クレジット：
Mathematics and Informatics Center メディアプログラミング入門 2020 山肩洋子



東京大学
THE UNIVERSITY OF TOKYO

CC BY NC ND

# メディアプログラミング入門

## 第７回：WebスクレイピングとWebAPI

火５@本郷 2020年7月14日
情報理工学系研究科 数理・情報教育研究センター
准教授 山肩 洋子

# 第６回の課題：手書き数文字を認識しよう！

- 手書き文字のデータセットMNIST(Mixed National Institute of Standards and Technology database)を使って、０から９までの10種類の記号に対する文字認識に挑戦していただきます
- 指定した構造やパラメータでモデルを設計・学習してください

著作権等の都合上、ここに挿入されていた画像を削除しました。

手書き文字の例

http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf
Fig.4

# 第７回：Web APIを使ったプログラミング

**講義内容：**Webから情報を取得したり、WebAPIを体験してみよう

**演習内容：**Webスクレイピングの仕組みと注意しなければならない事柄を学ぶ。
Web APIの仕組みとビジネスモデルを紹介したのち、いくつかのWeb APIを試行する

- 複雑なアルゴリズムは低スペックのスマホでは動かない
  → 画像や音声などのデータをサーバーに送り、サーバーで処理した結果を受け取ることで高度な処理を実現
- **インターネット経由で利用できるWeb APIが増加中**
  - Google, Microsoft, 楽天、Amazon, Facebookなど多くの企業が、自社のソフトウェアのAPIを提供


©いらすとや

**音声を送れば…**

**音声認識結果を返してくれる**
「こんにちは」

**音声合成APIもあるので、**
**音声対話アプリもかんたんに実装できる！**

**いろいろな企業が**
**音声認識APIを提供**

著作権等の都合により、ここに挿入されていた画像を削除しました。

Logo/Mark (Google Cloud Platform / IBM Watson / Microsoft Azure API Management / NTT docomo

# Webスクレイピング

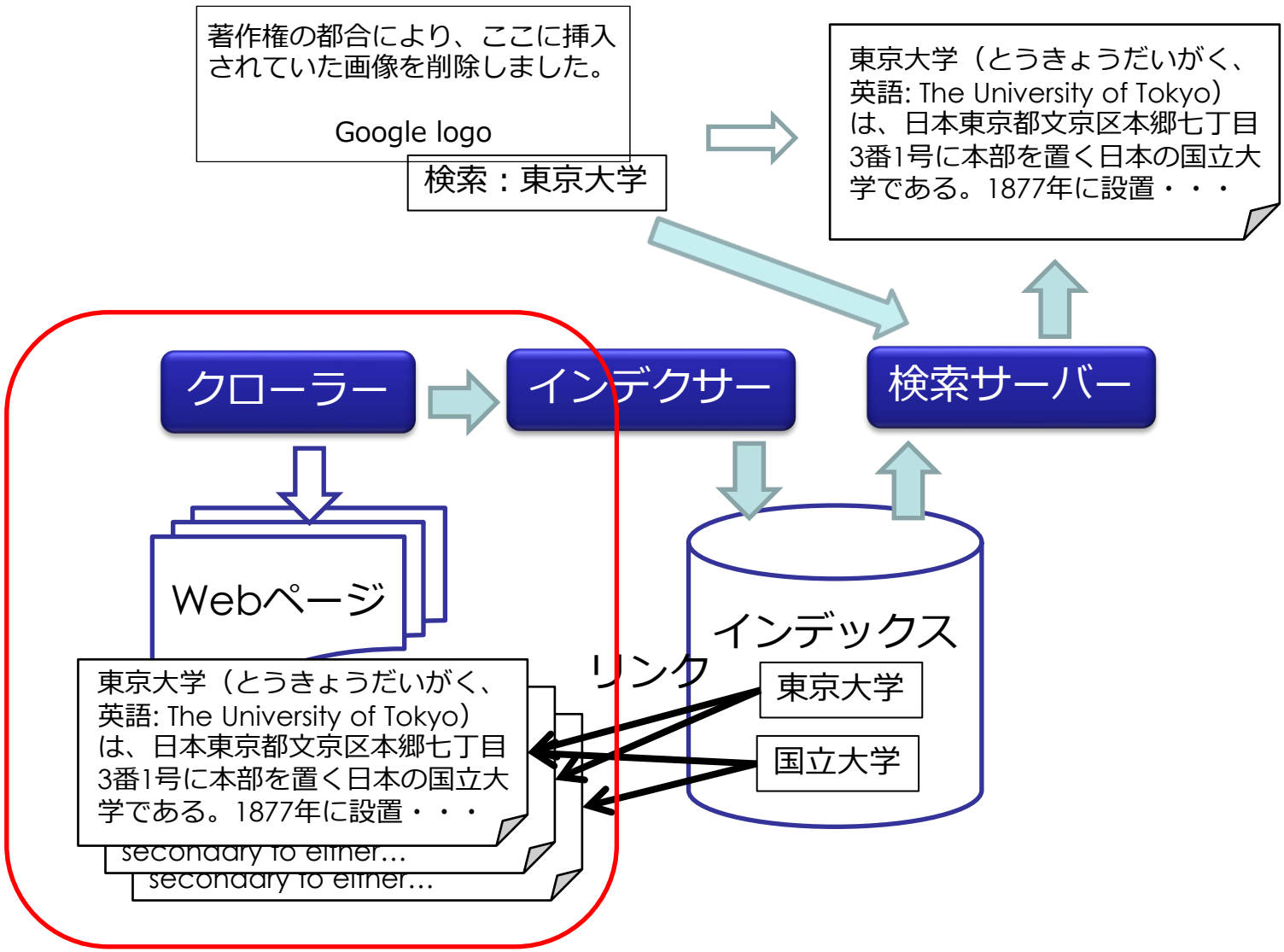
予習演習：WebDataProessing1.ipynb

# なぜメディア処理でWebスクレイピングなのか？

- メディア処理の中心は機械学習
  - データは多ければ多いほどいい！
    （ただし、データのクリーニングは必要）
- 情報解析のために、著作物データを収集して分析することは法律で認められている
  - 平成 30 年著作権法改正：三十条の四（後述）
  - 収集したデータを第三者に渡すことはできない場合が多い → 自分で集めるしかない
- スクレイピングには注意が必要
  - 対象とするWebサーバに頻回アクセスするなど、過度な負担を与えると、DoS攻撃とみなされる場合がある
  - 法律や規約に従う必要がある

# Web検索エンジンの仕組み



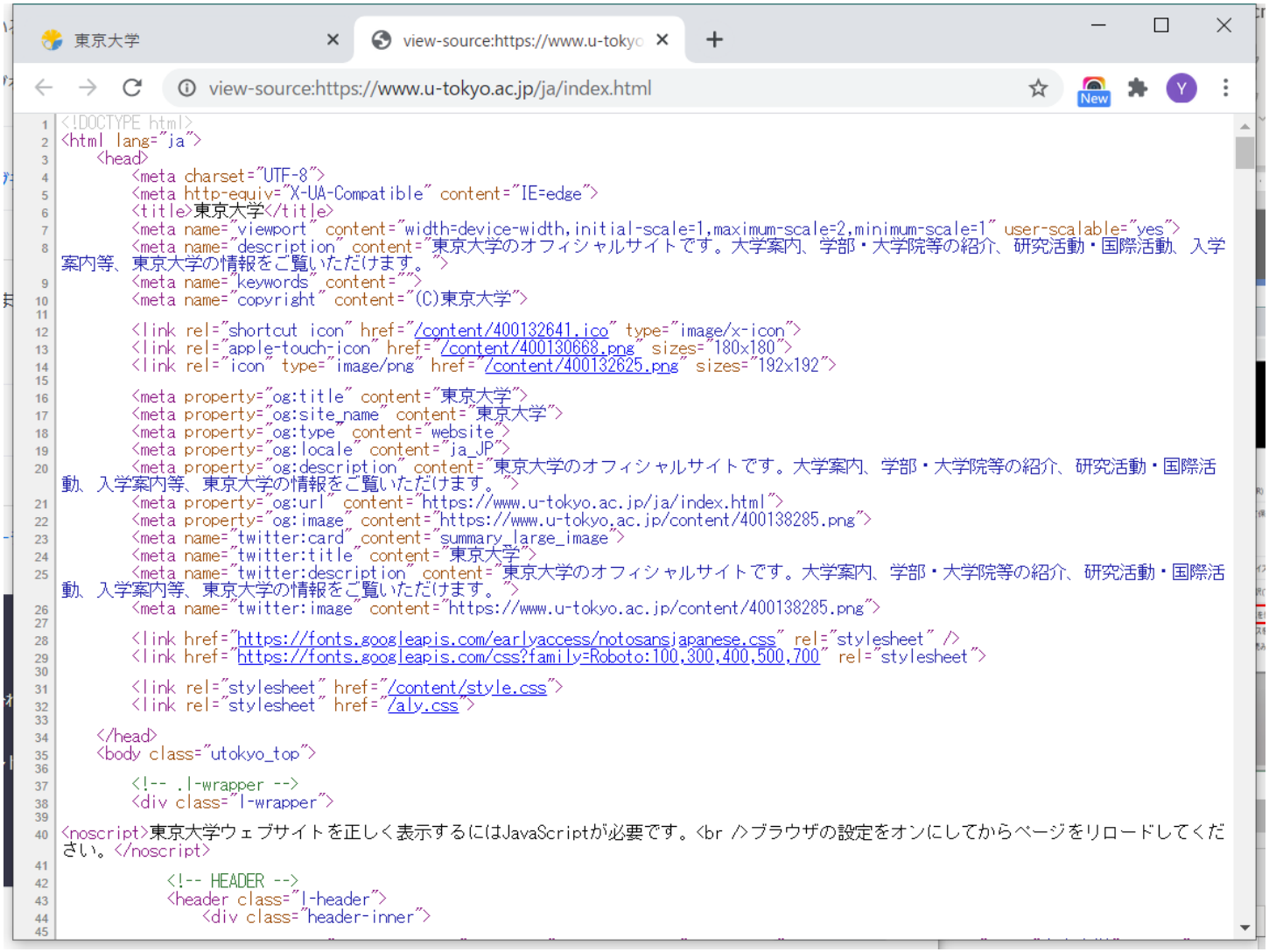
# HTTP (Hyper Text Transfer Protocol)

- ブラウザからWebサーバに対してリクエストを送ると、それに応じたデータを返してくる仕組み
- 送られてきたデータ
  - HTML (HyperText Markup Language)
  - 画像や映像などの情報
  - Javascriptなどの実行コード
- **ブラウザが整形して表示 ← 生のソースを見てみよう！**

# HTMLのソース


東京大学
u-tokyo.ac.jp/ja/index.html
志ある卓越。
東京大学
THE UNIVERSITY OF TOKYO
Language
ナビゲーションスキップ
お問い合わせ
東京大学基金
HOME
大学案内
学部・大学院等
入学案内
教育・学生生活
研究活動
社会連携
戻る(B) Alt+左矢印キー
進む(F) Alt+右矢印キー
再読み込み(R) Ctrl+R
名前を付けて保存(A)... Ctrl
印刷(P)... Ctrl
キャスト(C)...
お使いのデバイスに送信
日本語に翻訳(T)
ページのソースを表示(V) Ctrl+U
フレームのソースを表示(V)
フレームを再読み込み(F)
検証(I) Ctrl+Shift+I
UTokyo FOCUS
五神総長メッセージ
多様性を包摂する社会の実現を目指して
画面の上で
右クリックして
これを選択

