# Software Engineering For Data Science (SEDS)

**Class: 2nd Year 2nd Cycle**
**Branch: AIDS**

Dr. Belkacem KHALDI| ESI-SBA

## Lecture 06:

## Data Processing & Cleaning for Data Science: Data Wrangling Documents and Web Scraping

# Data Processing & Cleaning for Data Science

## Part II: Data Wrangling Documents and Web Scraping

1. Parsing and Processing Text Documents
2. Parsing and processing Web Pages

Dr. Belkacem KHALDI
e-mail: b.khaldi@esi-sba.dz

# Data Wrangling Documents

## Parsing and Processing Text Documents

- **Text Processing** is one of the most common task in many **ML** applications.
- Some examples of such applications:
  - Language Translation:
  - Sentiment Analysis
  - Spam Filtering
  - Question Answering
  - Information Retrieval & Extraction

- **Basic Text Processing & Analysis Operations**:
  - **Reading & Extracting Texts**
  - **Basic Text Cleaning:**
    - **Removing unnecessary punctuation, digits**
    - **Tokenization**
    - **Removing stop words**
  - **Basic words Analysis**

Dr. Belkacem KHALDI
e-mail: b.khaldi@esi-sba.dz

# Data Wrangling Documents

## Parsing and Processing Text Documents

- Reading & Extracting Texts
- **Basic Text Cleaning:**
  - **Removing unnecessary punctuation, digits**
  - **Tokenization** — convert sentences to words
  - **Removing stop words** — frequent words such as ”the”, ”is”, etc. that do not have specific semantic
- **Basic words Analysis**

➢ getting a list of concerned files: One easy way to do this is to use the built-in **glob** module:

```
from glob import glob
word_files = glob('<files_folder_path>/*.docx')
```

Top packages are: **textract**, **pythondocx** and **docx2text**

➢ Extracting text from the first file

```
import textract
text = textract.process(word_files[0]) #returns a Byte String type
text = text.decode('utf-8') #To transform Byte String to String
text[:200] #print the first 200 characters
```

➢ Extracting text into a dataframe

```
import pandas as pd
df = pd.DataFrame()
for f in word_files:
    text = textract.process(f)
    text = text.decode('utf-8')
    df = df.append([[text]],ignore_index=True)
df.columns=['content']
```

# Data Wrangling Documents

## Parsing and Processing Text Documents

- **Reading & Extracting Texts**
- **Basic Text Cleaning:**
  - **Removing unnecessary punctuation, digits**
  - **Tokenization** — convert sentences to words
  - **Removing stop words** — frequent words such as ”the”, ”is”, etc. that do not have specific semantic
- **Basic words Analysis**

➢ It is very useful to remove unnecessary characters such as punctuations and digits. This can be done by the following code:

```
import string
translator = str.maketrans('', '', string.punctuation + string.digits)
text = text.translate(translator)
```

➢ Example:

```
sentence = "My name is Belkacem, and I love Data Science."
sentence = sentence.translate(translator)
print(sentence)
```

```
My name is Belkacem and I love Data Science
```

# Data Wrangling Documents

## Parsing and Processing Text Documents

- **Reading & Extracting Texts**
- **Basic Text Cleaning:**
  - **Removing unnecessary punctuation, digits**
  - **Tokenization** — convert sentences to words
  - **Removing stop words** — frequent words such as ”the”, ”is”, etc. that do not have specific semantic
- **Basic words Analysis**

➢ Tokenization ➔ The process of segmenting text into sentences or words.

Top package to be used nltk

```
import nltk
tokens = nltk.word_tokenize(text)
```

➢ Example:

```
import nltk
tokens = nltk.word_tokenize(text)
print(tokens)
```

```
['My', 'name', 'is', 'Belkacem', 'and', 'I', 'love', 'Data', 'Science']
```

# Data Wrangling Documents

## Parsing and Processing Text Documents

- **Reading & Extracting Texts**
- **Basic Text Cleaning:**
  - **Removing unnecessary punctuation, digits**
  - **Tokenization** — convert sentences to words
  - **Removing stop words** — frequent words such as ”the”, ”is”, etc. that do not have specific semantic
- **Basic words Analysis**

