Python&Flaskで学ぶ

Web API 入門

全人類がわかる統計学 Presents

https://to-kei.net

今日の配布資料

- 本スライド
  - WebAPI.pdf
- 講義中に使用するプログラム(jupyter botebookのコード)
  - hello.ipynb
  - get.ipynb
  - post.ipynb
  - error.ipynb
- 講義中の練習問題
  - practice.ipynb
- 講義最後の演習問題
  - ensyu_answer.ipynb
  - ensyu_blank.ipynb
  - exercise_book.csv

本講座の目的・ゴール

- 開発したいサービスやデータ解析アプリケーションを外部に公開するにあたり、重要な項目を理解する
- すでにあるアプリケーションを外部サービスと連携する際に、Flaskを使ってどのようにAPIサーバーを実装すれば良いかの具体的なイメージを掴む

本講座では取り扱わない事柄

- Pythonの基礎文法について
- Webアプリケーションの構築方法
- Web APIをデプロイする際の詳細な手順

目次

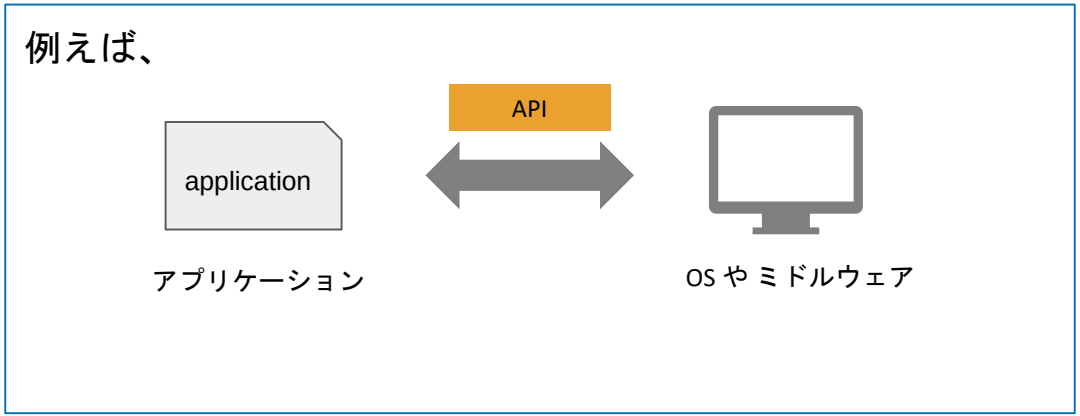
1. Web APIとは？
2. HTTPリクエストの概念と種類
3. Flaskでサーバーを立ててみよう
4. GET、POSTの値を受け取る方法
5. エラーハンドリングについて
6. 演習問題

まず初めに、

Web APIとは?

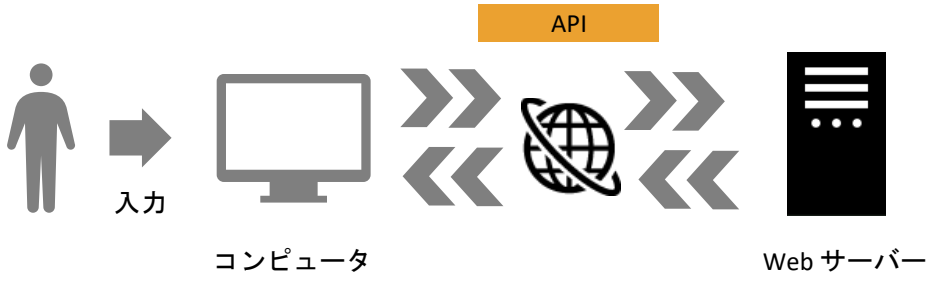
APIとは？

- Application Programming Interfaceの略
- ある機能に特化したプログラムで共有可能なもの
- 二つの物事を仲介する役割



Web APIとは?

- Web上で使用するAPI
- Web上に公開されていて、外部から呼び出し可能
- HTMLの代わりにプログラムで処理しやすいデータ形式を返すWebページの一種
  ex) JSON, XML



