IOWA STATE UNIVERSITY

Department of Computer Science

Data Acquisition and Processing

Adisak Sukul, Ph.D.,

Lecturer,

Department of Computer Science,

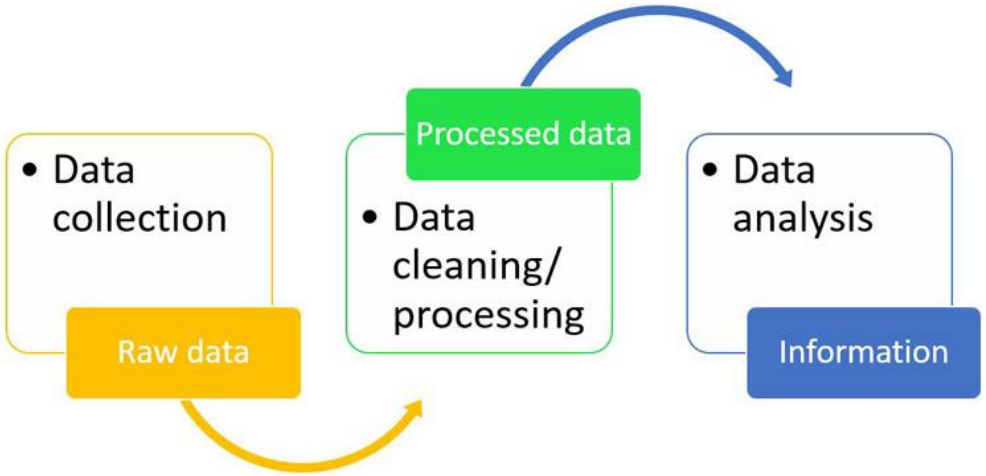
adisak@iastate.edu

http://web.cs.iastate.edu/~adisak/Bigdata/

Topics

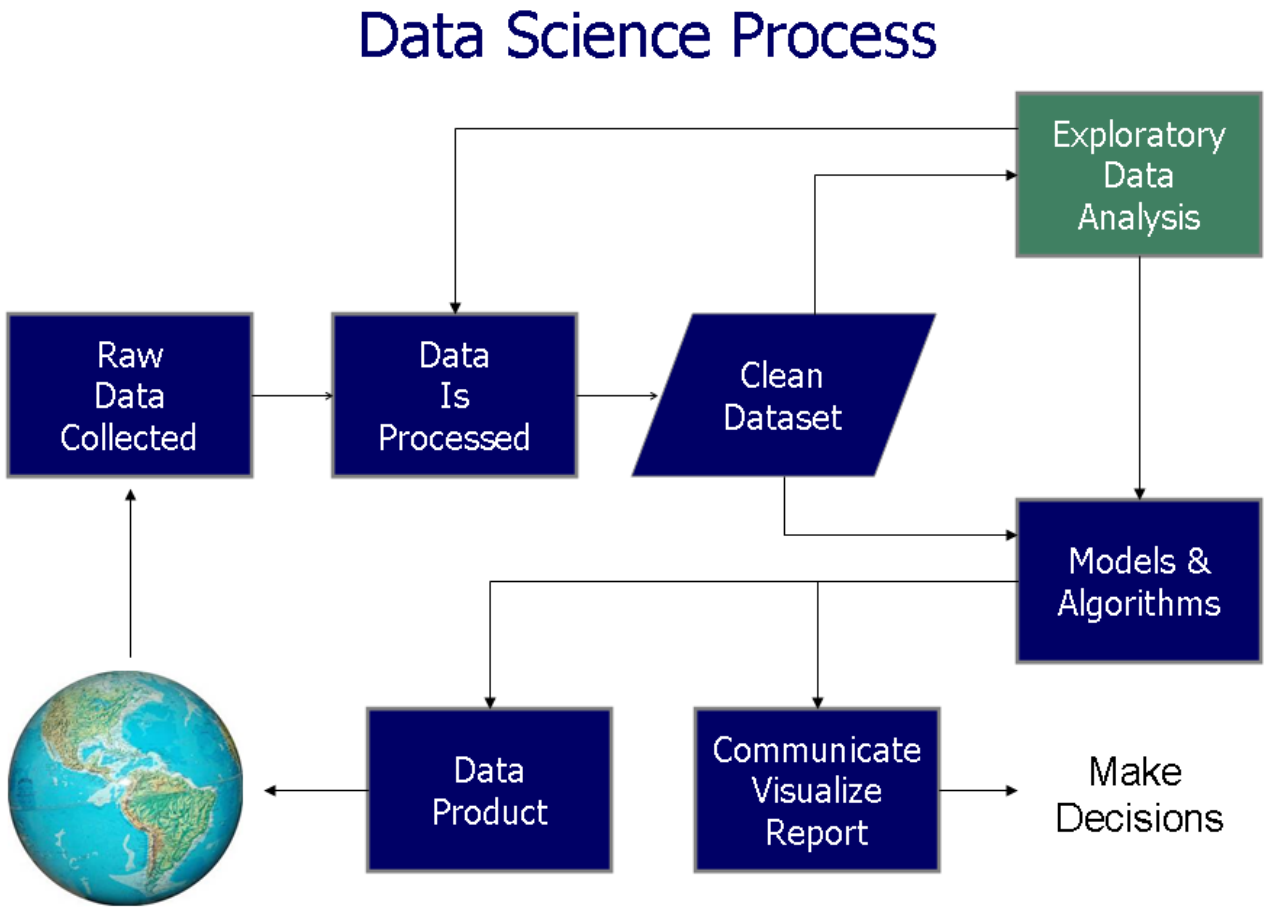
http://web.cs.iastate.edu/~adisak/Bigdata/

- Data Acquisition
- Data Processing



- Introduction to:
  - Jupyter/iPython Notebook
  - Anaconda
  - Web scraping using BeautifulSoap4
    - Project 1: Web scraping
    - Your own web scraping
  - Exploratory analysis in Python using Pandas
    - Project 2: tabular file processing
      - In Excel
      - without Pandas ← you should try this one
      - With Pandas

What are we doing?



https://en.wikipedia.org/wiki/Data_science

Installing Python and Anaconda package

Download Anaconda now

https://www.continuum.io/downloads

Anaconda for Windows

PYTHON 2.7	PYTHON 3.5
WINDOWS 64-BIT GRAPHICAL INSTALLER	WINDOWS 64-BIT GRAPHICAL INSTALLER
Windows 32-bit Graphical Installer	Windows 32-bit Graphical Installer

Behind a firewall? Use these zipped Windows installers.

