クレジット：
UTokyo Online Education Education コンピュータシステム概論 2018 小林克志

コンピュータシステム概論
第12回

小林克志

- 事務連絡
- 先週の課題、レビュー（振り返り）
- まくら
- ミニプロジェクトに向けて
- 演習:Flask + matplotlib サンプルを動かす
- 課題: Flask に対する機能追加

課題: Flask に対する機能追加

- GitHub cloned: 14
- ~~GitHub pushed: 7~~

課題: Flask に対する機能追加

- 配布した Flask サンプルをもとに以下を実装する。

1.http://127.0.0.1:5000/static/hello.html でアクセスすると、文字列 "Hellow World" を返すページを作る。

2.配布したオリジナルでは a, b には整数のみを入力できる、これらを小数点以下 1 桁まで対応させる。

- オリジナルの HTML ファイルは template/index.html。
- HTML input 要素の step 属性を設定すればよい。

3.べき乗 a^b の計算に対応させる。

- JavaScript ファイルは static/calculator.js。HTML, JavaScript プログラム両者を矛盾なく変更する。
- a が負数の場合の処理も検討すること。

4.(選択課題)数値が変更されれば直ちに、すなわち Exec ボタンを押すことなく、結果が返される

(選択課題)数値が変更されれば直ちに、すなわち Exec ボタンを押すことなく、結果が返される

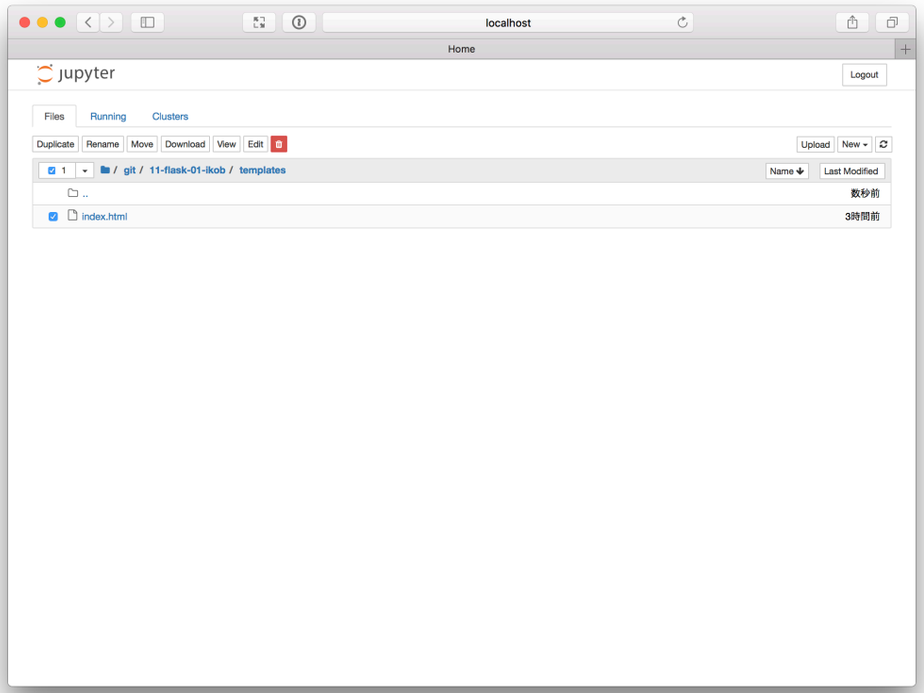
- 例えば calculator.js の 36-39 行目を参考に、数値変更に対応するイベントタイプをハンドラに登録する。オリジナルでは"Exec." ボタンの"click"タイプに対応させている。これを、数値入力フォーム、演算子セレクタの"change"タイプに対応させれば良い。
- イベントタイプ "keyup" を使えば、数値入力はさらに対話的になる。

