-charts-and-other-charts) - [plotly](#plotly) - [Quickstart for plotly](#quickstart-for-plotly) - [pyecharts](#pyecharts) - [Quickstart for pyecharts](#quickstart-for-pyecharts) - [pandas](#pandas) - [Quickstart for pandas](#quickstart-for-pandas) - [bokeh](#bokeh) - [Quickstart for bokeh](#quickstart-for-bokeh) - [ggplot](#ggplot) - [Quickstart for ggplot](#quickstart-for-ggplot) - [Data visualization Principles](#data-visualization-principles) - [Principle](#principle) - [Charts](#charts) - [Dashboard](#dashboard) - [GitHub repo](#github-repo) - [README.md](#readmemd) - [Presenting dataset](#presenting-dataset) - [Jupyter notebook](#jupyter-notebook) - [Write notes in Jupyter notebook](#write-notes-in-jupyter-notebook) - [Display the picture](#display-the-picture) - [Add HTML link](#add-html-link) - [Publish work on GitHub Pages](#publish-work-on-github-pages) - [Basic HTML](#basic-html) - [Bonus: CSS](#bonus-css) - [Single column layout](#single-column-layout) - [Integrated exercise: Publish a full work in a stand alone page](#integrated-exercise-publish-a-full-work-in-a-stand-alone-page) - [Save plotly chart and put it on gh-pages](#save-plotly-chart-and-put-it-on-gh-pages) - [Bonus: Continuously update GitHub Pages](#bonus-continuously-update-github-pages) - [Bonus: Slide show presentation](#bonus-slide-show-presentation) - [Bonus: Craft a data service](#bonus-craft-a-data-service) - [Code of conduct: Reproducible reporting and full reporting](#code-of-conduct-reproducible-reporting-and-full-reporting) - [References](#references)

## Objective

- `matplotlib`
- `seaborn`
- `plotly`
- `pyecharts`
- `pandas`

## Data Visualization Libraries

Demo data: [open rice data](#)

- 1D: plot price range bars
- 2D: plot price range bars w.r.t areas
  - Use faceting, i.e. multiple sub plots in one plot
  - Use grouping, i.e. grouped bar chart (can select a subset of areas)

### matplotlib

`Matplotlib` is the "goto library" for data visualization in Python. Many other libraries like built-in visualization routines in `pandas` and the famous statistics friendly library `seaborn` are based on `matplotlib`. It provides plotting commands to make Python work in `MATLAB` style.

## Quickstart for matplotlib

Preamble:

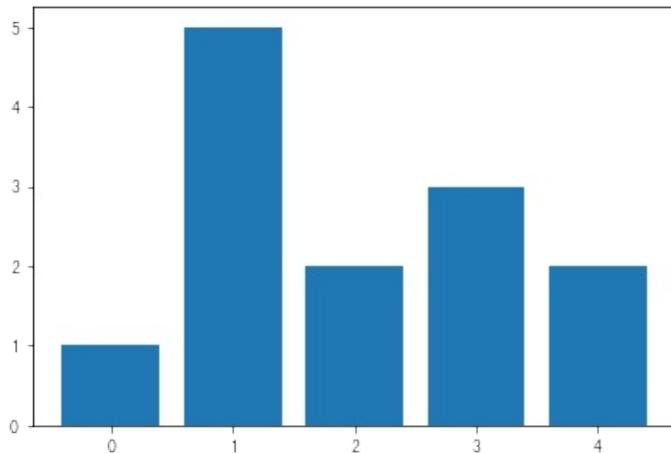
```
from matplotlib import pyplot as plt
%matplotlib inline
%config InlineBackend.figure_format = 'svg'
```

Note, the 3rd line above allows the notebook to render images in vector format. SVG is a common format to display vector graphics on the web. The output image on web will be blurred without this line (raster image).

Basic plotting:

```
from matplotlib import pyplot as plt
import pandas as pd
data = [1, 5, 2, 3, 2]
df = pd.DataFrame(data, columns=['value'])
#df

Pass x label value and y label value
plt.bar(df.index, df.value)
The following line can be omitted if the .bar()
is the last called function in this cell
plt.show()
```



## How to order the keys of bar chart

**Note:** Matplotlib doesn't support displaying Chinese characters, we need to do some setup work here. Please refer [the tutorial](#).

```
-*- coding: utf-8 -*-
import pandas as pd
from matplotlib import pyplot as plt
df = pd.read_csv('openrice_viz.csv')
#df.head()

#set for displaying Chinese characters here.
plt.rcParams['font.sans-serif']=['SimHei']

#set for displaying `-
plt.rcParams['axes.unicode_minus']=False

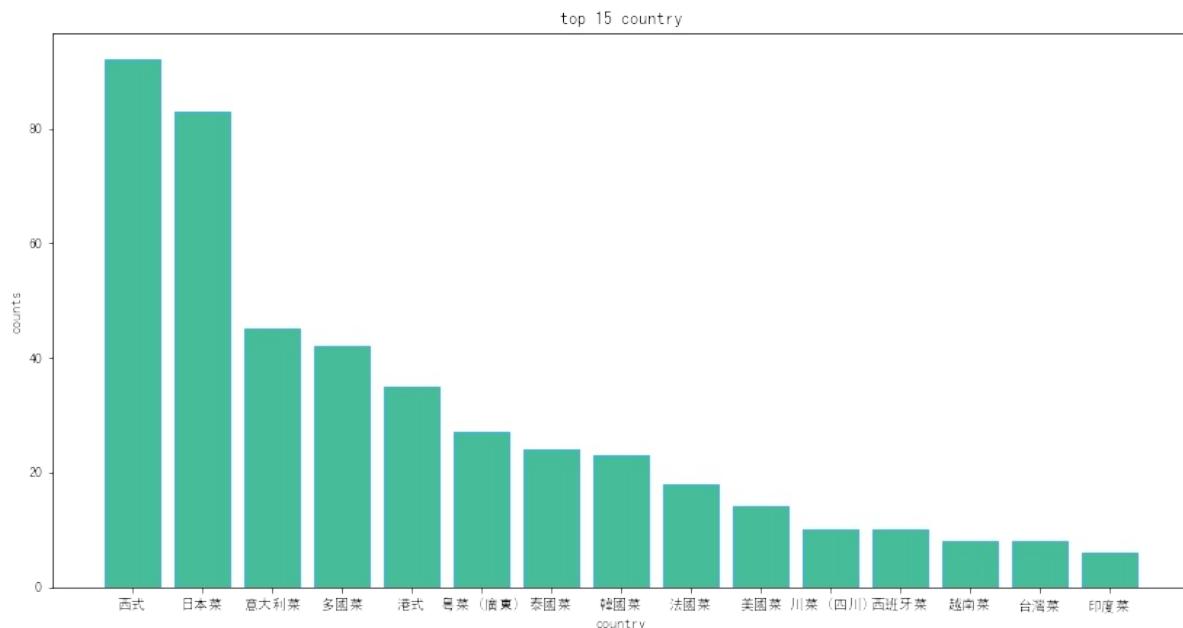
#sort values
country_counts=df['country'].value_counts()[:15].sort_values(ascending=False)
```

```
#adjust size
country = pd.DataFrame(country_counts)
fig = plt.figure(figsize=(14,7))

#change color of the bars
plt.bar(country.index, country.country,color = '#46bc99',edgecolor = '#40b4e5')

another way to plot bar in pandas:
country_counts=df['country'].value_counts()[:15].sort_values(ascending=False).plot(
kind='bar',color = '#46bc99',edgecolor = '#40b4e5')

#plot title and label name
plt.title('top 15 country')
plt.xlabel('country')
plt.ylabel('counts')
plt.show()
```

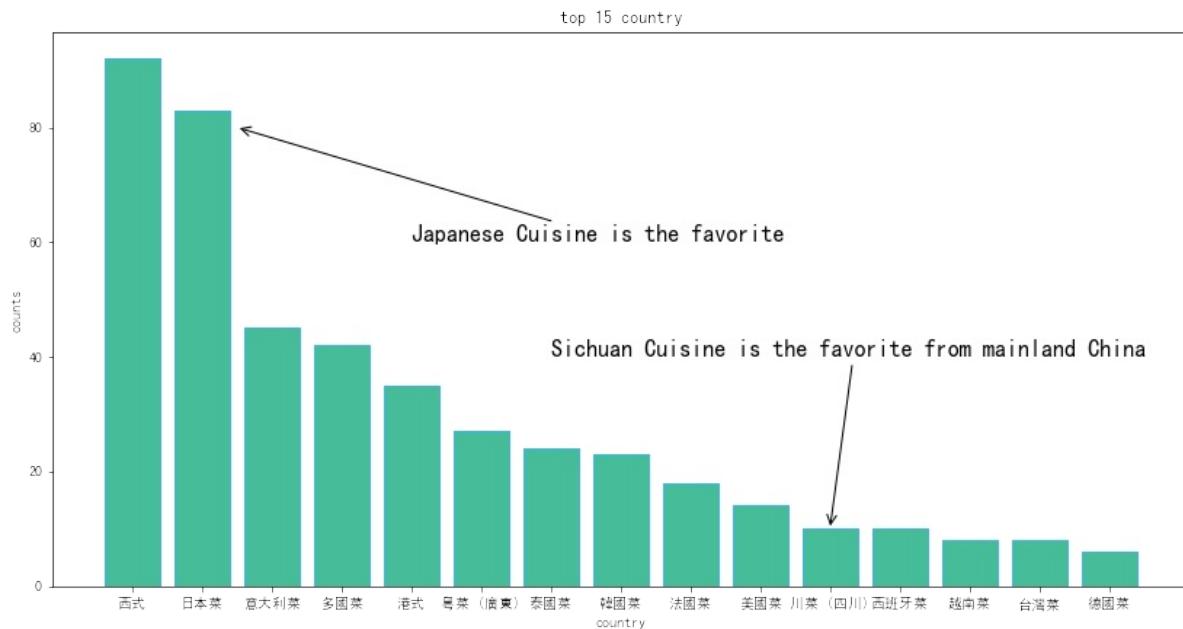


## How to add annotation in matplotlib

Sometimes, one may need to leave annotations to highlight or explain some data to help readers better understand the story. The following is an example of how to make annotation in matplotlib.

```
plt.rcParams['font.sans-serif']=['SimHei']
plt.rcParams['axes.unicode_minus']=False
fig = plt.figure(figsize=(14,7))
plt.bar(country.index, country.country,color = '#46bc99',edgecolor = '#40b4e5')
plt.title('top 15 country')
plt.xlabel('country')
plt.ylabel('counts')

you can change arrow style, color, text and text location
plt.annotate('Japanese Cuisine is the favorite', xy=(1.5, 80), xytext=(4, 60), fontsize=16,
 arrowprops=dict(arrowstyle='->', facecolor='black'))
plt.annotate('Sichuan Cuisine is the favorite from mainland China ', xy=(10,10), xytext=(6, 40), fontsize=16,
 arrowprops=dict(arrowstyle='->', facecolor='black'))
```

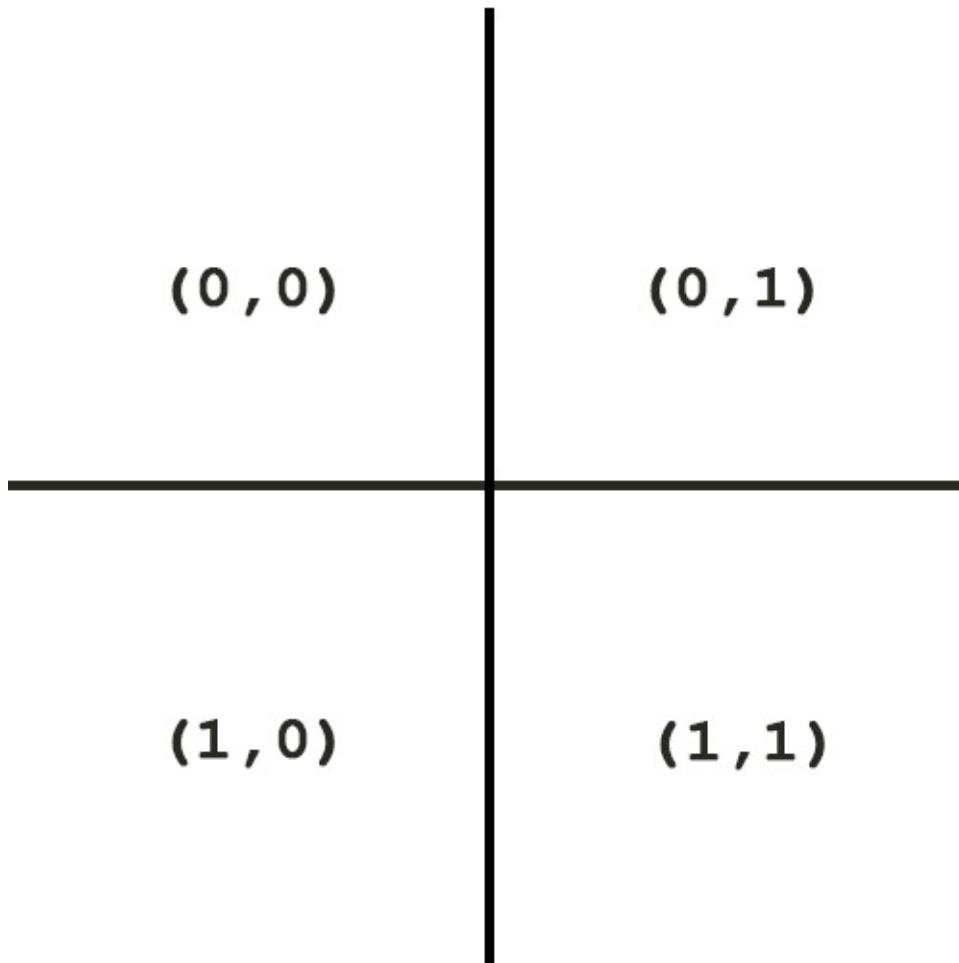


Also, you can refer [here](#) for another example by one of S2018 students.

## How to plot multiple chart in one input/ output cell

Sometimes, we can use `plt.subplots` function to plot multiple charts into one output cell, so that we can more easily to compare and tell the difference between different parameters.

```
#set a 2*2 canvas, adjust layout to more flexible
#adjust figure size, axes means the location of each subplots
#you can refer to the following picture below to learn more.
fig, axes = plt.subplots(nrows=2, ncols=2, constrained_layout=True, figsize=(30,20))
```



```
#plot price range count
price = pd.DataFrame(df['price'].value_counts())
ax1 = price.plot(kind = 'bar',color = '#46bc99',edgecolor = '#40b4e5',ax=axes[0,0],fontsize=24)
ax1.set_title("Price range count",fontsize=40)

#plot country count
country = pd.DataFrame(df['country'].value_counts()[:15])
ax2 = country.plot(kind = 'bar',color = '#46bc99',edgecolor = '#40b4e5',ax=axes[0,1],fontsize=24)
ax2.set_title("Country count",fontsize=40)

#plot type count
type = pd.DataFrame(df['type'].value_counts()[:15])
ax3 = type.plot(kind = 'bar',color = '#46bc99',edgecolor = '#40b4e5',ax=axes[1,0],fontsize=24)
ax3.set_title("Type count",fontsize=40)

#plot likes and bookmark scatter
likes_bookmark = df[['likes','bookmark']]
ax4 = likes_bookmark.plot(kind = 'scatter',x='likes',y='bookmark',color = '#46bc99',ax=axes[1,1],s=80,fontsize=24)
ax4.set_title("Like with Bookmark count",fontsize=40)
```



#### Note:

1. You can pass a lot of parameters like `kind`, `color`, `fontsize` into the function. For more usage examples, please refer [the documentation](#)
2. Axes is just like the position of the subplots. You can refer to the following picture for better understanding.

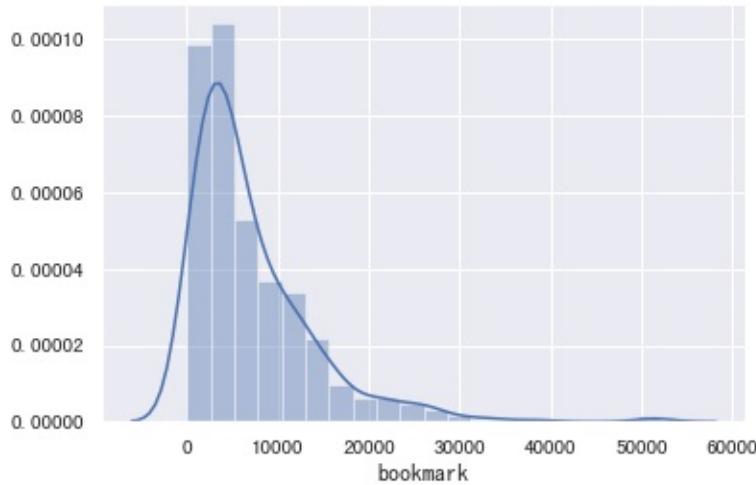
## seaborn

Seaborn is a Python data visualization library based on matplotlib, basically, you can regard it as the advanced version of matplotlib, and its closely integrated with `pandas data structures`, with which we can draw more attractive and informative statistical graphics. You can refer [it's documentation](#).

## Quickstart for seaborn

Basic usage example: (`pip install seaborn` if you have not done so yet)

```
import seaborn as sns
#one can directly pass pandas.series to plot histogram
sns.distplot(df["bookmark"],bins=20)
```



## Plot bar-charts and other charts

Example: Draw a bar chart between price range and likes, you can easily find that \$200-400 is the most popular price and acceptable range in Hong Kong.

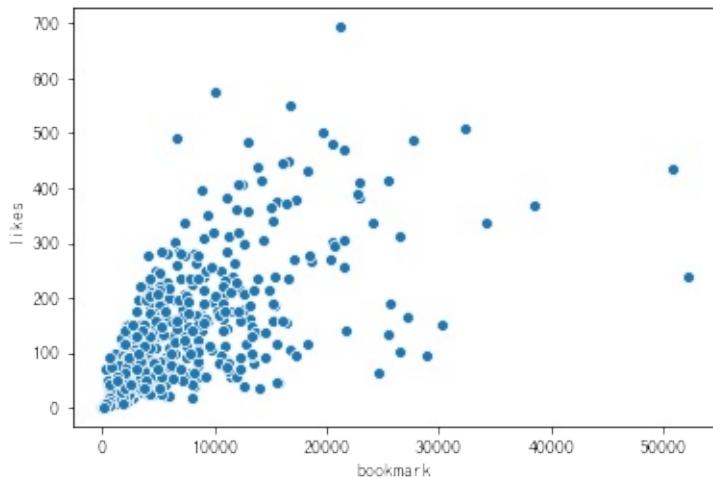
```
#set font to support Chinese character
sns.set(font='SimHei')

pd_df = df.groupby(['price'])['likes'].mean().reset_index().sort_values("likes", ascending=False)
ax = sns.barplot(x='price', y='likes', data=pd_df)
plt.title('likes_price', color='gray', fontsize=16, weight='bold')
```



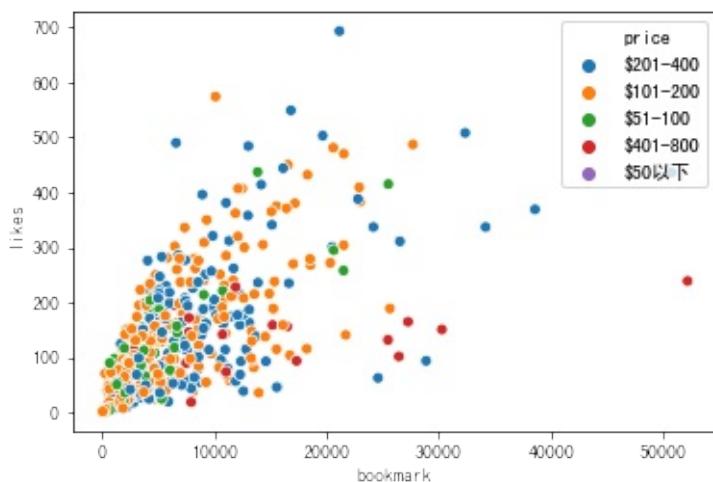
Example 2: Draw a scatter plot between bookmarks and likes to quickly checkout whether there is a relationship.

```
import seaborn as sns
ax = sns.scatterplot(x="bookmark", y="likes", data=df)
```



Apart from two dimensional analysis, seaborn can handle more complicated cases. We can add another parameters called `hue`. Hue is the name of variables in data or vector data, its an optional argument. Grouping variable that will produce points with different colors. It can be either categorical or numeric.

```
ax = sns.scatterplot(x="bookmark", y="likes", hue='price', data=df)
```



For more seaborn examples, you can refer [the official tutorial](#).

## plotly

Plotly is very powerful to make interactive, publication-quality graphs online. Including line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts. If you want to present and publish your work on html, with some fancy appearance and interactive experience, Plotly is a very recommended library.

## Quickstart for plotly

Basic usage example: (`pip install plotly` if you have not done so yet)

```
import plotly
"go" is short for Graph Object.
You can find many graph models under "go"
import plotly.graph_objs as go
You don't have to register and serve API keys using the offline mode.
```

```

You can try to import iplot and plot from plotly directly (without "offline").
In that way, it prompts a message to ask for API keys.
from plotly.offline import iplot, plot

If not init notebook with connection,
the iplot won't output chart onto Jupyter notebook webpage.
plotly.offline.init_notebook_mode(connected=True)

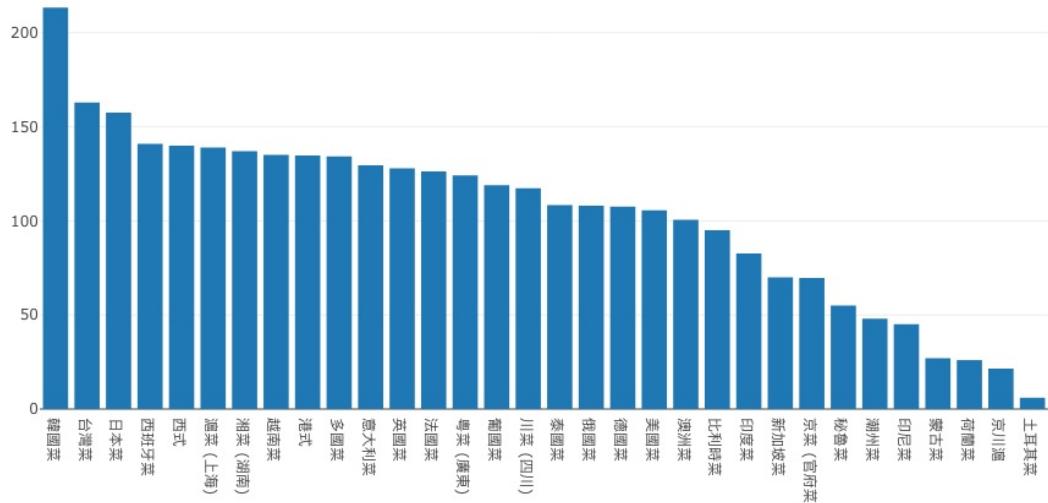
Calculate average likes for a country and sort in its descending order
pd_df2 = df.groupby(['country'])['likes'].mean().reset_index().sort_values("likes", ascending=False)

Organize data into plotly data structure
data = [
 go.Bar(
 x=pd_df2.country,
 y=pd_df2.likes)
]

Following line outputs your chart into "hello.html" in the CWD
Uncomment to see the result yourself.
plot(data, filename='hello')

Following version, aka "inline plot" plots the chart into Jupyter notebook directly
iplot(data, filename='hello')

```



For more plotly examples and tutorials, you can refer to [official documentation](#)

## pyecharts

Pyecharts is a library to generate charts using Echarts, which is an open source library from Baidu for data visualization in javascript. Pyecharts provides 30+ kinds of charts, especially with easy-to-use interactive graphs.

## Quickstart for pyecharts

Basic usage example: ( pip install pyecharts if you have not done so yet)

```

#you can change Bar to other kind of charts, like Line, Pie, HeatMap etc...
from pyecharts import Bar

#to see relationship between countries with likes and bookmarks

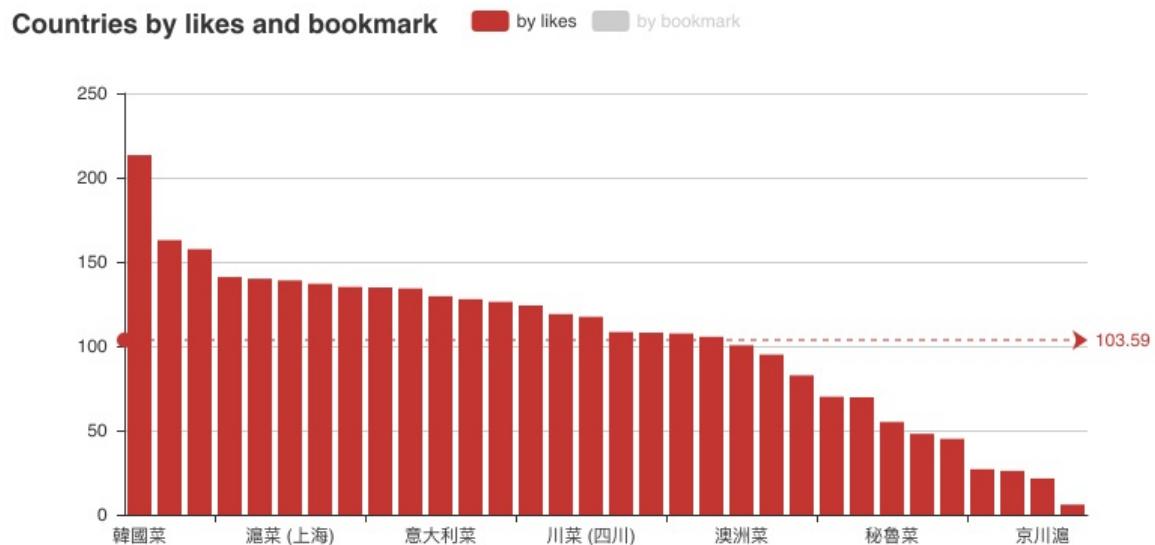
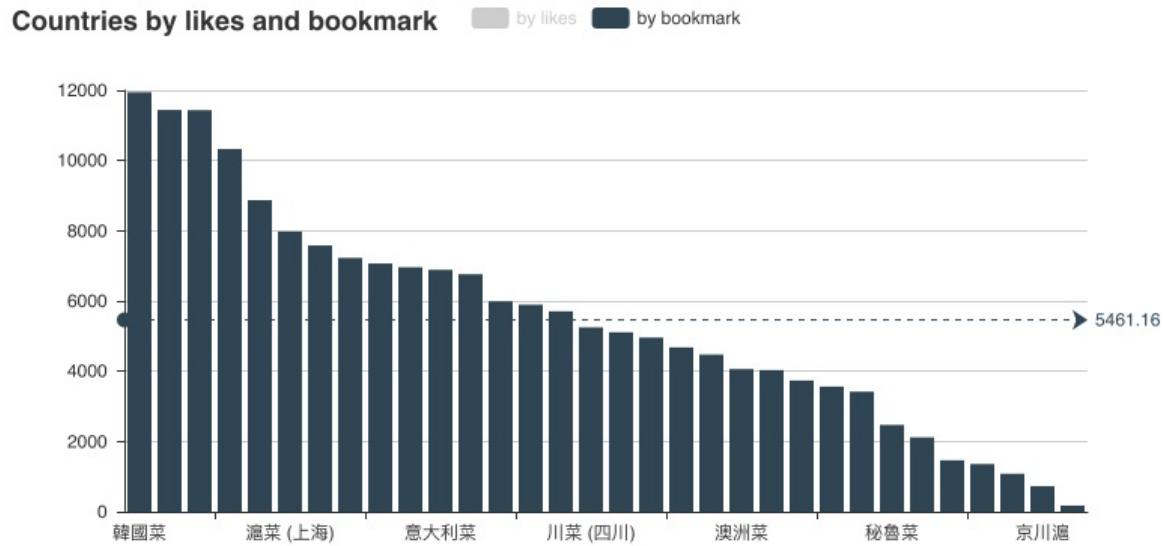
```

```

pd_df3 = df.groupby(['country'])['bookmark'].mean().reset_index().sort_values("bookmark", ascending=False)

#you can pass a list-like data here
attr = pd_df2.country
v1 = pd_df2.likes
v2 = pd_df3.bookmark
bar = Bar("Countries by likes and bookmark")
bar.add("by likes", attr, v1, mark_line=["average"])
bar.add("by bookmark", attr, v2, mark_line=["average"])
bar

```



For more pyecharts examples, you can refer [the official tutorial](#)

## pandas

One can also include "bar charts" in your DataFrame, from which you can easily find the distribution and the extreme values. For example:

## Quickstart for pandas

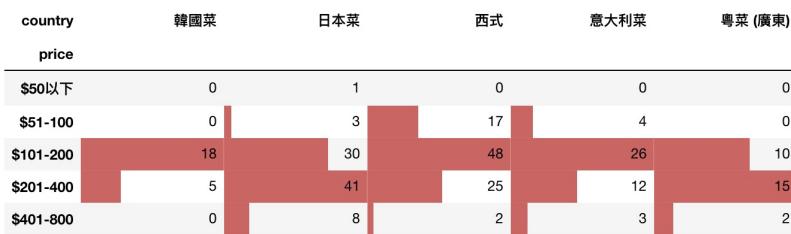
```

pd_df4 = df.pivot_table(index=['country'], columns=['price'], values='name', aggfunc='count')
#pd_df4

#select rows with popular cuisine
#changes rows to columns for better overview of each cuisine price range
#replace the nan value with 0
pd_df4 = pd_df4.loc[['韓國菜', '日本菜', '西式', '意大利菜', '粵菜 (廣東)']].fillna(0).T

the index is string, not the number, so we cannot directly sort_index
instead, we need to reindex it.
reorderlist = ['$50以下', '$51-100', '$101-200', '$201-400', '$401-800']
pd_df4 = pd_df4.reindex(reorderlist)
pd_df4.style.bar(color="#d65f5f")

```



For more inside pandas.DataFrame usage, please refer [it's documentation](#)

## bokeh

Bokeh is an interactive visualization library that targets modern web browsers for presentation.

### Quickstart for bokeh

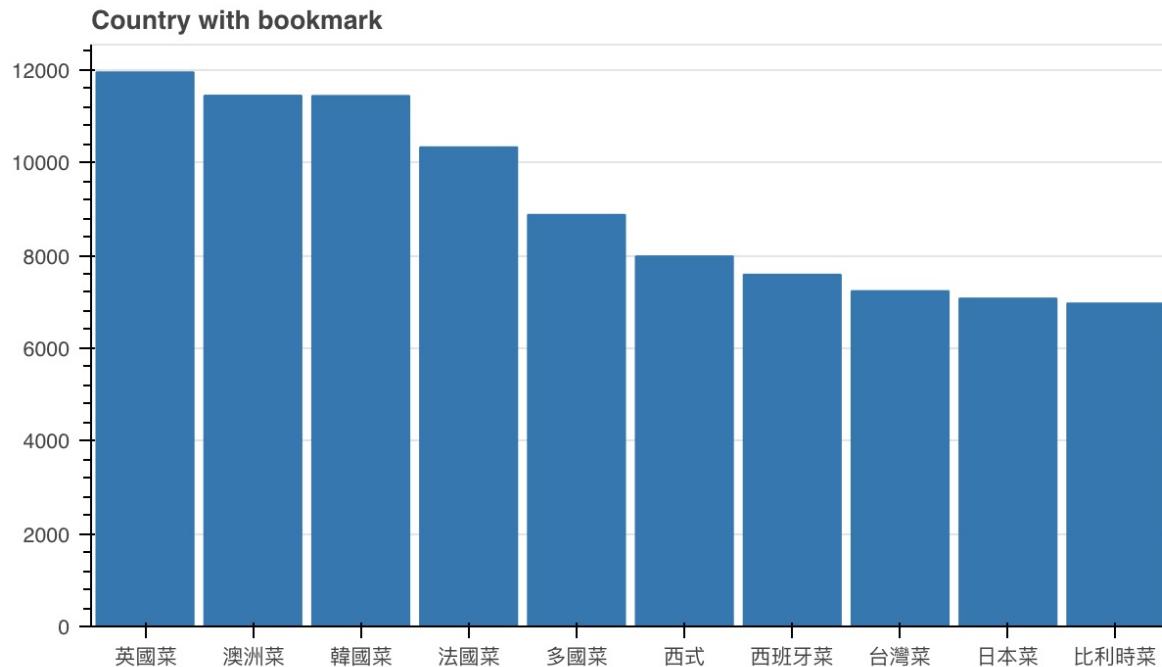
Basic usage example: ( `pip install bokeh` if you have not done so yet)

Plot top 10 country grouped with bookmark, and sorted by bookmark.

```

from bokeh.plotting import figure
from bokeh.io import show, output_file
output_file("bokeh_bar.html")
pd_df5 = df.groupby(['country'])['bookmark'].mean().reset_index().sort_values("bookmark", ascending=False)[:10]
source = ColumnDataSource(data=pd_df5)
p = figure(x_range=pd_df5.country, plot_height=350, title="Country with bookmark",
 toolbar_location=None, tools="")
p.vbar(x=pd_df5.country, top=pd_df5.bookmark, width=0.9)
p.xgrid.grid_line_color = None
p.y_range.start = 0
show(p)

```



For more bokeh examples, you can refer [the tutorial](#)

## ggplot

There is a Python implementation of the very famous `ggplot` that is also available in R language and other standalone tools. It is based on the Grammar of Graphics and is the Swiss Knife for visual analysis. However, the flexibility might impose a sharper learning curve on beginners. Other "model-based" charting libraries are usually easier to get started: 1) select a proper model (bar/ scatter/ line/ ...); 2) fit your data into the model.

## Quickstart for ggplot

**TODO:** some examples will be added here later.

# Data visualization Principles

**NOTE:** This section are on course slides.

## Principle

## Charts

## Dashboard

## GitHub repo

## README.md

"README" is a convention in computer world. You can find a file of this name in almost all software distribution. It means exactly the same as its name indicates: before you do anything, read me first! This file usually gives people instructions on first steps to work with the software. It may point to other more detailed tutorials or manuals. The `.md` in `README.md` is just a suffix to indicate that this file is written in markdown format. When GitHub sees this file, it renders the file into HTML (for your web browser) using a markdown compiler. You can check out the [README.md](#) of this current repo to get an idea.

Besides giving important project information and "play the open source way", a good README file is also an "elevator pitch" to potential readers/ users. You want to present the key functions/ highlights in quick/ direct way. Here are some tips for your consideration.

- Use one sentence to summarise the project. You can use analogy to help visitors quickly comprehend the content.
- Include a "demo" section to show the outcome. Screenshots may help.
- Include a "quickstart" to show user the painless operations that can give some preliminary results.
- Include a "license" section to make the file look professional
- A [tutorial](#) to use GIFs to better present your work on GitHub.

## Presenting dataset

When you present a dataset on GitHub, following information is helpful for the readers to quickly understand your project:

- **topic**
- data source
- data fields (type, sample data)
- data volume
- **license**
- obstacles and solutions
- future work

License is easy to forget. Some serious users may not use your project if there is no permissive license. One can refer to [this section](#) for some common licenses in the open source world. The suggested license as a default:

- If your work is reusable code, using `MIT` is common.
- If your work is creative content, either dataset or article, using `CC 4.0 BY` is common.

## Jupyter notebook

Jupyter notebook is very convenient and powerful to present your work, you can write notes by markdown, you can insert url links, pictures, interactive graphs, and of course, codes. Therefore, in most cases, one notebook is enough for us to present and share our works. The following is the introduction of how to use jupyter as the primary presenting method.

### Write notes in Jupyter notebook

We can change the cell type to present non-codes content. Click `cell --> cell type --> Markdown`. Then write the notes and stories you like, Jupyter also support markdown syntax.

The screenshot shows a Jupyter Notebook interface. The menu bar includes File, Edit, View, Insert, Cell, Kernel, and Help. The Cell menu is open, showing options like Run Cells, Run Cells and Select Below, Run Cells and Insert Below, Run All, Run All Above, Run All Below, Cell Type (with sub-options Code, Markdown, Raw NBConvert), Current Outputs, and All Output. A tooltip 'vn' is shown above the menu. The main area displays a table titled '# Top 15 directors in the world' with columns: Director, Year, Country, and Runtime. The table lists directors such as Jean-Luc Godard, Peter Weir, Woody Allen, Bruno Dumont, Lee Chang-dong, etc. Below the table is a bar chart titled 'top 15 directors' showing the number of movies directed by each director. The x-axis lists directors and the y-axis shows the count of movies from 0 to 16. The chart shows Jean-Luc Godard with 16 movies, Luis Bunuel with 15, John Ford with 14, Alfred Hitchcock with 13, Ingmar Bergman with 12, Akira Kurosawa with 12, Jean Renoir with 11, Stanley Kubrick with 11, Howard Hawks with 11, Federico Fellini with 10, Roberto Rossellini with 9, Fritz Lang with 9, Robert Bresson with 9, Charles Chaplin with 9, and Martin Scorsese with 9.

For shortcut:

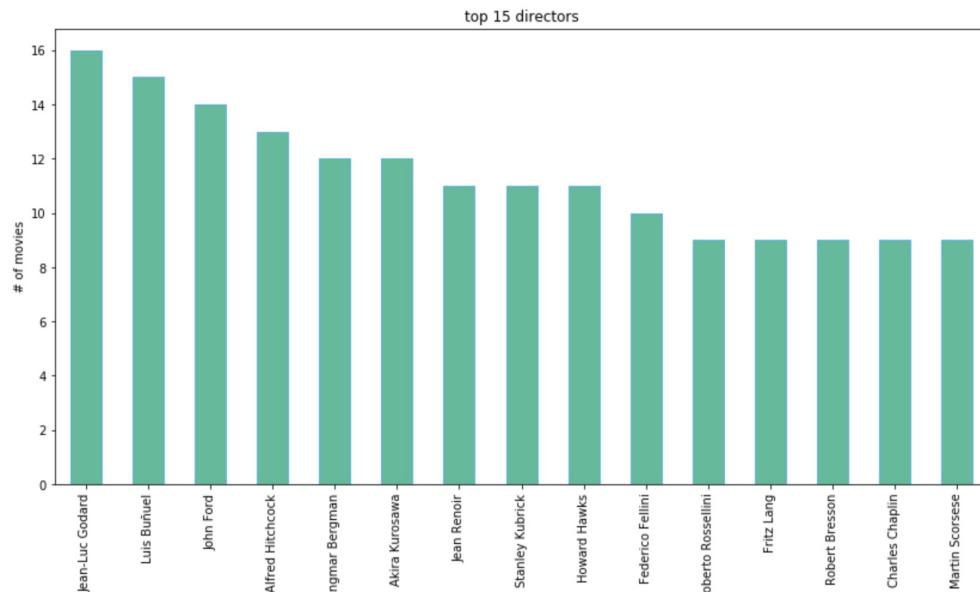
- type `m` to change cell to markdown mode
- type `y` to change cell to code mode

## Display the picture

Assuming that you have the pictures in the folder where your current jupyter notebook are. You can display the picture by the following method.

```
from IPython.display import Image
#change the name corresponding your own file
Image("top 15 directors.png")
```

```
from IPython.display import Image
Image("top 15 directors.png")
```



## Add HTML link

We can also insert `html` in jupyter. For example:

```
from IPython.core.display import HTML
HTML('Openbook')
```

Then you will find a clickable text `openbook` linking to the repo. For code blocks, you can write by `'''` to quote codes.

```
HTML('''
Openbook

item 1
item 1

''')
```

```
HTML('''
Openbook

item 1
item 1

''')
```

[Openbook](#)

- item 1
- item 1

Bonus: For better presentation, we need to clear the long and working-in-progress output. Click `cell --> Current Output --> Clear` can solve help you accomplish this.

The screenshot shows a Jupyter Notebook interface. In the top navigation bar, the 'Cell' menu is selected. A dropdown menu is open under 'Cell', showing options like 'Run Cells', 'Run All', etc., followed by a separator line, then 'Current Outputs', 'All Output', and finally 'Clear'. The 'Clear' option is highlighted with a red box. Below the menu, there are two sections: 'In [249]:' containing the command `df` and 'Out[249]:' showing a table of movie data. The table has columns: Directors, Ranking, year, region, length, and genre. The data includes entries for Orson Welles, Alfred Hitchcock, Stanley Kubrick, Federico Fellini, F.W. Murnau, John Ford, and Akira Kurosawa.

Directors	Ranking	year	region	length	genre
Orson Welles	1	1941	USA	119m	Drama
Alfred Hitchcock	2	1958	USA	128m	Romantic Mystery
Stanley Kubrick	3	1968	UK-USA	139m	Psychological Sci-Fi
		1939	France	113m	Comedy Drama
		1953	Japan	134m	Drama
Federico Fellini		1972	USA	175m	Gangster Film
F.W. Murnau			Italy	135m	Satire
John Ford			USA	110m	Melodrama
Akira Kurosawa			USA	119m	Western
			Japan	200m	Drama

Also we can write html codes at the beginning of your Jupyter notebook, which can generate a link that only present the cells with output. You can click the link to toggle between raw codes and plain documentation.

```
HTML('''<script>
code_show=true;
function code_toggle() {
 if (code_show){
 $('div.input').hide();
 } else {
 $('div.input').show();
 }
 code_show = !code_show
}
$(document).ready(code_toggle);
</script>
The raw code for this IPython notebook is by default hidden for easier reading.
To toggle on/off the raw code, click here.'''')
```

```
HTML('''<script>
code_show=true;
function code_toggle() {
 if (code_show){
 $('div.input').hide();
 } else {
 $('div.input').show();
 }
 code_show = !code_show
}
$(document).ready(code_toggle);
</script>
The raw code for this IPython notebook is by default hidden for easier reading.
To toggle on/off the raw code, click here.''')
The raw code for this IPython notebook is by default hidden for easier reading. To toggle on/off the raw code, click here.
```

The raw code for this IPython notebook is by default hidden for easier reading. To toggle on/off the raw code, click [here](#).

## Top 15 directors in the world

Through the pic of top 15 directors, I found that French director Jean-Luc Godard has most films listed in this chart with 16 films. One thing I want to mention is that quality is not only criteria but quantity also matters.

# Publish work on GitHub Pages

## Basic HTML

HTML is a declarative language. One only needs to "declare" what content is on the page, using "tags". HTML tags come in pairs, in the form like `<tag></tag>`. Tags can be nested so some content can be put inside other content (container tags) ("phrase" and "flow" element in HTML language). You can readily start building a web page by modifying other's code. Following tags are common:

- h1/ h2/ h3
- p
- img
- a
- ul/ ol/ li
- strong, em
- iframe

Note that, `iframe` is commonly used to embed external resources into the current web page.

## Bonus: CSS

CSS can be used to style the page. They usually come in three places:

- The `<style>` tag
- The `style=""` attribute in HTML element
- Use `<link rel="stylesheet" src="">` to include external style files.

Detailed explanation is omitted from this open book. To get started, you don't have to worry about CSS. Most common way of practice is to get some existing works and modify the content via HTML directly.

## Single column layout

With the wide spread of mobile devices, single column layout is trending. That is, there is only one column on every row. The web page is an interleaved layout of images and texts. No two images appear side by side. No two paragraphs appear side by side. If you need to do so, try to edit those two images into one outside the scope of HTML

so that one `img` tag is enough for the presentation.