Windows Anaconda Installation

1. Download the graphical installer.
2. Optional: **Verify data integrity with MD5**.
3. Double-click the .exe file to install Anaconda and follow the instructions on the screen.

Anaconda for OS X

PYTHON 2.7	PYTHON 3.5
MAC OS X 64-BIT GRAPHICAL INSTALLER	MAC OS X 64-BIT GRAPHICAL INSTALLER
Mac OS X 64-bit Command-Line installer	Mac OS X 64-bit Command-Line installer

OS X Anaconda Installation

Choose either the graphical installer or the command line installer for OS X.

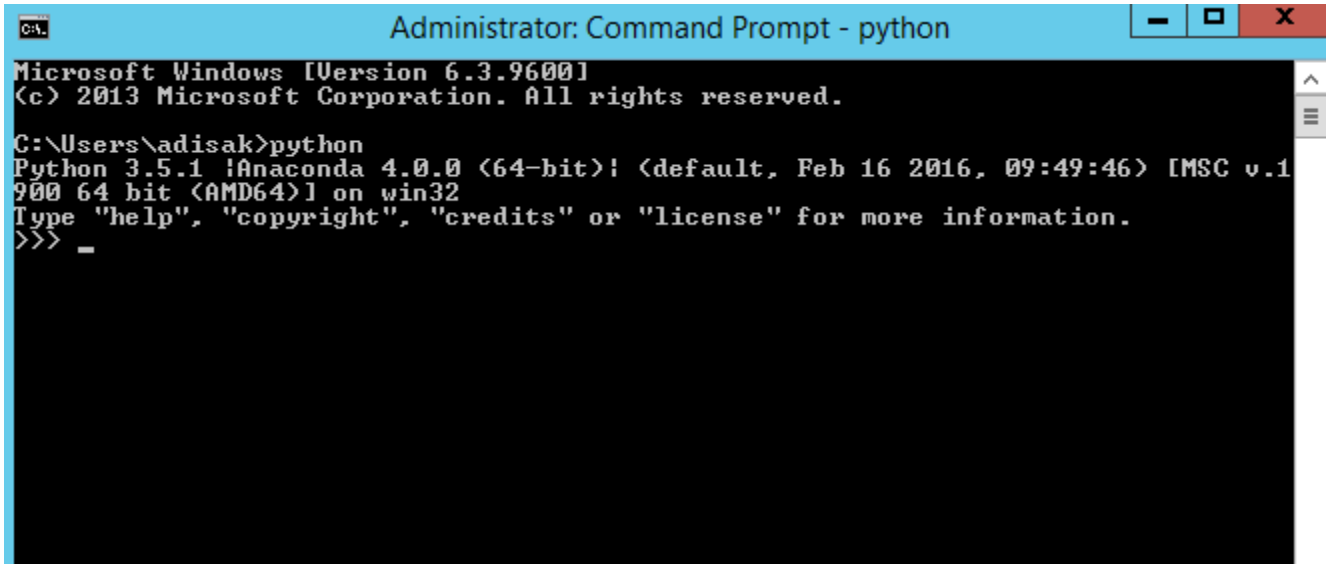
Graphical Installer:

1. Download the graphical installer.
2. Double-click the downloaded .pkg file and follow the instructions.

Command Line Installer:

1. Download the command line installer.
2. Optional: **Verify data integrity with MD5**.
3. In your terminal window, type one of the below and follow the instructions:

- Restart your system after installation complete


Administrator: Command Prompt - python
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Users\adisak>python
Python 3.5.1 |Anaconda 4.0.0 (64-bit)| (default, Feb 16 2016, 09:49:46) [MSC v.1
900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>