static/calculator.js

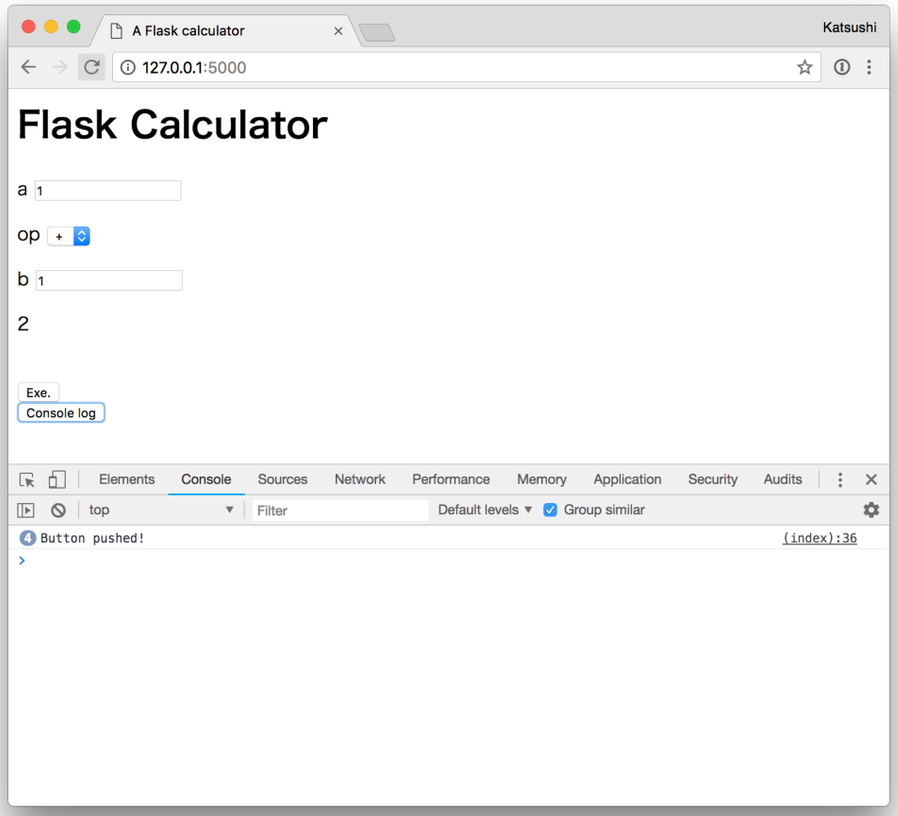
```
36 document.getElementById("exec").addEventListener("click", function(){
37     execCalculator();
38     changeButtonColor(document.getElementById("exec"));
39 }, false);
```

参考 : https://developer.mozilla.org/en-US/docs/Web/Events

Tips



- 編集は jupyter-notebook で可能
  - html ファイルの編集は、チェックボックスを on にし、メニューから Edit を選択
- デバグには Web ブラウザのデベロパーツールを使う。
  - Google Chrome の場合：
    表示 -> 開発/管理 -> デベロパーツール
- **Web コンテンツを修正した場合は再読み込みが必要（コンテンツはキャッシュされている）**
  - **Google Chrome の場合：**
    デベロパーツールを開いた状態で、リロードボタンを長押しする



