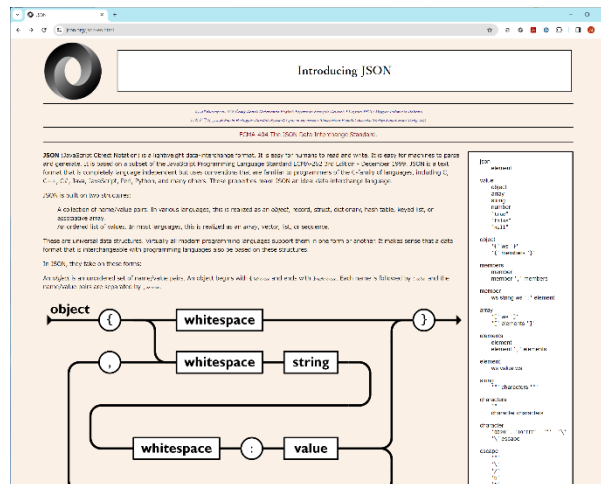


JSON

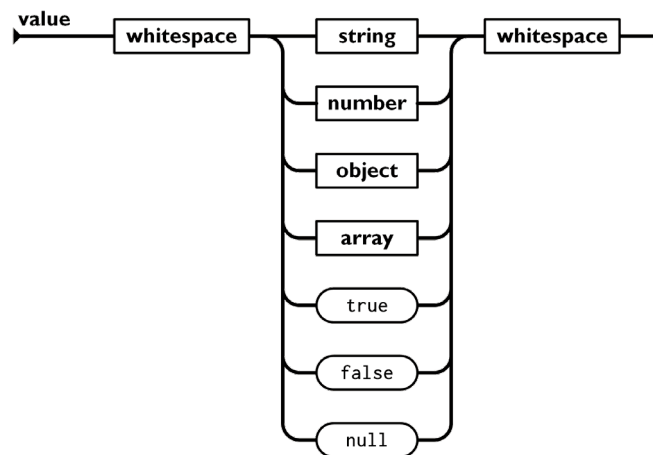
JavaScript Object Notation の頭文字。

JavaScript の Object 型データをソースコード内に記述する方法として開発された。

その記述法の手軽さと、記述内容の理解のしやすさから Python、Ruby、Java、C#、C++などのインターネットを媒介にする言語でデータ交換フォーマットとして使われている。



JSON で表現可能なデータ型は、「オブジェクト」、「配列」、「数値」、「文字列」、「真偽値」、「null」の 6 種類。



■JSON と Python の関係

Python には JSON の読み書きに必要なパッケージが標準で用意されている。

Python のデータ型	JSON のデータ型
リスト型(list/tuple)	配列(array)
辞書型(dict)	オブジェクト(object)
数値型(int/float)	数値型(string)
文字列型(str)	文字列型(string)
ブール型(bool)	真偽型(boolean)
None	ヌル (null)

■JSON ファイルの読み書き

◆JSON への書き込み

```
#1
import json
items = [
    {"name": "Aoki", "age": 30},
    {"name": "Ishida", "age": 32},
    {"name": "Inoue", "age": 29}
]

with open('test.json', 'w', encoding='utf-8') as fp:
    json.dump(items, fp, indent=4)
```

test.json に書き込んだデータ

```
[  
  {  
    "name": "Aoki",  
    "age": 30  
  },  
  {  
    "name": "Ishida",  
    "age": 32  
  },  
  {  
    "name": "Inoue",  
    "age": 29  
  }  
]
```

◆JSON に書き込んだデータを読み出す

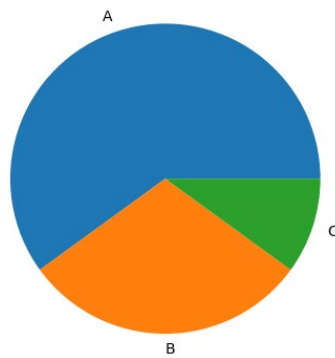
#1 で書き込んだ JSON ファイルを#2 で読み込み、表示します。

```
#2  
import json  
with open('test.json', 'r', encoding='utf-8') as fp:  
    data = json.load(fp)  
print(data[0]['name'], data[0]['age'])  
print(data[1]['name'], data[1]['age'])
```

■JSON のデータで円グラフを描く

```
#3
import matplotlib.pyplot as plt

values = [60, 30, 10]
labels = ['A', 'B', 'C']
plt.pie(values, labels=labels)
plt.show()
```