- Update packages

>conda update conda

>conda update jupyter

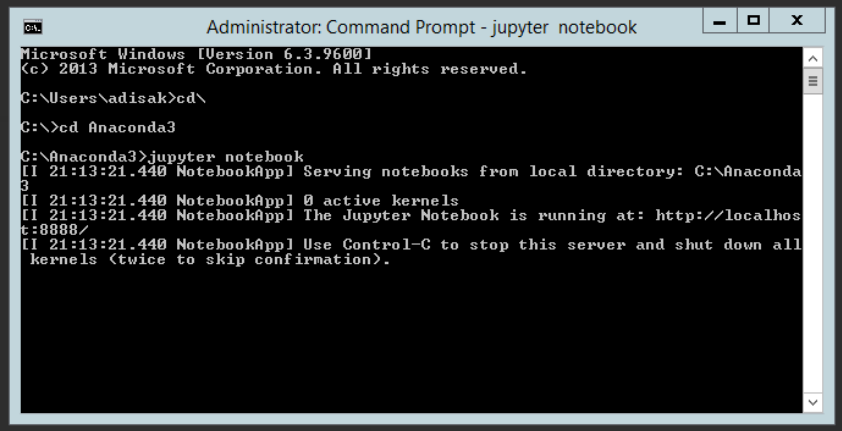
>conda update scikit-learn

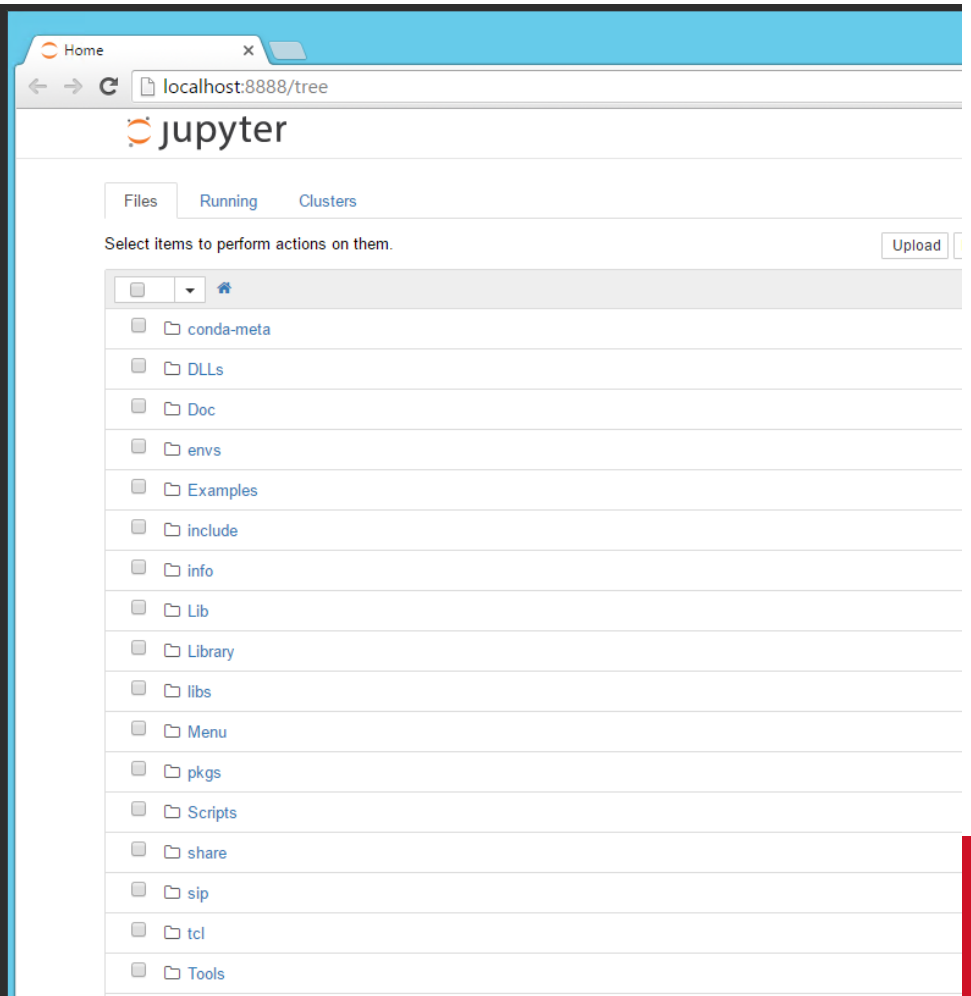
Command Prompt

```
C:\Anaconda3>conda update jupyter
Fetching package metadata .......
Solving package specifications: ..........

Package plan for installation in environment C:\Anaconda3:

The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    jupyter-1.0.0              |           py35_3           3 KB

The following packages will be UPDATED:

    jupyter: 1.0.0-py35_2 --> 1.0.0-py35_3

Proceed ([y]/n)? y

Fetching packages ...
jupyter-1.0.0- 100% |################################| Time: 0:00:00 665.20 kB/s
Extracting packages ...
[      COMPLETE      ]|#################################################| 100%
Unlinking packages ...
[      COMPLETE      ]|#################################################| 100%
Linking packages ...
[      COMPLETE      ]|#################################################| 100%
```

Start a Jupyter Notebook

- start Jupyter notebook by writing “jupyter notebook” on your terminal / cmd





Jupyter Notebook

- You can name a iPython notebook by simply clicking on the name – Untitled and rename it.

- The interface shows In [*] for inputs and Out[*] for output.

- You can execute a code by pressing “Shift + Enter”
- or “ALT + Enter”, if you want to insert an additional row after.

A little bit about me..

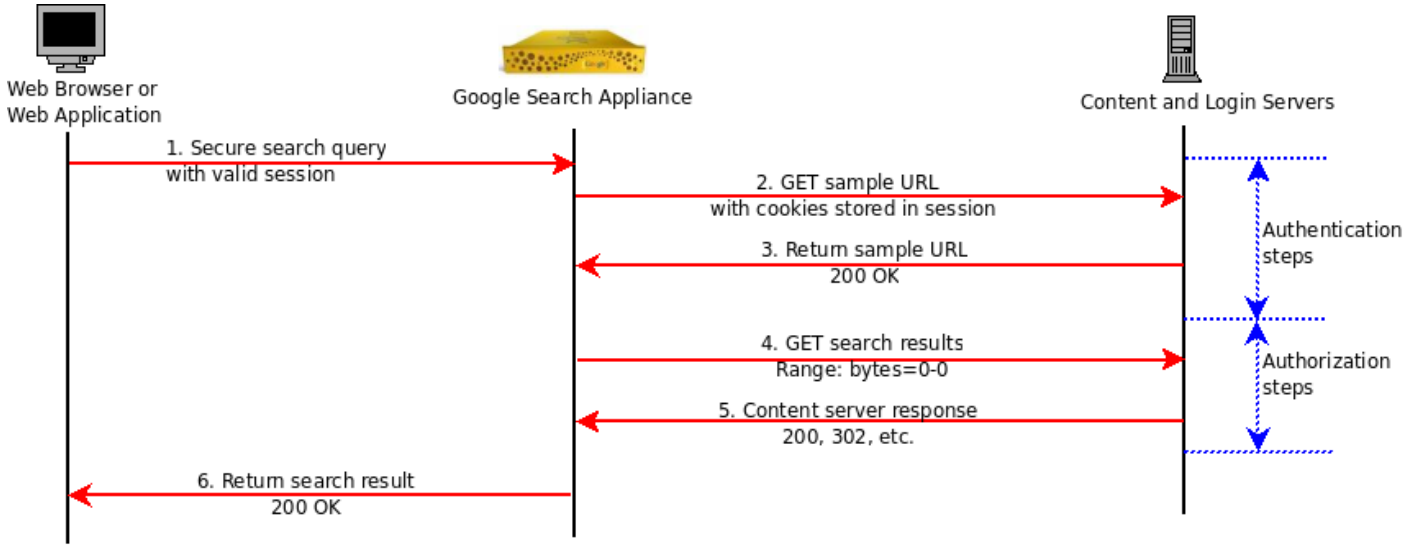
What I used to do…

- Cofounder of few "Startup" dealing with academic and libraries content
  - Doing basically everything…

My first we scraping project.

About me and Big Data projects

- Predicting Parliament Library of Thailand search patterns
  - Google search appliance vs. create your own Solr
    - Crawling problems
    - Data cleaning problems