# HTMLソース


東京大学
view-source:https://www.u-tokyo
view-source:https://www.u-tokyo.ac.jp/ja/index.html
<!DOCTYPE html>
<html lang="ja">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<title>東京大学</title>
<meta name="viewport" content="width=device-width,initial-scale=1,maximum-scale=2,minimum-scale=1" user-scalable="yes">
<meta name="description" content="東京大学のオフィシャルサイトです。大学案内、学部・大学院等の紹介、研究活動・国際活動、入学案内等、東京大学の情報をご覧いただけます。">
<meta name="keywords" content="">
<meta name="copyright" content="(C)東京大学">
<link rel="shortcut icon" href="/content/400132641.ico" type="image/x-icon">
<link rel="apple-touch-icon" href="/content/400130668.png" sizes="180x180">
<link rel="icon" type="image/png" href="/content/400132625.png" sizes="192x192">
<meta property="og:title" content="東京大学">
<meta property="og:site_name" content="東京大学">
<meta property="og:type" content="website">
<meta property="og:locale" content="ja_JP">
<meta property="og:description" content="東京大学のオフィシャルサイトです。大学案内、学部・大学院等の紹介、研究活動・国際活動、入学案内等、東京大学の情報をご覧いただけます。">
<meta property="og:url" content="https://www.u-tokyo.ac.jp/ja/index.html">
<meta property="og:image" content="https://www.u-tokyo.ac.jp/content/400138285.png">
<meta name="twitter:card" content="summary_large_image">
<meta name="twitter:title" content="東京大学">
<meta name="twitter:description" content="東京大学のオフィシャルサイトです。大学案内、学部・大学院等の紹介、研究活動・国際活動、入学案内等、東京大学の情報をご覧いただけます。">
<meta name="twitter:image" content="https://www.u-tokyo.ac.jp/content/400138285.png">
<link href="https://fonts.googleapis.com/earlyaccess/notosansjapanese.css" rel="stylesheet" />
<link href="https://fonts.googleapis.com/css?family=Roboto:100,300,400,500,700" rel="stylesheet">
<link rel="stylesheet" href="/content/style.css">
<link rel="stylesheet" href="/aly.css">
</head>
<body class="utokyo_top">
<!-- .l-wrapper -->

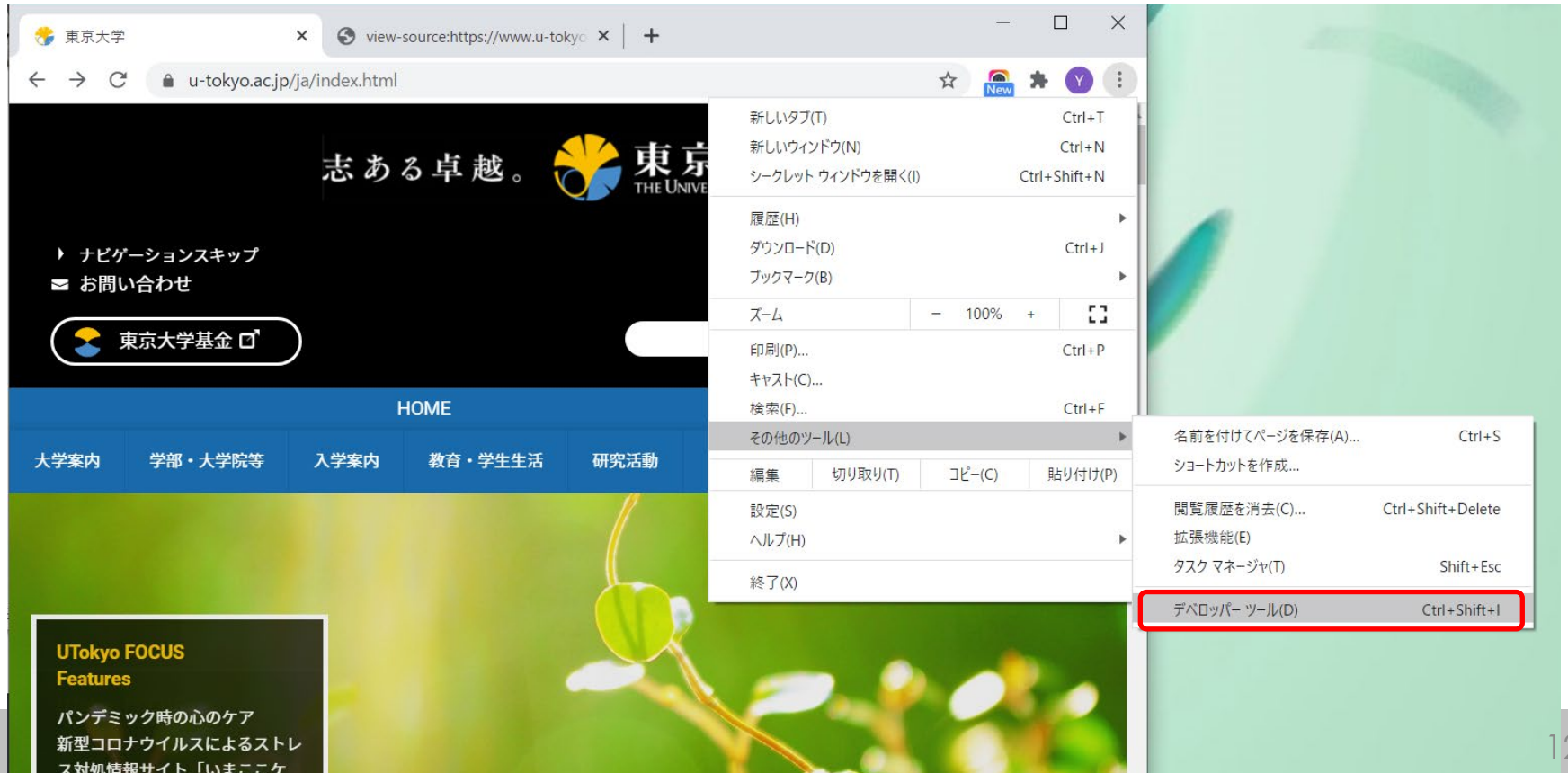
<noscript>東京大学ウェブサイトを正しく表示するにはJavaScriptが必要です。<br />ブラウザの設定をオンにしてからページをリロードしてください。</noscript>
<!-- HEADER -->
<header class="l-header">

# ブラウザのデベロッパーツール/開発者ツール



## 多くのWebブラウザには開発者ツールが内蔵されている

- Edge: 右上の「...」から「その他のツール> 開発者ツール」
- Safari: 「Safari」＞「環境設定」と選択して、「詳細」をクリックして、「メニューバーに“開発”メニューを表示」を選択



# PythonでHTTPリクエスト: requests

In [38]:
```
import random
import requests

url = 'https://www.u-tokyo.ac.jp/ja/index.html'

res = requests.get(url)
res.encoding = res.apparent_encoding  # 文字コードを設定
print(res.text)
```

```
<!DOCTYPE html>
<html lang="ja">
    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title>東京大学</title>
        <meta name="viewport" content="width=device-width,initial-scale=1,maximum-scale
=2,minimum-scale=1" user-scalable="yes">
        <meta name="description" content="東京大学のオフィシャルサイトです。大学案内、
学部・大学院等の紹介、研究活動・国際活動、入学案内等、東京大学の情報をご覧いただけま
す。">
        <meta name="keywords" content="">
        <meta name="copyright" content="(C)東京大学">

        <link rel="shortcut icon" href="/content/100074614.ico">

        <meta property="og:title" content="東京大学">
        <meta property="og:site_name" content="東京大学">
        <meta property="og:type" content="website">
        <meta property="og:description" content="東京大学のオフィシャルサイトです。大学
案内、学部・大学院等の紹介、研究活動・国際活動、入学案内等、東京大学の情報をご覧いただ
けます。">
        <meta property="og:url" content="https://www.u-tokyo.ac.jp/ja/index.html">
```



演習：WebDataProcessing1.ipynb >**ウェブデータ処理１：ウェブスクレイピング**

# HTMLソースの見方

- ペアとなるタグで囲まれている。入れ子もOK
  - <[タグの名前] [オプション]>コンテンツ</[タグの名前]>
- タグにはいろいろな種類がある
  - World Wide Web Consortiumが標準化（現在HTML5)
  - ブラウザはHTMLソースを解釈して画面を生成する

**HTMLの基本構造**

```
<html>
  <head>
    <title>タイトル</title>
  </head>
  <body>
    <h1>大きな文字</h1>
    <p>段落</p>
  </body>
</html>
```

**画像の例**

```
<img src="u-tokyo.jpg" alt="東京大学">
```

**ハイパーリンクの例**

```
<a href="access.html"> アクセスマップ</a>
```

**Javascript読み込みの例**

```
<script src="test.js"></script>
```

# HTMLを使いやすく整理する：HTML Parser

- Python標準HTMLパーザ：lxml
- パージングした情報を辞書にするなど使いやすくする: BeautifulSoup
- たとえば、あるサイトの画像をすべてダウンロードしたい
  1. requestでHTMLデータを取得
  2. lxml+BeautifulSoupでパージング＆整理
  3. BeautifulSoupの`find_all`で'img'タグのアドレスを取得
  4. requestで3.で見つけたアドレスにアクセスして画像をダウンロード

# クローリングのマナー

- 続けて同じサイトにアクセスするときは、時間間隔をあける
  - 30秒程度（一定期間内にダウンロードできる件数が少なるが仕方ない）
  - サーバによっては、クローリングと判断したら遮断するサイトもある
- サイト直下にある'robots.txt'の指示に従う[Googleの例]
- サイト直下にある'sitemap.xml'を利用する [Googleの例]

https://www.google.com/robots.txtより抜粋

```
User-agent: *
Disallow: /search
Allow: /search/about
Allow: /search/static
Allow: /search/howsearchworks
Disallow: /sdch
Disallow: /groups
Disallow: /index.html?
```

https://www.google.com/business/sitemap.xmlより抜粋

```
https://www.google.com/business/ https://www.google.com/business/how-it-works/ https://www.google.com/business/how-it-works/bookings/ https://www.google.com/business/how-it-works/insights/ https://www.google.com/business/how-it-works/posts/
```

# DoS攻撃：Denial of Service attack

- Webサーバやそれにつながる通信網に意図的に過剰な負荷をかけることで、 サービス提供を阻害すること
- Webサーバはアクセス元のIPアドレスを知っている
  → そのIPからの通信を遮断することで対処
- クロール自体は法律に違反していない

