

Flaskチュートリアル
ツイッターの分析ツールを作って
デプロイしよう！



Flaskチュートリアル - Pythonでツイッターの分析ツールを作ってデプロイしよう！（動画つき！） -



Dai

2018/08/04 13:20

¥2,980



Dai

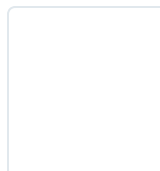
@never_be_a_pm

Youtuberデビューしました。

Flaskで作るWebアプリのチュートリアル、無料部分を動画で公開しました。

本当にnote見てできるの？って方向けに、無料部分に関しては動画で解説するようにしました。youtube.com/playlist?list=...

19:20 - 2018年8月4日



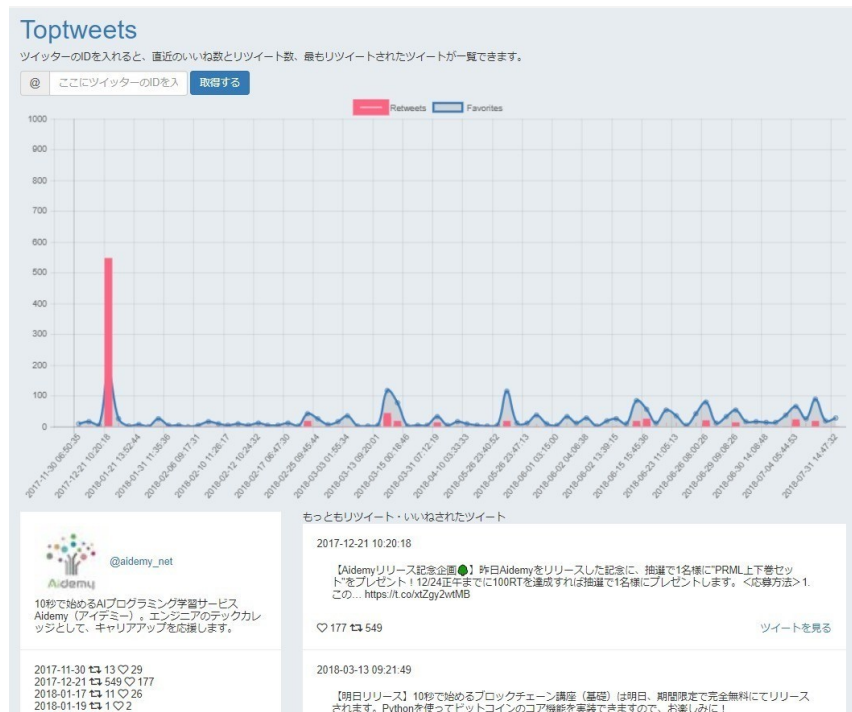
Flaskチュートリアル - Pythonでツイッターの...
youtube.com

149

28人がこの話題について話しています

PythonのWebフレームワークFlaskをご存知でしょうか？ PythonではDjangoに次いで有名なWebアプリケーションフレームワークです。

今回はFlaskでツイッターのAPIを利用して、ユーザーのリツイート数といいね数を日時集計し、グラフ描画できるアプリケーションの作り方をまとめてみたいと思います。完成はこんな感じのアプリケーションになります。



実際のアプリは、下記URLからアクセスしてみてください！

Top Tweets Finder

top-tweets.herokuapp.com

このチュートリアルで学ぶこと

以下の内容を学びます。

- Python (基礎についてはできていることを前提とします)
- HTML (基礎についてはできていることを前提とします)
- CSS (基礎についてはできていることを前提とします)
- Bootstrap : CSS/JavaScriptのフレームワーク
- Flask: PythonのWebフレームワーク
- Tweepy: PythonでTwitterのAPIを利用するモジュール
- Pandas: Pythonでデータを処理するモジュール
- font-awesome: ハートマークやリツイートマーク等を使えるCSSライブラ

リ

- ・ heroku: アプリをアップロードするシステム

このチュートリアルの対象者

こちら、メインはFlaskとディプロイを学ぶチュートリアルとなっています。ですので、Python、HTML、CSS、JavaScriptの基礎に関しては理解していることを前提とします。これらのスキルは必ずProgateさんで学んでおいてください。

Progate(プロゲート) | Learn to code, learn to be creative.

Progateはオンラインでプログラミングを学べるサービスです。プログラミングを学んでWEBアプリケーションを作ろう。

prog-8.com

また、途中でPandasやWeb APIを利用した処理を使います。もし難しければ、AidemyのPandasコースを先に受講してから学ぶのをおすすめします。

Aidemy | 10秒で始めるAIプログラミング学習サービスAidemy【アイデミー】

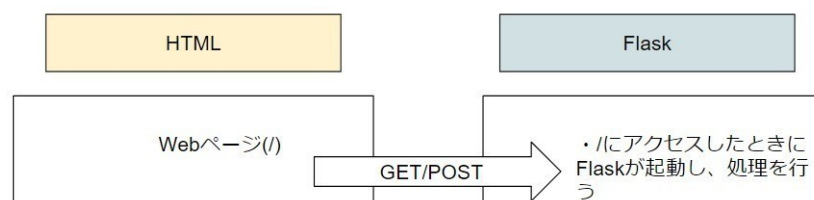
人工知能や機械学習を学んでいま話題のAIエンジニアにランクアップしよう。Aidemy【アイデミー】はエンジニアのためのデッ

