



python™

プログラミング勉強会

公式サイトから資料を
DLできます。
パスワード: pk_py1214
puzzleknot.wp.xdomain.jp



puzzleknot
Tohoku University Programming Club

今日の流れ

- 第5回
- Twitter APIの利用
- pythonスクリプトでTweetさせる
- BeautifulSoupを使いこなす
- 天気予報をTweetさせる
- 最後に

Twitter APIの利用

3

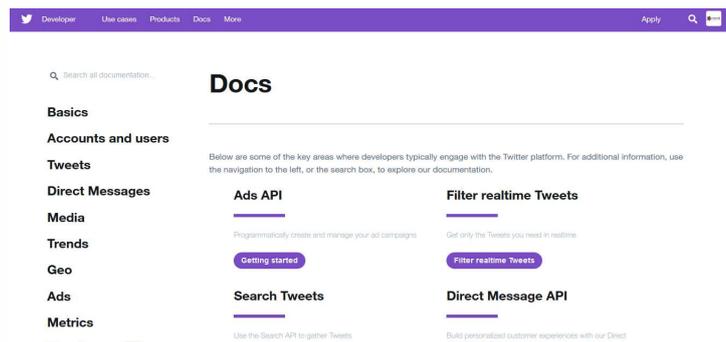
APIとは

- 外部からソフトウェアの一部を利用できるようにしたもの
- 公開されたAPIを利用すると、各種サービスで行えることがプログラム上で行える！
- 例えばTwitterなら…
 - タイムライン表示
 - ツイート
 - フォロワー など…

4

APIとは

- 大抵の場合、公式で用意されている説明は英語です（つらい）



PythonからAPIを使う

- しかし、こういう説明を読む必要があるのは自分でTwitterに関するモジュールを作ったり、既存のモジュールじゃ物足りない場合です
- とりあえずは既存のモジュールでAPIを利用しましょう

PythonからAPIを使う

- Pythonで使えるTwitter APIに関するモジュールはさまざまあります
- 今回は **tweepy** を使います
- `$ pip install tweepy`

7

PythonからAPIを使う

- Twitter APIを利用するには、公式からアプリ開発者として認証を受ける必要がある
- …とはいっても、そんなに大変なことではありません

8

PythonからAPIを使う

- ツイートさせたいアカウントでログインします
- 電話番号をアカウントに紐付けます
<https://twitter.com/settings/devices>

9

PythonからAPIを使う

- Twitter Appにアクセスして、使用用途などを書きます
- <https://apps.twitter.com/>
- [Create New App]

10

PythonからAPIを使う

- 書き方の例
- Name: 簡潔に (tweepy-test など)
- Description: 説明を(できれば)英語で
- Website: 本来はコードを公開するサイトのURLを書きますが、とりあえず `https://twitter.com/` でOKです

11

PythonからAPIを使う

Application Details

Name *

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens. (If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URL

Where should we return after successfully authenticating? OAuth 1.0a applications should regardless of the value given here. To restrict your application from using callbacks.

Privacy Policy URL

The URL for your application or service's privacy policy. The URL will be shared with users authorizing this application.

実際に運用している
botの例

PythonからAPIを使う

- Keys and Access Tokensタブで、「Consumer Key」「Consumer Secret」を確認し控える
- 下の方の「Token Actions」のところの「Create my access token」を押して、出てくる「Access Token」「Access Token Secret」を確認し控える

13

PythonからAPIを使う

compro-reminder Test OAuth

[Details](#) [Settings](#) **Keys and Access Tokens** [Permissions](#)

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key) [REDACTED]

Consumer Secret (API Secret) [REDACTED]

Access Level: Read and write (modify app permissions)

Owner: ComproReminder

Owner ID: 904250273379491841

Application Actions

[Regenerate Consumer Key and Secret](#) [Change App Permissions](#)