■人口推移グラフの作成

日本政府の統計「e-Stat」のサイトから、「人口推計（大正9年～平成12年）」

選択条件: [ファイル](#) / [人口推計](#) / [人口推計](#) / [人口推計](#) / [長期時系列データ](#) / [我が国の推計人口（大正9年～平成12年）](#) [政府統計一覧に戻る（すべて解除）](#)

データセット

キーワードを入力

✕

🔍

検索オプション

☒提供分類、表題を検索 ☒データベース、ファイルを検索 [検索のしかた](#)

データセット情報

人口推計 / 長期時系列データ 我が国の推計人口（大正9年～平成12年）

表示・ダウンロード

📄

EXCEL

[URLをコピー](#)

[データセット一覧に戻る](#)

政府統計名	人口推計	詳細
政府統計コード	00200524	
調査の概要	人口推計は、国勢調査による人口を基に、その後の各月における出生・死亡、入国・出国などの人口の動きを都の人口動態資料から得ることで、毎月1日現在の男女別、年齢階級別の人口を推計しています。また、毎年10月1日現在の全国各歳別結果及び都道府県別結果も推計しています。 推計結果は、各種白書や国勢機関における人口分析、経済分析等の基礎資料として利用されています。	
提供統計名	人口推計	
提供分類1	長期時系列データ	
提供分類2	我が国の推計人口（大正9年～平成12年）	
表番号	1	
表分類	全国	
統計表名	男女別人口（各年10月1日現在） - 総人口（大正9年～平成12年）、日本人口（昭和25年～平成12年）	
データセットの概要		
統計分野（大分類）	人口・世帯	
統計分野（小分類）	人口	
担当機関	総務省	
担当課室	統計局統計調査部国勢統計課	
政府統計URL	http://www.stat.go.jp/data/jinsui/index.htm	
統計の種類	基幹統計	
調査年月	2000生	
公開年月日時分	2007-12-18 19:44	
提供周期	-	
集計地域区分	該当なし	

e-Stat から得られたデータは Excel データ。

これを JSON 形式に作り替える。

```
[
  {
    "year": 1950,
    "total": 83200,
    "man": 40812,
    "woman": 42388
  },
  {
    "year": 1951,
    "total": 84541,
    . . . .
  }
```

こうしたデータは、国や行政機関が「オープンデータ」として公開しています。オープンデータとして公開されているものは、自由に使えて再配布もできます。つまり、著作権がありません。

データには、国政超査による人口統計や、気象庁の気象情報、災害情報、地方自治体では公共施設、医療機関、防犯災害情報などを公開しています。また、多くの国々で同様の情報を公開しています。

また、個人や企業などが公開している情報には、既に著的財産としての制限のない「Pd」（パブリックドメイン）や、範囲内でのライセンス使用が許されている「CC」（クリエイティブコモンズ）などがある。

なお、CC の種類には、

表示 (BY)	著作権者の表示が必要
非営利 (NC)	非営利目的での使用に限定
改変禁止 (ND)	改変を禁止
継承 (SA)	改編して公開する場合、もとの作品のライセンスを継承する必要がある