Web APIと一般的なWebサイト

	人間がみたときの見やすさ	プログラムでの扱いやすさ
Web API (例)	×	◎
一般的なWebサイト (例)	○	△

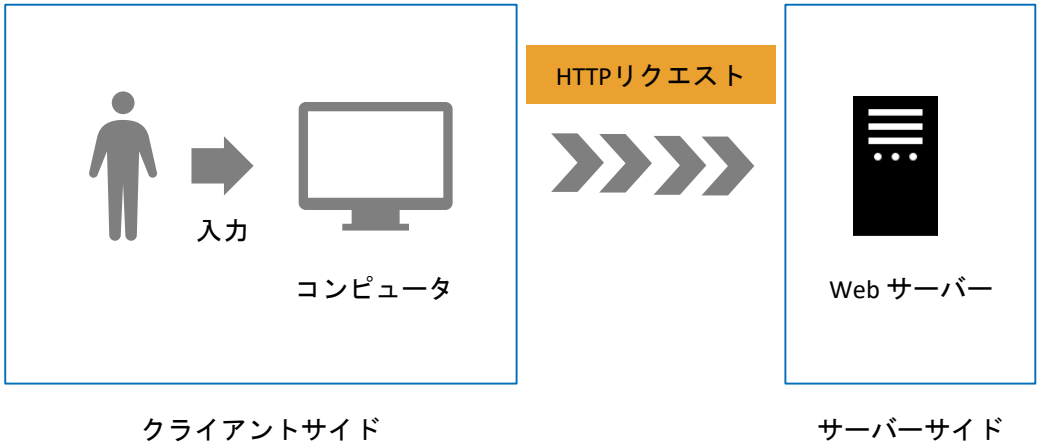
- Web APIでやり取りされる形式はデータが構造化されているため、プログラムで非常に扱いやすい
- 一般的なWebサイトからも情報を抽出することは可能だが、json形式には扱い易さは劣る