About me and Big Data projects

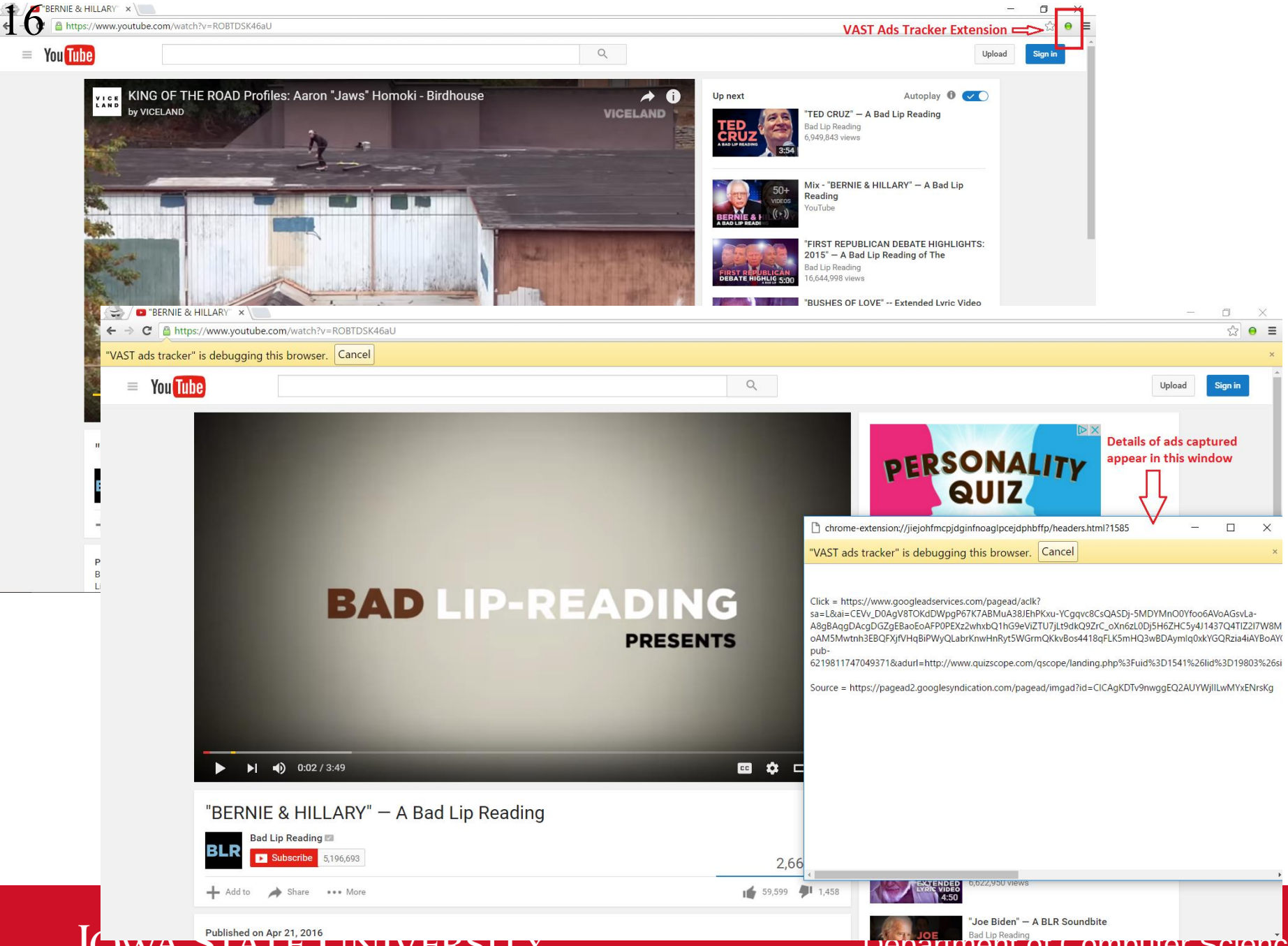
- Detect micro-targeting political ads on Youtube
- NSF grant
  - Department of Computer science and department of Political science
  - Detect micro-targeting ads based on user geographic

One of the current data-driven project.. CyAds tracker

- **Phase 1:**
  - Tracker application on single server
  - 2 groups of ads collections: with and without "political interest" profiles
  - 4,000 videos process per batch
  - 5 concurrent bots
- **Phase 2:**
- Create a tracker app as Chrome extension, become distributed.
- Collecting much more ads:
  - **methodology for collection of YouTube ads**
    - **Gender, Age, Language**
    - **Partisanship and other interests**
    - **Location**
  - 63 user profile
  - total of 207 bots
  - Our bots made 500K video request per day

CyAds– workflow

- The Ads capturing will work as follows:
- - The Ads Tracker extension when enabled, will monitor the network traffic of the current tab and look for video ads.
- - If it finds any, the ad details are displayed in the Tracker window.
- - Also whenever it finds such ads, it sends the following details to a jsp page residing in the server via AJAX post method.
- = Source Youtube video link
- = Ad Details(Source link, click through link, ad type)
- = Ad metadata(whole response xml)
- - On receiving the ads data, the jsp page(residing on server) will store them locally on the server. So all the ads related data will be consolidated on the server.


VAST Ads Tracker Extension
"VAST ads tracker" is debugging this browser.
Cancel
Details of ads captured appear in this window
chrome-extension://jiejohfmcpjdginfnoaglpcejdphbffp/headers.html?1585
Click = https://www.googleadservices.com/pagead/aclk?sa=L&ai=CEVv_D0AgV8TOKdDWpgP67K7ABMuA38JEhPKxu-YCgqvc8CsQASDj-5MDYMnO0Yfoo6AVoAGsvLa-A8gBAqgDAcgDGZgEBaoEoAFP0PEXz2whxbQ1hG9eViZTU7JLt9dkQ9ZrC_oXn6zL0Dj5H6ZHC5y4J1437Q4TIZ2I7W8M
Source = https://pagead2.googlesyndication.com/pagead/imgad?id=CICAgKDTv9nwggEQ2AUYWjIILwMYxENrsKg
BAD LIP-READING
PRESENTS
"BERNIE & HILLARY" — A Bad Lip Reading
Bad Lip Reading
Subscribe
5,196,693
Published on Apr 21, 2016
KING OF THE ROAD Profiles: Aaron "Jaws" Homoki - Birdhouse
by VICELAND
Up next
Autoplay
"TED CRUZ" — A Bad Lip Reading
Mix - "BERNIE & HILLARY" — A Bad Lip Reading
"FIRST REPUBLICAN DEBATE HIGHLIGHTS: 2015" — A Bad Lip Reading of The
"Joe Biden" — A BLR Soundbite

Data acquisition

- Web scraping is a fun example of data acquisition ☺

Several Python modules that make web scraping easy

- **webbrowser**. Comes with Python and opens a browser to a specific page.
- **Requests**. Downloads files and web pages from the Internet.
- **Beautiful Soup**. Parses HTML, the format that web pages are written in.
- **Selenium**. Launches and controls a web browser. Selenium is able to fill in forms and simulate mouse clicks in this browser.

Let’s have some scraping rules

- You should check a site's terms and conditions before you scrape them. Always honor their robots.txt
  - http://stackoverflow.com/robots.txt
  - https://en.wikipedia.org/robots.txt
  - https://www.youtube.com/robots.txt

- Be nice - A computer will send web requests much quicker than a user can. Make sure you space out your requests a bit so that you don't hammer the site's server.
- Scrapers break - Sites change their layout all the time. If that happens, be prepared to rewrite your code.
- Web pages are inconsistent - There's sometimes some manual clean up that has to happen even after you've gotten your data.

