

ご当地グルメマップを作ろう

Let's make! Local Food Map

PyCon APAC 2023 Day2

Hiroshi Sano / 佐野浩士



Agenda

- トークのモチベーション
 - 今回のトークでできること
1. お店情報をデータにする: WEBスクレイピング
 2. データを整形をする: CSVファイルにする
 3. データを使ってみる: Googleマイマップで呼び出す
- まとめ

Today's Document

本日の資料は公開されています

- GitHubリポジトリ: コードとスライド
<https://github.com/hrsano645/pyconapac2023-local-food-map>
- スライド: Speaker Deck
<URL載せ替えます>

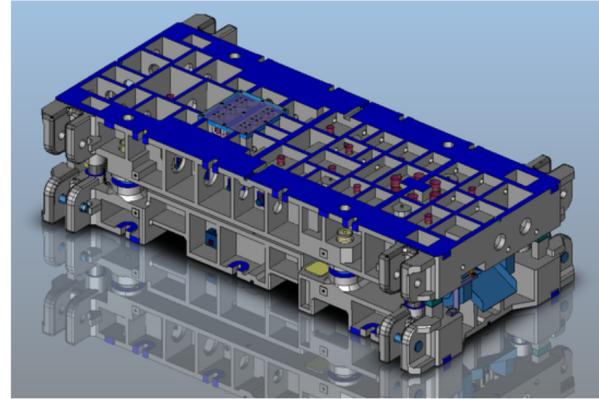
後日トライしたい方は参考にしてみてください

Self Infroduction

Hiroshi Sano(佐野浩士) [@hrs_sano645](#)

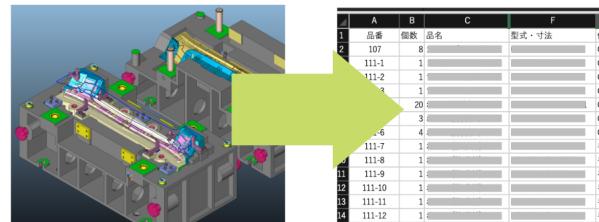
- 🌏: Shizuoka, Eastern part🗻
- 🏢: 株式会社佐野設計事務所 CEO
- 💬🤝
 - 🐍: PyCon mini Shizuoka Stuff / Shizuoka.py / Unagi.py / Python駿河
 - CivicTech, Startup Weekend Organizer
- Hobby: Camp🏕, DIY🛠️, IoT💡





3D機械設計 / 3Dモデリング

自動車向けプレス金型の3D設計
金型設計関連の3Dモデリング
製品データの3Dモデリング



製造業DX支援



設計データをデジタル化
製造業DX取り組みサポート

トークのモチベーション

- ご当地グルメを情報収集してマップを作りましょう！ 食べに行きましょう！
 - ご当地グルメ=B級グルメのこと
- PyCampを終えた人に: Pythonでデータを集めて作り利用するプロセスを学べます
 - 初学者向けの内容です
 - 話すこと: トークお題をPythonで実装する過程
 - 話さないこと: Pythonの基礎や文法の解説

PyCamp:Python Boot Campとは

- ・日本全国で開催されているPythonの初学者向けイベント
- ・Pythonの基礎を学び、簡単なプログラムを作る
 - 専用テキストを元に講師とTAがサポート



The screenshot shows the homepage of the "Python Boot Camp Text" documentation. It features a logo with a stylized figure and the text "Python Boot Camp". The main title "Python Boot Camp Text" is displayed prominently. Below the title, there is a brief introduction and links to GitHub and Read the Docs. A sidebar on the left contains navigation links for "Python Boot Camp Text ドキュメント", "Q. 検索", "Python Boot Camp 運営マニュアル", and "Python Boot Camp テキスト". The main content area includes sections for "目次" (Table of Contents) and "ライセンス" (License), with a Creative Commons Attribution 4.0 International License logo. At the bottom, it says "v: latest".

このトークはその続きからトライできるコンテンツを目指して作りました

PyCampの様子

静岡県3回目の様子

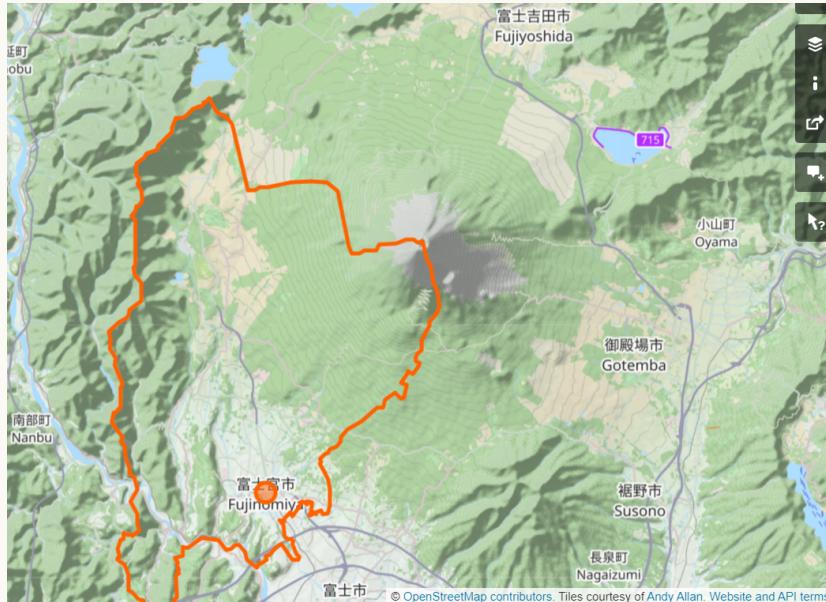


今回のお題: 本題のご当地グルメ

富士宮焼きそば



- 主に静岡の富士宮市周辺で食べられる焼きそば
- 麵は富士宮周辺でしか手に入らない。まさにローカルフード
- B級グルメグランプリ殿堂入り、NYで開催されたコナモングランプリで優勝した



地元の人は、多分月に数回は食べてる

ところで、みなさん

もう食べたくないになりましたね？ 😭

このあとはパーティ🎉ですし、お腹空きましたね 😭

富士宮焼きそばマップを作りましょう

今回のトークでできること

1. お店情報をデータにする: WEBスクレイピング
2. データを整形をする: CSVファイルにする
3. データを使ってみる: Googleマイマップで呼び出す