## Integrated exercise: Publish a full work in a stand alone page

[Big Road](#) is a minimalist solution to make sensible web stories on GitHub. You can follow the instructions there to put your stories on the web. It features:

- Only HTML
- Mobile-only and single column layout
- No CSS (actually we do have CSS there; but you don't have to worry about it)
- Responsive embedding
- Interactive chart demo

### Note:

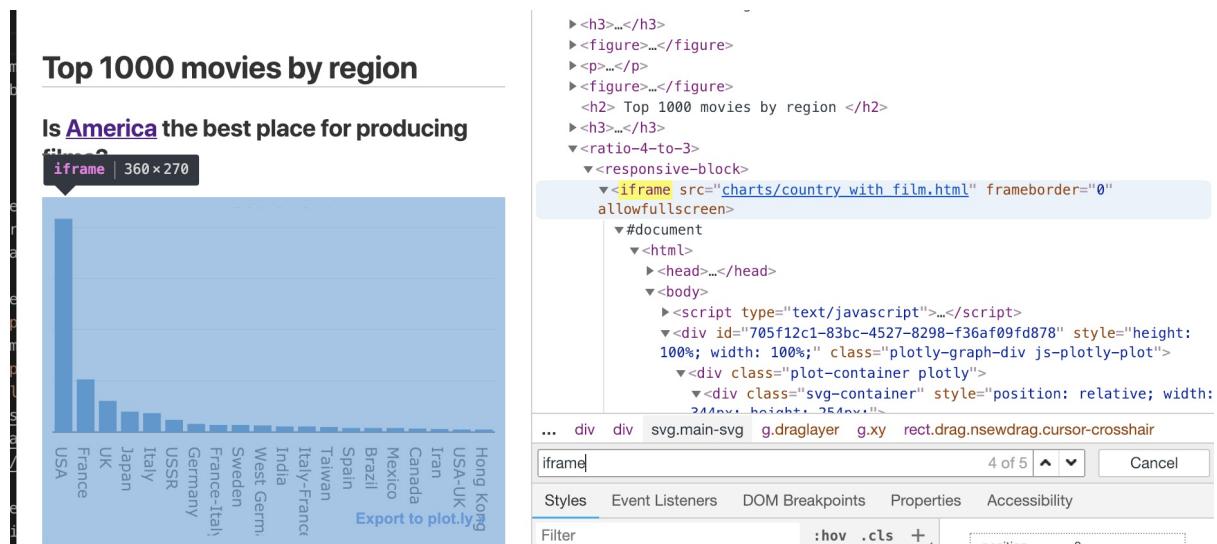
1. For the image size, the 640px width is usually enough for most occasions. A too much larger image will cause longer loading time, in which situation, you will lose your users. You can resize your image size by Photoshop or other picture processing apps.
2. Some html generated by pyecharts cannot fit in this template completely. To display a dynamic chart completely, one can initialize the pyecharts object by giving it a relative width/ height number, like the following, the HTML files rendered in this way can fit into Big Road responsive chart area easily. You can refer [one responsive pyecharts example](#).

```
chart = pyecharts.Bar(width='100%', height='90vh')
```

## Save plotly chart and put it on gh-pages

We can use offline mode of plotly to save the charts as html files like the example we used [above](#).

After we get htmls, we can embed them into other html with help of `iframe`. You can refer [this case](#) for more details.



**Note:** The default plotly chart includes a tool bar, making the graphical region too small on "Big Road" template. There are two ways to work around:

1. Use `<ratio-1-to-1>` tag to wrap the `<responsive-block>`.
2. Remove the tool bar from plotly chart. Use `"displayModeBar": False` in `config`. Checkout [sample code](#)

The second way is recommended. When using the 1st solution, there will be a large chunk of blank on the page. This area is intended to show the tool bar when hovering your mouse on the chart. Hovering does not make sense on mobile devices.

## Bonus: Continuously update GitHub Pages

The GitHub repository can be updated continuously to ensure the data presented there is latest. One common strategy is:

- Use Python to handle the data collection and data processing.
- Generate JSON data files to interface between Python and JavaScript.
- Build interactive charts that takes JSON data as input, from a designated location on gh-pages.
- Periodically run the Python script to make the data up to date. One may find `cron` on Linux or Mac OSX helpful.

## Bonus: Slide show presentation

[Reveal.js](#) is the most commonly used library to present slides on the web. It is a JavaScript library that parses special HTML tags and present the slides. It may be too verbose for casual use. A lightweight wrapper is [reveal-md](#) that allows one to write content in markdown and turn it into web based slideshow instantly. [RISE](#) is a Jupyter notebook extension that adds a "slide" cell type. Using `RISE`, one can switch between notebook and slideshow instantly. Moreover, `RISE` allows one to execute codes on the webpage directly.

## Bonus: Craft a data service

Two components of the system:

- Frontend -- things that a regular users can see
- Backend -- things that a regular users can not see, and usually unaware of

Frontend is based on the languages: HTML/ CSS/ JavaScript. Backend can have many different alternatives, notably `Python` , `JavaScript` (`NodeJS`), `Ruby` and `PHP` . There are two styles of connecting frontend and backend:

- Server Side Rendering -- The web framework of the backend compiles HTML/ CSS/ JavaScript and then send to the user browser in a batch. This way is more efficient in terms of execution but not less flexible in web design.
- Client Side Rendering -- The backend only provisions API (the same concept you encountered in week-04). Frontend reads data from API and assembles the web page for user's action. This is the current mainstream approach. This way decouples frontend and backend, and has certain engineering advantages.

There are some useful libraries in Python for you to build a backend:

- `django` -- a very comprehensive library, with a rich eco-system.
- `flask` -- medium weight library. It is recommended for a serious but small scale project.
- `bottle` -- light weight library. One can build up a web service in 10 minutes. It is good as an initial trial.

## Code of conduct: Reproducible reporting and full reporting

- Reproducible reporting: Using Jupyter notebook can make most of the content reproducible by default. The readers can find dataset, codes, results, charts and explanation all in one place. However, note that
  - Sometimes you execute the cells in the Jupyter notebook in a different order from their appearance on the notebook. That is normal in the trial and error stage. Before you publish the notebook, make sure you restart the kernel and execute from beginning to end in one batch. This makes sure the other readers can **reproduce** the notebook.

- Sometimes, manual intervention is required in the process, e.g. clicking a button, substituting cookies, etc.  
You need to put down notes where the workflow is not fully automated. The core concept of reproducible reporting, is to make sure the readers can reproduce your result, either by code, or by human operation.
- Full reporting: not only report the successful instances; but also report the unsuccessful instances. Sometimes, you find contradictory results from one dataset. Many authors are selective. In our code of conduct, you need to do a full reporting at one intermediate stages, i.e. showing the possible results/ alternative results that you already find. In the narratives of final report, being selective is usually one necessary evil to make a compelling story. You need to make sure you don't over-state anything.

## References

- First two chapters (i.e. before "3D") of the article [The Art of Effective Visualization of Multi-dimensional Data](#) by Dipanjan Sarkar
- 

If you have any questions, or seek for help troubleshooting, please [create an issue here](#)

# Week 12: Text data

- [Week 12: Text data](#week-12-text-data) - [Text processing](#text-processing) - [String functions recap](#string-functions-recap) - [Case 1: Get full URL from HTML A tag's href attribute](#case-1-get-full-url-from-html-a-tags-href-attribute) - [Case 2: Substring matching and substitution](#case-2-substring-matching-and-substitution) - [Case 3: Most frequent names in tweets](#case-3-most-frequent-names-in-tweets) - [encode and decode](#encode-and-decode) - [String matching and Regular Expression (RegEx)](#string-matching-and-regular-expression-regex) - [RegEx case 1: match Twitter username from tweets](#regex-case-1-match-twitter-username-from-tweets) - [RegEx case 2: match telephone numbers in a piece of text](#regex-case-2-match-telephone-numbers-in-a-piece-of-text) - [Bonus: Text substitution](#bonus-text-substitution) - [RegEx in shell](#regex-in-shell) - [Word frequency](#word-frequency) - [Use dict to count the words frequency](#use-dict-to-count-the-words-frequency) - [Use pandas.series.value\_counts()](#use-pandas-series-value-counts) - [Stopwords](#stopwords) - [Set stopwords](#set-stopwords) - [Remove stopwords](#remove-stopwords) - [Visualize word frequency](#visualize-word-frequency) - [with bar chart](#with-bar-chart) - [wordcloud with matplotlib](#wordcloud-with-matplotlib) - [wordcloud with pyecharts](#wordcloud-with-pyecharts) - [Word segmentation](#word-segmentation) - [jieba for Chinese](#jieba-for-chinese) - [How to add new terms to the wordseg dictionary](#how-to-add-new-terms-to-the-wordseg-dictionary) - [How to adjust term weight in the wordseg dictionary](#how-to-adjust-term-weight-in-the-wordseg-dictionary) - [Bonus: TF.IDF](#bonus-tfidf) - [Bonus: Topic model](#bonus-topic-model) - [Bonus: Sentiment analysis](#bonus-sentiment-analysis) - [Bonus: word2vec](#bonus-word2vec) - [Further readings](#further-readings)

## Text processing

### String functions recap

Outline:

- split
- replace
- join
- format
- find
- slicing ( : )

### Case 1: Get full URL from HTML A tag's href attribute

To Scrape Initiumlab articles' urls, which we used as an example in [chapter 7](#)

- Using `split + slice + format`

When scraping certain elements in webpage, the elements sometimes are folded or shorted, we need to split to different parts, use list slicing to get the part we want and re-format the strings we want. For example:

```
▼<h1 class="post_title" itemprop="name headline">
 ▶<a class="post_title-link" href="../../../blog/20160730-mediawiki-wiki-
 knowledge-management-system/" itemprop="url">... == $0
```

the article url we get is `"../../../blog/20160730-mediawiki-wiki-knowledge-management-system/"`, but what we expect is this `"http://initiumlab.com/blog/20160730-mediawiki-wiki-knowledge-management-system/"`. We can format the new string by the following method.

```
s = "../../../blog/20160730-mediawiki-wiki-knowledge-management-system/"
```

```

s = s.split('/blog')
#s
s1 = s[-1]
#s1
s2='{0}{1}'.format('http://initiumlab.com/blog',s1)
#s2

```

## Case 2: Substring matching and substitution

Check out certain words in a string, replace certain words and separate the words with some notations.

- Using `find + replace + join`

In certain occasion, we need to check out whether there is one word in the string, and need to replace the word.

```

s = "this is string example....this is the string we will test, is it"
s.find('string')
#output: 8, the position of the first characters
s.find('python')
#output: -1, it will return -1 if there is no this word in the string
s.replace("is", "was") #replace all
#output: 'thwas was string example....thwas was the string we will test, was it'
s.replace("is", "was", 3) #replace first 3
#output: 'thwas was string example....thwas is the string we will test, is it'
s2 = s.split(' ')
'|'.join(s2) #you can add different things in the string
#output: 'this|is|string|example....this|is|the|string|we|will|test,|is|it'

```

## Case 3: Most frequent names in tweets

Apply a function onto every element of the dataframe - Check out the most frequent names in the tweets text

- Using `str.lower() + in operator + pandas.apply()`

Exercise with the Tweeter Troll Data, you can download [here](#).

1. Check out times that certain name appears in the dataset.

```

#import and check out the dataset
import pandas as pd
df = pd.read_csv('regular_reader_tweets.csv')
#len(df)
#df.sample(10)

#check certain user name in the text
def check_name(x):
 return 'ten_gop' in str(x).lower()
df['text'].apply(check_name).value_counts()
#return results true only
#df['text'].apply[check_names].value_counts()[True]

```

Output:

```

False 202991
True 491
Name: text, dtype: int64

```

1. Check out the most common names in the tweet text.

```

filter out all the user and reset to dataframe
s_user = df['user_key'].value_counts()

```

```

df_users = s_user.reset_index()
#df_users

count_retweeted_number of all the users with apply function
def count_retweeted_number(name):
 def check_name(x):
 return name in str(x).lower()
 return df['text'].apply(check_name).value_counts().get(True, 0)

df_users['count'] = df_users['index'].apply(count_retweeted_number)
df_users.sort_values(by='count', ascending=False)

```

Output:

	index	user_key	count
14	ten_gop	3194	491
2	giselleevns	6652	479
76	chrixmorgan	481	394
88	danageezus	394	381
59	jenn_abrams	857	314
83	worldofhashtags	426	261
4	thefoundingson	3663	211
82	politweecs	441	166
130	dominicvalent	156	161
96	gloed_up	327	150

## encode and decode

- UTF-8
- GBK

UTF-8 is the most widely used implementation of Unicode on the Internet, while GBK is mainly used for coding Chinese character.

When scraping the webpage, the results of `r.text` & `r.content` is different. Let's get straight of their relationships.

```

url = 'http://www.jour.hkbu.edu.hk/faculty/'
r = requests.get(url)
r.text

```

This is the results of `r.text` its a string.

```

r.text
'<!DOCTYPE html><html lang="en-US" xmlns:og="http://opengraphprotocol.org/schema/" xmlns:fb="http://www.facebook.com/2008/fbml"><head><meta charset="UTF-8"><meta name="viewport" content="width=device-width, initial-scale=1.0"><title>Faculty | HKBU | Department of Journalism</title><meta name="description" content="Hong Kong Baptist University - Department of Journalism was the first of its kind in Hong Kong when it was founded in 1968." /><meta property="og:title" content="Faculty | HKBU | Department of Journalism" /><meta property="og:description" content="HKBU Journalism was the first of its kind in Hong Kong when it was founded in 1968." /><meta property="og:type" content="website" />
```

```

r.content

```

This is the results of `r.content` its a byte.

```

r.content
b'<!DOCTYPE html><html lang="en-US" xmlns:og="http://opengraphprotocol.org/schema/" xmlns:fb="http://www.facebook.com/2008/fbml"><head><meta charset="UTF-8"><meta name="viewport" content="width=device-width, initial-scale=1.0"><title>Faculty | HKBU | Department of Journalism</title><meta name="description" content="Hong Kong Baptist University - Department of Journalism was the first of its kind in Hong Kong when it was founded in 1968." /><meta property="og:title" content="Faculty | HKBU | Department of Journalism" /><meta property="og:description" content="HKBU Journalism was the first of its kind in Hong Kong when it was founded in 1968." />
```

The conversion between string and byte is as follows:

- byte to string: `r.content.decode()` . Pass the decode method like `gbk` into the bracket.
  - string to byte: `r.text.encode()` . Pass the encode method like `utf-8` into the bracket.

Why we need to convert between two types?

When scraping one webpage, we need to get the `string` type so that we can use `BeautifulSoup` to parse the string and extract the value we want. Therefore, usually, we just use `r.text` is enough.

But if the website uses other encoding methods than `utf-8`, like `gbk`, we need to decode the content first. In such circumstance, using `r.content.decode()` to get the string. Usually, we can get the website encoding method in their `html meta tag`.

Another method to solve the encoding problem is making a statement at the beginning like the following.

```
r = requests.get('http://www.jour.hkbu.edu.hk/faculty/')
r.encoding = 'utf-8' #using this line, change the encoding method corresponding to their webpage encoding method

data = BeautifulSoup(r.text,"html.parser")
```

Case1: You can refer [Chapter 8 - encoding](#) for another example.

Case2 : scraping Chinese websites like 电影天堂 .

```
url = 'https://www.dytt8.net/'
r = requests.get(url)
html_str = r.text
```

You will find the results is messy because the website is using `gb2312` method to encode the html, which is kind of `GBK` methods.

```

<!doctype html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" class="gr_dytt8_net">
 <head>
 ... <meta http-equiv="Content-Type" content="text/html; charset=gb2312"> == $0
 <title>电影天堂_免费电影_迅雷电影下载</title>
 <meta content="免费电影下载,电影下载,最新电影" name="keywords">
 <meta content="最好的迅雷电影下载网, 分享最新电影, 高清电影、综艺、动漫、电视剧等下载!" name="description">
 <!-- Start: injected by AdGuard -->
 ▶<style type="text/css" nonce="A207FC5A2C634B56A8E8BFDA4837627">...</style>
 ▶<style type="text/css" nonce="A207FC5A2C634B56A8E8BFDA4837627">...</style>
 <!-- End: injected by AdGuard -->
 <link href="/css/dytt8.css" rel="stylesheet" type="text/css">
 <base target="_blank">

```

Therefore, we need to firstly decode the content.

```

html = r.content.decode('gbk')
html

```

```

'<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.
dtd">\r\n<html xmlns="http://www.w3.org/1999/xhtml">\r\n<head>\r\n<META http-equiv="Content-Type" content="text/html; c
harset=gb2312">\r\n<title>电影天堂_免费电影_迅雷电影下载</title>\r\n<META content="免费电影下载,电影下载,最新电影" name="keywor
ds">\r\n<meta content="最好的迅雷电影下载网, 分享最新电影, 高清电影、综艺、动漫、电视剧等下载!" name="description">\r\n<link href
="/css/dytt8.css" rel="stylesheet" type="text/css" /><base target="_blank">\r\n <script language="javascript">kstatus
(); function kstatus(){self.status="喜欢本站请使用 Ctrl+D 进行收藏,记得分享给您的朋友哦, TA会感谢您, 谢谢支持! "; setTimeout("k
status()",0); } function a() { alert("亲,请使用 Ctrl+D 进行添加收藏!"); }</script>\r\n</head><body>\r\n<div id="header">
<div class="contain"><h4></h4>\r\n<t><t><div id="headerright"><div id="about" align="right">\r
n <SCRIPT src="/js1/760h.js"></SCRIPT>\r\n<t><t><t><div id="menu"><div class="contain">\r\n<t><t><t>\r
n最新影片\r
n国内电影\r
n欧美电影\r
n其它电影\r
n华语电视\r
n欧美电视\r
n最新综艺\r
n旧版综艺\r
n动漫资源\r
n旧版游戏\r
n游戏下载\r
n高分经典
\r
n收藏本站\r
nAPP下载\r
n\r
n<t><t><t><div>\r
n<t><t><t>
```

After decoding, the Chinese characters can display appropriately. And when writing data into csv, you can use a more widely used method `utf-8` to encode it.

```

with open('dy.csv', 'a', newline='', encoding='utf-8') as f:
 writer = csv.writer(f)
 ...

```

## String matching and Regular Expression (RegEx)

The `.find()` and `in` operator on `str` has limited matching capability. They can only perform precise matching.

What if we want to do fuzzy matching? Common situations:

- Find all the user names from the Twitter message. i.e. those look like `@xxxx`, i.e. start with `@` and end with `(blank)`.
- Find all the phone numbers from a paragraph of texts.

RegEx can solve those problems in a very concise way. We show you the power of RegEx in following subsections.

Once you decide to move further, the [official doc on re](#) is a good source for you to further study RegEx.

### RegEx case 1: match Twitter username from tweets

Here's the solution for question 1.

```
>>> tweet = '@tom @jacky and @will, please come to my office at 10am.'
>>> import re
>>> pattern = re.compile(r'@[a-zA-Z0-9]+') # pattern of the Twitter username
>>> pattern.findall(tweet)
['@tom', '@jacky', '@will']
```

The key of learning RegEx is to learn the pattern string, i.e. `@[a-zA-Z0-9]+` in the above example. Let's decode this pattern as follows:

- `@` - matches a single `@` character.
- `[]` - is a collection notation, matching the current position to any valid character in this collection.
  - In our example, the collection is composed of
    - All characters from `a` to `z`, and from `A` to `Z`, and from `0` to `9`.
    - The hyphen `-` here is a notation to specify a range of characters.
- `+` - is a repetition number, meaning matching one or more characters using the preceding pattern character
  - The preceding pattern character is a collection, i.e. `[a-zA-Z0-9]`.

To sum it up, the above RegEx pattern can match all any substring that starts with a `@` character and then followed by one or more alphanumeric characters.

## RegEx case 2: match telephone numbers in a piece of text

Learning RegEx is like learning a new language. It is very easy to match the pattern you want. However, it may be difficult to exclude things that you don't want to match. Let's give a demo using question 2. The first trial would be:

```
>>> introduction = 'Student usually can enrol up to 16 courses in 1 semester. If you want to enrol in more courses, please contact Ms A at 34119999 or Mr B 34119998'
>>> pattern = re.compile(r'\d+')
>>> pattern.findall(introduction)
['16', '1', '34119999', '34119998']
```

In this pattern, `\d` means the collection of numbers, i.e. `[0-9]`. The `+` specifies a repetition count of one or multiple. You can see that all the numeric substrings are extracted. However, not all of them are telephone numbers. Given a bit domain knowledge, we know that the phone numbers in Hong Kong have 8 digits. So we can adjust our pattern to use a repetition number of `{8}`:

```
>>> pattern = re.compile(r'\d{8}')
>>> pattern.findall(introduction)
['34119999', '34119998']
```

The pattern works at present, but does not work when the text includes other 8-digit non telephone number substrings:

```
>>> introduction = 'Student usually can enrol up to 16 courses in 1 semester. If you want to enrol in more courses, please contact Ms A at 34119999 or Mr B 34119998. Students who enrol in extra courses will be charged 12345678 dollar per course.'
>>> pattern = re.compile(r'\d{8}')
>>> pattern.findall(introduction)
['34119999', '34119998', '12345678']
```

The tuition fee is also match because our pattern is not precise enough. In order to fix this problem, we need to further apply our domain knowledge. We know that HKBU's telephone numbers start with `3411`. So we can revise our pattern and get following result:

```
>>> pattern = re.compile(r'3411\d{4}')
```

```
>>> pattern.findall(introduction)
['34119999', '34119998']
```

The pattern is to match any 8 character string that starts with "3411" and ends with any four digits. The curious readers may ask: what if the tuition fee is "34113411 dollar"? As you guess, the above pattern fails. How to refine? The short answer is that there is no ultimate way. As long as "34113411" itself is a valid telephone number, we can hardly exclude it from our matching by just looking at itself.

This issue is quite common text processing, or further more **Natural Language Processing (NLP)**. The token itself is not enough for us to understand its meaning. One similar example is: "Apple is good". We are not sure if "the Apple" that produces iPhone is good; or apple, as a fruit, is good. We human being needs context in order to understand this sentence. So is computer. This discussion is beyond our curriculum. Interested readers can search for "NLP".

Back to our focus on basic pattern matching, we can conclude that it is easy to match the things one want using RegEx, but rather difficult to fully exclude other unwanted stuffs. When building your RegEx solution, you usually starts with a collection of **cases**. You keep on adding new cases to the collection and test them with your RegEx pattern. You may need to refine multiple times in order to find the one that works well on your dataset.

## Bonus: Text substitution

RegEx can let you substitute some matching parts. This is very similar to the `str.replace` function but does more than that. It even allows one to interpolate variables using values from matching part.

Suppose the university decides "3411" is a bad prefix and "8888" sounds good. How do we change all the numbers? Checkout following solution:

```
>>> pattern = re.compile(r'3411(\d{4})')
>>> re.sub(pattern, r'+852 8888-\1', introduction)
'Student usually can enrol up to 16 courses in 1 semester. If you want to enrol in more courses, please contact
Ms A at +852 8888-9999 or Mr B +852 8888-9998. Students who enrol in extra courses will be charged 12345678 do
llar per course.'
```

Besides changing the telephone prefix, we also added the country code and a hyphen between university prefix and the phone number. Note how this happens:

- `( )` is called a "group" in RegEx. The brackets do not match any character. However, patterns inside a pair of brackets are of the same group. The match substring will be put in the same group for further reference.
- `\1` means the first group. There could be multiple groups matched by one pattern. You can use `\n` notation to refer to the `n`-th group. `\0` means the entire string.

## RegEx in shell

Following commands can be used to perform RegEx operation. Those commands sometimes can make your work more efficient.

- `grep`
- `egrep`
- `fgrep`

## Word frequency

Word frequency refers to the number of times a list of given word appears in the file, which gives you a quick overview of the top words in one text data and help find the news point for further analysis.

## Use dict to count the words frequency

```
words_list = meaningful_words #the list of words you have
dict_words_frequency={}
for n in words_list:
 dict_words_frequency[n]=words_list.count(n)
#dict_words_frequency
```

## Use pandas.Series.value\_counts()

Use the [assignment 0](#) as an example:

```
import os
import pandas as pd

def read_txt(path): #read files and get content
 all_text = []
 for file in os.listdir(path):
 f=open(file, "r+",encoding="utf8",errors="ignore")
 contents= f.read()
 all_text.append(contents)
 all_words = "".join(all_text)
 return all_words

words = read_txt("text/") #pass your own file path that include list of .txt
words = words.split()
word_count = pd.Series(words).value_counts().sort_values(ascending=False)[0:15]
word_count
```

Output:

the	327
to	187
of	149
and	132
a	124
in	99
that	68
is	58
as	53
on	52
China	50
with	49
US	47
trade	47
Chinese	43

You can find that the above word frequency list is not so good because there are many meaningless words, like `the`, `to` ... Those are generally we called `stopwords`.

## Stopwords

Stop words are words which are filtered out before or after processing of natural language data (text). *From [wiki](#)*

The stopwords may change when handling different text analysis cases. We can get stopwords from the open source channel or customize your own stopwords.

## Set stopwords

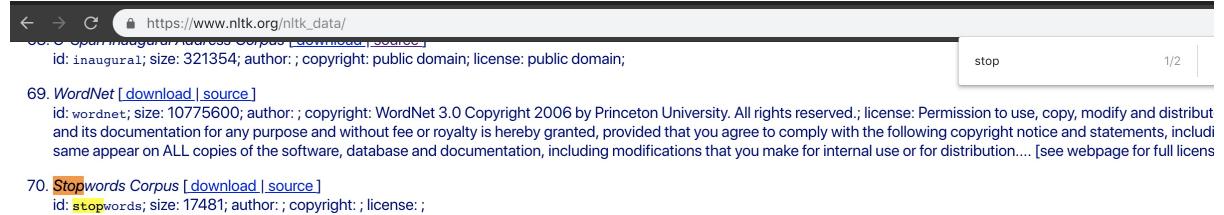
1. You can download the `stopwords.txt` from the internet and load when you used, [example](#).

```
filepath = './stopwords.txt'
stopwords = [line.strip() for line in open(filepath, 'r', encoding='utf-8').readlines()]
```

### 1. import stopwords from `nltk` library

```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
stopwords = stopwords.words('english')
```

**Note:** According to our helpers feedback, Windows users can successfully set stopwords with this method, for Mac users, you need to visit their [website](#), search `stopwords` and download.



Then put the language txt file in the current folder where your Jupyter notebook are.

### 1. import stopwords from `pypi`

For installation and documentation, you can refer [here](#), they provide a diverse languages of stopwords.

### 1. Customize your own stopwords

You can add new stopwords by the case need.

```
stopwords = ['a', 'b'] #the original stopwords list
newstopwords = ['stopword1', 'stopword2']
stopwords.extend(newstopwords)
```

## Remove stopwords

Move stopwords is easy, you just loop them to determine whether the words are in the stopwords, if true, then remove them. Expect the stopwords, we need also handle some punctuation and letter case. Following the above `assignment 1 example`:

```
import os
import pandas as pd

def read_txt(path): #read files and get content
 all_text = []
 for file in os.listdir(path):
 f=open(file, "r")
 contents= f.read()
 all_text.append(contents)
 all_words = ''.join(all_text)
 #remove punctuation
 for ch in '\s+\.\!\/_,\$%^*(+\"\\')]+|[+--()?:\[\]\\"":
 words = all_words.replace(ch, " ")
 return words

def stopwordslist(filepath): #set stopwords
 stopwords = [line.strip() for line in open(filepath, 'r').readlines()]
 return stopwords
```

```

def remove_stopwords(words): #remove stopwords
 processed_word_list = []
 for word in words:
 word = word.lower() # in case they are not all lower cased
 if word not in stopwords:
 processed_word_list.append(word)
 return processed_word_list

words = read_txt("text/") #pass your own file path that include list of .txt
words = words.split()
stopwords = stopwordslist('./stopwords_eng.txt')
stopwords = set(stopwords)
processed_word_list = remove_stopwords(words)
word_count = pd.Series(processed_word_list).value_counts().sort_values(ascending=False)[0:15]

```

After remove stopwords, you can see that the word frequency list is more meaningful, if there are still some stopwords, you can add them into your `stopword.txt` and run the codes again.

china	69
trade	52
chinese	43
trump	32
war	25
beijing	20
u.s.	17
tariffs	16
america	15
american	14
president	14
global	14
economic	11
administration	11
foreign	10

**Note:** If arising error when import the data like the following:

```
'utf-8' codec can't decode byte 0x80 in position 3131
```

You can refer [here](#) for solutions.

## Visualize word frequency

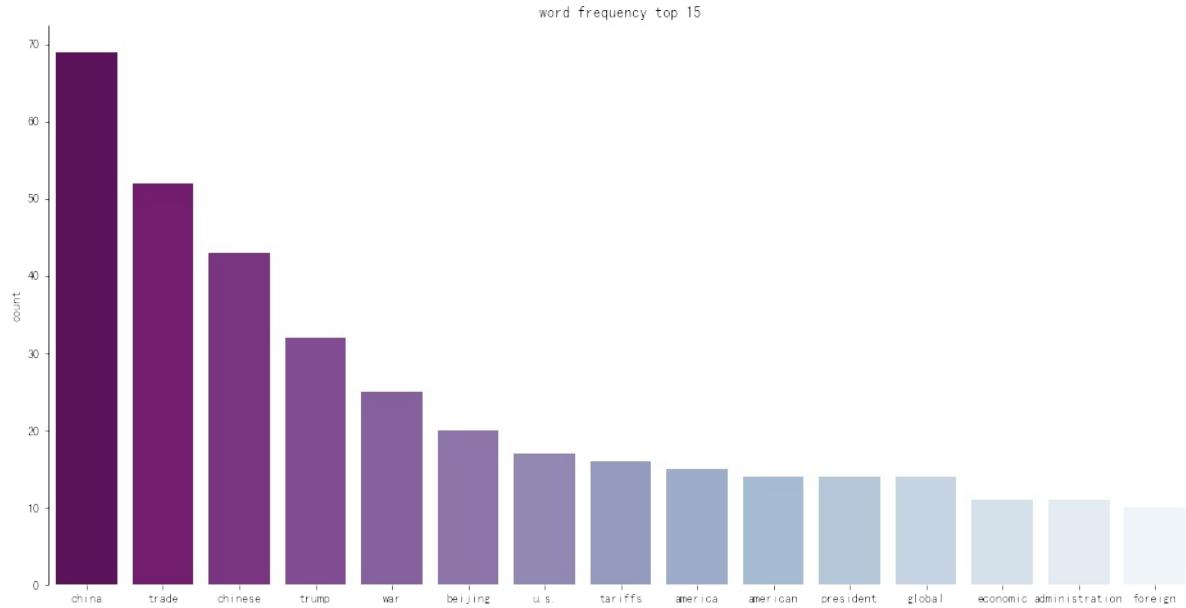
### with bar chart

```

#you can use other visualization library
import seaborn as sns
import matplotlib.pyplot as plt
def word_count(processed_word_list):
 word_count = pd.Series(processed_word_list).value_counts().sort_values(ascending=False)[0:15]
 fig = plt.figure(figsize=(16,8))
 x = word_count.index.tolist()
 y = word_count.values.tolist()
 sns.barplot(x, y, palette="BuPu_r")
 plt.title('word frequency top 15')
 plt.ylabel('count')
 sns.despine(bottom=True)
 #plt.savefig('./word_count_bar.png',dpi=400)
 plt.show()

word_count(processed_word_list)

```



## wordcloud with matplotlib

Tag cloud is widely used, for aesthetics purpose. You can have a more vivid picture about the top frequency words.

```
from PIL import Image
import wordcloud
import numpy as np

def tag_cloud(text):
 mask = np.array(Image.open('newspaper.png')) #set mask, you can change to the picture you like, but it must
 have a high color contrast
 wc = wordcloud.WordCloud(mode='RGBA', background_color='white', max_words=2000, stopwords=stopwords, max_font_size=300, random_state=42, mask=mask)
 wc.generate_from_text(' '.join(text))
 plt.figure(figsize=(12,12))
 plt.imshow(wc, interpolation='bilinear')
 plt.axis("off")
 plt.title('word frequency top 15 tag cloud', loc='Center', fontsize=20)
 plt.show()
 return plt.show()

tag_cloud(words)
```

## word frequency top 15 tag cloud



## wordcloud with pyecharts

One can also plot interactive tag cloud with pyecharts, one thing is that its very simple, you only need to get the clean words with its frequency . Another thing is you can save the efforts to set the font and environment to support displaying Chinese characters with matplotlib. Following is a example of wordloud with pyecharts, you can find more in their [documentation](#).

```
from pyecharts import WordCloud
words = ['特朗普', '关税', '美国', '中国', '加拿大', '表示', '贸易', '征收', '产品', '上周五', '亿美元', '征税', '威胁', '政府', '争端', '准备', '谈判', '进行', '商品', '美国政府']
value = [19, 17, 17, 17, 13, 8, 8, 7, 6, 5, 5, 5, 5, 5, 4, 4, 4, 4, 4, 4, 3]

wordcloud = WordCloud(width=1300, height=620)
wordcloud.add("", word, value, word_size_range=[20, 100])
wordcloud.render('trade-war-wordcloud.html')

from IPython.display import IFrame
IFrame('trade-war-wordcloud.html', width=700, height=620)
```



尚

## Word segmentation

### jieba for Chinese

For English words, it's easy to segment words, just by the blank space between the words. But for Chinese words, it's more complicated, because there is no blank space in Chinese sentence and the combination of the characters in one sentences may diverse too. In order to handle this situation, `jieba` is the most useful libraries we can get out here, which is a Chinese specialized word segmentation module.

Install and import:

```
!pip3 install jieba
import jieba
```

Basic usage:

```
s = '我在漫会大学学python'
jieba.cut(s)
#output: <generator object Tokenizer.cut at 0x10f3a3048>
list(jieba.cut(s)) #turn it into list
#[我', '在', '漫会', '大学', '学', 'python']
```

Case1: Cut an article, you can download the article [here](#).

```
import jieba
text = open('trade-wars-zh.txt', "r").read()
#cut words
words = jieba.cut(text)
#set stopwords
filepath = 'stopwords.txt'
stopwords = [line.strip() for line in open(filepath, 'r', encoding='utf-8').readlines()]
#remove stopwords
processed_word_list = []
for word in words:
 if word not in stopwords and len(word)>1: #remove single word
 processed_word_list.append(word)

#processed_word_list
```

## How to add new terms to the wordseg dictionary

Example 1: Customize your own words dict

For basic needs, using `suggest_freq(segment, tune=True)` and `add_word(word)` are recommended.

```
print('/'.join(jieba.cut('特朗普已经准备好对所有进口美国的中国商品征收关税'))
#中国/已/表示/计划/对/美国/的/任何/贸易壁垒/进行/报复/。
jieba.suggest_freq(['贸易', '壁垒'], True)
print('/'.join(jieba.cut('中国已表示计划对美国的任何贸易壁垒进行报复。'))))
#中国/已/表示/计划/对/美国/的/任何/贸易/壁垒/进行/报复/。
jieba.add_word('贸易壁垒')
print('/'.join(jieba.cut('中国已表示计划对美国的任何贸易壁垒进行报复。'))))
#中国/已/表示/计划/对/美国/的/任何/贸易壁垒/进行/报复/。
```

Example 2: Ambiguous word segmentation

### 1. 如果 & 果汁

```
s1 = '狮子山的山泉水如果汁一般好喝'
'/.join(jieba.cut(s1))
#'狮子山/的/山泉水/如果/汁/一般/好喝'
jieba.suggest_freq(['如', '果'], True)
'/.join(jieba.cut(s1))
#'狮子山/的/山泉水/如/果汁/一般/好喝'
```

### 1. 乒乓球拍 卖完了& 乒乓球 拍卖完了

```
s2 = '乒乓球拍卖完了'
'/.join(jieba.cut(s2))
#'乒乓球/拍卖/完/了'
jieba.add_word('乒乓球拍')
jieba.suggest_freq(['拍', '卖'], True)
#'乒乓球拍/卖完/了'
```

For a complicated cases, you can refer [here](#) to customize your own words dict.

## How to adjust term weight in the wordseg dictionary

### Bonus: TF.IDF

- `TF` - Term Frequency
- `IDF` - Inverse Document Frequency
- `TFIDF = TF * IDF`

TFIDF is a measure of (a term's importance to a document) in (a collection of documents), called "corpus". We put the previous sentence in brackets so that it is easier to read. The rationale is very straight forward:

- TF -- importance to a specific document -- the more one term appears in one document, the more important it is to the document.
- IDF -- importance in a collection of documents -- if a term appears too frequently in all documents, e.g. stop words, it does not carry much importance to the current document.

[Read more](#)

### Bonus: Topic model

Topic model is a typical example of machine learning: discover meaningful lower dimension space out of higher dimension observations. The higher dimension space is called data space. The lower dimension space is called latent space. The basic assumption is that, despite the complexity of text/ human language, the intrinsic structure is simple. The texts we observe are just statistical variables generated from the intrinsic structure.

Consider two courses about data journalism:

1. The data analysis course, e.g. [current course](#)
2. The data visualization course, e.g. <http://datavis.studio>

We know the "topic" of the two courses are different. How can we tell? If we check out the word frequency of the two course, we may find that:

1. Frequent terms: Data, scraper, web, Python, jupyter, pandas, numpy, matplotlib, ...
2. Frequent terms: Data, Javascript, CSS, HTML, web, responsive, bootstrap, echart, ...

By looking at the two different lists, we can tell they are of different topics. Computers can also recognise topics in a similar way. In a usual topic modeling procedure, we start with a matrix composed of "document vectors". The vector has a coordinate system using all the potential terms, so every element in the vector represents an intensity of this term in the document. A "topic vector" has the same shape of a "document vector" -- a collection of terms with different weights. Some terms may be stronger indicator of certain topic, like "Python" and "Javascript" in above example. Some other terms may be a weaker indicator, like "web" in above example, where one course emphasize more on "web scraping" and another course emphasize more on "responsive web". In the technical language, "topic vector" is a (mostly "linear") combination of "document vectors". The number of topics is much less than the number of documents, which can be told from the original rationale:

- We have many documents but only a few topics
- The intrinsic structure (number of docs) is much simpler than the observations (documents)

Here is a [tutorial](#) of topic mining using `nltk` and `gensim`. The algorithm used is called LDA.

## Bonus: Sentiment analysis

The task of sentiment analysis is to get polarity and subjectivity from a piece of text. Polarity can be positively mooded or negatively mooded. Subjectivity can be subjective or objective. It finds good applications in media monitoring. When a crisis emerges, you may want to know how the public reacts to the incident. With massive data from the social network and real-time sentiment analysis, one can devise a better PR strategy.

The procedure of sentiment analysis usually starts with a collection of positive terms and a collection of negative terms. If one piece of text contains more positive terms, it is likely to be an overall positive statement; Vice versa. How to get the collection of positive or negative terms? We usually start with human labels on some training texts. After human judges the sentiment of the training texts, we throw them to the computer. The computer builds the term collection using the collection that if one term appears frequently in a positive statements, that term is likely to be positive; Vice versa.

The above is just an intuition of sentiment analysis. The real work involves more sophisticated statistical models. You may not be able to fully understand them but following are some pointers for you to get working codes:

- Construct classifier using `sklearn`. Checkout the [tutorial](#).
- Online API like [text-processing](#).
- `TextBlob` is also useful and applied in [S2018 group 2's work](#).

## Bonus: word2vec

word2vec is a set of algorithms to perform "word embedding". It is similar to "topic modeling" in terms of the structure:

- Topic modeling - A document is represented by a (linear) combination of topics; A topic is represented by a (linear) combination of terms.
- Word embedding - A document is represented by a variable length sequence of words; A word is represented by a lower-dimension vectors; the element of word vectors may not have physical interpretation.

The upside of word embedding is to enable vector arithmetics like `king-man+woman=queen`. Also, the word vector can be computed offline (pre-computation). That makes it easier to handle new documents.

Checkout [Wikipedia](#) for background information and [this blog](#) for a quick tutorial on LDA (topic model), word2vec and LDA2vec.

## Further readings

1. [Unicode, ASCII,UTF-8 and GBK](#)
- 

If you have any questions, or seek for help troubleshooting, please [create an issue here](#)

# Week 13: Datetime and Time Series

- [Week 13: Datetime and Time Series](#week-13-datetime-and-time-series) - [Objective](#objective) - [Datasets](#datasets) - [Datetime](#datetime) - [Create datetime object](#create-datetime-object) - [Convert from string to datetime](#convert-from-string-to-datetime) - [Parse ambiguous dates](#parse-ambiguous-dates) - [Parse incomplete times](#parse-incomplete-times) - [A failed parsing case](#a-failed-parsing-case) - [Convert from datetime to utctimestamp and vice versa](#convert-from-datetime-to-utctimestamp-and-vice-versa) - [What is a timestamp](#what-is-a-timestamp) - [Get a datetime from a timestamp:](#get-a-datetime-from-a-timestamp) - [Format a datetime object to string](#format-a-datetime-object-to-string) - [Get the present time](#get-the-present-time) - [Formatting with different style](#formatting-with-different-style) - [Arithmetics on datetime](#arithmetics-on-datetime) - [Compare two datetime object](#compare-two-datetime-object) - [Know timedelta object](#know-timedelta-object) - [Get difference between two datetime objects](#get-difference-between-two-datetime-objects) - [Add timedelta to a datetime object](#add-timedelta-to-a-datetime-object) - [Bonus: how to get a datetime object for the current time without microseconds](#bonus-how-to-get-a-datetime-object-for-the-current-time-without-microseconds) - [Bonus: Deal with different scales of time durations](#bonus-deal-with-different-scales-of-time-durations) - [Time Series Basics](#time-series-basics) - [Resample, aggregate and plot](#resample-aggregate-and-plot) - [Case: Twitter keywords variation by time](#case-twitter-keywords-variation-by-time) - [Sample](#sample) - [Resample](#resample) - [Bonus: explore resample](#bonus-explore-resample) - [aggregate](#aggregate) - [plot](#plot) - [Time Series Advanced Topics](#time-series-advanced-topics) - [Smoothing technique: Moving average](#smoothing-technique-moving-average) - [Bonus: Time Series forecasting models](#bonus-time-series-forecasting-models) - [References](#references)

## Objective

- Understand the principle of timestamp and datetime format
- Master basic computation on datetime values, including handling timezone conversion
- Understand periodical analysis (daily, weekly, monthly, seasonal, etc)

Modules:

- `datetime`
- `dtparser`
- `pandas`
  - basic visualisation using polyline `.plot()`
  - zoom in/ out: `.resample` , `.aggregate`
- `seaborn`

## Datasets

- NBC Russian Troll on Twitter dataset
- Twitter Data of the Donald & Ivanka Trump analysis -- reproduce the charts.
- Financial data usually comes in form of time series, e.g. [Yahoo Finance API](#).
- Bitcoin transactions are available via [blockchain.com API](#). A free [cryptocurrency API](#) is available to query the exchange ratio between two symbols.

## Datetime

The `datetime` module supplies classes for manipulating dates and times in the fields like time parsing, formatting or even arithmetic. You can read its document [here](#).

## Create datetime object

First we create a datetime object:

```
from datetime import datetime
dt = datetime(year=1993, month=10, day=4,
 hour=9, minute=8, second=7)
dt
```

Or you can use a positional arguments

```
from datetime import datetime
dt = datetime(1993, 10, 4, 9, 8, 7)
dt
```

Output:

```
datetime.datetime(1993, 10, 4, 9, 8, 7)
```

The result is a `datetime` object. A `datetime object` is a single object containing all the information from a time point.