著作権の都合により
ここに挿入されていた画像を削除しました

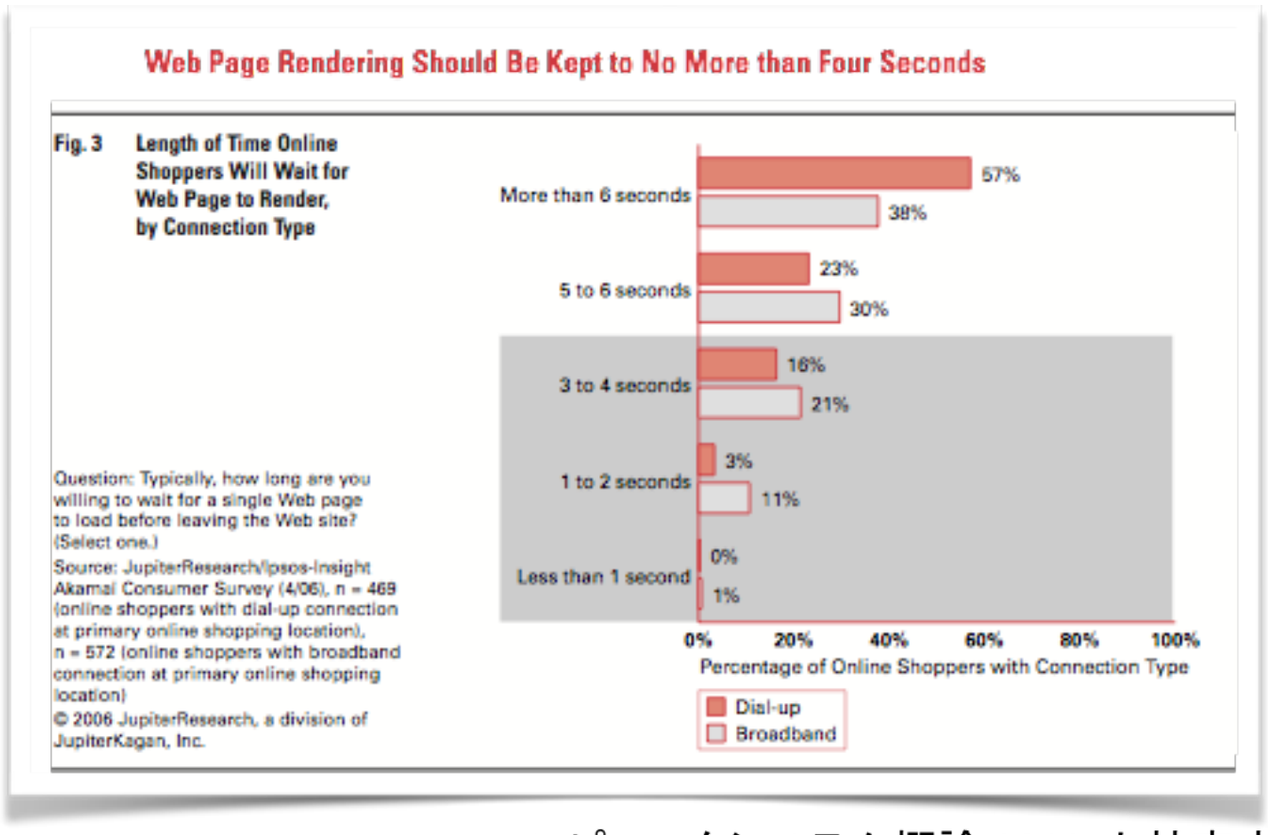
“New evidence supports ’five-second rule’ of dropped food”

New York Daily News
AFP RELAXNEWS, Mar 11, 2014, 11:07 AM
https://www.nydailynews.com/life-style/health/five-second-rule-article-1.1717615

Four/Two second rule in Web services

Application
Presentatiom
Session
Transport
Network
Datalink
Physical

- ＜4 sec. render completion limit to keep customers’ attention @ 2006
  - The limit is decreasing, e.g., 2 sec.@2013
- Poor satisfaction decreases customer loyalty and revisiting.
  - To miss opportunity, and to lost revenue on e-Commerce.



Jupiter Research, “RETAIL WEB SITE PERFORMANCE Consumer Reaction to a Poor Online Shopping Experience”, , p5, Fig.3, Jan 1, 2006

https://www.akamai.com/us/en/multimedia/documents/report/akamai-site-abandonment-final-report.pdf

ミニプロジェクト

- 目標：Web サービスをつくる
  - 前半で作った可視化をもとに、対話インターフェースつき Web サービスをつくる
  - 他のサービスでも良いが、対話インターフェースをもつこと
  - 下記の README.md なしに、内容が把握できること。
- 提出方法：
  - **締切 7/25（水）**
  - 関連ファイルを GitHub にアップロード
  - ~~最終日、5 分 / 人でプレゼン or~~
    GitHub にアップロードする。
  - README.md にA4 2 ページ程度で以下を記載する、
    プログラムの内容 / 工夫したところ / 苦労した点 /
    やりたりなかったこと/コメント
- 道具：
  - HTTP
  - HTML
    - DOM
  - JavaScript
  - Python
    - Jupyter Notebook
    - Matplotlib, Numpy, Pandas
    - Flask(Web フレームワーク)
- 評価基準：
  - アイディア・完成度
  - CSS などで実現できる「見栄え」は重視しない。
- 注意:
  - データの取得元を明記すること