データを読む/取り込む



データを加工/出力する



データを使う/活用する

今回のトーキでできること

1. お店情報をデータにする: WEBスクレイピング
2. データを整形をする: CSVファイルにする
3. データを使ってみる: Googleマイマップ



1. データを読む/取り込む
2. データを加工/出力する
3. データを使う/活用する

データを読む/取り込む



データを加工/出力する



データを使う/活用する

1. データを読む/取得する

- 🔎WEBスクレイピングで収集する
- [付録] 📸画像識別で加工を試みる

機械可読性はどちらも微妙

WEBスクレイピングはまだやりやすい

ご当地グルメの情報はどこにあるか

- 地域の情報を収集
- **その情報は機械可読ができるか？**
 - 大体が紙ベースが多い   (画像識別の手段は使える)

観光情報を探ってみると…

- 市役所で紹介されてたり
- 観光協会で紹介されてたり
- ご当地グルメの公式サイト (よく〇〇学会とも言われる)

今回は、富士宮焼きそば学会の公式サイトを例にしています。

<https://umya-yakisoba.com/shop/>

利用するライブラリ

- requests: HTTPアクセス→情報取得（今回はHTML）
- BeautifulSoup4: HTML（マークアップ言語）解析と抽出

```
pip install requests  
pip install beautifulsoup4
```

※:スライドのコードは説明向けです。そのままだと動かないこともあります
資料のリポジトリから動作するスクリプトをDL可能です。

この構造からBeautifulSoup4を使って必要な情報を取り出します。

The screenshot shows a web browser window with the following details:

- Header:** Fujinomiya Yakisoba Society 富士宮やきそば学会
- Title:** 富士宮やきそば店一覧
- Content:** A list of shops with their details:
 - あ好焼 あき**
富士宮市野中東町112-1
1
0544-27-0004
定休日：火曜日
 - あさ家**
富士宮市元城町13-20
0544-22-3029
定休日：なし
- HTML Inspector:** An open developer tools panel showing the DOM structure. A red box highlights the HTML for the first shop entry, which corresponds to the shop listed on the left.

The highlighted HTML structure is as follows:

```
<a href="https://umya-yakisoba.com/shop/3776/">
  <figure class="p-shopthumb-ph-no">...</figure>
  <div class="p-shopthumb-text"> == $0
    <div>お好焼 あき</div>
    <div>富士宮市野中東町112-1</div>
    <div>0544-27-0004</div>
    <div>定休日：火曜日</div>
  </div>
</a>
```

```
> <header class="article-header entry-header">...</header>
<div class="article">
  <div class="p-shopList" style="flex">
    <a href="https://umya-yakisoba.com/shop/3776/">
      <figure class="p-shopthumb-ph-no" style="flex">...</figure>
      <div class="p-shopthumb-text" style="flex">...</div>
        <div>お好焼 あき</div>
        <div>富士宮市野中東町112-1</div>
        <div>0544-27-0004</div>
        <div>定休日：火曜日</div>
      </div>
    </a>
    <a href="https://umya-yakisoba.com/shop/3777/">...</a>
    <a href="https://umya-yakisoba.com/shop/3774/">...</a>
    <a href="https://umya-yakisoba.com/shop/3616/">...</a>
    <a href="https://umya-yakisoba.com/shop/3772/">...</a>
    <a href="https://umya-yakisoba.com/shop/3779/">...</a>
    <a href="https://umya-yakisoba.com/shop/3618/">...</a>
    <a href="https://umya-yakisoba.com/shop/3778/">...</a>
    <a href="https://umya-yakisoba.com/shop/3780/">...</a>
    <a href="https://umya-yakisoba.com/shop/3782/">...</a>
    <a href="https://umya-yakisoba.com/shop/3627/">...</a>
    <a href="https://umya-yakisoba.com/shop/3783/">...</a>
    <a href="https://umya-yakisoba.com/shop/3784/">...</a>
    <a href="https://umya-yakisoba.com/shop/3633/">...</a>
    <a href="https://umya-yakisoba.com/shop/3769/">...</a>
```

構造の中にあるタグから必要な情報を取得する

```
import requests
from bs4 import BeautifulSoup

url = "https://umya-yakisoba.com/shop/"
shopinfo_list = []
res = requests.get(url)
soup = BeautifulSoup(res.text, 'html.parser')

# ここではdiv.p-shopList > a にURLがある
# その中にお店情報がまとまっているので、aタグから取り出す
shopinfo_tags = soup.find(
    'div', class_='p-shopList'
).find_all("a")
```