## Convert from string to datetime

In many cases, we may need to standardize the format of date/time we scraped from the Internet into datetime objects for further application. We can use `parse` in `dateutil` library. See this case:

```
from dateutil.parser import parse
dt_1 = parse("Thu Sep 25 10:36:28 2018")
dt_2 = parse('19/May/2017 04:10:06')
dt_3 = parse('2018.2.3')
dt_4 = parse('June 12, 2018')
dt_1, dt_2, dt_3, dt_4
```

Output:

```
(datetime.datetime(2018, 9, 25, 10, 36, 28),
 datetime.datetime(2017, 5, 19, 4, 10, 6),
 datetime.datetime(2018, 2, 3, 0, 0),
 datetime.datetime(2018, 6, 12, 0, 0))
```

All the time strings in different formats have been transferred into datetime objects. The level of details depends on how explicit the information the raw data provided is.

## Parse ambiguous dates

In some cases, we may need to parse some ambiguous dates like `parse("10-09-2003")`. We need to give the parameter what the first figure represents:

```
from dateutil.parser import parse
dt_1 = parse("10-09-2003", dayfirst=True)
dt_2 = parse("10-09-03", yearfirst=True)
dt_1, dt_2
```

Output:

```
(datetime.datetime(2003, 9, 10, 0, 0), datetime.datetime(2010, 9, 3, 0, 0))
```

## Parse incomplete times

Many times on the Internet may not be as normative as `2018/12/22`. Instead, many of them are `12/22` or even `Thu 10:36:28`. We can define a default time.

```
from datetime import datetime
from dateutil.parser import parse
DEFAULT = datetime(2018, 11, 25)
dt_1 = parse("Thu Sep 10:36:28", default=DEFAULT)
dt_2 = parse("Thu 10:36:28", default=DEFAULT)
dt_3 = parse("12/25", default=DEFAULT)
dt_4 = parse("10:36", default=DEFAULT)
dt_1, dt_2, dt_3, dt_4
```

### Output

```
(datetime.datetime(2018, 9, 27, 10, 36, 28),
datetime.datetime(2018, 11, 29, 10, 36, 28),
datetime.datetime(2018, 12, 25, 0, 0),
datetime.datetime(2018, 11, 25, 10, 36))
```

- The parameter `default` here means how python autofill the time which miss some units. For example, `Thu Sep 10:36:28` lack the information 'in which year' and `12/25` lack 'on which time' and 'in which year'. Python will autofill them according to the corresponding part in `default`.

## A failed parsing case

However, The parsing will fail if we input a time against the regulations.

```
from dateutil.parser import parse
parse("2月15日 10:36:28")
```

This line will raise a `ValueError`:

```
ValueError: ('Unknown string format:', '2月15日 10:36:28')
```

- If we are going to parse `parse("2月15日 10:36:28")`, we need to convert it into a format without Chinese characters. Try to use `str.replace()` before parsing.

Generally, the `dateutil` module provides powerful extensions to the standard `datetime` module. You can check more parse examples [here](#)

## Convert from datetime to utctimestamp and vice versa

### What is a timestamp

The `timestamp` is the time in seconds since an *epoch* as a floating point number. The *epoch* is the point where the time starts, and is platform dependent. On Windows and most Unix systems, the epoch is January 1, 1970, 00:00:00 (UTC).

```
from datetime import datetime, timezone
dt = datetime(
```

```
1993, 10, 4, 9, 8, 7,
tzinfo=timezone.utc)
ts = dt.timestamp()
ts
```

Output:

```
749725687.0
#the output figure depends on your current time
```

## Get a datetime from a timestamp:

```
from datetime import datetime
datetime.utcfromtimestamp(ts)
```

Output:

```
datetime.datetime(1993, 10, 4, 9, 8, 8, 12345)
```

Bonus: [UTC](#) (abbreviated from *Coordinated Universal Time*) is the primary time standard by which the world regulates clocks and time.

## Format a datetime object to string

### Get the present time

```
from datetime import datetime
dt = datetime.now()
ds = dt.isoformat(timespec='seconds', sep=' ')
print(ds, type(ds))
```

Output:

```
2018-11-19 16:45:44 <class 'str'>
```

- Question: What is the type of `ds`? You can try to change its parameters in `.isoformat()` or remove it to see what will happen.
- You can also use `str(dt)` to transfer a datetime object into string.

### Formatting with different style

In some cases, we may need to convert a datetime into a specific format. Now we can use `strftime()`. Here is an example:

```
from datetime import datetime
dt = datetime(2018, 11, 20, 14, 0, 0)
print(dt.strftime('%I:%M%p %m/%d(%a),%Y'))
print(dt.strftime('%H:%M:%S %Y/%m/%d'))
```

Output:

```
02:00PM 11/20(Tue),2018
14:00:00 2018/11/20
```

In this case, `%H` and `%I` represent hour in 24-hour clock and 12-hour clock respectively. For 12-hour clock, we use `%p` to get AM/PM from the datetime object. You can click [here](#) to see what each parameter represents in `strftime()`.

## Arithmetics on datetime

### Compare two datetime object

One can perform boolean comparison on `datetime` objects:

```
from datetime import datetime
result_1 = datetime(2018, 6, 12, 0, 0) > datetime(2018, 2, 3, 0, 0)
result_2 = datetime(2018, 6, 12, 0, 0) == datetime(2018, 2, 3, 0, 0)
result_3 = datetime(2018, 6, 12, 0, 0) < datetime(2018, 2, 3, 0, 0)
print(result_1, result_2, result_3)
```

Output:

```
True False False
```

### Know timedelta object

A `timedelta` object represents a duration, the difference between two dates or times. We can build a `timedelta` object like this:

```
from datetime import timedelta
td = timedelta(days = 1)
td
```

Output:

```
datetime.timedelta(days=1)
```

The parameter `days` can be replaced with `seconds`, `microseconds`, `milliseconds`, `minutes`, `hours` and `weeks`. We can also combine them like `timedelta(weeks = 1, days = 2, hours = 12)`.

### Get difference between two datetime objects

To get the duration between two datetime objects, we can calculate like this:

```
from datetime import datetime
datetime(2018, 6, 12, 0, 0) - datetime(2018, 2, 3, 0, 0)
```

Output:

```
datetime.timedelta(days=129)
```

### Add timedelta to a datetime object

We can also do calculation between a datetime object and a `timedelta` object.e.g.what is the date 4 weeks later?

```
from datetime import datetime, timedelta
td_today = datetime(2018, 11, 19)
#td_today = datetime.date.today()
#td_today = datetime.now()
td = td_today + timedelta(weeks = 4)
str(td)
```

Output:

```
'2018-12-17 00:00:00'
```

In this case, you can use `datetime.date.today()` instead to get the real-time date.

## Bonus: how to get a datetime object for the current time without microseconds

You may have found that `datetime.now()` will return `datetime.datetime(2018, 11, 19, 16, 48, 33, 369859)`. In the former part, we use `dt.isoformat(timespec='seconds', sep=' ')` to omit the **microseconds**, but this method will convert the datetime object into a string. We can hold the current time as a datetime object for further calculation with these lines:

```
from datetime import datetime
dt = datetime.now()
dt_without_microseconds = datetime(dt.year, dt.month, dt.day, dt.hour, dt.minute, dt.second)
dt_without_microseconds
```

Output:

```
datetime.datetime(2018, 11, 19, 16, 41, 15)
```

## Bonus: Deal with different scales of time durations

After we have scape the strings referring to time from a [website](#), we may need to deal with a group of different scales of time durations:

```
from datetime import datetime, timedelta
time_list = ['30 Minutes ago', '12 Hours ago',
 '4 Days ago', '3 Weeks ago',
 '2 Month ago', '1 Year ago']
dt = datetime.now()
now = datetime(dt.year, dt.month, dt.day, dt.hour, dt.minute, dt.second)
print('The current time is {}'.format(now))
mylist=[]
for i in time_list:
 if i.lower().find('minute') != -1:
 post_time = now - timedelta(minutes = int(i.split()[0]))
 elif i.lower().find('hour') != -1:
 post_time = now - timedelta(hours = int(i.split()[0]))
 elif i.lower().find('day') != -1:
 post_time = now - timedelta(days = int(i.split()[0]))
 elif i.lower().find('week') != -1:
 post_time = now - timedelta(weeks = int(i.split()[0]))
 elif i.lower().find('month') != -1:
 post_time = now - timedelta(days = int(i.split()[0]) * 30)
 elif i.lower().find('year') != -1:
 post_time = now - timedelta(days = int(i.split()[0]) * 365)
 mylist.append(post_time)
output = 'The time {} is {}'.format(i, post_time)
```

```
print(output)
```

Now we get their precise time point. Output:

```
The current time is 2018-11-19 10:57:50.
The time 30 Minutes ago is 2018-11-19 10:27:50.
The time 12 Hours ago is 2018-11-18 22:57:50.
The time 4 Days ago is 2018-11-15 10:57:50.
The time 3 Weeks ago is 2018-10-29 10:57:50.
The time 2 Month ago is 2018-09-20 10:57:50.
The time 1 Year ago is 2017-11-19 10:57:50.
```

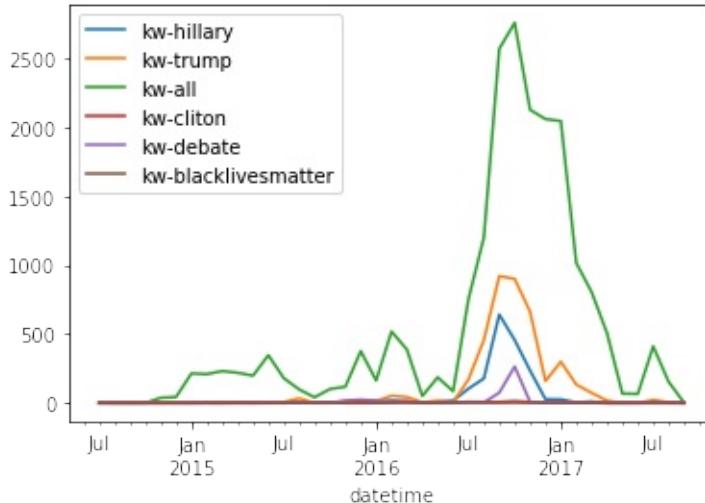
## Time Series Basics

### Resample, aggregate and plot

- Basic requirement: plot time series at different granularity, e.g. by hour, by day, by week, ... Articulate the findings on the polyline plot.
- Checkout [this notebook](#) for a concrete case of analysing term frequency changes over time in the Tweets.

The core codes are as follows:

```
df_kws = df.set_index('datetime').resample('1m').aggregate('sum')
df_kws.plot()
```



The key points of plotting time series using pandas:

- First you need to put `datetime` type of data onto index. This usually involves
  - `.apply` a function to [convert from string to datetime](#)
  - `.set_index` to move the `datetime` type from column to index. This is essential step to perform time series operation because later functions all refer to index for the datetime value.
- Use `.resample` to put the data points into different buckets. This is essentially a `.groupby` operation. Instead of working on categorical values like `.groupby`, `.resample` works on datetime ranges. One can specify a time length when performing resample operation, e.g. one week `1w` and two days `2d`.
- Use `.aggregate` to turn the bucket of data points into a single value. This is the same process like [groupby + aggregate](#) approach, but applied on datetime data types.

There is a small missing piece of the above Tweets keyword time series from current discussion. Besides handling the index, you also need to have numeric data on columns, e.g. `kw-hillary` as you can see from the chart. You can checkout [Most Common Names in Tweets](#) example to see how to encode tweet text into such numeric indicator variables.

## Case: Twitter keywords variation by time

A **time series** is a series of data points indexed (or listed or graphed) in time order. They are very frequently plotted via line charts and used in many fields like statistics, pattern recognition, mathematical finance, weather forecasting, earthquake prediction, astronomy and communications engineering. Check here for more information: [Time series - Wikipedia](#). Time series will become more important when we are dealing with the rather bigger datasets. See this case:

```
import pandas as pd
df = pd.read_csv('https://raw.githubusercontent.com/hupili/python-for-data-and-media-communication/master/text-analysis/regular_reader_tweets.csv')
print('The length of df is {}'.format(len(df)))
df.head()
```

Output:

```
The length of df is 203482
```

	<b>user_key</b>	<b>created_str</b>	<b>text</b>
203477	nojonathonno	11/1/2016 12:13	RT @AndreaChalupa: In intel circles, the story...
203478	judelambertusa	6/18/2015 1:04	RT @KansasCityDNews: Tonganoxie police: Middle...
203479	patriotblake	1/10/2017 18:50	RT @signsinyork: Getting the right #company lo...
203480	dailysandiego	11/20/2016 21:49	The Latest: Obama affirms continuity of ties w...
203481	willisbonnerr	12/19/2016 23:00	RT @futureguru100: U cant just Upload a CD onl...

There are more than 200 thousand lines in this dataframe. However, this is the very beginning and we can extract data by different time series from it.

## Sample

In early stage, we can use `sample()` to return a random sample of items from an axis of object. The sample procedure may lower the reliability but help us deal with large amount of data which are hard for making a census. One can make inferences or extrapolations from the sample to the population. See this step of sampling:

```
import pandas as pd
df = df.sample(frac=0.1)
print('After sampling, the length of df is {}'.format(len(df)))
df.head()
```

Output:

```
After sampling, the length of df is 20348
```

	<b>user_key</b>	<b>created_str</b>	<b>text</b>
51456	ten_gop	7/11/2016 4:10	Sheriff Clarke: 'Obama Is Like a Pyromaniac Wh...
122682	thefoundingson	1/14/2017 16:15	Manual for dummies\r\n#tcot #pjnet https://t.c...
183231	patriotblake	10/7/2016 8:16	RT @yournewswire: Snopes Caught Lying For Hill...
193656	ten_gop	10/2/2016 1:37	VIDEO: Loud Islamic call to prayer in Michigan...
13992	giselleevns	12/14/2016 15:29	RT @serradicassano: #GiftIdeasForPoliticians A...

We can find that there are 1/10 (because of `frac=0.1`) data have been randomly selected and the data has been disrupted the order. You can also learn more about the regulations of sampling in [pandas official document](#).

## Resample

In `pandas` library, `resample()` is a convenience method for frequency conversion and resampling of time series. Its object must have a index composed by datetime-like values like Datetime or Timedelta. Therefore, let's first utilise what we learned before to parse these twitts' post time, formatting them into machine recognizable ones:

```
from datetime import datetime
from dateutil import parser
import numpy
def parse_datetime(x):
 try:
 return parser.parse(x)
 except:
 return numpy.nan
df['datetime'] = df['created_str'].apply(parse_datetime)
```

Now we can use `resample('1W')` to know how many twitts emerged every week.

```
df.set_index('datetime').resample('1W').aggregate('count').tail()
```

Output:

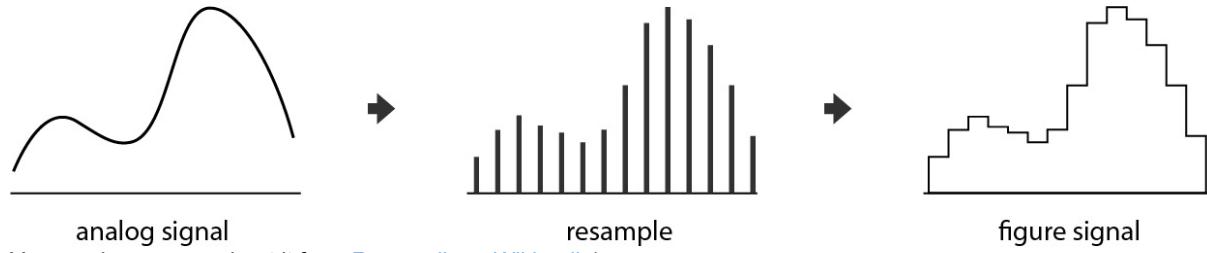
	<b>user_key</b>	<b>created_str</b>	<b>text</b>
	<b>datetime</b>		
2017-09-03	7	7	7
2017-09-10	1	1	1
2017-09-17	3	3	3
2017-09-24	0	0	0
2017-10-01	1	1	1

Notes:

- Setting the 'datetime' column as index is necessary, for `resample()` must have a index composed by datetime-like values.
- '`1W`' is an essential positional argument which means we collect twitts per 7-day period. You can also use the parameters like '`s`' (second), '`M`' (minute), '`M`' (month), '`SM`' (semi-month) and so forth. You can check [here](#) to read more instructions.
- `aggregate('count')` counts how many Tweets posted on a weekly level. We will introduce 'aggregate' in the next part.

## Bonus: explore resample

In statistics, **resampling** is method for drawing randomly with replacement from a set of data points, including exchanging labels on data points when performing significance tests or validating models by using random subsets. The resampling as a methodology has been widely used in the field of analogue signal processing or audio compression for many years. See its basic mode:



You can learn more about it from [Resampling - Wikipedia](#).

## aggregate

The **aggregate** is a process where the values of multiple rows are grouped together. It is aimed to form a single value of more significant meaning or measurement e.g. a sum, a max or a mean. See how it works in this case:

```
def has_hillary(t):
 return 'hillary' in str(t).lower()
def has_trump(t):
 return 'trump' in str(t).lower()
df['kw-hillary'] = df['text'].apply(has_hillary)
df['kw-trump'] = df['text'].apply(has_trump)
df.head()
```

Output:

	user_key	created_str	text	datetime
44413	evagreen69	12/29/2015 9:37	RT @realDonaldTrump: The voters the Republican...	2015-12-29 09:37:00
165333	cassiewelch	12/21/2016 16:26	Some cool kicks #SecondhandGifts https://t.co/...	2016-12-21 16:26:00
17515	lazylkstafford	12/3/2016 5:13	RT @Franklin_Graham: They're trying for a medi...	2016-12-03 05:13:00
113998	peterkistner	7/21/2016 11:20	@Gisela_Piltz würde Frau Merkel 4. Amtszeit sc...	2016-07-21 11:20:00
84809	patriotblake	2/15/2017 2:46	RT @youlivethrice: Shepard Smith bleeds libera...	2017-02-15 02:46:00

```
df.set_index('datetime').resample('1W').aggregate('sum').tail()
```

Output:

datetime	kw-hillary	kw-trump
2017-07-30	0.0	2.0
2017-08-06	0.0	3.0
2017-08-13	0.0	2.0
2017-08-20	0.0	0.0
2017-08-27	0.0	0.0
2017-09-03	0.0	0.0
2017-09-10	0.0	0.0
2017-09-17	0.0	0.0
2017-09-24	0.0	0.0
2017-10-01	0.0	0.0

The sum of each line have been figured. You can check [here](#) to learn more about what aggregate can do.

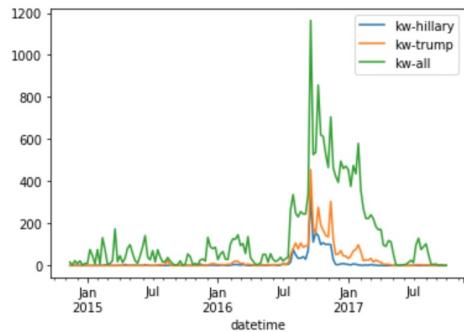
## plot

After resample and aggregate, we can use `plot()` to do the visualisation. Here is an example:

```
df['kw-all'] = df['text'].apply(lambda x: 1)
df.set_index('datetime').resample('1w').aggregate('sum').plot()
```

Output:

<matplotlib.axes.\_subplots.AxesSubplot at 0x111e61fd0>



You can check out more visual aid analysis in [this notebook](#). It is a concrete case of analysing term frequency changes over time in the Tweets.

## Time Series Advanced Topics

### Smoothing technique: Moving average

When analysing/ visualising time series, one most common issue is to deal with short period fluctuations. This is especially important in technical analysis of stock price. Stock price can fluctuate a lot in minutes but the fluctuation is less impactful when viewed in the larger time span. We need to smooth the time series curves in order to discover long term trend. `pandas` provides `DataFrame.rolling_mean` and `Series.rolling` to calculate "moving average" (The "MA-xx" curves you see in stock software). The moving average captures the momentum in the data and the crossing of two MAs of different length are usually used as indicators of buy/ sell signals. Checkout [this notebook](#) for more details.

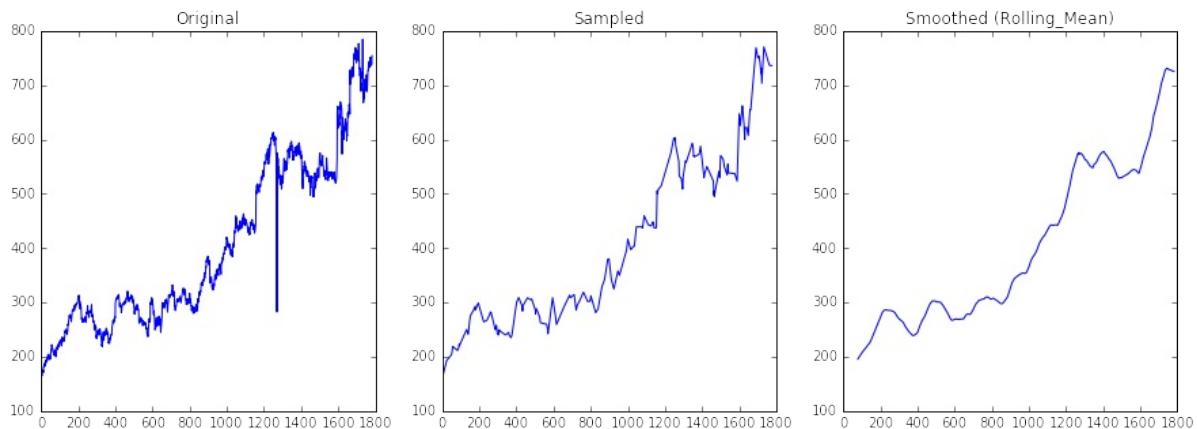
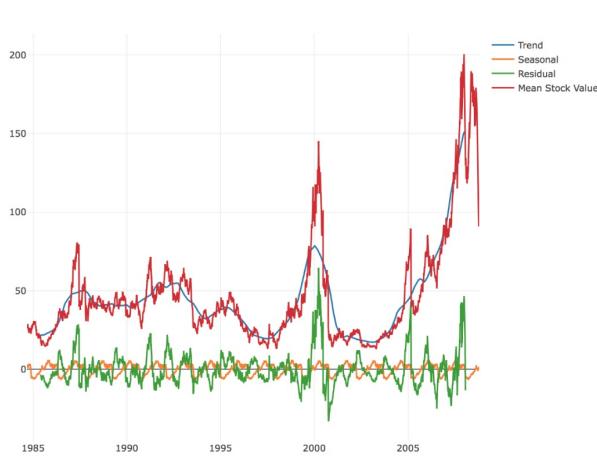


Image credit: Michael Galarnyk

### Bonus: Time Series forecasting models

A time series usually involves several components:

- Trend - the non-repeating movement in the data, e.g. increasing stock price
- Seasonal - repeating movement in the data, e.g. more sells before tax payment period.
- Noise/ residual - other movements that do not belong to the above, e.g. interruptions in the market/ move-the-market news.



```
from statsmodels.tsa.seasonal import seasonal_decompose
decomposition = seasonal_decompose(df.Mean, freq=365)
trace1 = go.Scatter(
 x = df.Date,y = decomposition.trend,
 name = 'Trend',mode='line'
)
trace2 = go.Scatter(
 x = df.Date,y = decomposition.seasonal,
 name = 'Seasonal',mode='line'
)
trace3 = go.Scatter(
 x = df.Date,y = decomposition.resid,
 name = 'Residual',mode='line'
)
trace4 = go.Scatter(
 x = df.Date,y = df.Mean,
 name = 'Mean Stock Value',mode='line'
)
data = [trace1,trace2,trace3,trace4]
plot(data)
```

Image credit: Jae Duk Seo

Next question is how to forecast a time series? Predictive analysis is not a requirement from this introductory course. Our main focus is on the descriptive part. Interested readers can checkout the following models from other literatures.

- AR
- MA
- ARMA
- ARIMA

Checkout [this tutorial](#) for an implementation of ARIMA using `pyramid-arima` (on [pypi](#)) and `statsmodels`.

[This tutorial](#) has a more detailed explanation of AR and MA and its decompositions.

Note that the above models are highly simplified presentation of the reality. It works resonably with one-way market like sales forecast, where vendor and consumer have clear roles. It does not work well in stock market price prediction, because the market participants play the counterpart of each other and their predictions affect their behaviour which further affect the market status.

## References

- timestamp usually come in unit of milliseconds (1/1000) of a second. [An example](#) to parse this timestamp format into `datetime` format.
- Past notes of `datetime` from spring 2018.
- Brockwell, Peter J., and Richard A. Davis. Introduction to Time Series and Forecasting. 2nd ed. Springer Texts in Statistics. New York: Springer, 2002.

## Week 14: Network data

- [Week 12: Network data](#week-12-network-data) - [Graph introduction](#graph-introduction) - [Network analysis with NetworkX](#network-analysis-with-networkx) - [Basic usage of networkx](#basic-usage-of-networkx) - [Common Network Analysis Routine via Les Misérables dataset](#common-network-analysis-routine-via-les-misérables-dataset) - [Graph visualization](#graph-visualization) - [Basic visualization](#basic-visualization) - [Adjust layout](#adjust-layout) - [Group the nodes with same color](#group-the-nodes-with-same-color) - [Measure Node and Edge Importance](#measure-node-and-edge-importance) - [Degree](#degree) - [Centrality Measures](#centrality-measures) - [Basic statistics of graph](#basic-statistics-of-graph) - [Degree distribution](#degree-distribution) - [Clustering coefficient](#clustering-coefficient) - [Structure of a graph](#structure-of-a-graph) - [Cliques](#cliques) - [Connected components](#connected-components) - [Community detection](#community-detection) - [Other Graph algorithms](#other-graph-algorithms) - [Shortest path](#shortest-path) - [Other Network Visualization Libraries and Tools](#other-network-visualization-libraries-and-tools) - [kumu.io](#kumuio) - [Google Fusion Table](#google-fusion-table) - [pyecharts](#pyecharts) - [Reference examples](#reference-examples)

### Graph introduction

Graph theory is the study of graphs, which are mathematical structures used to model pairwise relations between objects. A graph in this context is made up of nodes, which are connected by edges, arcs, or lines. A graph may be undirected, meaning that there is no distinction between the two vertices associated with each edge. (from [wiki](#))

There are different kind of graphs.

## Type of Graphs

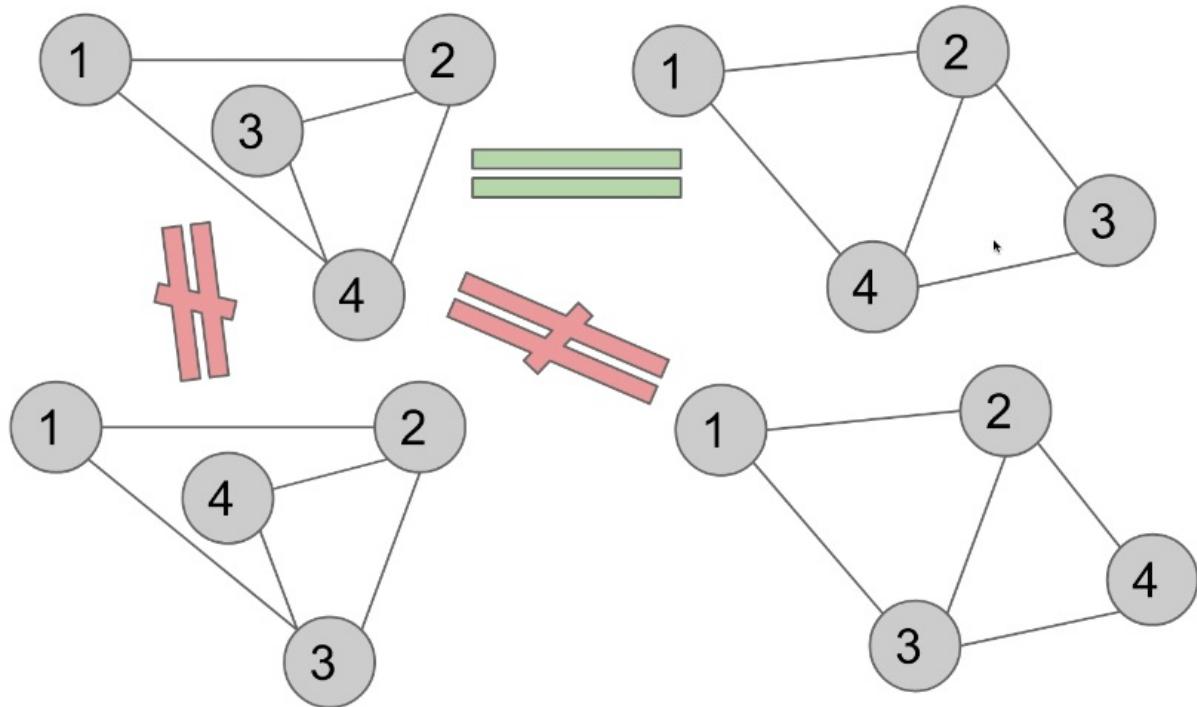
Undirected



Directed

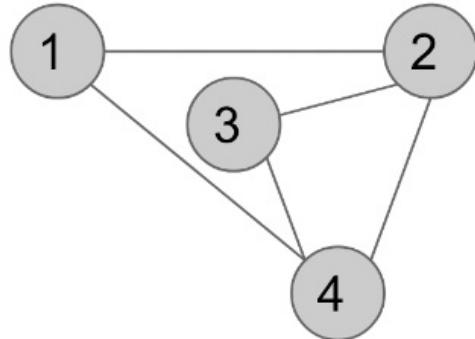


The criteria to judge whether one graph is the same as others is not the appearance, but the relationship between different nodes.



- Nodes: the fundamental unit of which graphs are formed, also called endpoint. They are connected by the edge like  $A$  and  $B$  in above picture.
- Edge: the line that connect two nodes. Each edge has two nodes to which it is attached. Edges may be directed or undirected

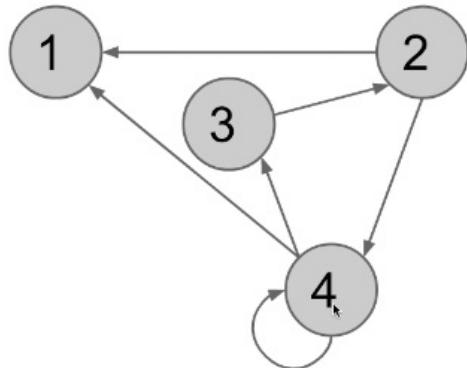
Case1: Try to count the edge between those circles.



	1	2	3	4
1	0	1	0	1
2	1	0	1	1
3	0	1	0	1
4	1	1	1	0

This undirected table is symmetric. It shows that 1 and 2 has one edge. 2 and 3 is the same, but there is no edge between 1 and 3.

## Adjacency Matrix (Self-Loop)

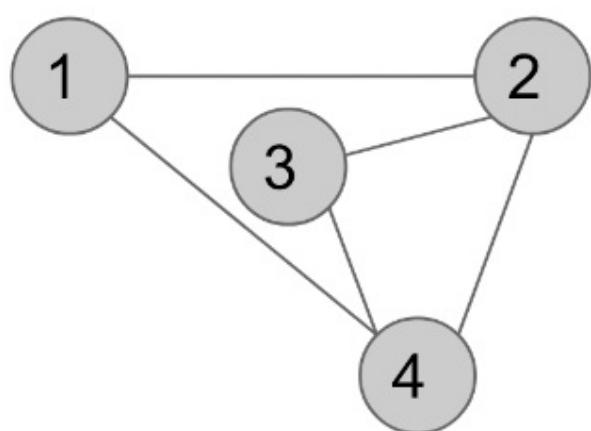


	1	2	3	4
1	0	0	0	0
2	1	0	0	1
3	0	1	0	0
4	1	0	1	1

While in the directed relationships, like the above picture,  $3 \rightarrow 2$  has one edge,  $2 \rightarrow 3$  has zero edge, this is caused by the edge directions.

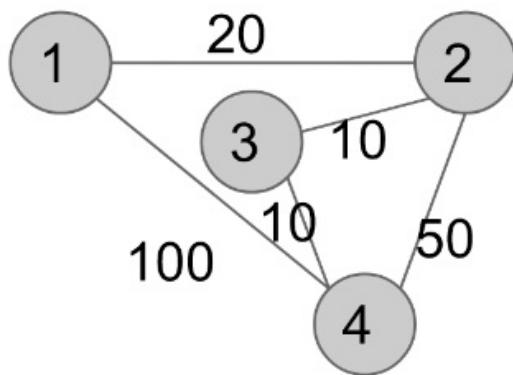
There are different ways to show the relationships.

# Edge List



(1,2)
(2,3)
(2,4)
(3,4)
(1,4)

# Edge List (Weighted)



(1, 2, 20)
(2, 3, 10)
(2, 4, 50)
(3, 4, 10)
(1, 4, 100)

## Network analysis with NetworkX

Basic usage of networkx

NetworkX is a Python package for study of the structure, dynamics, and functions of complex networks. With which we can analyze the network structure, the relationship between different nodes and generate different kind of graphs.

Import and install:

```
!pip install networkx
import networkx as nx
```

Basic usage: From the above explanation, we can know that, the network is formed by nodes and edges. Therefore, when using `networkx`, we need get the nodes and edges first, then we can draw the graph.

```
#draw an empty graph
g=nx.Graph()
g
#<networkx.classes.graph.Graph at 0x10eb6a1d0>
```

```
#add nodes
#help(g.add_node) to check out the parameters and syntax
g.add_node('A')
g.add_node('B')
g.add_node('C')
```

It adds the nodes. Then `g.nodes` to check.

```
g.nodes
#NodeView(['A', 'B', 'C'])
```

Add edges, draw the graph.

```
#help(g.add_edge) to check out the parameters and syntax
g.add_edge('A', 'B')
```

Graph show.

```
nx.draw(g)
```

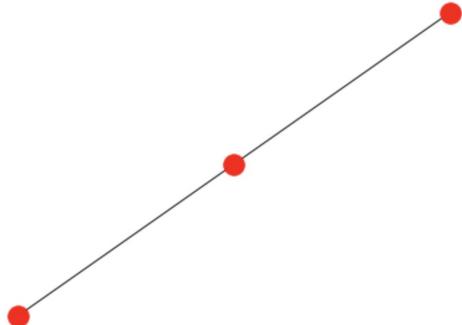
```
nx.draw(g)
```



```
#add another node between c and b
g.add_edge('C', 'B')
nx.draw(g)
```

After that, we can get one simple graph.

```
nx.draw(g)
```



## Common Network Analysis Routine via Les Misérables dataset

In the following notes, we will use characters in book *Les Misérables* to demo the analysis process. You can download the dataset [here](#)

### Graph visualization

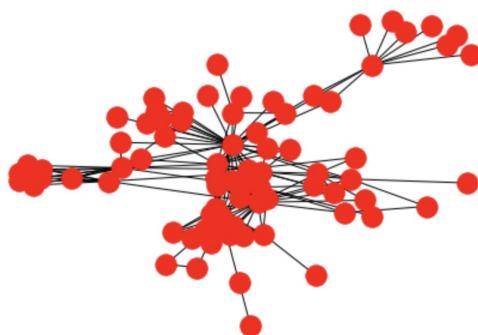
#### Basic visualization

Add all nodes and edges.

```
import json
data = json.loads(open('miserables.json').read())
data
#data.keys()
#data['nodes'] checkout nodes
#data['links'] checkout links
import networkx as nx
g = nx.Graph()

#add nodes
for n in data['nodes']:
 g.add_node(n['id'], group=n['group'])

for l in data['links']:
 g.add_edge(l['source'], l['target'])
nx.draw(g)
```



From this graph, we can know that there are some groups, but we don't know who are them. Next step for us is improving the graphs. To solve the following questions:

- Are there some groups in the network?
- Who are in the same group?

## Adjust layout

Add labels on the graph.

```
help(nx.draw) #to learn about the function and parameters. What may be useful for us are parameters and see also functions. Those are helpful for optimizing our graphs.
```

```
Parameters

G : graph
 A networkx graph

pos : dictionary, optional
 A dictionary with nodes as keys and positions as values.
 If not specified a spring layout positioning will be computed.
 See :py:mod:`networkx.drawing.layout` for functions that
 compute node positions.

ax : Matplotlib Axes object, optional
 Draw the graph in specified Matplotlib axes.

kwds : optional keywords
 See networkx.draw_networkx() for a description of optional keywords.

Examples

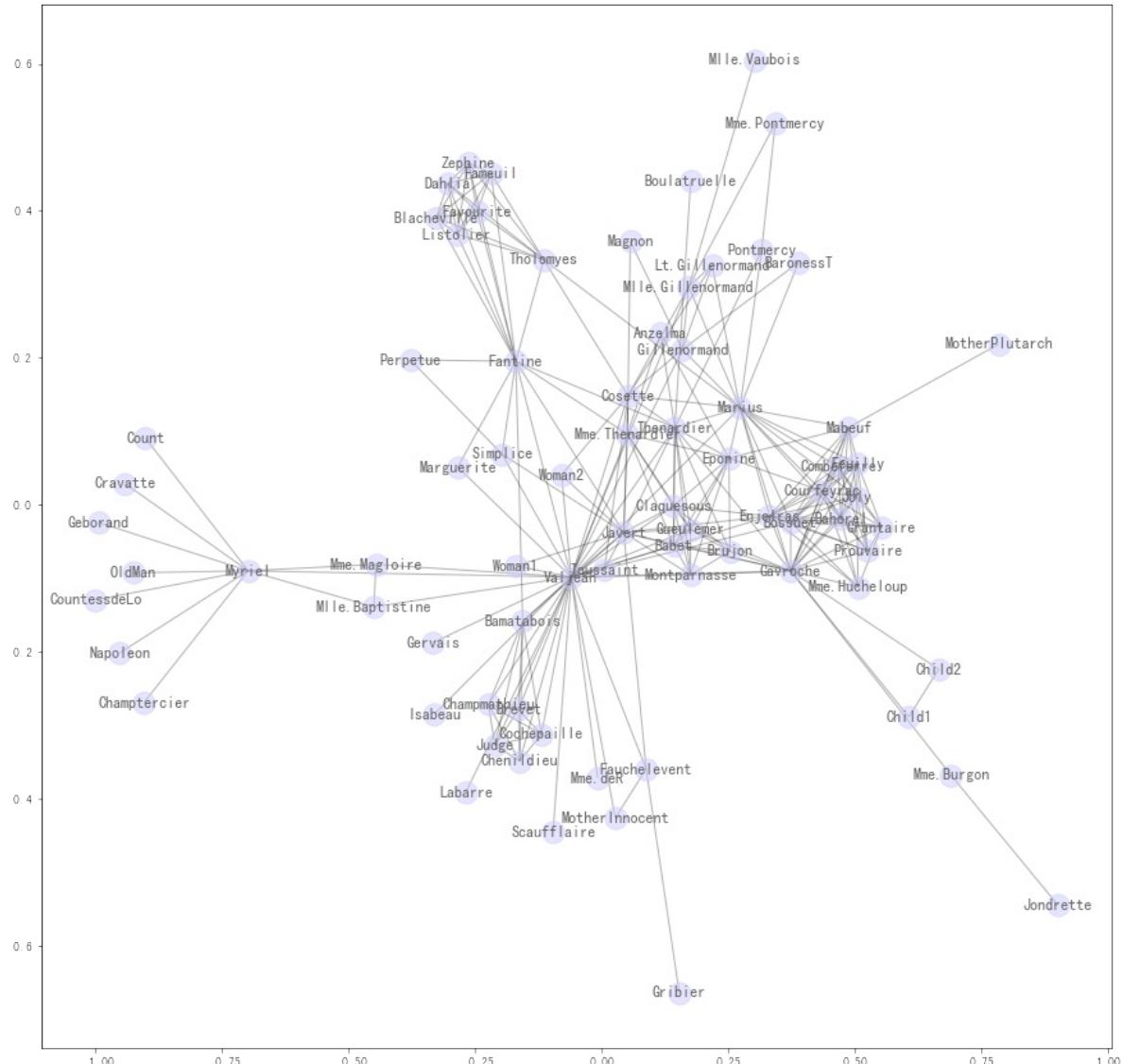
>>> G = nx.dodecahedral_graph()
>>> nx.draw(G)
>>> nx.draw(G, pos=nx.spring_layout(G)) # use spring layout

See Also

draw_networkx()
draw_networkx_nodes()
draw_networkx_edges()
draw_networkx_labels()
draw_networkx_edge_labels()
```

```
from matplotlib import pyplot as plt

plt.figure(figsize=(15, 15))
pos = nx.spring_layout(g)
nx.draw(g, pos=nx.spring_layout(g))
nx.draw_networkx_nodes(g, pos, node_color="#ccccff", alpha=0.5) #change nodes style
nx.draw_networkx_edges(g, pos, width=1.0, alpha=0.3) #change edges style
labels = dict([(n, n) for n in g.nodes]) #add labels
_ = nx.draw_networkx_labels(g, pos, labels=labels, font_color="#666666") #draw labels
```



## Group the nodes with same color

Group the nodes according to the group number in the json. Every node has a group number, we can group those nodes with the color.

```

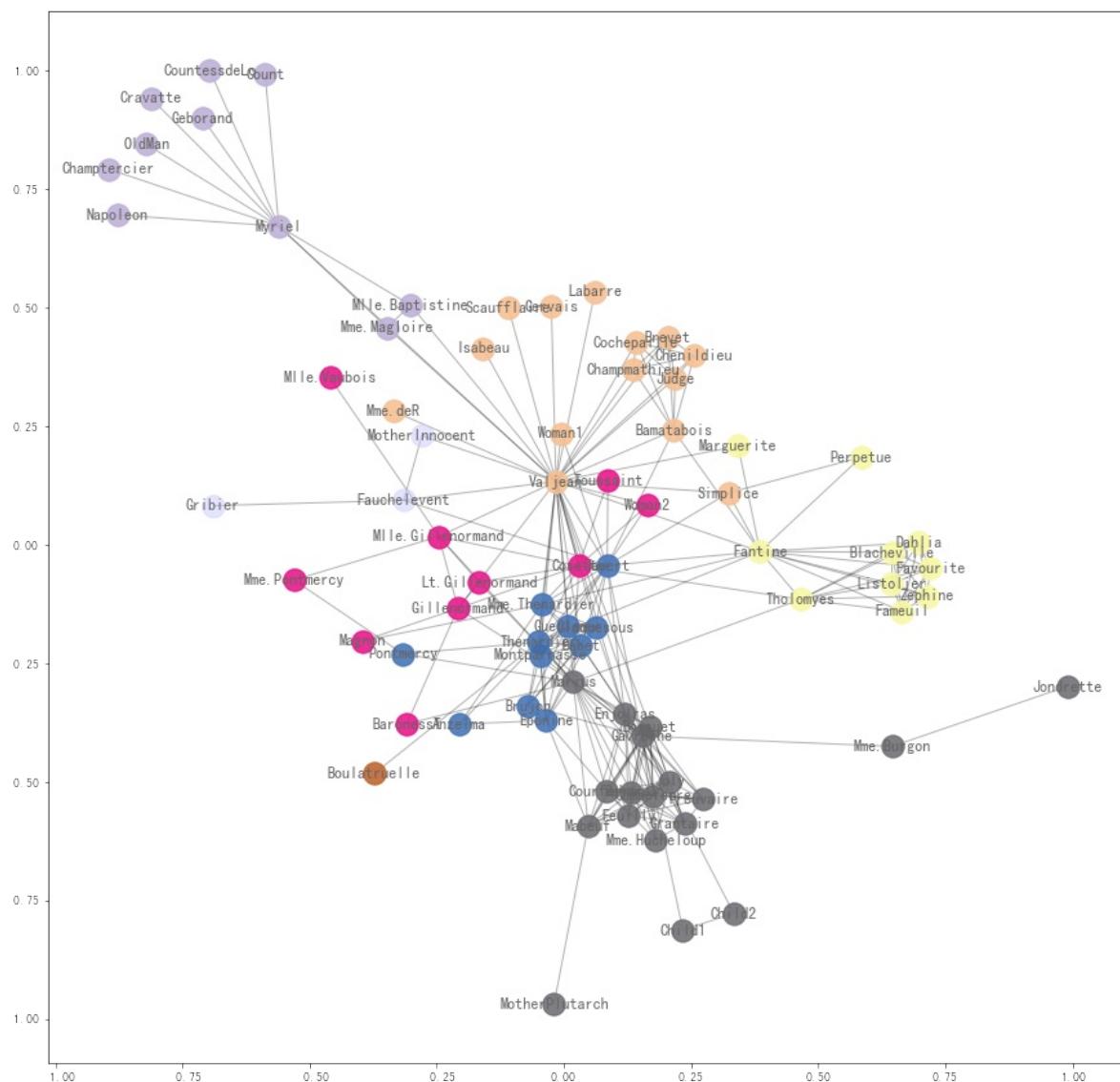
import matplotlib
color = matplotlib.cm.Accent
import color map, there are many color maps, you can checkout the color maps by the following:
import matplotlib.cm as cm
dir(cm)

plt.figure(figsize=(15, 15))
pos = nx.spring_layout(g)
nx.draw_networkx_nodes(g, pos, node_color="#ccccff", alpha=0.5)
nx.draw_networkx_edges(g, pos, width=1.0, alpha=0.3)
labels = dict([(n, n) for n in g.nodes])
_ = nx.draw_networkx_labels(g, pos, labels=labels, font_color="#666666")

for group in range(1, 20):
 nodelist = [n for n in g.nodes if g.nodes[n]['group'] == group]
 # If g.nodes's group = 1, add those nodes into the nodelist. They will be the same color 1 . If g.nodes's group = 2, they will be added to another nodelist ,and be colored 2.
 #print(nodelist)

```

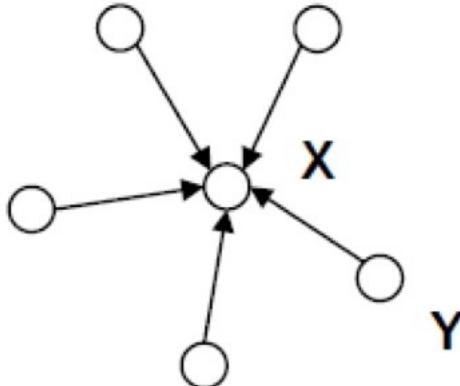
```
nx.draw_networkx_nodes(g, pos, nodelist=nodelist, node_color=color(group), alpha=0.8)
```



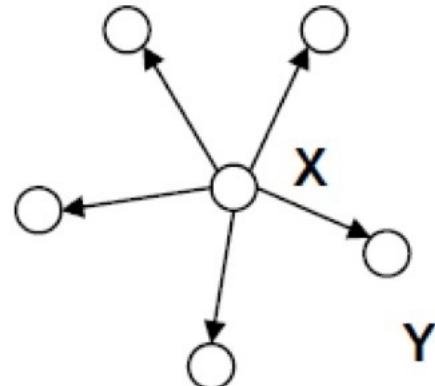
## Measure Node and Edge Importance

### Degree

In graph theory, the degree is the number of edges incident to the nodes. The degree usually represent the **importance** of nodes. Degree can be divided into In-Degree and Out-Degree. In-Degree is how many other nodes point to one node, while Out-Degree is how many other nodes one node points to. Degree is the sum of In-Degree and Out-Degree.



indegree

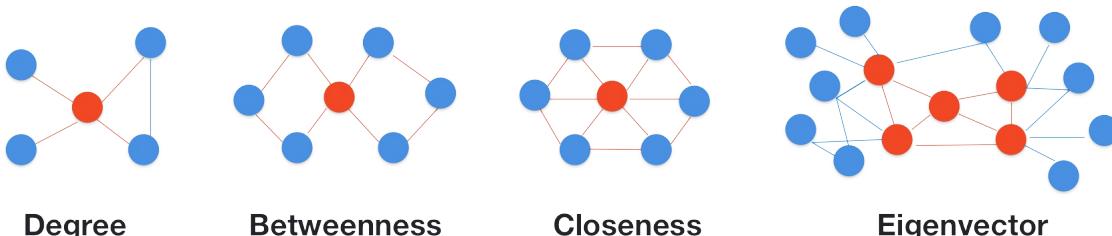


outdegree

## Centrality Measures

As we mentioned above, Degree can be divided into In-Degree and Out-Degree. The importance of nodes with the same total degree but different In-Degree and Out-Degree are different. That's the reason why We need Centrality Measures.

