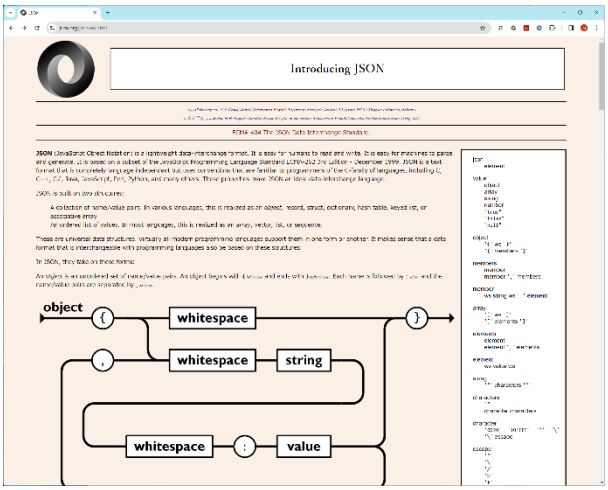
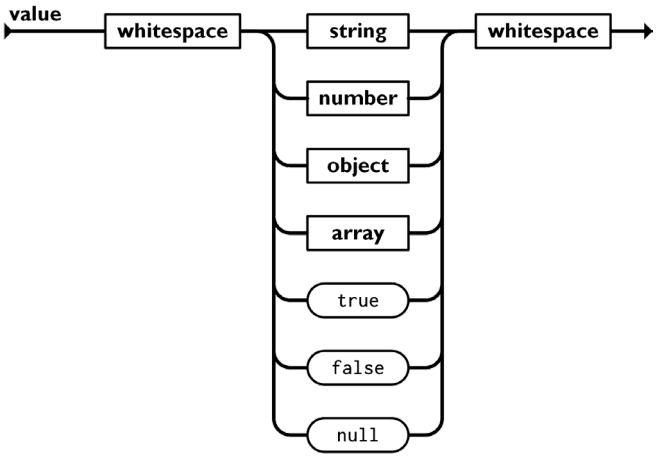
JSON

`JavaScritp 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

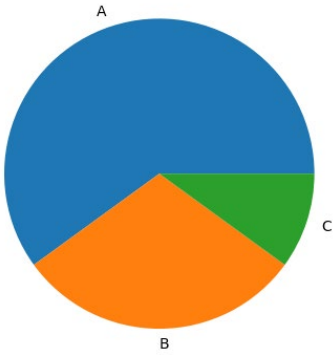
#1 で書き込んだ JSON ファイルを#2 で読み込み、表示します。

```
#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)」

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。

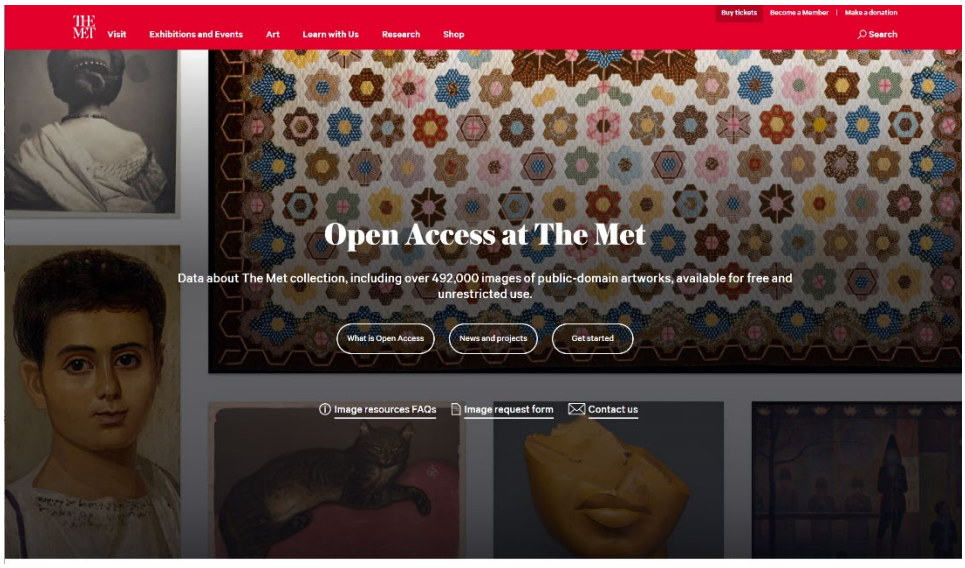
なお、CC の種類には、

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。

・ニューヨーク「メトロポリタン美術館」

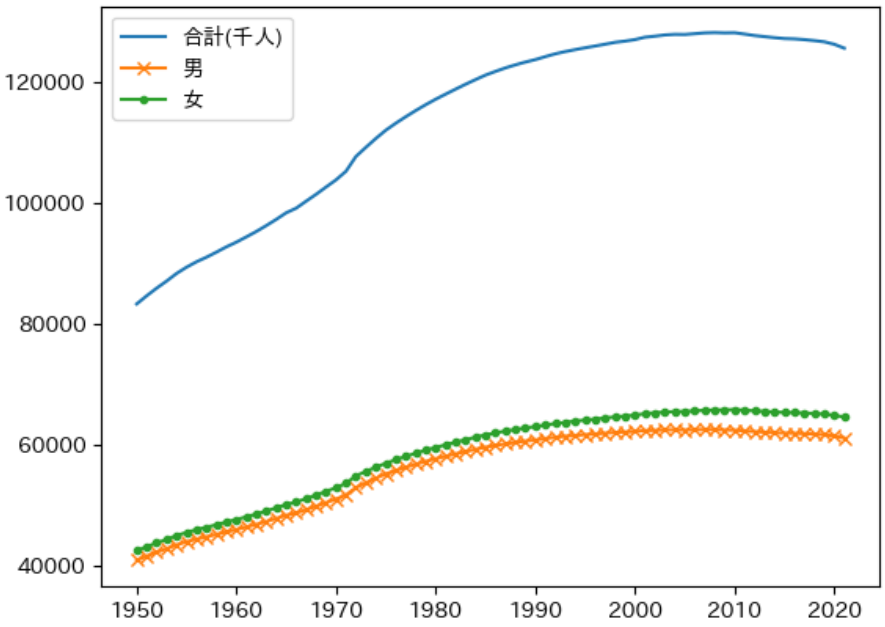


収蔵作品にアクセスができる。

◆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()
```