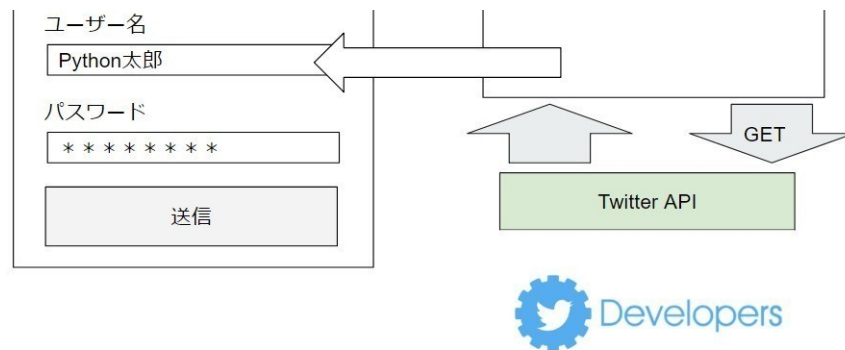
aidemy.net

また、ローカルの環境構築等も、Macで行われていることを前提としていますので、Pythonの環境構築はできているものとして進めます。

Flaskの基礎を学ぼう

さて、さっそくFlaskについて学びましょう。Flaskの役割を、一度図にして理解してみましょう。この章ではいくつか必要なキーワードを学んでいきます。





ルーティング

あるURL(ルーティング)にアクセスすると、HTMLが読み込まれるようになっているとします。例えば、noteだったら、“note.me/users/”みたいなURLにアクセスすることになりますね。ここで、ある特定のURLを、ルーティングといいます。

GET/POSTリクエスト

さて、上記のようなページがあるとしましょう。あるページでは、二種類のリクエストがあります。一つは、単純にページにアクセスするときに行うGETリクエストです。あるページにアクセスする場合は、そのページにたどり着くリクエスト、GETリクエストを送り、HTMLのデータをGETすることになります。

一方で、同じページでも、例えばログインフォームのように、サーバーに情報を送信する（上の図だとユーザー名とパスワード）場合は、送信するリクエスト、POSTリクエストを送ります。

重要な点は、ある一つのルーティングでも、複数のリクエストを持つ可能性があるということです。先ほどの例だと、ログインページであれば、

- ・あるページにアクセスする場合 => GETリクエスト
- ・あるページから、情報を送信する場合 => POSTリクエスト

となります。

先に完成した際のコードをお見せすると、以下のコードのように同じページでも複数のリクエストの場合によって処理を分けることがあるので、覚えておいてください。

```
@app.route('/', methods = ["GET" , "POST"])
def index():
    if request.method == 'POST':
        user_id = request.form['user_id']
        tweets_df = get_tweets_df(user_id)
        grouped_df = get_grouped_df(tweets_df)
        sorted_df = get_sorted_df(tweets_df)
        return render_template(
            'index.html',
            profile=get_profile(user_id),
            tweets_df = tweets_df,
            grouped_df = grouped_df,
            sorted_df = sorted_df
        )
    else:
        return render_template('index.html')
```

Flaskの下準備

それでは、Flaskを実際に使えるように環境設定していきましょう。

Macの場合はこちらの記事を参照してください。

PythonとFlaskでWebサーバを立ち上げる - Qiita

- 単体のPythonスクリプトとか、GoogleAppEngineで簡単なプログラムを動かしたことはあるんですが、単体
qiita.com

Winの場合はこちらの記事を参照してください。

Flaskを使ってみる - Qiita

公式 : <http://flask.pocoo.org/> Pythonのウェブフレームワーク
であるFlaskを公式のドキ
qiita.com

必要なライブラリをインストールした前提で進めていきます。

Flaskで作成するアプリのフォルダを作成しましょう。保存したいディレクトリを選択して

```
$ mkdir top-tweets # 注 mkdirはmake directoryの略で、フォルダを作成するコマンド
```

と入力し、top-tweetsフォルダを作成します。top-tweetsの中には、以下のファイルを保存することになります。空のファイルを作成します。

```
top-tweets
├─ app.py
├─ config.py
├─ templates
└─ index.html
```

app.pyはFlaskを保存するPythonのファイルです。config.pyはTwitterのアクセストークン等の認証情報を入れるファイルとなります。

注意する点は、templatesフォルダです。templatesフォルダの中には、HTMLファイルを入れることになっています。そこで、index.htmlという空のファイルを入れておいて下さい。こうすることで、FlaskからのちほどHTMLを読み込めるようになります。

app.pyとindex.htmlを編集して、Hello Worldを表示する

さて、下準備が完了したので、さっそくapp.pyを編集していきましょう。

```
from flask import Flask, render_template, request, logging, Response, redirect

# Flask の起動
app = Flask(__name__)

@app.route('/')
def index():
    return render_template('index.html')

app.run(host="localhost")
```

さて、ここまでを確認しておきます。ここに関しては、Flaskのライブラリなので、全部入れておいてください。

```
from flask import Flask, render_template, request, logging, Response, redirect
```

次に、flaskを初期化して、インスタンスを作成します。このappから、各種flaskのメソッドを呼び出すことができるようになります。

```
app = Flask(__name__)
```

次に、一番大切なところについてご説明します。さきほど説明した、ルーティングについてです。あるルーティングにアクセスしたときに、index.htmlを呼び出すという、いたって単純なコードとなります。

```
@app.route('/') # "hoge.com/"にアクセスしたときに、実行される関数
def index():
    return render_template('index.html') # index.htmlを呼び出す
```

最後に、起動する処理を書きます。

```
app.run(host="localhost")
```

ここまでできれば、あとは実行してみましょう。python app.pyでファイルを実行すると、http://localhost:5000/ 上でFlaskアプリが起動するようになります。

```
$ python app.py
* Running on http://localhost:5000/ (Press CTRL+C to quit)
```

これで起動できるようになります。ちなみに、ローカルホストとは、「自分自身」を意味するホスト名となります。よくわからない人はこちらの記事を参照してください。

Error 403 (Forbidden) | 「分かりそう」で「分からない」でも「分かった」気になれるIT用語辞典

403エラーページです。用語の意味を「ざっくりと」理解するためのIT用語辞典です。

wa3.i-3-i.info

起動した際、もろもろNo module called "なんとか"みたいなエラーが出てきたら、必要なライブラリがインストールされていない可能性がありますので、pip install なんとか をしてあげてください。