演習:Flask + matplotlib サンプルを動かす
Git レポジトリのコピー

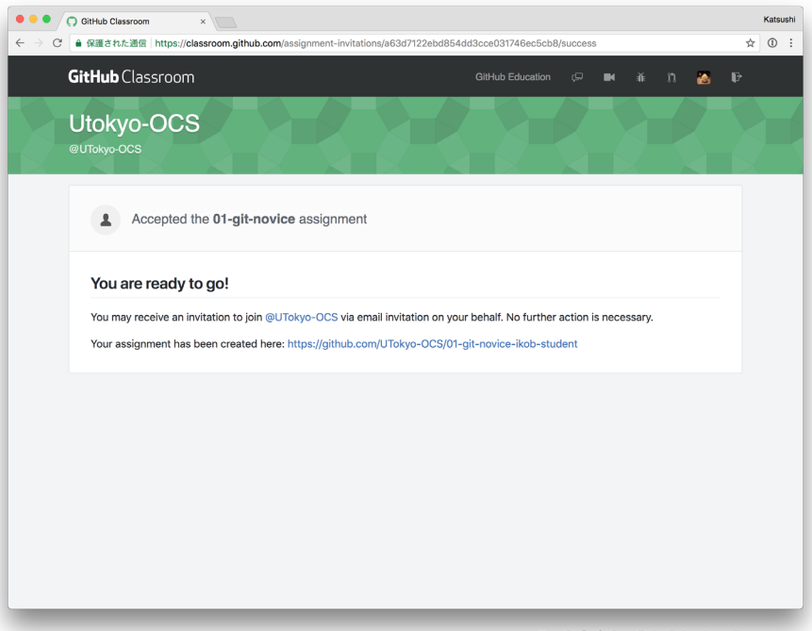
1.講義ページのお知らせページ
『第12回の課題でアクセスする GitHub Classroom の URL』のリンクをクリック

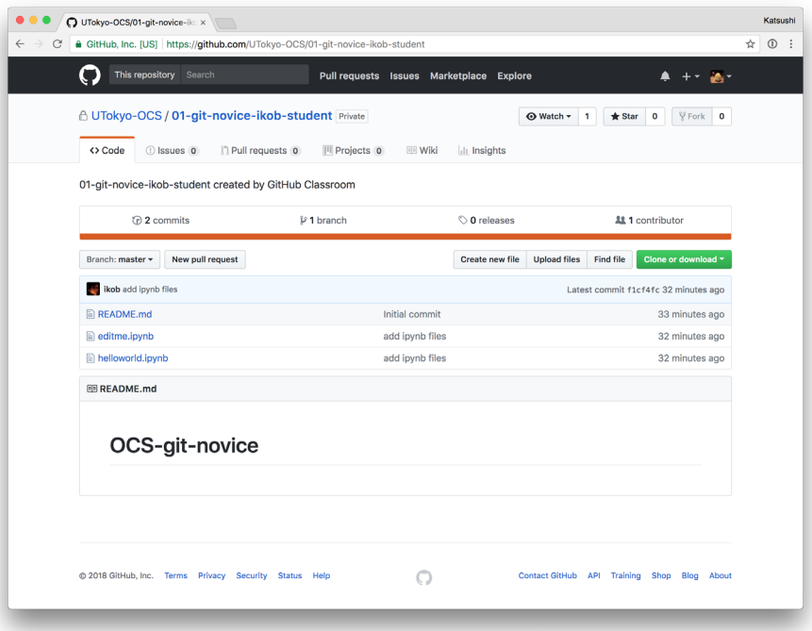
2.課題を Accept するかどうか聞かれるので、Accept する（右上）

3.“Your assignment has been created here: ”以降にアクセスしてみる

4.自身の GitHub レポジトリにアクセスできる。（右下）

5.メールが飛ぶので Accept するようにしてください。





演習:Flask + matplotlib サンプルを動かす
URL 取得と git clone

1.GitHub Classroom 登録で作られた GitHub レポジトリページに移動

2.“Clone or download” ボタンをクリック、URL を表示

3.URL をコピーする(右端のコピーボタンをクリック)