◆さまざまなオープンデータ

「青空文庫」では、文学作品も公開されている。

青空文庫

メイン

お知らせ

別館

資料

運営

www.aozora.gr.jp 内を検索

Googlebinggoo

インターネットの電子図書館、青空文庫へようこそ。

「[青空文庫、新館準備中](#)」

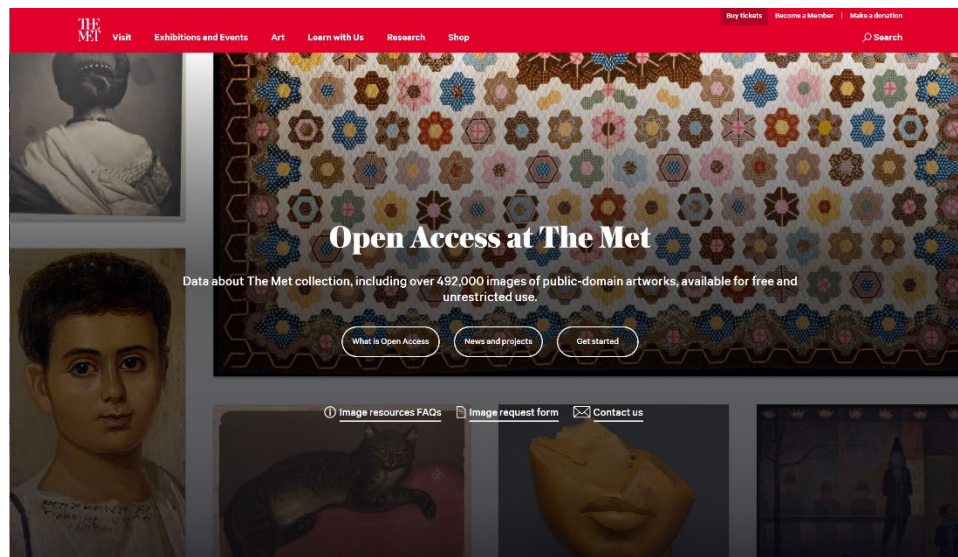
初めての方はまず「[青空文庫早わかり](#)」をご覧ください。

ファイル利用をお考えの方は、[こちら](#)をご一読ください。

「[青空文庫収録ファイルを用いた朗読配信をお考えのみなさまへ](#)」

メインエリア											
青空文庫早わかり	青空文庫の使い方と約束事を紹介しています。初めての方、ファイルやキャプチャーの取り扱いについて知りたい方も、こちらへどうぞ。										
総合インデックス	作家名、作品名の50音別に、公開作品と入力・校正作業中の作品を一覧できるインデックスです。公開中の作品を探すときは、下の近道もご利用ください。										
公開中 作家別:	あ行	か行	さ行	た行	な行	は行					
	ま行	や行	ら行	わ行	他						
公開中 作品別:	あ	い	う	え	お	か	き	く	け	こ	
	さ	し	す	せ	そ	た	ち	つ	て	と	
	な	に	ぬ	ね	の	は	ひ	ふ	へ	ほ	
	ま	み	む	め	も	や	ゆ	よ	ら	り	ろ
	わ	を	を	を	を	を	を	を	を	を	他
作業中:	作業別・作品別										
青空文庫 分野別リスト	分野別に公開作品を一覧できる、インデックスです。										

- ・ ニューヨーク「メトロポリタン美術館」

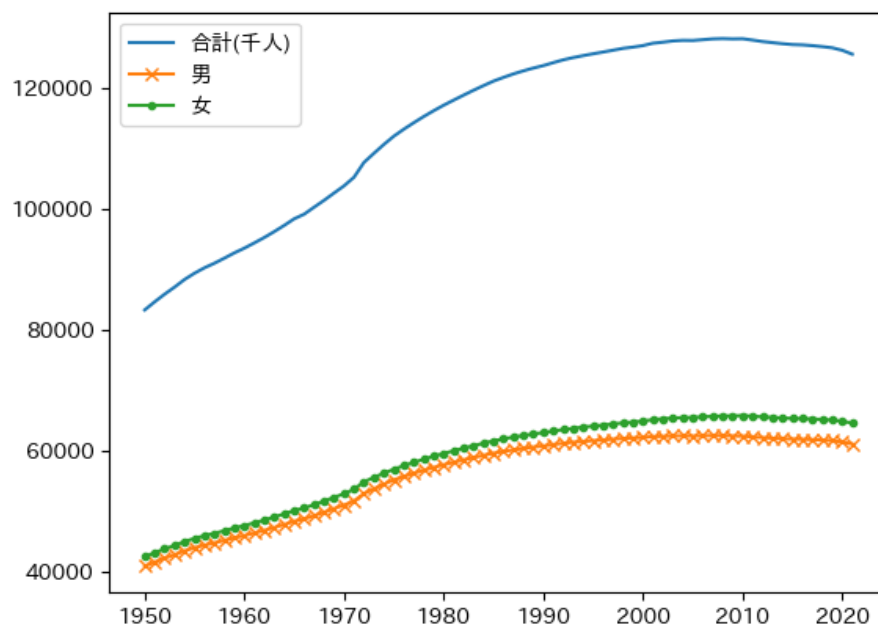


収蔵作品にアクセスができる。

◆matplotlib の日本語化

起動のためには、「japanize-matplotlib」が必要なので、事前に Anaconda の PowerShell などを使ってインストールしておく。

```
$ pip install japanize-matplotlib
```



```
import json, japanize_matplotlib
import matplotlib.pyplot as plt

# 人口推移の JSON ファイルを読む
data = json.load(open('pop.json', encoding='utf-8'))

# 複数の線グラフを描画するようにデータを分割
x, totals, man, woman = [], [], [], []
for row in data:
    x.append(row['year']) # 西暦年
    totals.append(row['total']) # 男女合計
    man.append(row['man']) # 男性
    woman.append(row['woman']) # 女性

# グラフを描画
p_total = plt.plot(x, totals, label='合計(千人)')
p_man = plt.plot(x, man, marker='x', label='男')
p_woman = plt.plot(x, woman, marker='.', label='女')
plt.legend() # 凡例を表示
plt.show()
```