aタグの中にあるタグから必要な情報を取得する

```
> <header class="article-header entry-header">...</header>
<div class="article">
  <div class="p-shoplist"> flex
    <a href="https://umya-yakisoba.com/shop/3776/">
      <figure class="p-shopthumb-ph-no">...</figure>
      <div class="p-shopthumb-text"> == $0
        <div>お好焼 あき</div>
        <div>富士宮市野中東町112-1</div>
        <div>0544-27-0004</div>
        <div>定休日：火曜日</div>
      </div>
    </a>
    <a href="https://umya-yakisoba.com/shop/3777/">...</a>
    <a href="https://umya-yakisoba.com/shop/3774/">...</a>
    <a href="https://umya-yakisoba.com/shop/3616/">...</a>
    <a href="https://umya-yakisoba.com/shop/3772/">...</a>
    <a href="https://umya-yakisoba.com/shop/3779/">...</a>
    <a href="https://umya-yakisoba.com/shop/3618/">...</a>
    <a href="https://umya-yakisoba.com/shop/3778/">...</a>
    <a href="https://umya-yakisoba.com/shop/3780/">...</a>
    <a href="https://umya-yakisoba.com/shop/3782/">...</a>
    <a href="https://umya-yakisoba.com/shop/3627/">...</a>
    <a href="https://umya-yakisoba.com/shop/3783/">...</a>
    <a href="https://umya-yakisoba.com/shop/3784/">...</a>
    <a href="https://umya-yakisoba.com/shop/3633/">...</a>
    <a href="https://umya-yakisoba.com/shop/3769/">...</a>
```

```
for shopinfo_tag in shopinfo_tags:
    shopdata = {}
    # divは上から店名、住所、電話番号、定休日。
    # ここではurlと店名だけまとめたリストを作る
    shopdata['specurl'] = shopinfo_tag.get('href')
    shopdata['店名'] = replace_text(
        shopinfo_tag.find_all("div")[1].text
    )

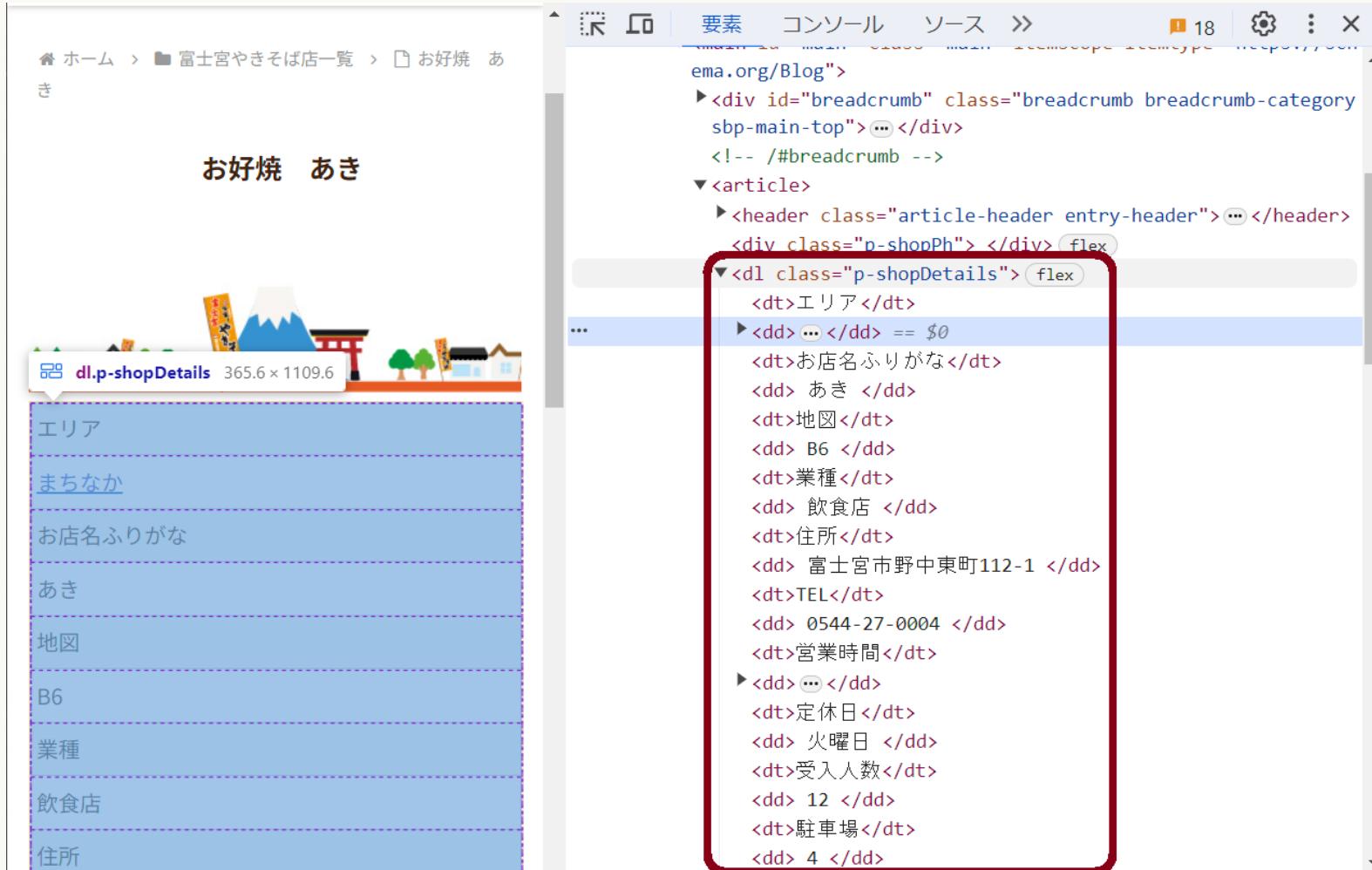
    shopinfo_list.append(shopdata)
```

※ replace_text関数:
店名に出てくる空白文字を置き換える独自関数
(後述します)

お店の詳細URL、店名が集まりました🎉

```
>>> shopinfo_list
[{'specurl': 'https://umya-yakisoba.com/shop/3776/', '店名': 'お好焼 あき'},
 {'specurl': 'https://umya-yakisoba.com/shop/3777/', '店名': 'あさ家'},
 {'specurl': 'https://umya-yakisoba.com/shop/3774/', '店名': 'あるばとろす'},
 {'specurl': 'https://umya-yakisoba.com/shop/3616/', '店名': 'いつぶく亭'},
 ...
]
```

収集した詳細URLのリストを使って、お店情報収集



The screenshot shows a browser window with developer tools open. On the left is the website content, which includes a header 'お好焼 あき' and a sidebar with categories like 'エリア', 'まちなか', 'お店名ふりがな', 'あき', '地図', 'B6', '業種', '飲食店', and '住所'. A tooltip indicates the sidebar has a width of 365.6 x 1109.6 pixels. On the right is the element inspector with the following HTML structure:

```
<div id="breadcrumb" class="breadcrumb breadcrumb-category sbp-main-top">...</div>
<!-- /#breadcrumb -->
<article>
  <header class="article-header entry-header">...</header>
  <div class="p-shopPh"> </div> flex
  <dl class="p-shopDetails" style="flex">
    <dt>エリア</dt>
    <dd>...</dd> == $0
    <dt>お店名ふりがな</dt>
    <dd> あき </dd>
    <dt>地図</dt>
    <dd> B6 </dd>
    <dt>業種</dt>
    <dd> 飲食店 </dd>
    <dt>住所</dt>
    <dd> 富士宮市野中東町112-1 </dd>
    <dt>TEL</dt>
    <dd> 0544-27-0004 </dd>
    <dt>営業時間</dt>
    <dd>...</dd>
    <dt>定休日</dt>
    <dd> 火曜日 </dd>
    <dt>受入人数</dt>
    <dd> 12 </dd>
    <dt>駐車場</dt>
    <dd> 4 </dd>
  </dl>

```

A red box highlights the `dt` and `dd` elements under the `p-shopDetails` `dl` element, specifically the first item: 'エリア' and its value 'あき'.