HTMLを編集する

さて、起動して読み込むとエラーが出ると思います。なぜなら、アクセスしたときにHTMLファイルを読み出すはずなのですが、HTMLのファイルが存在しないからです。そこで、HTMLファイルを編集していこうと思います。

Bootstrap

BootstrapのテンプレートをHTMLに貼り付けます。Bootstrapは、簡単に言うと、CSSやJavaScriptを使わなくても、HTMLファイルだけである程度簡単なデザインができるフレームワークです。今回は、Bootstrap CDNを利用して、直接CSSファイル等をダウンロードするのではなく、ネット上でCSSファイルを読み込むようにします。

Bootstrap3日本語リファレンス

Bootstrapは、HTML、CSS、Javascriptからなる人気のフレームワーク。Bootstrap3日本語リファレンス
bootstrap3.cyberlab.info

index.htmlに以下のBootstrapのテンプレートコードをコピーして貼り付けてください。

```
<!DOCTYPE html><html lang="ja">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Bootstrapの基本テンプレート</title>

<!-- ここがBootstrapのCDN -->
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap.min"
<!--[if lt IE 9]>
    <script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js">
    <script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></sc
<![endif]-->
</head>
<body>
    <h1>Bootstrapの基本テンプレート</h1>

    <!-- ここがBootstrapのCDN -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.r
    <!-- ここがBootstrapのCDN -->
    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstra
</body>
</html>
```

< >

再度、以下のコードを実行して、Flaskを起動します。

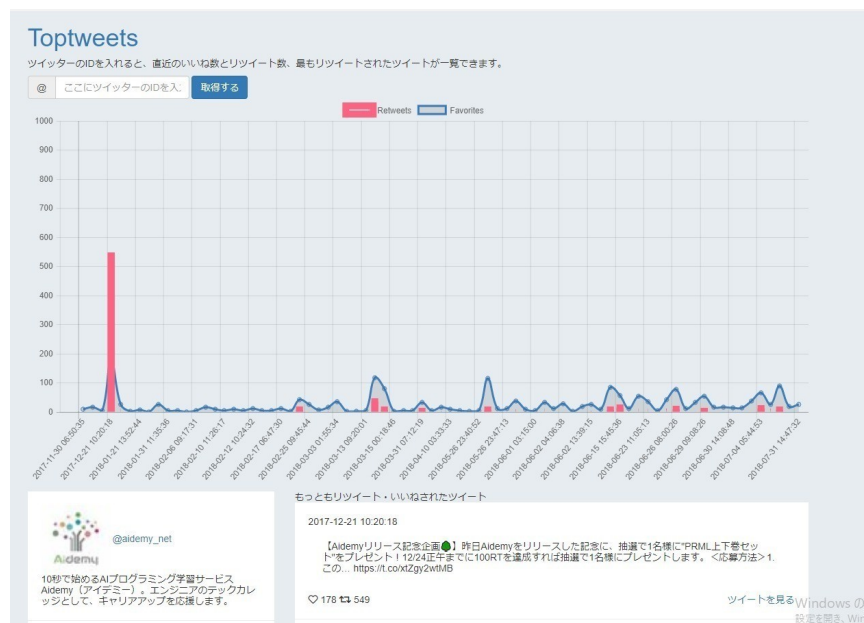

```
$ python app.py
* Running on http://localhost:5000/ (Press CTRL+C to quit)
```

そしてhttp://localhost:5000/を起動し、アクセスすると、「Bootstrapの基本テンプレート」と表示されていれば、うまくいっています。

こんな感じで、Flaskアプリは利用できます。

Bootstrapで、見た目を作る

さて、Bootstrapでひな形を作ったところで、さっそく見た目の部分を作成していこうと思います。最終的な画面としては、このような画面を創っていきます。



さきに、完成形のindex.htmlをお見せしようと思います。そこから、細かく解説していきます。今は理解できなくてもよいので、コード全体でどのような処理をしているのかざっとみてみましょう。

```
<!DOCTYPE html><html lang="ja">
<head>
<meta charset="utf-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>Top Tweets Finder</title>
<link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/css/bootstrap.min.css" rel="stylesheet">
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.2.0/css/all.css">
<script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/2.1.4/Chart.min.js"></script>
<!--[if lt IE 9]>
<script src="https://oss.maxcdn.com/html5shiv/3.7.2/html5shiv.min.js"></script>
<script src="https://oss.maxcdn.com/respond/1.4.2/respond.min.js"></script>
<![endif]-->
<style>
body{
background-color:#e6ecf0;
}
div{
line-height: 1.2;
}
.white-container{
border-bottom: 1px solid #e6ecf0;
padding:20px;
background-color:#ffffff;
}
#tweet-text{
padding:20px;
margin-bottom:10px;
line-height: 1.2;
}
</style>
</head>
<body>
<div class="container">
<div class="row">
<div class="col-md-12">
<h1><a href="/">Toptweets</a></h1>
<p>ツイッターのIDを入れると、直近のいいね数とリツイート数、最もリツイートされたツイートが取得できる。</p>
<form class="form-inline" method="post">
<div class="form-group">
<label class="sr-only" for="user_id"></label>
<div class="input-group">
<span class="input-group-addon">@</span>
<input id="user_id" name="user_id" placeholder="ここにツイッターのIDを入力">
</div>
</div>
<button type="submit" class="btn btn-primary">取得する</button>
</form>
</div>
</div>
<!-- chart -->
<div class="row">
<div class="col-md-12">
<canvas id="chart"></canvas>
</div>
</div>
<div class="row">
<!-- プロフィール -->
<div class="col-md-4">
{% if profile %}
<div class="white-container">
<div>

<a href="https://twitter.com/{{profile.user_id}}">@{{profile.user_id}}</a>
</div>
<div>{{profile.description}}</div>
</div>
{% endif %}
</div>
```

```

{% if profile %}
<div class="white-container">
  <div>
    {% for index, row in grouped_df.iterrows() %}
      <div>
        {{index}}
        <i class="fas fa-retweet"></i> {{row["retweets"]}}
        <i class="far fa-heart"></i> {{row["fav"]}}
      </div>
    {% endfor %}
  </div>
{% endif %}
</div>
<div class="col-md-8">
{% if profile %}
<div>もっともリツイート・いいねされたツイート</div>
{% for index, row in sorted_df.iterrows() %}
  <div class="white-container">
    <div>{{row["created_at"]}}</div>
    <div id="tweet-text">{{row["text"]}}</div>
    <div>
      <span><i class="far fa-heart"></i> {{row["fav"]}}</span> <span><i
      <a class="pull-right">ツイートを見る</a>
    </div>
  </div>
{% endfor %}
{% endif %}
</div>
</div>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.mi
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.1/js/bootstrap.m
<script>
{% if profile %}
// bar chart data
var barData = {
  labels : [{% for index, row in tweets_df.sort_values(by="created_at", asc
    "{{row['created_at']}}",
    {% endfor %}],
  datasets : [
    {
      label: "Retweets",
      backgroundColor: 'rgba(255, 99, 132, 0.2)',
      borderColor: 'rgba(255,99,132,1)',
      borderWidth:10,
      bezierCurve : false,
      data : [{% for index, row in tweets_df.sort_values(by="created_a
        {{row["retweets"]}},
        {% endfor %}]
    },{
      label: "Favorites",
      data : [{% for index, row in tweets_df.sort_values(by="created_a
        {{row["fav"]}},
        {% endfor %}],
      type: 'line',
      borderColor: 'rgb(63, 127, 191)',
    }
  ]
}
// draw bar chart
var mychart = document.getElementById("chart");
var chart = new Chart(mychart, {
  type:'bar',
  data:barData,
  options: {
    scales: {
      yAxes: [
        {
          ticks: {
            beginAtZero: true,
            min: 0,
            max: 1000
          }
        }
      ]
    }
  }
});