[Centrality](#) is a classical concept in graph analysis. It measures the "importance" of nodes. The notions of "importance" are different. We only provide some samples in following sections.



There are different centrality measures, like [Degree](#) , [Eigenvector](#) [Closeness](#) and [Betweenness](#) . The following is the interpretation of measures:

Centrality measure	Interpretation in social networks
Degree	How many nodes can one node reach directly?
Betweenness	How likely is this node to be the most direct route between two nodes in the network?
Closeness	How fast can this node reach every other nodes in the network?
Eigenvector	How well is this node connected to other well-connected nodes?

You can refer to the [documentation](#) and online resources to understand those centrality measures. Try other centrality measures that are not covered in this tutorial. See what interesting findings you can get.

```
#check out the methods of centrality, generally speaking, the greater centrality of a node, and the more import
```

```

ant the node is in the network.
nx.degree_centrality(g)
#nx.closeness_centrality(g)
#nx.betweenness_centrality(g)
#nx.eigenvector_centrality(g)

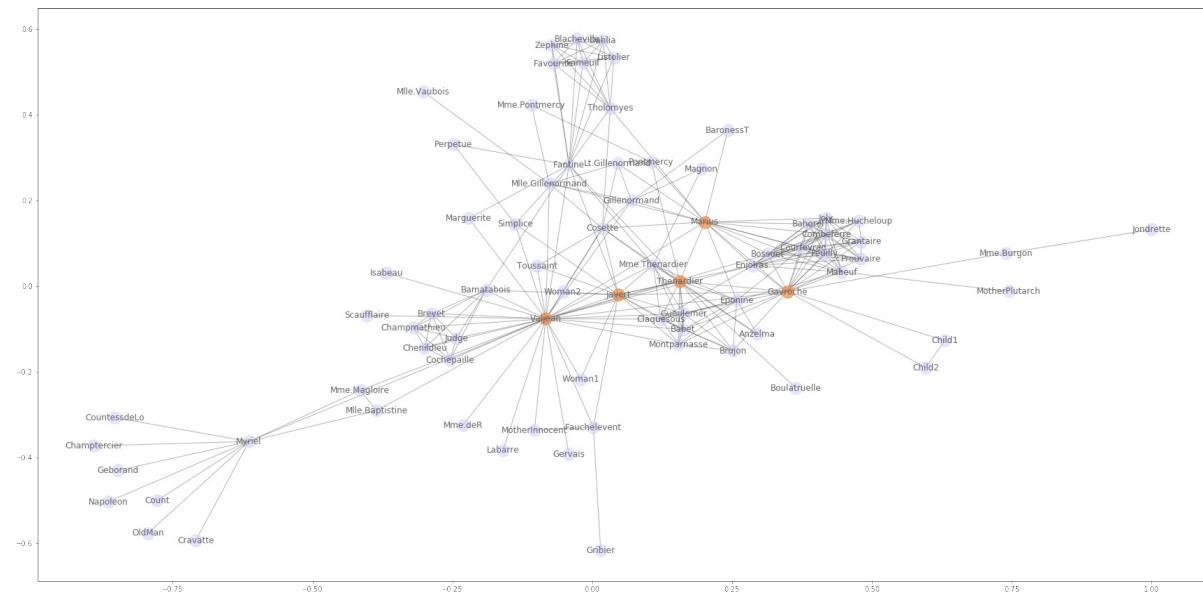
import pandas as pd
df = pd.DataFrame()
df['degree'] = pd.Series(nx.degree_centrality(g))
df['closeness'] = pd.Series(nx.closeness_centrality(g))
df['betweenness'] = pd.Series(nx.betweenness_centrality(g))
df['eigenvector'] = pd.Series(nx.eigenvector_centrality(g))

#draw degree centrality
df_top_nodes = df.sort_values('degree', ascending=False)[:5]

plt.figure(figsize=(30, 15))
we don't run the spring layout again; to keep the positions in this section
#pos = nx.spring_layout(g)
nx.draw_networkx_nodes(g, pos, node_color='#ccccff', alpha=0.5)
nx.draw_networkx_edges(g, pos, width=1.0, alpha=0.3)
labels = dict([(n, n) for n in g.nodes])
_ = nx.draw_networkx_labels(g, pos, labels=labels, font_color='#666666')
nx.draw_networkx_nodes(g, pos, nodelist=list(df_top_nodes.index), node_color='#ff7700', alpha=0.5)

df_top_nodes
#change degree to other three columns to see the different top nodes.

```



From centrality analysis, we can figure out the `key figures` and nodes in the network, and get the next step analysis leads.

Relative reading:

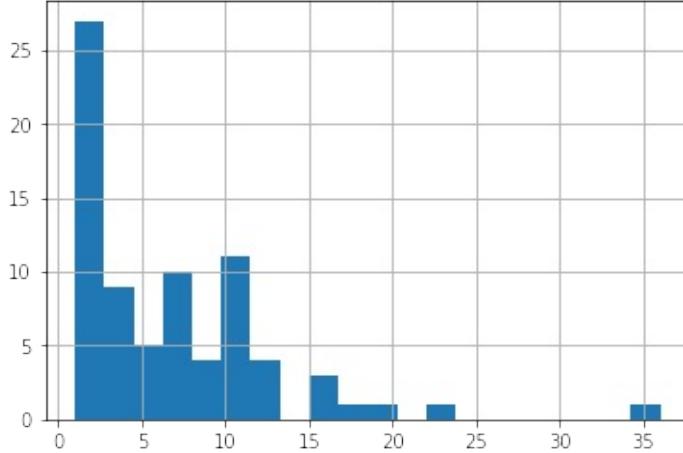
- 政商網絡系列文：陳電鋸.
- Network analysis of Game of Thrones

## Basic statistics of graph

### Degree distribution

In the above session, we mentioned that the degree of a node is the number of edges it has to other nodes. The degree distribution is the probability distribution of these degrees over the whole network.

```
g.degree
pd.Series(dict(g.degree())).hist(bins=20)
```



`dict(g.degree())` and then `Series`. Then Draw a picture.

From the histogram, we can see that the minority nodes have large number of edges while majority have less edges. This is a [Heavy tail distribution](#), which is famous for rich will be richer and poor will be poorer.

## Clustering coefficient

A clustering coefficient is a measure of the degree to which nodes in a graph tend to cluster together. *From [wiki](#)*

There are different kind of clustering coefficient, including global clustering coefficient, local clustering coefficient, average clustering coefficient. The simplest is `global clustering coefficient`, which is the number of closed triplets (or 3 x triangles) over the total number of triplets (both open and closed). The larger the clustering coefficient is, the closer one node is wth other nodes. For usage in `networkx`, you can refer [here](#).

```
nx.algorithms.clustering(g, ['XXX', 'XXX', 'XXX']) #pass a set of nodes in the list
nx.average_clustering(g)
```

```
gorithms.clustering(g, ['Myriel', 'Champtercier', 'Count', 'Cravatte', 'Napoleon', 'Geborand', 'CountessdeLo',
'OldMan'])
{'Champtercier': 0,
'Count': 0,
'CountessdeLo': 0,
'Cravatte': 0,
'Geborand': 0,
'Myriel': 0.06666666666666667,
'Napoleon': 0,
'OldMan': 0}
nx.average_clustering(g)
0.5731367499320134
nx.average_clustering(nx.complete_graph(5))
1.0
```

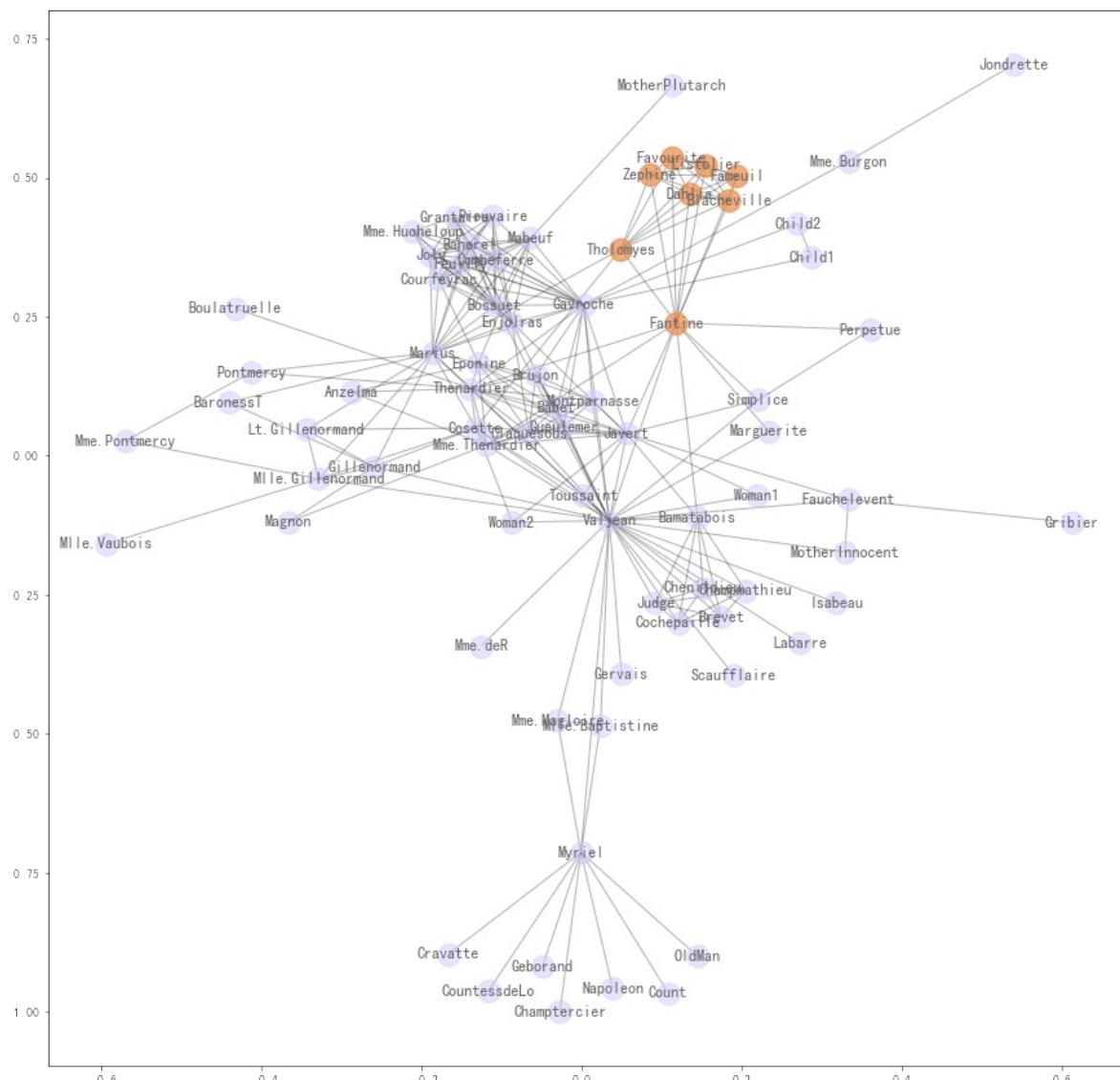
## Structure of a graph

### Cliques

Simply speaking, a clique is a subset of nodes in an undirected graph. Which is a segmented group in a bigger community. Highlight those cliques may help us know the core figures and groups in one network.

```
cliques = list(nx.find_cliques(g))
#len(cliques)
cliques[0:2]

plt.figure(figsize=(15, 15))
pos = nx.spring_layout(g)
nx.draw_networkx_nodes(g, pos, node_color='#ccccff', alpha=0.5)
nx.draw_networkx_edges(g, pos, width=1.0, alpha=0.3)
labels = dict([(n, n) for n in g.nodes()])
_ = nx.draw_networkx_labels(g, pos, labels=labels, font_color="#666666")
nx.draw_networkx_nodes(g, pos, nodelist=cliques[12], node_color='#ff7700', alpha=0.5)
#draw any clique by changing nodelist=cliques[12]
```



## Connected components

To find those who are not connected by any others.

```
components =list(nx.connected_components(g))
len(components)
```

## Community detection

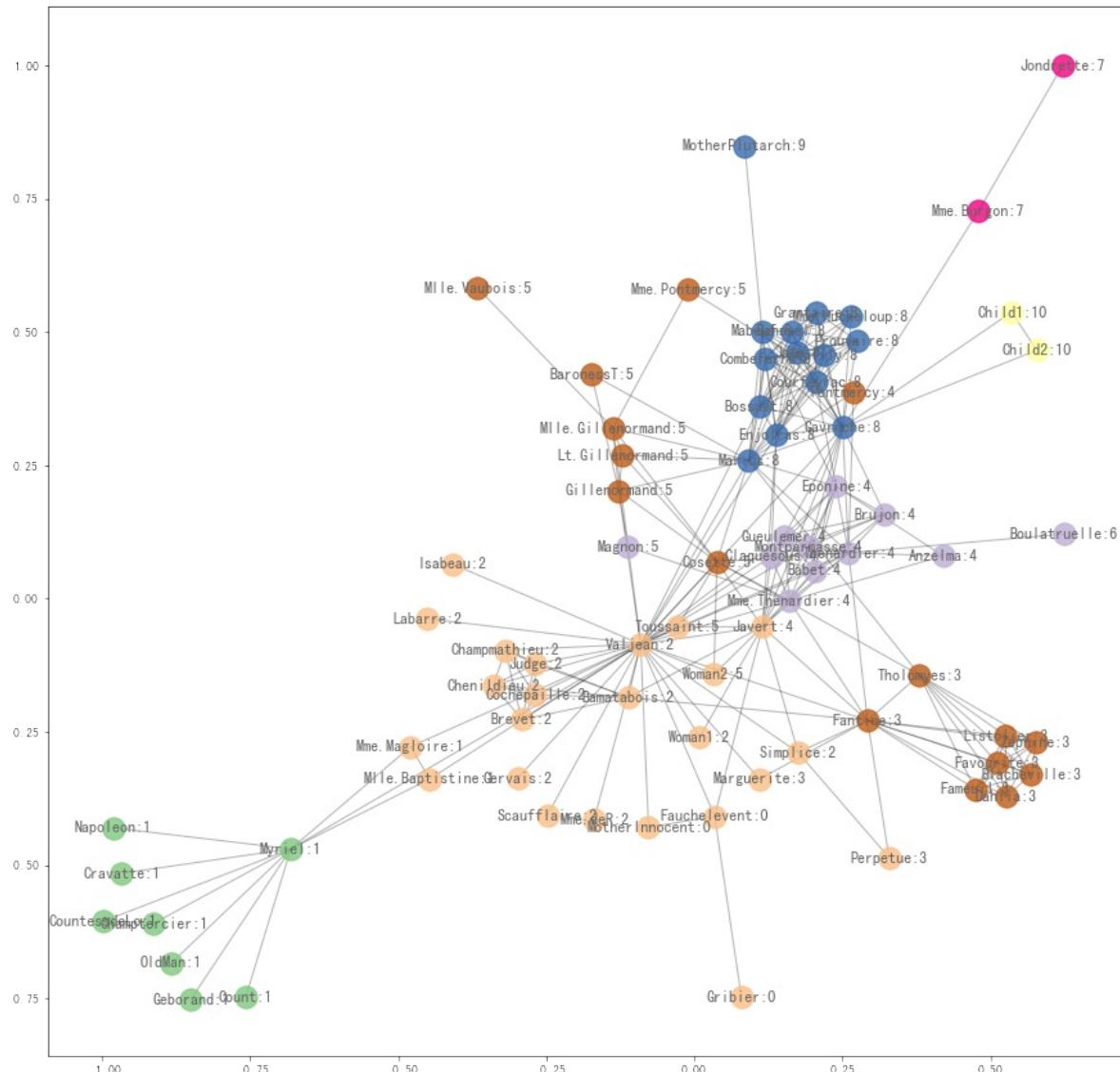
In the network analysis, community structure refers to the occurrence of groups of nodes in a network that are more densely connected internally than with the rest of the network *From wiki*.

Community detection can help us categorize every single node into different groups based on different characteristics so that we can study them as whole. Other importance of community detection is to help us find the `missing links` and `identify the false links` in one network.

```
from networkx.algorithms import community
communities = list(community.label_propagation_communities(g))
#communities[0]

plt.figure(figsize=(15, 15))
pos = nx.spring_layout(g)
#nx.draw_networkx_nodes(g, pos, node_color='ccccff', alpha=0.5)
nx.draw_networkx_edges(g, pos, width=1.0, alpha=0.3)

for i in range(0, len(communities)):
 nodelist = communities[i]
 print(nodelist)
 nx.draw_networkx_nodes(g, pos, nodelist=nodelist, node_color=color(i), alpha=0.8)
 labels = dict([(n, '%s:%s' % (n, g.nodes[n]['group'])) for n in nodelist])
 nx.draw_networkx_labels(g, pos, labels=labels, font_color='#666666')
```



## Other Graph algorithms

## Shortest path

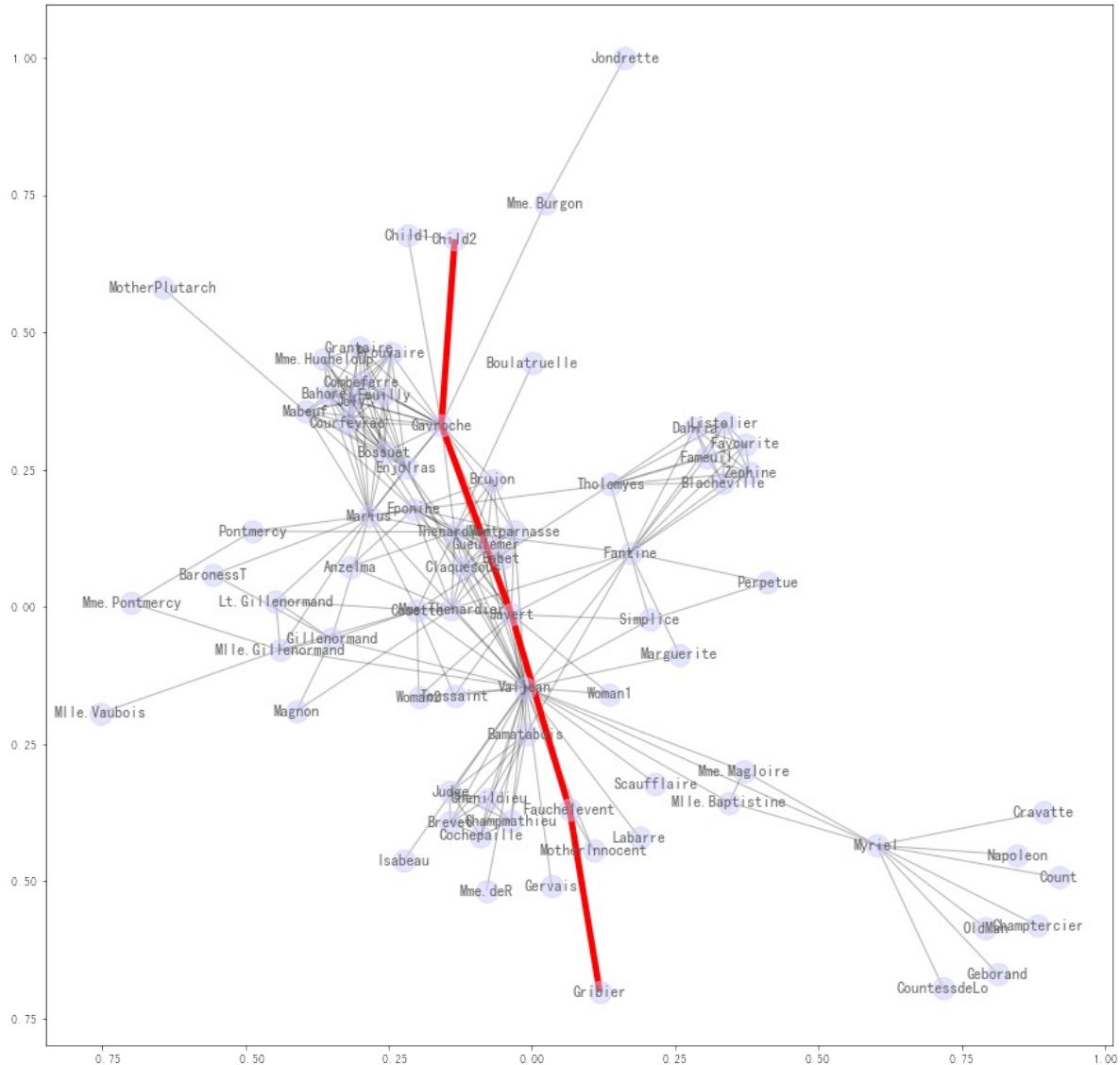
Draw the shortest path between two nodes.

```
sp = nx.shortest_path(g, 'Gribier', 'Child2')
#you can change to any other two nodes
sp

plt.figure(figsize=(15, 15))
#pos =nx.spring_layout(g)
nx.draw_networkx_nodes(g, pos, node_color='#ccccff', alpha=0.5)
nx.draw_networkx_edges(g, pos, width=1.0, alpha=0.3)
labels = dict([(n, n) for n in g.nodes])
_ = nx.draw_networkx_labels(g, pos, labels=labels, font_color="#666666")

nx.draw_networkx_edges(g,
 pos,
 edgelist=list(zip(sp[:-1], sp[1:])),
 width=5,
 edge_color='r'
)
```

```
#help(nx.draw_networkx_edges)
#edgelist:collection of edge tuples
#list(zip(sp[:-1], sp[1:])) check out the edgelist
[('Gribier', 'Fauchelevent'),
('Fauchelevent', 'Javert'),
('Javert', 'Gavroche'),
('Gavroche', 'Child2')]
```



## Other Network Visualization Libraries and Tools

### kumu.io

#### References:

- Use Kumu To Make A Relationship Graph For Les Misérables <https://dnnnsociety.org/2018/01/17/use-kumu-to-make-a-relationship-graph-for-les-miserables/>
- 關係圖表速成方案: Google Fusion Tables & Kumu, <https://dnnnsociety.org/2017/08/28/%E9%97%9C%E4%BF%82%E5%9C%96%E8%A1%A8%E9%80%9F%E6%88%90%E6%96%B9%E6%A1%88%EF%BC%9Agoogle-fusion-tables-kumu/>

## Google Fusion Table

- 關係圖表速成方案: Google Fusion Tables & Kumu ,  
<https://dnnssociety.org/2017/08/28/%E9%97%9C%E4%BF%82%E5%9C%96%E8%A1%A8%E9%80%9F%E6%88%90%E6%96%B9%E6%A1%88%EF%BC%9Agoogle-fusion-tables-kumu/>

## pyecharts

## Reference examples

- Who control the discourse power in 红楼梦 by Group 8 2018S
  - Li's family business map and spring layout analysis by Group 7 2018S
- 

If you have any questions, or seek for help troubleshooting, please [create an issue here](#)

# Week 15: Geographical data

- [Week 15: Geographical data](#week-15-geographical-data) - [Objective](#objective) - [Geographical System](#geographical-system) - [Geocoding: turn string address data into geo coordinates](#geocoding-turn-string-address-data-into-geo-coordinates) - [Geographical Reference Systems (GRS)](#geographical-reference-systems-grs) - [Projection system](#projection-system) - [Mercator projection](#mercator-projection) - [File Formats](#file-formats) - [GeoSeries and GeoDataFrame](#geoseries-and-geodataframe) - [GeoJSON](#geojson) - [TopoJSON](#topojson) - [KML](#kml) - [Mapping](#mapping) - [Map types](#map-types) - [Map components](#map-components) - [When map is used to show correlation](#when-map-is-used-to-show-correlation) - [When map hides key information](#when-map-hides-key-information) - [Case studies](#case-studies) - [Shanghai rental sources map visualization](#shanghai-rental-sources-map-visualization) - [Air crash map using plotly](#air-crash-map-using-plotly) - [Openrice Sichuan Food using folium](#openrice-sichuan-food-using-folium) - [Global data journalist distribution and contribution map using plotly](#global-data-journalist-distribution-and-contribution-map-using-plotly) - [American journalist job market map using plotly](#american-journalist-job-market-map-using-plotly) - [Tourists footprint of Domestic Tourist Cities](#tourists-footprint-of-domestic-tourist-cities) - [Number of journalists killed in different countries](#number-of-journalists-killed-in-different-countries) - [England and Ireland pubs using matplotlib](#england-and-ireland-pubs-using-matplotlib) - [Hong Kong property price bubble chart using folium](#hong-kong-property-price-bubble-chart-using-folium) - [United States unemployment rate 2012 choropleth using folium](#united-states-unemployment-rate-2012-choropleth-using-folium) - [Bonus: Other GIS and mapping tools](#bonus-other-gis-and-mapping-tools) - [QGIS](#qgis) - [ArcGIS](#arcgis) - [Carto](#carto) - [D3](#d3) - [MapShaper](#mapshaper) - [Google Fusion Table](#google-fusion-table) - [Exercises](#exercises) - [Distances among cities](#distances-among-cities) - [Extended exercise of geo distance](#extended-exercise-of-geo-distance)

## Objective

Understand geographical data

Libraries:

- `geopandas`
- `geopy`
- `folium`
- `plotly`
- `matplotlib`

## Geographical System

Following are the major steps and considerations when dealing with geographical data:

1. Geocode: turn geographical names into longitude and latitude coordinates. For example, you can not plot `Hong Kong` on a map, but you can plot `(114.141, 22.362)` on the map. (you can use [geojson.io](#) to quickly get the data).
2. Projection: even if you get the geo coordinates somehow, it still can not be plotted on the screen directly. We need a translation from the geo coordinates to screen coordinates. For example, if we want to put HK in the center of the a `640px by 480px` 2D map, we need to establish a mapping like `(114.141, 22.362) --> (320px, 240px)`. This process is called projection. The actual project is more complex than that. Here's a demo of [different methods of projection](#).
  - Scatter plot/ bubble plot -- simply project the point coordinates
  - Choropleth -- one needs to project a geometry
3. Base layer: maps are usually organised into layers. Besides putting the data points we are interested in onto the

map, we also show some geographical information, like constituency boundaries, streets and contours. This is the benefit of map -- put new data points onto a plate that people are already familiar with. This kind of information usually comes with the "base layer", whereas the above plotted elements are in "data layers". Choices for base layer are like Google Maps, Open Street Map, Mapbox, etc.

## Geocoding: turn string address data into geo coordinates

Geocoding is usually done via a web service. The service is costly so you can seldom find free service nowadays. `geopy` has encapsulated many useful geocoding services for your selection. `Nominatim` is a frequently used free service. You just need to specify your user agent (any string would work) and control the request rate.

```
from geopy.geocoders import Nominatim
geolocator = Nominatim(user_agent='specify_your_app_name_here')
location = geolocator.geocode('The address that you want to geocode')
location.point
```

Google Map once provided a free API. It ceased operation since July 2018. Now you must apply a Google API key before using this service. The first hundreds requests everyday are free. Followup requests are charged at US\$5 per 1000 requests. You can checkout details in the [billing plan](#). The core code is as follows:

```
from geopy.geocoders import GoogleV3
geolocator = GoogleV3(api_key='Your API Key from Google API')
location = geolocator.geocode('The address that you want to geocode')
location.point
```

One can refer to [this notebook](#) for a real and complete working example of geocoding. It is part of the [HK Sichuan food growth map](#) visualization.

## Geographical Reference Systems (GRS)

A Point-Of-Interest (POI) is denoted by a two dimensional coordinate system (two-element tuple). The location of POI is always in a relative sense. In order for systems to communicate with each other and accurately refer to the same location on earth, Geographical Reference Systems (GRS) needs to be specified. GRS is like the protocol between GIS systems. One GRS specifies the followings:

- The projection method
- Center of the map
- Scaling factor of the map

For example, the geocoding results from above section are a pair of (longitude, latitude) values, which are referencing to the "WGS1984 CRS" ([EPSG:4326](#)), i.e. longitude in range  $[-180, 180]$  and latitude in range  $[-90, 90]$ . If you checkout the [district council boundary file](#) from Hong Kong's Census and Statistics Department, you will find the coordinates are very large numbers. That is because Hong Kong conventionally used "Hong Kong 1980 CRS" ([EPSG:2326](#)) in government official files. If you put those files into some visualisation tools like [mapshaper](#), there is no problem displaying them individually. However, when you use those files in modern mapping libraries, the plotted geographical elements may not be at the location you expect.

Most modern mapping library and GeoJSON file use WGS1984 CRS. Usually this step is hidden from a normal user. However, if you encountered some ancient files, you may need to handle the CRS conversion. [Here](#) is a practical case of CRS conversion.

## Projection system

We live on a spherical surface but the computer screen is rectangular. The process to convert shapes from the former to the latter is called "projection". Here is a demo of [different methods of projection](#).

## Mercator projection

Mercator projection is the most widely used projection. One can see the most familiar world map using Mercator projection.

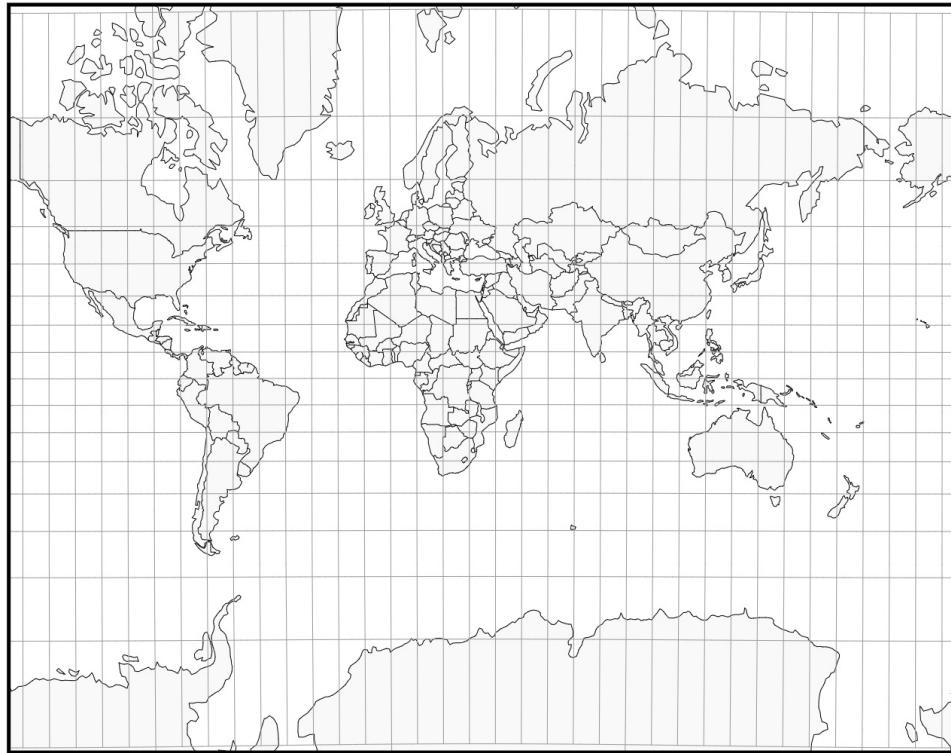


Image from [Jason Davies's map projection explorer](#)

[Here](#) is an excellent video to show you how our conventional world map can be misleading.

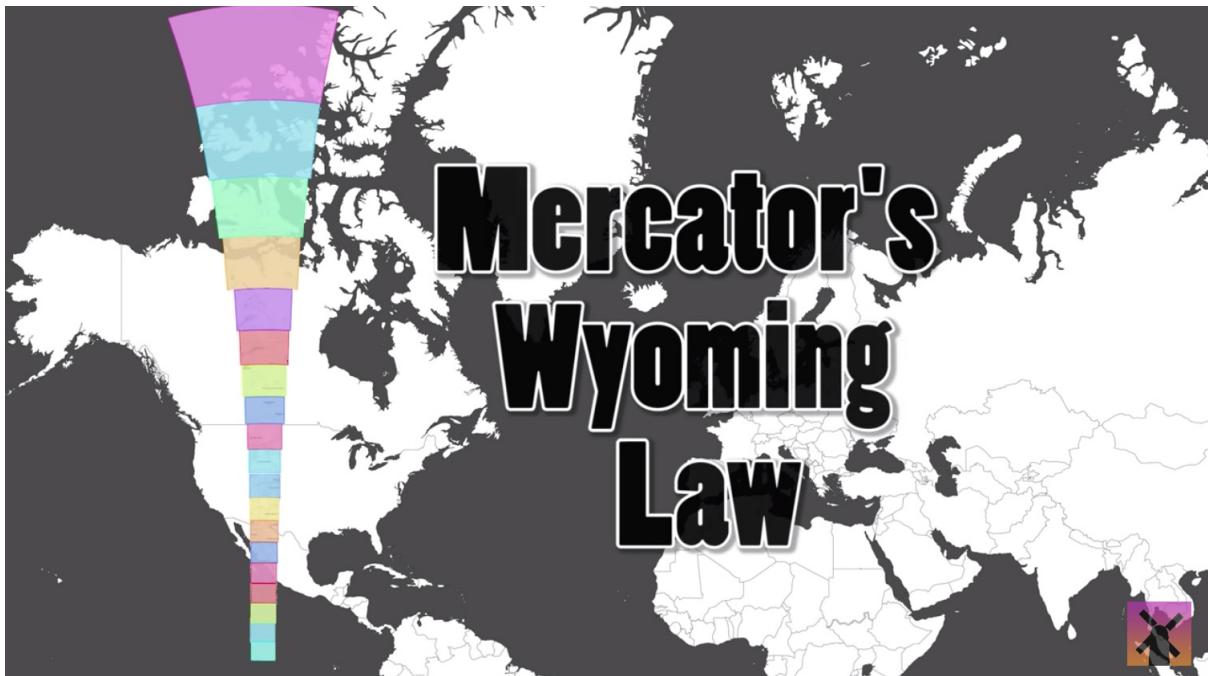


Image from "How the World Map Looks Wildly Different Than You Think"

The key take-away is that, the farther away from earth equator, the larger the distortion.

## File Formats

### GeoSeries and GeoDataFrame

Those are the subclasses (extension) of `pandas`'s `DataFrame` and `Series`. In essence, it adds one `geometry` column to the ordinary `pandas` table. Technically, there can be more columns representing geometry but one needs to be active in order to enable geometric arithmetics. The `geopandas` module implements all the above GIS basic operations, like geocoding, converting CRS, calculating projections, etc. It also supports geometric arithmetics like `intersects` and `contains`. Those operations can be carried out on a vector for convenience. There are also counterpart like `join` two `DataFrame` in `GeoDataFrame` -- `sjoin`, a.k.a "spatial join", who leverages supported geometric arithmetics on geo elements.

### GeoJSON

GeoJSON is a lightweight file format to store geographical data. It is based on JSON and can be easily load/processed by many programming languages. Read more about the file format specification on <http://geojson.org/> and try to draw GeoJSON files on <http://geojson.io/>.

### TopoJSON

GeoJSON format can result in very large files. It can be a prohibitive factor for widely deployed web service.

TopoJSON can significantly reduce the file size. It is based on the following key ideas:

- Reduce redundant/ shared arcs between geometries to save space
- Use fixed-precision delta-encoding for integer coordinates

### KML

KML was an early format intended for web based mapping services. It is supported by Google and still works as main (or only) format in Google services. Since it is XML like file format, it usually has larger file size than JSON based format (GeoJSON/TopoJSON). Find more on [wiki](#) and try to visualise KML via Google Fusion Map.

## Mapping

"Mapping" refers to the process of visualizing data on maps, a.k.a data visualization on maps. So the key issue of mapping is to determine which visual element is used to present the data. That leads to three major types of maps: visualise by point, by line and by area.

### Map types

This section discusses some common map types.

Plotting points: (Point of Interest; POI)

- Scatter plot -- plot the coordinates on map
  - [England and Ireland seen from pub locations](#)
- Bubble plot -- use size/ radius to represent a 3rd dimension data
  - [Sichuan earth quake in 100 years](#)
- Heat map -- diffuse a point to surroundings and shows the density
  - [Mapbox heatmap example](#)
- Clustering map -- group nearby points to clusters to make scatter plot less cluttered. Clustering map to heatmap is like [histogram to KDE](#)
  - [World airport clustering map](#)

Plotting lines:

- Line graph/ path graph
  - [Boeing 787-8 draws a self portrait in the air](#)

Plotting areas:

- Choropleth -- represent data value by color; a choropleth is equivalent to a bar chart, where data x-axis of the bar becomes geometry on a map. Think twice before you use choropleth, because in many cases bar chart works better.
  - [US unemployment rate via plotly](#)
  - [HK Census 2011 visualization](#)
- Cartogram -- represent data value by size of geometry (distortion)
  - [Cargogram of the US](#)
  - [US 2016 election forecast by FiveThirtyEight](#)

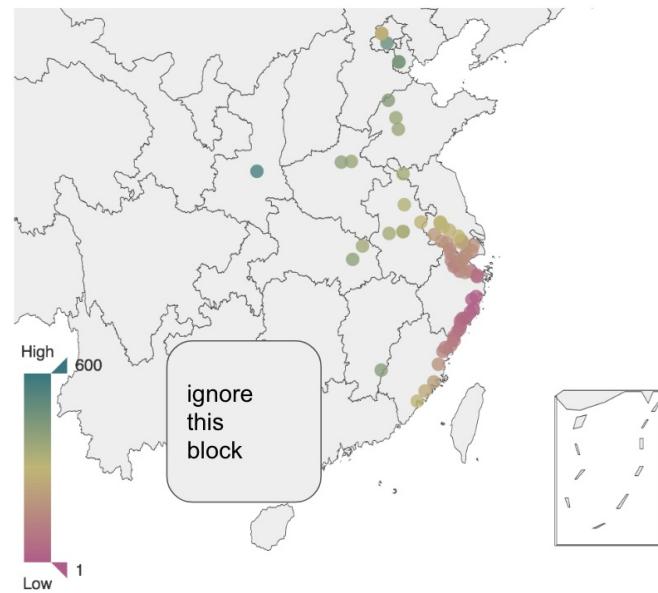
### Map components

- Feature
- Layer
  - Layer
  - Background Layer
- Auxiliary
  - Tooltip
  - Highlight
  - Toolbox

## When map is used to show correlation

### Correlation demonstrated by a map

- Plot the hours needed to go from Taizhou to major cities by railway.
- Correlation: farther away, longer the hours.
- Findings: going south is easier.



Map can be very effective when the data value is correlated with geolocation. It helps one to identify patterns and discover anomalies.