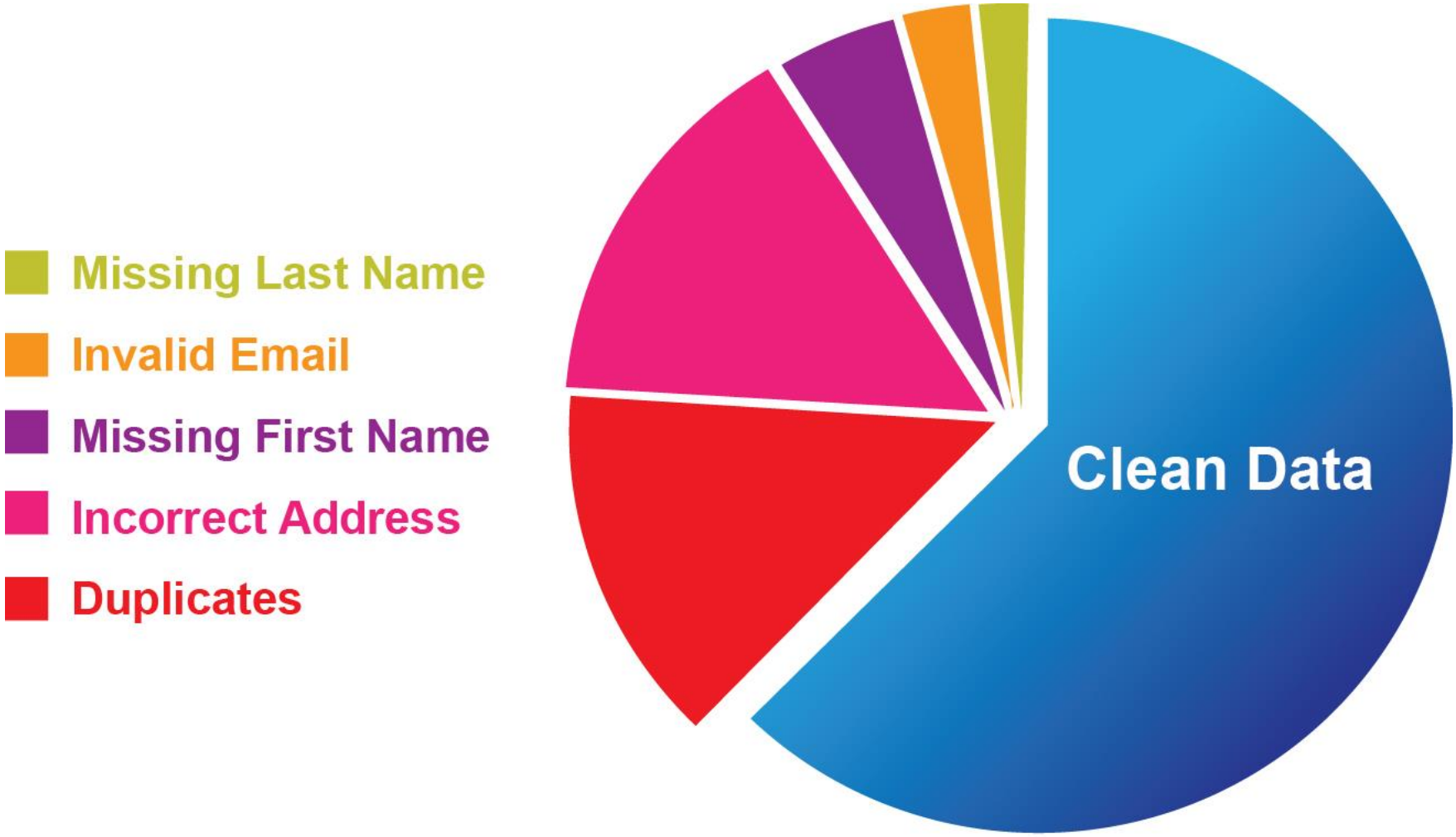
- Web scraping project.

Your turn 1

- Clean yourself up!
- Find list of people on your department webpage that contain your name with some info.
- Write a code to get a names and info of people from that page.

Data processing

- Data cleaning



Study above indicates that up to 30% of your CRM data could be wrong.

Python (standard) Data Structures

- **Lists** – Lists are one of the most versatile data structure in Python.
- A list can simply be defined by writing a list of comma separated values in square brackets. Lists might contain items of different types, but usually the items all have the same type.
- Python lists are mutable and individual elements of a list can be changed.

Lists

A list can be simply defined by writing comma separated values in square brackets.

```
In [1]: squares_list = [0,1,4,9,16,25]
```

```
In [2]: squares_list
Out[2]: [0, 1, 4, 9, 16, 25]
```

Individual elements of a list can be accessed by writing the index number in square bracket. Please note that the first index of a list is 0 and not 1

```
In [3]: squares_list[0] #Indexing returns the item
Out[3]: 0
```

A range of script can be accessed by having first index and last index

```
In [4]: squares_list[2:4] #Slicing returns a new list
Out[4]: [4, 9]
```

A Negative index accesses the list from end

```
In [5]: squares_list[-2] #It should return the second last element in the list
Out[5]: 16
```

A few common methods applicable to lists include: append() extend() insert() remove() pop() count() sort() reverse()

- **Tuples** – A tuple is represented by a number of values separated by commas.
- Tuples are **immutable** and the output is surrounded by parentheses so that nested tuples are processed correctly. Additionally, even though tuples are immutable, they can hold mutable data if needed.
- Since Tuples are immutable and can not change, they are faster in processing as compared to lists. Hence, if your list is unlikely to change, you should use tuples, instead of lists.
- It just faster OK!

