



P E R S O L

パーソル R&D

Python中級講座

様々な領域で使用されているpythonについて、スクレイピングを軸に諸々紹介する講座です。

はじめに

本講習ではpythonを扱います。

python言語はコードを、シンプルかつ直感的、かつ短く表現することを目指しており、これに沿ったコードを **pythonicだ**(pythonらしいコーディングだ)と表現しているほどです。

このpythonicな文化が好まれて、pythonには多様なライブラリ、フレームワーク、ラッパーが存在します。大抵のやりたいことに対して、pythonで道具そのものが提供されたり、道具を使いやすく包んだラッパーが提供されています。(※)

本講習ではこのpythonの充実したライブラリ群の一部を紹介します。

(※)リアルタイム処理等、常にpythonを使うことが最適解とは限らない。

はじめに

pythonライブラリ、フレームワーク、ラッパー例

数学演算系(numpy)

機械学習(pytorch, scikit-learn)

ロボットの制御フレームワーク(ros)

組み込み系の通信制御(シリアル通信、USB通信等)

ゲーム開発 (pygame)

クラウドAPI (google/twiter等)

データベース (sqlite3/mysql/sqlserver等)

データ集計 (pandas)

スクレイピング(beautifulsoup/scrapy)

グラフ描画(matplotlib)

サーバサイドWeb(flask)

デスクトップUI(qt)

講座で紹介する機能

本講習では「データ分析」の現場で使用されている技術「スクレイピング」を中心に、以下の機能を紹介します。

スクレイピング

スクレイピングとはWebサイトを解析して、自分の求めるデータを収集することです。主な作業は「HTML構造の解析」と「テキストの抽出」です。

beautifulsoupというライブラリを使用します。簡単にスクレイピングできるscrapyというフレームワークもありますが、スクレイピングの本質を理解するにはこちらが良いです。

クラウドAPI

YoutubeにはYoutubAPIが、TwitterにはTwitterAPIが存在し、いずれも「自社サービスへの協力」のために公開されているサーバ機能です。人気動画や、フォロー/リツイート等を様々なサイトに埋め込んで宣伝してもらうことが主目的です。

講座で紹介する機能

これらのクラウドAPIでサイトから様々な情報を収集できます。

twitterの各アカウントもスクレイピングによってプロフィール、ツイート等を収集できますが、そもそもtwitter社がスクレイピングを禁止しています。
(過去ツイートをきりなくスクレイピングされるとサーバ負荷があがるため)

このような「スクレイピングできないサイト」のためにクラウドAPIの使用例を紹介します。

データ保存

収集データは一旦、データベースやyamlファイルの形式で保存します。
データはそのままでは数字の羅列に過ぎず、統計を取ったり、グラフ表示するためには一旦保存するのが望ましいからです。

pythonはsqlite3, mysql, sqlserver, mongodb等様々なデータベースを扱えますが、クライアント・サーバシステムでなく、アプリに組み込める軽量データベースsqlite3を使用します。

データ集計

収集データを「このカラムの合計出して、平均値出して・・・」とかできますが、そもそも、そういったことを自動でやってくれるデータ集計ライブラリpandasがあるのでそちらを使います。

pandasは辞書形式フォーマット(json, yaml)、csv、データベースファイル等の様々なデータストリームを取り込み、いろいろ集計してくれます。

集計結果プロット(グラフ描画)

解析したデータを視覚化すると、より一層データの傾向が掴みやすくなります。データをプロット(グラフ描画)するため、よく使用されるmatplotlibを使用します。

python文字列表記

pythonで文字列を扱う際、

```
print('hello, world')  
print( "hello, world")
```

上記のようにシングルクォート、ダブルクォートのどちらも使えますが、英語キーボードでは、シングルクォートとダブルクォートは同じキーに割り当てられていて、ダブルクォートはシフト有りなので、アメリカではシングルクォートを使う人が多い、という傾向があるらしいので、本講習ではシングルクォートを使っていきます。

シングルクォートは数字の「7」キー + シフトキーです。

Python構文：正規表現

正規表現

「特定のパターンに一致する文字列」を発見、抽出する機能。
スクレイピングでは必須機能の一つ。Webページから抽出したテキストの中で、実際に欲しいパラメータ、数値を取り出すために使う。

(例)

```
import re                # 正規表現ライブラリ
age = '永遠の17歳'
re.search('[0-9]{1,5}歳', age)
```