14

PythonからAPIを使う

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access tok

Access Token	[REDACTED]
Access Token Secret	[REDACTED]
Access Level	Read and write
Owner	ComproReminder
Owner ID	904250273379491841

- 控えておきます

15

pythonスクリプトで
Tweetさせる

16

pythonスクリプトで Tweetさせる

```
import tweepy

consumer_key = "#####"
consumer_secret = "#####"
token = "#####"
token_secret = "#####"

```

- ##### は各自のものに書き換えてください

17

pythonスクリプトで Tweetさせる

- 認証情報をひとまとめにしたデータ構造を返す関数を宣言しておきます

```
def Twitter_OAuth():
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(token, token_secret)
    return auth

```

18

pythonスクリプトで Tweetさせる

- これをTwitter上のデータと照合させるとAPIが利用できるようになります

```
# 認証情報を照合し、APIを利用可能にする
Twitter_api = tweepy.API(Twitter_OAuth())

# 文字列 test をTweetする関数
# (Twitter_apiの属性)
Twitter_api.update_status(status="test")
```

19

pythonスクリプトで Tweetさせる

- このプログラムを実行して、Tweetできるか確認してみてください
- 公式サイト上でもコードを公開しています

20

pythonスクリプトで Tweetさせる

- プログラムとしては、たったの20行程度です！
- 他にもいろんなことをさせられます
- 例えば…
 - タイムラインを取得する
 - 新規フォロワーを自動でフォローする
- tweepyのDocumentationを読みましょう
- <http://tweepy.readthedocs.io/en/v3.5.0/>

21

pythonスクリプトで Tweetさせる

- タイムラインのTweetを取得し表示

```
for i in Twitter_api.home_timeline():  
    print(i.text)
```

22

pythonスクリプトで Tweetさせる

- `home_timeline()` で返ってくるのは `Status` というデータ構造のlistです
- `Status` 1つに1Tweetの情報が入っているようです
- `Status` にはさまざまなアトリビュートがあり、`text` はそのTweet本文です

23

pythonスクリプトで Tweetさせる

- `Status` にはどんなアトリビュート(属性)があるの？
- `dir()` を使うと持っているアトリビュートがわかります

```
print(dir(Twitter_api.home_timeline()[0]))
```

24

pythonスクリプトで Tweetさせる

- 一覧にしている人がいました
- <https://qiita.com/Ryo87/items/61b5d54cbfd7ae520fe6>
- いろいろ遊んでみよう

25

BeautifulSoupを
使いこなす

26

BeautifulSoupを 使いこなす

- さて、もう少しWebスクレイピングを勉強してみましょう
- BeautifulSoupは日本語版ドキュメントがあります！
- <http://kondou.com/BS4/>

27

BeautifulSoupを 使いこなす

```
import bs4
import urllib.request

html = urllib.request.urlopen("http://www.jma.go.jp/jp/yoho/312.html")
bsObj = bs4.BeautifulSoup(html, "html.parser")

table = bsObj.find_all("table", {"class": "forecast"})[0]
rows = table.find_all("tr")
```

- 前回使ったスクリプトです

28

BeautifulSoupを 使いこなす

```
import bs4
import urllib.request

html = urllib.request.urlopen("http://www.jma.go.jp/jp/yoho/312.html")
bsObj = bs4.BeautifulSoup(html, "html.parser")

table = bsObj.find_all("table", {"class": "forecast"})[0]
rows = table.find_all("tr")
```

- urlopenでは、与えられたURLのHTMLファイルをオブジェクトとして返します（そのままじゃ読めない）

29

BeautifulSoupを 使いこなす

```
import bs4
import urllib.request

html = urllib.request.urlopen("http://www.jma.go.jp/jp/yoho/312.html")
bsObj = bs4.BeautifulSoup(html, "html.parser")

table = bsObj.find_all("table", {"class": "forecast"})[0]
rows = table.find_all("tr")
```