  平成21年通常国会　著作権法改正等（平成22年1月1日施行）
  - インターネットで情報検索サービスを実施するための複製等
  - 過去の放送番組等をインターネットで二次利用する際に権利者が所在不明等である場合の利用
  - 国立国会図書館における所蔵資料の電子化
  - その他（インターネット販売等での美術品等の画像掲載、情報解析研究のための複製、送信の効率化等のための複製、 電子機器利用時に必要な複製）
- 情報解析のために、著作物データを収集して分析することは法律で認められている
  - 平成 30 年著作権法改正：三十条の四

  http://www.bunka.go.jp/seisaku/chosakuken/hokaisei/h21_hokaisei/

# 著作権法の一部を改正する法律（平成30年法律第30号）について

- 改正の趣旨
  - デジタル・ネットワーク技術の進展により、新たに生まれる様々な著作物の利用ニーズに的確に対応 するため、著作権者の許諾を受ける必要がある行為の範囲を見直し、情報関連産業、教育、障害者、 美術館等におけるアーカイブの利活用に係る著作物の利用をより円滑に行えるようにする。
- 著作権制度について
  - ＜著作権の保護＞ 他人の著作物（例：小説、論文、新聞、写真、美術、音楽、映画、コンピュータプログラム等）を利用※する場合、 著作権者の許諾が必要。
    （※）権利が付与されている行為：コピー（複製）、ネットワークでの送信（公衆送信）、演奏、上映、譲渡、貸与 等
  - ＜著作権の例外（「権利制限規定」）＞ 法律で定める一定の場合※は、著作者の権利が制限され、許諾を得なくても自由に利用することが可能。
    （※）引用、報道のための利用、学校の授業での著作物のコピー、教科書への著作物の掲載、図書館での文献のコピー、 インターネット情報検索のためのウェブサイトの情報のコピー等、様々な場合について規定が整備されている。

ref. 2020/07/14 文化庁「著作権法の一部を改正する法律の概要」、
https://www.bunka.go.jp/seisaku/chosakuken/hokaisei/h30_hokaisei/pdf/r1406693_01.pdf

# デジタル化・ネットワーク化の進展に対応した柔軟な権利制限規定の整備

（第30条の4、第47条の4、第47条の5等関係）

- 著作物の市場に悪影響を及ぼさないビッグデータを活用したサービス等※のための著作物の利用 について、許諾なく行えるようにする。
- イノベーションの創出を促進するため、情報通信技術の進展に伴い将来新たな著作物の利用方 法が生まれた場合にも柔軟に対応できるよう、ある程度抽象的に定めた規定を整備する。

（※） 例えば現在許諾が必要な可能性がある以下のような行為が、無許諾で利用可能となる。

- 所在検索サービス（例：書籍情報の検索）
  →著作物の所在（書籍に関する各種情報）を検索し、その結果と共に著作物の一部分を表示する。
- 情報解析サービス（例：論文の盗用の検証）
  →大量の論文データを収集し、学生の論文と照合して盗用がないかチェックし、盗用箇所の原典の一部分を表示する。

ref. 2020/07/14 文化庁「著作権法の一部を改正する法律の概要」、
https://www.bunka.go.jp/seisaku/chosakuken/hokaisei/h30_hokaisei/pdf/r1406693_01.pdf

# 詳しくは文化庁のHPで！

著作権法の一部を改正する法律（平成30年法律第30号）について（https://www.bunka.go.jp/seisaku/chosakuken/hokaisei/h30_hokaisei/）


https://www.bunka.go.jp/seisaku/chosakuken/hokaisei/h30_hokaisei/pdf/r1406693_02.pdf

# 犯罪とみなされるケース

- 分散型サービス妨害攻撃
  （Distributed Denial of Service attack：DDoS攻撃）
  - 2014年高校１年生の男子生徒がDDoS攻撃で書類送検
  - 海外のDDoS攻撃代行サービスを利用
- IT分野の法整備は不十分
  - 次々と新しい技術が生み出されるが、法整備は追いついていない
  - 法律は社会的コンセンサスから派生
  - IT業界では常識、世間では非常識

    岡崎市立中央図書館事件
    - 2010年3月頃に岡崎市立図書館の蔵書検索システムにアクセス障害が発生し、利用者の一人が逮捕された事件である。利用者に攻撃の意図はなく、また、根本的な原因が図書館側のシステムの不具合にあったことから論議を呼んだ。逮捕された人物が取調べの後、Librahackというサイトを立ち上げて解説をしたことから、Librahack事件とも呼ばれる。(wikipediaより)
    - 図書館長談
      「図書館に了解を求めることなく、繰り返しアクセスしたことが問題だ」
  - **情報倫理を学ぶことは重要**

# WebAPI

予習演習：WebDataProessing2.ipynb

# 第７回：Web APIを使ったプログラミング

- **API（Application Programming Interface）とは？**
  - 既存のソフトウェアの機能を外部のプログラムから利用できるようにする仕組み

  例）iPhoneのカメラアプリの機能（撮影、フォーカス、ホワイトバランス、露出の制御等）を使った料理写真撮影アプリを独自開発
- **インターネット経由で利用できるWeb APIが増加中**
  - Google, Microsoft, 楽天、Amazon, Facebookなど多くの企業が、自社のソフトウェアのAPIを提供


©いらすとや

**音声を送れば…**

**音声認識結果を返してくれる**
「こんにちは」

**音声合成APIもあるので、**
**音声対話アプリもかんたんに実装できる！**

**いろいろな企業が**
**音声認識APIを提供**

著作権等の都合により、ここに挿入されていた画像を削除しました。

Logo/Mark (Google Cloud Platform / IBM Watson / Microsoft Azure API Management / NTT docomo

# さまざまなWeb API

- **情報を提供するAPI**
  - **SNS**: Twitter, facebook, Instagram, Pinterest
  - **旅行**: 楽天トラベル、じゃらん、JTB
  - **ショッピング**：Amazon, 楽天、Yahoo
  - **グルメ**：ホットペッパー、食べログ、ぐるなび
  - **天気**：livedoor天気、OpenWeatherMap
  - **レシピ**：楽天レシピ、スパイスレシピ
  - **地図**：Google map, Yahoo, いつもNAVI
  - **路線検索**：えきすぱあと、NAVITIME
  - **動画**：youtube, ニコニコ動画、Ustream
- **送られたデータに対して行った処理の結果を返すAPI**
  - Google, Microsoft, IBM, docomoなど
  - 画像認識（物体カテゴリ認識、文字認識等）、画像加工、音声認識・合成、機械翻訳、動作推定、人工知能、対話
- **指定された操作を行うAPI**
  - Line, Facebookなど
  - メッセージや写真の投稿、「いいね」の投稿

# 多くの大手IT企業がWebAPIを提供

- Google Cloud API
- Microsoft Azure Cognitive Services
- AWS Amazon API Gateway (WebAPIを構築するサービス)
- facebook for developpers
- IBM Watson API
- Twitter
- Yahoo! Japan
- docomo

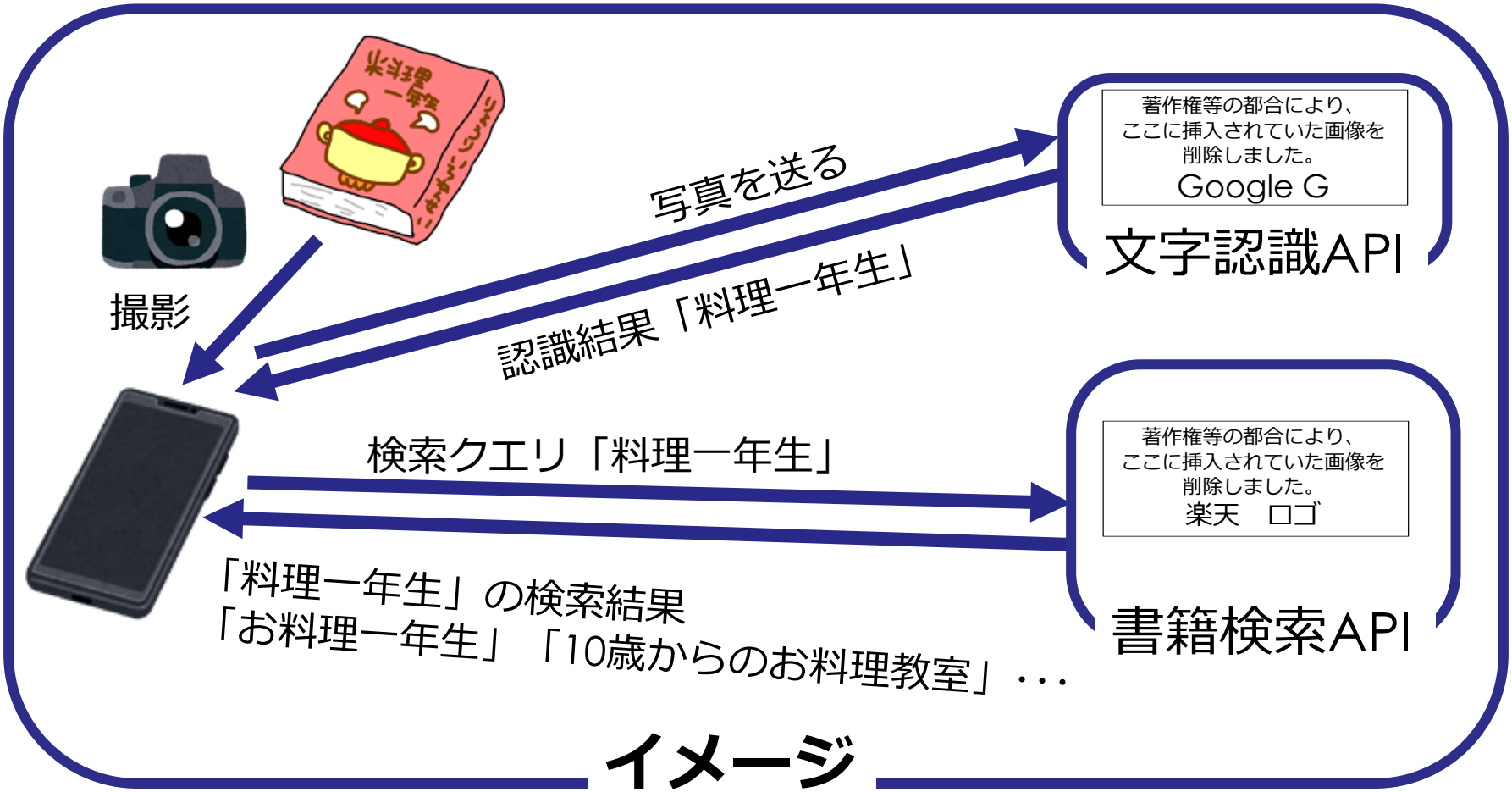
**例えばGoogleだと…**

地図、経路、自然言語処理、音声認識、機械翻訳、動画処理、Googleドライブ、Googleカレンダー、Gmail、Googleスプレッドシート、Marketplace、機械学習、Youtube、Google＋、AdSense、Play Game、Fitness, …