実装する前に

HTTPリクエストの概念と種類

HTTPリクエストとは

HTTPを使用し、クライアント側のPCからwebサーバーに情報を要求すること。



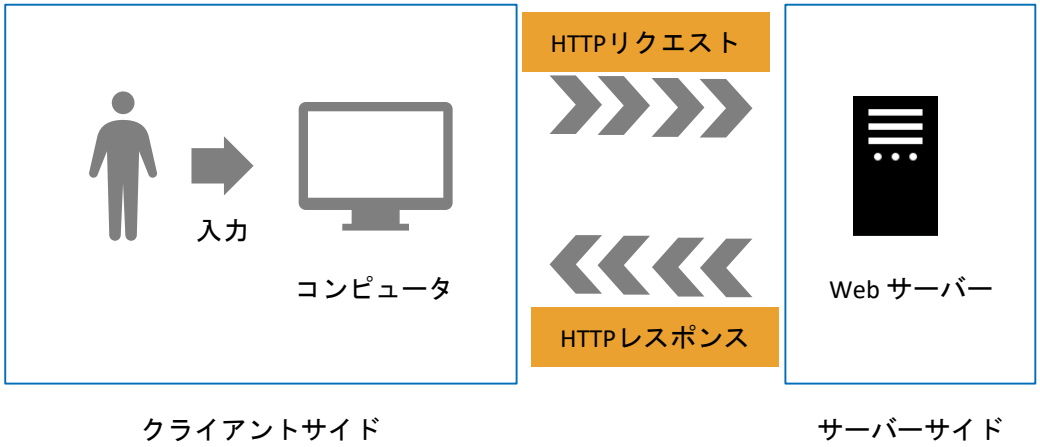
HTTPリクエストとレスポンス

HTTPリクエスト

→ クライアント側のPCからwebサーバーに情報を要求すること。

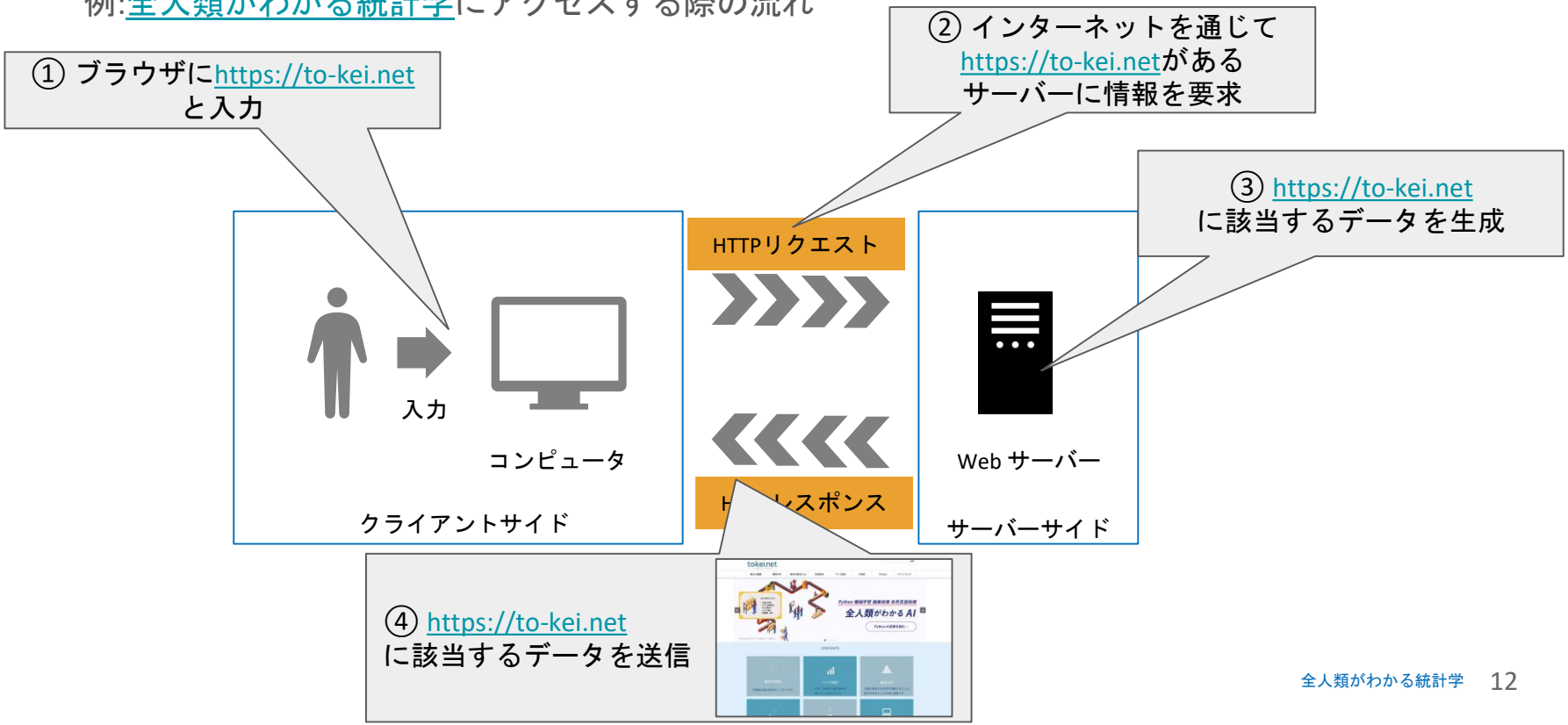
HTTPレスポンス

→ webサーバークライアント側のPCからwebサーバーに情報を渡すこと。



Webサイトにアクセスする際の流れ

例:全人類がわかる統計学にアクセスする際の流れ



Web APIの実用例

- google map
- Twitter
- LinkedIn
- facebook
- Instagram

google map api

- 世界中の地図データ
- 世界中の場所の位置情報
- リアルタイムの交通状況データ

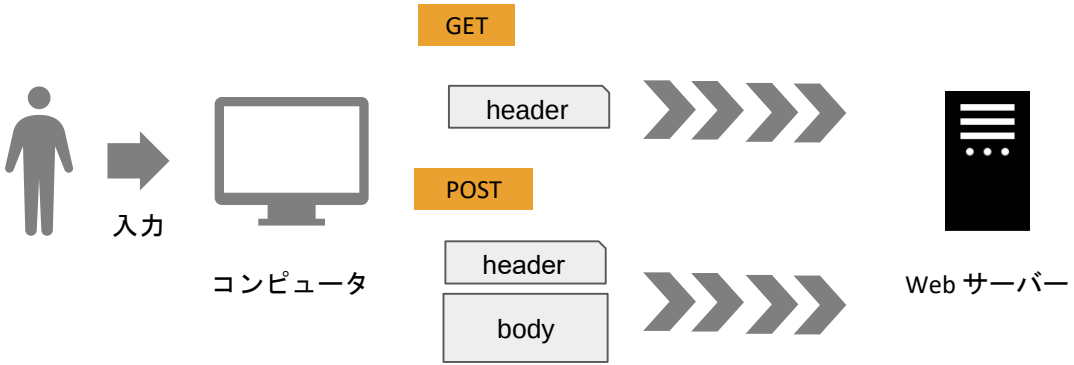
twitter api

- タイムラインのデータ
- ツイートの投稿
- リツイートやいいねの実行
- ツイートの検索

HTTPリクエストの種類

GET : URIにパラメータを付加して、webサーバーにリクエスト。
通常、Webサイトを閲覧するときの通信メソッド

POST : bodyにパラメータを付加して、webサーバーにリクエスト。
GETよりもより多くの情報を持ってリクエスト可能なメソッド



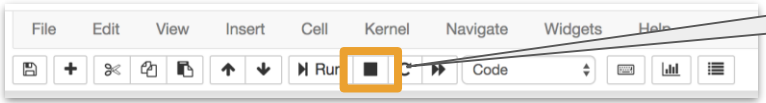
※ PUT,DELETE,OPTIONSなどのリクエスト方法もあるが、本講座では頻繁に利用するGETとPOSTのみを扱う

いざ、実装！！

Flaskでサーバーを立ててみよう

Flaskでサーバーを立てる

hello.ipynb



セルを選択し、■ を押すと
起動中のサーバーは停止

[*]になっているときはサーバー起動中

```
In [*]:
# Flaskをインポート
from flask import Flask, Response

# Flaskのインスタンスを作成。
# Flaskで使用できる機能を使用できるようにする。
app = Flask(__name__)

# ファンクションを起動するURIをFlaskに指定
@app.route("/hello")

# URIが指定された際に実行されるファンクションを指定
def hello():

    # Responseで引数をtext形式で返す。(mimetypeを指定しないとhtml形式で返される)
    return Response("Hello World!", mimetype="text/plain")

#   ローカルサーバーでアプリケーションを実行
app.run()

 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

レスポンス

クリック

Flaskでサーバーを立てる

hello.ipynb