➢ Stop words Removal ➔ removes common language prepositions such as “**and**”, “**the**”, “**a**”, and so on in **English**.

```
import nltk
nltk.download('stopwords')
```

we have to download the **stopwords** list

➢ Now, we can import them and remove the **stopwords** from our text.

```
from nltk.corpus import stopwords
en_stopwords = stopwords.words('english')
en_stopwords = set(en_stopwords)

words = text.lower().split()
words = [w for w in words if w not in en_stopwords]
```

➢ When applied to the **sentence** example variable:

```
['My', 'name', 'is', 'Belkacem', 'and', 'I', 'love', 'Data', 'Science']
```

⬇

```
['name', 'belkacem', 'love', 'data', 'science']
```

# Data Wrangling Documents

## Parsing and Processing Text Documents

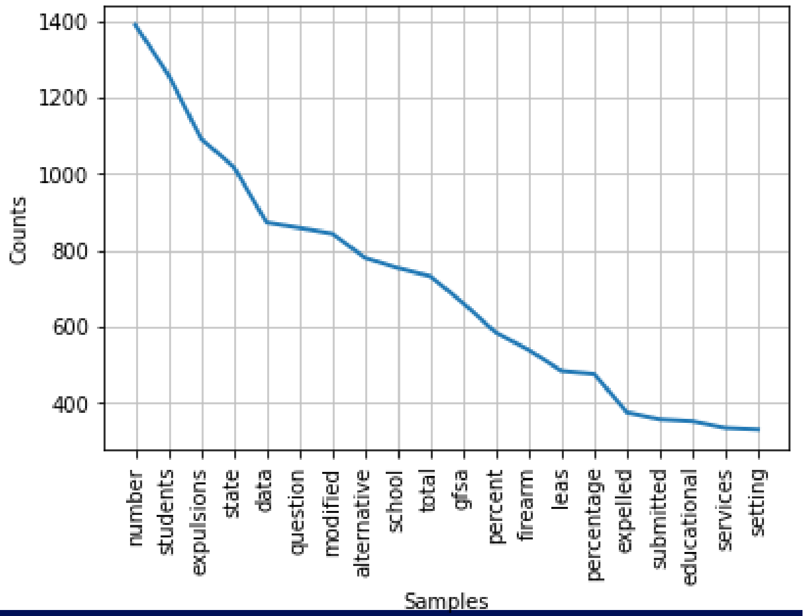
- **Reading & Extracting Texts**
- **Basic Text Cleaning:**
  - **Removing unnecessary punctuation, digits**
  - **Tokenization** — convert sentences to words
  - **Removing stop words** — frequent words such as ”the”, ”is”, etc. that do not have specific semantic
- **Basic words Analysis**

➢ The simplest way to analyze words from our text is to look at **count frequencies (**Use **FreqDist nltk** Class**)**.

```
cnt_fdist = nltk.FreqDist(words)
```

```
cnt_fdist.most_common(20)
```

➢ The top 20 words



```
[('number', 1390),
 ('students', 1260),
 ('expulsions', 1092),
 ('state', 1018),
 ('data', 872),
 ('question', 858),
 ('modified', 843),
 ('alternative', 780),
 ('school', 753),
 ('total', 731),
 ('gfsa', 661),
 ('percent', 583),
 ('firearm', 538),
 ('leas', 483),
 ('percentage', 475),
 ('expelled', 374),
 ('submitted', 356),
 ('educational', 351),
 ('services', 333),
 ('setting', 329)]
```

# Data Wrangling Documents

## Parsing and Processing Text Documents

- **Reading & Extracting Texts**
- **Basic Text Cleaning:**
  - **Removing unnecessary punctuation, tags**
  - **Tokenization** — convert sentences to words
  - **Removing stop words** — frequent words such as ”the”, ”is”, etc. that do not have specific semantic
- **Basic words Analysis**