https://console.cloud.google.com/apis/library

# Web API活用事例：マッシュアップ アプリケーション

本の表紙を撮影したらその本の検索結果を表示するアプリを作りたい！



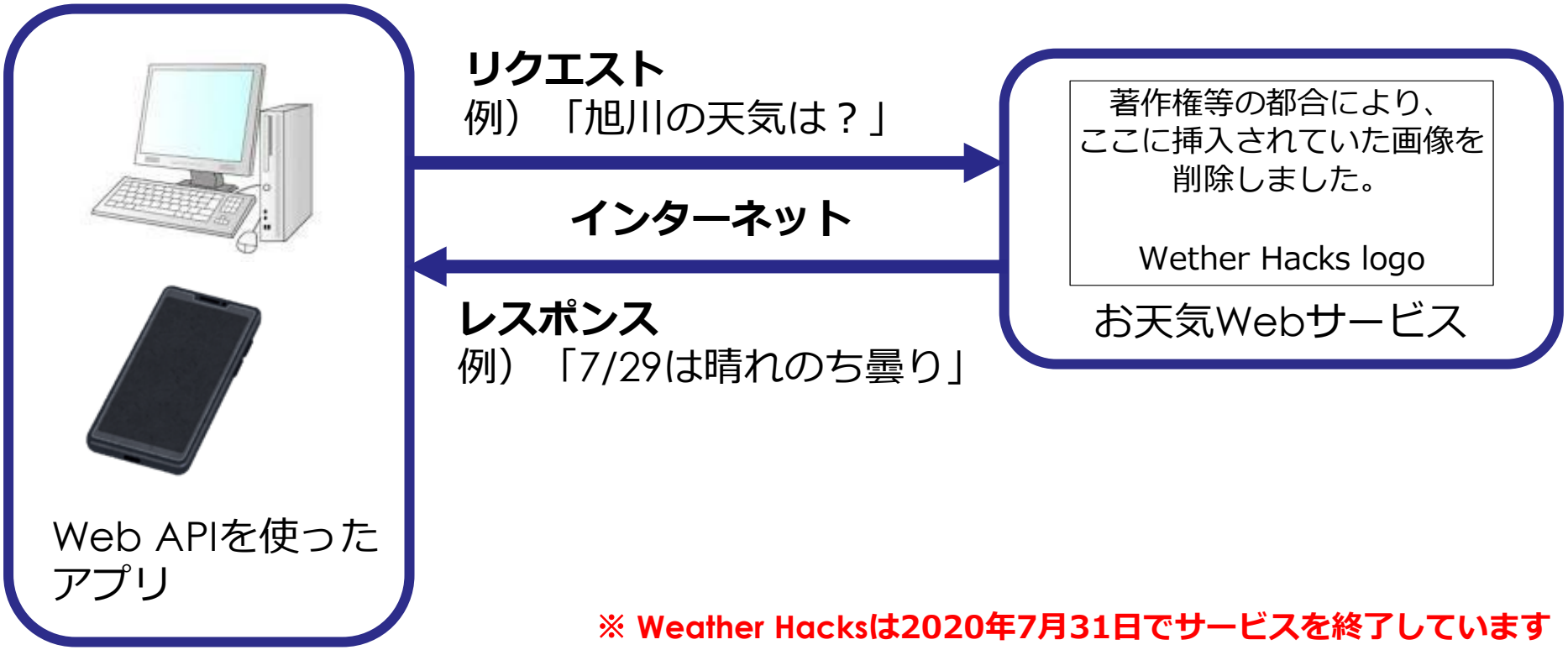
# 企業はなぜWeb APIを提供するのか？

## Web APIのビジネスモデル：APIエコノミー

- **無償提供モデル**：自社のAPIを広く使ってもらうことでブランド力や競争力を得るのが狙い
  - 通常、単位時間あたりのアクセス数など利用制限がある
  - 悪用されないためにクレジットカードを登録されるサービスが増えている
- **課金モデル**：アクセス回数に応じて課金
  例）更新されている可能性があります！！！
  Google Translation API：1か月あたり最大$10の無料使用分が適用、$20/100万文字
  IBM Watson翻訳：月100万文字まで無料、以降は2,10円/25万
  Google音声認識： 月60分まで無料、以降は$0.006/15 sec.
  IBM Watson音声認識：月500分まで無料、以降は2.04円/ sec.
- **間接売り上げモデル**：
  アプリケーションを通じて得た売り上げの数%を徴収

# Web API におけるデータ送受信の例

## Web APIの例：Weather Hacks:
## 地域を指定すると天気情報を返す




**※ Weather Hacksは2020年7月31日でサービスを終了しています**

# Webページで仕様を確認



※ Weather Hacksは2020年7月31日でサービスを終了しています

天気予報 | 災害情報

天気予報 | スポット天気 | Weather Hacks

トップ > Weather Hacks > お天気Webサービス仕様

Weather Hacks
ウェザーハックス

## お天気Webサービス仕様

お天気Webサービス（Livedoor Weather Web Service / LWWS）は、現在全国142カ所の今日・明日・あさっての天気予報・予想気温と都道府県の天気概況情報を提供しています。

### リクエストパラメータ

JSONデータをリクエストする際のベースとなるURLは以下になります。
**http://weather.livedoor.com/forecast/webservice/json/v1**
このURLに下の表のパラメータを加え、実際にリクエストします。

| パラメータ名 | 説明 |
| --- | --- |
| | 地域別に定義されたID番号を表します |

# Webページで仕様を確認

## リクエストパラメータ

JSONデータをリクエストする際のベースとなるURLは以下になります。
**http://weather.livedoor.com/forecast/webservice/json/v1**
このURLに下の表のパラメータを加え、実際にリクエストします。

| パラメータ名 | 説明 |
|---|---|
| city | 地域別に定義されたID番号を表します。<br>リクエストする地域とidの対応は 全国の地点定義表（RSS） 内の<br>「1次細分区（cityタグ）」のidをご参照下さい。（例・佐賀県 伊万里=410020） |

**（例）「福岡県・久留米の天気」を取得する場合**

下記URLにアクセスしてJSONデータを取得します。
基本URL + 久留米のID（400040）
http://weather.livedoor.com/forecast/webservice/json/v1?city=400040

**※ Weather Hacksは2020年7月31日でサービスを終了しています**

# 参照：全国の地点定義表（RSS）

```
<rss xmlns:ldWeather="http://weather.livedoor.com/%5C/ns/rss/2.0" version="2.0">
  <channel>
    <title>1次細分区定義表 - livedoor 天気情報</title>
    <link>http://weather.livedoor.com/?r=rss</link>
    <description>
      livedoor 天気情報で使用されている1次細分区の定義表。それぞれの地点のRSSフィードURLと、 お天気Webサービスで対応
    </description>
    <lastBuildDate>Wed, 26 Jul 2017 22:00:00 +0900</lastBuildDate>
    <author>livedoor Weather Team.</author>
    <language>ja</language>
    <category>天気情報</category>
    <generator>http://weather.livedoor.com/</generator>
    <copyright>(C) LINE Corporation</copyright>
    <image>
      <title>livedoor 天気情報</title>
      <link>http://weather.livedoor.com/</link>
      <url>http://weather.livedoor.com/img/cmn/livedoor.gif</url>
      <width>118</width>
      <height>26</height>
    </image>
    <ldWeather:provider name="(株)ハレックス" link="http://www.halex.co.jp/halexbrain/weather/"/>
    <ldWeather:provider name="日本気象協会" link="http://tenki.jp/"/>
    <ldWeather:source title="全国" li                    m/forecast/rss/index.xml">
      <pref title="道北">
        <warn title="警報・注意報" sourc                    ://weather.livedoor.com/forecast/rss/warn/01a.xml"/>
        <city title="稚内" id="011000" source="http://weather.livedoor.com/forecast/rss/area/011000.xml"/>
        <city title="旭川" id="012010" source="http://weather.livedoor.com/forecast/rss/area/012010.xml"/>
        <city title="留萌" id="012020" source="http://weather.livedoor.com/forecast/rss/area/012020.xml"/>
      </pref>
      <pref title="道東">
        <warn title="警報・注意報" source="http://weather.livedoor.com/forecast/rss/warn/01c.xml"/>
        <city title="網走" id="013010" source="http://weather.livedoor.com/forecast/rss/area/013010.xml"/>
        <city title="北見" id="013020" source="http://weather.livedoor.com/forecast/rss/area/013020.xml"/>
        <city title="紋別" id="013030" source="http://weather.livedoor.com/forecast/rss/area/013030.xml"/>
        <city title="根室" id="014010" source="http://weather.livedoor.com/forecast/rss/area/014010.xml"/>
        <city title="釧路" id="014020" source="http://weather.livedoor.com/forecast/rss/area/014020.xml"/>
        <city title="帯広" id="014030" source="http://weather.livedoor.com/forecast/rss/area/014030.xml"/>
```

旭川のIDは012010

**※ Weather Hacksは2020年7月31日でサービスを終了しています**

# Web API リクエスト・レスポンスの送受信

**リクエスト:（例）「北海道・旭川の天気」を取得する場合**

下記URLにアクセスして聞きたい内容を送ります。

基本URL + 旭川のID（012010）

http://weather.livedoor.com/forecast/webservice/json/v1?city=012010

**サービスサイトのアドレス**

**旭川のID**

自分が作ったアプリ

**インターネット**

**要求内容**

著作権等の都合により、
ここに挿入されていた画像を
削除しました。

Wether Hacks logo

お天気Webサービス

**※ Weather Hacksは2020年7月31日でサービスを終了しています**

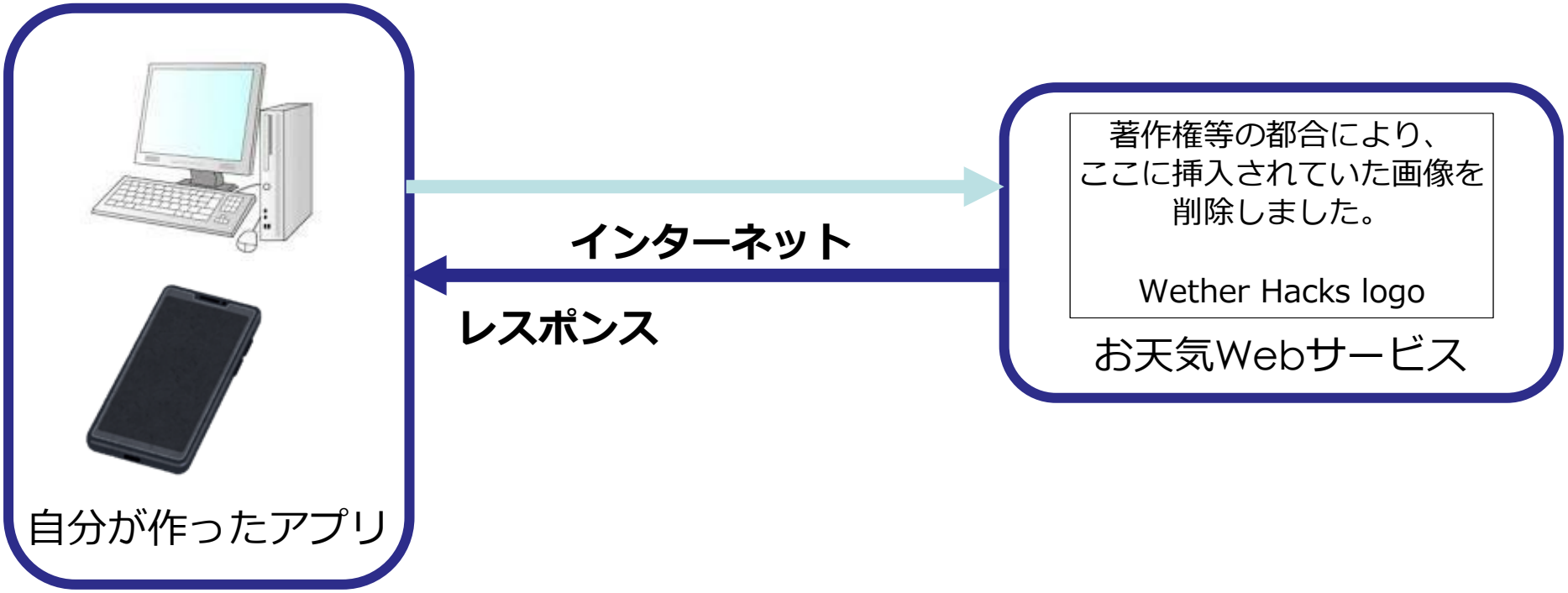
# Pythonのコード

```
import urllib  #インターネットの情報を取ってくるモジュールを読み込み
import json    # jsonファイルを読み込むモジュール

# Web APIのアドレスを変数”url”に代入
url = 'http://weather.livedoor.com/forecast/
webservice/json/v1?city= 012010’

# 変数”url”で指定されたアドレスにリクエスト
html = urllib.request.urlopen(url)

jsonfile = json.loads(html.read().decode('utf-8'))
print([(item['date'], item['telop']) for item in
jsonfile['forecasts']])
```