```
# Flaskをインポート
from flask import Flask, Response

# Flaskのインスタンスを作成。
# Flaskで使用できる機能を使用できるようにする。
app = Flask(__name__)

# ファンクションを起動するURIをFlaskに指定
@app.route("/hello")

# URIが指定された際に実行されるファンクションを指定
def hello():

    # Responseで引数をtext形式で返す。(mimetypeを指定しないとhtml形式で返される)
    return Response("Hello World!", mimetype="text/plain")

# ローカルサーバーでアプリケーションを実行
app.run()
```

練習問題1

今日の日付を受け取ってみよう。

ヒント：
今日の日付の取得の仕方

```
import datetime
today = datetime.date.today()
```

解答ファイル：practice.ipynb

リクエストに情報を埋め込んでみよう

GETで値を受け取る方法

URIパラメータを使って値を受け取る

どのようにしてURIから引数を送信するのか?

[ファンクションを指定したURI]?変数=引数

get.ipynb

```
from flask import Flask, Response, request

app = Flask(__name__)

#リクエスト方法をGETで指定する
@app.route('/getter', methods=['GET'])
def getter():

    # request.args.getで指定した引数を取得
    name = request.args.get("name")
    return Response(name, mimetype="text/plain")

app.run()
```

```
* Serving Flask app "__main__" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

http://127.0.0.1:5000/getter?name=avilen

URIパラメータを使って値を受け取る

get.ipynb

```
from flask import Flask, Response, request

app = Flask(__name__)

#リクエスト方法をGETで指定する
@app.route('/getter', methods=['GET'])
def getter():

    # request.args.getで指定した引数を取得
    name = request.args.get("name")
    return Response(name, mimetype="text/plain")

app.run()
```

複数の値を受け取る

どのようにしてURIから複数の引数を送信するのか?

[ファンクションを指定したURI]？変数1＝引数1＆変数2＝引数2

get.ipynb

```
from flask import Flask, Response, request

app = Flask(__name__)

@app.route('/getters', methods=['GET'])
def getters():

    # request.args.getで指定した引数を複数取得
    name = request.args.get("name")
    data = request.args.get("data")

    # "文字列"+"文字列"で文字列の連結
    return Response(name + "_" + data, mimetype="text/plain")

app.run()
```

```
 * Serving Flask app "__main__" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

http://127.0.0.1:5000/getter
?name=avilen&data=webapi

複数の値を受け取る

get.ipynb

```
from flask import Flask, Response, request

app = Flask(__name__)

@app.route('/getters', methods=['GET'])
def getters():

    # request.args.getで指定した引数を複数取得
    name = request.args.get("name")
    data = request.args.get("data")

    # "文字列"+"文字列"で文字列の連結
    return Response(name + "_" + data, mimetype="text/plain")

app.run()
```

JSONでレスポンス

get.ipynb

```
from flask import Flask, jsonify

app = Flask(__name__)

@app.route("/response-json",methods=["GET"])
def response_json():

    # dicに辞書型でデータを保存
    dic= {
          "sports":{
             "baseball":9,
             "soccer":11
             }
          }

    # JSON形式でレスポンス
    return jsonify(dic)

app.run()
```

練習問題2

GETリクエストを用いて姓と名を受け取り、それぞれを下記のようなjson型に変換して
レスポンスするプログラムを作成せよ

リクエストURI例

http://127.0.0.1:5000/practice2?last=taro&first=sato

レスポンス例

```
{
    "user_name":
        {
            "first_name":"sato",
            "last_name":"taro"
        }
}
```

解答ファイル：practice.ipynb

URI上に引数を埋め込んで値を送信する

get.ipynb

```
from flask import Flask, jsonify, Response

app = Flask(__name__)

# URI上に引数を埋め込み、値を送信
@app.route("/<apinum>/apiget",methods=["GET"])
def apiget(apinum):
    return Response("you get " + apinum, mimetype="text/plain")