## When map hides key information



Source: 朱骋远、杨祝娟、纪麟、季勇伟

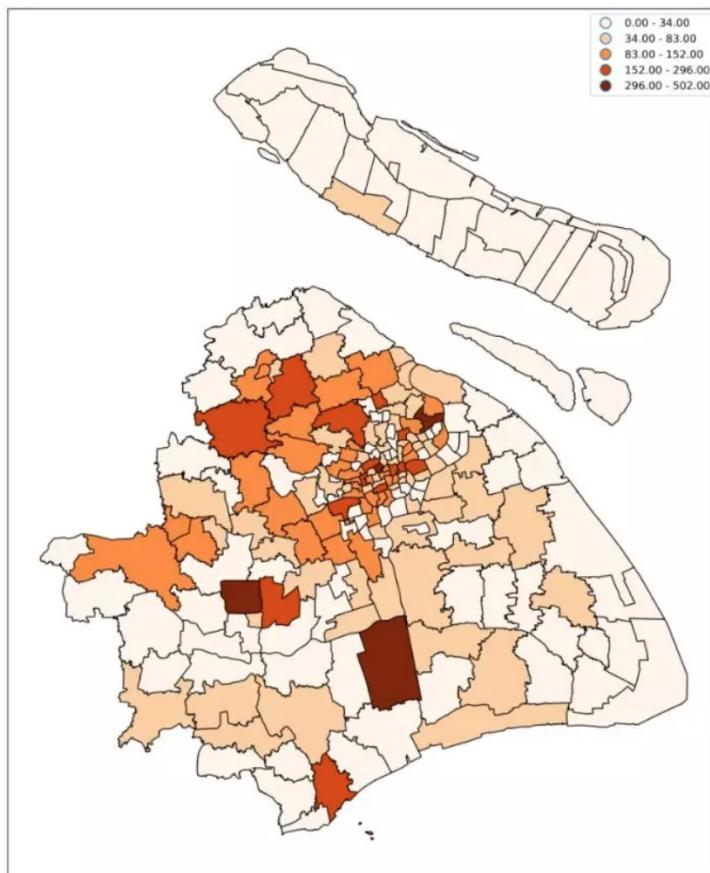
**Map can often hide key information**, because the larger area on a map does not carry an equivalent importance of data. In the above example, Europe, especially Spain, needs to be highlighted because those are the major targets of Qingtian migrants. However, due to the map area issue, Brazil catches one's eye more easily.

One way to solve this problem technically is to plot Cartogram. However, bar chart could come handy most of the time.

## Case studies

This section includes some selected map visualization cases **made in Python**. There are many other tools that can help you make maps, most notably QGIS, D3 and Carto. We leave pointers in the "other tools" section for readers' reference.

### Shanghai rental sources map visualization



- Use `geopandas` for geocoding, visualization.
- Use `GeoDataFrame.sjoin()` to correlate points into polygons (administrative areas).
- Use `GeoDataFrame.groupby().count()` to count the POIs in each area.

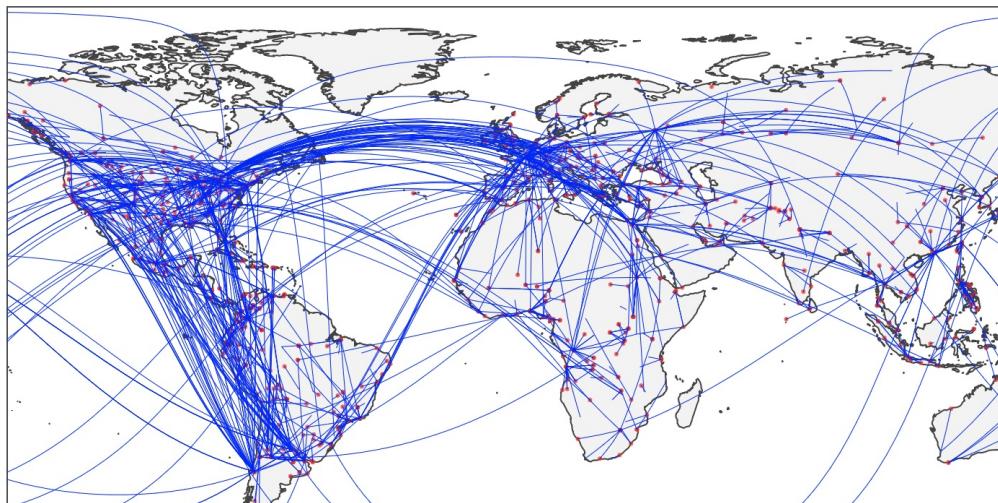
More details can be found on [this article](#).

### Air crash map using plotly

Following is an example of plotting interactive map with plotly. It's a report about the air crashes in the past 70 years around the world.

For the visualization of the map, the key data we should get is the longitude and latitude of each cities, and organize the start station and the end station of each path. Then, with help of `plotly`, we can get an interactive map.

flight path



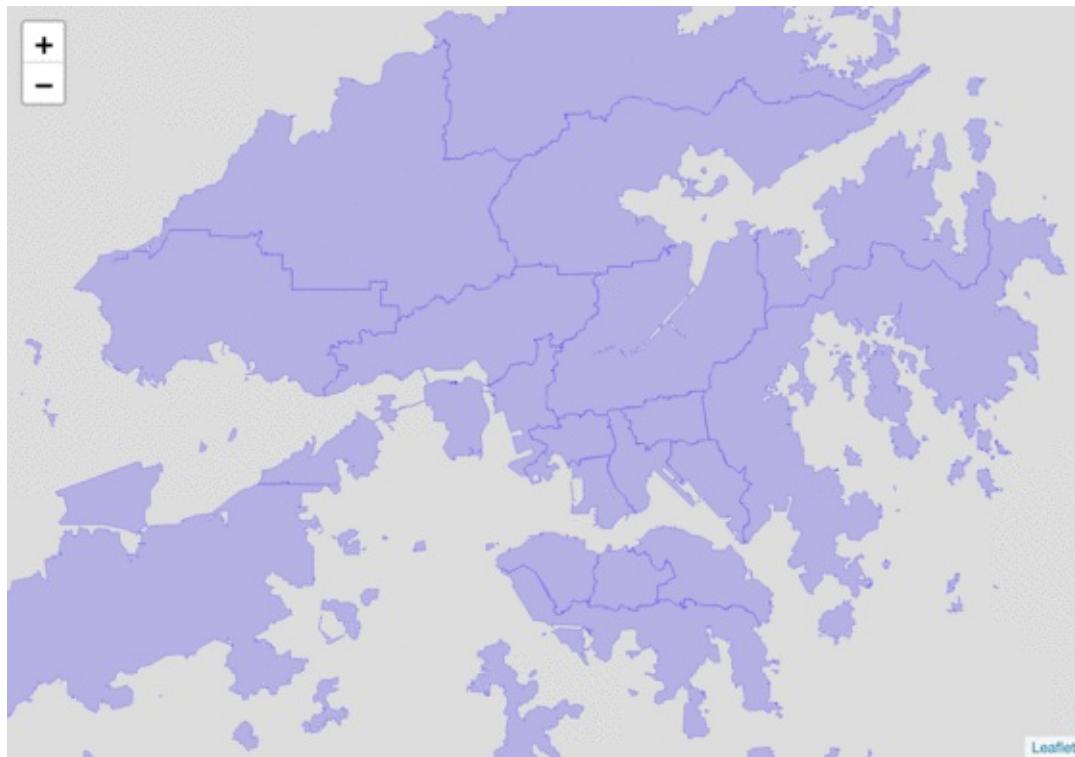
The tools and process:

- Get the data from the open source website in Socrata, World Bank, and NGIA.
- Use `pandas` to curate and restructure all the data source
- Use `plotly` to visualize the map

You can refer [here](#) for the whole story. The dataset and codes can be found [here](#).

## Openrice Sichuan Food using folium

Following is an animated map showing how Sichuan restaurants rolled out in Hong Kong.



The tools and process:

- Use `requests`, `selenium`, and `beautifulsoup` to collect data
- Use `geopy` to perform geocoding, i.e. turn address into geo-location
- Use `folium` (built-on leaflet.js) to visualise circles on map
- Use selenium to take screenshot
- Use ImageMatick and `gifsicle` to combine screenshots into gif

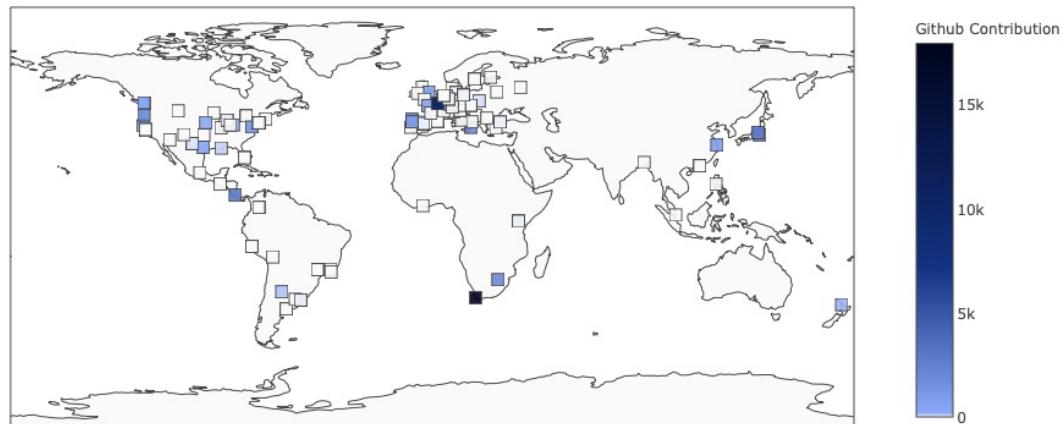
Code repo: <https://github.com/hupili/openrice-data-blog-201811>

## Global data journalist distribution and contribution map using ploty

Following is an example of scatter plots on maps about how data journalists distribute all over the world.

Like the air crash map above, this map's key data that we should get is also the longitude and latitude of each cities. In addition, we need another dimension to assign the color of each point on this map. In this case, the depth of color represents a journalist's overall github contribution from 2008.

Global Data Journalist Distribution and Their Github Contributions  
Source: Global Data Journalist Directory



The tools and process:

- Get the geographical data and other information of a journalist from the csv files
- Use `aggregate()` to accumulate each journalist's Github contribution data from 2008
- Use `plotly` to visualize the map

The dataset and codes can be found [here](#).

## American journalist job market map using ploty

This is an example of choropleth map and a report about the condition of the employment market for journalists in the U.S. In this map, the key data is the number of opening positions in each state of the U.S. In addition, this map recognise the states with their abbreviations(VA, NY...).



The tools and process:

- Get the data from `o jobs.csv`

- Use `pandas` to manipulate the location data from `0_jobs.csv` into `us-States.csv` to code the states of the U.S into their abbreviations
- Use `list.count()` to categorise the jobs into different states
- Use `plotly` to visualize the map

The dataset and codes can be found [here](#).

## Tourists footprint of Domestic Tourist Cities

This is the interactive map about footmarks of tourists from the top5 domestic tourist cities. As you can see these dots and arrows, most of them are not only hot destinations, but also hot origins. Therefore, they connect with each other and form a network. These footmarks almost concentrate on Eastern China.



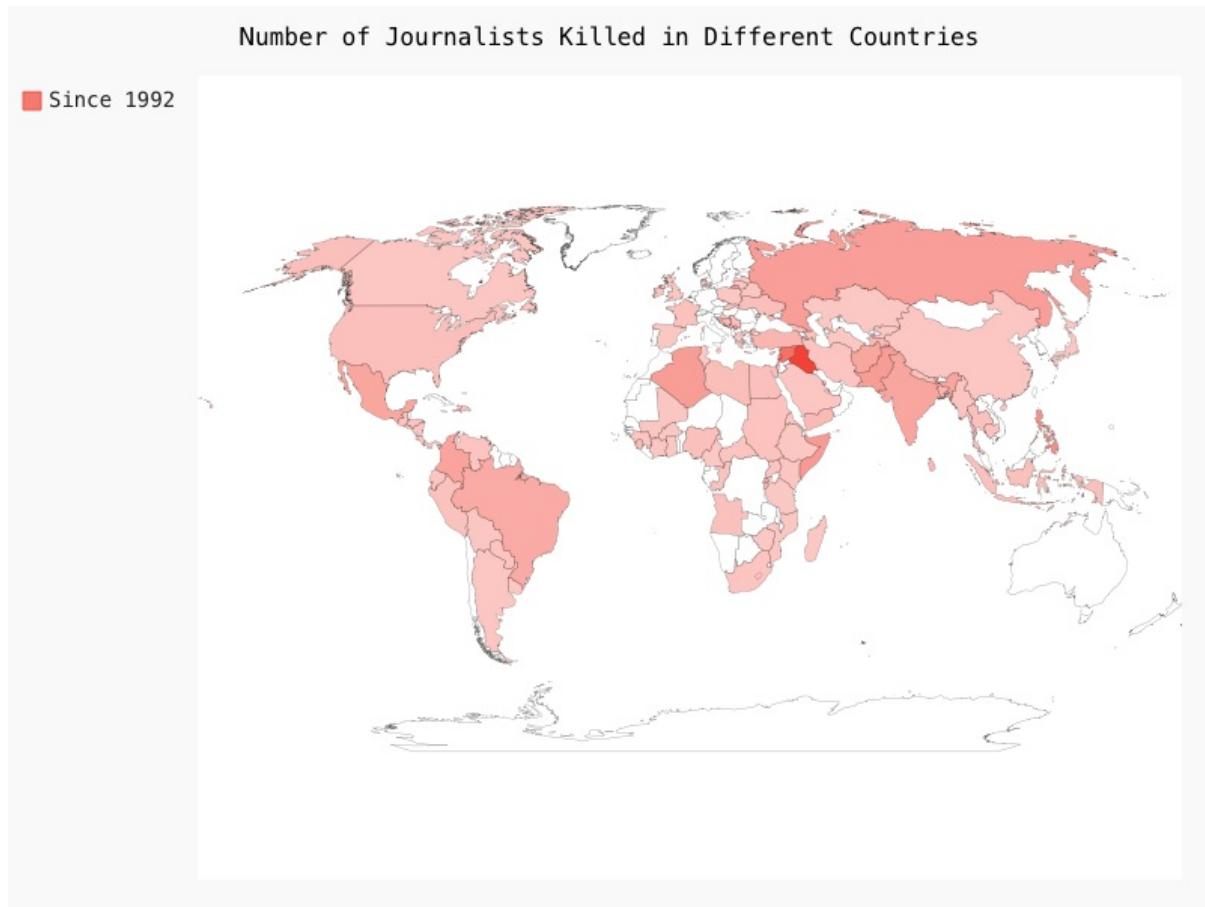
The tools and process:

- Get the data from `new_final.csv`
- Use `pyecharts` and `GeoLines` to visualize the maps.

The dataset and codes can be found [here](#).

## Number of journalists killed in different countries

The following map is an interaction diagram(svg) about numbers of journalists killed worldwide. The deeper the color of a country, the more news workers were killed in that country and the more dangerous it is for reporters. However, there is not any interaction in this page. You can click [here](#) to try out the interaction effect.



The tools and process:

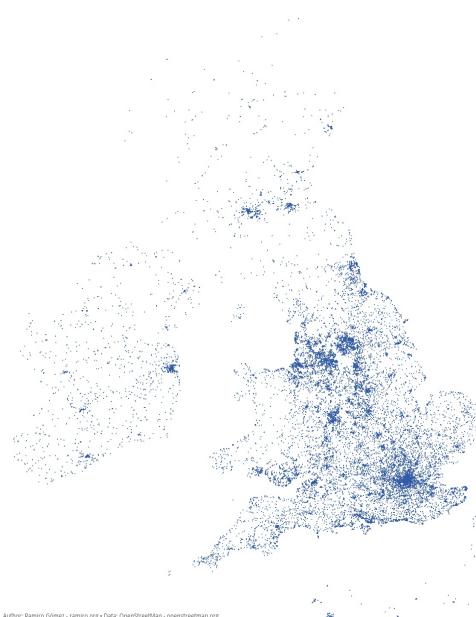
- Get data from [CPJ](#)
- Use `pygal.maps.world` to plot the map

The dataset and codes can be found [here](#).

## England and Ireland pubs using matplotlib

A map of Britain and Ireland pubs.

## Britain & Ireland drawn from pubs



Author: Ramiro Gómez - ramiro.org • Data: OpenStreetMap - openstreetmap.org

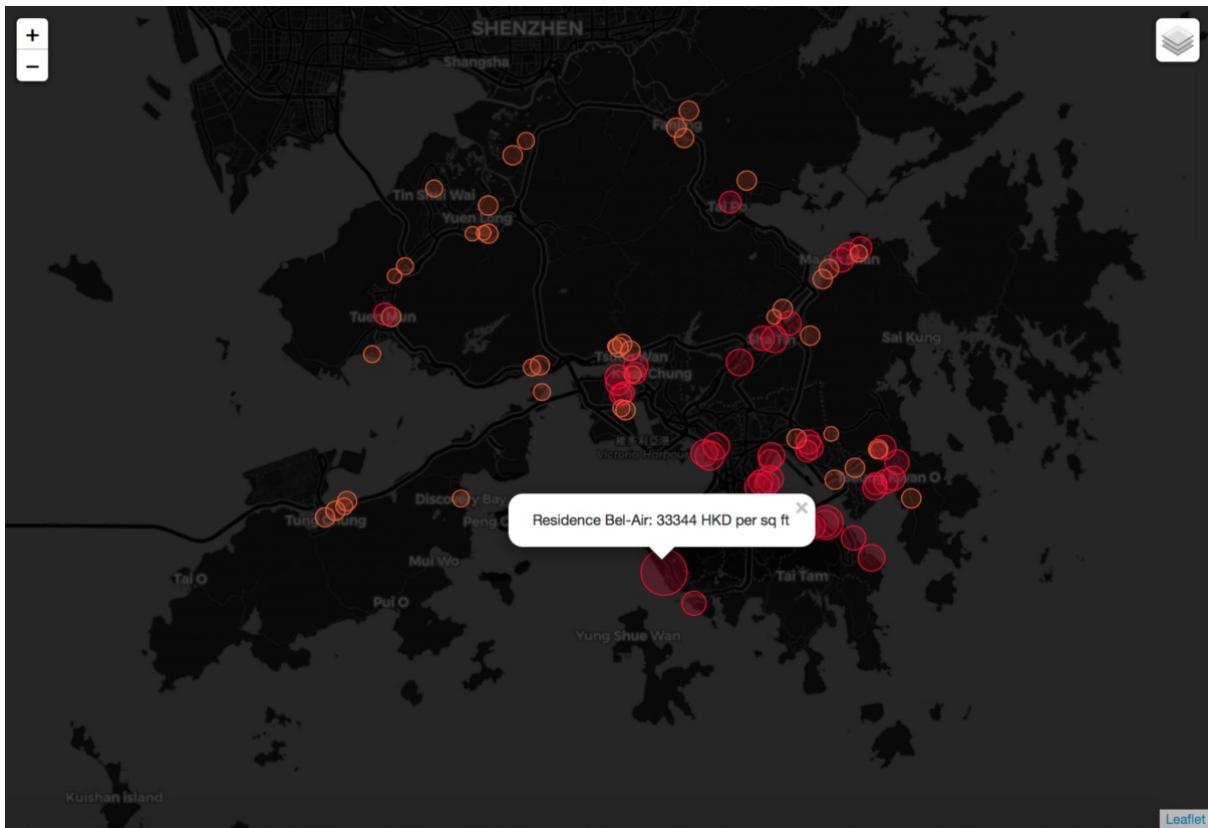
The tools and process:

- Get data from `OpenStreetMap` and provided by `osm-x-tractor`.
- Draw geo scatter plot via `matplotlib`

Data and codes can be found here: [England and Ireland seen from pub locations](#).

## Hong Kong property price bubble chart using folium

Visualizing various property prices in Hong Kong.



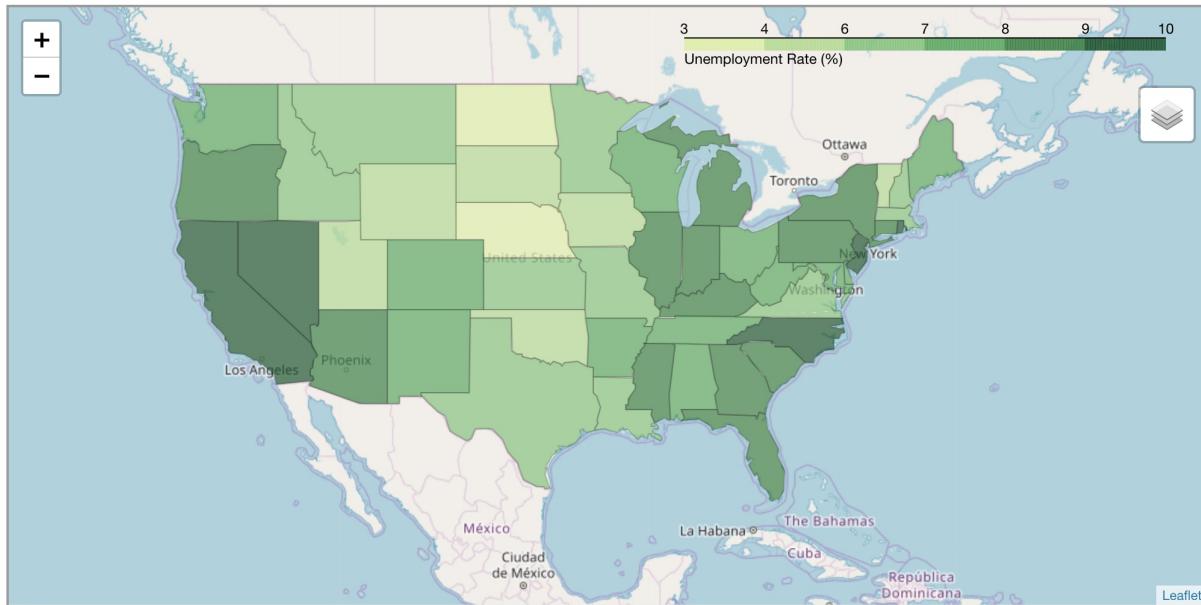
The tools and process:

- Dataset extracted from Midland Realty Property Price Chart with help of `pdf` to `csv` converter
- `overpy` for geocoding.
- Using `pandas` to combine the property names, prices, and coordinates into one huge dataframe for mapping.
- Drawing a map with `Folium`

Data and codes can be found here: [Visualising HK property prices](#).

## United States unemployment rate 2012 choropleth using folium

An example of a choropleth map made using the Folium library. This example comes directly from the documentation of this library, you can find more examples [here](#).



The tools and process:

- First get A shape file in the geojson format & A data frame that gives the values of each zone in your case
- Plot choropleth using `folium`.

Data and codes can be found here: [United States unemployment rate 2012 choropleth map](#).

## Bonus: Other GIS and mapping tools

### QGIS

<https://www.qgis.org/en/site/>

QGIS is written in Python. It provides a nice GUI so people without coding background can also use this tool. It integrates very well with Python. One can first try to process the geographical data via QGIS GUI. Once the prototyping is done, one can automate the workflow and take it to a massive scale using some glue code written in Python.

One major advantage of QGIS is being [FOSS](#).

### ArcGIS

<https://www.arcgis.com/index.html>

It is a high quality commercial GIS system.

### Carto

<https://carto.com/>

Very easy to use online tools. However, the free version limits number of POIs to 10,000. You may consider your data scale before using this tool.

### D3

D3 is a widely used data visualization library in Javascript. It provides convenience tools for the users to handle geo project and turn GeoJSON data into SVG `path` elements. The tools is highly flexible and favoured by many web designers.

Here is a case made by [D3](#)

## MapShaper

<http://mapshaper.org/>

MapShaper can help one to preview maps files and convert between different formats. It is also available as a command line tool. See it in action in the [geohk](#) project.

## Google Fusion Table

Being in the Google toolchain is a major advantage. However, this online tool requires KML format to plot a map. The current de facto standard GeoJSON is not supported as of this writing.

## Exercises

### Distances among cities

1. Calculate the "straight line" distance on earth surface from several source cities to Hong Kong. The source cities: New York, Vancouver, Stockholm, Buenos Aires, Perth. For each source city, print one line containing the name of the city and distance.
2. You can find "Great-circle distance" formula [here](#).
3. Use list and for loop to handle multiple cities.
4. Use function to increase the reusability.
5. Modules you need: `math`, you may need to use trigonometric functions.

**NOTE:** Our objective of the whole course is to get you onboard a new tool -- Python. You should use the tool but not be constrained by this tool. When you get stuck with a challenge, try to use your way, combining non-Python methods, to solve it and then iterate for better solution. For example, one key question for this exercise is to get the geo-locations of the cities in terms of longitudes and latitudes. Only with those coordinates, you can fit them into the great-circle distance formula. You can do this by searching Google, Google Map or Wikipedia as a start. After you have a basic version, try to think of automatic ways, in case there are a large number of interested cities in our real challenge, which makes the manual searching method infeasible.

### Extended exercise of geo distance

There is a package called `geopy`. It can automatically search the geo-locations in terms of longitude and latitude based on the location names. Further more, it can directly compute the distance between two geo-locations, without requiring to write the formula all by one's own.

# Week 16: High Dimensional Data

- Week 16: High Dimensional Data
  - Recap of data points and variables
  - Measure the similarity between data points
  - Dimensionality Reduction
    - Principal Component Analysis (PCA)
    - Locally Linear Embedding (LLE)
    - t-SNE
    - Graph Laplacian methods (Spectral Embedding)

## Recap of data points and variables

Two flavours of data:

- Social science research, e.g. survey data: many variable; a few data points
- Big data, e.g. user behaviour, server log: many data points; a few variables
  - Some may also get "many variables" but those many variables are usually of the same nature

## Measure the similarity between data points

When the data points come at low dimension, it is very easy to tell how similar they are. For persons A, B, C of height 170cm, 171cm, 192cm, we can easily tell that A and B are closer and C is a bit far away from the rest.

As to data points in the high dimensional space, the first task is to tell the similarity between the data points. For example, we can use two bag-of-words vectors (see text analysis) to represent two documents. Since there are many possible words in English, the two vectors are of very high dimension. Now we want to tell how similar one document is to another.

Following are some conventional similarity measures:

- inner product
- cosine
- Jaccard index
- correlation

Following are some conventional dissimilarity measures:

- Euclidean distance
- Hemming distance
- Editing distance

## Dimensionality Reduction

### Principal Component Analysis (PCA)

PCA is one of the earliest **linear** dimensionality reduction algorithm. After dimensionality reduction. One needs to check the "residual energy" to articulate the effectiveness of the algorithm. This is a natural requirement from the definition, where one alternative definition of PCA is "to capture the largest variance in the dataset" -- largest variance means "most energy".

The data-driven report from Initium Media on [5th term of Hong Kong Legislative Council Election](#) uses PCA to calculate the "political spectrum" of the Legco members. Note the "largest variance" in the mathematical formulation of PCA. That variance in Hong Kong turns out to be the political preference. The physical meaning of the principal axis comes as an aftermath when we check the members' standings on the axis. Mathematics only guarantees "largest variance"; it is our domain knowledge to find the "largest variance" coincides with political standing, in the case of HK Legco.

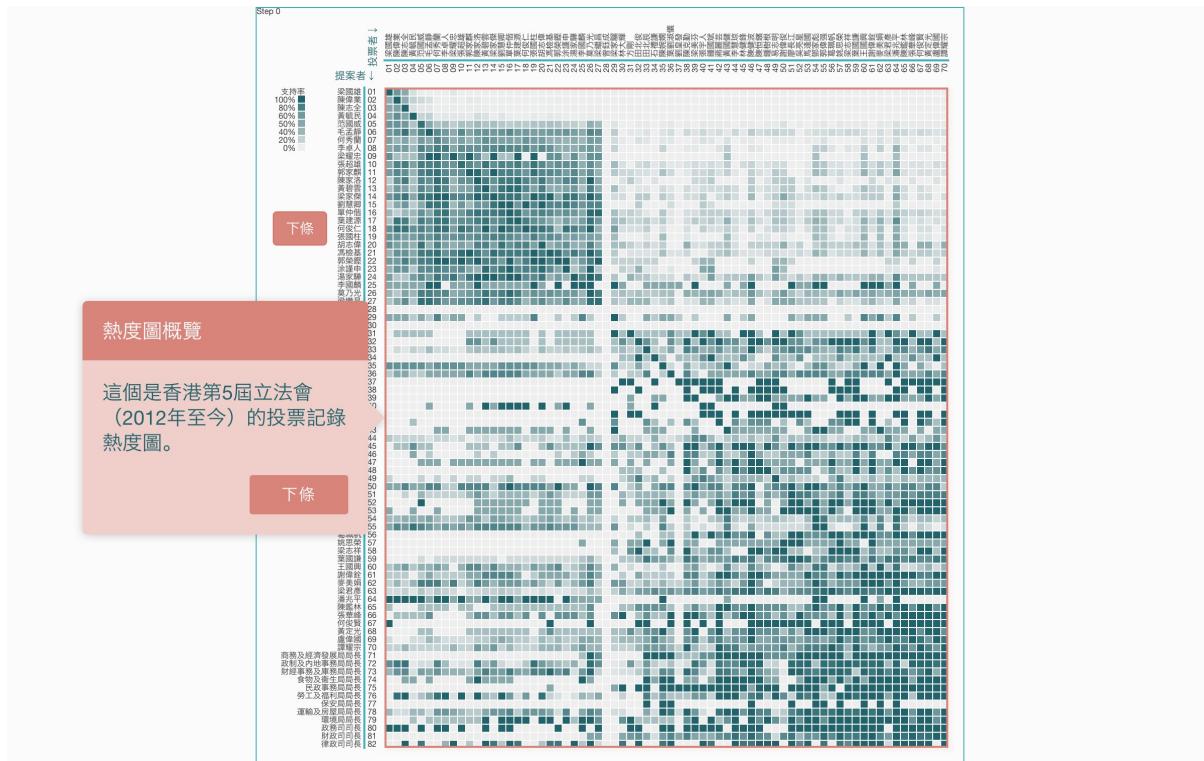


Image from [HK Legco interactive heatmap](#).

In the above heat map, the heatmap part is simple, how to order the Legco member is a hard problem and PCA was used. The core code is as follows:

```
from sklearn.decomposition import PCA
pca = PCA(n_components=1)
X is of dimensions: n_samples x n_features
X_reduced = pca.fit_transform(X)
```

In `sklearn`, there are many "decomposition" methods. They have very similar interfaces. Just instantiate the object from a chosen model with some super parameters, e.g. how many components (dimensions) to keep, and then call `fit_transform()` function to get the reduced dimension results. Many dimensionality reduction methods involve `matrix decomposition` at a certain step. That is why we get the term "decomposition". It is used as a synonym for dimensionality reduction sometimes, because the decomposed version of the matrices are usually of lower dimension.

#### References:

- 肖超, May 12, 2018, 第六屆香港立法會投票記錄分析 (2016-2018) , [LINK](#)
- Initium Lab. "23萬投票紀錄 回顧第五屆香港立法會," July 30, 2016. [LINK](#)
- 胡辟礪, and 巢恬逸. "嶄新展示——立法會數據一目了然." In 獨家新聞解碼2. 天地圖書, 2017. [LINK](#)

## Locally Linear Embedding (LLE)

LLE is one of the early non-linear dimensionality reduction algorithm. It captures the intrinsic structure on the low dimension manifold. The selection of super parameter is crucial to the sucessful dimensionality reduction result. The neighbourhood size can not be too large or too small.

The core codes are as follows:

```
embedding = LocallyLinearEmbedding(n_components=2)
X_transformed = embedding.fit_transform(X[:100])
```

References:

- [Official doc on sklearn-LLE](#)

## t-SNE

[t-Distributed Stochastic Neighbor Embedding \(t-SNE\)](#) is a popular non-linear dimensionality reduction algorithm in recent years. The interface is very like PCA. Core code:

```
x = np.array(A)
X_embedded = TSNE(n_components=2).fit_transform(X)
```

Being non-linear allows t-SNE be able to capture the intrinsic structure. Following is the comparison between different methods. The "S" shape plots data points in original space. On the right hand side are results from several manifold learning algorithms.

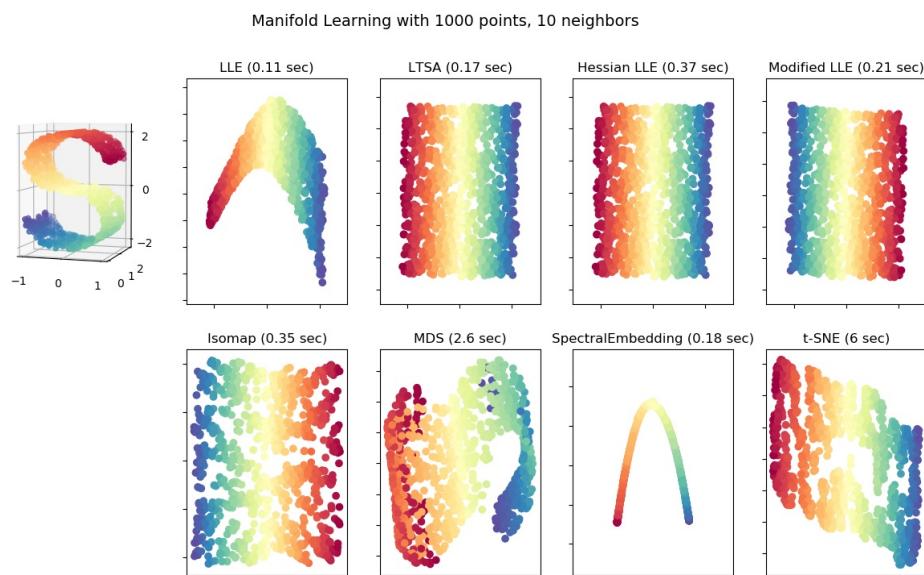


Image from [sklearn comparison of manifold learning methods](#)

References:

- S2018 students, COMM7780/ JOUR7280, An analysis of recipes from Xia Chu Fang website. [LINK](#)

## Graph Laplacian methods (Spectral Embedding)

Reference:

- Pili Hu, 2014, Spectral Clustering, <https://github.com/hupili/tutorial/tree/master/spectral-clustering>



# **Week 17: Machine learning primer: clustering, classification, regression**

- [Week 17: Machine learning primer: clustering, classification, regression](#week-17-machine-learning-primer-clustering-classification-regression) - [A few interwind concepts: data mining, machine learning, artificial intelligence](#a-few-interwind-concepts-data-mining-machine-learning-artificial-intelligence) - [Clustering](#clustering) - [Classification](#classification) - [Regression](#regression) - [Bonus: Deep learning](#bonus-deep-learning)

## **A few interwind concepts: data mining, machine learning, artificial intelligence**

### **Clustering**

### **Classification**

### **Regression**

### **Bonus: Deep learning**

# Week 18: Python Engineering and Data Engineering

Best practices collected for your reference.

## Python Engineering

*This section is under development and it is optional.* Interested readers can do some self study based the following pointers.

### Write code in professional style

"Style Guide" is a kind of popular documentation in programming world. There are usually multiple ways to do one thing in programming world but some ways are apparently better. Writing code in a common style helps the readers to quickly understand the content and is less error prone. Python's traditional philosophy is that *there is one and only one way to do the right thing*.

Python community has this famous style guide called [PEP8](#). You can automate this process by using the `pep8` executable (can be installed via `pip`). Another common reference is the [Google Python style guide](#)

### A word on syntactical sugar

Avoid syntactical sugars like

```
b = a if a > 0 else 0
```

The full and "stupid" format is favoured for beginners. Consider the following style:

```
if a > 0:
 b = a
else:
 b = 0
```

The readability of code is very important. It is a good practice to create less barrier for future readers, of which one may be yourself.

### File structure for a project

TODO

### Object Oriented Programming

TODO

### Top-down or Bottom-up?

As a beginner, you are suggested to adopt a bottom-up approach and Python's flexibility/ simplicity is well suited for this approach.

When you become more senior and when you gradually take up architectural roles, you will more likely to work using a top-down approach. Common tools are like concept design, flow chart, sequence diagram, Object Oriented Design (OOD) and E-R graph.

## Inside a Python module

TODO

## Efficiency/ complexity

The computer science language of talking "efficiency" is "complexity". It usually refers to how the time needed to execute a program increase with the scale of problem size. It is a grand topic in algorithm design to talk about complexity. In our discussion, you may want to notice that the choice of data structure matters because their complexity is different. You don't have to master this from very beginning. Having this concept in mind allows you to sense the problem when it comes. Then you can more effectively ask for help.

Here is [one example](#) to show you that `set` and `list` have drastically different efficiency on solving one problem.  
(You will learn jupyter notebook in [notes-week-01.md](#))

# Data Engineering

## Data pipelining

TODO: The fish bone structure in organising data pipeline

## Data versioning

TODO:

- Keep data, code and article at one place.

## Data wrangling journal

TODO:

- automate as many as you can
- take note of manual processing part

## From backend to frontend

TODO: Data is the interface

## Course Admin

### **Q: Is it mandatory to own a MAC Notebook in order to attend this course?**

A: No. Some of our TAs and class helpers use Windows and they can help you to setup the environment. There will be some difference between the two operating systems but once the environment is setup, most coding part will be the same. Besides, our classroom/ lab (CVA517 for F19 sessions) has MAC computers for you to use.

# Grading scheme

Last updated: 20181127

Basic grades:

- 3x assignments:
  - **10%** 1x bridging exercise
  - **15%** 1x data scraping
  - **15%** 1x data analysis/ visualisation/ presentation -- based on the data above
- 0x Quiz (or mid-term)
- **60%** 1x Group-based final project

Bonus grades:

- Bonus part in the assignments
- Class participation
- Online participation: GitHub/ Slack
- Notable side project (welcome to bounce ideas)

## Notes to bonus and the overall workflow

Bonus is meant to be bonus -- only benefit those who do the part but does not drag down other's grades.

After collecting the weighted grades, a curving processing will be applied -- mapping numbers grades to letter grades. Once the curving is done, those who have collected bonus points via the above listed channels, will have upgraded letter grades. In the curving stage, A is controlled to be **15% - 20%**. After adding bonus part, A-grade can be up to **40%**.

- [Guide for Contributor](#)
    - [Screenshot on MAC](#)
    - [Heading levels](#)
    - [Git commit message](#)
    - [File naming convention](#)
    - [Table of Contents \(TOC\)](#)
    - [Link whenever possible](#)
      - [Link to GitHub comments](#)
      - [Link to resources in this repo](#)
      - [Link to other repos](#)
    - [Editing environment](#)
    - [Coding style](#)
- 

## Guide for Contributor

### Screenshot on MAC

By default, we prefer taking screenshot of proper sized window. Please check out "How to take a screenshot of a window" of [this post](#).

### Heading levels

- `#` -- The main title of this page. It will show as `<h1>` in rendered format, or in the `<title>` of an HTML page's meta sections.
- `##` -- The first level of section heading.
- `###` -- The second level of section heading.

### Git commit message

Write a brief commit message so others can quickly comprehend the changes in this commit. If the commit message is complex, write a brief line within 70 characters and the remaining texts in the body after line break.

Bad example:

Update notes-week-00.md	Verified		9b30bed	
ChicoXYC committed 2 days ago				
Update notes-week-00.md	Verified		233ce1b	
ChicoXYC committed 2 days ago				
pics			0a2dfe5	
ChicoXYC committed 2 days ago				
basic introduction			98edf3d	
ChicoXYC committed 2 days ago				
Update notes-week-01.md	Verified		8f93cb3	
ChicoXYC committed 3 days ago				
Update notes-week-01.md	Verified		70eb8a3	
ChicoXYC committed 3 days ago				
Update first-question.md	Verified		555c6b8	
ChicoXYC committed 3 days ago				
Update first-question.md	Verified		f1b8ea3	
ChicoXYC committed 3 days ago				
Update python-2-vs-python-3.md	Verified		34fb021	
ChicoXYC committed 3 days ago				
Update notes-week-01.md	Verified		d8ede37	
ChicoXYC committed 3 days ago				
Update notes-week-01.md	Verified		b5286d7	
ChicoXYC committed 3 days ago				

Good example:

Commits on Jul 29, 2018			
Adk for chapter		1c472fd	
hupili committed 5 minutes ago			
Notes of troubleshooting help		a472503	
hupili committed 7 minutes ago			
improve template		a0458d1	
hupili committed 9 minutes ago			
Add issue template		1d583b4	
hupili committed 19 minutes ago			
Merge branch 'master' of github.com:hupili/python-for-data-and-media-...		9770edd	
hupili committed 38 minutes ago			
Add more API exercises		a38f421	
hupili committed an hour ago			
Switch order of 04 and 05; add notes		d439528	
hupili committed an hour ago			
heading levels		bc141d2	
hupili committed 2 hours ago			

## File naming convention

Adopt the [URL-slug](#) convention:

- All lowercase. Exception can be made when part of the phrase is an acronym.
- Replace blanks by dash, i.e. `-`.
- Put informative phrases into the path, so that the content/ meaning of the file can be quickly comprehended from its path/ URL.

- Only use alphabets, numbers, dash and dot (for extension name).

The way to check if a file path is URL-friendly or not is straightforward. Just copy and paste this path into your browser's address bar and see if it is changed to another string. For example, a single blank will be changed to %20 in the address bar. If it stays the same, we think it is URL-friendly.

Good example:

```
git-commit-message-good.png
```

Bad example 1: (blanks are not URL friendly)

```
git commit Message Good.png
```

Bad example 2: (not informative enough; may clash with other screenshots)

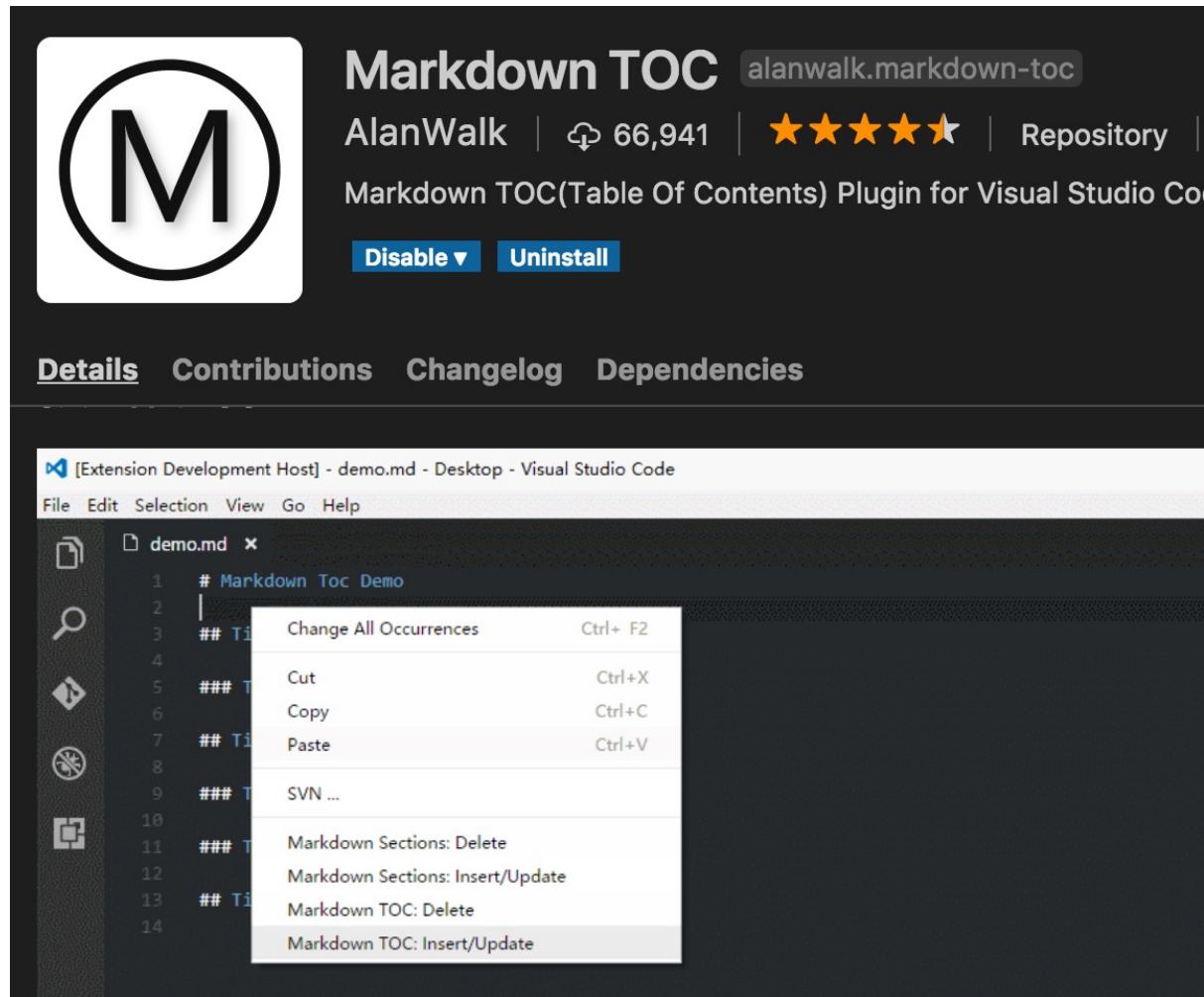
```
git.png
```

Bad example 3: (not informative enough; may clash with other screenshots)

```
commit.png
```

## Table of Contents (TOC)

In VSCode, install the following plugin and follow the menu to generate TOC for a markdown document.