※ Weather Hacksは2020年7月31日でサービスを終了しています

# Web API リクエスト・レスポンスの送受信

## レスポンス:（例）「北海道・旭川の天気」を取得する場合




※ Weather Hacksは2020年7月31日でサービスを終了しています

# レスポンス：帰ってきたのは・・・？



**コマンドラインでjsonfileと打つと、その中身を表示してくれる**

{'copyright': {'image': {'height': 26,
'link': 'http://weather.livedoor.com/',
'title': 'livedoor 天気情報',
'url': 'http://weather.livedoor.com/img/cmn/livedoor.gif',
'width': 118},
'link': 'http://weather.livedoor.com/',
'provider': [{'link': 'http://tenki.jp/', 'name': '日本気象協会'}],
'title': '(C) LINE Corporation'},
'description': {'publicTime': '2017-07-26T16:42:00+0900',
'text': ' 前線が本州南岸に停滞しています。¥n¥n【関東甲信地方】¥n 関東甲信地方は曇りで、雨や雷雨となっている所があります。¥n¥n 26日は、前線や湿った空気の影響により、曇りや雨で、雷を伴って激し¥nく降る所があるでしょう。¥n¥n 27日は、湿った空気の影響により曇りで、雨の降る所がある見込みです¥n。¥n¥n 関東近海では、27日にかけて、うねりを伴って波が高いでしょう。また¥n、所々で霧が発生しています。船舶は高波や視程障害に注意してください。¥n¥n【東京地方】¥n 26日は、曇りで、雨の降る所もあるでしょう。¥n 27日は、曇りで、朝晩は雨の降る所がある見込みです。'},
'forecasts': [{'date': '2017-07-26',
'dateLabel': '今日',
'image': {'height': 31,
'title': '曇り',
'url': 'http://weather.livedoor.com/img/icon/8.gif',
'width': 50},

**※ Weather Hacksは2020年7月31日でサービスを終了しています**

# Webページで仕様を確認

## レスポンスフィールド

**※ Weather Hacksは2020年7月31日でサービスを終了しています**

取得したJSONデータは以下の定義に基づいて構成されています。(プロパティ名は順不同。)

| プロパティ名 | 内容 |
|---|---|
| location | 予報を発表した地域を定義<br>**プロパティ名** / **内容**<br>area: 地方名（例・九州地方）<br>pref: 都道府県名（例・福岡県）<br>city: 1次細分区名（例・八幡） |
| title | タイトル・見出し |
| link | リクエストされたデータの地域に該当するlivedoor 天気情報のURL |
| publicTime | 予報の発表日時 |
| description | 天気概況文<br>**プロパティ名** / **内容**<br>text: 天気概況文<br>publicTime: 天気概況文の発表時刻 |

# Webページで仕様を確認

| | |
|---|---|
| forecasts | 府県天気予報の予報日毎の配列 |

| プロパティ名 | 内容 |
|---|---|
| date | 予報日 |
| dateLabel | 予報日(今日、明日、明後日のいずれか) |
| telop | 天気（晴れ、曇り、雨など） |
| image | プロパティ名 / 内容<br>tit<br>lin … edoor 天気情報のURL<br>url 天気アイコンのURL<br>width 天気アイコンの幅<br>height 天気アイコンの高さ |
| temperature | **max**・・・最高気温<br>**min**・・・最低気温<br>プロパティ名 / 内容<br>celsius 摂氏<br>fahrenheit 華氏 |

forecastsの中のtelopの中に
お天気情報がある

**※ Weather Hacksは2020年7月31日でサービスを終了しています**

# レスポンス：帰ってきたのは・・・？



**コマンドラインでjsonfileと打つと、その中身を表示してくれる**

※ Weather Hacksは2020年7月31日でサービスを終了しています

・・・(前略)・・・


'forecasts': [{'date': '2017-07-26',
'dateLabel': '今日',
'image': {'height': 31,
'title': '晴れ',
'url': 'http://weather.livedoor.com/img/icon/1.gif',
'width': 50},
'telop': '晴れ',
'temperature': {'max': None, 'min': None}},
{'date': '2017-07-27',
'dateLabel': '明日',
'image': {'height': 31,
'title': '晴のち曇',
'url': 'http://weather.livedoor.com/img/icon/5.gif',
'width': 50},
'telop': '晴のち曇',
'temperature': {'max': {'celsius': '28', 'fahrenheit': '82.4'},
'min': {'celsius': '14', 'fahrenheit': '57.2'}}},
{'date': '2017-07-28',


**2017/7/26のデータ**

**2017/7/27のデータ**

**2017/7/28のデータ**

# 要約すると・・・

```
{'forecasts':[
    {
        'date': '2017-07-26',
        'telop': '晴れ',
    },
    {
        'date': '2017-07-27',
        'telop': '晴のち曇'},
    },
    {
        'date': '2017-07-28',
        'telop': '曇り'}
    }
]}
```

**2017/7/26のデータ**

**2017/7/27のデータ**

**2017/7/28のデータ**

**※ Weather Hacksは2020年7月31日でサービスを終了しています**

# Pythonのコード（つづき）

```
import urllib  #インターネットの情報を取ってくるモジュールを読み込み
import json   # jsonファイルを読み込むモジュール

# Web APIのアドレスを変数"url"に代入
url = 'http://weather.livedoor.com/forecast/
webservice/json/v1?city= 012010'

# 変数"url"で指定されたアドレスにリクエスト
html = urllib.request.urlopen(url)

# jsonファイルの読み込んで"jsonfile"に格納
jsonfile = json.loads(html.read().decode('utf-8'))

# 'forecast'の各要素について'date'と'telop'の値を書き出し
print([(item['date'], item['telop']) for item in
jsonfile['forecasts']])
```

文字コード（仕様を確認）

**出力**

```
[('2017-07-26', '晴れ'), ('2017-07-27', '晴のち曇'), ('2017-07-28', '曇り')]
```

※ Weather Hacksは2020年7月31日でサービスを終了しています

# 第7回の発展的演習

- Twitter API（'TwitterAPI.ipynb'）
  - 要アカウント取得
  - クレジットカード番号登録不要
  - WebAPIを利用するアプリケーションの目的について審査アリ
- IBM Watson API ('IBMWatsonAPI.ipynb')
  - 要アカウント取得
  - クレジットカード番号登録不要
  - 毎月の制限を超えると利用できなくなる（続けて利用したい場合はクレジットカードを登録して利用料金を使っただけ支払い）

# 時系列データ解析

# 時系列データ

- 一連のデータで一つの事象を表す
- ある程度同じ時間間隔でデータが記録される
  - 固定間隔の場合はサンプリングレートとして指定（ex. 音、映像）
  - 幅が一定でない、あるいは欠落がある場合は、個々のデータに**タイムスタンプ**を付与（ex. 株価は週末や独立記念日、クリスマスなどで株式市場が休場のためサンプルがない）

# Pandas-datareader：WebAPIとして公開されている様々な統計データを取得

- 株式や投資信託、為替などの履歴を取得可能
- その他、OECDがOECD加盟国のGDPや雇用指数など各種統計情報を提供したり、World Bankが人口やCO2排出量等のデータを提供

**リクエスト**

例）「2019年1月1日から2019年9月31日までのトヨタの株価は？」

```
import pandas_datareader.data as web
from datetime import datetime

start = datetime(2019, 1, 1) # 取得開始日
end = datetime(2019, 3, 31) # 取得終了日
df = web.DataReader('TM', 'iex', start, end)
df.head(5)
```

WebAPIを使用するプログラムを実行

**インターネット**

著作権等の都合により、ここに挿入されていた画像を削除しました。

Stooq logo

ポーランドの株価や為替データリポジトリ

**レスポンス: DataFrame形式**

例）

| date | open | high | low | close | volume |
|---|---|---|---|---|---|
| 2019-01-02 | 114.68 | 116.53 | 114.6800 | 116.28 | 169884 |
| 2019-01-03 | 115.85 | 116.28 | 114.2900 | 114.65 | 309478 |
| 2019-01-04 | 117.94 | 120.17 | 117.5057 | 119.73 | 307513 |
| 2019-01-07 | 120.71 | 122.18 | 120.3200 | 121.28 | 156119 |
| 2019-01-08 | 123.16 | 123.48 | 121.9600 | 122.31 | 150370 |

# pandas-datareaderで利用可能なサービス



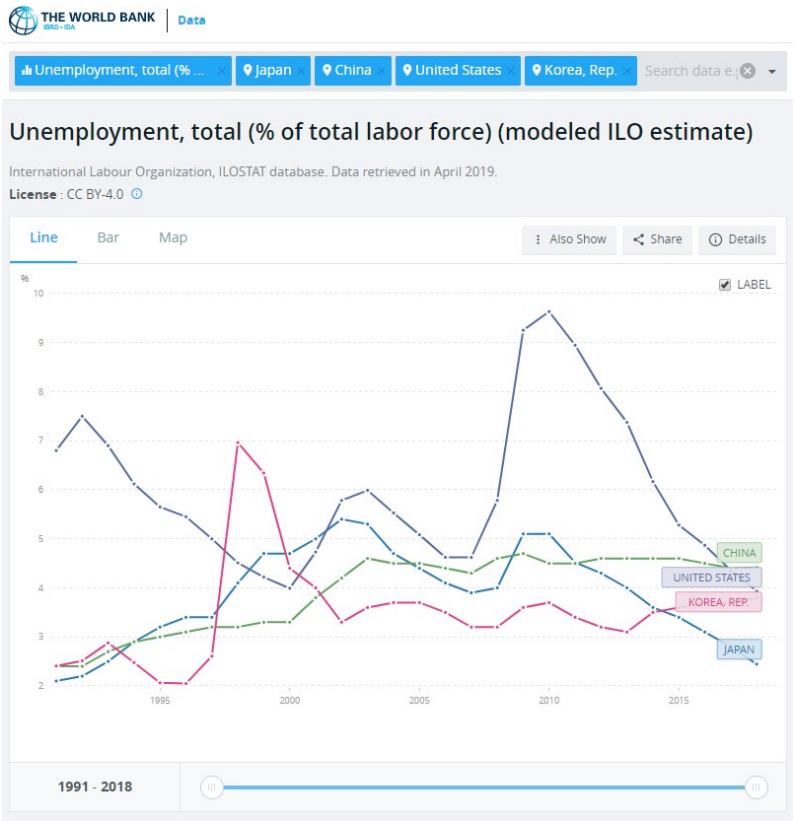
これらのサービスは変更する可能性があります。最新はこちらで確認

- Federal Reserve Economic Data (FRED)：米国経済（2万種類以上）
- Fama-French Data：ファーマ-フレンチの3ファクターモデル
- Bank of Canada：カナダ銀行
- Engima：世界最大の公開データリポジトリ
- Eurostat：欧州の統計データ
- **The Investors Exchange (IEX)：株価履歴 ←演習で使用**
- Moscow Exchange (MOEX)：モスクワ証券取引所
- Morningstar：株式、投資信託、ETF
- NASDAQ：米国のベンチャー向け株式市場
- OECD：OECD諸国の統計データ
- Quandl：株価、ETF等
- Robinhood：株価
- **Stooq.com：株価 ←日本の株価データも提供（サンプルプログラム：Stooq.ipynb）**
- Tiingo：株価、投資信託、ETF等
- Thrift Savings Plan (TSP)：投資信託
- **World Bank：約8000の開発指標 (サンプルプログラム：WorldBank.ipynb)**