app.run()
```

少し複雑なデータを受け取ってみよう

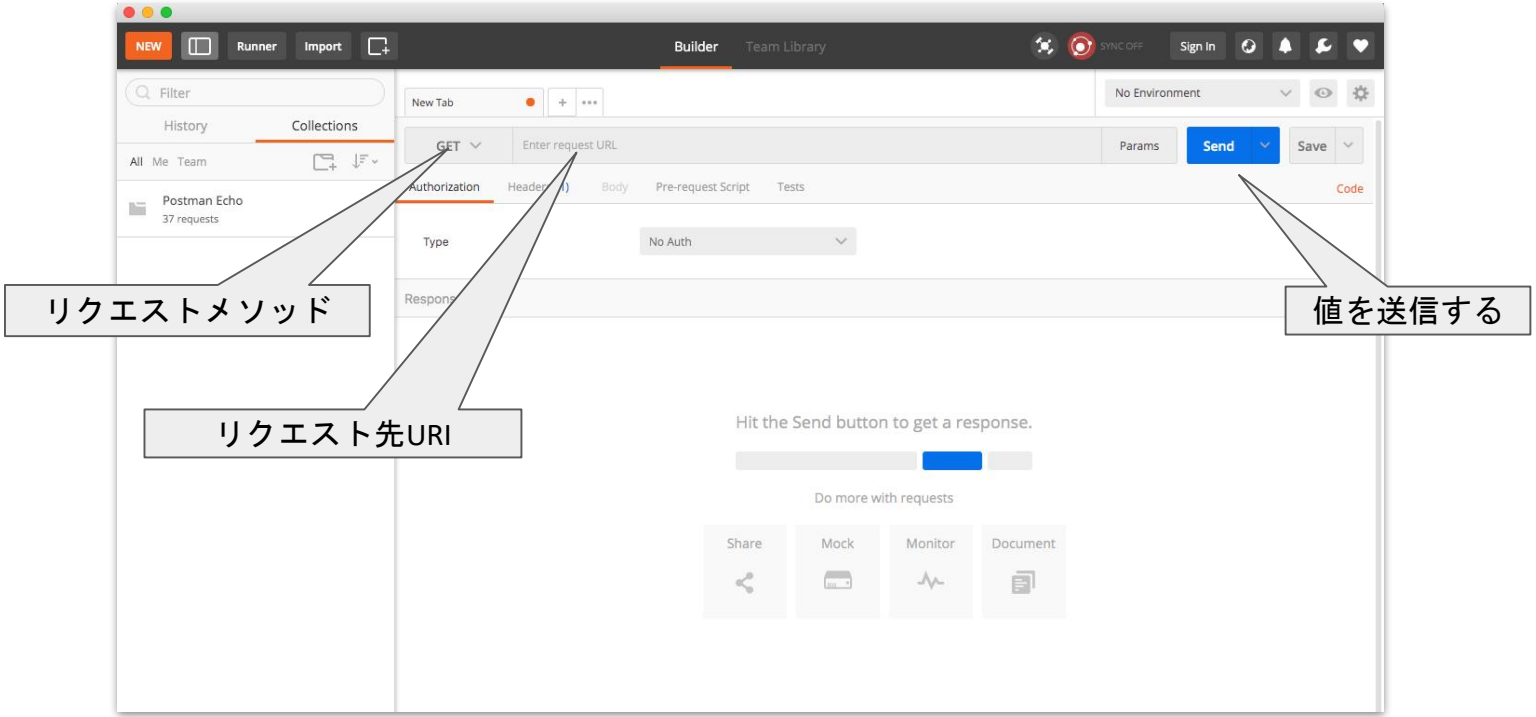
POSTで値を受け取る方法

POSTで値を送信する

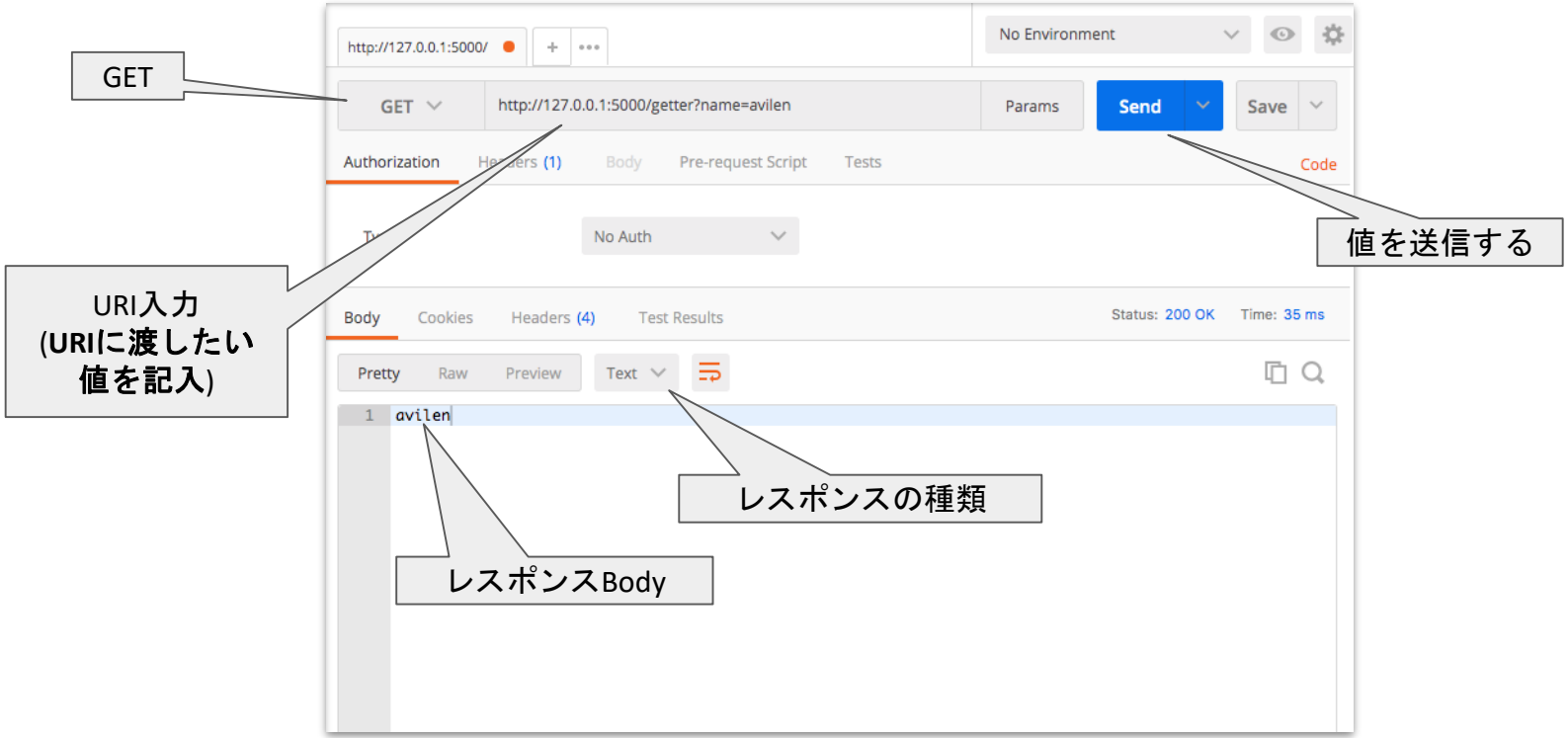
Postmanを使用してPOSTで値を送信してみよう(GETも使用可能)