The TOC generated by this plugin (Markdown TOC by AlanWalk) is automatically updated. Please do not touch the content in between

```
<!-- TOC -->
... some listing here
<!-- TOC -->
```

To help our future development, e.g. render the openbook via a static site generator, please wrap the TOC by another layer of `<div id="toc">`. The whole TOC code block looks like this:

```
<div id="toc">
<!-- TOC -->
... some listing here
<!-- TOC -->
</div>
```

Potential issues:

- Latest VSCode clashes with the plugin. Checkout [this issue](#) for solution. Changing `file.eol` in the settings can temporarily solve the issue.

## Link whenever possible

The beauty of Internet and one of its original design goal is to **connect everything**. URL/ URI provides a system for inter linking between documents. In our openbook, contributors try the best to link to relevant resources if those resources are public already. When making quote or taking screenshots, it is a convention to link to the original article/interface.

For example, when explaining how the discussion forum works on GitHub, one do not only have screenshots but also can experience the live version. This [Git diff](#) demonstrates how and [here](#) is the rendered version.

## Link to GitHub comments

Click the date and get a full URL including the [URI fragment](#). Read more on [Stack Overflow](#).

## Link to resources in this repo

Use relative links so that moving/ renaming this repo does not break the links. One example is the [README.md](#) file.

## Link to other repos

Other repos are usually out of your control. The GitHub web page by default show you the latest content. However, "the latest" will be changed later. To ensure future readers can see exactly what you see, you can use PermaLink to GitHub, i.e. a link that includes a commit hash (see more on [notes-week-00.md](#)), which will be resolved to exactly the same version no matter when the future visitors come. You can press `y` so the URL in the browser address bar is changed to PermaLink. See [official explanation](#).

## Editing environment

We suggest to use [VSCode](#) with the following extensions:

- `Markdown Preview Enhanced` -- See how it works [here](#)
- `markdownlint` -- See how it works [here](#)
- `Markdown TOC` by AlanWalk -- See more instructions in the current document, [this section](#)

## Coding style

It is hard to exhaust coding style issues. We collect some sample discussions here and conclude later if necessary:

- [Commit on plotly sample](#)

# Project guidelines

Key dates:

- **Dec 4 (Tue)**: Final presentation for JOUR7280
- **Dec 7 (Fri)**: Final presentation for COMM7780
- **Dec 16 (Sun)**: Final submission (report/ dataset/ codes/ aux files) due for JOUR7280/ COMM7780, i.e. finalise your GitHub repo by this date

## Final Presentation Guideline

Every group has up to 20 min including Q/A time. It is suggested to control your presentation time to **~12 min**, up to **15 min**. 15 min will be a hard cut. We will stop you immediately at 15 min no matter how many slides are left. The presentation is rated by the lecturer, as well as invited guests from the industry. The formula of a good presentation:

- It usually catches people's attention very quickly and then goes through the main story points without messing the audience with too many technical details.
- People then raise various questions, related to the presentation but may well beyond the presentation itself. Since you have studied the topic in depth via the final project, you have the best expertise to answer those questions. You can prepare some "backup slides", so when people ask, you can jump there.

The rule of thumb: when people challenge you, they are actually interested to know more. The more attention they devote to your project, the deeper impression they will have about your project. Don't feel bad when people ask questions. Try to make them excited by your inspiring and insightful answers.

## Final Submission Guideline

Technical requirements:

- Create a **repository dedicated for the project** and keep updating. You can choose to put the repo under an individual user, or more preferably you create [an organisation](#) for your final project. We advocate using GitHub as a **collaboration platform**, instead of merely a place to publish your works. [GitHub Desktop](#) is very convenient for you to synchronize files between the group members, or between your multiple devices.
- Include **at least one** Jupyter notebook (`.ipynb`) in the repo, showing how you reach all the story points and conclusions. That is, the notebook needs to be **reproducible**. When there is gap, e.g. timing of execution/ login credentials, please leave enough notes, so that others are able to execute your script with the help of your notes.
  - You are welcome to leave more Jupyter notebooks in the repo when applicable. Usually, we organise our data pipeline into several stages and use one notebook for one stage. There are usually three stages: 1) data collection; 2) data analysis & visualisation; 3) data presentation. In this way, when people try to reproduce the analysis part, they don't have to spend time re-executing the data collection script. When people first come to your project, they can start with the presentation notebook, which loads intermediate data files produced by upstream. If they find one part interesting, they may dig out the processing details from upstream.
- Put all data files, photos, videos, documents and slides related with your final project into the repo. If the files are too large (> 100MB), find us to discuss case by case.
- Put a `README.md` at the root of the repo, which includes the following information:
  - Title
  - Team members
  - Background and motivation

- (optional) Executive summary
- Quick pointers to key files
- References

Data requirements:

- **Original data:** The project needs to have at least one original dataset, in the form of *raw dataset*, not summarised statistics. The volume of this dataset is no less than that of Assignment 1.
- You can use existing public datasets to assist your data-driven report. That may come in the form of downloadable data files; or available via API. Please cite the source properly.

You can refer to [the archive of past projects](#) to get some ideas. Note that, not all the projects there conform to our current standard. The requirements are evolving, so please observe the above closely. Feel free to ask whenever in doubt.

## Content guideline: JOUR7280 -- Data Journalism

The final output is a **complete story** powered by data. Depending on the group expertise, you can emphasize more on data, or more on story. Besides using data, you can leverage conventional techniques from journalism production, e.g. interview, photo, video. We adopt the broad sense of "newsworthiness" -- new knowledge on old topic also counts. Of course, if the project can tackle a question with social impact, that is better. The best one starts from a current affair, scrape/ find relevant datasets, and mine the data to get new insights not yet told/ not well told in the public. Introducing diversified perspectives are always highly appreciated. Constructive discussions, either from the group or from interviewees, are also highly appreciated. Two past references suitable for JOUR7280 are [AirCrash](#) (more data) and [Syria War](#) (more story).

## Content guideline: COMM7780 -- Data-driven report on communication issues

Being data-driven should be easy for most students who have taken the course. Finding questions concerned in communication field and use data-driven insights to answer them could be hard. Following are potential ideas and treatment related to communication, for your reference:

- Study brands' outlook on rating sites. Ref. [Fastfood on Dianping](#).
- Analyse current trend in Internet product, and devise content strategies.
- Auto media monitoring, e.g. keywords alert on Twitter/ Weibo; sentiment analysis.
- Study career prospect for communication students. Ref. [HKBU MA career study](#)
- User profiling for a website/ product/ service.
- Content profiling for a website/ product/ service. Ref. [Recipe on Xiachufang](#)
- Build a chatbot to conduct Customer Relationship Management (CRM). Ref. [Twitter earthquake bot](#)
- Analyse network structure to get insights of community operation. Ref. [wechat ego-net](#)
- Build a categorized journalist's catalogue for PR specialist. Ref. [global data journalist directory](#), pointers from [this post](#)
- Conduct SEO auditing. Ref. [SEO auditing exercise](#)
- Market analysis that supports a startup idea.

We will keep the list growing when more ideas and references are available. You are suggested to discuss with the instructor as early as possible, to help evaluate the relevance and viability. The final deliverable could be a report like industrial research report, e.g. [Reuters X Oxford digital media report](#). However, you are not limited to this form. A product document, planning document, or pitch deck/ business proposals are all valid. You can even build a web-

based data query service if you'd like to. Note that, one important factor is **communication perspective**. A report that purely enumerates data/ statistics/ charts and describes only what they are, will not meet the requirement. You need to extract communication insights or make derivative works that are relevant to a communication issue.

## Bonus: Release and promote your work on mobile web

NOTE: Doing bonus part will not hard other's performance; However, you may get upgrade once the curving on basic part is done.

Mobile web has been the major channel for people to access and read content today. Although GitHub ecosystem and Jupyter notebook are convenient for us professionals to share among each other, it is hard for normal readers to access. To help your work to reach a wider audience and create an impact, you may want to release it on mobile web. We have prepared a 10-minutes template intended for people with no background: <http://project.hupili.net/big-road/>

You can make a landing page, or put your entire story on this web page. It is subject to your preference and your understanding of the potential audience. In the end of the web (e.g. footer), please include the following information:

- Course code/ title
- Project team
- Link to your GitHub repo -- so they can read more or try to reproduce the report
- Link to the [open book](#) -- so they can also learn and jump into this field

Note following two technical details if you want to participate in the bonus competition:

- There are several `<script>` tags at the end of the page that includes a Google Analytics code.
- Remember to modify `<title>` tag in `<head>` region. Or, the traffic may be incorrectly recorded elsewhere.

We will measure the impact of your project using Google Analytics data and assign bonus grades.

# Setup Python Environment on Windows and MAC

## MAC

Modern Mac OS versions come with Python 2.7.x installed (or Python 2.6.1 if an older Mac OS X version), but with several reasons, many Python users may need to update Python in Mac OS to a newer version like Python 3.

- You can type `python --version` in terminal to check out the version on your computer.

### How to Install the Updated Python 3 in Mac OS X

1. by using the Python package installer from [python.org](http://python.org)
  - Go to Python.org and download the [latest Python installer package](#)
  - Run the Python installer package and install Python 3 onto the Mac

Once Python 3 is installed you will find a Python3 folder within the /Applications directory of your Mac.
2. Install Python 3 with Homebrew
  - [Homebrew](#) is a commonly used package manager for MAC software. The usual way to install package by homebrew is `brew install {package}`
  - Reference: [MAC OSX 正確地同時安裝 PYTHON 2.7 和 PYTHON3](#)

## Windows

The Unix/ Linux world has a collection tools that nicely work together. However, you will need to troubleshoot environment frequently on Windows. I suggest you to use MAC or rely on our lab for this semester, to reduce such hassle. Once you master the essence, it is easier to move to other environments later. I have not used Windows for a long time. There are problems stitching things together. For those who have tried/ are trying Windows, please share your experience, no matter it is successful case or not. Here are some pointers to get started.

- Windows 10 users can check out <https://www.windowscentral.com/how-install-bash-shell-command-line-windows-10>. The system has native support but in beta stage.
- Users of other versions can checkout <https://www.cygwin.com/>. Cygwin is a classical project to emulate Linux environment on Windows. It should still work on most versions.
- Some students who learned Python before told me the Python installer on Windows is shipped with a shell by default. You may try this route.
- <http://gitforwindows.org/> Git for Windows ships with a shell by default. You can key in most Linux-like commands there.
- <https://pythonhosted.org/spyder/> and <https://conda.io/docs/user-guide/install/windows.html> . Those are two integrated Python environments. It saves time installing dependencies.
- Another shell like environment you can try on Windows: <https://github.com/bmatzelle/gow/wiki> . GOW is the rising star alternative to Cygwin.

### Install Python3 on Windows and Set Environment

This question is usually accompanied by another problem/error:

不是内部或外部命令，也不是可运行的程序或批处理文件  
or  
'python' is not recognized as an internal or external command

Details please see [here](#).

## Other integrated environments you can consider

There exists some integrated environments that get you the full dependencies in one shot. You don't have to worry about the environment as a beginner. Those solutions are very easy to try so we do not repeat their own documentations here. Please find some choices as follows:

- Anaconda: <https://conda.io/docs/user-guide/install/download.html>
- Spyder: <https://pythonhosted.org/spyder/installation.html>

### Difference between conda and pip

`conda` and `pip` are both package managers. `pip` is the default package manager shipped with Python core. `conda` is the package manager for "Anaconda" distribution, powered by Continuum Analytics, a commercial company. Find many insightful discussions on [StackOverflow thread](#) and also [misconceptions of Anaconda](#) summarised by Continuum Analytics itself. Here are some highlights:

- For beginners or Windows users, if you have trouble trying out other solutions, Anaconda distribution may be more direct. It saves the bootstrapping time.
- If possible, we suggest to stick with `pip`, or its offsprings because of the support of Python core community. Note that `easy_install` was once a mainstream solution before `pip` took off. Python community is always evolving. The latest packaging solution is `Pipfile` and `pipenv`. Since it is not as stable as `pip` yet, we will leave this to self testing for those who are interested.
- `conda` had the advantages of 1) able to install binary packages (pre-compiled packages) and 2) install non-Python packages. The first advantage was once prevailing, especially on Windows because Windows usually lack the dependencies to compile the packages from source codes. It is less significant now, since latest `pip` also supports to install binaries. Another advancement is `pipenv` who highlighted in the project pitch text that *Windows is a first-class citizen, in our world.*

# Shell

- [Shell](#)
  - [Permission denied](#)
  - [Command not found](#)
  - [Difference between Terminal and Shell](#)
  - [Different shell commands in Win and Mac](#)
  - [Verify your current working folder](#)
  - [Check the information in terminal](#)

## Permission denied

Sometimes a command that is not existing will induce this error.

For example, I have a variable `abc = 3` and I am trying to write it to a file (which doesn't exist and I assume python will create it on its own), then PERMISSION DENIED will present in your shell.

## Command not found

The first hurdle is that we need to tell the shell where to find the program. If we don't put any directory indication, computer can only run executable files located in the executable search path described by the `PATH` environment variables. The current directory may not in the search path unless you put it there.

## Difference between Terminal and Shell

Terminal is the (graphical) "window" you can see on your computer. It is the bridge between the user and shell. Shell is the tool that gets input from user (via Terminal), executes programs and sends output to the Terminal (for user consumption). Shell itself is a program and is usually shipped with the operating system by default, especially in the [UNIX-like](#) families (including MAX OS X).

## Different shell commands in Win and Mac

please see [here](#)

## Verify your current working folder

Most of the path related errors are caused by wrong current working folder. One need to run the following two commands before starting other commands to verify the current working folder:

- `pwd` -- prints the working directory. You can check the path name.
- `ls` -- list directory. This shows the files in the current directory for your double check.

## Check the information in terminal

1. `which pip3` to learn where is pip3 in your computer. You can also input `which pip` to know the location of pip.

You can also drag the pip file and pip3 file into the Visual Studio Code, after you use `which` to know their location.

1. `cat pip` to check the content.

# Python Language Basics

- [Python Language Basics](#python-language-basics) - [Syntax](#syntax) - [Blanks](#blanks) - [Double quotes and single quotes](#double-quotes-and-single-quotes) - [\[ \] does not work to specify precedence in formula](#\[ \]-does-not-work-to-specify-precedence-in-formula) - [Multiplication operator `\*` can not be omitted in formula](#multiplication-operator--can-not-be-omitted-in-formula) - [Letter case difference](#letter-case-difference) - [List range pattern](#list-range-pattern) - [== and =](#and-) - [Append VS Extend](#append-vs-extend) - [For loop range](#for-loop-range) - [Name clash with reserved word](#name-clash-with-reserved-word)

## Syntax

### Blanks

code "print ()", then execute

### Double quotes and single quotes

Double quotes equal to single quotes

### \[ \] does not work to specify precedence in formula

() is used to specify precedence in formula rather than []

For example, ((1+r)\*n-1) can be executed, but [(1+r)\*n-1] can not be executed.

[] have other function, like range\[1:50\]

### Multiplication operator \* can not be omitted in formula

Take an example:

when we exercise the mortgage calculator: A=P\*r\*(1+r)  
is different from A=Pr\((1+r)\), the latter one can not be executed.

### Letter case difference

```
p = 1
print(P)
```

You will see the error message NameError: name 'P' is not defined . That is because Python is **case-sensitive**. The lowercase p and uppercase P mean different variables.

### List range pattern

in range (a, b) , it's close to the range in the left, open the range in the right.

```
for i in range(1, 10):
 print (i)
```

You will see the outcome: 1,2,3,4,5,6,7,8,9 without 10, because the left range is included, while the right range is excluded.

## **== and =**

In python, notation `==` represents making a judgment / comparison. Notation `=` means assign values.

For example:

```
df['location'] == '旺角'
#this means Iterate all elements in location column to compare whether its equal to 旺角
df['location'] = '旺角'
#this means assign 旺角 to every element in the column location
```

## **Append VS Extend**

when a list of elements `append` to another list, the list passing into the function `()` will be treated as an whole element.

```
x = ['a', 'b']
y = [0, 2, 4, 8]
x.append(y)
x
```

Output:

```
['a', 'b', [0, 2, 4, 8]]
```

`extend` method extract every single items of one list and add those items one by one into the new list.

```
x = ['a', 'b']
y = [0, 2, 4, 8]
x.extend(y)
x
```

Output:

```
['a', 'b', 0, 2, 4, 8]
```

## **For loop range**

Function 'Range' has 3 parameters. From XX to XX with the step size XX. The 3rd parameter is the step size. If you don't input the 3rd one, it will take 1 in default.

```
for i in range(5,10):
 print(i)
#5
#6
#7
#8
#9
```

```
for i in range(5,10,2):
 print(i)
#5
```

```
#7
#9
```

The parameters can be negative. At the same time, you need adjust the positions of the first and second parameters.

```
for i in range(5,10,-1):
 print(i)
#10
#9
#8
#7
#6
```

## Name clash with reserved word

We find the following error very common when students start to work on bigger projects, especially in Jupyter notebook.

```
TypeError: 'str' object is not callable
```

or

```
TypeError: 'list' object is not callable
```

In the first example, you are likely to get following result when checking the type of `str`:

```
type(str)
str
```

This is an indicator that `str` is reassigned to some other value before. Check if you have `str=xxx` statement earlier on. `str` is a built-in function. The variable assignment changes it to another object. That is why Python complains "str' object is not callable".

The same issue goes with `list`. Students may use `list = []` and then `list.append(xxx)` later to collect data entries. The name is intuitive but the assignment pollutes the built-in function.

Even if you change that line of assignment, Python still gives the same complaint. The reason is that Jupyter notebook has a continuously running Python engine (called "kernel") in the back. The assignment is already effective and will keep being effective in the whole session. You need to "restart kernel" after fixing the code.

One needs to keep off reserved word like `list`, `dict`, `str`, etc. Except for that, you also want to keep away from the modules/ classes you import. One useful trick for beginner is to add `my` to your own variables when you are not sure if the variable name might clash with existing object or not.

Related issues:

- [#97](#)

# FAQ: Python 2 and Python 3

## Difference between python 2 and 3

Python 2 and Python 3 are mostly the same during our discussions. However there are some differences if you explore further. During this semester, we base our discussions and exercises on Python 3. We highlight some main differences here for your reference. To read further, you can click [here](#)

### syntax of print

Python 2's print statement has been replaced by the `print()` function, meaning that we have to wrap the object that we want to print in parentheses. Python 2 doesn't have a problem with additional parentheses, but in contrast, Python 3 would raise a `SyntaxError` if we called the print function the Python 2-way without the parentheses. However, if we have multiple objects inside the parentheses, we will create a tuple, since `print` is a "statement" in Python 2, not a function call.

### str, bytes, encoding and decoding

Python 2 has ASCII `str()` types, separate `unicode()`, but no `byte` type. Now, in Python 3, we finally have Unicode (utf-8) `str`, and 2 byte classes: `byte` and `bytearray`.

Sometimes you need to convert between `bytes` and `str` (unicode string). You can use `bytes.decode()` or `str.encode()` for the conversion. Read more on the [official unicode support doc](#).

### Division

Python 2 interprets `/` as integer division, whereas Python 3 interprets it as decimal division. Here's the mapping:

	<b>Python 2</b>	<b>Python 3</b>
Floating point division	<code>float(a)/b</code>	<code>a / b</code>
Integer division	<code>a / b</code>	<code>a // b</code>
Division for remainder	<code>%</code>	<code>%</code>

### Return iterable instead of list

When you read tutorials online, one frequent confusion is your code returns iterable but the sample code returns a list. You need to call `next()` on a iterable to find the next element. This is an efficiency oriented design in Python3. Given the small amount of data we are handling in this course, you can use the "brute force" way to convert valid Python2 code into Python3. The common trick is to wrap previous codes by `list()`. For example, if `a = map(...)` works in Python2, you simply write `a = list(map(...))` and that works in Python3. Having another layer of list won't harm (in terms of grammar correctness).

### Rounding to closest even number

Python 3 rounds to the closest even number. This causes little difference in most of your program.

```
%python3
>>> round(16.5)
16
```

```
>>> round(17.5)
18
>>> round(15.5)
16

%python2
>>> round(16.5)
17.0
>>> round(17.5)
18.0
>>> round(15.5)
16.0
```

## How to set up Python 2.7 and Python 3 at the same time on your Mac OSX?

You can follow the steps in this[Link](#) to set up both of them.

## Install modules in Python3

If you want to install modules for Python3, please use `pip3` (not `pip`). To install a module in user mode, you can use

```
pip3 install --user {name-of-the-module}
```

**NOTE:** Some online tutorials instruct you to install python module with `sudo` prefix, e.g. `sudo pip3 install pandas`. Don't do that unless other methods do not work. It is better practice to enter virtual env and run `pip install ...` (or `pip3 install`) in shell or `!pip install xxx` in notebook. In this way, the modules and other environment updates only happen within `venv`. Using `sudo xxx` is usually a quick solution but not recommended. `sudo` basically gives the followup command super user privilege which makes the system vulnerable to malicious software. One convention is to refrain from `sudo` whenever possible because -- with great power comes great responsibility -- `sudo` has too much power that is not always needed. If you are not in `virtualenv`, or if you want to install a module in larger scope, not only for the current project, you can use the `--user` mode as shown above. This installs the module in a sub folder under your user home directory (`echo $HOME`).

# Dataprep

Dataprep is short for "Data Preparation" it includes the collection and pre-processing stages of a conventional data mining pipeline. The upstream of Dataprep are multiple, probably unstructured and/ or heterogeneous, data sources. The downstream of Dataprep is data processing/ mining modules that take structured data as input and produce visuals, insights or intermediate datasets for further analysis.

Dataprep generally involves data cleaning, normalisation and joining. From a DB perspective, it is like series of ETL process. As DMaaS (Data Mining as a Service) and AlaaS (Artificial Intelligence as a Service) become widely available and accurate enough for practical applications, the difficulty now lies in how to prepare the dataset that is suitable for those algorithms. Besides writing your own scripts, following may be some useful pointers:

- Google Cloud Dataprep, released in 2017: <https://cloud.google.com/dataprep/>
- Tableau, in its recent "BI trends 2018", implied as 11th trend that it would enrich/ extend Tableau with Dataprep capability: <https://www.tableau.com/reports/business-intelligence-trends>
- Open Refine, a classical data cleaning tool, can well address part of Dataprep problems: <http://openrefine.org/>

The [Quartz bad data guide](#) is a well developed guide for people who starts to handle data sourcing and cleaning.

## Pro Tips

Good use of hotkeys and shorthands can boost your efficiency.

### Asking questions like a Pro

Asking questions like a Pro is the first step to boost your journey in programming world. As long as you adopt the best practice in asking questions, people can easily help you. Developers in the open world like challenges and never mind your silly questions. Here are some tips to help to ask questions like a Pro:

- Clearly state what you want to achieve. Text and images can help.
- Show people what you have tried so far. It is good to let them know what are already known to not work.
- Make a minimum reproducible example of the issue. This helps people focus on the technical problem, instead of your business logics.

In our Python case, it is suggested to make a Jupyter notebook to put the above information in one place. Besides showing the GitHub URL to your notebook, it is even better to show the [NBViewer preview link](#). That helps people to quickly understand the issue. Note, dynamic chart (via Javascript) is not shown on GitHub due to security reasons.

Here are some hand-picked issues for your reference:

- [#90](#) is a very good demo of effectively asking questions.
- [This comment](#) shows a real minimum reproducible example. It is simpler than the OP. In the open source world, people who come up with minimum reproducible example are highly appreciated. They are the stepping stones for those who eventually solve the issue. Issue [#104](#) is also a complete demo how people react to issues in open source world. People with enough time can checkout [troubleshooting comment](#) and [problem summary comment](#); People who are hurrying to solution can checkout [solution 1](#) and [solution 2](#).

## Shell

### auto completion

Use "Tab" button. For example, type "python ev", then use "Tab", you can see the "python evening.py" (assume the file name is evening)

### find last command

Use upward arrow key.

Try `control+r` to conduct a search in command history.

## Visual Studio Code (VSCode)

### batch comment/ uncomment codes

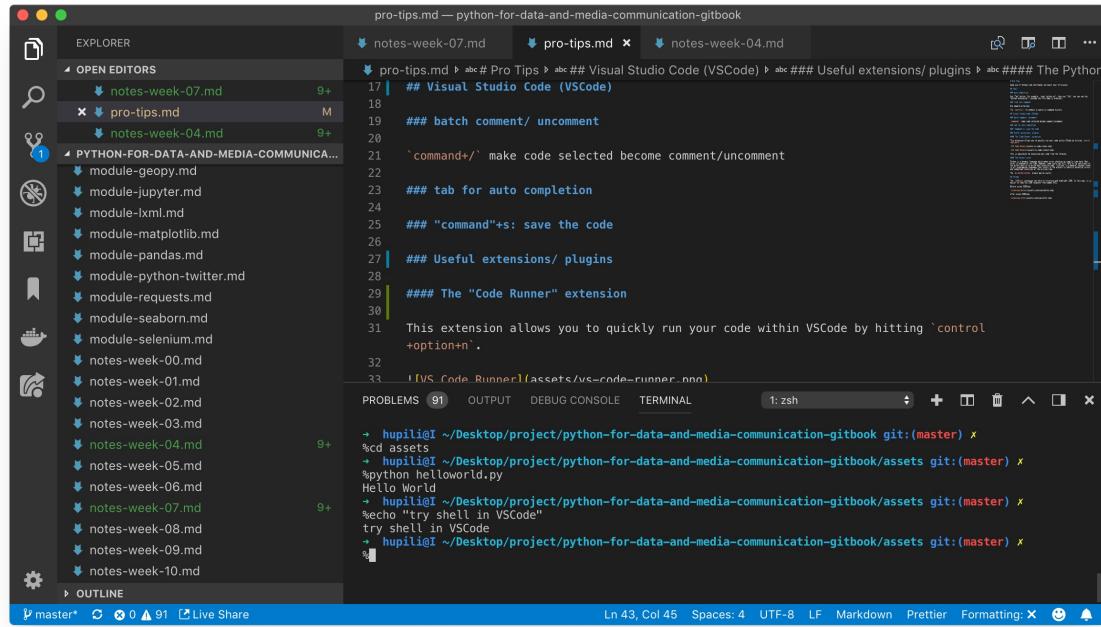
`command + /` make code selected become comment/ uncomment. This hotkey can give proper comment blocks according to the current language identified by VSCode. For example, Python scripts will see `#` prefixed lines as comment blocks. markdown/ HTML files will see `<!-- comment -->` style blocks.

## tab for auto completion

"command"+s: save the code

## Run shell commands inside VSCode window

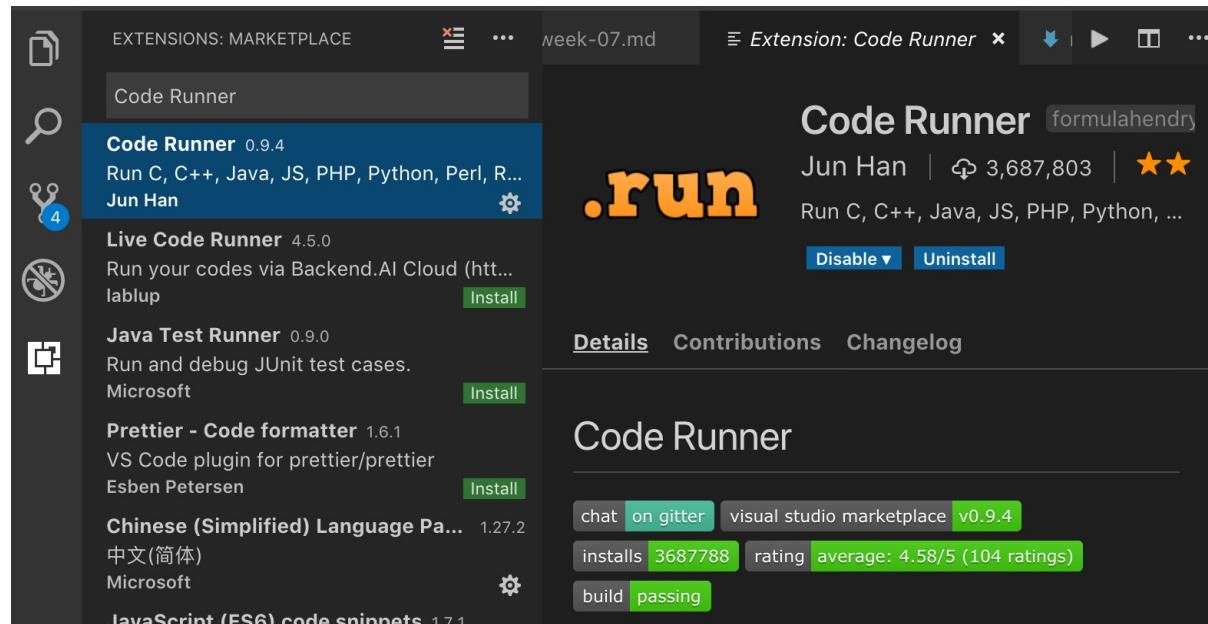
You don't have to leave VSCode in order to run Python scripts.



## Useful extensions/ plugins

### The "Code Runner" extension

This extension allows you to quickly run your code within VSCode by hitting `control+option+n`.



```

1 Fixed_Cost = 30000
2 Content_Cost = 70000
3
4 num = float(input('please input your estimate number of subscribers:'))
5
6 for i in range(0,int(num)):
7
8 Revenue = (1*i) + (0.1*15*i)
9
10 if i < 50000:
11 Total_Cost = Fixed_Cost + Content_Cost
12 else:
13 Total_Cost = Fixed_Cost + Content_Cost + 0.1 * (i - 50000)
14 Net_Income = Revenue - Total_Cost
15 if Net_Income >= 0:
16 print('subscribers=', i)
17 break
18
19 if Net_Income < 0:
20 print('Net_Income=', Net_Income)
21

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

bash: /anaconda3/etc/profile.d/conda.sh: No such file or directory
Chicoxyc:~$ python -u "/Users/xuyucan/Desktop/test.py"
please input your estimate number of subscribers:40000
('Net_Income', -2.5)
Chicoxyc:~$ █

```

Ln 21, Col 1 Tab Size: 4 UTF-8 LF Python 🎉 🔔

This is equivalent by executing your code from the Terminal.

## The Python linter

Python is a dynamic language which makes error checking at compile time hard. Most error is exposed at run time. However, some tools can still help us to catch most of the errors and try to write best practice code. "Linting" is a general concept found in all programming languages that refers to the process to identify potential errors and suboptimal practices at the writing time.

The `ms-python.python` plugin may be useful.

## Chrome

The `JSONView` extension can help to structure and highlight JSON. In this way, it is easier to read the JSON response from common APIs.

Before using JSONView:

```
{
 "count": 20,
 "start": 0,
 "total": 23,
 "subjects": [
 {
 "rating": {
 "max": 10,
 "average": 8.1,
 "stars": "40",
 "min": 0
 },
 "genres": [
 "动作",
 "犯罪"
],
 "title": "无双",
 "casts": [
 {
 "alt": "https://movie.douban.com/celebrity/1044899/",
 "avatars": {
 "small": "https://img3.doubanio.com/view/celebrity/s_ratio_celebrity/public/p205.webp",
 "large": "https://img3.doubanio.com/view/celebrity/s_ratio_celebrity/public/p205.webp",
 "medium": "https://img3.doubanio.com/view/celebrity/s_ratio_celebrity/public/p205.webp"
 },
 "name": "周润发",
 "id": "1044899"
 },
 {
 "alt": "https://movie.douban.com/celebrity/1041390/",
 "avatars": {
 "small": "https://img3.doubanio.com/view/celebrity/s_ratio_celebrity/public/p49475.webp",
 "large": "https://img3.doubanio.com/view/celebrity/s_ratio_celebrity/public/p49475.webp",
 "medium": "https://img3.doubanio.com/view/celebrity/s_ratio_celebrity/public/p49475.webp"
 },
 "name": "郭富城",
 "id": "1041390"
 }
]
 }
]
}
```

After using JSONView:

```
{
 count: 20,
 start: 0,
 total: 23,
 - subjects: [
 {
 - {
 - rating: {
 max: 10,
 average: 8.1,
 stars: "40",
 min: 0
 },
 - genres: [
 "动作",
 "犯罪"
],
 title: "无双",
 - casts: [
 - {
 alt: "https://movie.douban.com/celebrity/1044899/",
 - avatars: {
 small: "https://img3.doubanio.com/view/celebrity/s_ratio_celebrity/public/p205.webp",
 large: "https://img3.doubanio.com/view/celebrity/s_ratio_celebrity/public/p205.webp",
 medium: "https://img3.doubanio.com/view/celebrity/s_ratio_celebrity/public/p205.webp"
 },
 name: "周润发",
 id: "1044899"
 },
 - {
 alt: "https://movie.douban.com/celebrity/1041390/",
 - avatars: {
 small: "https://img3.doubanio.com/view/celebrity/s_ratio_celebrity/public/p49475.webp",
 large: "https://img3.doubanio.com/view/celebrity/s_ratio_celebrity/public/p49475.webp",
 medium: "https://img3.doubanio.com/view/celebrity/s_ratio_celebrity/public/p49475.webp"
 },
 name: "郭富城",
 id: "1041390"
 }
]
 }
 }
]
}
```

# Reading Materials

Here is a curated list of reading materials to help one reinforce the knowledge learned from the open book.

## Class references

- [Sample GitHub repo for homeworks](#)
- [Application of data skills on media and communication problems](#) -- contributed by spring 2017 students.

## Assistive tools

- [pythontutor](#) is a powerful tool to help one visually explore every step of Python interpreter. You can see current instruction pointer, variable values and function call stacks. This [Youtube video](#) shows you how it works.  
(Recommended by [Kevin Xu](#))

## Beginner exercises

DataCamp provides a good set of multiple choice questions to get you familiar with the language: [Link](#)

The screenshot shows a DataCamp course interface. At the top, there is a blue header bar with the DataCamp logo on the left and the course title "Introduction to Python for Data Science" on the right. Below the header, there is a horizontal progress bar consisting of several light blue segments. The main content area contains a question titled "Select the correct option". The question asks: "What is the output of the following code?" followed by a code snippet:

```
x = [11, 12, 13, 14]
y = x
y[2:4] = [15, 16]
print(x)
```

Below the code, there are four multiple-choice options, each preceded by a radio button:

- [11, 12, 13, 14] (selected)
- [11, 12, 15, 16]
- [15, 16, 13, 14]
- [11, 15, 16, 14]

To the right of each option, there is a small teal button labeled "press" followed by a number (1, 2, 3, or 4) enclosed in a dark teal box. At the bottom right of the content area, there is a grey "Check" button.

## Online classes

- ★★★★★ [Python Basics](#) class notes and sample codes from ZHANG Honglun. This is good supplementary materials to the **week00-week03** in this book. There are abundant exercises to get one familiar with this language. You are strongly suggested to study this repo after we finish **week03** if not study earlier.
- ★★★★☆ [Google's Python class](#) This one is organised in a way suitable for developers who have experience in another language or who already have learned Python and use it to strengthen the knowledge. The best time to study this one is before the whole class here, or after you learned **week00-week04** (after we cover file operations and HTTP requests/ responses). This study material provides you another perspective of approaching Python. It is always good to have two or more different narratives when you learn a new programming concept.

## Related GitHub repos

- ★★★☆☆ [Project based learning for Python](#) -- a curated list by @tuvtran that includes practical projects you can make with Python. This repo is good for people who have mastered **week00-week04** of this open book.
- ★★★☆☆ [A collection of Python Q/A translated from Stackoverflow](#). This gitbook is good for people who have mastered **week00-week03**.
- ★★★☆☆ [A collection of interview questions about Python](#). This may be overshooting for the intended audience of our current repo. The interview questions are mainly for programmer positions. Still, it is good to test how far you can push. You are suggested to read this repo after having enough exercises, e.g. after **week12**. You can use the [language feature section](#) to enhance your understanding of the language basics. For people interested in "algorithm", the [last section](#) includes some entry level exercises.
- ★★★☆☆ [Python machine learning notebooks](#). This repo is recommended for readers who have finished the whole book and would like to learn more about machine learning.

**TODO:** Feel free to propose your entry into this list

## Other resources

Note: following resources are not fully reviewed and evaluated by our team. Those may be good further reading resources but may be far reaching for the intended audience of this open book.

- Artificial Intelligence in Python, by Prof. Ikhlaq Sidhu. Open source slides and Jupyter notebooks <https://github.com/ikhlaqsidhu/data-x> . Also see more on: <https://data-x.blog>
- List of Python resources (awesome-list style): [Link](#)
- A Python data science book targeted for non-tech background reader. This is a suggested reading for those who master our weekly notes. [Link](#).

# GitHub

- GitHub
  - Host open data sets on GitHub
  - How to download a file from GitHub web page
  - why we should preview Jupyter notebook on NBview? Are there any relationship with Github?
  - How to change default branch for GitHub pages?
    - gh-pages
  - What is index.html
  - Any real world example of using GitHub issue tracker?

## Host open data sets on GitHub

Open datasets contains three parts:

1. The dataset, in format of `.csv`, `.json`, or a directory of those files.
2. The scripts involved for generating the dataset, e.g. scraper, data cleaning logics, data transformations. See [Dataprep](#) for more information.
3. A `README.md` to show the basic information of this dataset.

Create a new file called `README.md`:

The screenshot shows a GitHub repository page for 'hupilidemo / hkbu-big-data-media'. The repository has 1 watch, 0 stars, and 0 forks. The navigation bar includes Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. The branch is set to master. In the top right, there are buttons for Unwatch, Star, Fork, and a red-dashed oval highlights the 'Create new file' button. Below the navigation, there's a file upload section with a 'Create new file' button, 'Upload files', 'Find file', and 'History' buttons. The main content area shows three files: 'imdb.ipynb' and 'mymovies.csv' both added via upload a day ago, and 'add files via upload' for 'hkbu-big-data-media / homework2 /'. The latest commit was 793fa6a a day ago.

Write the description file which usually contains introduction of data source, the background of research, data fields, and data size, limitation and license. If you do not know what licenses are available, we suggest you to use [CC 4.0](#). The file is written in [markdown](#) language, which has simple syntax that is legible in either plaintext format or rendered HTML format.

# IMDB dataset

## Data source

This dataset contains IMDB's 2017 movie list scraped from the website.

Starting page: <http://www.imdb.com/list/ls058982125/>

## Data fields

- \* `title` - String. e.g. `Thor: Ragnarok`
- \* `rating` - Int. e.g. `8.0`
- \* `runtime` - String, format is number of minutes plus "min". e.g. `130 min`

## Data volume

- \* 100 rows (movies)

The overall shape of an open dataset looks like this: (you are looking at the rendered version of the markdown file)

## IMDB dataset

### Data source

This dataset contains IMDB's 2017 movie list scraped from the website.

Starting page: <http://www.imdb.com/list/ls058982125/>

### Data fields

- `title` - String. e.g. Thor: Ragnarok
- `rating` - Int. e.g. 8.0
- `runtime` - String, format is number of minutes plus "min". e.g. 130 min

### Data volume

- 100 rows (movies)

See [homework2](#) for a complete example.

Note: The "limitation" is an important section in your README file. For example, you may only be able to crawl 95% of the original dataset due to technical problems. Highlighting that in your description file is crucial for other people to base their analysis on your dataset. No dataset is ideal. Incomplete dataset is also valuable. The principle is **full reporting**.

## How to download a file from GitHub web page

Example: We will use the data from Openrice as an example and do the restaurant analysis. Assuming that we have already got certain amount of data from Openrice and saved it into csv file.

Here is the link of csv file which can be downloaded [here](#).

Branch: master ▾

Create new file Upload files Find file History

[python-for-data-and-media-communication / scraper-examples / open\\_rice /](#)

ChicoXYC	scrape all and merged into one csv	Latest commit f6a96e2 a day ago
..		
<a href="#">openrice_csvs</a>	scrape all and merged into one csv	a day ago
<a href="#">openrice.ipynb</a>	scrape all and merged into one csv	a day ago
<a href="#">openrice_sample.ipynb</a>	item first & missing data dealing	2 days ago
<a href="#">openrice_total.csv</a>	scrape all and merged into one csv	a day ago
<a href="#">openrice_urls-selenium.ipynb</a>	scrape all and merged into one csv	a day ago
<a href="#">sample.csv</a>	item first & missing data dealing	2 days ago

Click "raw" on the right upper corner.

Branch: master ▾ [python-for-data-and-media-communication / scraper-examples / open\\_rice / sample.csv](#) Find file Copy path

ChicoXYC item first & missing data dealing a5985d4 2 days ago

1 contributor

246 lines (245 sloc) | 35.7 KB

Raw Blame History

Search this file...

name	location	price	country	style
LAB EAT Restaurant & Bar	436	尖沙咀金巴利道87-89號僑豐大廈地下1-2號舖	\$201-400	西式
Shine	692	尖沙咀北京道12A號太子集團中心6樓	\$201-400	西式
Espuma	610	尖沙咀厚福街8號H8 2樓	\$101-200	西班牙菜
The Captain's House	511	尖沙咀厚福街8號H8 18樓	\$201-400	西式
Yadllie Plate	487	旺角西洋菜街1號兆萬中心11樓	\$101-200	韓國菜
Day and Nite by Master Kama	462	旺角山東街50號1-2樓	\$101-200	日本菜
漁獲浜燒 Toretore Hamayaki	503	銅鑼灣軒尼詩道525號澳門逸園中心18樓	\$201-400	日本菜
The Grill Room	508	銅鑼灣駱克道459-461號The L. Square 5樓	\$201-400	西式

You can see the raw csv file as below.