- HTMLパーサー（構文解析器）を用いることで読めるようになります
- それがbs4.BeautifulSoup()関数

30

BeautifulSoupを 使いこなす

```
import bs4
import urllib.request

html = urllib.request.urlopen("http://www.jma.go.jp/jp/yoho/312.html")
bsObj = bs4.BeautifulSoup(html, "html.parser")

table = bsObj.find_all("table", {"class": "forecast"})[0]
rows = table.find_all("tr")
```

- bsObjをprintすると、HTMLのコードがまるっと出てきます
- これをフィルタリングして、必要な情報を取り出していきます

31

BeautifulSoupを 使いこなす

```
import bs4
import urllib.request

html = urllib.request.urlopen("http://www.jma.go.jp/jp/yoho/312.html")
bsObj = bs4.BeautifulSoup(html, "html.parser")

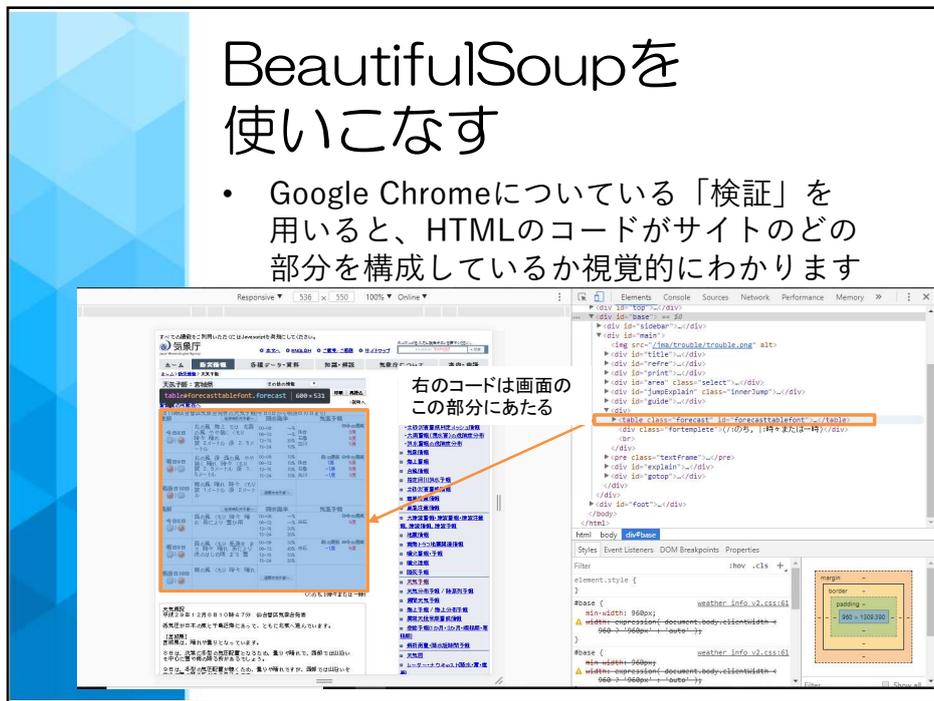
table = bsObj.find_all("table", {"class": "forecast"})[0]
rows = table.find_all("tr")
```

- find_all()関数(findAllでも可)はフィルタリングを行う関数です
- 引数で指定されたタグに囲まれた要素を取り出します

32

BeautifulSoupを 使いこなす

- Google Chromeについている「検証」を用いると、HTMLのコードがサイトのどの部分を構成しているか視覚的にわかります



BeautifulSoupを 使いこなす

```
import bs4
import urllib.request

html = urllib.request.urlopen("http://www.jma.go.jp/jp/yoho/312.html")
bsObj = bs4.BeautifulSoup(html, "html.parser")

table = bsObj.find_all("table", {"class": "forecast"})[0]
rows = table.find_all("tr")
```

- 天気予報の情報はforecastという名前(class)が付いたtableに囲まれているようです

BeautifulSoupを 使いこなす

```
table = bsObj.find_all("table", {"class": "forecast"})[0]
rows = table.find_all("tr")

# または
table = bsObj.find_all("table", class_="forecast")[0]
rows = table.find_all("tr")
```