<補>JSON を使った円グラフ

```
import matplotlib.pyplot as plt
import json

# JSON データ。円グラフのための値とラベルを含む
data_json = '{"values": [60, 30, 10], "labels": ["A", "B", "C"]}'

# JSON データを解析する
data = json.loads(data_json)

# JSON データから取得した値とラベルを使用して円グラフを描画
plt.pie(data["values"], labels=data["labels"])
plt.show()
```

`json.loads(data_json)`は、JSON 形式の文字列を Python のデータ構造（この場合は辞書）に変換するために使用されます。ここでの具体的な説明は以下の通りです。

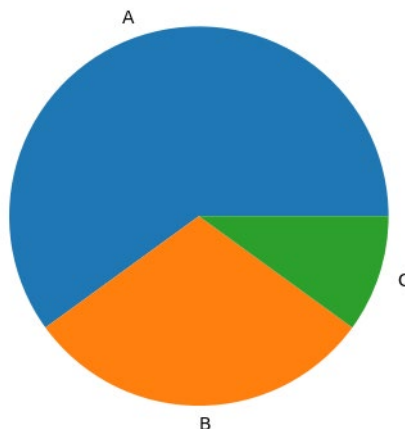
json: json は標準ライブラリで、JSON データを扱うためのモジュール

loads: この関数は `load string` の略で、文字列形式の JSON データを Python のデータ構造に変換するために使う

data_json: ここで扱うデータ。

この例では `{"values": [60, 30, 10], "labels": ["A", "B", "C"]}`

`json.loads(data_json)`を実行すると、`data_json`に含まれる JSON データが Python の辞書に変換される。この辞書はキーと値のペアを持ち、この例では `{"values": [60, 30, 10], "labels": ["A", "B", "C"]}` の形になる。これにより、プログラムは `data["values"]` や `data["labels"]` といった方法で、それぞれの値やラベルにアクセスできるようになる。



■テキストデータとバイナリデータ

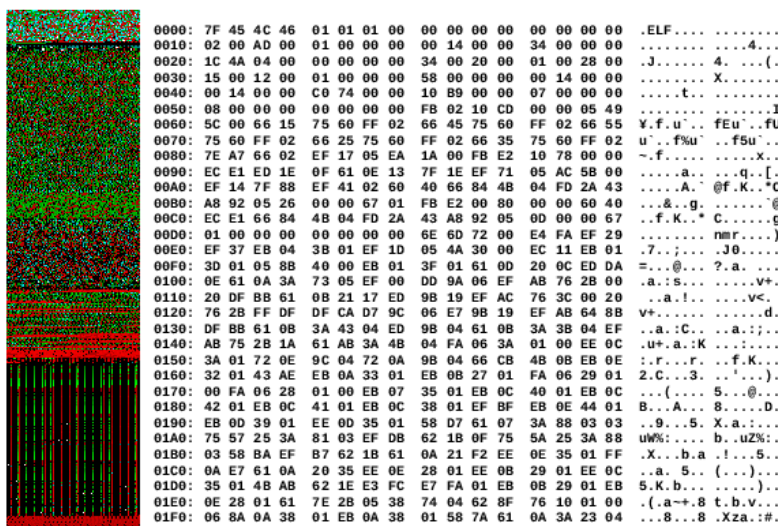
- ・テキストデータ

英文字、記号などで構成されたデータ

JSON、html、CSV、プログラミング言語のソースなど

- ・バイナリデータ

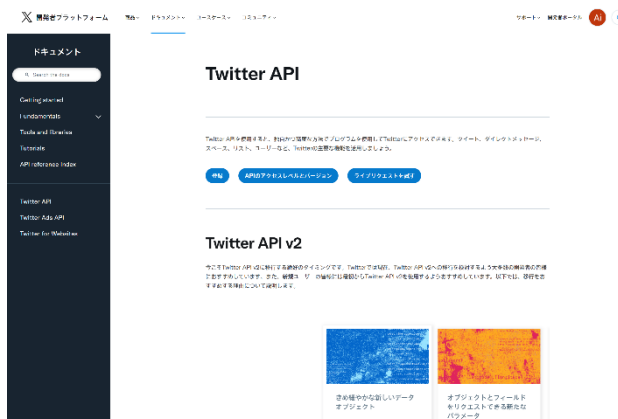
コンピュータの全範囲のデータを扱える (ex.画像データなど)



