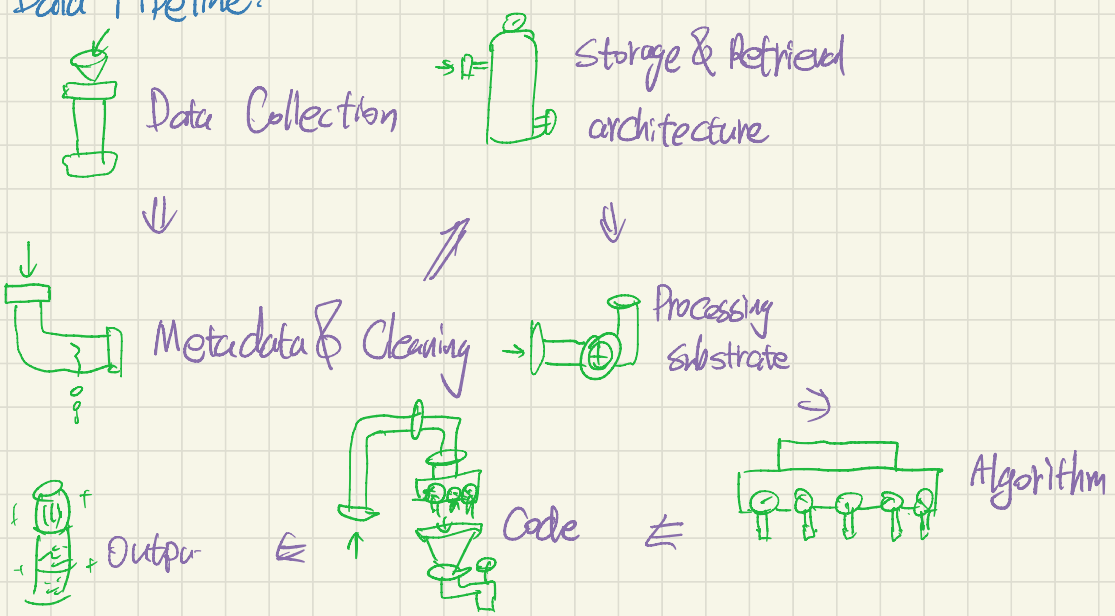
Data Engineering

Three Roles
1) Domain Expert
2) Statistician
3) Software Developer.

Data Science often needs "Data Wrangling"
- Data Wrangling: any data activity other than analysis and presentation.

Data Pipeline:



Specific Data Engineering topic:
1) Gathering
2) Transporting
3) Ethics
4) Validating
5) Transforming
6) Storing
7) Enhancing
8) Integrating
9) Monitoring

Data Gathering

Data gathering: the use of software to locate, access and extract useful data from a digital source.

Data Ethics

- Cite/reference the source of the data
- Respect data sources that do not want to be visited by tools, bots, etc.
  - Inclusion: sitemaps
  - Exclusion: robots.txt
  - Terms of Use documentation
- If there is a REST API available for this data, then access it that way
- Avoid behaving like a DDOS attack!

Common Issues for Data Gathering

- Do you have permission for gathering the data?
- is it free?
- is it in a format that you can handle
- Do you have reliable access to it?
- Are there bandwidth limits?
- Security Issues

robots.txt

- A robots.txt file tells search engine crawlers which URLs the crawler can access on your site.
- basic grammar:
  - User-agent: xxx
    Disallow: xxx
    Allow: xxx

  User-agent: *
  Disallow: /

  User-agent: *
  Disallow: /seo/
  Allow: /seo/*.html

Common Case: Web Pages

- key technology: BeautifulSoup

BeautifulSoup

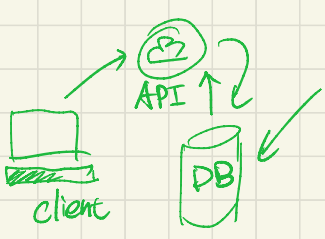
- BeautifulSoup is a Python library for pulling data out of HTML and XML files.