実行結果

```
<_sre.SRE_Match object; span=(3, 6), match='17歳'>
```

正規表現の説明

[0-9] … 0から9までの数値を使用している。

{1, 5} … 上記で指定した0-9までの数値が1～5個連続している。

歳…上記の後に文字「歳」が続く

課題 1

以下の文字列から「身長」を正規表現を使って抜き出してください。

彼の体重は50kgだが、身長は180cmもある。

ターミナルで以下のように入力し、'xxx'の部分を上手く埋めて身長のみ抽出してください。

```
>>>import re  
>>>text = '彼の体重は50kgだが、身長は180cmもある。'  
>>>re.search('xxx', text)
```

課題 1 : 回答

```
>>>import re  
>>>text = '彼の体重は50kgだが、身長は180cmもある。'  
>>>re.search('[0-9]{3}cm', text)
```

Python構文：リスト内包表記

python(pythonicな文化)では「繰り返し処理をなるべく直感的、かつ、短く」記述しようとしています。このために**リスト内包表記**が用意されています。

例えば以下のforループがあったとします。

```
for vtuber in vtuber_list:  
    print(vtuber['name'])
```

vtuber_listという配列（Vtuberの名前格納）があり、配列から一個ずつデータを取り出して出力しています。

これがリスト内包表記を用いると、以下のようになります。

```
[print(vtuber['name']) for vtuber in vtuber_list]
```

Python構文：リスト内包表記

```
[print(vtuber['name']) for vtuber in vtuber_list]
```

[]はプログラム言語一般の処理として配列に使うものですが、
「配列 = 繰り返しデータが格納されている」ということから転じて、
「配列 = 繰り返し**処理ができる**」という解釈がpythonでは実装されています。

リスト内包表記を応用すると、「年齢が17歳未満だけ」という条件付けも
1行で記述できます。

```
[print(vtuber['name']) for vtuber in vtuber_list if vtuber['age'] < 17]
```

Python構文：特殊メソッド

pythonの特殊メソッドはpythonの標準ライブラリと組み合わせて使う、便利なメソッドです。

`__str__`

定義されたクラス変数を文字列のように扱える特殊メソッド。

(定義例)

```
class A(object):  
    def __init__(self):  
        self._name = "mano takashi"  
    def __str__(self):  
        return self._name
```

(活用例)

```
>a = A()  
>print(a)  
"mano takashi"
```

Aクラスのインスタンスが「どんな情報を保持しているのか」簡単にdumpできます。

Python構文：特殊メソッド

`__iter__`

イテレータを定義する。定義されたクラス変数を配列のように扱える特殊メソッド。

(定義例)

```
class A(object):  
    def __init__(self):  
        no_list = [1, 2, 3]  
  
    def __iter__(self):  
        for no in no_list:  
            yield no  
        else:  
            raise StopIteration()
```

(活用例)

```
a = A()  
for no in a:    <= for in 構文を使って、Aクラス内の配列を一つずつ処理できる。  
    print(a)
```

日本語処理(文字コード)

スクレイピングの際には、pythonにおける日本語処理(文字コード)について少し知っておくと良いです。文字コード処理は大抵ライブラリが吸収してくれますが、そうでないライブラリを使わざるを得ないケースもあります。

Unicode

python3の文字列型(str)はUnicodeで実装されています。文字列 = Unicodeです。

つまり、外部から渡される文字列がutf-8等の文字コードであれば、それをUnicodeに変換する必要があります。

(文字列変数の例)

```
a = 'あ'
```

aという変数は文字列型(str)であり、Unicodeで実装されています。

日本語処理(文字コード)

unicodeは「世界中の文字をそれぞれ唯一の番号で採番してしまおう」という試みです。

(例)

'a' = 97

'あ' = 12354

'ㇿ' = 950

上記は「コードポイント」と呼ばれるunicodeが各文字に採番した番号です。
pythonでは、ord('a')のように、ライブラリ関数で取得できます。

一方、最近の文字コード標準となっているutf-8は以下のように表現されています。

'a' = 0x61

← asciiコードはそのまま

'あ' = 0xe3 0x81 0x82

← 日本語コードは3バイトで表現。

渡される文字列がutf-8形式であれば、unicodeに変換する必要があります。

utf8_text.decode('utf-8') ← utf-8形式のコードをデコード(unicode形式に変換)。

日本語処理(文字コード)

URLエンコードについて

WebページのURLにはURLエンコードという手法が使用されています。
従って、ブラウザのURL欄には以下のように表示されていても、



URLをコピーしてpythonプログラム中にペーストすると以下のようになります。

<https://wikiwiki.jp/nijisanji/%E9%88%B4%E9%B9%BF%E8%A9%A9%E5%AD%90>

この現象は、URLがマルチバイト文字(日本語等)を直接扱えないために発生します。

日本語処理(文字コード)

「鈴鹿詩子」は、utf-8において以下のコード列で表現されます。

E9 88 B4 E9 B9 BF E8 A9 A9 E5 AD 90

URLエンコードは自身では文字コードを定義していないので、渡されたコード列の並びに% を付与してマルチバイト文字を表現します。

%E9 %88 %B4 %E9 ...