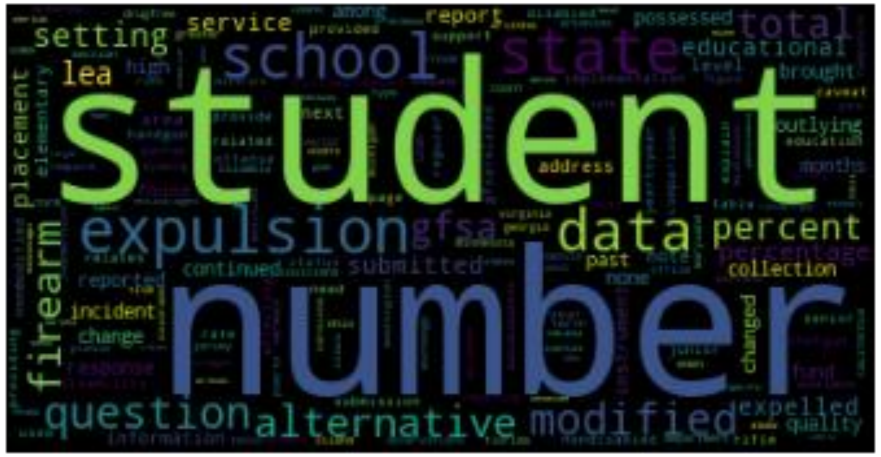
➢ **WordClouds** package is also often used in the Data Science Community to analyze words

```
from wordcloud import WordCloud
wordcloud = WordCloud(collocations=False).generate(' '.join(words))

plt.imshow(wordcloud)

plt.axis("off")
plt.show()
```

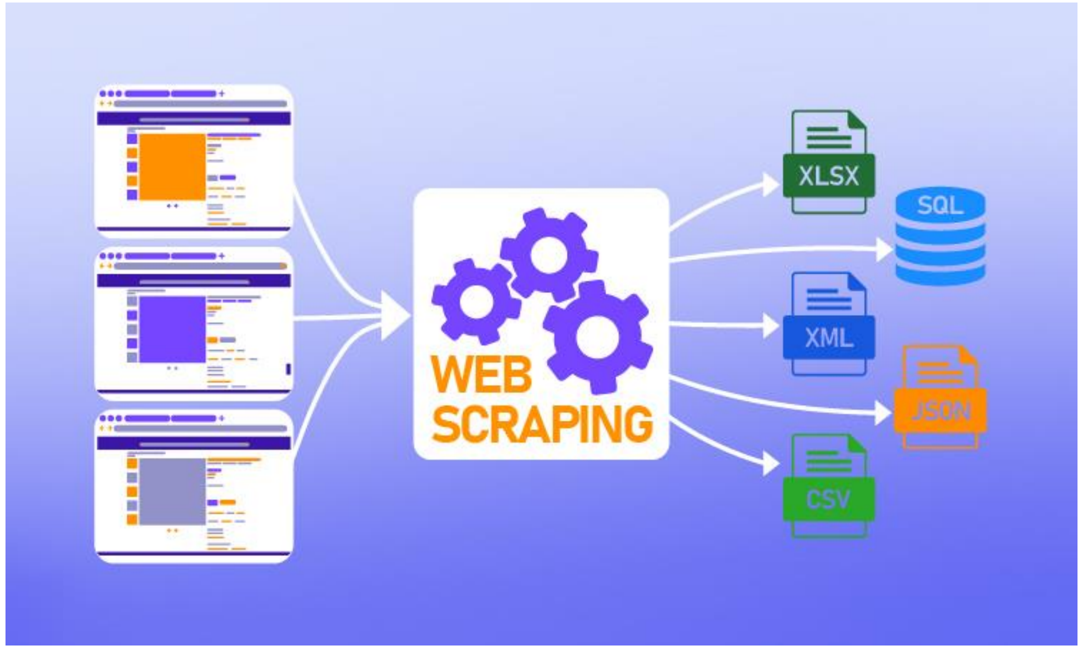
➢ Each word is sized in proportion to the frequency of its occurrence.



# Web Scraping

## Parsing and processing Web Pages

- **Web scraping** ➔ Extraction of data from a website.
  - The data is collected and then exported into a format that is more useful for the user.



- **Is web scraping legal?**
  - In short, **web scraping** isn't **illegal**.
  - Web scraping becomes **illegal** when **non publicly** available data becomes extracted.
  - Legal for **data publicly available**

- **Python libraries used in this lectures:**
  - **urllib** and
  - **requests**

# Web Scraping

## Parsing and processing Web Pages

- **Performing Simple Web Scraping with urllib:**
  - With **urllib** library, we can easily download the content of a webpage or a file.
- **Example:**
  - Let's download the Wikipedia page for Data_science
  - Print out the first bit

```python
from urllib.request import urlopen

url = 'https://en.wikipedia.org/wiki/Data_science'
page = urlopen(url).read()

print(page[:50])  # This a byte String object
print(page[:50].decode('utf-8')) # This a String object
```