JSON はさまざまな形式のデータを変換して、WEB の共通データ形式としての役割が大きい。

■Web API と JSON

API (Application Programming Interface) の略。アプリケーションをプログラミングするためのインターフェース。多くの Web サービスでは Web API を提供している。開発者は、API を自分のプログラムの中で使うことで、そのサービスを自分のプログラミングの中でも使えます。



■アンケートのグラフ表示

サイト「青いクジラ」にある簡易アンケートの結果を利用する。
ここでは「質問8」の好きな OS の結果を使う。



@aoikujira.com

公開されている php データを取得して、実際のデータを見てみる。

```
{
  "result":true,
  "author":"名無し",
  "question":"好きな OS は?",
  "answers":[
    {
      "label":"Windows",
      "point":62
    },
    {
      "label":"macOS",
      "point":40
    },
    . . . . .
  ],
```

◆Web API で JSON データの取得

Web API から JSON データを取得する。`requests.get` メソッドを使うことで手軽に Web 上のリソースを取得できる。

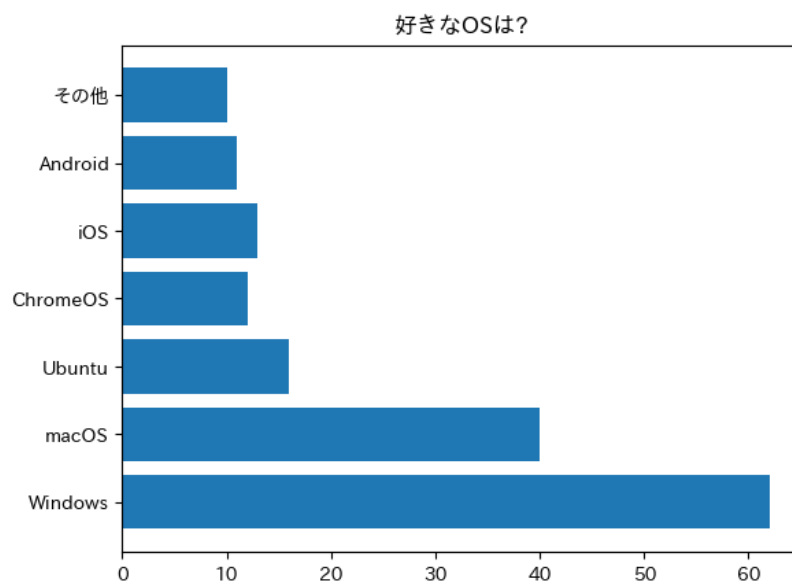
```
import json, requests, japanize_matplotlib
import matplotlib.pyplot as plt

# Web API から好きな OS に関する JSON データを取得
url = 'https://api.aoikujira.com/like/api.php?m=get&item_id=8'
r = requests.get(url)

# 取得した JSON を Python で扱えるように変換
data = json.loads(r.text)

# グラフ描画のためにデータを分ける
labels, values = [], []
for it in data['answers']:
    labels.append(it['label'])
    values.append(it['point'])

# グラフを描画
plt.barh(labels, values)
plt.title('好きな OS は?')
plt.show()
```



■ジャンケンデータの蓄積とグラフ表示

コンピュータとジャンケン「グー (0)」、「チョキ (1)」、「パー (2)」を実施して、結果を蓄積するプログラム。終了は「3」(jyanken.py)

この数値を使うことで、ジャンケンの勝敗を数式で求めることができる。

$$\text{判定値} = (\text{相手の手} - \text{自分の手} + 3) \% 3$$

判定値が0なら「あいこ」、1ならば「勝ち」、2ならば「負け」になる。

このプログラムを繰り返し実行することで、同じディレクトリ内に「jyanken_history.json」ファイルが生成され、結果が書き込まれる。

◆JSON 結果の分析

結果は、「jyanken_analyzer.py」で分類し表示させる。

JSON データの中身は、

```
com : 1,  
user : 0,  
result 勝ち
```

のように保存、蓄積されていく。