`dt` と `dd` タグは記述リスト、説明リストと呼ばれるHTMLタグ

```
> <header class="article-header entry-h1">
  <div class="p-shopPh"> </div> flex
  <dl class="p-shopDetails"> flex
    <dt>エリア</dt>
    > <dd>...</dd> == $0
    <dt>お店名ふりがな</dt>
    <dd> あき </dd>
    <dt>地図</dt>
    <dd> B6 </dd>
    <dt>業種</dt>
    <dd> 飲食店 </dd>
    <dt>住所</dt>
    <dd> 富士宮市野中東町112-1 </dd>
    <dt>TEL</dt>
    <dd> 0544-27-0004 </dd>
    <dt>営業時間</dt>
    > <dd>...</dd>
    <dt>定休日</dt>
    <dd> 火曜日 </dd>
    <dt>受入人数</dt>
    <dd> 12 </dd>
    <dt>駐車場</dt>
    <dd> 4 </dd>
```

```
for shopinfo in shopinfo_list:
    # URLから店舗情報を取得
    res = requests.get(shopinfo['specurl'])
    soup = BeautifulSoup(res.text, 'html.parser')

    # dl.p-shopDetails > dt/dd構造でdtが項目、
    # ddが値になっている。これを辞書形式にする
    shopspecs = {}
    for dt, dd in zip(
        soup.find('dl', class_='p-shopDetails').find_all('dt'),
        soup.find('dl', class_='p-shopDetails').find_all('dd')
    ):
        # 値に 改行や空白文字があるので取り除く
        shopspecs[dt.text] = replace_text(dd.text)

    # 店舗情報をマップ情報に追加
    shopinfo.update(shopspecs)
    # INFO:ここにランダム待機時間があるとよさそう
```

最終的にできるデータ

```
>>> from pprint import pprint
>>> pprint(shopinfo_list)
[{'TEL': '0544-27-0004',
 'specurl': 'https://umya-yakisoba.com/shop/3776/',
 'お店名ふりがな': 'あき',
 'エリア': 'まちなか',
 '住所': '富士宮市野中東町112-1',
 '受入人数': '12',
 '営業時間': '10:00-21:00',
 '地図': 'B6',
 '定休日': '火曜日',
 '店名': 'お好焼 あき',
 '料金目安': '350~600円',
 '業種': '飲食店',
 '焼き方': 'お店',
 '調査員おすすめメニュー': 'キムチとチーズ入り',
 '調査員が見た特徴': 'キャベツとネギが多めに入っている',
 '駐車場': '4'},
 # 以下お店情報の辞書が続いて入る
 # ...]
```

上記コードの注意点

※: サイト上に見えない文字があることがあります → 文字列置換をしましょう

※:  WEBスクレイピングは注意が必要です

randomモジュールやtimeモジュールを組み合わせてランダム時間待機します
ポリシーを守りましょう

※: この例ではサイトのページネーションに対応していません
ページネーションについては資料のコードで対応しています

文字列の置き換え

見えない空白や改行などを取り除くと綺麗に使えます。

```
def replace_text(text: str) -> str:  
  
    replace_text_map = {  
        # 置き換え対象文字列:置き換え先の文字列  
        "\u3000": " ",  
        "\n": "",  
    }  
    replaced_text = text  
    for src, dst in replace_text_map.items():  
        replaced_text = replaced_text.replace(src, dst)  
    return replaced_text  
  
# 例:  
print(replace_text("お好焼\u3000さの"))  
# お好焼 さの
```

⚠️ WEBスクレイピングの注意点 !

※スクレイピング対象のサイトへ多数のアクセスはしないように注意

- よくあるトラブル
 - ループ構造が深くてアクセス回数が膨れる
 - エラーの時にリトライしすぎて
- 試すとき少し時間を置きながらアクセス
 - ランダム時間置いてみる
 - 回数リミットをつけて待つ
- サイトポリシーがあればそれに従う
 - 規約に掲載されている
 - クローラー向けのポリシー: robot.txtを見る

ランダム時間待機できる関数の例

```
import random
from time import sleep

def random_sleep(a: int,b: int) -> None:
    """
    aからbまでのランダムな秒数を待つ
    """
    time.sleep(random.randint(a,b))

# 2~5秒の間でランダムに待つ
random_sleep(2, 5)
```

データを読む/取り込む



データを加工/出力する



データを使う/活用する

2.データを加工/出力する

マップのもとになるデータを作成します

-  **情報を整理して表形式ファイルで書き出す**
- [付録]  **地理情報を集める**

情報を整理

どのフォーマットで書き出すか？

よくある地理データ構造、フォーマット形式

- CSV（区切り表形式、汎用性高）
- GeoJSON（WEB APIで広く流通しているJSON形式の地理情報向け）
- KML（XML形式）

CSVライブラリを使って書き出せます

`csv.DictWriter` を使うと辞書のキーを使って列見出しを用意できる

```
with open('mapdata.csv', 'w', newline='') as csvfile:  
    # お店の詳細情報の各項目:辞書のキー が部分的にあったりなかつたりしたので  
    # 全ての辞書のキーから全ての項目をカバーしたリストを生成する  
    fieldnames = list(set().union(*shopinfo_list))  
  
    writer = csv.DictWriter(csvfile, fidnames=fieldnames)  
    writer.writeheader()  
    for shopinfo in shopinfo_list:  
        writer.writerow(shopinfo)
```

お店ごとに詳細情報の項目数が違う

```
>>> shopinfo_list[0].keys()
dict_keys(['TEL', 'specurl', 'お店名ふりがな', 'エリア', '住所', '受入人数', '営業時間',
'地図', '定休日', '店名', '料金目安', '業種', '焼き方',
'調査員おすすめメニュー', '調査員が見た特徴', '駐車場'])
>>> shopinfo_list[1].keys()
dict_keys(['TEL', 'specurl', 'お店名ふりがな', 'エリア', '住所', '受入人数', '営業時間',
'地図', '定休日', '店名', '料金目安', '業種', '焼き方'])
```