```

```

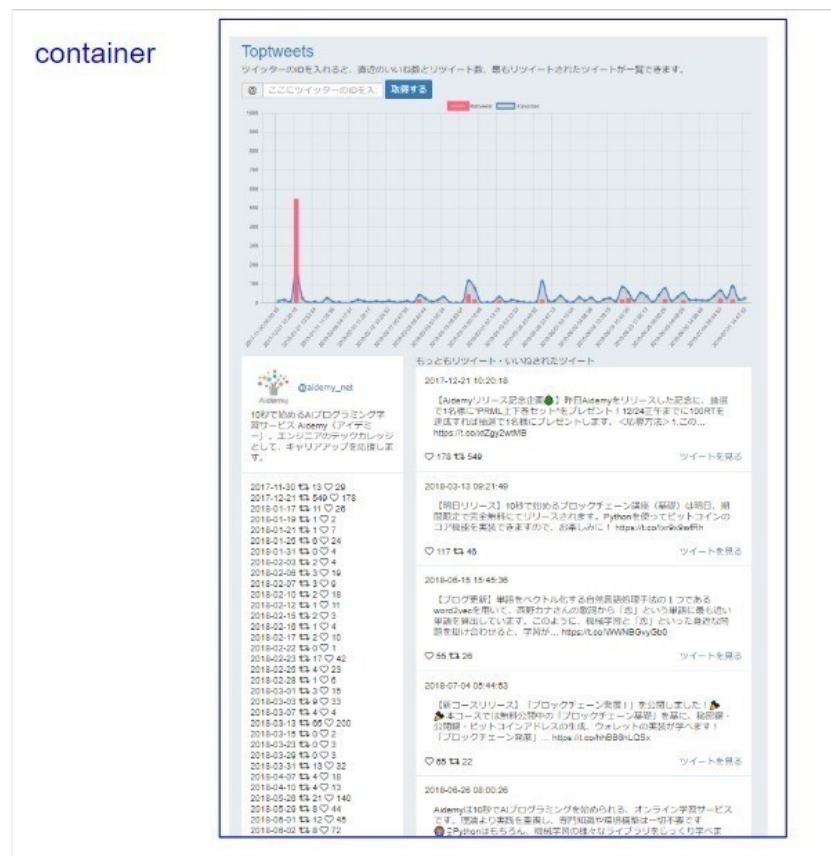
    {% endif %}
</script>
</body>
</html>

```

Bootstrapのグリッドデザイン

さてさて、Bootstrapでは、グリッドデザインというシステムを利用しています。配置に関することなのですが、container, row, columnという単位にわけて、HTML上でコードの場所を指定することができます。

といっても、よくわからないので、先に図で示します。こちらがcontainerのイメージ。



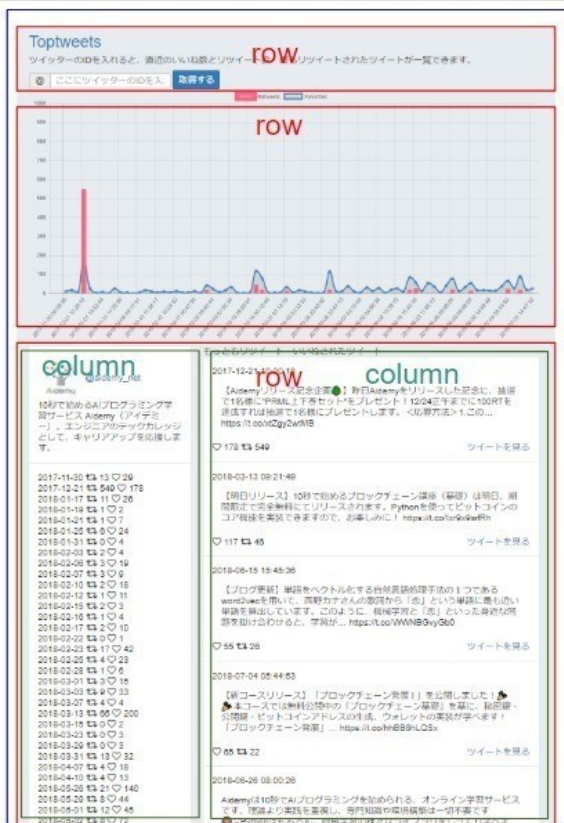
次に、こちらがrowのイメージ





で、このrow(行)の中を、column(列)でさらに分割すると、こんなイメージになります。

container



まとめると

- containerが全体を覆う箱
- rowが行
- columnが行 (row) を分割した列のこと

となります。

さてさて、columnですが、いま1:2くらいで、一番下のrowを分割しているのがわかるかと思います。普通にCSSファイルでこれをやろうとすると、divに対して細かいサイズを指定してあげなければなりません。