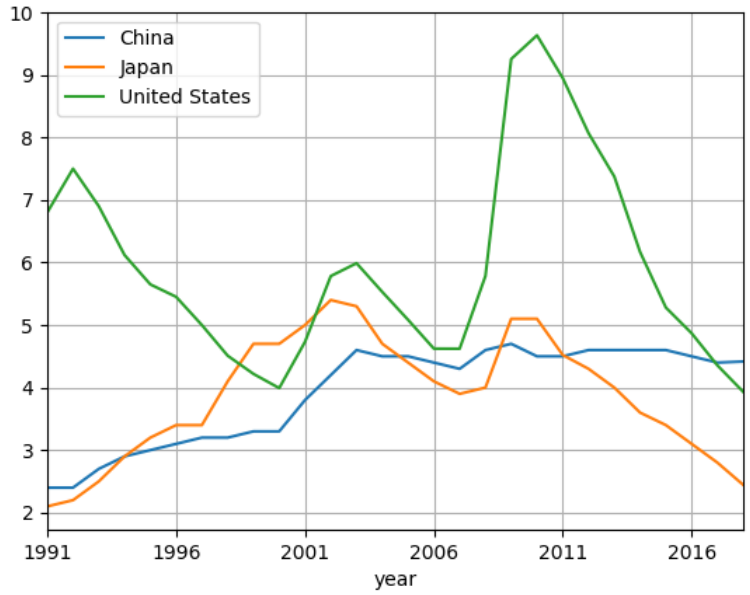
演習：TimeSeriesDataAnalysis1.ipynb > **2. 株価の取得と記録**

# World Bank (世界銀行)

- サンプルプログラム：WorldBank.ipynb
- 貧困、経済、気候変動、保険、教育、ジェンダーなどの分野で、約8000の開発指標を無料公開




pandas-dataframeでダウンロードし、matplotlibでグラフ化






演習（Optinal）：WorldBank.ipynb

# 株価データの取得と
# ローソク足チャートの描画

演習ファイル：WebDataProcessing3.ipynb

# 演習１）TimeSeriesDataAnalysis1.ipynb

**IEXからオンラインで指定銘柄の株価の変動履歴情報を取得し描画する**

- DataReaderの第１引数である'TM'はTOYOTA MOTOR CORPのティッカーコード（銘柄コード）
- 取得できるのは以下の5種類
  - open : 始値
  - high : 高値
  - low : 安値
  - close : 終値
  - volume: 出来高
- このうちopen, high, low, closeの単位はドル。これら４つを合わせてOHLCという略称で呼ぶ。
- 出来高はその日売買が成立した株数

# pandasにおけるIndexオブジェクト

- ローソク足チャートを簡単に描画できるmpl_financeとパッケージのcandlestick_ohlcという関数を利用
- この関数は、1行目から時間順に株価データが並んでおり、1列目に日付（ただし数値化されたもの）、2～4列目にOpen, High, Low, Closeが並んだ二次元配列を受け取る
- TimeSeriesDataAnalysis1.ipynb「4.1 前処理」を参照

- DataFrameのindexはタイムスタンプ（日付）
- これはstring型ではなく、datetimeという日付・時間型

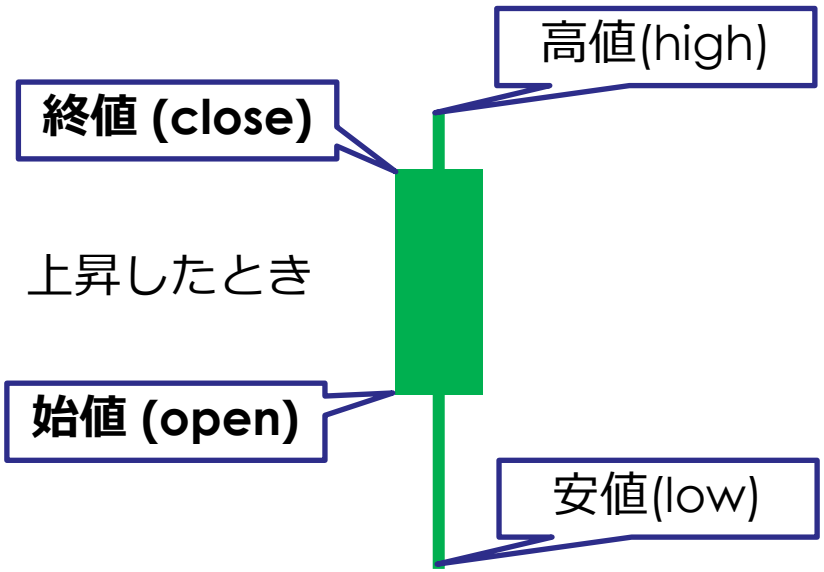
| date | open | high | low | close | volume |
|---|---|---|---|---|---|
| 2019-01-02 | 114.68 | 116.53 | 114.6800 | 116.28 | 169884 |
| 2019-01-03 | 115.85 | 116.28 | 114.2900 | 114.65 | 309478 |
| 2019-01-04 | 117.94 | 120.17 | 117.5057 | 119.73 | 307513 |
| 2019-01-07 | 120.71 | 122.18 | 120.3200 | 121.28 | 156119 |
| 2019-01-08 | 123.16 | 123.48 | 121.9600 | 122.31 | 150370 |

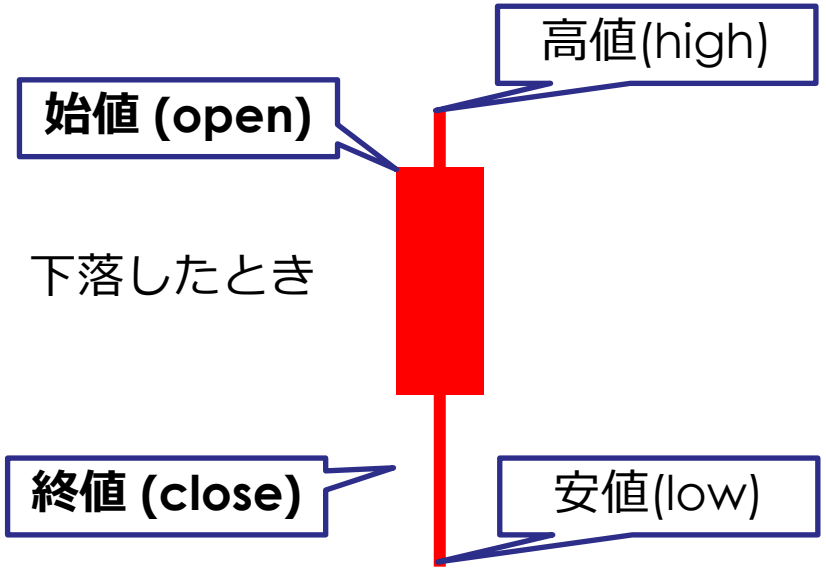
# datetime：基本的な日付型および時間型

- 日付や時刻を対象にした**四則演算が可能**
  - 1日後や1週間後、1時間前を計算したり、「1990年3月15日から今日までの経過日数は？」などの計算も可能
  - 演習ファイル：Datetime.ipynb
- DataFrame型の時系列データをグラフ化する際、同じ間隔でサンプルされていないデータでも、datetime型のタイムスタンプがインデックスに設定されていれば、経過時間によってメモリの幅を自動調整

# ローソク足チャート

- 株価などの相場の値動きを時系列に沿って図表として表す手法の一つ
- 単位期間中に初めに付いた始値、終値、高値、安値をローソクと呼ばれる一本の棒状の図形に作図し、時系列に沿って並べて値段の変動をグラフとして表したもの
- 上昇した時と下落した時で色が変わる



# mpl_financeによるローソク足チャートの描画

- ローソク足チャートを簡単に描画できるmpl_financeとパッケージのcandlestick_ohlcという関数を利用
- この関数は、1行目から時間順に株価データが並んでおり、1列目に日付（ただし数値化されたもの）、2～4列目にOpen, High, Low, Closeが並んだ二次元配列を受け取る
- TimeSeriesDataAnalysis1.ipynb「4.1 前処理」を参照

**DataFrame型**

| date | open | high | low | close | volume |
|---|---|---|---|---|---|
| 2019-01-02 | 114.68 | 116.53 | 114.6800 | 116.28 | 169884 |
| 2019-01-03 | 115.85 | 116.28 | 114.2900 | 114.65 | 309478 |
| 2019-01-04 | 117.94 | 120.17 | 117.5057 | 119.73 | 307513 |
| 2019-01-07 | 120.71 | 122.18 | 120.3200 | 121.28 | 156119 |
| 2019-01-08 | 123.16 | 123.48 | 121.9600 | 122.31 | 150370 |

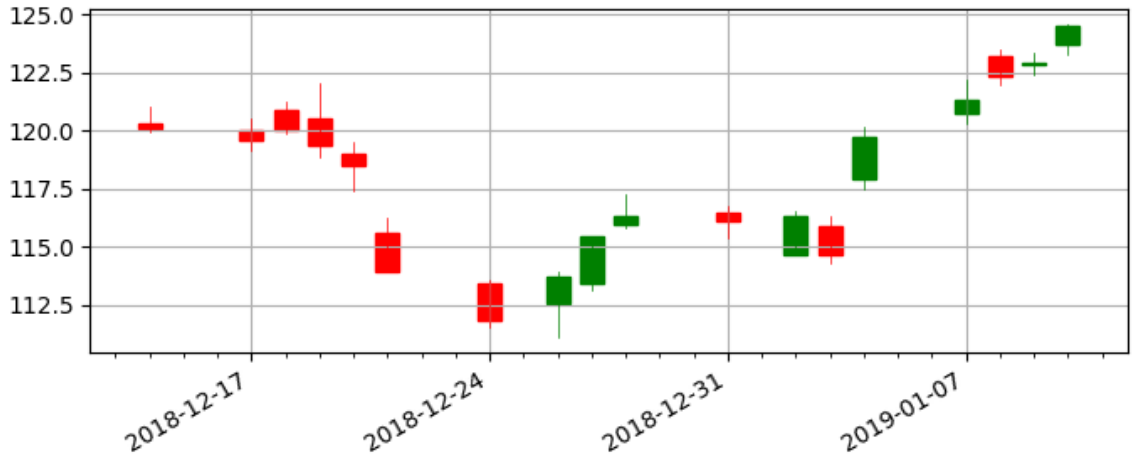
matplotlib.dates.date2numにより
西暦1年1月1日からの経過日数に変換

**リスト型**

```
[[7.370610e+05 1.146800e+02 1.165300e+02 1.146800e+02 1.162800e+02]
 [7.370620e+05 1.158500e+02 1.162800e+02 1.142900e+02 1.146500e+02]
 [7.370630e+05 1.179400e+02 1.201700e+02 1.175057e+02 1.197300e+02]
 [7.370660e+05 1.207100e+02 1.221800e+02 1.203200e+02 1.212800e+02]
 [7.370670e+05 1.231600e+02 1.234800e+02 1.219600e+02 1.223100e+02]]
```