https://www.getpostman.com/downloads/

Postmanの使い方



Postman : GETでリクエスト

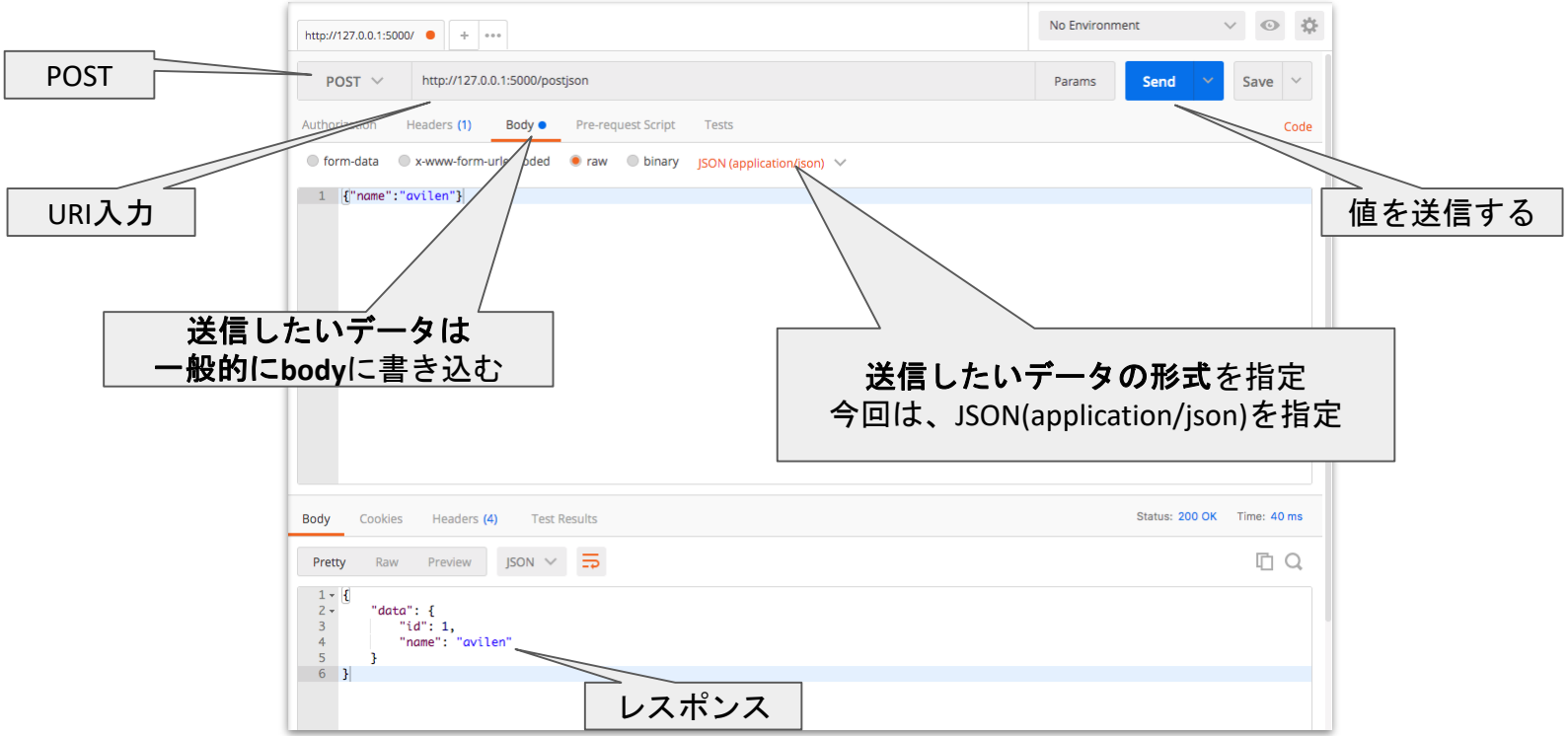


POSTでJSON形式のデータを受け取る

post.ipynb