それぞれの項目をすべて合わせたリストを作る

```
>>> print(list(set().union(*shopinfo_list)))
['エリア', '駐車場', '焼き方', '受入人数', 'specurl', 'TEL', '店名', '定休日', '業種',
'調査員が見た特徴', '住所', '営業時間', 'お店名ふりがな', '地図', '料金目安', '調査員おすすめメニュー']
```

出力できたCSVファイル

mapdata.csv

1 住所,営業時間,使用麺,ホームページ,駐車場,焼き方,調査員おすすめメニュー,TEL,エリア,
お店名ふりがな,受入人数,specurl,地図,店名,業種,定休日,調査員が見た特徴,料金目
安
2 富士宮市野中東町112-1,10:00~21:00,,,4,お店,キムチとチーズ入り,0544-27-0004,まち
なか,あき,12,<https://umya-yakisoba.com/shop/3776/>,B6,お好焼 あき,飲食店,火曜日,
キャベツとネギが多めに入っている,350~600円
3 富士宮市元城町13-20,7:00~19:00,,,なし,お店,塩焼そば,0544-22-3029,まちなか,あさ
や,10,<https://umya-yakisoba.com/shop/3777/>,C4,あさ家,飲食店,なし,具たくさん太
盛、朝定食あり,400円~
4 富士宮市淀師523-4,11:00~21:00(休憩14:00~17:00),,,10,,,0544-27-6005,郊外
・北部,あるばとろす,30,<https://umya-yakisoba.com/shop/3774/>,あるばとろす,飲食
店,月曜日(第4週日・月連休),,500円~
5 富士宮市宮町2-9,平日11:00~16:00土日祝11:0017:00,曾我麺,,なし,お店,焼きそば、ピリ
辛焼きそば,0544-26-3849,まちなか,いっぷくてい,20,<https://umya-yakisoba.com/shop/3616/>,B4,いっぷく亭,鉄板焼・軽食,月曜・第3日曜,麺はもちもちでコシのある歯ごたえ
が好評で、野菜は細く切り麺とソースと絡んで食べやすく,550~750円
6 富士宮市淀師468-2,10:30~17:00※材料がなくなり次第終了,,<https://tabelog.com/shizuoka/A2204/A220401/22002448/>,25,お店,五目焼きそば,0544-27-6494,まちなか,いと
う,50(3部屋あり),<https://umya-yakisoba.com/shop/3772/>,A1,好み食堂伊東,飲食
店,月曜・火曜・大きな鉄板でアツアツをどうぞ,550~850円

	2 店名	住所	営業時間	使用麺	ホ
3	お好焼 あき	富士宮市野中東町112-1	10:00~21:00		
4	あさ家	富士宮市元城町13-20	7:00~19:00		
5	あるばとろす	富士宮市淀師523-4	11:00~21:00 (休憩14:00~17:00)		
6	いっぷく亭	富士宮市宮町2-9	平日11:00~16:00土日祝11:0017:00	曾我麺	
7	好み食堂伊東	富士宮市淀師468-2	10:30~17:00※材料がなくなり次第終了		h
8	焼きそば好みうめづ	富士宮市外神1273-5	11:00~19:30		
9	うるおいてい	富士宮市淀師415-2	火~金11:30~14:00 16:30~18:30土日祝 11:00~18:00		h
10	大阪屋	富士宮市元城町12-10	10:00~20:00		h
11	海友	富士宮市西町28-9	18:00~22:30		
12	キッチントロ	富士宮市中央町7-18	11:00~17:00		
13					
14					
15					
16					
17					
18					

データを読む/取り込む

3. データを使う/活用する

旅行中に使うためのツールとして

- 巨人の肩に乗る:

 Googleマイマップで使おう

- [付録]  ポータブルに扱う: 印刷をする
- [付録]  専用のWEBアプリを作ろう

Googleマイマップとは

- オリジナルマップを作成できる
- スマホ版Googleマップでも表示可能

データを加工/出力する

データを使う/活用する



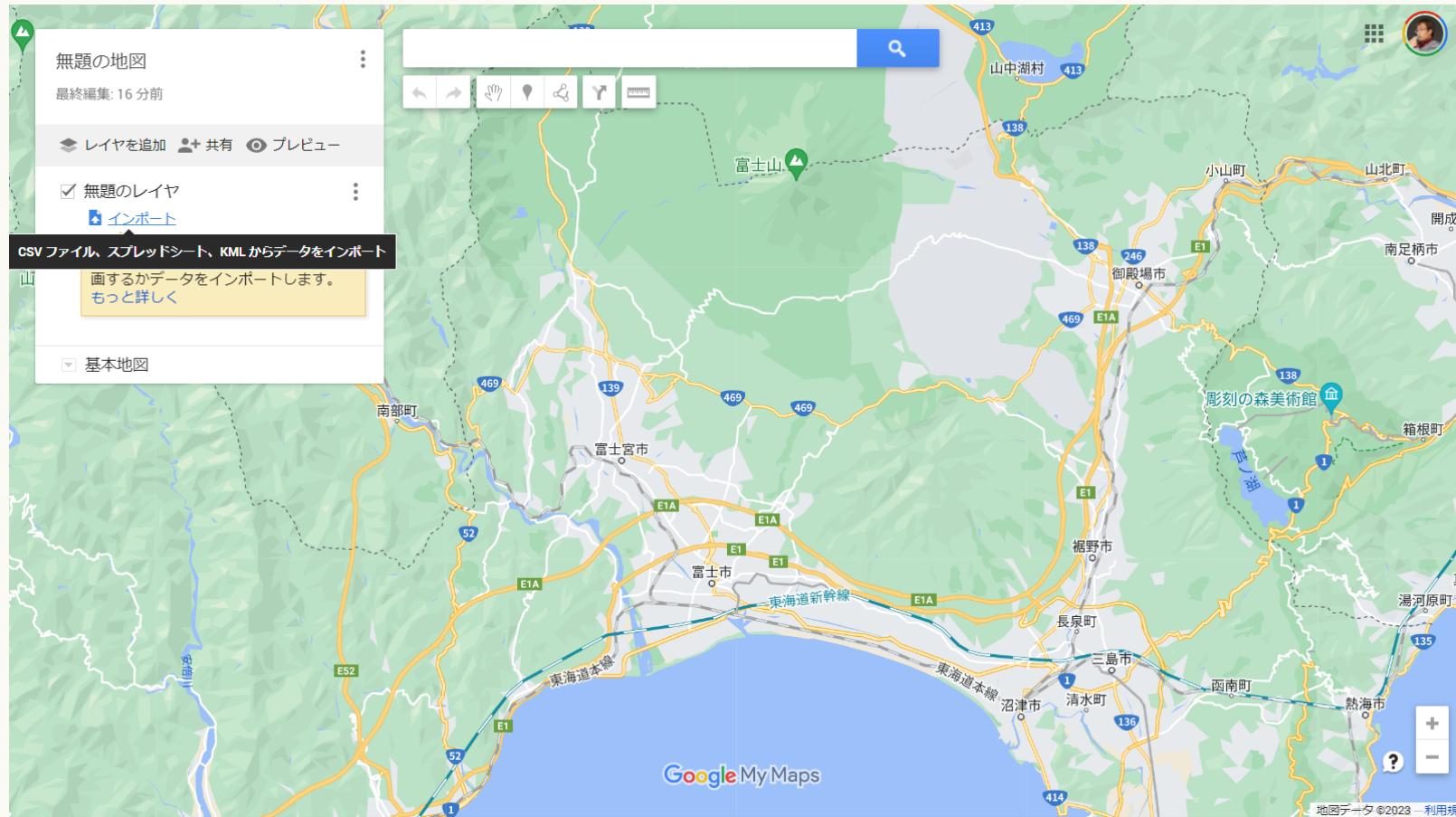
扱い方

<https://www.google.com/maps/d/> ヘアクセスして利用します。

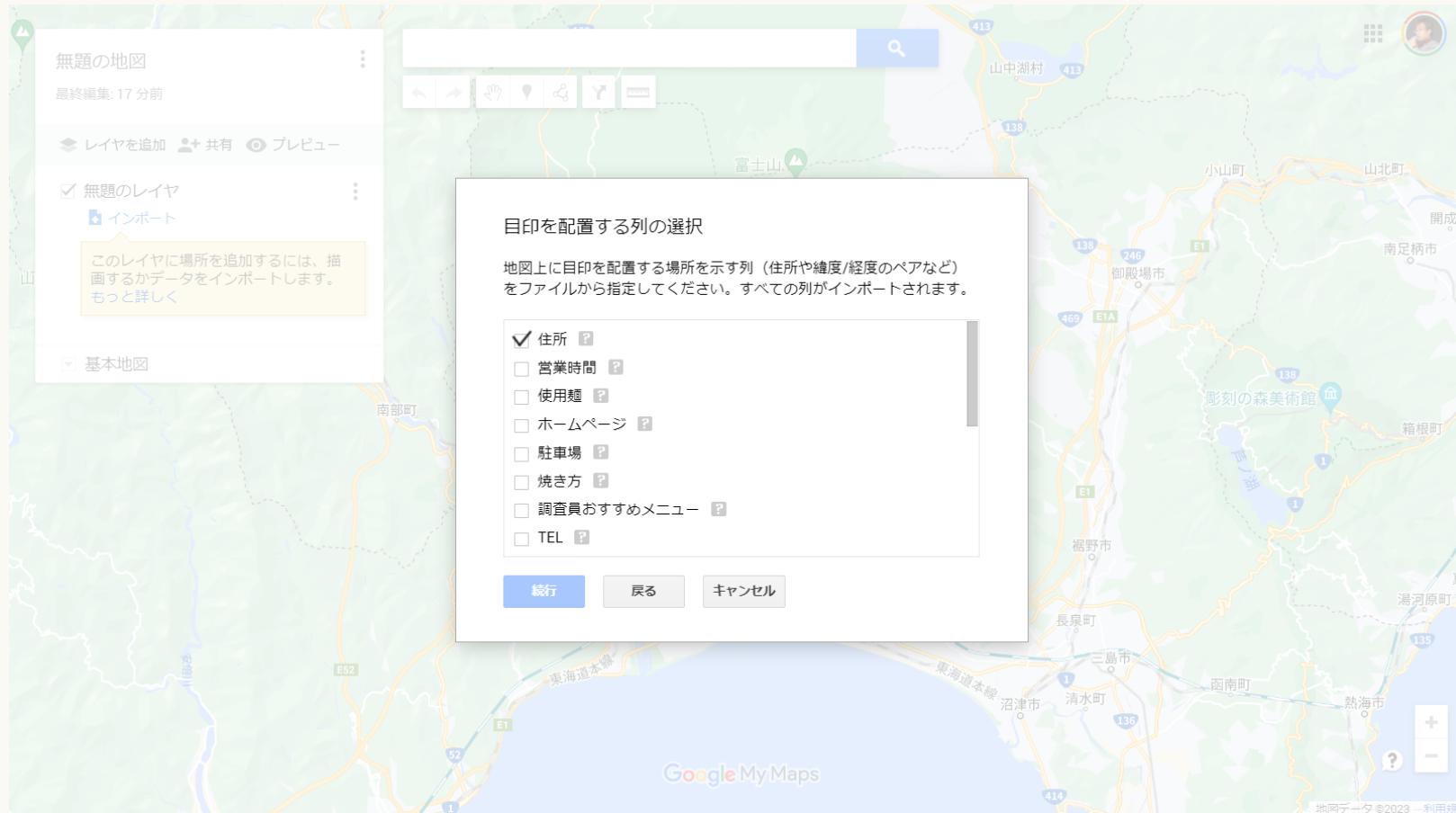
「新しい地図を作成」ボタンを押す



新規レイヤーへCSVファイルをインポートします

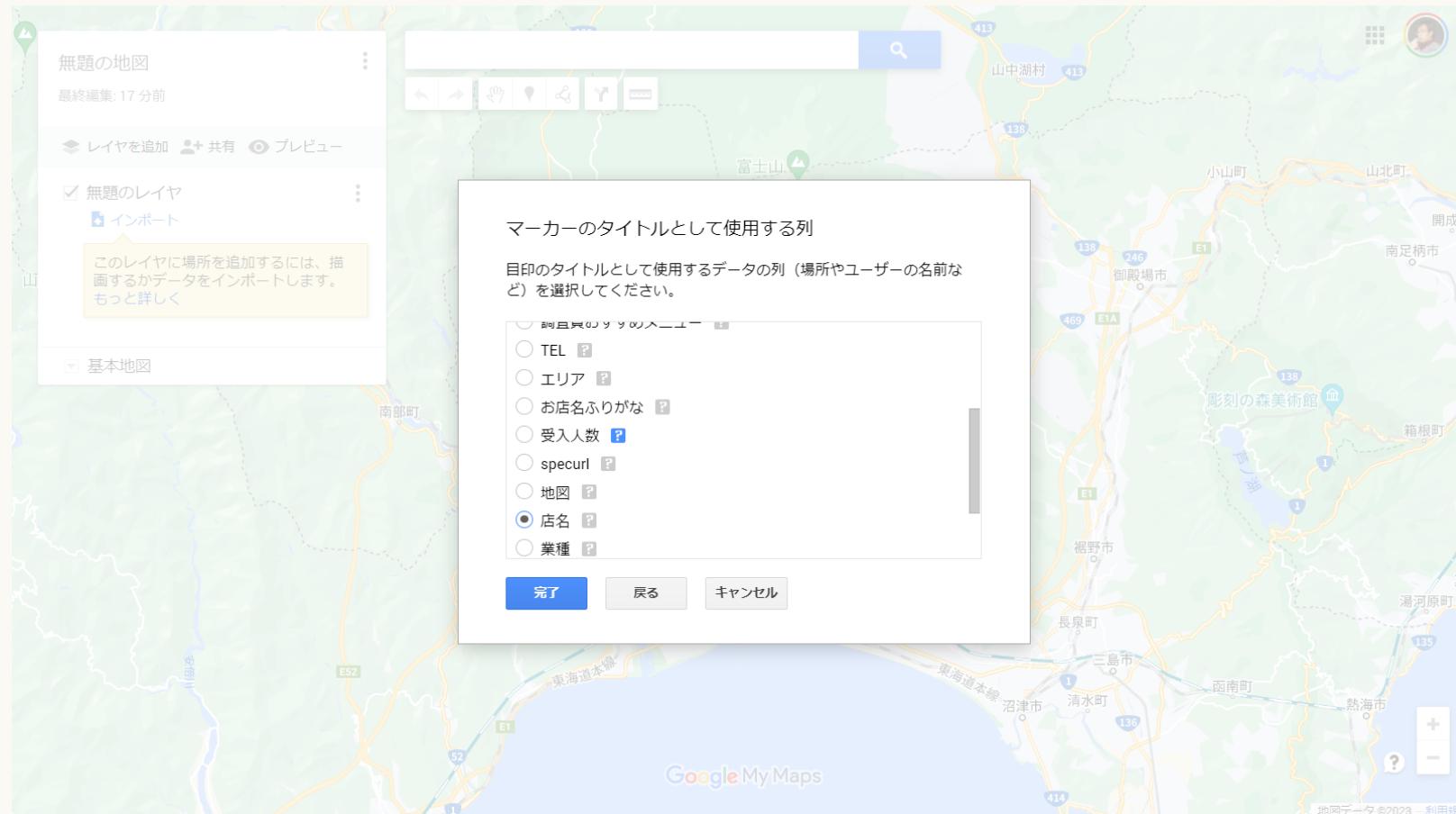


マップへのポイントは住所を使います

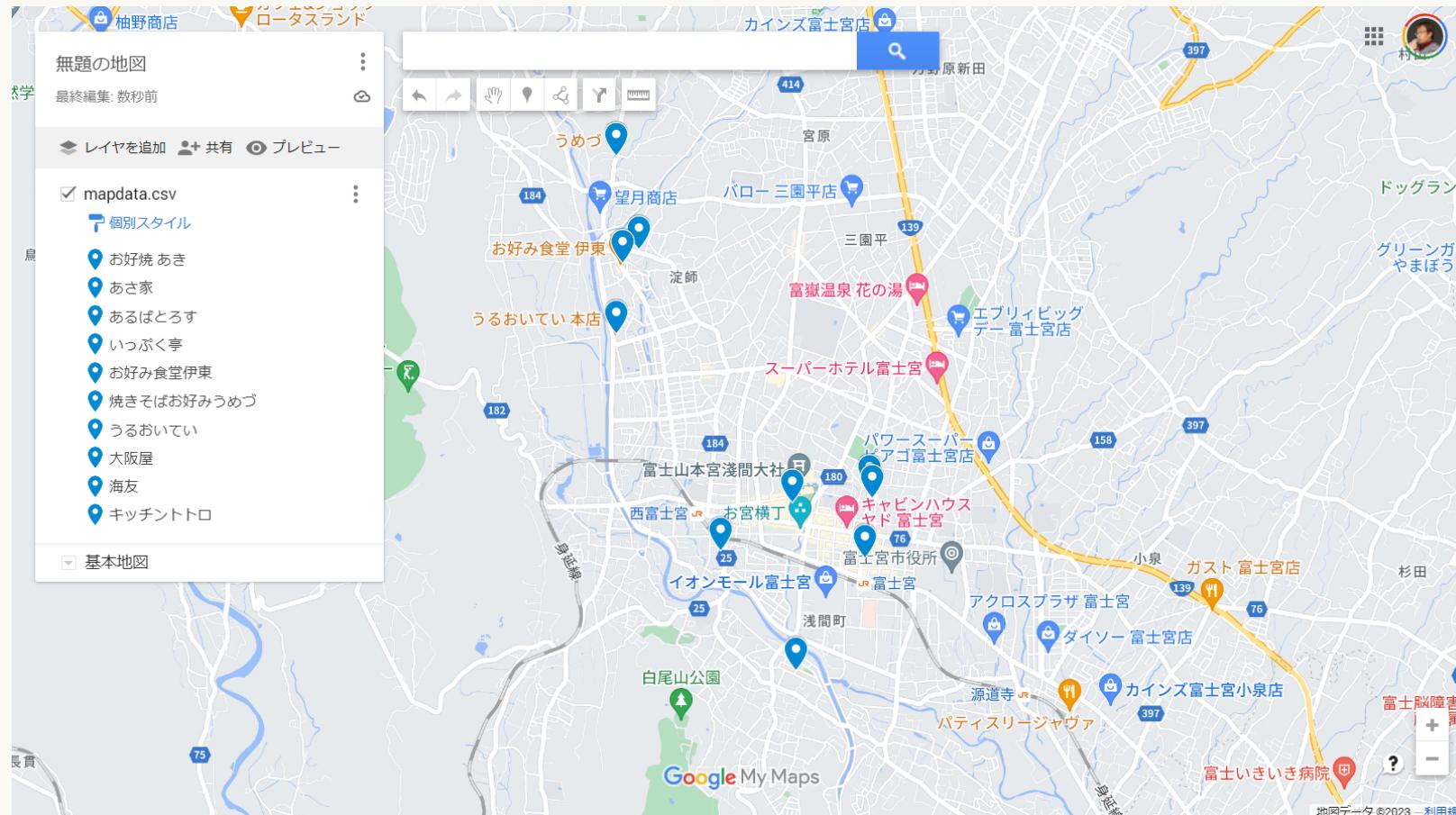


ポイントした部分への簡易説明（マーカー）を入れる。

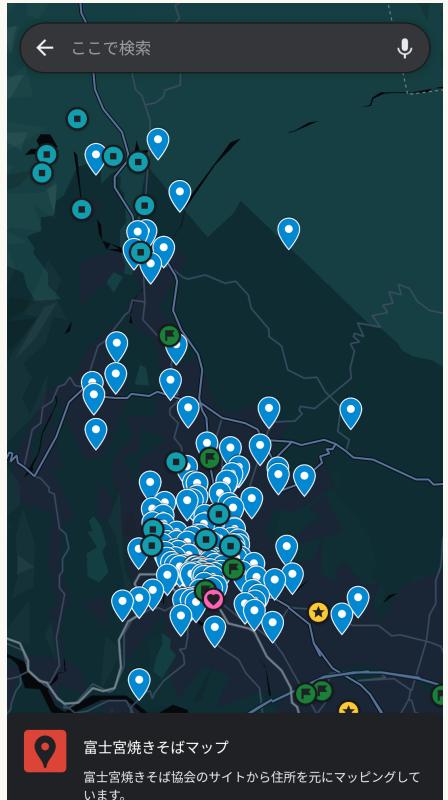
今回は店名にしました。



マッピングされました!



スマホで見ることもできます！



そのほか選択肢

今回はGoogleマイマップを利用しましたが、推奨しているわけではなくてデータを作るといろんなサービスと連携できることをお伝えしました。

※: オリジナルのマップを作るサービスは他にも多数あります。一例を載せます

※: それぞれ特徴や無料有料があるので、使いやすいものを探すと良いと思います

- OpenStreetMap uMap: <https://umap.openstreetmap.fr/ja/>
- proxi: <https://www.proxi.co/>
- [日本向け]国土地理院: <https://maps.gsi.go.jp/>
- etc...

トークのまとめ

データを読む/取り込む



データを加工/出力する