しかし、Bootstrapでは、rowを12分割して、そのうちのcolumnのサイズを指定することによって、なんと簡単にrowの中を分割することができるようになります。

実際に、一部コードを省略して、最後の行がどのように設定されているか見てみましょう。12を1:2に分けるのであれば、4:8に分けてあげれば大丈夫です。それを以下コードで実装しています。

```
<div class="row">
  <!-- プロフィール -->
  <div class="col-md-4">
    {% if profile %}
      <div class="white-container">
        <div>
          
          <a href="https://twitter.com/{{profile.user_id}}">@{{profile.user_
        </div>
        <div>{{profile.description}}</div>
      </div>
    {% endif %}
    {% if profile %}
      <div class="white-container">
        <div>
          {% for index, row in grouped_df.iterrows() %}
            <div>
              {{index}}
              <i class="fas fa-retweet"></i> {{row["retweets"]}}
              <i class="far fa-heart"></i> {{row["fav"]}}
            </div>
          {% endfor %}
        </div>
      </div>
    {% endif %}
  </div>
  <div class="col-md-8">
    {% if profile %}
      <div>もっともリツイート・いいねされたツイート</div>
      {% for index, row in sorted_df.iterrows() %}
        <div class="white-container">
          <div>{{row["created_at"]}}</div>
          <div id="tweet-text">{{row["text"]}}</div>
          <div>
            <span><i class="far fa-heart"></i> {{row["fav"]}}</span> <span><i
            <a class="pull-right">ツイートを見る</a>
          </div>
        </div>
      {% endfor %}
    {% endif %}
  </div>
</div>
```

<div class="row">の中で、<div class="col-md-4">と書いて、divファイルを4/12のサイズにしてください！と指示しています。また、下のほうを見ると、<div class="col-md-8">指示していますが、これも8/12のサイズにしてください！という風に指示をしているので、一つのrowの中に1:2でしっかりと分割されて表示されているというわけです。ちなみにcolはcolumn(列)の略で、最後の数字が分割する大きさを示した数になります。

さて、途中のmdとは？となった方もいらっしゃると思いますが、これはmediumの略です。これは、画面サイズを指定したものなんですよ。表示するデバイスの画面領域によって、サイズを変更することができます。mdの場合はタブレット等のサイズになりますね。

画面幅	Extra small	Small	Medium	Large	Extra large
クラスの書き方	.col-xs-	.col-sm-	.col-md-	.col-lg-	.col-xl-

仮にスマホとPCの列を変えたい場合は、col-sm-12 col-md-4みたいな形に変えてあげれば、スマホとタブレットの列のサイズを変えることができるわけです。これは圧倒的にCSSより楽だと思いませんか？

このように、デバイスごとによって画面領域を変えられるデザインのことを、**レスポンスデザイン**といたりします。

ここまで理解ができたなら、復習のため、こちらのサイトを見てみるとよいでしょう。

Bootstrapのグリッドシステムの使い方を初心者に向けておさらいする

Bootstrapのグリッドシステムとは？ 実例と一緒にグリッドシステムに用意されたクラスを復習します
websae.net

ヘッダーとフォームをHTML上に作成する

では、最初にヘッダーとフォームの部分を、HTML上で作成しようと思います。ここにあたる部分ですね。



```
<div class="container">
  <div class="row">
    <div class="col-md-12">
      <h1><a href="/">Toptweets</a></h1>
      <p>ツイッターのIDを入力すると、直近のいいね数とリツイート数、最もリツイートされたツイートが一覧できます。
      <form class="form-inline" method="post">
        <div class="form-group">
          <label class="sr-only" for="user_id"></label>
          <div class="input-group">
            <span class="input-group-addon">@</span>
            <input id="user_id" name="user_id" placeholder="ここにツイッターのID"
          </div>
        </div>
        <button type="submit" class="btn btn-primary">取得する</button>
      </form>
    </div>
  </div>
</div>
```



順にコードを解説していきます。まず、全体をおおうcontainerをbodyタグの下に追加しましょう。

その中に行にあたるrowを設定していきます。ここは1列しか使わないので、col-md-12を指定しています。そのあとにh1タグで、サイトのルートを指定してあげています。

bootstrapのtemplateを利用して、フォームを作成しよう。

ではでは、さっそくbootstrapの神髄を見ていこうと思います。Bootstrapには、フォームのテンプレートがあるので、さっそくそれを利用してフォームを作成していきましょう。こちらのURLを見てみて下さい。

インライン・フォーム内のインプット・グループ « フォーム « CSS « Bootstrap3日本語リファレンス

Bootstrap3では、インライン・フォーム内でインプット・グループを使うこともできる。
bootstrap3.cyberlab.info

このフォームですね。

インライン・フォーム内のインプット・グループ

Input groups

インライン・フォーム内でインプット・グループを使うこともできる。

サンプル

¥

00

送金

ソースコード

```
<form class="form-inline">
  <div class="form-group">
    <label class="sr-only" for="inputEmail">金額</label>
    <div class="input-group">
      <span class="input-group-addon">¥</span>
      <input type="text" class="form-control">
      <span class="input-group-addon">00</span>
    </div>
  </div>
  <button type="submit" class="btn btn-default">送金</button>
</form>
```