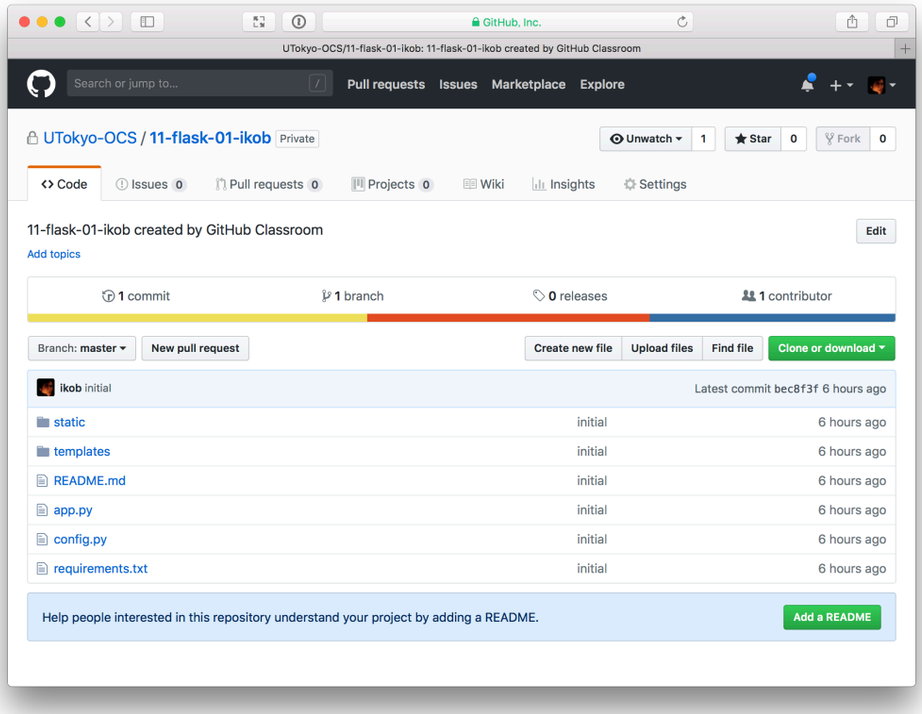
4.CLI で以下の git clone コマンドを実行する。途中で GitHub のユーザ名・パスワードを聞かれる:

```
$ git clone <コピーした URL をここにペーストする>
Cloning into '12-flask-02-ユーザ名'...
remote: Counting objects: 9, done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 9 (delta 0), reused 9 (delta 0), pack-reused 0
$
```

5.ディレクトリ”12-flask-02-ユーザ名”(以降作業ディレクトリ)が確認できれば複製は成功。

6.作業ディレクトリに移動、状態・ログを表示してみる

```
$ cd 12-flask-02-ユーザ名
$ git status
On branch master
Your branch is up to date with 'origin/master'.
$ git log
コミットログの表示
```