```
from flask import Flask, jsonify, request

app = Flask(__name__)

# リクエスト方法をPOSTで指定
@app.route("/postjson",methods=["POST"])
def postjson():

    # request.args_json()で送信したJSON形式のデータを取得
    json = request.get_json()

    #  JSONデータのname部分を取得
    myname = json["name"]

    # resultのnameにmynameを使用
    result = {
            "data": {
                "id" :1,
                "name":myname
                }
            }
    return jsonify(result)

app.run()
```

Postman : POSTで値を受け取る



練習問題3

POSTで{“id”:1,”height”:170,”weight”:50}を送信し、JSON形式でidとBMI数値をレスポンスせよ

ヒント
BMIの計算の仕方

```
BMI = 体重[kg] / 身長[m]**2
```

解答ファイル：practice.ipynb

実践

エラーハンドリングについて

エラーハンドリングはなぜ必要か

- サーバーは人がずっと監視しているわけにはいかない
- レスポンスに**プログラムのエラーをそのまま記述するとセキュリティリスク**になりうる
- エラーのログをファイルまたはDBに保存しておく必要がある
- エラーログは**どのようなケースでエラーが起きたかを把握**できるようにするもので、アプリによって設計方法は異なる

  → 特定のデバイスからのアクセスによるエラーがあるかも、、
  → ユーザーIDを記録してエラーを出したユーザーにサポートが行えるようにしたい、、

  様々なケースが想定しうる、、、

- 当講座では簡単なエラーハンドリング手法の１つを紹介

エラーコードとは?

エラーが起きた際に返されるステータスコードのこと。

- 400番台がクライアントのリクエストに起因するエラー
- 500番台がサーバー側に問題があった場合のエラー

400番台のステータスコード
- 400 : リクエストが正しくない
- 403 : アクセスが禁止されている
- 404 : 指定したリソースが見つからない
- 405 : 指定されたメソッドを使うことができない

500番台のステータスコード
- 500 : サーバー側でエラーが発生した
- 503 : サーバーが一時的に停止している

エラーの内容をレスポンスする

error.ipynb

```
from flask import Flask, Response, request

app = Flask(__name__)

@app.route("/try",methods=["POST"])
def try_example():
    json = request.get_json()
    # try以下のコードが通らなかった時にexcept以下が実行
    try:
        # 入力が文字列だとint型に変換できないためエラーが発生
        num = int(json["num"])
        return Response(str(num), mimetype = "text/plain")

    except:
        # ステータスコードは２つ目の返り値で指定
        return Response("有効な数字ではありません。",mimetype = "text/plain"), 500

app.run()
```

エラーログを残す 1

error.ipynb

```
from flask import Flask, Response, request
import datetime #現在時刻を取得するのに利用
import sys #エラーハンドリングの際に利用

app = Flask(__name__)

@app.route("/errorhand",methods=["POST"])
def errorhand():
    try:
        json = request.get_json()
        num = int(json["num"])
        return Response(str(num),mimetype = "text/plain")
    except Exception as e:
        error_detail = errorlogging()#エラーロギングの関数にエラー内容を渡す

        #プログラムのエラーはそのままレスポンスしない
        return "INTERNAL SERVER ERROR", 500
```

エラーログを残す 2

error.ipynb

```
def errorlogging():
    fn = "flask_error.log"#ログの出力先を指定
    now = datetime.datetime.now()#現在時刻を取得
    now = now.strftime("%Y/%m/%d %H:%M:%S")#時刻のフォーマットを変更
    _, error_message, tb=sys.exc_info()#エラー情報を取得
    lineno = tb.tb_lineno#エラーが発生したプログラムの行数を取得
    error_detail = {
                        "timestamp":now,
                        "program":"error.ipynb",
                        "line":lineno,
                        "message": str(error_message)
                       }
    error_detail = str(error_detail)#一行でファイル出力するためにstr型に変換
    #エラーの詳細を書き出す
    with open(fn,"a") as f:
        f.write(error_detail+"\n")
    return error_detail
```

最後に、

GETとPOSTを使った演習問題

演習問題

1. genreと金額をGETで受け取り、そのgenre内のある金額より安い本の一覧をレスポンスせよ

1. JSON{“genre”:”python”,”title”:”python演習本”,”price”:5000,”date”:”2019/5/18”}を送信し、exercise_book.csvに受け取った値を追加挿入せよ。

exercise_book.csv

genre	title	price	date
js	Node.jsデザインパターン 第2版	4536	2019/5/18
design	インタフェースデザインの実践教室	3240	2013/4/17
python	Pythonデータサイエンスハンドブック	4536	2018/5/26
python	PythonによるAIプログラミング入門	3672	2019/3/20
python	入門 自然言語処理	4104	2010/11/10
python	PythonとJavaScriptではじめるデータビジュアライゼーション	4104	2017/8/25
python	プログラミングROS	4104	2017/12/13
js	JavaScriptパターン	3024	2011/2/15
design	デザイニング・インターフェース 第2版	4968	2011/12/24
js	シングルページWebアプリケーション	4104	2014/5/24
python	Pythonではじめるデータラングリング	3996	2017/4/26

回答ファイル：ensyu_answer.ipynb

演習問題１のヒント

ensyu_blank.ipynb