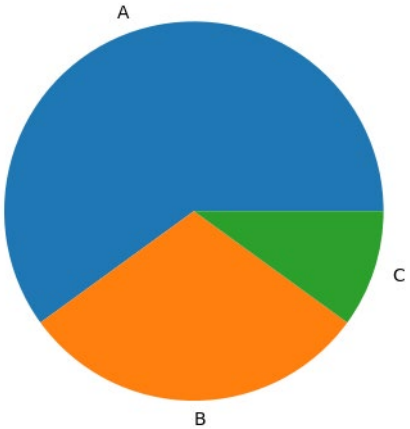
`json.loads(data_json)`は、JSON 形式の文字列を Python のデータ構造（この場合は辞書）に変換するために使用されます。ここでの具体的な説明は以下の通りです。

`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ーデータ

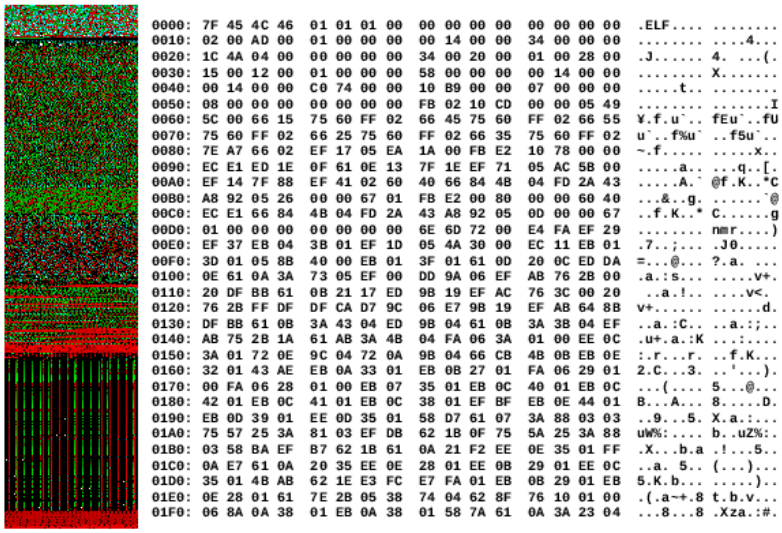
・テキストデータ

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

・バイナリーデータ

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）



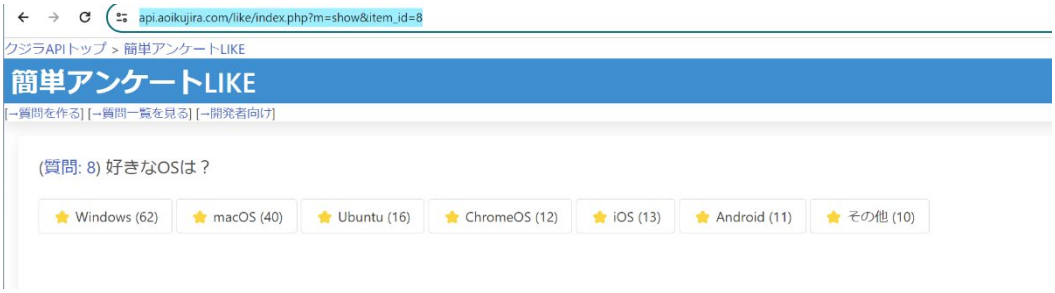
JSON はさまざまな形式のデータを変換して、WEB の共通データ形式としての役割が大きい。

■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。

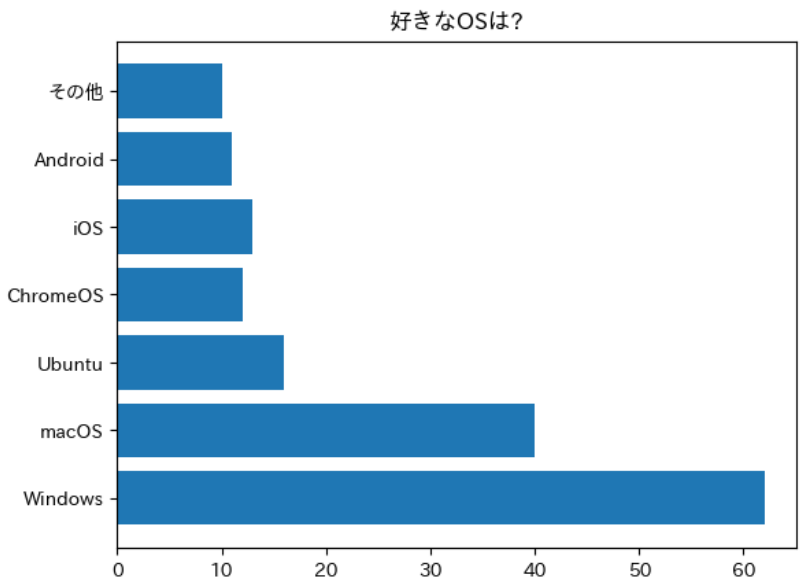
```
import json, requests, japanize_matplotlib
import matplotlib.pyplot as plt

# 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。

判定値＝(相手の手 ― 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_analizer.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。