このフォームをそのままコピーして、一部を変更します。

まず、form内のメソッドをpostに指定してあげます。こうすることで、formを送信ボタンを押したときに、postリクエストを送信することができます。

また、inputクラスのnameにuser_idを指定してあげてください。こちらのnameに指定した値を、あとでFlask上で読み込むことができるようになります。

```
<form class="form-inline" method="post">
  <div class="form-group">
    <label class="sr-only" for="user_id"></label>
    <div class="input-group">
      <span class="input-group-addon">@</span>
      <input id="user_id" name="user_id" placeholder="ここにツイッターのID">
    </div>
  </div>
  <button type="submit" class="btn btn-primary">取得する</button>
</form>
```

<

>

あとは最後に、buttonの色を指定してあげます。クラスに、btn btn-primaryを指定してあげると、青い感じのボタンになります。

ここまですべてをHTMLで保存して、もう一度

```
$ python app.py
```

で起動してあげると、フォームが反映されると思います。Bootstrapを使うと、かなりいい感じにおしゃれなフォームやボタンを作れるので、フロントエンドに手が回らない人には非常におすすめです。

HTMLから入力した値を、Flask上で受け取ろう

さて、さきほどformを作成しました。POSTリクエスト（フォームの値送信時）と、ただそのページにアクセスしたGETリクエストの時で、処理をかえます。それがこちらのコードです。

```
@app.route('/', methods = ["GET" , "POST"])
def index():
    if request.method == 'POST':
        return render_template('index.html')
    else:
        return render_template('index.html')
```

変わった点としては、methodsにリスト形式でGETとPOSTが追加されたのと、条件分岐でPOSTリクエストが送られたときの値が増えました。

さて、最初にルーティングの章で学びましたが、同じindex.htmlでも、ただページにアクセスするGETと、ツイッターIDを送信するPOSTの場合があります。form上でPOSTを送信する場合は、ただ表示するのではなく、今後ツイッターIDをもとにRT数やいいね数を取得して、その内容をindex.htmlに反映するので、別の処理を行うことになります。そこで、POSTリクエストの処理を加えてあげるわけですね。

送信したuser_idを受け取る

フォームから送信した値を受け取りましょう。HTML上では、nameでuser_idと指定したので、その値を受け取るには以下のコードになります。これで、HTMLから送信した際のuser_idを取得することができます。

この取得したuser_idを、HTML上に反映する場合は、この受け取った値をindex.htmlに返してあげることができます。それがこちらのコードです。

```
@app.route('/', methods = ["GET" , "POST"])
def index():
    if request.method == 'POST':
        user_id = request.form['user_id'] # formのname = "user_id"を取得
        return render_template('index.html', user_id = user_id)
    else:
        return render_template('index.html')
```

順に解説していこうと思います。まず、このコードで、HTTPのPOSTリクエストが送られた時を条件にします。

```
if request.method == 'POST':
```

そして、今度はHTMLから受け取った値を、python上で取得します。

```
user_id = request.form['user_id'] # formのname = "user_id"を取得
```

そのユーザーIDを、再度HTMLに反映するには、以下のコードを実行する必要があります。

```
return render_template('index.html', user_id = user_id)
```

これで、以下の処理を実行することができます。

さて、今HTML上にデータが送信されました。この送信したデータを、受け取れるようにしましょう。index.htmlをひらき、以下のコードを追加します。

```
<!-- コードを追加 -->
{% if user_id %}
<p>{{user_id}}</p>
{% endif %}
<!------->
<form class="form-inline" method="post">
  <div class="form-group">
    <label class="sr-only" for="user_id"></label>
    <div class="input-group">
      <span class="input-group-addon">@</span>
      <input id="user_id" name="user_id" placeholder="ここにツイッターのIDを入力">
    </div>
  </div>
  <button type="submit" class="btn btn-primary">取得する</button>
</form>
```



最初に、index関数から送信した値の引数のデータが存在するかを確認します。Flask上のデータをHTMLで確認するには、{% 条件節 %}というフォーマットで記入してあげます。また、実際の値を反映する場合は、{{ 入力したい値 }}のように入力します。

ここまでの流れを一度振りかえりましょう。

1. HTMLで特定の値を入れて、POSTリクエストを送る
2. Flask上で、ルーティング（URL）とHTTPリクエスト（GET・POST）に合致した関数を実行する
3. 関数を処理した後に、特定のHTMLに特定の値を送信する
4. HTMLに関数から送信された値を取得し、反映する

ここまでの流れを実際に反映されているかどうか、確認してみましょう。


```
$ python app.py
```

を実行し、さきほどのフォーム入力画面をひらき、実際に値を入力してみます。そして、送信をクリックしたら、再度ページが遷移するかと思います。その時に、一度入力したデータがHTML上に反映されていれば完了です。

この続きをみるには

(残り19,673文字/画像7枚)

このノートが含まれている マガジンを購入・購読する



基礎チュートリアルが1980円、実践チュートリアルが2980円～です。基礎チュートリアルを3本以上、もしくは実践チュートリアルを2本以上ご購入いただく方は、こちらのマガジンを購読していただくほうがお得です。



Daiの技術チュートリアル (月...
Daiの技術チュートリアルの月額ブ...
月額4,980円

購読して続きをみる

または

このノートを単体で購入する



Flaskチュートリアル - Pytho...
Dai
2,980円

購入して続きをみる

この記事が気に入ったら、サポートをしてみませんか？気軽にクリエイターを支援できます。

サポートをする

このクリエイターのおすすめノート

Google Data StudioとスプレッドシートとIFTTTで、ツイッターハックのデータ分析基盤を作ってみた。

23 Dai



Aidemyで学んだことを、実際に自分でコーディングしたいときにおススメのツールを紹介 -Microsoft Azure Not...

34 Dai



Oculus Go買ってしまいました

16 Dai



【Twitter X Python】結局、何をしたらフォロワーが増えるのかデ...