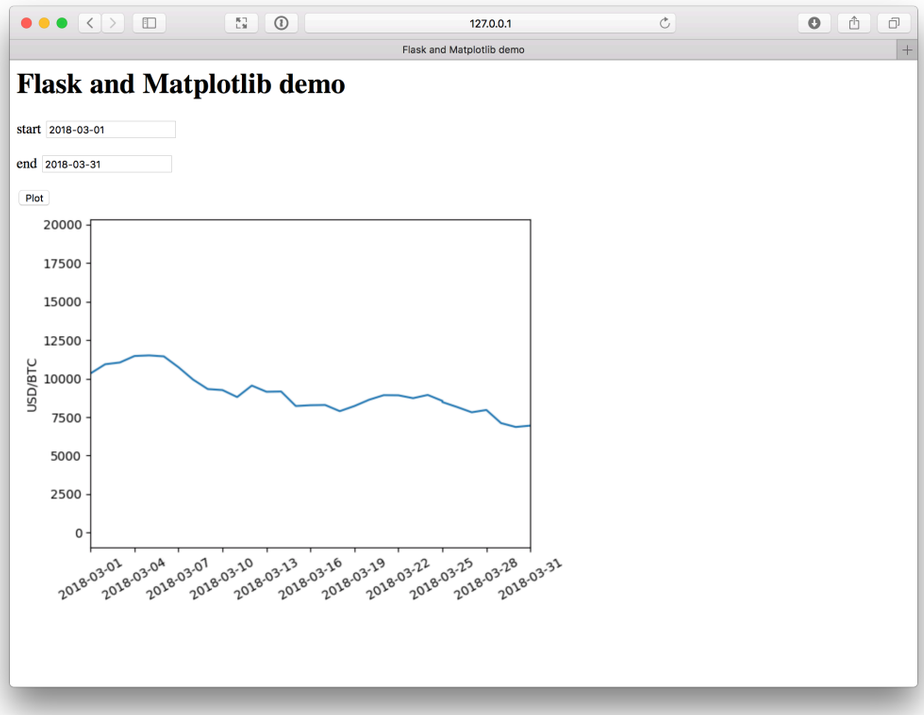


Flask + matplotlib サンプルを動かす
Flask を起動する

1.python インタプリタから app.py を起動する

```
$ python app.py
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 634-123-011
$
```

2.GoogleChrome から http://127.0.0.1:5000 にアクセス、動作を確認する。



演習:Flask サンプルを動かす
レポジトリの構成

```
12-flask-02-ikob/
├── BTC-USD.csv
├── README.md
├── app.py
├── config.py
├── requirements.txt
├── static
│   └── plotbtc.js
└── templates
    └── index.html
```

- BTC-USD.csv : ビットコイン/USD 相場データ
- app.py : Python プログラム本体
- config.py : 設定ファイル
- static : 静的コンテンツ
  - calculator.js : JavaScript プログラム
- templates : テンプレート
  - index.html : HTML テンプレート

Flask(再掲)

- Python 向けマイクロ Web フレームワーク
  - werkzeug, jinja2, good intentions を活用
- 概観

1.デコレータによる URL ルーティングと関数定義で Web サービスを構築できる

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello, World!'
```

2.URL の一部を変数として関数に渡せるため、REST (Representational State Transfer (REST))との相性が良い

```
@app.route('/user/<username>')
def show_user_profile(username):
    # show the user profile for that user
    return 'User %s' % username
```

3.HTTP メソッドも URL ルーティングで対応