```
import pandas as pd
from flask import Flask, jsonify, Response, request

app = Flask(__name__)
app.config['JSON_AS_ASCII'] = False


def readcsv():
    df = pd.read_csv("exercise_book.csv")
    df["price"] = df["price"].astype(int)
    return df

@app.route("??????",methods=["GET"])
def summry1(genre):
    threshold = ?????
    buf = df[df["genre"] == genre]
    result = buf[buf["price"]<threshold]
    titles = [i for i in result["title"]]
    res = {"titles":titles}
    return jsonify(res)

df = readcsv()#アプリの初回起動時にcsvファイルを読み込む
app.run()
```

リクエストURI例

http://127.0.0.1:5000/python/book?threshold=3000

レスポンス例

```
{
    "titles": [
        "作って動かすALife",
        "Think Stats 第2版",
        "アジャイルデータサイエンス",
        "Think Bayes",
        "Pythonチュートリアル 第3版",
        "Python & AWS クックブック"
    ]
}
```

演習問題2のヒント

リクエストURI例

http://127.0.0.1:5000/python/book?threshold=3000

リクエスト body

```
{
        "genre":"python",
        "title":"python演習本",
        "price":5000,
        "date":"2019/5/18"
}
```

ensyu_blank.ipynb

```
import json
import pandas as pd
from flask import Flask, jsonify, Response, request

app = Flask(__name__)

# 総合問題(post)
#{"genre":"python","title":"python演習本","price":5000,"date":"2019/5/18"}
@app.route("/exercise2",methods=["POST"])
def exercise2():
    df = pd.read_csv("exercise_book.csv")
    input_data = ?????
    genre = ?????
    title = ?????
    price = ?????
    date = ?????
    s = pd.Series([genre,title,price,date],index=df.columns,name="append")
    df = df.append(s)
    df.to_csv("./result.csv",index=False)
    return Response("ok",mimetype="text/plain")

app.run()
```

最後に

講義後に確認して欲しいこと

実際に業務で利用するにあたって

- GETによる送信は、URIにパラメータを付加して送る
- POSTによる送信は、bodyにパラメータを付加して送る
- エラーログは、エラーデバッグに必要な情報をユースケースに合わせて出力
- (演習問題のような) **データの取り出しや追加はDBと連携**して行うのが一般的
- Flaskだけでサーバーを立てるのは開発時のみ
- デプロイ/リリース時は**ApacheやNginxなどと連携**してFlaskを動かすのが一般的

  → Flaskでたてたサーバーは多くのアクセスを捌けない

次回の受講に向けて

学習フローチャートを参考に最適な勉強方針を立てましょう。

学習フローチャートを拡大して見たい方はこちら

体系的長期コースについて

日本ディープラーニング協会の一員として、公式認定を受けた長期コースも開講しています。

ハンズオンorオンライン講座

【業界最安・合格率 88.6%】

E資格対応 全人類がわかる
ディープラーニングコース
30時間 全６回

オンライン講座

【不合格者全額返金！！】

G検定対策
AIジェネラリスト育成コース
8時間 + 演習問題320問

その他体系化されたコースの詳細はこちらから

日本ディープラーニング協会(JDLA)について知りたい方はこちらから

お仕事に関するご相談

本日は最後までご静聴ありがとうございました。

法人向けAIスペシャリスト育成研修について

成果に直結する２つの研修プランを提供しています。

AI人材育成eラーニング

AIエンジニアの基礎を学ぶ体系化されたeラーニング

当プランに含まれるもの
・70時間分の講義動画
・60時間分の演習問題
・2ヶ月間のメンターサポート

特設ページはこちら

AIプロジェクトOJT研修

研修に加えて、専門家が貴社のビジネスを推進します。

当プランに含まれるもの
・事前にプロが無料で案件定義
・分析方針のフォローや進捗管理
・遠隔メンターサポート

特設ページはこちら

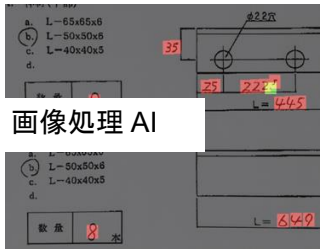
法人向けAIテクノロジー開発支援について

貴社の業務に特化した高精度モデルを提供するサービス「AVILEN AI」

AVILEN AI

高精度AIモデル提供サービス

AVILEN AIの詳細はこちら



画像処理AI導入事例：製造工場における加工書自動読み取りAIシステム
自然言語処理AI導入事例：国会の議事録文書の自動分類モデル
テーブルデータ分析AI導入事例：栄養士AIの開発

アンケートのお願い

今後の講座のクオリティ向上にご協力お願いします。

https://seminar.to-kei.net/qt/

スマホでアンケート回答はこちらから

研修のご依頼・事業の相談お待ちしております。

メールでのお問い合わせはこちら
contact@avilen.co.jp

Webサイトからのお問い合わせはこちら
https://avilen.co.jp/contact/

Thank you