# matplotlib.datesでロケータの設定

- matplotlibの軸で日付を扱うためのモジュール
- datetimeをインデックスに持つDataFrameをグラフ化
- **Locator**: 主軸は補助軸の**間隔**を設定する
  - set_major_locator: 主軸の間隔
  - set_minor_locator: 補助軸の間隔
- **formatter**: **軸ラベル**の出力形式を指定する
  - set_major_formatter: 主軸のラベル
  - set_minor_formatter: 補助軸のラベル（ないことが多い）



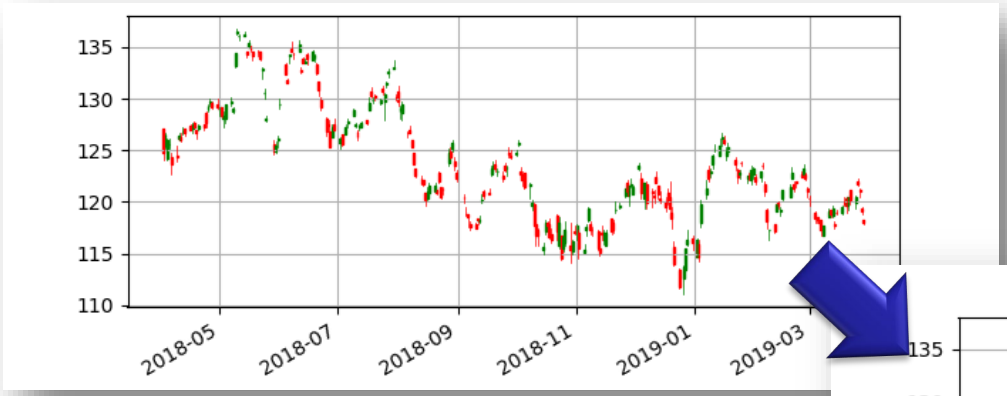

- 主軸は毎週月曜日ごと
- 補助軸は日ごと
- ラベルは自動設定

# ダウンサンプリングと
# 移動窓

演習ファイル：WebDataProcessing4.ipynb

# 1年分の日足データのローソク足チャートを描画すると…




日足データのローソク足チャート

週足データのローソク足チャート


- 日足データでは細かすぎて見づらい…
- 日足を週足に変換する
  → サンプルを記録する頻度を減らす
- 時系列データに対してサンプルの記録頻度を減らすことを**ダウンサンプリング**と呼ぶ

# n項移動平均（○日移動平均）を追加しよう


ローソク足チャートのみ

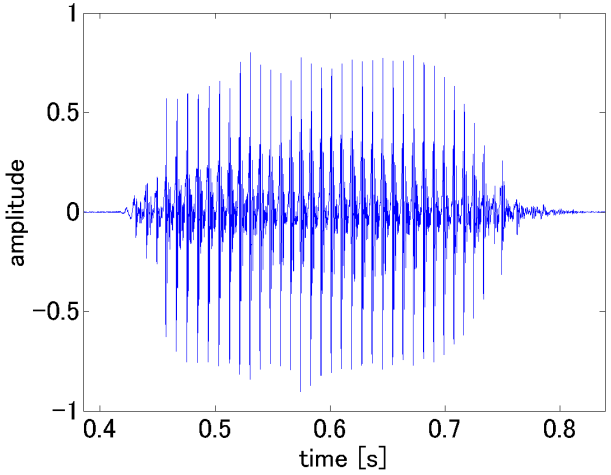
移動平均グラフを追加


- 株価には長期的に見れば上がり傾向、下がり傾向がある（これを上昇トレンド、下降トレンドと呼ぶ）
- その日を含め、過去の一定期間（**移動窓**）の株価の平均を可視化するのが有効→これを**n項移動平均**と呼ぶ
  - n項には5日間、25日間、50日間などがよく使われるらしい
  - 株式市場の休場日（すなわち株価データが欠落している日）は含まないのが一般的

# 音データに対するダウンサンプリングと移動窓

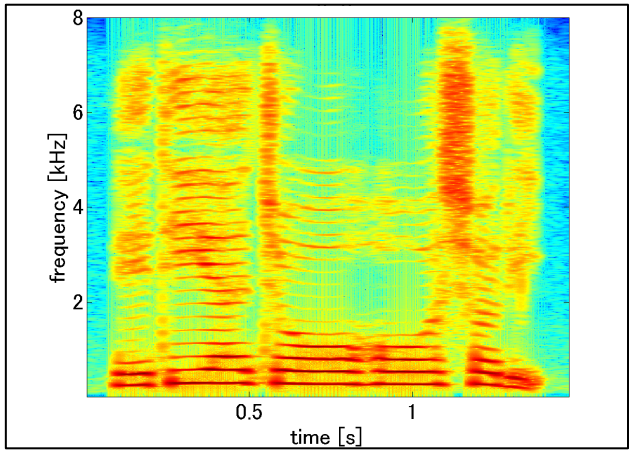
音信号処理で使った概念

- 音のダウンサンプリング
  - データサイズを小さくしたいときに行われる
    例）CDの音 (44.1kHz)を16kHzのmp3に圧縮
  - 音質は低下する（高周波成分が欠落、雑音が乗る）
- 音の移動窓
  - スペクトログラムを計算する際に使う




音声波形

rollingしたあと
区間ごとに
周波数変換




スペクトログラム

# 時系列データに対するresampleとrolling

時系列データであるDataFrameあるいはSeriesが対象

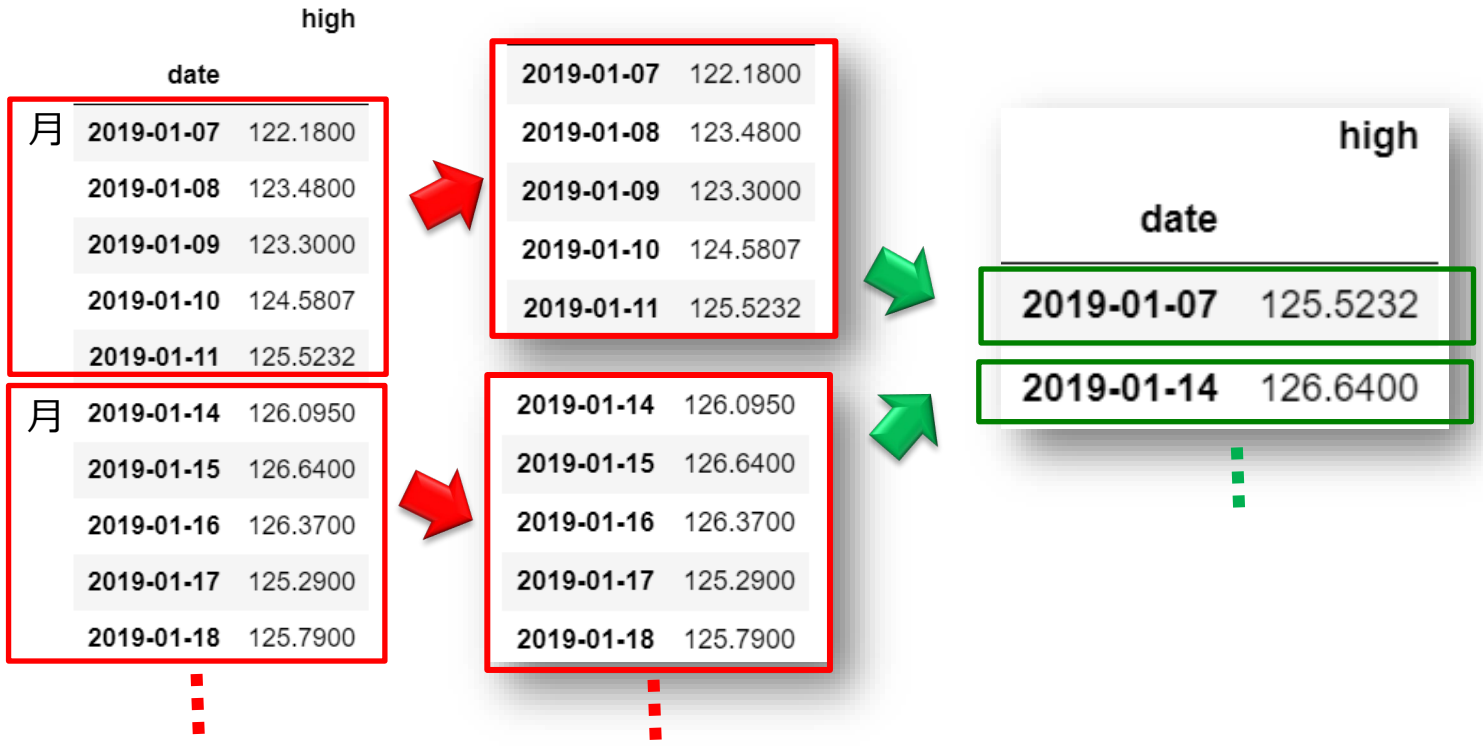
- **ダウンサンプリング：resample**
  時系列を指定した区間ごとにグルーピングして処理を行う
  演習）日足データから週足データを生成

- **移動窓 （moving window)：rolling**
  時系列の各点に対し、それ以前あるいは前後の指定した区間を切り出して処理を行う
  演習）各時刻からさかのぼって1週間分のデータから算出した平均や最大値などを、その時刻のデータに置き換える

# 日足データから週足データを生成

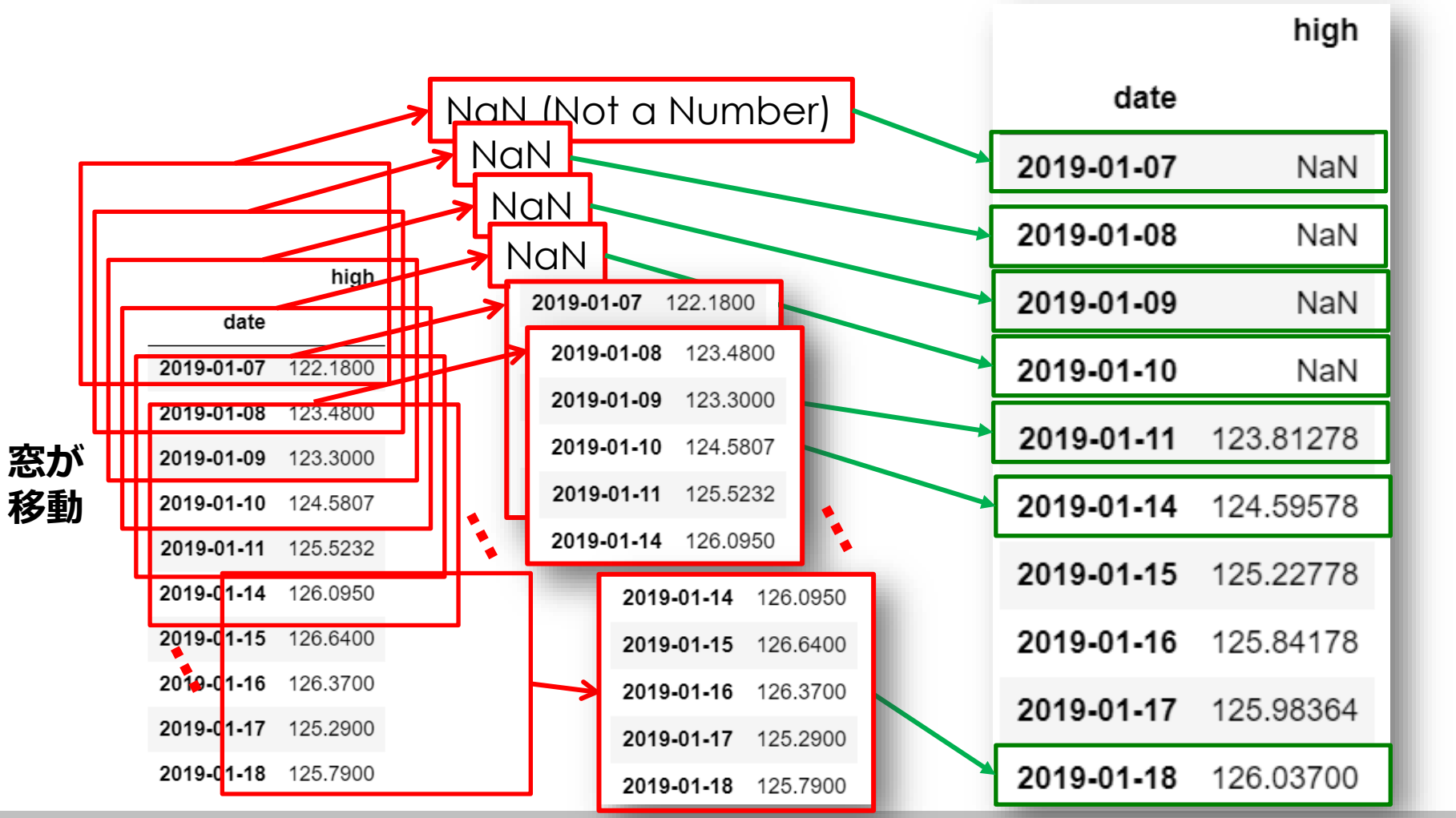
1. 月曜から日曜までの日足データを切り出す
2. 始値(open)は初日の始値 (first)、終値(close)は最終日の終値 (last)、
   高値(high)は最大値 (max)、安値(low)は最小値 (min)をそれぞれ抽出

```
resample('W-MON', closed='left', label='left')
        .agg({'open': 'first', 'high': 'max', 'low': 'min', 'close': 'last', 'volume': 'sum'})
```