This reads the data by fetching the page. Then we print out the first 50 characters.

```
b'<!DOCTYPE html>\n<html class="client-nojs" lang="en'
```

```
<!DOCTYPE html>
<html class="client-nojs" lang="en
```

# Web Scraping

## Parsing and processing Web Pages

- **Performing Simple Web Scraping with urllib:**
  - With **urllib** library, we can easily download the content of a webpage or a file.
- **Example:**
  - We can also use it to retrieve data files.
  - Downloading a MISO **Multiday Operating Margin (MOM)** report

```
datafile_url =
'https://docs.misoenergy.org/marketreports/20210203_mom.xlsx'
mom_data = urlopen(datafile_url).read()

print(mom_data[:20])
```

```
b'PK\x03\x04\n\x00\x08\x08\x08\x00j\xa4CR\x00\x00\x00\x00\x00\x00'
```

Since the content is an excel file, we can load it directly to a dataframe.

```
import pandas as pd
df = pd.read_excel(mom_data)
df.head()
```

Or directly:

```
df = pd.read_excel(datafile_url)
```

Alternatively, we may save the file locally as follows:

```
from urllib.request import urlretrieve
urlretrieve(datafile_url, 'mom_report.xlsx')
```

# Web Scraping

## Parsing and processing Web Pages

- **Performing Simple Web Scraping with request:**

- Request ➔ A package with is an advanced capabilities for the modern web, like multipart file uploads and SSL.
- **Example:**
    - We can scrape the HTML content of the Wikipedia Data_Science page:

```
import requests as rq
url = 'https://en.wikipedia.org/wiki/Data_science'

response = rq.get(url)
response.text[:50]
```

```
'<!DOCTYPE html>\n<html class="client-nojs" lang="en'
```

The content attribute gives us the raw data as a bytestring. We can use this to download a file, like with **urllib**:

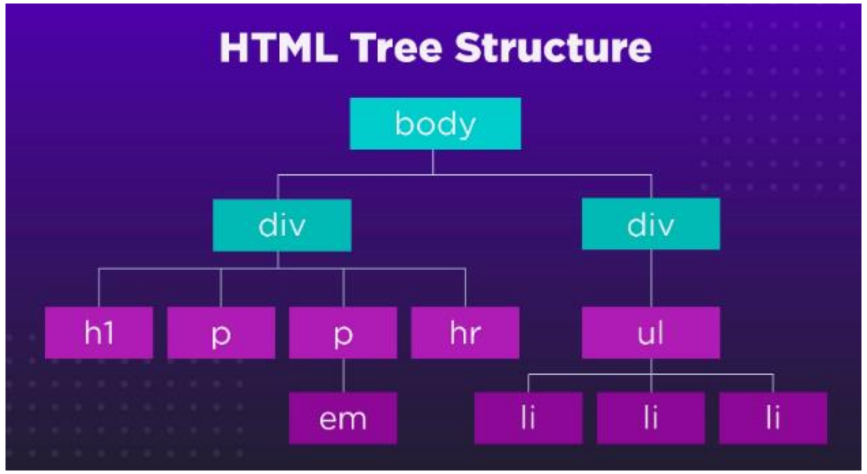
```
datafile_url =
'https://docs.misoenergy.org/marketreports/20210203_mom.xlsx'

res = rq.get(datafile_url)
df = pd.read_excel(res.content)
```

# Web Scraping

## Parsing and processing Web Pages

- **Parsing HTML from scraped pages**



```
url = 'https://en.wikipedia.org/wiki/Data_sience'
wiki_text = urlopen(url).read().decode('utf-8')
```

Two main libraries for parsing and searching HTML are:

- **Beautiful Soup (bs4)** for parsing
- **lxml** for searching

```
from bs4 import BeautifulSoup as bs
import lxml

soup = bs(wiki_text)
```

**bs** has a second param that is that is the parser, which could be:

- **html.parser** – built-in with Python
- **lxml** – fast
- **html5lib** – best for broken HTML

```
links = soup.find_all('a')
print(links[100])
```

**Parsing href html links**

```
<a href="/wiki/Montpellier_2_University" title="Montpellier 2 University">University of Montpellier II</a>
```