URLはこのURLエンコードで表現されますが、ブラウザで「鈴鹿詩子」と表示されるのは、ブラウザが画面描画時にこのURLエンコードをデコード(utf-8の文字に変換)しているからです。

スクレイピング : サンプル 1

実際にサンプル用のhtmlを使って、スクレイピングをしていきます。

VTuberとは？

バーチャルYouTuber(バーチャルユーチューバー)は
YouTuberとして動画配信・投稿を行う
コンピュータグラフィックスの日本発祥のキャラクター(アバター)、
またキャラクター(アバター)を用いて動画投稿・配信を行う人。

別名 : VTuber、Vチューバー (ブイチューバー) 。

[VTuber Wikiページ](#)

スクレイピング : サンプル 1

Webページの構造

スクレイピング : タグの参照と、テキストの抽出

タイトル(title)

VTuberとは？

見出し(h1)

バーチャルYouTuber(バーチャルユーチューバー)は
YouTuberとして動画配信・投稿を行う
コンピュータグラフィックスの日本発祥のキャラクター(アバター)、
またキャラクター(アバター)を用いて動画投稿・配信を行う人。

段落 (p)

別名 : VTuber、Vチューバー (ブイチューバー) 。

強調 (b)

[VTuber Wikiページ](#)

リンク (a)

スクレイピング : サンプル 1

ターミナルで以下のように実行してください。

```
>>>from bs4 import BeautifulSoup
>>>f=open('sample1.html', encoding='utf-8')
>>>bs = BeautifulSoup(f.read(), 'html.parser')
>>>bs
```

コードの説明

Webページの中身(sample.html)を渡してBeautifulSoup(以降BS)のインスタンスを生成すると、**Webページの解析が完了**する。

BSのインスタンスを使って、**HTMLのタグ構造にアクセス**できる。
上記例では解析したHTML全体にアクセスしている。

スクレイピング : サンプル 1

```
<html>
<head>
<title>スクレイピング : タグの参照と、テキストの抽出</title>
</head>
<body>
<h1>VTuberとは ? </h1>
<p>
バーチャルYouTuber(バーチャルユーチューバー)は<br/>
YouTuberとして動画配信・投稿を行う<br/>
コンピュータグラフィックスの日本発祥のキャラクター(アバター)、 <br/>
またキャラクター(アバター)を用いて動画投稿・配信を行う人。<br/>
</p>
<b>別名 : VTuber、Vチューバー (ブイチューバー) 。</b>
<br/><br/>
<a href="https://ja.wikipedia.org/wiki/バーチャルYouTuber">VTuber Wikiページ</a>
</body>
</html>
```

スクレイピング：サンプル1

(1) HTMLタグ(タイトル)の参照

```
>>>bs.title
```

```
<title>スクレイピング：タグの参照と、テキストの抽出</title>
```

BSでは**メンバ変数**でHTML内の各タグにアクセスできます。

(2) テキスト(タイトル)の抽出

```
>>>bs.title.text
```

```
'スクレイピング：タグの参照と、テキストの抽出'
```

BSでは、textプロパティでタグ内のテキストのみ、抽出できます。

(3) HTMLタグ属性(リンク)の参照

```
>>>bs.a['href']
```

```
'https://ja.wikipedia.org/wiki/バーチャルYouTuber'
```

タグの属性(等)は、各タグに辞書形式で格納されているため、a['href']のようにアクセスできます。

課題 1

前ページを参考に、ページ内の見出し(h1)、強調表示(b)、段落(p)について、タグ参照、テキスト抽出を実行してみてください。

課題 1 : 回答

見出し(h1)

```
>>> bs.h1
```

```
<h1>VTuberとは? </h1>
```

```
>>> bs.h1.text
```

```
'VTuberとは?'
```

強調(b)

```
>>> bs.b
```

```
<b>別名 : VTuber、Vチューバー（ブイチューバー）。</b>
```

```
>>> bs.b.text
```

```
'別名 : VTuber、Vチューバー（ブイチューバー）。'
```

課題 1 : 回答

段落(p)