# 5日移動平均を生成

1. 各日ごとに、その日を含んでさかのぼって5日間を切り出し
2. 終値 (close)の平均値(mean)を抽出



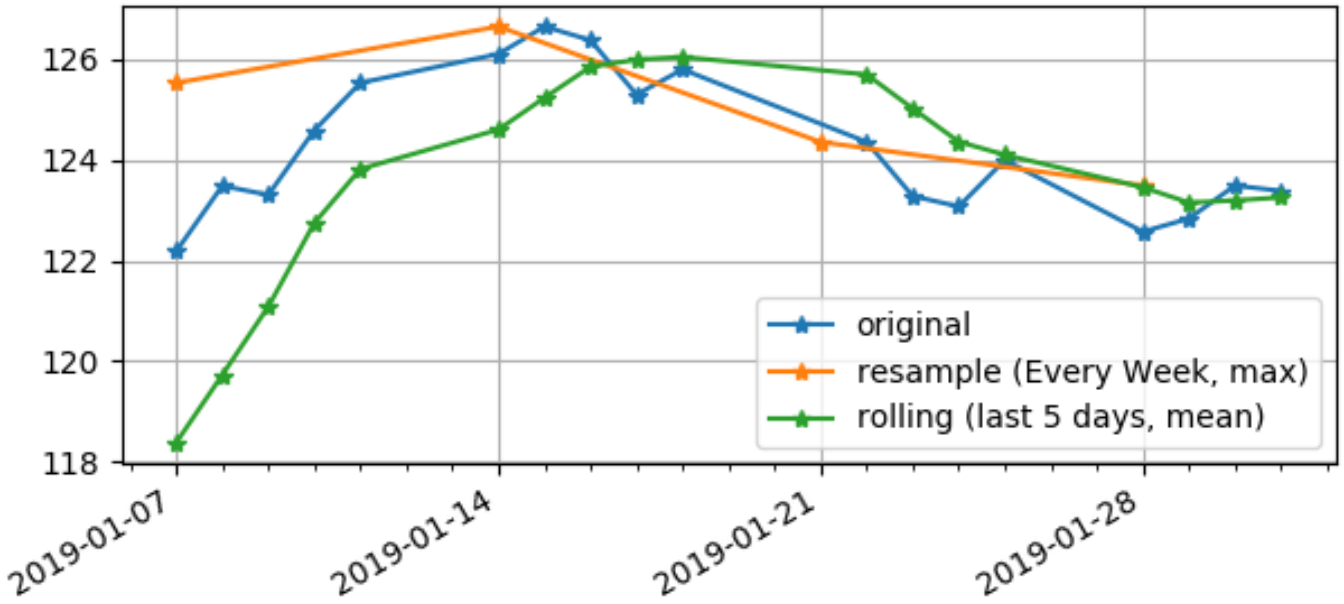
# 時系列データに対するResampleとRolling

- **resample**:
  時系列を指定した区間ごとに切り分けて処理（**ダウンサンプリング**）

  resample('W-MON', closed='left', label='left').agg({'high': 'max'})

- **rolling**: 時系列の各点に対し、それ以前の指定した区間に対して処理（**移動窓・窓関数**）

  rolling(5).mean()



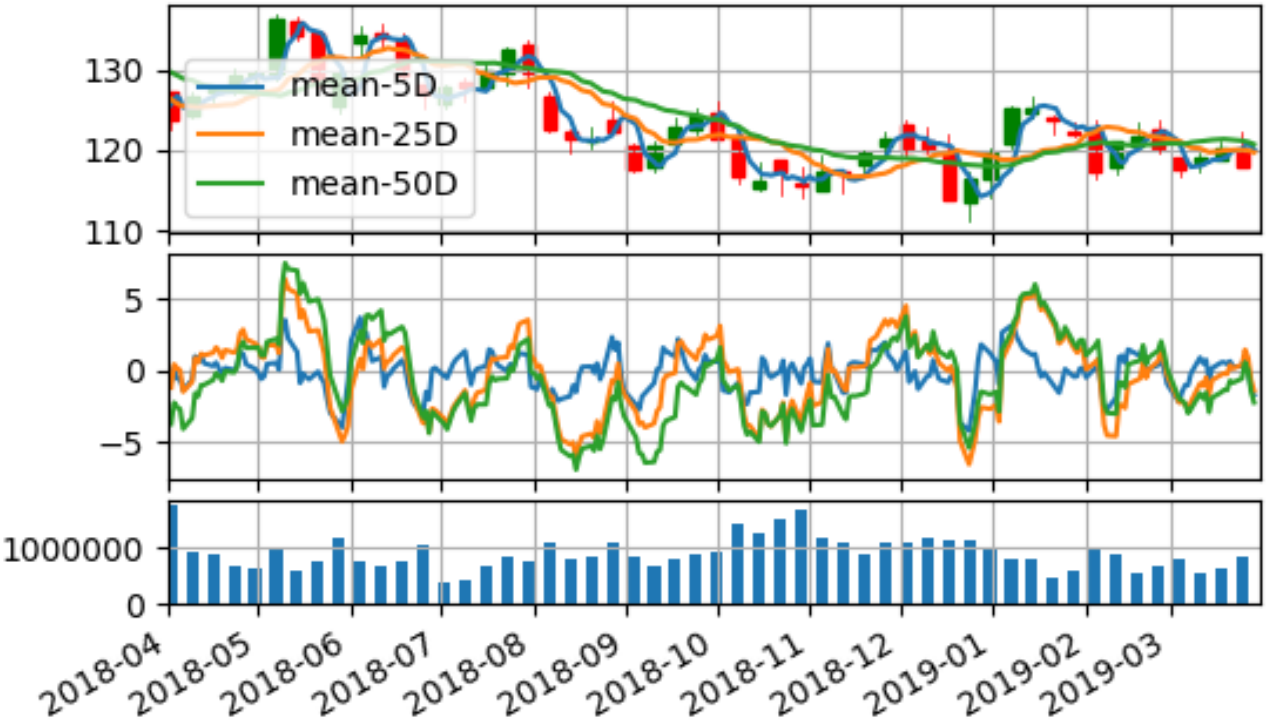
# resampleとrollingに対する処理

| 関数 | 処理内容 | resample | rolling |
|---|---|---|---|
| count | 数を数える | ○ | ○ |
| sum | 和 | ○ | ○ |
| var | 分散 | ○ | ○ |
| mean | 平均値 | ○ | ○ |
| median | 中央値 | ○ | ○ |
| std | 標準偏差 | ○ | ○ |
| max | 最大値 | ○ | ○ |
| min | 最小値 | ○ | ○ |
| quantile | 分位数・パーセンタイル | ○ | ○ |
| nunique | ユニークな要素の数 | ○ | |
| first | 最初の値 | ○ | |
| last | 最後の値 | ○ | |
| ohlc | ohlcデータ | ○ | |
| prod | 積 | ○ | |
| size | サイズ | ○ | |

| 関数 | 処理内容 | resample | rolling |
|---|---|---|---|
| corr | 相関係数 | | ○ |
| cov | 畳み込み | | ○ |
| skew | 歪度 | | ○ |
| kurt | 尖度 | | ○ |
| apply | 関数を適応 | | ○ |
| aggregate | 集計 | | ○ |

# matplotlibを使った複数グラフの配置

- 1枚のfigureに複数のグラフを配置
  - G=matplotlib.gridspec.GridSpec(6, 1）
    → 縦6×横1の領域を生成
  - ax0 = plt.subplot(G[:2, 0])
    → 縦3個分(0-2番目)、横1個分(0番目）の枠で一つ描画領域を生成



# matplotlibでできること（一部）

- フォントのサイズを変える
  - 図全体を変える：plt.rcParams["font.size"] = 16
  - 個別に変える

    plt.title(“Title”, fontsize=20)、plt.xlabel(“xlabel”, fontsize=12)、plt.ylabel(“ylabel”, fontsize=12)、plt.legend(fontsize=16)、plt.tick_params(labelsize=10)など
- 折れ線グラフの色や書式を変える
  - plot()の中で以下のパラメータを設定

    線の太さ：linewidth, 線のスタイル：linestyle（‘solid’ (実線), ‘dashed’ (破線), ‘dashdot’ (破線&点線), ‘dotted’ (点線) ), 色：color, マーカの種類：marker, マーカのサイズ：markersize, マーカの輪郭線の太さ：markeredgewidth、マーカの輪郭線の色：markeredgecolor, マーカの塗りつぶしの色：markerfacecolor、マーカの塗りつぶし方：fillstyle, アンチエイリアス（線を滑らかにする機能）
- 折れ線グラフの色や書式を変える
  - bar()の中で以下のパラメータを設定

    太さ:width、下側の余白：bottom、色：color、棒の輪郭の色：edgecolor、棒の輪郭の太さ: linewidth、yerr: y軸方向にエラーバーを追加、xerr: x軸方向にエラーバーを追加、ecolor: エラーバーの色、capsize：エラーバーの傘のサイズ

# 第7回の課題：Webスクレイピング

**課題１：ヘッダ情報の取得**

東京大学のUTokyo Focusのタイトル（タグ"TITLE"でマークアップされているテキスト）を取り出して、**変数titleに代入してください**。

**課題２：UTokyo FOCUS>IN THE NEWS一覧の取得**

東京大学のUTokyo Focusのページ のHTMLからタグ情報を読み取って、IN THE NEWSの一覧を取得して、そのテキストを**newsという名前のリストに代入してください**。