```
from flask import request

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        return do_the_login()
    else:
        return show_the_login_form()
```

4.静的ファイル、テンプレートへの対応

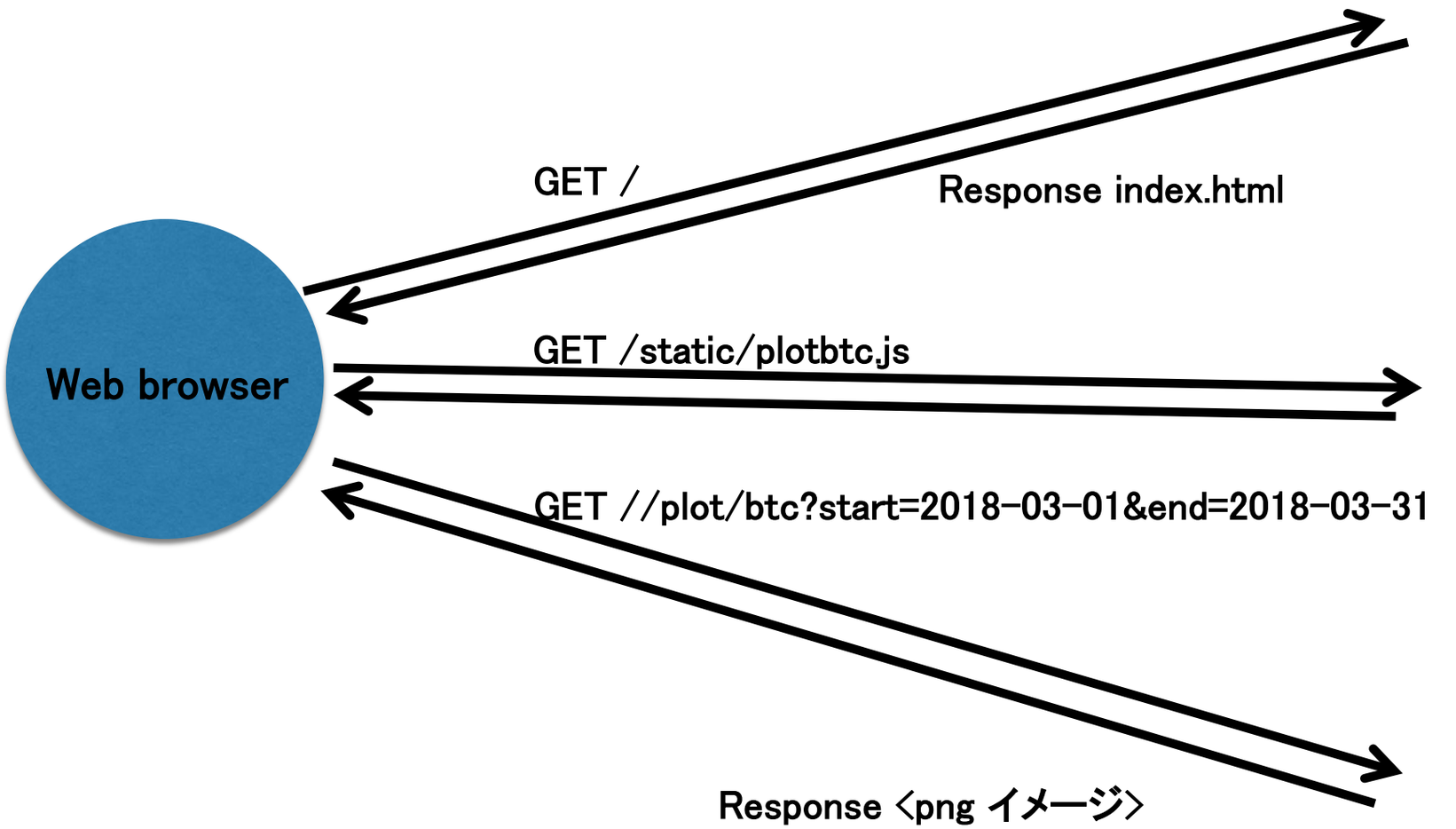
5.クライアントからサーバに送られるリクエストデータへのアクセスが容易

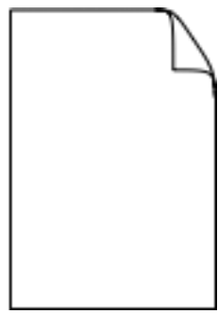
6.Cookie, ユーザセッションの取り扱い

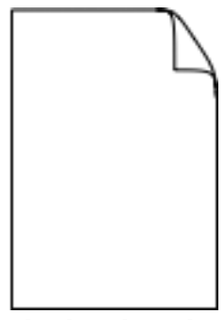
- 参考文献

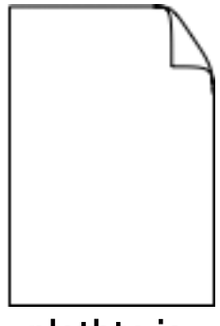
Robert Picard, "Explore Flask Documentation", https://exploreflask.com/en/latest/index.html, 2017
Robert Picard, 濱野司(訳),探検! Python Flask, 達人出版協会

Flask + matplotlib サンプルの構成




app.py


index.html


plotbtc.js

Note: 配布した Flask サンプルはかならずしも index.html はテンプレートである必要はない。
理由は app.py で書き換えないので。

app.py