データを使う/活用する

今回のトーキでできること

1. お店情報をデータにする: WEBスクレイピング
2. データを整形をする: CSVファイルにする
3. データを使ってみる: Googleマイマップ



1. データを読む/取り込む
2. データを加工/出力する
3. データを使う/活用する

プログラミングでもっとも行われる行為

- **どこから** データを取り出すか
- **どんな** データを作るか
- **どこへ** データを渡すか

- どこから データを取り出すか
 - データベースから取り出す
 - 現実の統計データを使う
 - センサーデータで現実環境を扱う
- どんな データを作るか
 - データを加工して新しいデータを作る
 - データを整形する
- どこへ データを渡すか
 - データの可視化をして分析
 - データを使って業務工程改善
 - データを使ってAIで予測

- どこから データを取り出すか
 - データベースから取り出す → データベース接続、API
 - 現実の統計データを使う → データセット、オープンデータ
 - センサーデータで現実環境を扱う → ハードウェア操作
- どんな データを作るか
 - データを加工して新しいデータを作る → 前処理、後処理
 - データを整形する → ファイル形式、データ構造
- どこへ データを渡すか
 - データの可視化をして分析 → グラフ、ダッシュボード
 - データを使って業務工程改善 → オートメーション、システム連携
 - データを使ってAIで予測 → 機械学習、ディープラーニング

最後に伝えたいこと

PyCampやPythonの基礎を学んだ方の一歩先として！

自分が使いたい、利用したい、ものやことでトライしましょう

それができれば、とても楽しいし感動します。オススメです 😊

Happy Hacking!!

and, Have a nice trip!!

付録

今回扱わなかった他の方法については、またどこかで解説できたらと思います