- Documentationによると、class名を指定する際は{}でdict型として渡す
- または、class_="" の形で渡します(このような引数をキーワード引数という)

35

BeautifulSoupを 使いこなす

```
table = bsObj.find_all("table", {"class": "forecast"})[0]
rows = table.find_all("tr")

# または
table = bsObj.find_all("table", class_="forecast")[0]
rows = table.find_all("tr")
```

- find_allは見つかったものすべてをlistとして返します
- 最初に見つかったforecast tableのみ使うので、[0]をつけて値を取り出します

36

BeautifulSoupを 使いこなす

```
table = bsObj.find_all("table", {"class": "forecast"})[0]
rows = table.find_all("tr")

# または
table = bsObj.find_all("table", class_="forecast")[0]
rows = table.find_all("tr")
```

- table内では tr というタグが行を表しています
- これを取り出しましょう

37

BeautifulSoupを 使いこなす

```
for row in rows:
    ls = []
    for cell in row.find_all(['td', 'th']):
        text = cell.get_text()
```

- ls には各行に詰まった要素(からゴミを取り除いたもの)を格納することになります
- tr タグ内には更にtd, thタグがあり、その中に表の中身が入っています

38

BeautifulSoupを 使いこなす

```
for row in rows:
    ls = []
    for cell in row.find_all(['td', 'th']):
        text = cell.get_text()
```

- for文を使って表の各行に対し、中身を取り出していきます
- get_text()はタグを取り除いた本文を取り出す関数です

39

BeautifulSoupを 使いこなす

```
# 改行コードやタブの削除
text = text.replace('\n', '')
text = text.replace('¥u3000', '')
text = text.replace('¥t', '')
text = text.replace('¥xa0', '')
if(text):
    ls.append(text)
```

- 改行コードなどを削除します
- text が "" でなければ ls に追加 (空文字列 "" はFalseとみなされます)

40

BeautifulSoupを 使いこなす

- なんとなく取り出し方がわかった
でしょうか？
- ブラウザについている分析ツールは
非常に有用なので活用してみ
てください

41

天気予報をTweetさせる

42

天気予報をTweetさせる

- これらを組み合わせれば、天気予報を取得しTweetさせることが可能になります！

43

やってみよう

- 天気予報を取得しTweetするスクリプトを書いてみましょう
- 天気予報のスクレイピングはサンプルコードをうまく活用(必要な情報のみ取り出す)

44

やってみよう

- Tweetするテキストを作成する際は、

```
text = "今日の仙台の天気:" + tx1 + "¥n最高気温:"  
      + tx2 + "¥n最低気温:" + tx3
```

のように足し算してtextをTweetさせる

- または、

```
text = "今日の仙台の天気:{0}¥n最高気温:{1}¥n最低気温:{2}".format(tx1, tx2, tx3)
```

のようにformat()関数を使うと良いです
(tx1, tx2, tx3はそれぞれ文字列変数)

45

最後に

46

最後に

- プログラミングがある程度できるようになるためには、
 - まずはいろいろ書いてみる
 - エラーやDocumentationを自分で読むことが大事です
- よほどのことがないとPCは壊れないので、失敗を恐れずいろいろ書いてみましょう

47

最後に

- なにをやったらいいかわからない人
⇒ 競技プログラミングを
やってみましょう
- プログラミングの技術が身につき、作りたいものが見つかったときに役立ちます

48

最後に

- なにをやったらいいかわからない人
⇒ 競技プログラミングを
やってみましょう
- プログラミングの技術が身につき、作
りたいものが見つかったときに役立ち
ます

49

宣伝！

- 自由にプログラミングをする
サークルです
- 勉強会の主催や競技大会(ICPC,
CODE FESTIVAL)に向けた活動
などを行っています
- 新入部員募集中！



50