Tuples

A tuple is represented by a number of values separated by commas.

```
In [10]: tuple_example = 0, 1, 4, 9, 16, 25
```

```
In [11]: tuple_example #output would be enclosed in paranthesis
Out[11]: (0, 1, 4, 9, 16, 25)
```

```
In [12]: tuple_example[2] #Single elements can be accessed in similar fashion
Out[12]: 4
```

```
In [13]: tuple_example[2] = 6 #Tuples are immutable and hence this should result in error

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-6c410e816018> in <module>()
----> 1 tuple_example[2] = 6 #Tuples are immutable and hence this should result in error

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

- Dictionary is an unordered set of **key: value** pairs, with the requirement that the keys are unique (within one dictionary).
- A pair of braces creates an empty dictionary: { }.
- It good for a reference table.

Dictionary

A dictionary is an unordered set of key: value pairs, with the requirement that the keys are unique (within one dictionary). A pair of braces creates an empty dictionary: {}.

In [20]:
```
extensions = {'Kunal': 9073, 'Tavish' : 9128, 'Sunil' : 9223, 'Nitin' : 9330}
extensions
```

Out[20]:
```
{'Kunal': 9073, 'Nitin': 9330, 'Sunil': 9223, 'Tavish': 9128}
```

In [22]:
```
extensions['Mukesh'] = 9150
extensions
```

Out[22]:
```
{'Kunal': 9073, 'Mukesh': 9150, 'Nitin': 9330, 'Sunil': 9223, 'Tavish': 9128}
```

In [23]:
```
extensions.keys()
```

Out[23]:
```
['Sunil', 'Tavish', 'Kunal', 'Mukesh', 'Nitin']
```

Project 2 – simple file processing

- Without Pandas

- It all start from Excel
  - Let's see some excel
- Excel can solve this in 5 mins
- Python would take hours, due to lack of proper data structure to handle tabular data
  - Let's see the struggle

Pandas

Pandas

- **pandas** is a **Python** package providing **fast**, **flexible**, and **expressive data structures** designed to make working with "relational" or "labeled" data both easy and intuitive.

- Pandas is the data-munging Swiss Army knife of the Python world. Often you know how your data should look but it's not so obvious how to get there, so Pandas should help with that data manipulation.

Pandas data structure

pandas introduces two new data structures to Python - Series and DataFrame, both of which are built on top of NumPy (this means it's fast).

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
pd.set_option('max_columns', 50)
%matplotlib inline
```

Series

- A Series is a one-dimensional object similar to an array, list, or column in a table.
- It will assign a labeled index to each item in the Series. By default, each item will receive an index label from 0 to N, where N is the length of the Series minus one.

```
# create a Series with an arbitrary list
s = pd.Series([7, 'Heisenberg', 3.14, -1789710578, 'Happy Eating!'])
s
```

```
0                 7
1       Heisenberg
2             3.14
3      -1789710578
4    Happy Eating!
dtype: object
```

Series (Continue)

- Alternatively, you can specify an index to use when creating the Series.

```
s = pd.Series([7, 'Heisenberg', 3.14, -1789710578, 'Happy Eating!'],
              index=['A', 'Z', 'C', 'Y', 'E'])
s
```

```
A                 7
Z       Heisenberg
C              3.14
Y      -1789710578
E    Happy Eating!
dtype: object
```

- The Series constructor can convert a dictonary as well, using the keys of the dictionary as its index.

```
d = {'Chicago': 1000, 'New York': 1300, 'Portland': 900, 'San Francisco': 1100,
     'Austin': 450, 'Boston': None}
cities = pd.Series(d)
cities
```

```
Austin            450
Boston            NaN
Chicago          1000
New York         1300
Portland          900
San Francisco    1100
dtype: float64
```

DataFrame

- is a tablular data structure comprised of rows and columns, akin to a spreadsheet, database table, or R's data.frame object.
- You can also think of a DataFrame as a group of Series objects that share an index (the column names).
- For the rest of the tutorial, we'll be primarily working with DataFrames.

- To create a DataFrame out of common Python data structures, we can pass a dictionary of lists to the DataFrame constructor.

```
data = {'year': [2010, 2011, 2012, 2011, 2012, 2010, 2011, 2012],
        'team': ['Bears', 'Bears', 'Bears', 'Packers', 'Packers', 'Lions', 'Lions', 'Lions'],
        'wins': [11, 8, 10, 15, 11, 6, 10, 4],
        'losses': [5, 8, 6, 1, 5, 10, 6, 12]}
football = pd.DataFrame(data, columns=['year', 'team', 'wins', 'losses'])
football
```

	year	team	wins	losses
0	2010	Bears	11	5
1	2011	Bears	8	8
2	2012	Bears	10	6
3	2011	Packers	15	1
4	2012	Packers	11	5
5	2010	Lions	6	10
6	2011	Lions	10	6
7	2012	Lions	4	12

Numpy array

1-D Array

9	14	4	4	9	8	1	8	8	0

```
arr[3:7]
arr[0]
```

- 2-D Array

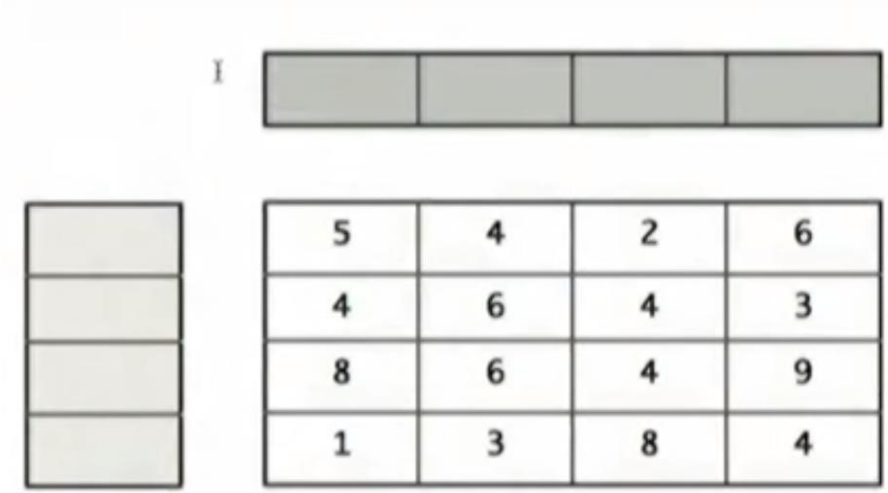
5	4	2	6
4	6	4	3
8	6	4	9
1	3	8	4

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

- Pandas is an Indexed NumPy array, with index on

A	B	C	D	E	F	G	H	I	J

9	14	4	4	9	8	1	8	8	0



- Pandas = “Dictionary based NumPy”

DataFrame

	A	B	C	D
one	5	4	2	6
two	4	6	4	3
three	8	6	4	9
four	1	3	8	4

Series

one	5
two	4
three	8
four	1

```
df['A']
```

- Select multiple columns by item

	A	B	C	D
one	5	4	2	6
two	4	6	4	3
three	8	6	4	9
four	1	3	8	4

	A	C
one	5	2
two	4	4
three	8	4
four	1	8

```
df[['A', 'C']]
```

- Slice rows

	A	B	C	D
one	5	4	2	6
two	4	6	4	3
three	8	6	4	9
four	1	3	8	4

```
df.ix['two':'four']
```