- WEBスクレイピングしづらいサイトもカバーする
 - selenium + headless chrome
- 画像識別でお店情報を収集する
 - OCR, Googleなど
- 緯度経度を収集
 - 世界: Google?
 - 日本: 東大CSIS
- 印刷物を作る -> テンプレートエンジンで印刷しやすいHTMLを生成
- fletで自分専用のマップアプリを作る

WEBスクレイピング: Seleniumの例

- 動的な (javascriptなど利用した) サイトはrequestsで対応が難しいことがあります
- 本物のブラウザ + ブラウザ自動操作ツール(selenium)を使った例も載せておきます

-> url

画像識別

結構難しい分野（ちょっと自信ない）

- Google Cloud Vision
 - -> url

位置情報

- 世界: Google Maps Platform ->
<https://developers.google.com/maps/documentation/geocoding/overview?hl=ja>
 - -> url
- 日本: 東大CSIS -> jageocoder
<https://github.com/t-sagara/jageocoder>
 - 無料で利用できる（リクエスト数は少なめがよし
 - -> url

印刷用マップを作つてみる

印刷用のHTMLファイルを作つて印刷してみる

- Mapboxで概要と詳細の地図を用意
- お店の情報をテーブルで埋め込む
- 印刷用にCSSで調整する

-> url

モバイル向けのWEBアプリ

実験的な扱いです。うまく動かなければ...

- fletを使って作ってみました
- staticなページでWASMとして埋め込んで作ってみる
 - <https://flet.dev/docs/guides/python/publishing-static-website/>
- ※サイト自体の公開はしません

-> url