<https://raw.githubusercontent.com/hupili/python-for-data-and-media-communication/master/food.csv>

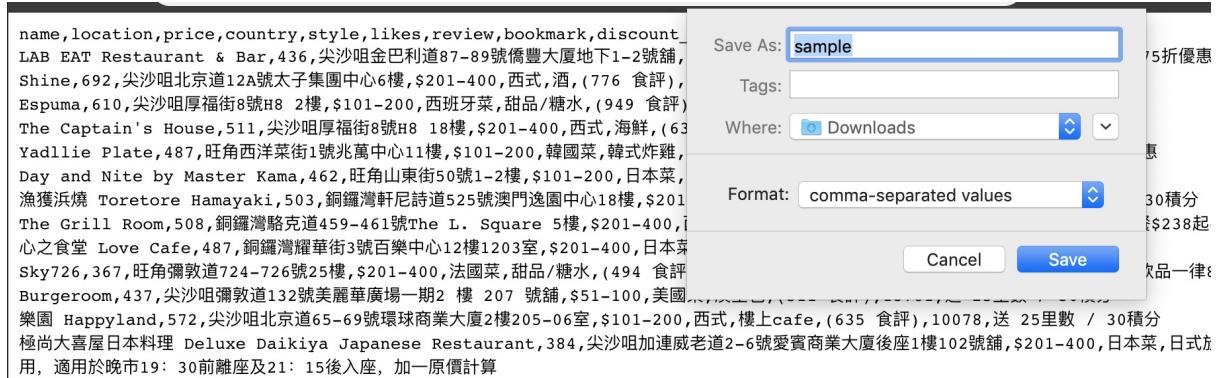
name,location,price,country,style,likes,review,bookmark,discount\_info  
LAB EAT Restaurant & Bar,436,尖沙咀金巴利道87-89號僑豐大廈地下1-2號舖,\$201-400,西式,海鮮,(565 食評),50664,網上訂座可享75折優惠  
Shine,692,尖沙咀北京道12A號太子集團中心6樓,\$201-400,西式,酒,(776 食評),21069,送 25里數 / 30積分  
Espuma,610,尖沙咀厚福街8號H8 2樓,\$101-200,西班牙菜,甜品/糖水,(949 食評),40374,送 25里數 / 30積分  
The Captain's House,511,尖沙咀厚福街8號H8 18樓,\$201-400,西式,海鮮,(634 食評),33139,送 25里數 / 30積分  
Yadllie Plate,487,旺角西洋菜街1號兆萬中心11樓,\$101-200,韓國菜,韓式炸雞,(630 食評),27542,早鳥及消夜時段訂座 可享全單八折優惠  
Day and Nite by Master Kama,462,旺角山東街50號1-2樓,\$101-200,日本菜,海鮮,(595 食評),28151,送 25里數 / 30積分  
漁獲浜燒 Toretore Hamayaki,503,銅鑼灣軒尼詩道525號澳門逸園中心18樓,\$201-400,日本菜,日本菜,(524 食評),19622,送 25里數 / 30積分  
The Grill Room,508,銅鑼灣駱克道459-461號The L. Square 5樓,\$201-400,西式,扒房,(662 食評),32287,訂座驚喜星期六及日: 套餐\$238起-全日酒精飲品Happy Hour 價錢  
心之食堂 Love Cafe,487,銅鑼灣耀華街3號百樂中心12樓1203室,\$201-400,日本菜,壽司/刺身,(542 食評),6551,送 25里數 / 30積分  
Sky726,367,旺角彌敦道724-726號25樓,\$201-400,法國菜,甜品/糖水,(494 食評),38441,自選餐牌主菜買一送一優惠 或 甜品8折優惠: 飲品一律8折  
Burgeroom,437,尖沙咀彌敦道132號美麗華廣場一期2 樓 207 號舖,\$51-100,美國菜,漢堡包,(511 食評),13781,送 25里數 / 30積分  
樂園 Happyland,572,尖沙咀北京道65-69號環球商業大廈2樓205-06室,\$101-200,西式,樓上cafe,(635 食評),10078,送 25里數 / 30積分  
極尚大喜屋日本料理 Deluxe Daikiya Japanese Restaurant,384,尖沙咀加連威老道2-6號愛賓商業大廈後座1樓102號舖,\$201-400,日本菜,日式放題,(507 食評),48390,特選時段 : 用, 適用於晚市19: 30前離座及21: 15後入座, 加一原價計算  
The Hwaduk,450,銅鑼灣開平道1號Cubus 21樓,\$101-200,韓國菜,韓式炸雞,(498 食評),16523,送 25里數 / 30積分  
Cafe Paradise,383,太子界限街23號地下C舖,\$51-100,意大利菜,甜品/糖水,(473 食評),25254,送 25里數 / 30積分  
Espuma,557,中環威靈頓街2-8號威靈頓廣場M88 17樓,\$101-200,西班牙菜,甜品/糖水,(690 食評),20889,送 25里數 / 30積分  
和匠日式燒肉店 Wa Shou Yakiniku,484,佐敦長樂街23-27號德威大廈地下3號舖,\$201-400,日本菜,烤肉,(537 食評),13005,送 25里數 / 30積分  
嵐山日本料理 Arashiyama Japanese Restaurant,341,尖沙咀登徑11號地舖,\$201-400,日本菜,壽司/刺身,(382 食評),15078,送 25里數 / 30積分  
Doux,380,旺角彌敦道726號18樓全層,\$101-200,多國菜,甜品/糖水,(473 食評),17124,送 25里數 / 30積分

Right click(or control +click in Mac) and choose "save as"

name,location,price,country,style,likes,review,bookmark,discount\_info  
LAB EAT Restaurant & Bar,436,尖沙咀金巴利道87-89號僑豐大廈地下1-2號舖,\$201-400,西式,海鮮,(565 食評),50664,網上訂座可享75折優惠  
Shine,692,尖沙咀北京道12A號太子集團中心6樓,\$201-400,西式,酒,(776 食評),21069,送 25里數 / 30積分  
Espuma,610,尖沙咀厚福街8號H8 2樓,\$101-200,西班牙菜,甜品/糖水,(949 食評),40374,送 25里數 / 30積分  
The Captain's House,511,尖沙咀厚福街8號H8 18樓,\$201-400,西式,海鮮,(634 食評),33139,送 25里數 / 30積分  
Yadllie Plate,487,旺角西洋菜街1號兆萬中心11樓,\$101-200,韓國菜,韓式炸雞,(630 食評),27542,早鳥及消夜時段訂座  
Day and Nite by Master Kama,462,旺角山東街50號1-2樓,\$101-200,日本菜,海鮮,(595 食評),28151,送 25里數 / 30積分  
漁獲浜燒 Toretore Hamayaki,503,銅鑼灣軒尼詩道525號澳門逸園中心18樓,\$201-400,日本菜,日本菜,(524 食評),19622,送 25里數 / 30積分  
The Grill Room,508,銅鑼灣駱克道459-461號The L. Square 5樓,\$201-400,西式,扒房,(662 食評),32287,訂座驚喜  
心之食堂 Love Cafe,487,銅鑼灣耀華街3號百樂中心12樓1203室,\$201-400,日本菜,壽司/刺身,(542 食評),6551,送 25里數 / 30積分  
Sky726,367,旺角彌敦道724-726號25樓,\$201-400,法國菜,甜品/糖水,(494 食評),38441,自選餐牌主菜買一送一優惠 或 甜品8折優惠  
Burgeroom,437,尖沙咀彌敦道132號美麗華廣場一期2 樓 207 號舖,\$51-100,美國菜,漢堡包,(511 食評),13781,送 25里數 / 30積分  
樂園 Happyland,572,尖沙咀北京道65-69號環球商業大廈2樓205-06室,\$101-200,西式,樓上cafe,(635 食評),10078,送 25里數 / 30積分  
極尚大喜屋日本料理 Deluxe Daikiya Japanese Restaurant,384,尖沙咀加連威老道2-6號愛賓商業大廈後座1樓102號舖,\$201-400,日本菜,日式放題,(507 食評),48390,特選時段 : 用, 適用於晚市19: 30前離座及21: 15後入座, 加一原價計算  
The Hwaduk,450,銅鑼灣開平道1號Cubus 21樓,\$101-200,韓國菜,韓式炸雞,(498 食評),16523,送 25里數 / 30積分  
Cafe Paradise,383,太子界限街23號地下C舖,\$51-100,意大利菜,甜品/糖水,(473 食評),25254,送 25里數 / 30積分  
Espuma,557,中環威靈頓街2-8號威靈頓廣場M88 17樓,\$101-200,西班牙菜,甜品/糖水,(690 食評),20889,送 25里數 / 30積分  
和匠日式燒肉店 Wa Shou Yakiniku,484,佐敦長樂街23-27號德威大廈地下3號舖,\$201-400,日本菜,烤肉,(537 食評),13005,送 25里數 / 30積分  
嵐山日本料理 Arashiyama Japanese Restaurant,341,尖沙咀登徑11號地舖,\$201-400,日本菜,壽司/刺身,(382 食評),15078,送 25里數 / 30積分  
Doux,380,旺角彌敦道726號18樓全層,\$101-200,多國菜,甜品/糖水,(473 食評),17124,送 25里數 / 30積分

Save As... Print... Cast... Translate to English  
① 1Password  
② Diigo  
③ Evernote Web Clipper  
④ Save Image to Eagle  
⑤ Save to Zotero  
⑥ Table Capture - Display inline  
View Page Source Inspect

Then the csv file can be saved as csv(comma-separated values).



## why we should preview Jupyter notebook on NBview? Are there any relationship with Github?

One can directly preview a Python notebook on GitHub. However, GitHub prohibits Javascript execution for security reasons. If you have interactive chart, e.g. from `echart`, `plotly`, those will not render on GitHub. NBViewer supports javascript and it is the first free online tool to preview Python notebook, so we recommend it. For concrete examples of dynamic charts, @ChicoXYC can find one notebook from our project archive: <https://github.com/data-projects-archive>.

## How to change default branch for GitHub pages?

please see [here](#)

**gh-pages**

### What is index.html

Basically, index.html is the default file served by the web server. So it is equivalent to visit example.com and example.com/index.html. Naming your file as index.html can lead to this more concise notation in browser's address bar and in communication campaigns -- the naming in the world of web is usually the shorter the better. More explanations are [here](#).

## Any real world example of using GitHub issue tracker?

Use issue tracker as Q/A forum:

- [Builder Book](#)

Use issue tracker as blog post backend:

- [@fouber's blog](#), written in Chinese, from a senior frontend engineer.

Use issue tracker as web comment store:

- [gitalk](#). See the comment thread [here](#)

# HTML FAQ

## href attribute in a tag

The `href` attribute is a URL pointing another resource from current page. There are several schemas for the `href` attribute in `<a>` tag:

```
http://yourdomain.com/images/example.png
//yourdomain.com/images/example.png
/images/example.png
images/example.png
```

Suppose the URL of your current page is (notice the "s" in protocol part)

```
https://mydomain.com/documents/index.html
```

The above "partial forms will be automatically converted by your web browser to following results:

```
http://yourdomain.com/images/example.png
https://yourdomain.com/images/example.png
https://mydomain.com/images/example.png
https://mydomain.com/documents/images/example.png
```

Explanation:

1. href starting with `http://`, `https://`, or `//` are fully qualified URLs. Browser will use the string as full URL directly.
2. href starting with `//` is protocol agnostic, i.e. the resources can be accessed by either `http` or `https`. So the browser refers to the protocol used by current page to complete the protocol part.
3. The browser uses protocol and domain to complete the URL because the href starts with `/`, meaning reference to the "root folder" on this web server.
4. This needs to be contrasted with 3rd form. The href is just a relative path, i.e. without prefixing `/`. The browser uses protocol, domain and path of current page (`https://mydomain.com/documents/`) to complete the URL.

Read more on [Stack Overflow](#) and [Wikipedia](#)

# Encoding

- Encoding
  - `u'\ufffe'`
  - `iconv`
  - Encoding of Python script
    - Declare file encoding other than the default

`u'\ufffe'`

This is the Byte Order Mark in some encoding schemes. It is usually added by mistake or malformed software. You can remove this special character using `dos2unix filename.txt`. This command will modify the file in place.

Read more [here](#)

## iconv

`iconv` is a convenient command line tool to convert between two different file encodings. Suppose you have a `file.txt` which is GBK encoded and want to have UTF-8 version, you can use the following command.

```
iconv -f gbk -t utf-8 file.txt
```

The converted result is sent to standard output by default. You can use `stdout` redirect to save that in a file called `output.txt`.

```
iconv -f gbk -t utf-8 file.txt > output.txt
```

Suppose you have original GBK encoded files in `orig/` and want to output UTF-8 encoded text files to `utf-8/`. Following command can help you do the batch conversion.

```
ls orig -1 | xargs -I{} sh -c 'iconv -f gbk -t utf-8 orig/"{}" > utf-8/"{}"
```

MAC user can install iconv via `brew`.

## Encoding of Python script

### Declare file encoding other than the default

By default, the `.py` file you write uses the system encoding. For MAC and Linux, this had long been unified by `utf-8`. Windows systems are usually a bit tricky. The Chinese version Windows could use one of some common encodings other than `utf-8`: `gbk` in mainland; `big5` in Hong Kong and Taiwan. That makes your script less portable to other systems. Or you will meet problems, especially when string operation is heavily used in your program.

To declare an encoding other than the default one, a special comment line should be added as the first line of the file. the syntax is as follows:

```
coding={the codec name here}
```

```
Remaining part of your Python script
```

For example, if you have Chinese characters in your code and your operating system is also Chinese version of Windows, it is better to declare the coding by writing the following as the first line in your source code file:

```
coding=utf-8 or # -*- coding: utf-8 -*-
```

There might be other situations and encoding format, and its a case by case situation, if you encounter more situation, please free feel to open an issue.

# pip

- pip
  - [pip2 and pip3](#)
  - [Install pip3](#)
  - [pip3 install modules](#)

`pip` is the default package manager in Python. It is like the `npm` and `gem` in Javascript and Ruby.

## pip2 and pip3

Depending on when and how your system is installed, you may end up with `pip`, `pip3`, `pip`. Double check who is who from the version information:

```
%pip2 --version
pip 18.0 from /usr/local/lib/python2.7/site-packages/pip (python 2.7)
%pip3 --version
pip 18.0 from /usr/local/lib/python3.7/site-packages/pip (python 3.7)
%pip --version
pip 18.0 from /usr/local/lib/python2.7/site-packages/pip (python 2.7)
```

In my current system, `pip` refers to `pip2` which is used in combination with Python 2. Different from them, `pip3` is used in combination with Python 3.

## Install pip3

`pip` is already installed if you are using Python 2  $\geq 2.7.9$  or Python 3  $\geq 3.4$ . You can type `python --version` to check out current version. If you are the older versions, please check out [here](#) to install and upgrade `pip`.

If you install `pip` in anaconda environment, try to use `!conda install pip3` in Jupyter.

## pip3 install modules

please see [here](#). If you install in Jupyter, add `!` ahead of the command.

# Computational Thinking

## Divide and conquer

### Recursion implementation

```
students = list(range(21))
cases = list(range(5))
print('students:', students)
print('cases:', cases)

def distribute(s, c):
 if len(c) == 1:
 print('Case:', c[0], 'Students:', s)
 else:
 p = len(s) // len(c)
 print('Case:', c[0], 'Students:', s[: p])
 distribute(s[p:], c[1:])

distribute(students, cases)
```

Try out other number of students and cases to see if this still works.

## File IO FAQ

### Do not see the content in file after successful .write statement

This is usually due to [File I/O buffer, or Data buffer](#). Computer systems use buffer to avoid frequent I/O operations because those operations are usually costly. Producer side (in our case, Python program) keeps writing to the buffer. When buffer is flushed, the consumer side (in our case, the file on disk) can see the content. File buffer is usually flushed at two events:

- When you `.close()` a file. That is why we ask you not to forget closing a file after using it.
- When the cache is full. To know the default size of the cache, use the following script:

```
>>> import io
>>> io.DEFAULT_BUFFER_SIZE
8192
```

### Write to closed file

This error message is direct: `ValueError : I/O operation on closed file. .`

You can not `.write()` or `.read()` after closing a file. You can re-`.open()` this file and try again.

# Leetcode

- Leetcode
  - Getting started
  - Control flow
  - List
  - String
  - Set
  - Algorithms
    - Problem comprehension and simulation
    - Cummulative summation
    - Search, recursion, backtacking
    - Dynamic programming

Leetcode is a programming training website specially designed for Internet companies. The target audience is generally computer science background students. The problems on leetcode may involve some advanced data structure (more than `list`, `dict` stuffs you learn here) and advanced algorithms. A common heuristic is that, once you can finish first 200 random leetcode, you have very good chance to get into tier one Internet companies in the world. We do not require our students to be as proficient as that. However, this is the place to get you started with "computational thinking". We curate the following list with annotations for those curious minds. You can use the following list to test your basic Python knowledge.

Most of the those problems below are labelled "Easy" on leetcode. We re-estimate their difficulty using 5-stars taking our student background into consideration. The 1-star and 2-star problems are meant to be tractable by an average students from the class. The 3-star and 4-star problem require extra efforts but within your reach most of the time. You are suggested to strike to 4-star problems in order to gain good efficiency. That helps in the long run, if you want to pursue a data analyst career after graduation. 5-star problems touch the surface of "algorithm design". It is an advanced topic that even some computer science students do not feel comfortable with. You are **strongly** suggested to find a mentor who can give you some general directions before you start, or it is highly like to be a waste of time (no meaninful results after hours/ days)...

## Getting started

We use `reverse-string` as an example to get you started:

```
class Solution:
 def reverseString(self, s):
 """
 :type s: str
 :rtype: str
 """
 return s[::-1]
```

You can win points by copy and pasting the above one-line solution to the input text box (select `Python3` as the language). As a matter of ethics, do not direclty copy and paste other's code in the future. Try to comprehend the solution and make your own. We use this example to help you quickly get an idea of what is an Online Juedge (OJ) and how it works. Leetcode problems follow the same format. There is a class called `Solution` and an entry point method/ function, in this case `reverseString`, to be called when the OJ evaluates your code. Your task is to fill in the body of this function. The function takes input data from argument list and gives answer using the `return` statement.

You can use "Run Code" button and "Customise Testcase" to try different input values. Once you "Submit", the OJ will test different input dataset with your method and give evaluation results. Once you see "Accepted", you are successful.

## Control flow

- ★★★☆☆ [lemonade-change](#) - involves `for`, `if` and variables. A very realistic problem in our life. Good to exercise logical thinking.
- ★★★★☆ [utf-8-validation](#) - a practical example for you to exercise advanced control flow. If you can pass this question with only a few number of trials, that is a good sign of mastery of control flow. Hint: you can use `'{:08b}'.format(d)` to quickly turn an integer into binary (0/1) presentations; the exercise involves, `for`, `if`, `else`, `continue`, `return`. "early return" is a useful trick to simplify code in a function.

## List

- ★★★☆☆ [rotate-array](#) - basic solution involves list slicing. Need to take care when `k` is larger than the list length. There are many alternative solutions that may require some logical thinking.
- ★★★★☆ [pascals-triangle](#) - Exercise the list-of-list structure. Try to find the pattern between each two rows.

## String

- ★☆☆☆☆ [to-lower-case](#) - Python has string function to directly solve it. It will be a good exercise to implement the `str.lower()` function yourself using `for`, `ord`, `chr` and string concatenation
- ★★★★☆ [license-key-formatting](#) - involves `str` functions, integer division's quotient and remainder. Requires a bit sense of math.

## Set

- ★★★☆☆ [unique-email-addresses](#) - mainly str processing. Use `set()` to efficiently deduplicate and count. This question is also good for people to learn the alternative email address you can use, by adding `.` and `+`. You can try one alternative address with your friend.

## Algorithms

Algorithm problems are usually integrated practice of all the above. We do not label the Python basics involved in the problems anymore.

### Problem comprehension and simulation

Those problems are "straightforward" in the view of algorithm engineers. They do not involve much algorithmic ideas. Once you fully understand the problem, you can use computer program to simulate the process as described by the problem. The simulation will get you the answer directly.

- ★★★★★ [degree-of-an-array](#) - requires good problem comprehension and problem conversion. Design staged solution once you can re-interpret the problem as: *among the most common numbers, find the one whose minimum index and maximum index are closest.*
- ★★★★★ [long-pressed-name](#) - a very practical problem. First think how you compare two strings and then adapt the basic algorithm to this problem. Watch out for corner cases (boundary conditions).

## Cummulative summation

Cummulative summation (cumsum) is a common technique to save computation on a sequence of elements. Suppose we have a list of numbers in `A`. Let's denote cumsum of `A` as `c`, where `c[i] = sum(A[: i])`. After this pre processing, a range sum query can be answered by `sum(A[i: j]) = c[j - 1] - c[i - 1]`. The LHS (original sum) involves summation over a series of elements. The RHS (cumsum) involves only a **single subtraction**, thus more efficient.

- ★★★★★ [flip-string-to-monotone-increasing](#) -- The key step is to find an index `i` such that `s[: i]` will be flipped into `0` and `s[i: ]` will be flipped into `1`. To efficiently compute how many flips are needed, we need to get cumsum of: 1) number of 1's on the left and 2) number of 0's on the right.

## Search, recursion, backtacking

Search problem usually involves exponential problem space. For example, playing Go is a search problem in essence. Every step, there are up to 400 potential moves. If you consider `n` steps, the space is exponentially large -- `400 ** n`.

The method to explore such large space is called recursion/ backtracking. The implementation usually looks like some function calling itself with different parameters. When recursion/ backtracking hits boundary condition, simple and straight solution exists (e.g. when problem size is 0 or 1).

```
def search_n_step(n):
 if n == 0:
 return a_simple_solution
 solution_to_sub_problem = search_n_step(n - 1)
 solution = integrate_current_step_into_sub_problem(step, solution_to_sub_problem)
 return solution
```

## Dynamic programming

Dynamic Programming (DP) is closely related with search problem. DP problem seems to be solvable by search at first glance. However, due to the large search space, the naive algorithm is not efficient enough. DP utilizes the problem structure, that *state could be memoryless* -- that is:

- how we reach this state is not important
- our future optimal decision is purely based on the current state

The concept is abstract, and can be explained via following exercises:

- ★★★★★ [minimum-falling-path-sum](#) -- We can handle the matrix row by row. For `A[i][j], F[i][j] = min(F[i - 1][j - 1], F[i - 1][j], f[i - 1][j + 1])`. We calculate all `F` by increasing `i`, i.e. row by row. How we reach the minimum at `i-1` th row is not important. The only thing that matters is what is the value at `F[i - 1][:]`.

## FAQ related with geopy

### Service timeout error

You will face this problem if you tried to request this address many times then the environment blocked you. Another potential reason is the internet speed is too slow to cache your results.

# Requests

- [Requests](#requests) - [Modify User Agent](#modify-user-agent) - [HTTP status code](#http-status-code) - [Return empty results](#return-empty-results)

## Modify User Agent

Some websites combat crawler by detecting the [user agent](#). User agent can be simply regarded as the name of your browser. Websites may stop your HTTP request if it detects you are not using a normal browser. That is because `requests` will tell the website its identity by default. You can modify this behaviour using `headers` parameter of `requests.get` :

```
r = requests.get(url, headers = {'user-agent': 'Put-User-Agent-String-Here'})
```

Use [Open Rice](#) as an example:

```
url = 'https://www.openrice.com/en/hongkong/restaurants?what=sushi'
r = requests.get(url, headers = {'user-agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/64.0.3282.167 Safari/537.36'})
```

A more complete experiment can be found [here](#)

## HTTP status code

When you make a request to a website, there might be different status responded. Common examples here:

- 200 OK
- 400 Bad Request
- 401 Unauthorized
- 403 Forbidden

For more examples, please refer to [here](#).

## Return empty results

Case: Airbnb

```
import requests
from bs4 import BeautifulSoup
r = requests.get('https://www.airbnb.com/s/all?adults=1&children=0&infants=0&guests=1&toddlers=0&refinement_paths%5B%5D=%2Ffor_you')
html_text = BeautifulSoup(r.text,"html.parser")
hotels = html_text.find_all('div')
hotels
```

```
: hotels = html_text.find_all('div')
hotels
[: [<div id="smart-banner"></div>,
 <div class="alert alert-with-icon alert-error no-js-alert">
 <i class="icon alert-icon icon-alert-alt"></i>
 We're sorry, some parts of the Airbnb website don't work properly without JavaScript enabled.
 </div>,
 <div id="field-guide-container"></div>,
 <div class="flash-container"></div>,
 <div data-hypernova-id="83cb20b3-c531-4aee-8777-26397210b0df" data-hypernova-key="spaspabundlejs"><div data-reactroot="" dir="ltr"><div class="_rls6e9y"><header class="_8igo2b" role="banner"><div class="_lw9i3c80"><div class="_hgs47m"><div class="_ni9axhe"><div><div class="_ousp44"><div><div class="_pawvzww"><a aria-label="Airbnb homepage" class="_lg2dfiu" href="/logo=1"><div class="_5nim5h"><div class="_36rlri"><div class=".iiid5y"><svg aria-hidden="true" focusable="false" role="presentation" style="height:1em;width:1em;display:block;fill:currentColor" viewBox="0 0 1000 10
```

You will find the content you wanted is not there and if you save the content in a html and reopen it, it's a blank page.

```
open('mypage.html', 'w').write(r.text)
```

This is indicator that this page is loaded dynamically, you may need to use `selenium` or `splinter` to scrape instead.

## CSV

The so-called CSV (Comma Separated Values) format is the most common import and export format for spreadsheets and databases.

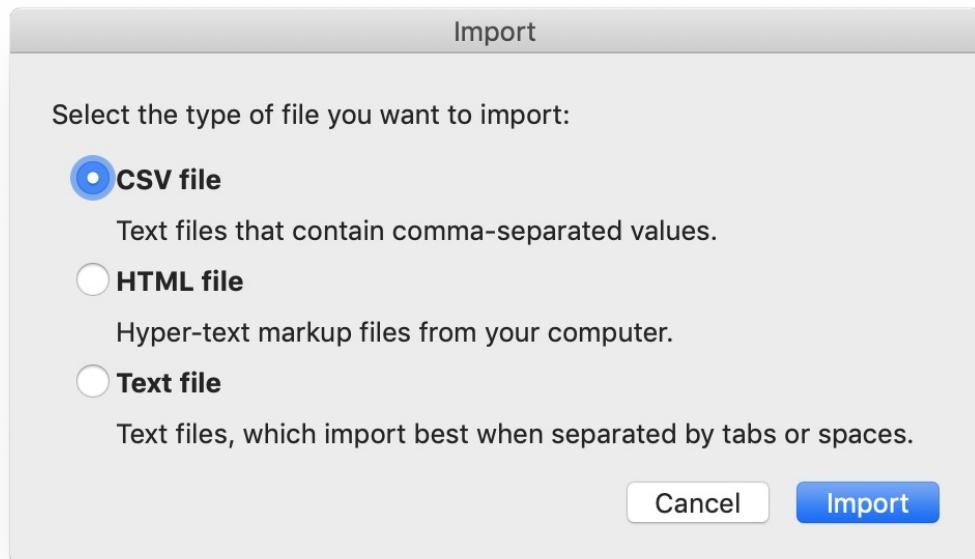
## Why present garbled words when you export the CSV?

Because the spreadsheets cannot recognize Chinese, then it presents garbled words on the screen. No worry about the garbled words if your results are right.

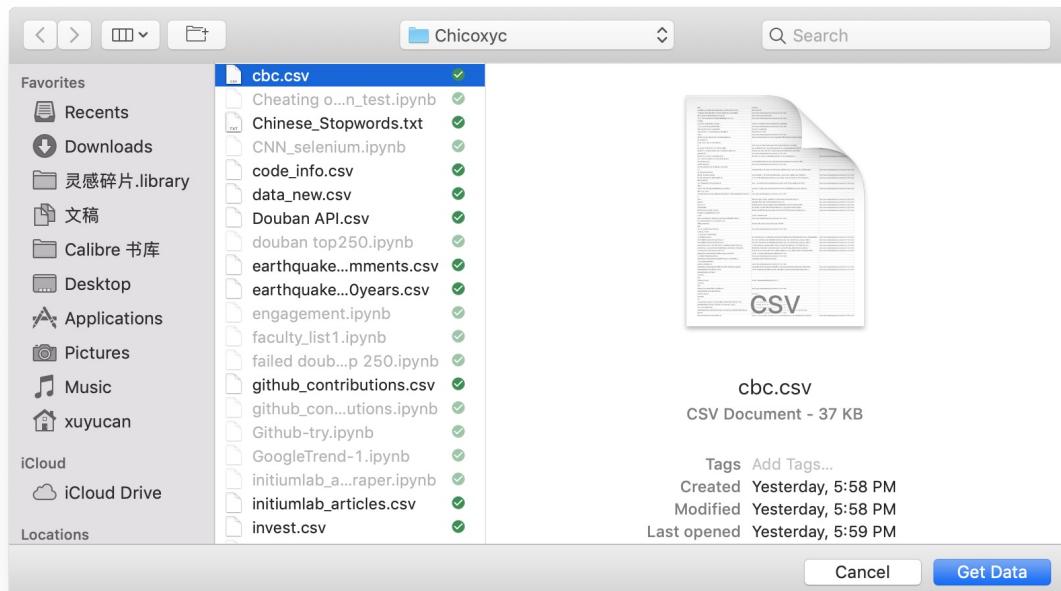
## Encoding issue about Microsoft excel

Open csv in Excel avoiding encoding error:

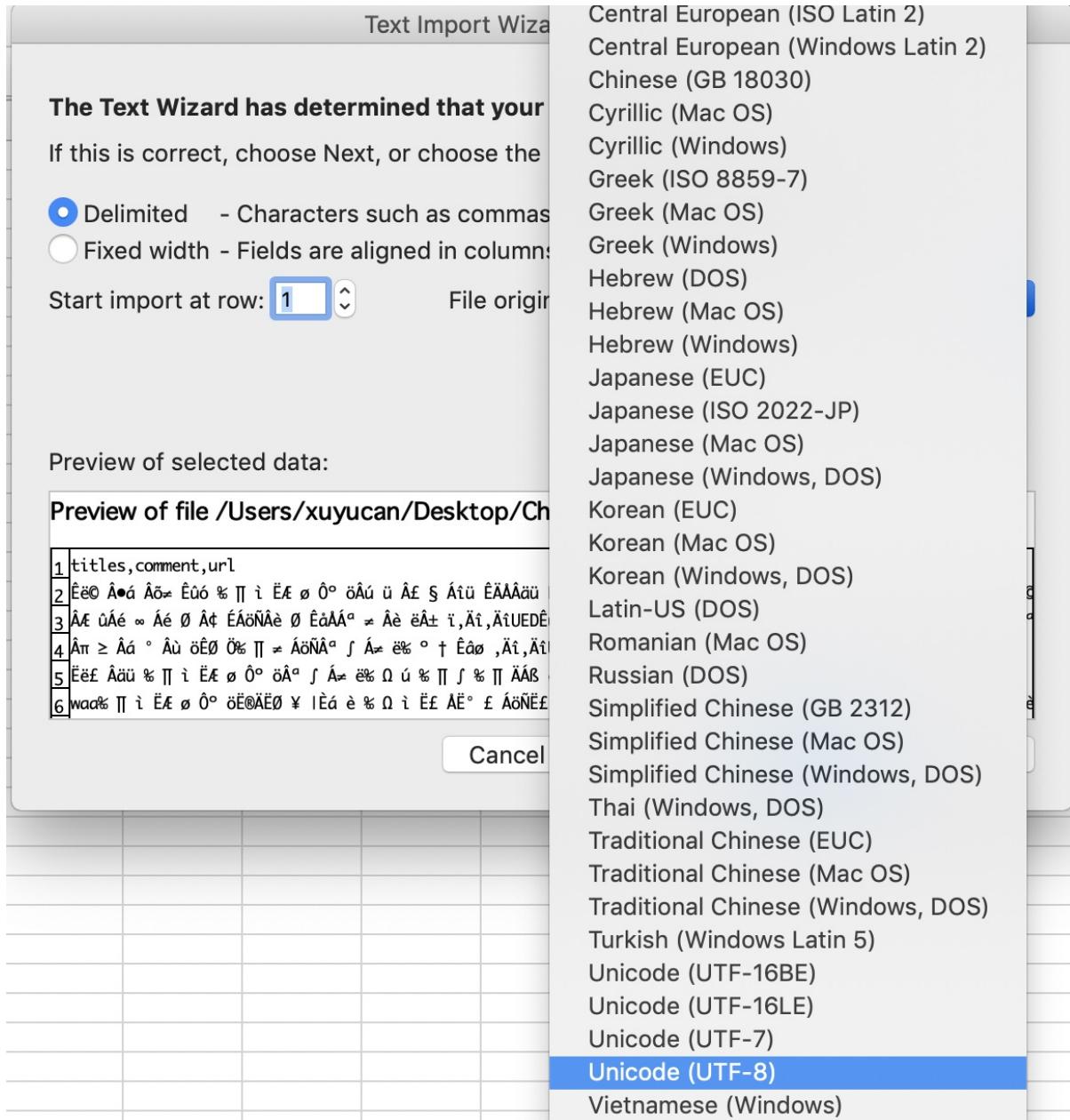
1. Open excel, choose to create a new file / blank workbook.
2. Click file --> import --> csv



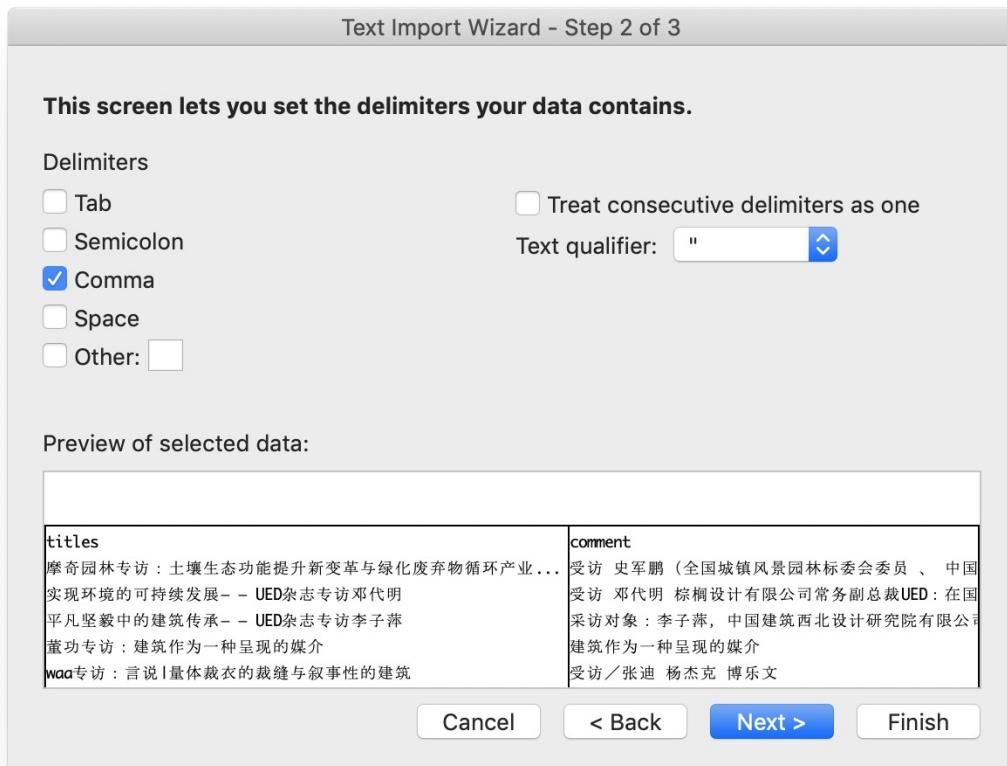
1. Import the csv you want, click `get data`



1. Click `file origin` to choose `unicode(UTF-8)`



1. Click `next` and choose delimiter, in csv, we usually use `comma`, then click next --> finish --> OK.



1. Then, your csv can displayed in Excel correctly.

A	B	C
1 titles		
2 摩奇园林专访：土壤生态功能提升新变革与绿化废弃物循环产业...	受访 史军鹏（全国城镇风景园林标委委员、中国风景园林学会企业工作委员会...	<a href="http://www.chinabuildingcentre.com/show-41-74">http://www.chinabuildingcentre.com/show-41-74</a>
3 实现环境的可持续发展——UED杂志专访邓代明	受访 邓代明 棕榈设计有限公司常务副总裁UED：在国家新型城镇化的大背景之下，生...	<a href="http://www.chinabuildingcentre.com/show-41-74">http://www.chinabuildingcentre.com/show-41-74</a>
4 平凡叙事中的建筑传承——UED杂志专访李子萍	采访对象：李子萍，中国建筑西北设计研究院有限公司专业总建筑师采访媒体：《城市...	<a href="http://www.chinabuildingcentre.com/show-41-74">http://www.chinabuildingcentre.com/show-41-74</a>
5 董功专访：建筑设计为一种呈现的媒介	建筑作为一种呈现的媒介	<a href="http://www.chinabuildingcentre.com/show-41-74">http://www.chinabuildingcentre.com/show-41-74</a>
6 was专访：言论载体裁剪的链接与叙事性的建筑	受访 / 张迪 杨杰克 博乐文	<a href="http://www.chinabuildingcentre.com/show-41-73">http://www.chinabuildingcentre.com/show-41-73</a>
7 WOHA专访：不断发现有新意且可持续发展的解决方案	受访 / 黄文新（Wong Mun Summ）、理查德·哈塞爾（Richard Hassell）WOHA 建筑事务所	<a href="http://www.chinabuildingcentre.com/show-41-73">http://www.chinabuildingcentre.com/show-41-73</a>
8 【专访】伊纳克·阿巴罗思：实用主义的人文商业情怀	“现在已经不是看地标建筑的最高，而是要看里面的人如何生活在高处。”	<a href="http://www.chinabuildingcentre.com/show-41-73">http://www.chinabuildingcentre.com/show-41-73</a>
9 【专访】丹尼尔·里伯斯金：当艺术遇见商业	我相信只有当建筑师动手设计，才会拥有普雷斯特经历的那种时刻——碰巧走到心灵的...	<a href="http://www.chinabuildingcentre.com/show-41-73">http://www.chinabuildingcentre.com/show-41-73</a>
10 金陵美术馆   “工业厂房”为起点，以“工业材料”为终点	金陵美术馆是20世纪60-70年代的工业厂房改建。它仿佛一位在场的使者悄然矗立在南京...	<a href="http://www.chinabuildingcentre.com/show-41-73">http://www.chinabuildingcentre.com/show-41-73</a>
11 华黎：地·景中的建筑   UED百名建筑师展	华黎：地·景中的建筑   UED百名建筑师展地·景中的建筑Building in L...	<a href="http://www.chinabuildingcentre.com/show-41-73">http://www.chinabuildingcentre.com/show-41-73</a>
12 从表想到深绿——作为绿色建筑产业化引领者的深度思考与探索	走进天友集团位于天津的绿色设计中心，迎面而来的就是鱼水景和透光性极强的大空间...	<a href="http://www.chinabuildingcentre.com/show-41-73">http://www.chinabuildingcentre.com/show-41-73</a>
13 【为城市而设计·智慧改变城市】王中：郑州1904公园	案例的意义并不在于项目本身，而在于我们提出“艺术激活空间”主张的落地实践。事...	<a href="http://www.chinabuildingcentre.com/show-41-73">http://www.chinabuildingcentre.com/show-41-73</a>
14 【为城市而设计·智慧城市】巴索那那建筑面面观	采访来自《城市·环境·设计》（UED）杂志第95期 2015年9月刊 主持人：里卡德...	<a href="http://www.chinabuildingcentre.com/show-41-73">http://www.chinabuildingcentre.com/show-41-73</a>
15 【为城市而设计·景观城市】野趣·野去	每一个在城市里忙忙碌碌穿梭于鳞次栉比的钢筋水泥中的人都有一个身在郊野的梦想，而...	<a href="http://www.chinabuildingcentre.com/show-41-73">http://www.chinabuildingcentre.com/show-41-73</a>
16 单军：建筑应当关注地域性  【从北京到伦敦展】访谈	采访 / 王军（UED）（原载于《UED》，2012（08）：从北京到伦敦——当代中国建筑...	<a href="http://www.chinabuildingcentre.com/show-41-72">http://www.chinabuildingcentre.com/show-41-72</a>
17 单军：关键不在于是否在境外盖房子  【境外实践专题】访谈	采访 / 范曾（原载于《世界建筑》，2015（01）：建在别处·境外实践专辑）中国建筑师...	<a href="http://www.chinabuildingcentre.com/show-41-72">http://www.chinabuildingcentre.com/show-41-72</a>
18 单军对话王路：之间	关于钟祥市博物馆暨明代帝王文化博物馆设计单军 王路（原载于《时代建筑》，2013...	<a href="http://www.chinabuildingcentre.com/show-41-72">http://www.chinabuildingcentre.com/show-41-72</a>
19 内外之间	——建筑地·点性表达的两种途径单 军 铁霞（原载于《世界建筑》，2010(08)）...	<a href="http://www.chinabuildingcentre.com/show-41-72">http://www.chinabuildingcentre.com/show-41-72</a>
20 春秋之同	——春秋双门单军铁霞（原载于《建筑学报》，2010(11)）...	<a href="http://www.chinabuildingcentre.com/show-41-72">http://www.chinabuildingcentre.com/show-41-72</a>
21 山水之间	——钟祥市博物馆暨明代帝王文化博物馆设计单军 卢向东 王鑫 铁雷（原载于《建...	<a href="http://www.chinabuildingcentre.com/show-41-72">http://www.chinabuildingcentre.com/show-41-72</a>
22 飞地办公双重地域性	——中国商务部驻印尼商务设计单军 刘玉龙 铁雷 赵巍（原载于《建筑学报》...	<a href="http://www.chinabuildingcentre.com/show-41-72">http://www.chinabuildingcentre.com/show-41-72</a>
23 《地区建筑设计系列研究丛书》连载	单军（原载于《地区建筑设计系列研究丛书》，2014）谈到地区建筑设计系列的研究的缘起，...	<a href="http://www.chinabuildingcentre.com/show-41-72">http://www.chinabuildingcentre.com/show-41-72</a>
24 【UED】城市与未来——模块化建筑多元专访	2015年7月23日建筑纪元网于上海展览中心举办 Benoy 上海区总经理庞长受邀参加，并进...	<a href="http://www.chinabuildingcentre.com/show-41-72">http://www.chinabuildingcentre.com/show-41-72</a>
25 广东博物馆新馆首登！主设计师严迅奇回应三大争议	开放不到半个月的广东博物馆，已经成为广州城内最受关注的一颗明星建筑，每天都有...	<a href="http://www.chinabuildingcentre.com/show-41-72">http://www.chinabuildingcentre.com/show-41-72</a>
26 上完这一课——全世界都会听到的——设计公司产品创新Workshop	引言设计企业业绩大爆发！设计人的自我角色迷失！好日子过去了，我们很怀念它。如...	<a href="http://www.chinabuildingcentre.com/show-41-71">http://www.chinabuildingcentre.com/show-41-71</a>
27 陈阳说产品创新：抓住从0到1的机会（下）	引言当下设计行业的市场有所变化，设计企业面临各种的问题，企业的管理思维提升以...	<a href="http://www.chinabuildingcentre.com/show-41-71">http://www.chinabuildingcentre.com/show-41-71</a>
28 陈阳说产品创新：抓住从0到1的机会（上）	引言当下设计行业的市场有所变化，设计企业面临各种的问题，企业的管理思维提升以...	<a href="http://www.chinabuildingcentre.com/show-41-71">http://www.chinabuildingcentre.com/show-41-71</a>
29 艺术建筑：使每个人都参与到艺术中来——Aedas 文化艺术团队专访	——中国建筑报道专访Aedas 文化艺术团队主题：演艺建筑——使每个人都参与到艺术中...	<a href="http://www.chinabuildingcentre.com/show-41-71">http://www.chinabuildingcentre.com/show-41-71</a>
30 设计企业不要错过从0到1的机会——CBC学院专访ADU首席顾问陈阳老师	——CBC学院专访ADU首席顾问陈阳老师采访主题：市场竞争与目前设计企业的发展趋势...	<a href="http://www.chinabuildingcentre.com/show-41-71">http://www.chinabuildingcentre.com/show-41-71</a>
31 为何古罗马建筑可以屹立上千年不倒	古罗马建筑为何能上千年屹立不倒？利用AL5光学线12 3 2...一种超导体弯曲磁铁反射微...	<a href="http://www.chinabuildingcentre.com/show-41-71">http://www.chinabuildingcentre.com/show-41-71</a>
32 【视频】UED深圳华大讲堂——蓝天组prinx深圳讲座	2014年11月19日 奥地利蓝天组创始人 Wolf D Prix莅临“UED深圳华大讲堂”，...	<a href="http://www.chinabuildingcentre.com/show-41-71">http://www.chinabuildingcentre.com/show-41-71</a>
33 【视频】UED深圳华大讲堂——理查德·瑞杰斯特：未来城市——...	由UED杂志社、深圳华汇设计有限公司、万科集团以及C&P国际生态技术研究所联合主办...	<a href="http://www.chinabuildingcentre.com/show-41-71">http://www.chinabuildingcentre.com/show-41-71</a>
34 【视频】UED深圳华大讲堂——哈尼·拉什德：形式追随未来	美国知名建筑师哈尼·拉什德受由《城市·环境·设计》（UED）杂志社、深...	<a href="http://www.chinabuildingcentre.com/show-41-71">http://www.chinabuildingcentre.com/show-41-71</a>
35 【视频】UED大师讲堂——承孝哲：地文	2014年6月6日 韩国知名建筑师承孝哲，受由《城市·环境·设计》（UED）杂志社、深...	<a href="http://www.chinabuildingcentre.com/show-41-71">http://www.chinabuildingcentre.com/show-41-71</a>
36 【视频】UED大师讲堂——布兰登·麦克法兰：“场所与身份”	2014年7月25日晚，UED大师讲堂·布兰登·麦克法兰“场所与身份”讲座在深圳华夏艺术...	<a href="http://www.chinabuildingcentre.com/show-41-70">http://www.chinabuildingcentre.com/show-41-70</a>
37 【视频】UED大师讲堂——汤姆·梅恩(Thom Mayne)	汤姆·梅恩以“公共精神Public-ness”为主题，结合十多个案例，分享了自己对于城...	<a href="http://www.chinabuildingcentre.com/show-41-70">http://www.chinabuildingcentre.com/show-41-70</a>
38 【视频】UED大师讲堂——丹尼尔·里伯斯金：建筑是一种语言	美国知名建筑师丹尼尔·里伯斯金(Daniel Libeskind)1946年出生在波兰一个纯犹太家庭...	<a href="http://www.chinabuildingcentre.com/show-41-70">http://www.chinabuildingcentre.com/show-41-70</a>

# BeautifulSoup

## How to get attributes of the HTML element

There is the `attrs` member variable in the `bs4.element.ResultSet` object, which is returned by `find` or `find_all`. You can use it to access HTML tag attributes.

```
import requests
import bs4
from bs4 import BeautifulSoup

r=requests.get(
 'https://hk.jobsdb.com/hk/jobs/sales-cs-business-devpt/4',
 headers={
 'User-Agent': 'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_5) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/65.0.3325.181 Safari/537.36'
 }
)
mypage = BeautifulSoup(r.text)
price1 = mypage.find_all('a', attrs={'class':'posLink'})
print(price1[0].attrs['href'])
```

One output is:

```
https://hk.jobsdb.com/hk/en/job/senior-sales-engineer-sales-engineer-100003006011403
```

# Jupyter

- Jupyter
  - Virtual environment
    - Create virtual environment
    - Enter virtual environment
    - Exit virtual environment
  - Jupyter Notebook
    - Install Jupyter notebook
    - Run Jupyter notebook
    - Quit the Jupyter notebook
    - Set jupyter environment in CVA517
  - NBViewer
    - How to use NBViewer to view the notebook online
    - Codes on NBViewer is not updated
  - Basic usage
  - Runtime troubleshooting guide
  - Install frequently used dependencies
  - Windows
    - Instructions of Installing Jupyter Notebook on Windows
  - Display charts in nbviewer
    - Import interactive charts in Jupyter notebook
    - Cannot display some static charts

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. In our course, Jupyter notebook will be our daily tool to write, test, and sharing our codes and works. It's very useful for us to learn and make **reproducible works**. The good advantage of Jupyter notebook includes:

- The Notebook has support for over 40 programming languages, including Python, R, Julia, and Scala.
- Notebooks can be shared with others using email, Dropbox, GitHub and the Jupyter Notebook Viewer.
- Your code can produce rich, interactive output: HTML, images, videos, LaTeX, and custom MIME types.