	A	B	C	D
two	4	6	4	3
three	8	6	4	9
four	1	3	8	4

- Select intersection of rows and columns

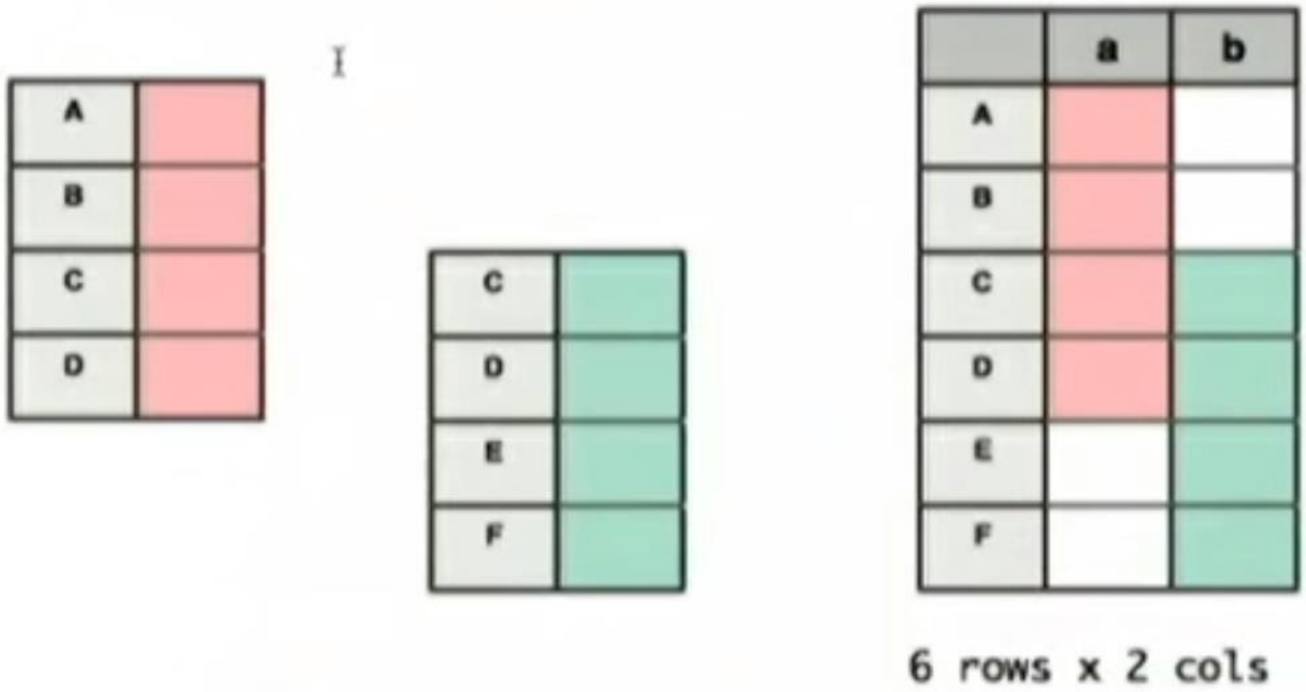
	A	B	C	D
one	5	4	2	6
two	4	6	4	3
three	8	6	4	9
four	1	3	8	4

```
df.ix[ row , column ]
```

```
df.ix[['two','three'],
      ['B','C']]
```

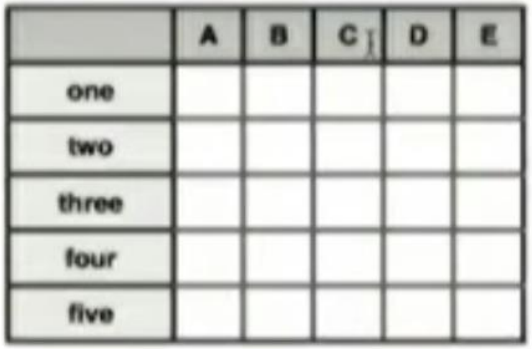
Pandas can handle a missing data.

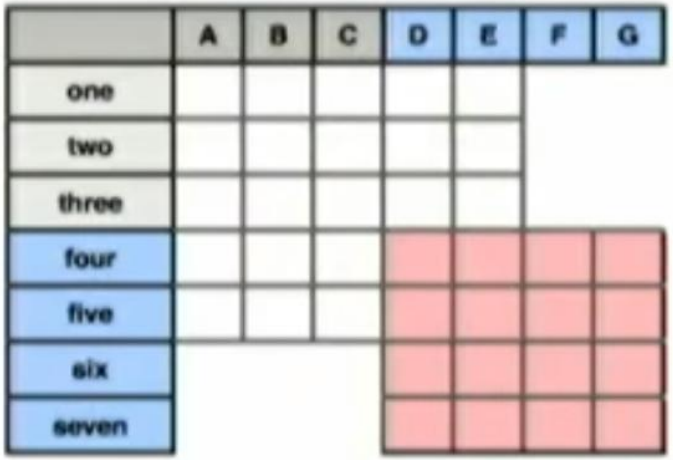
- Series -> DataFrame with missing data




```
s1 = pd.Series([1,2,3,4], index=list("ABCD"))
s2 = pd.Series([5,6,7,8], index=list("CDEF"))
df = pd.DataFrame({'a':s1, 'b':s2})
```