```
 1 #app.py
 2 from flask import Flask, request, render_template
 3 import urllib
 4 import numpy as np
 5
 6 import matplotlib.pyplot as plt
 7 from matplotlib.dates import date2num
 8
 9 from io import BytesIO
10
11 import pandas as pd
12 from datetime import datetime, timedelta
13
14 app = Flask(__name__)
15
16 df = pd.read_csv("BTC-USD.csv", index_col = 0, parse_dates=True)
17 fig, ax = plt.subplots(1,1)
18 ax.plot(df["Open"])
19
20 @app.route("/")
21 def index():
22     return render_template("index.html")
23
24 @app.route("/plot/btc")
25 def plot_btc():
26
27     # Obtain query parameters
28     start = datetime.strptime(request.args.get("start", default="2017-12-1", type=str), "%Y-%m-%d")
29     end = datetime.strptime(request.args.get("end", default="2017-12-31", type=str), "%Y-%m-%d")
30
31     if start > end:
32         start, end = end, start
33     if (start + timedelta(days=7)) > end:
34         end = start + timedelta(days=7)
35
36     png_out = BytesIO()
37
38     ax.set_xlim([start, end])
39     ax.set_ylabel("USD/BTC")
40
41     plt.xticks(rotation=30)
42
43     plt.savefig(png_out, format="png", bbox_inches="tight")
44     img_data = urllib.parse.quote(png_out.getvalue())
45
46     return "data:image/png:base64," + img_data
47
48 if __name__ == "__main__":
49     app.run(debug=True, port=5000)
```

- 2 行目: flask モジュールの読み込み
- 6 行目: Flask オブジェクトの生成
- 16 行目: BTC 相場データの読み込み
  - **この例では相場データは事前に読み込んでおく**
- 17 – 18 行: 描画
- 20 – 12行: ルート "/" への　URL ルーティングと処理関数
- 22 行目: template の指定
- 24 – 25行: "/plot/btc" への URL ルーティングと処理関数
- 28 – 29 行: URL の query ブロックを Datetime オブジェクトに変換
- 31–34 行: 表示期間のチェック
- 36 行目: BytesIO オブジェクト、出力バッファの生成
- 38 行目: 表示幅の設定
- 43 行目: PNG 形式画像データの生成
- 44 行目: バイナリデータを base64 エンコーディングに変換
- 46 行目: (MIME ヘッダをつけて)画像データを返す

templates/index.html

```
 1 <html>
 2
 3 <head>
 4     <title>Flask and Matplotlib demo</title>
 5     <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
 6 </head>
 7 <body>
 8     <h1>Flask and Matplotlib demo</h1>
 9     <div>
10         <p>start
11         <input type=date id=start value="2018-03-01" min="2010-07-16" max ="2018-04-30">
12         </p>
13         <p>end
14         <input type=date id=end value="2018-03-31" min="2010-07-16" max ="2018-04-30">
15         </p>
16
17         <!-- The event handler is defined in the calculator.js. -->
18         <input type=button id=plot value="Plot">
19     </div>
20     <img id=plotimg></img>
21 </body>
22 <script type="text/javascript" src="/static/plotbtc.js"> </script>
23
24 </html>
```

- 7 – 38 行: ボディ
- 10 – 12 行:開始日設定
- 13 – 15 行: 終了日設定
- 20 行目: 画像要素
- 39 行目 : /static/plotbtc.js スクリプトの読み込み

- 3 – 6 行: ヘッダ
- 5 行目: jQuery の読み込み

static/plotbtc.js

```
 1 //
 2 // BTC plot
 3 //
 4 function plotbtc() {
 5 // Buid query parameter
 6     var param = {};
 7     param["start"] = document.getElementById("start").value;
 8     param["end"] = document.getElementById("end").value;
 9     var query = jQuery.param(param);
10
11 // Query with a new parameter
12     $.get("/plot/btc" + "?" + query, function(data) {
13         document.getElementById("plotimg").src = data;
14     });
15 };
16 //
17 // Register Event handler
18 //
19 document.getElementById("plot").addEventListener("click", function(){
20     plotbtc();
21 }, false);
22 plotbtc();
```

- 7-9 行: start, end 日付を取得、query パラメータを作成
- 12 行目: "/plot/btc" + "query パラメータ"に対する HTTP GET リクエストを に送る
- 13 行目 : GET リクエストの結果で id = plotimg のイメージ要素の src 属性(イメージ)を書き換える。
- 19 行目: id=plot で指定されるボタンクリック時の挙動を設定する(イベントハンドラの登録)
- 21 行目: (初期画面のため)一回実行する

ミニプロジェクトの例: Flask に対する機能追加

- 配布した Flask サンプルをもとに、講義前半におこなった matplotlib, cartopy 可視化を Web アプリケーションとして組み込む。
  - 描画パラメータを対話的に設定できる。
- 体感品質(UX: User eXperience) も考慮すること。
  - 例えば、描画に時間がかかりすぎるとストレスは大きい。(2/4　秒ルール)
  - UX に関しては、"User Experience Honeycomb"が知られている

User Experience Honeycomb
http://semanticstudios.com/user_experience_design/
© 2018 Semantic Studios