It is suggested to enter virtual environment before using Jupyter notebook. Because in the following study, we may need to install some packages and modules in Jupyter notebook. It's necessary to keep those files in the same path, so Jupyter can source the modules when you want to use.

## Virtual environment

The following are the usual path to setup jupyter environment. For users in CVA 517 LAB, please see [here](#).

### Create virtual environment

```
pyvenv venv
```

### Enter virtual environment

```
source venv/bin/activate
```

When you see `(venv)` appear in front of your command line prompt, that means the you are in the virtual environment. Always check this prefix to make sure you are working in the right environment.

## Exit virtual environment

You can use `deactivate` command to exit current virtual environment.

# Jupyter Notebook

Now you can install and enter Jupyter notebook.

## Install Jupyter notebook

Note: **you only need to do this once for every virtual environment**

```
pip3 install jupyter
```

## Run Jupyter notebook

Always run your Jupyter notebook in the virtual environment.

```
jupyter notebook
```

By default the notebook will be available at <http://localhost:8888/tree>

So, next time when you use your jupyter notebook, you just need to type following commands in your terminal

```
source venv/bin/activate
jupyter notebook
```

## Quit the Jupyter notebook

```
$ control+C
```

Pay attention to the text. Then you will get the following picture. Please input `y` in 5 seconds.

```

^C[I 11:00:26.519 NotebookApp] interrupted
Serving notebooks from local directory: /Users/xuyucan
7 active kernels
The Jupyter Notebook is running at:
http://localhost:8888/?token=1e46e8e94f7a7b21c4276f8fe6cf23e8d69804ac47190ccb
Shutdown this notebook server (y/[n])? y
[1 11:00:29.703 NotebookApp] Shutdown confirmed
[1 11:00:29.716 NotebookApp] Shutting down 7 kernels
[1 11:00:32.799 NotebookApp] Kernel shutdown: 3920b7db-1327-4085-ac99-6ddd17c4a1
8b
[1 11:00:32.805 NotebookApp] Kernel shutdown: 3e7bcd30-2bae-402a-8ae1-5c4b098994
2f
[1 11:00:32.808 NotebookApp] Kernel shutdown: 38d7bff8-1c7a-46f0-b3dc-4210d48321
72
[1 11:00:32.810 NotebookApp] Kernel shutdown: e2993708-53ec-4f44-9191-a1d7a3e28f
46
[1 11:00:32.812 NotebookApp] Kernel shutdown: e6d5b986-1449-4ba1-9dd0-ac3216c182
81
[1 11:00:32.813 NotebookApp] Kernel shutdown: 47f75bc5-f080-4c7c-8005-d9be53107a
4b
[1 11:00:32.826 NotebookApp] Kernel shutdown: 36c08a18-50b7-4f2f-ad17-4690b1f37f
b5
xuyucan@MacBook-Pro:~ xuyucan$

```

## Set jupyter environment in CVA517

Due to the jupyter and the python conflict, there are problems of installing jupyter by the usual way. Instead, the following will work. For more details explanation, please see [here](#).

```

pyenv venv
source venv/bin/activate
pip install --upgrade pip
pip3 install jupyter
pip3 install 'ipython ==6.5.0'
pip3 install 'prompt-toolkit ==1.0.15'

```

## NBViewer

### How to use NBViewer to view the notebook online

You can use [NBViewer](#) to check out a Jupyter notebook hosted on GitHub. You only need to input the GitHub URL link into the input box.

### Codes on NBViewer is not updated

Sometimes, you change codes after first preview on NBViewer. You find the codes are updated on GitHub. However, NBViewer still shows the outdated codes used before. This is a common "cache" problem. You can try to trigger a cache invalidation by adding query string in the URL.

Suppose this is your original NBviewer link:

```
https://nbviewer.jupyter.org/xxxxx/yyyyy.ipynb
```

You can add anything after the `?` mark:

```
https://nbviewer.jupyter.org/xxxxx/yyyyy.ipynb?fffffff
```

The `fffffff` here can be arbitrary. Note, in your submission of the work, if the an NBviewer link is required, please paste the one that shows exactly the result you want, i.e. the one with some tailing querystring with which you can see the latest result. Or else, we may see the old content.

## Basic usage

1. click new to create a new python 3 notebook
2. write codes like you usually do in text editors, and press `shift+return` to run the code. It will return the results or errors under the cell.
3. use `! pip3 install module_name` to install modules in jupyter notebook.
4. in front of every cell, there is an `[ ]` sign, the number in `[]` means the sequences of cells, and if there is `*` in `[]`, means that this cell is still running, you can either wait it finish or click `stop` under the kernel to exit from the running, pressing `I` twice will also do the trick.
5. cell. run cell run step by step. run all above to run and check the previous steps of coding.
6. kernel. kernel is a tool for interactive input and output all the things you did from the beginning. By clicking `restart`, you can give a variable another value.
7. `type()` more to check the object. It is very useful when we write complicated codings. Eg:`a = 1, type(a)`
8. `help()` to know the details. Eg: `help(str.strip)`
9. `print` step by step to check where the error is. (In Jupyter, you can just input the variables without the function of print.) Like, type data in cell equals to `print(data)`.

## Runtime troubleshooting guide

- Rerun all the cells
- Restart the kernel
- Clear the output cells

## Install frequently used dependencies

You do not want to install and re-install dependencies every time. It is more convenient to setup a virtual environment and install common dependencies/ modules for data analysis.

You can download this [requirement.txt](#) and then run the following command (inside virtual environment)

```
pip install -r requirements.txt
```

## Windows

### Instructions of Installing Jupyter Notebook on Windows

please see [here](#).

## Display charts in nbviwer

### Import interactive charts in Jupyter notebook

Different libraries to save `.html` locally.

```
#plotly
plotly.offline.plot(data, filename='file name')
#pyecharts
bar/line.render('file name.html')
#bokeh
output_file("file name.html")
```

Import the html file it generate on your local computer.

```
from IPython.display import IFrame
IFrame('file_name.html', width=700, height=400)
```

### Cannot display some static charts

This is caused by the temporary cache in the browser. The first solution is you can change another browser to visit the link. The second, add several `???` in the nbviwer link to refresh the link, then the chart will display correctly.

---

If you have any questions, or seek for help troubleshooting, please [create an issue here](#)

# Pandas

## Rename csv header

We can use `rename` function to change the header.

For example, I want to change column A to B, column C to D.

Syntax:

```
df.rename(columns={'A':'B', 'C':'D'}, inplace=True)
```

For more explanation, you can refer [here](#)

## Wordcloud

When you did word-clouds project in python, you may see the key error as below:

command'/usr/bin/clang' failed with exit status 1

You can add one line of code showed here `xcode-select --install`, then you can solve the problem.

## Check column type and convert column type to numeric

A very typical example when performing numeric computations in pandas, e.g. calculating correlation:

```
df['Price'].corr(df['Num of equipment'])
```

One may encounter the following error:

```
76 if isinstance(ret, np.ndarray):
77 ret = um.true_divide(
--> 78 ret, rcount, out=ret, casting='unsafe', subok=False)
79 if is_float16_result and out is None:
80 ret = arr.dtype.type(ret)

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

This hints that the data type is incorrect. We can check the type as follows:

```
df_sort['Price'].dtype
dtype('O')
```

Convert the column to numeric in following way:

```
df_sort['p'] = pd.to_numeric(df_sort['Price'])
df_sort['p'].dtype
dtype('int64')
```

The `dtype('int64')` means that this columns is good for numeric computation now.



## seaborn

# matplotlib

## Troubleshooting Chinese characters on MAC

This notebook gives detailed steps in troubleshooting the Chinese characters on MAC. You will learn:

- Where are the fonts stored on your system
- How to filter out the font files that include CJK characters
- How to load a font file into matplotlib. Note that `ttc` does not work for some version of matplotlib. `ttf` works.

## How to display Chinese characters when using matplotlib

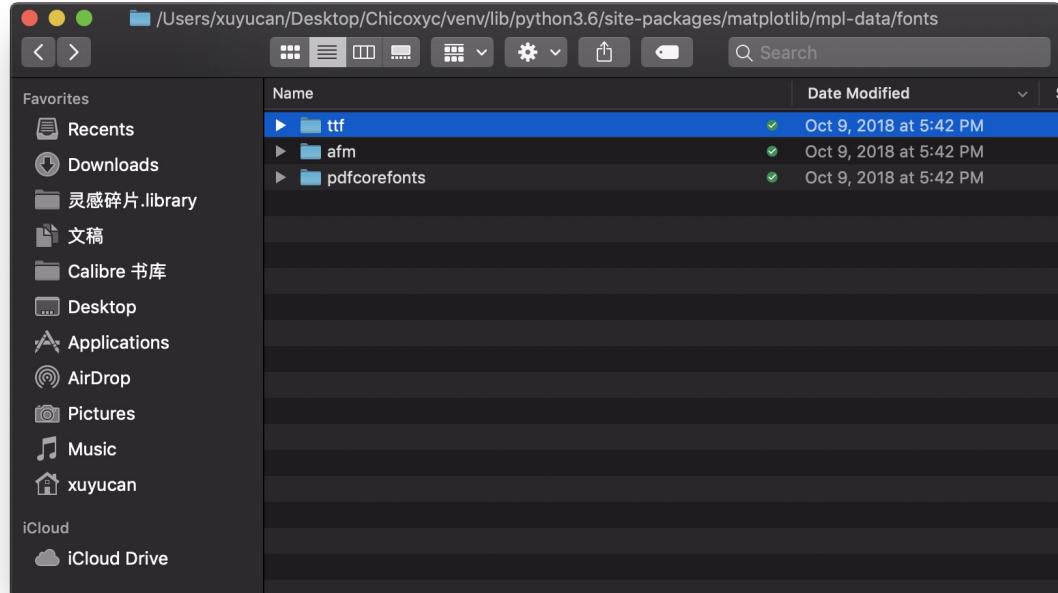
Step 1: Download the SimHei font [here](#)

Step 2: Find font folder in `mpl-data` folder

```
import matplotlib
print(matplotlib.matplotlib_fname())
#/Users/xuyucan/Desktop/Chicoxyc/venv/lib/python3.6/site-packages/matplotlib/mpl-data/matplotlibrc

!open /Users/xuyucan/Desktop/Chicoxyc/venv/lib/python3.6/site-packages/matplotlib/mpl-data
```

Step 3: Enter the font folder, put the font you just download into the folder, then click to install



Step 4: Back to `mpl-data` folder, click `matplotlibrc` command+f search font family, delete `#`; search font.sans-serif, delete `#`, and add `SimHei`; search axes\_unicode\_minus, replace `True` to `False`.

```

expanded, extra-expanded, ultra-expanded, wider, and narrower. This
property is not currently implemented.
##
The font.size property is the default font size for text, given in pts.
10 pt is the standard value.

font.family : sans-serif delete #
#font.style : normal
#font.variant : normal
#font.weight : normal
#font.stretch : normal
note that font.size controls default text sizes. To configure
special text sizes tick labels, axes, labels, title, etc, see the rc
settings for axes and ticks. Special text sizes can be defined
relative to font.size, using the following values: xx-small, x-small,
small, medium, large, x-large, xx-large, larger, or smaller
#font.size : 10.0
#font.serif : DejaVu Serif, Bitstream Vera Serif, Computer Modern Roman, New
Century Schoolbook, Century Schoolbook L, Utopia, ITC Bookman, Bookman, Nimbus Roman No9
|, Times New Roman, Times, Palatino, Charter, serif
font.sans-serif : SimHei, DejaVu Sans, Bitstream Vera Sans, Computer Modern Sans Serif,
Lucida Grande, Verdana, Geneva, Lucid, Arial, Helvetica, Avant Garde, sans-serif
#font.cursive : Apple Chancery, Textile, Zapf Chancery, Sand, Script MT, Felipa,
cursive delete # and add SimHei
#font.fantasy : Comic Sans MS, Chicago, Charcoal, ImpactWestern, Humor Sans, xkcd,
fantasy
#font.monospace : DejaVu Sans Mono, Bitstream Vera Sans Mono, Computer Modern
Typewriter, Andale Mono, Nimbus Mono L, Courier New, Courier, Fixed, Terminal, monospace

TEXT

```

#### Step 5: Delete the cache

```

import matplotlib as mpl
mpl.get_cachedir() #find the path of cache
!open /Users/xuyucan/.matplotlib
#delete files in this folder

```

#### Step 6: Restart the Jupyter notebook

1. command+c in terminal, enter y
2. type jupyter notebook again

#### Step 7: add following two lines into your codes

```

plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号

```

There is a similar issue about Chinese characters (CJK characters) when you use wordcloud. Please refer to [this section](#) for the solution.

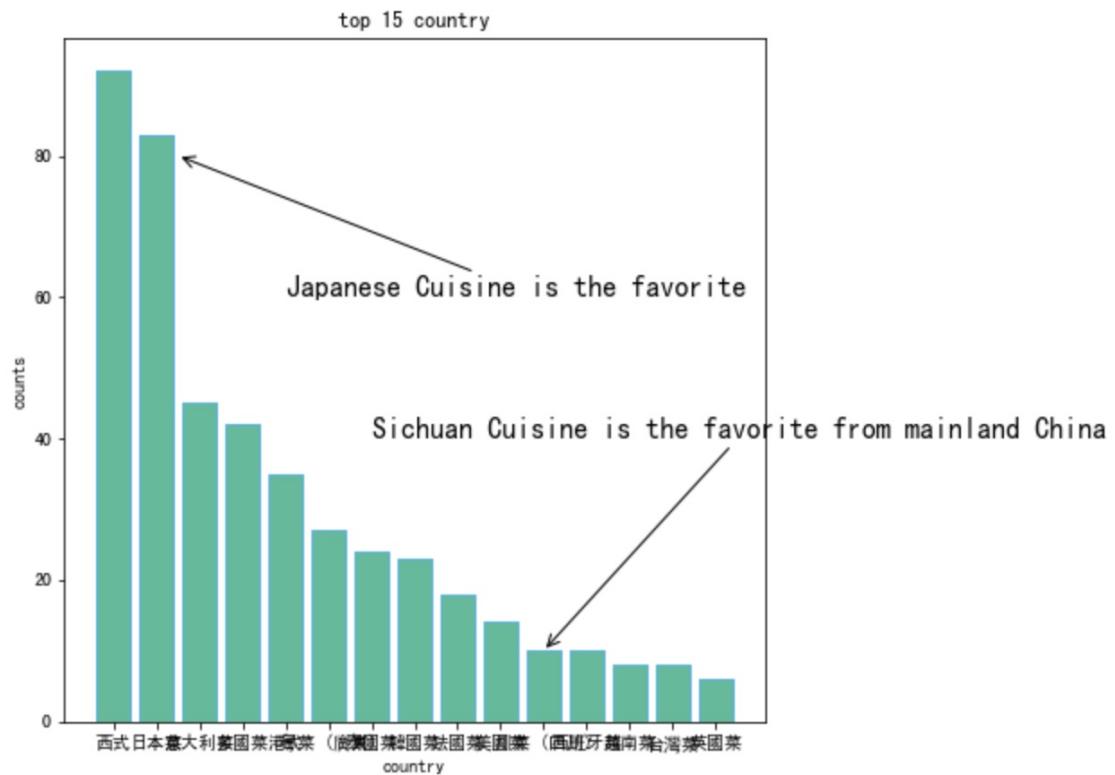
## Characters overlapping in text marker

When plotting the charts, you may encounter problem like characters overlapping. One solution is to change the `figure size` to enlarge the text margin.

```

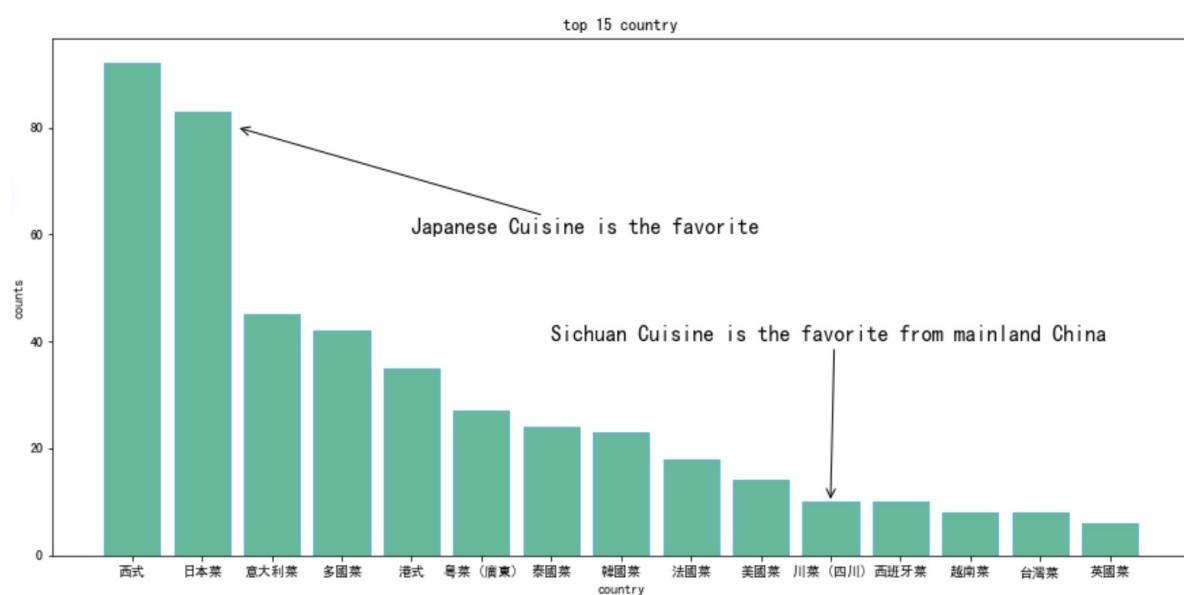
fig = plt.figure(figsize=(7,7))

```



After resizing by change the `figsize`

```
fig = plt.figure(figsize=(15, 7))
```



## **lxml**

## python-twitter

### Search public tweets on entire Twitter

The `GetSearch` function is useful, you can get the posts from one specific keyword searching.

```
GetSearch(term=None, raw_query=None, geocode=None, since_id=None, max_id=None, until=None,
since=None, count=15, lang=None, locale=None, result_type='mixed', include_entities=None,
return_json=False) [source]
```

Return twitter search results for a given term. You must specify one of term, geocode, or raw\_query.

For more functions, please refer to this [documentation](#).

## How to apply twitter API key?

The following are the applying experience of one of the 2018 fall students. credited to @iiiJenny.

### Step 1

Preparation: Twitter Account (highly recommend to use Gmail Account to Sign up!) . Then Go to <https://apps.twitter.com/app/new> and click 'Apply for a developer account'.

### Step 2

Choose "Personal use" & Enter your desired Application Name, Primary country (hk) and so on. The most important is : Describe in your own words what you are building.

## Describe in your own words what you are building

Please describe what you would like to build with Twitter's APIs. Be sure to give detailed answers to the following questions. If the question does not apply to your solution, please explicitly state that. The more detailed the response, the easier it is to review and approve.

1. What is the core use case, intent, or purpose for your use of Twitter's APIs?
2. Do you intend to analyze Tweets, Twitter users, or their content? If so, share details about the analyses you plan to conduct and the methods or techniques you plan to use.
3. Does your use case involve Tweeting, Retweeting, or liking content? If so, share how you will interact with Twitter users or their content.
4. How will Twitter data be displayed to users of your solution? If you plan to display Twitter content off of Twitter, explain how and where Tweets and Twitter content will be displayed to users of your product or service. Will individual Tweets and Twitter content be displayed, or will information about Tweets or Twitter content be displayed in aggregate?

To expedite your access approval, please be detailed...

Minimum characters: 300

you have to answer each question as detailed as possible. the following is an example:

1. I'm using Twitter's APIs to practice my data collection and analysis skills in the Big Data Analysis course. I am currently a postgraduate student in Hong Kong Baptist University, majoring in Communication, and normally collect users comment data for mass communication study research.
2. As for the methods and techniques I plan to conduct, here is our course GitHub open book. You can take it as reference. <https://github.com/hupili/python-for-data-and-media-communication-gitbook/blob/master/notes-week-04.md#use-api-via-function-calls-to-other-modules-packages>
3. I only use it to do data collection practice, will not use case to tweeting, retweeting or liking content.
4. Tweets will be displayed on our final project presentation for academic use.
5. Finally, I will comply with the Policies of twitter. I am looking forward to your favorable reply.

## Will your product, service, or analysis make Twitter content or derived information available to a government entity?

In general, schools, colleges, or universities do *not* fall under this category.

No

Yes

and then, submit your application! -> verify your email

after that, your application may under review or! jump to a new page :

## Welcome!

Congratulations! You have successfully created a new Twitter developer account. With this account, you now have access to new APIs, app management, and tools to facilitate and support development.

Below are a few steps to help you create an app and to get up and running with the new premium APIs.

*If you're planning to use our standard APIs instead of our premium APIs, simply follow the steps below to create an app, then refer to the "**Getting started**" guide in our documentation for next steps.*

## Delete key from Git? Regenerate key on Twitter?

Since Git remembers the whole history, the key can still be found later. The simplest is to check the [file changes](#), or commit detail page. If that key is on a test account, things should be fine. However, if that key is on a personal account, you may want to [regenerate one pair on Twitter](#).

## **datetime**

# Selenium

- Selenium
  - Basic steps
    - Step 1 install selenium
    - Step 2 install webdriver and put it in the project's working directory
    - Step 3 start from opening a dynamic page web url
  - Chrome driver not found in path error
  - Find elements by Xpath
  - Find elements by css selector
  - Infinite scroll in a single page

Selenium module can be used in dynamic page web scraping on Windows Systems. It supports Firefox, IE, Chrome and Remote.

## Basic steps

### Step 1 install selenium

pip install selenium

### Step 2 install webdriver and put it in the project's working directory

For example, I use Chrome, so I need download a ChromeDriver. Download link:

<https://sites.google.com/a/chromium.org/chromedriver/home>

### Step 3 start from opening a dynamic page web url

```
browser = webdriver.Chrome()
browser.get('http://money.cnn.com/search/index.html?sortBy=date&primaryType=mixed&search=Search&query=trade%20war')
time.sleep(10)
```

For detailed use, refer to the Selenium documentation: <https://selenium-python-zh.readthedocs.io/en/latest/index.html>

For more selenium examples, you can refer [here](#).

## Chrome driver not found in path error

Message: 'chromedriver' executable needs to be in PATH

This is common error. You can download [Chrome Driver](#) and put it in your [Current Working Folder](#). Note, for venv and jupyter user, this is not your project root folder, or the venv folder. It is the path at which you issued the `jupyter notebook` command -- a.k.a. "current working folder".

MAC user has a shortcut to install chromedriver in the path.

```
brew cask install chromedriver
```

## Find elements by Xpath

For `XPath` method, XPath uses path expressions to select nodes or node-sets in an XML document. The following are basic expression rules and expression path examples. For a more detailed usage, you can check out this [tutorial](#).

Basic expression rules:

Expression	Description
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the parent of the current node
@	Selects attributes

Path Expression examples:

Path Expression	Results
/bookstore/book[1]	Selects the first book element that is the child of the bookstore element.
//book	Selects all book elements no matter where they are in the document
//@lang	Selects all attributes that are named lang
//title[@lang='en']	Selects all the title elements that have a "lang" attribute with a value of "en"

## Find elements by css selector

When we use the find elements in css selector. Two common error are that:

1. the element cannot be located.
2. matching imprecisely. We can find the element, but that's not what we want.

The following are useful workflow that can help us debug when encounter those problems:

1. CSS Selector needs to be precise some time. Being too broad may risk matching something else.
2. The best practice is to use `select_elements_x` method (notice s) first to verify if the matching is precise. If it is, use the non-"s" version to find the element. If it is not, check if the elements come in specific order. If so, one can use list navigation to locate the precise one.

For more details explanation, you can refer to this [example](#).

## Infinite scroll in a single page

Usually, the website will not allow you to do infinitely scrolling, there usually some limitations. Therefore we can do some manually scroll testing and see how many times we can scroll. After that we can use a for loop to adjust the scrolling times by your observation.

```
for i in range(1,100): #change time by your observation
 try:
 time.sleep(2)
 #scraping codes here
 browser.execute_script('window.scrollTo(0, document.body.scrollHeight);')
 except:
 ...
```

**Note:** When executing web emulation, remember to set a sleep time to avoid risen error. It's suggested to set it as 2 seconds if there is a big loading work. You can increase or reduce by your own observation.

Another solution is using XHR. When a page adopts an infinite scroll design, asynchronous data loading is inevitable. When you encounter those cases, network trace analysis may give more concise solution. There are usually XHR interfaces. You don't even need dynamic crawling (browser simulation). You can refer [here](#) for a detailed example.

## wordcloud

- [wordcloud](#)
  - [Display Chinese characters when plotting tag cloud](#)

### Display Chinese characters when plotting tag cloud

```
wc = wordcloud.WordCloud(background_color="white", font_path='/Library/Fonts/Songti.ttc', max_words=3000,max_font_size=60, random_state=40)
```

When generate tag cloud, we need give a `font path` which support Chinese characters.

Usually we use `SimHei` and `Songti` to display Chinese characters. There are two ways to get the font path.

1. `command + blank` to open spotlight, search `font book`, select one font that support Chinese characters. The font path will be like this: `/Library/Fonts/Songti.ttc` or `/Library/Fonts/SimHei.ttf`.
2. If there is no font that can support Chinese character, you need to download the font, and install, then get the font path by step 1. Or you can put the font in the current folder where your jupyter notebook are. The font path will be like this `"/simhei.ttf"`.

## Install pyproj error

`pyproj` is the dependency of `geopandas`, which is further dependence for some of `plotly`'s mapping functions. When installing the library in Python 3.7, part of the error messages look like:

```
/usr/local/Cellar/python/3.7.0/Frameworks/Python.framework/Versions/3.7/include/python3.7m/pystate.h:238:15
: note: 'curexc_traceback' declared here
 PyObject *curexc_traceback;
 ^
15 warnings and 15 errors generated.
error: command 'clang' failed with exit status 1
```

#137@`pyproj` shows that this is compatibility issue with Python 3.7. This is already fixed on the latest `pyproj` library, so one can install from code/ compile from code to solve the issue. And another error appears at this stage:

```
_proj.c does not exist in a repository copy.
ImportError: Cython must be installed in order to generate _proj.c
to install Cython run `pip install cython`
```

Installing `Cython` can solve the problem. So the complete solution for this issue is:

```
pip install cython
pip install git+https://github.com/jswhit/pyproj.git
pip install geopandas
```

More discussion and further questions can go to [#87](#).

# Other Frequently Asked Questions

- Other Frequently Asked Questions
  - Notes
  - Environment
    - Two basic modes of executing codes: script and interactive
    - Setup virtual venv and install Jupyter Notebook
    - Install modules in venv, it's equivalent to install in Jupyter
  - Tricks & Hot key
    - Restart kernel helps
  - Frequently asked bugs
    - No such file or directory in jupyter or file xxx does not exist
  - Encoding & Decoding
    - U+FEFF encoding issue
    - Csv writer newline
    - Cannot read csv files downloaded from a website
    - Cannot import list of files
    - Expecting value: line 1 column 1 (char 0)
  - Data extraction - slice elements when one of them may be None

## Notes

This is the page for "other" uncategorised FAQs. When some problems of a new area appear, they are likely to arrive here as first stop. When there are enough number of Q/As under certain topic, they will be diverted to the page dedicated to that area.

The whole catalogue is the first entry point for usual readers. It can be found here: [FAQ Catalogue](#)

## Environment

### Two basic modes of executing codes: script and interactive

please see [here](#)

### Setup virtual venv and install Jupyter Notebook

please see [here](#)

### Install modules in venv, it's equivalent to install in Jupyter

If you are using Jupyter for the first time, you should create virtual environment first by `pyvenv venv`. Then enter the `venv` environment by typing `source venv/bin/activate`. This is the right place where you should install all the modules you will use, after that you can enter the Jupyter by command `jupyter notebook`. Once you finish your work, use `deactivate` to exit `venv` environment. Do remember, you just need to create virtual environment once, next time when you use Jupyter, just type `source venv/bin/activate + jupyter notebook` will work.

## Tricks & Hot key

1. Exit Python interactive mode: `quit()`, `exit()` or `Control-D` on Mac, `Control-Z` on Windows.
2. Auto supplementation when executing the code files: type the first letter of the file name then `tab`
3. Indent: `tab`
4. Indent back: `tab + shift`
5. Comment single line: `#` ahead of the line
6. Comment multiple lines: select multiple lines, then type `command + /`, type again to un-comment
7. Force quit boxes - `command + option + esc`

## Restart kernel helps

In the coding process, we may keep renaming the variables or adjust the sequences of the cells. In such circumstances, there might be errors arisen. Therefore, a good practice here is always try `restarting kernels` if encounter the weird errors.

## Frequently asked bugs

### No such file or directory in jupyter or file xxx does not exist

If you download the csv from some where, and you want to import it in Jupyter notebook. you should put the csv file in the folder where your venv folder are first. Usually, it's in the user path. All the files you write and read should in this folder. Another method is to type `pwd` in Jupyter, it will return the path, where you should put the file in.

## Encoding & Decoding

### U+FEFF encoding issue

CSV Sample output:

```
['U+FEFF/name', 'id', 'gender', 'location', 'phone']
['Chico', '1742', 'M', 'KLN', '3344']
['Ri', '1743', 'F', 'LOS', '5168']
['Ivy', '1655', 'F', 'MK', '7323']
```

In some cases, the csv you read may have some encoding issue like the above. `U+FEFF` is the byte order mark, or BOM, and is used to tell the difference between big- and little-endian UTF-16 encoding. To omit BOM, just add a encoding line in the `with....open` command, as `with open('chapter4-example-name_list.csv',encoding='utf-8-sig')` as `f: .` Further reading about this [issue](#).

### Csv writer newline

For more explanation, please refer to the documentation on the `open` function

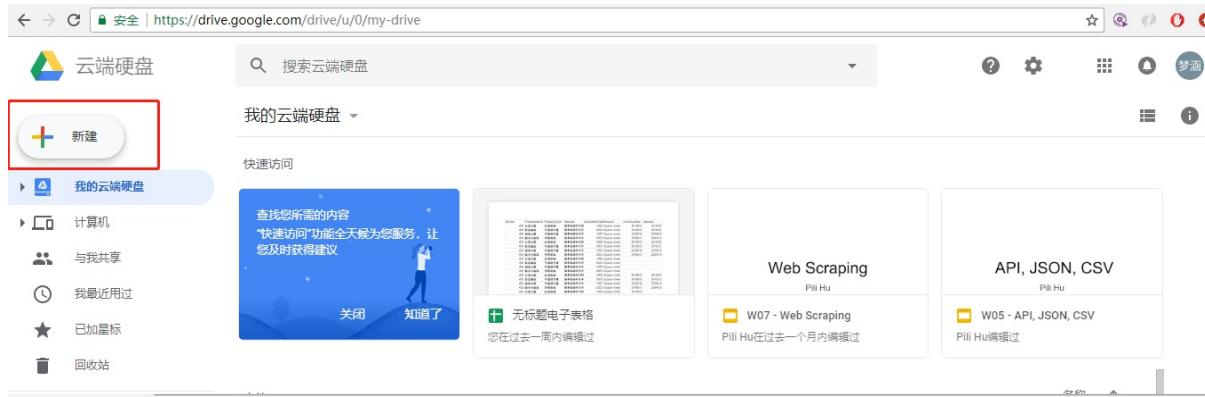
### Cannot read csv files downloaded from a website

This error is usually raised as the following:

```
UTF-8 cannot decode byte 0xff in position x: invalid byte
```

Solution:

First, try to copy and paste the contains/data to Google drive with a new sheet, and downloaded it as a new csv.



The screenshot shows the Google Drive interface. A red box highlights the '新建' (New) button in the top-left corner of the sidebar. The sidebar also includes links for '我的云端硬盘' (My Cloud Drive), '计算机' (Computer), '与我共享' (Shared with me), '我最近用过' (Recently used), '已加星标' (Starred), and '回收站' (Trash). The main area displays a search bar, a '我的云端硬盘' (My Cloud Drive) dropdown, and several items: '无标题电子表格' (Untitled spreadsheet) by 'Pili Hu' (last edited 1 week ago), 'Web Scraping' by 'Pili Hu' (last edited 1 month ago), and 'API, JSON, CSV' by 'Pili Hu' (last edited 1 month ago).


The screenshot shows the Google Sheets interface. A red box highlights the 'Google 表格' (Google Sheets) icon in the sidebar. A modal dialog box is open, prompting the user to choose a template for a new spreadsheet. Options include '空白电子表格' (Blank spreadsheet) and '来自模板' (From template). The background shows a preview of a spreadsheet with data.


The screenshot shows a Google Sheets document titled '无标题电子表格'. The spreadsheet contains 12 rows of data, with columns labeled A through I. The data includes dates like '2006/8/31' and '2007/4/27', and various property details such as '白馬大廈' and '財富廣場'. The last row is empty.

	A	B	C	D	E	F	G	H	I	J
1	DeclareDate	EndDate	SecurityID	Symbol	PropertyName	PropertyType	Address	CompletedYear	Areaunit	Co
2	2006/8/31	2006/6/30	2.12E+11	405	白馬大廈	批發商場	廣東省廣州市越秀區	1990	Square meter	
3	2006/8/31	2006/6/30	2.12E+11	405	財富廣場	甲級寫字樓	廣東省廣州市天河區	2003	Square meter	
4	2006/8/31	2006/6/30	2.12E+11	405	城建大廈	甲級寫字樓	廣東省廣州市天河區	1997	Square meter	
5	2006/8/31	2006/6/30	2.12E+11	405	維多利廣場	零售商場	廣東省廣州市天河區	2003	Square meter	
6	2007/4/27	2006/12/31	2.12E+11	405	白馬大廈	批發商場	廣東省廣州市越秀區	1990	Square meter	
7	2007/4/27	2006/12/31	2.12E+11	405	財富廣場	甲級寫字樓	廣東省廣州市天河區	2003	Square meter	
8	2007/4/27	2006/12/31	2.12E+11	405	城建大廈	甲級寫字樓	廣東省廣州市天河區	1997	Square meter	
9	2007/4/27	2006/12/31	2.12E+11	405	維多利廣場	零售商場	廣東省廣州市天河區	2003	Square meter	
10	2007/8/27	2007/6/30	2.12E+11	405	白馬大廈	批發商場	廣東省廣州市越秀區	1990	Square meter	
11	2007/8/27	2007/6/30	2.12E+11	405	財富廣場	甲級寫字樓	廣東省廣州市天河區	2003	Square meter	
12										



Then, we can successfully read the data by the usual way.

```
-*- coding: utf-8 -*-
import pandas as pd
df = pd.read_csv('REITs.csv', 'rb')
```

```
In [8]: df = pd.read_csv('REITs.csv')
In [9]: df
Out[9]:
 DeclareDate EndDate SecurityID Symbol PropertyName PropertyType Address CompletedYear Areaunit ConstrucArea Usearea RentalArea un
 0 2006/8/31 2006/6/30 2.120000e+11 405 白馬大廈 批發商場 廣東省廣州市越秀區站南路16號 1990 Square meter 50199.3 48100.6 50128.9 HKD'00
 1 2006/8/31 2006/6/30 2.120000e+11 405 財富廣場 甲級寫字樓 廣東省廣州市天河區體育東路114-118號 2003 Square meter 40356.2 30752.3 40356.2 HKD'00
```

## Cannot import list of files

For example, using `os.listdir` to import a list of files.

```
def read_txt(path): #read files and get content
 all_text = []
 for file in os.listdir(path):
 f=open(file,"r")
 contents= f.read()
 all_text.append(contents)
 all_words = "".join(all_text)
 for ch in '\s+\.\!\/_,\$%^+(\"")]+|[+—()?: \t"\'':
 words = all_words.replace(ch, " ")
 return words
words = read_txt("text/") #pass your own file path that include list of .txt
```

Error: 'utf-8' codec can't decode byte 0x80 in position 3131

Even though your files are encodes by `utf-8`, somehow it may still rising error above, you can using following solution in the terminal:

```
cd to desktop/chicoxyc/text #cd to the folder where your txt files are
ls -a #list all the files
rm .DS_Store #delete .DS_Store if there is any
```

## Expecting value: line 1 column 1 (char 0)

It's a common error of `JSONDecodeError` meaning that there is no JSON can be parsed. You need to check the response object before doing further processing.

For some specify example solutions, you can refer to [here](#).

**NOTE:** This is not necessarily caused by encoding problem. sometimes malformed JSON format will also cause the problem. Network problem may also lead to this type of error. The response object is not a valid JOSN. It may be some error code.

## Data extraction - slice elements when one of them may be None

We can use list slicing, if...else or try...except to test the boundary condition and separate different elements we want.

example: following is the content of series `df['time_countries']`, and we want to get country and time separately.

```
上映时间: 1993-01-01
上映时间: 1994-10-14(美国)
上映时间: 1953-09-02(美国)
上映时间: 1994-09-14(法国)
上映时间: 1972-03-24(美国)
上映时间: 1998-04-03
上映时间: 1993-07-01(中国香港)
上映时间: 2001-07-20(日本)
上映时间: 1940-05-17(美国)
上映时间: 1939-12-15(美国)
```

You might notice that some entries have no country, therefore we have to handle two different situations. `if...else` here can helps. We can see that countries starts from index 15 in the string, therefore, we can use 15 to set the condition.

```
def get_country(x):
 #if length > 15, we first separate by (, get the second parts and then separate), get the first part, which is pure countries name we want.
 #if length < 15, there is no countries, we return blank
 if len (x) > 15:
 return x.split('(')[1].split(')')[0]
 else:
 return ''
df['country'] = df['time_countries'].apply(get_country)

def get_time(x):
 #every entries have time, therefore we dont need set condition here. We first separate by :, get the second parts and then separate (, get the first part, which is pure time we want.
 return x.split(':')[1].split('(')[0]
df['time'] = df['time_countries'].apply(get_time)
```