Common Case: REST APIs (a style)

- REpresentational State Transfer (broad)
- JSON
  JavaScript Object Notation

REST API

- GET
  POST
  GET
  PUT
  DELETE



Data Gathering Activity:

This lab touches on:
1) Data extraction from the web using Python's Beautiful Soup module
2) Data transformation using Python's Pandas library
3) Data visualization using Python's Matplotlib library

Before we start:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline   // It's a magic function in IPython.

from urllib.request import urlopen
from bs4 import BeautifulSoup

url = "http://www.hubertiming.com/results/2017GPTR10K"
html = urlopen(url)

# Parse html
soup = BeautifulSoup(html, 'lxml')
type(soup)

# Soup object allows us extract interesting information
title = soup.title
print(title)

# We can also get the text of the webpage and quickly print it out to check if it's what we expect
text = soup.get_text()
#print(soup.text)

# Use the find_all() method of soup to extract useful html tags within a webpage.
soup.find_all('a')
```

```
# print out hyperlinks in "a"
all_links = soup.find_all("a")
for link in all_links:
    print(link.get("href"))
```

```
# to print out table rows only, pass the 'tr' argument in soup.find_all().
rows = soup.find_all('tr')
print(rows[:10])
```

```
# Convert list into a pandas dataframe.
for row in rows:
    row_td = row.find_all('td')
print(row_td)
type(row_td)
```

```
# NO HTML Tags
str_cells = str(row_td)
cleantext = BeautifulSoup(str_cells, "lxml").get_text()
print(cleantext)
```

```
# No HTML Tags by using regular expressions
import re
list_rows = []
for row in rows:
    cells = row.find_all('td')
    str_cells = str(cells)
    clean = re.compile('<.*?>')
    clean2 = (re.sub(clean, '', str_cells))
    list_rows.append(clean2)
print(clean2)
type(clean2)
```

```
# Convert the list into pandas dataframe
df = pd.DataFrame(list_rows)
df.head(10)  # to show 10 rows of data
```

Data Transformation

```
# To learn how to use "split" to transform
df1 = df[0].str.split(',', expand=True)
df1.head(10)
```

```
# delete the square brackets surrounding each row
df1[0] = df1[0].str.strip('[')
df1.head(10)
```

```
# find_all() method
col_labels = soup.find_all('th')
```

```
# Use BeautifulSoup to extract text in between html tags for table headers
all_header = []
col_str = str(col_labels)
clean_text2 = BeautifulSoup(col_str, 'lxml').get_text()
all_header.append(cleantext2)
print(all_header)
```

```
# Convert the table headers to a new pandas dataframe
df2 = pd.DataFrame(old_header)
df2.head()
```

	0
0	[Place, Bib, Name, Gender, ...

```
# Again, split column "0" into multiple column at the comma position for all rows.
df3 = df2[0].str.split(',', expand=True)
df3.head()

# Concatenate the two data frames into one [ concat() method]
frames = [df3, df1]
df4 = pd.concat(frames)
df4.head(10)  # only show 10 rows in cell
# Re-configure the data frame
df5 = df4.rename(columns=df4.iloc[0])
df5.head()
```

```
# Start by getting an overview of data as shown below.
df5.info()
df5.shape
# drop all rows with any missing values
df6 = df5.dropna(axis=0, how='any')
df6.info()
df6.shape
# Drop this redundant row
df7 = df6.drop(df6.index[0])
df7.head()
# rename the title
df7.rename(columns={'Place': 'place'}, inplace=True)
df7.rename(columns={'Team]': 'Team'}, inplace=True)
df7.head()
# Strip the xxx]
df7['Team'] = df7['Team'].str.strip(']')
df7.head()
```

Data Analysis and Visualization

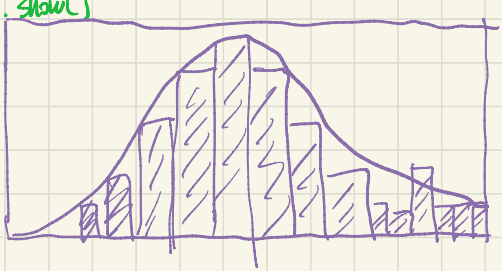
```
# Time_mins
time_list = df7['Chip Time'].tolist()

time_mins = []
for i in time_list:
    if len(i.split(":")) == 2:
        m, s = i.split(":")
        h = 0
    else:
        h, m, s = i.split(":")
    math = (int(h)*3600 + int(m)*60 + int(s))/60
    time_mins.append(math)
print(time_mins)
```

```
# Add new column
df7['Runner_mins'] = time_mins
df7.head()
```

```
# "describe" method
df7.describe(include=[np.number])
```

```
# boxplot
from pylab import rcParams
rcParams['figure.figsize'] = 15, 5
df7.boxplot(column='Runner_mins')
plt.grid(True, axis='y')
plt.ylabel('Chip Time')
plt.xticks([1], ['Runners'])
```


```
x = df7['Runner_mins']
ax = sns.distplot(x, hist=True, kde=True,
 rug=False, color='m', bins=25, hist_kws=
 {'edgecolor': 'black'})
plt.show()
```



```
# Display a side-by-side boxplot comparison of male and female finish times.
g_stats = df7.groupby("Gender", as_index=True).describe()
print(g_stats)
```