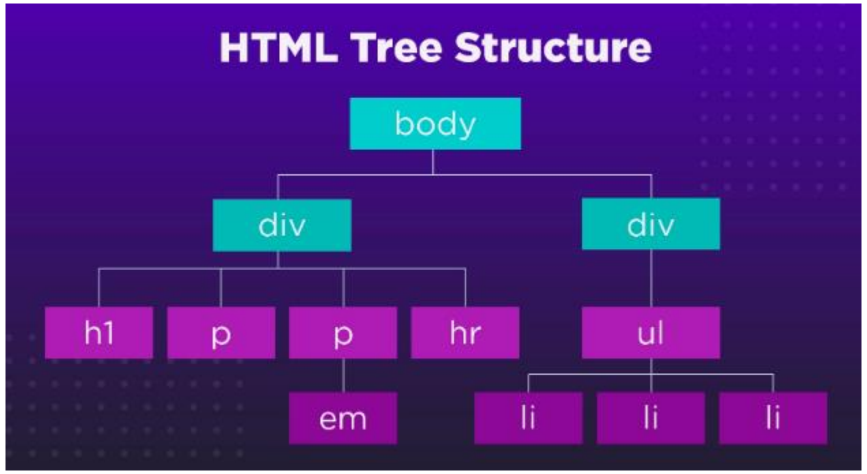
```
'University of Montpellier\xa0II'
```

```
print(links[100].text)
```

# Web Scraping

## Parsing and processing Web Pages

- **Parsing HTML from scraped pages**



The **find_all** method can be also used by specifying **title**, and **text:**

```
soup.find_all('a', {'title': 'Montpellier 2 University'})
soup.find_all('a', {'text': 'science'})
```

Matching a pattern instead ➔ use **request** package

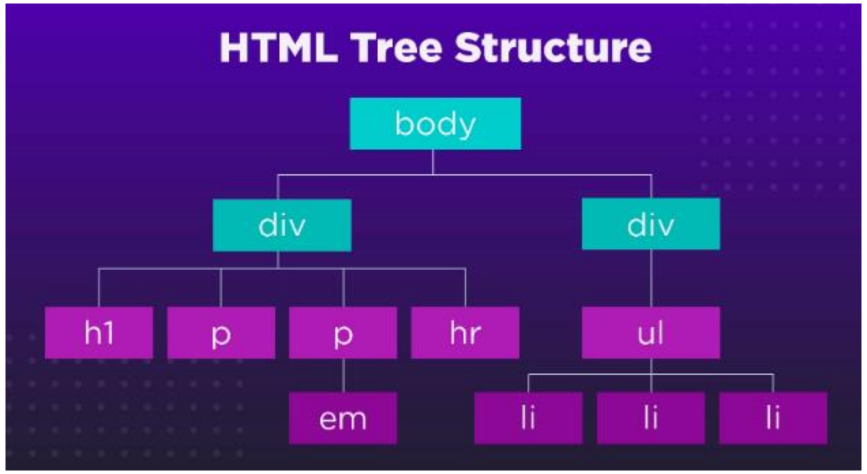
```
import re
# we use re,compile to generate a regx.
soup.find_all('a', text=re.compile('.*science.*'))
```

```
[<a href="/wiki/Information_science" title="Information science">information science</a>,
 <a href="/wiki/Computer_science" title="Computer science">computer science</a>,
 <a href="/wiki/Information_science" title="Information science">information science</a>,
 <a href="/wiki/Computer_science" title="Computer science">computer science</a>,
 <a class="external text" href="http://cacm.acm.org/magazines/2013/12/169933-data-science-and-prediction/fulltext" rel="nofollow">"Data science and prediction"</a>,
 <a class="external text" href="https://doi.org/10.1126%2Fscience.1170411" rel="nofollow">10.1126/science.1170411</a>,
 <a class="external text" href="https://doi.org/10.3390%2Fmake1010015" rel="nofollow">"Defining data science by a data-driven quantification of the community"</a>,
 <a class="external text" href="http://archive.nyu.edu/handle/2451/31553" rel="nofollow">"Data science and prediction"</a>,
 <a class="external text" href="https://statmodeling.stat.columbia.edu/2013/11/14/statistics-least-important-part-data-science/" rel="nofollow">"Statistics is the least important part of data
science « Statistical Modeling, Causal Inference, and Social Science"</a>,
 <a dir="ltr" href="https://en.wikipedia.org/w/index.php?title=Data_science&amp;oldid=1120574273">https://en.wikipedia.org/w/index.php?title=Data_science&amp;oldid=1120574273</a>,
 <a href="/wiki/Category:Information_science" title="Category:Information science">Information science</a>]
```