6-7月のメディア収益と施策、8月から年内の目標



Dai

教育ITスタートアップ → メガベンチャー → AIスタートアップ | プログラミング初心者が、自分自身でプロダクトを作れるようになる技術チュートリアル・試行錯誤した結果などを記事にまとめています。 https://twitter.com/never_be_a_pm

フォロー



Daiの技術チュートリアル (月額プラン)

Daiの技術チュートリアルの月額プランです。定期購読している期間は、その月に発行されたすべてのチュートリアルが見えるようになります。また、新しいチュートリアルがリリースされても、その内容は購読期間であれば見ることができます。過去のnoteを見るためには、こちらのフォームから...

4,980円 / 月

こちらもおすすめ

7/1東京で「写ルンですフォトウォーク」を開催します！

14 0



平井裕士



...

「低価格」の先にあるものの

19 0



はるのひ



...

facebookとfacebook liteをトレースして比較

11 0



somewhere coffee



...

AbemaTVのアクセシビリティ 小さな一歩

85 0



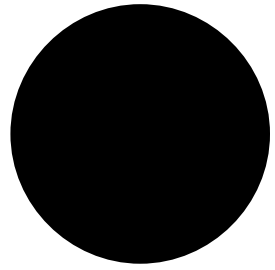
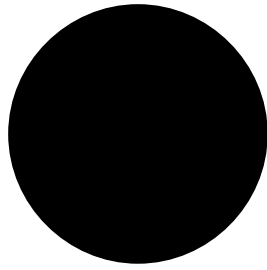
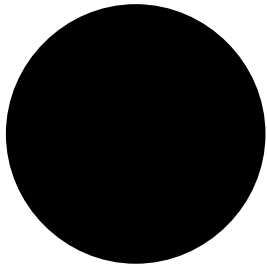
しゅんずけ | AbemaTV

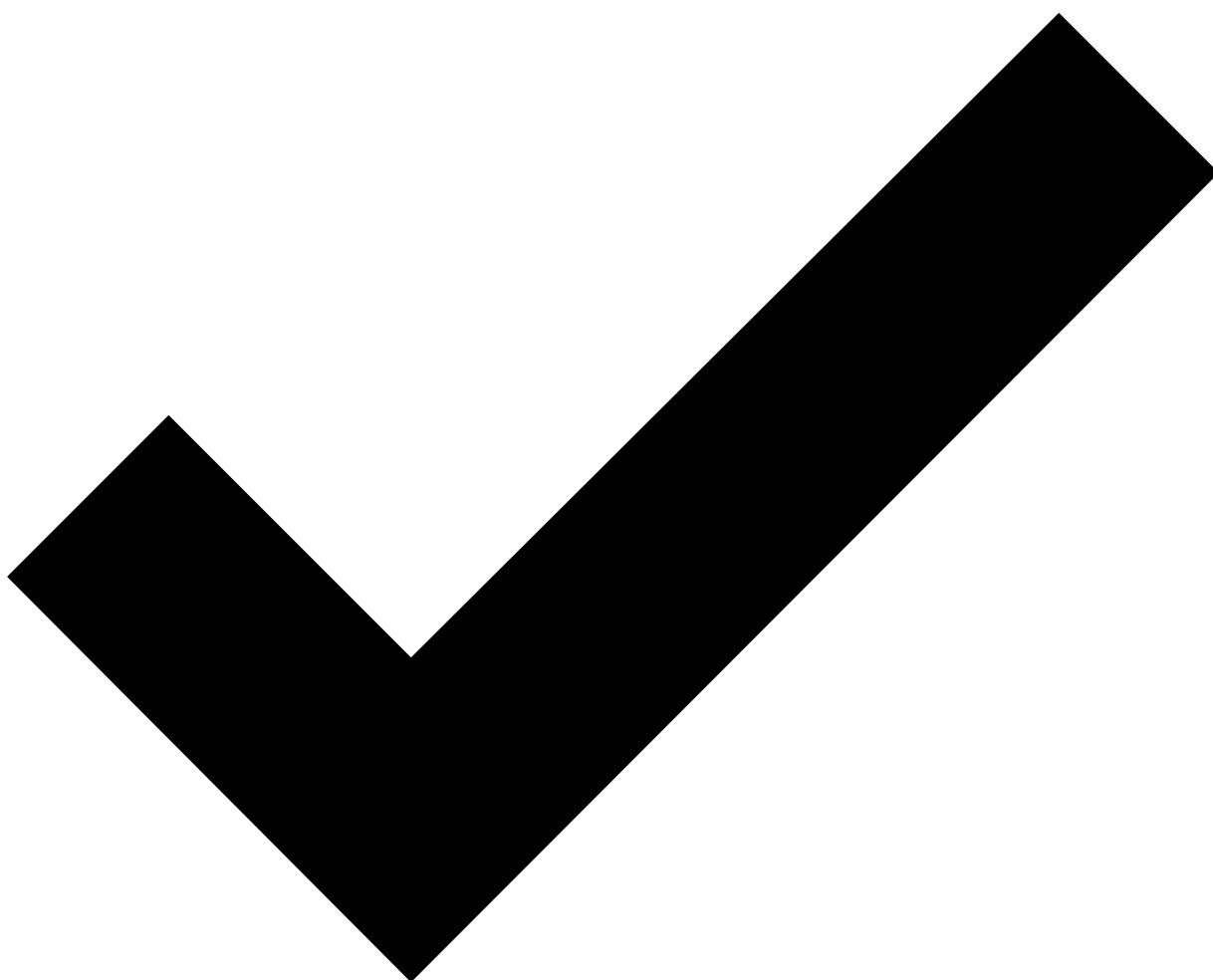


...