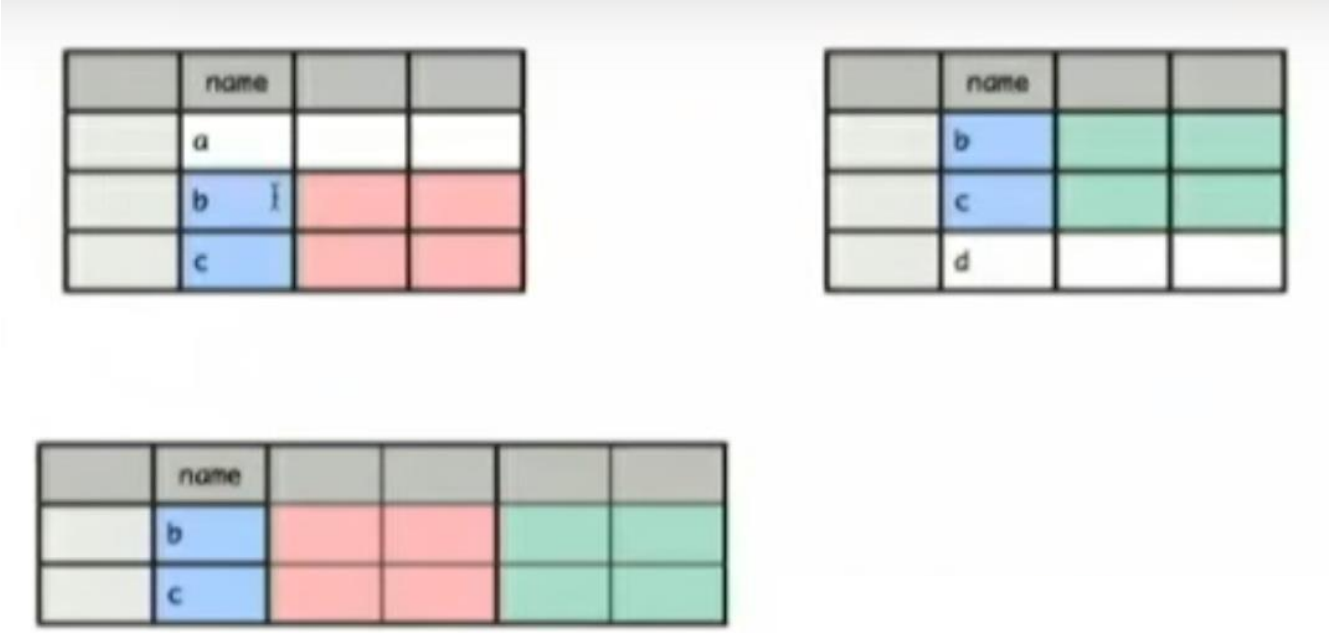
- Some Pandas's trick on slicing





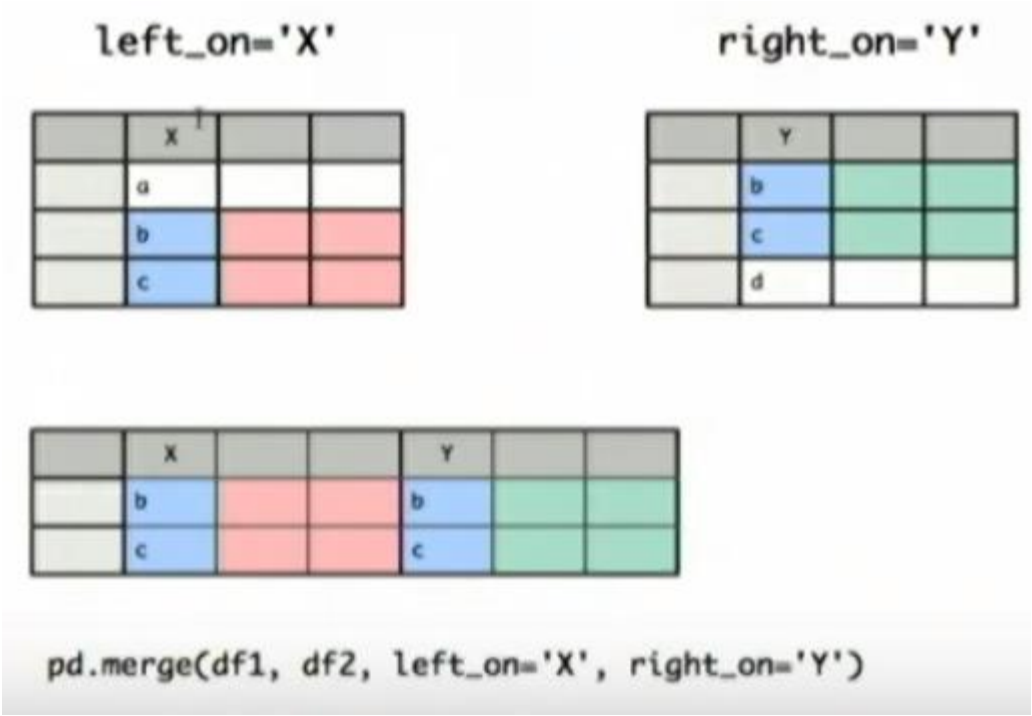

```
df.reindex(index=['four','five','six','seven'],
           columns=['D', 'E', 'F', 'G'])
```

- Merge two DataFrames.




```
pd.merge(df1, df2)
```

- Merging on different columns



Pandas tutorials

- http://pandas.pydata.org/pandas-docs/version/0.18.1/tutorials.html

- Now with Pandas, File Processing should be simple.

- **PyCon and SciPy conferences are great source!**
- https://www.youtube.com/watch?v=5il9zH9-Odg
- https://www.youtube.com/watch?v=w26x-z-BdWQ
- https://www.youtube.com/watch?v=0CFFTJUZ2dc
- https://www.youtube.com/watch?v=iWLBWqK4_ng
- https://www.youtube.com/watch?v=kSM8S76qYz0
- https://www.youtube.com/watch?v=mlt7MrwU4hY

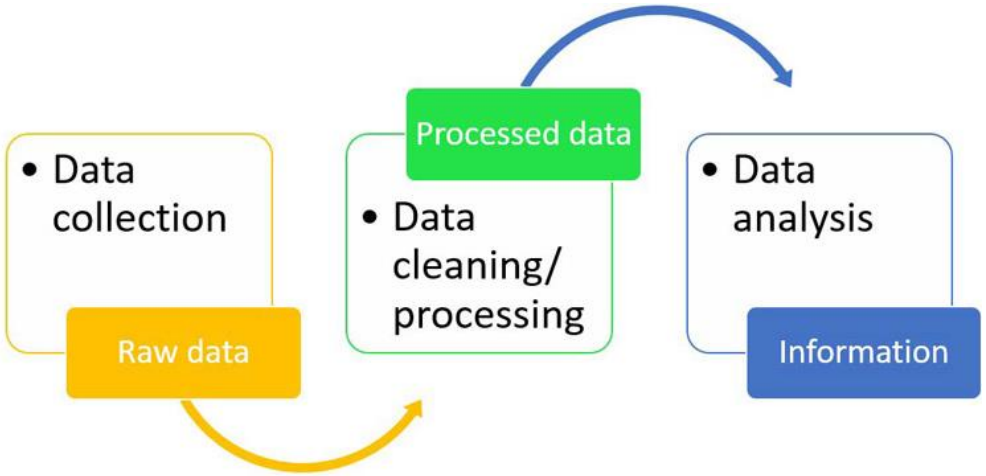
- http://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-learn-data-science-python-scratch-2/

- Web scraping
- https://blog.hartleybrody.com/web-scraping/

Thank you

http://web.cs.iastate.edu/~adisak/Bigdata/

- Data Acquisition
- Data Processing



- Introduction to:
  - Jupyter/iPython Notebook
  - Anaconda
  - Web scraping using BeautifulSoap4
    - Project 1: Web scraping
    - Your own web scraping
  - Exploratory analysis in Python using Pandas
    - Project 2: tabular file processing
      - In Excel
      - without Pandas ← you should try this one
      - With Pandas