# Web Scraping

## Parsing and processing Web Pages

- **Parsing HTML from scraped pages**



**Collecting data from several pages:**

```
all_text = []
for link in list_links:
         html = rq.get(link).text
         soup = bs(html)
         paragraph_text = soup.find_all('p')
         all_text.extend([p.text for p in paragraph_text])

text = ' '.join(all_text)
```

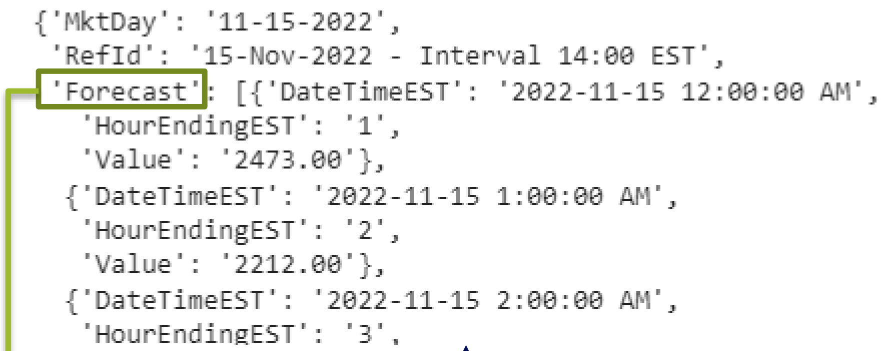
# Web Scraping

## Parsing and processing Web Pages

- **Using APIs to collect data**
  - Data can be collected as **json** format using **APIs**
- **For example:**
  - **let's use one of the** MISO's APIs listed in https://www.misoenergy.org/markets-and-operations/RTDataAPIs/

**Json** file returned from **Day Ahead Wind Forecast** API

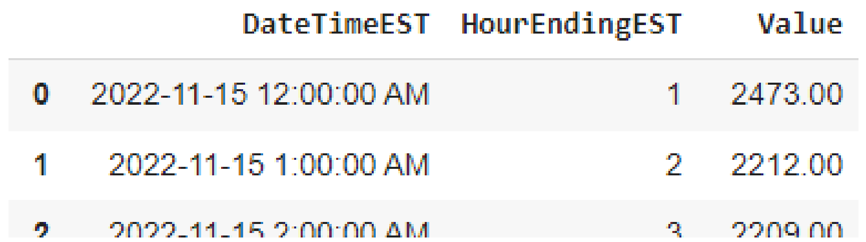
```
{"MktDay":"02-06-2021","RefId":"06-Feb-2021 - Interval 22:00 EST","Fore
cast":[{"DateTimeEST":"2021-02-06 12:00:00 AM","HourEndingEST":"1","Val
ue":"12764.00"},...DateTimeEST":"2021-02-07 11:00:00 PM","HourEndingEST":"24"
,"Value":"2079.00"}]}
```



```
{'MktDay': '11-15-2022',
 'RefId': '15-Nov-2022 - Interval 14:00 EST',
 'Forecast': [{'DateTimeEST': '2022-11-15 12:00:00 AM',
   'HourEndingEST': '1',
   'Value': '2473.00'},
  {'DateTimeEST': '2022-11-15 1:00:00 AM',
   'HourEndingEST': '2',
   'Value': '2212.00'},
  {'DateTimeEST': '2022-11-15 2:00:00 AM',
   'HourEndingEST': '3',
```

**Creating a DataFrame from json data**

```
df = pd.json_normalize(res.json()['Forecast'])
```



```
              DateTimeEST  HourEndingEST    Value
0  2022-11-15 12:00:00 AM              1  2473.00
1   2022-11-15 1:00:00 AM              2  2212.00
2   2022-11-15 2:00:00 AM              3  2209.00
```

**Using request to get json data**

```
url =
'https://api.misoenergy.org/MISORTWDDataBroker/DataBrokerSer
vices.asmx?messageType=getWindForecast&returnType=json'
res = rq.get(url)
res.json()
```

# Thanks for your Listening