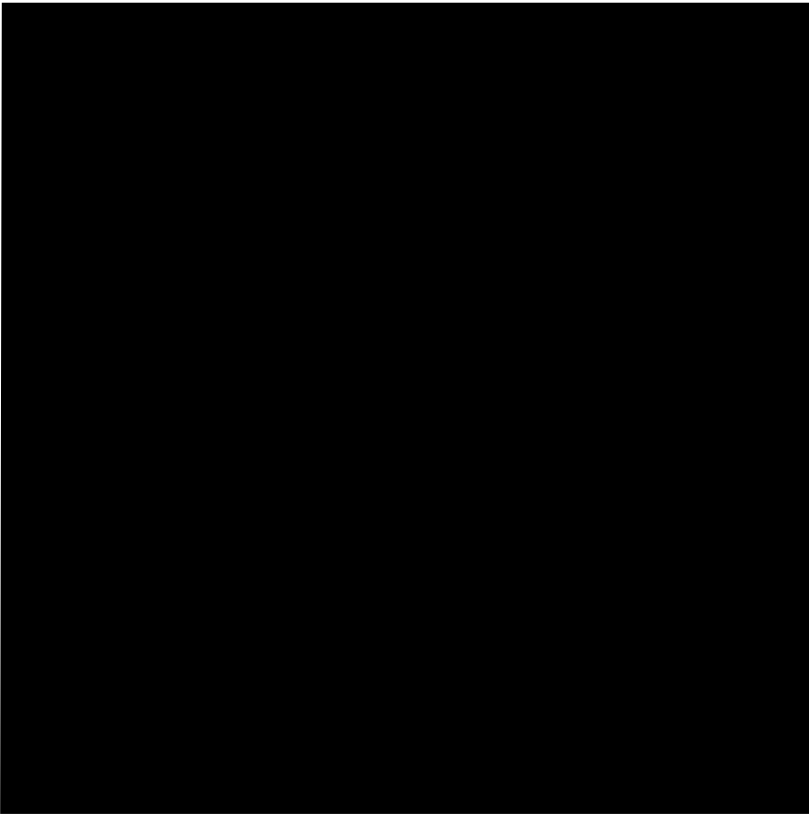


ページが保存されました！





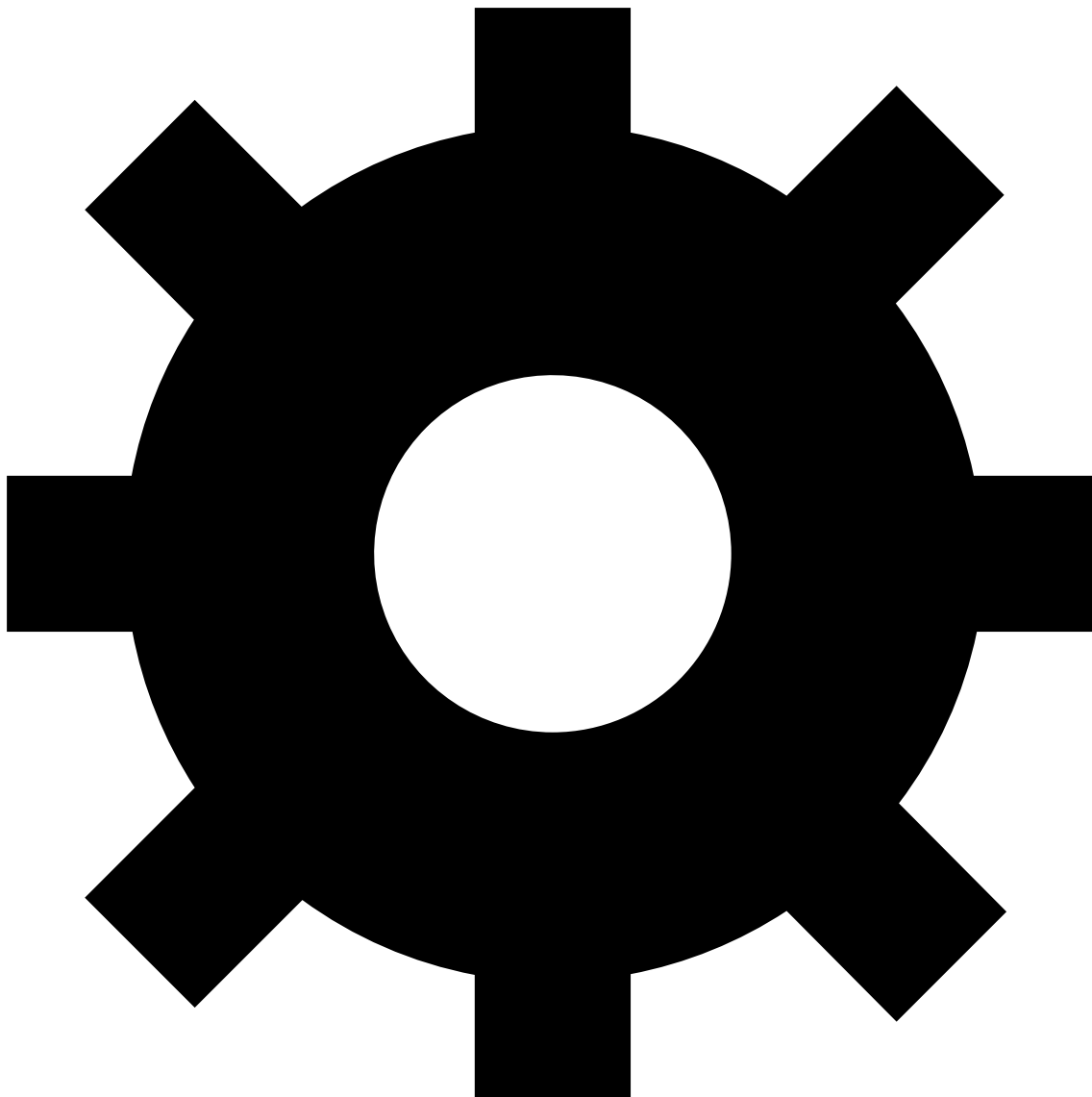
[ページをアーカイブ](#)



ページを削除



Pocket を開く



設定

タグを追加