>>> bs.p

<p>

バーチャルYouTuber（バーチャルユーチューバー）は、YouTuberとして動画配信・投稿を行う

コンピュータグラフィックスの日本発祥のキャラクター（アバター）、

またキャラクター（アバター）を用いて動画投稿・配信を行う人。

</p>

>>> bs.p.text

'¥nバーチャルYouTuber（バーチャルユーチューバー）は、YouTuberとして動画配信・投稿を行う¥nコンピュータグラフィックスの日本発祥のキャラクター（アバター）、¥nまたキャラクター（アバター）を用いて動画投稿・配信を行う人。¥n'

<p>タグが取り払われ、HTMLにおける改行タグである
は通常の改行コード(¥n)に変換されます。

スクレイピング：サンプル2

今度はテーブルを含んだサンプルをスクレイピングしてみます。

VTuber:キズナアイ

プロフィール

年齢	16歳
身長	156cm
体重	46kg

テーブル(table)

スクレイピング : サンプル2

ターミナルで以下のように実行してください。

```
>>>f=open('sample2.html', encoding='utf-8')
>>>bs = BeautifulSoup(f.read(), 'html.parser')
>>>bs
```

...

```
<table bgcolor="#e3f0fb" border="1" style="border-collapse: collapse"
width="300px">
```

```
<caption><b>プロフィール</b></caption>
```

```
<tbody>
```

```
<tr><th>年齢</th><td>16歳</td></tr>
```

```
<tr><th>身長</th><td>156cm</td></tr>
```

```
<tr><th>体重</th><td>46kg</td></tr>
```

```
</tbody>
```

```
</table>
```

...

スクレイピング : サンプル2

```
<table bgcolor=…>  
<caption><b>プロフィール</b></caption>  
<tbody>  
<tr><th>年齢</th><td>16歳</td></tr>  
<tr><th>身長</th><td>156cm</td></tr>  
<tr><th>体重</th><td>46kg</td></tr>  
</tbody>  
</table>
```

テーブルタグ(**table**)の中に、三つのレコード(**tr**)が格納されています。
レコードはヘッダ(**th**)とデータ(**td**)から構成されています。

スクレイピング : サンプル2

(1) HTMLタグ(テーブル)の参照

```
>>> bs.table
```

```
<table ...>
```

```
<caption><b>プロフィール</b></caption>
```

```
<tbody>
```

```
<tr><th>年齢</th><td>16歳</td></tr>
```

```
<tr><th>身長</th><td>156cm</td></tr>
```

```
<tr><th>体重</th><td>46kg</td></tr>
```

```
</tbody>
```

```
</table>
```

(2) テキスト(テーブル)の抽出

```
>>> bs.table.text
```

```
'¥nプロフィール¥n¥n年齢16歳¥n身長156cm¥n体重46kg¥n¥n'
```

テーブルからテキストを抽出すると、BSがテーブル構造をなぞってテキストに連結してくれます。これを文字列処理して必要データを抽出することもOKです。

スクレイピング : サンプル2

(3) HTMLタグ(レコード) の参照

```
>>> bs.table.tr
```

```
<tr><th>年齢</th><td>16歳</td></tr>
```

テーブルの**最初のレコード**を返します。最初のレコードに年齢データが格納されていることがあらかじめわかっているなら、

```
>>> bs.table.tr.td.text
```

```
'16歳'
```

このように、直接VTuberの年齢をスクレイピングできます。
では身長データ(二つ目のレコード)にはどうやってアクセスできるのか？

```
<tr><th>年齢</th><td>16歳</td></tr>
```

```
<tr><th>身長</th><td>156cm</td></tr> ← こちらにはどうアクセス？
```

```
<tr><th>体重</th><td>46kg</td></tr>
```

スクレイピング : サンプル2

(4) HTMLタグ(レコード) の検索

```
>>> bs.table.find_all('tr')
```

```
[<tr><th>年齢</th><td>16歳</td></tr>,  
<tr><th>身長</th><td>156cm</td></tr>,  
<tr><th>体重</th><td>46kg</td></tr>]
```

find_all('タグ名')により、親タグ(ここではtable)下にあるすべてのタグを配列として返します。

従って以下の記述で直接、身長データをスクレイピングできます。

```
>>> bs.table.find_all('tr')[1].td.text  
'156cm'
```

find_allで返ってきた配列の**2番目**に身長レコードが格納され、その**td**タグのテキストを抽出すると身長データです。

課題 2

前ページを参考に、キズナアイの体重データをスクレイピングしてください。

課題 2 : 回答

```
>>> bs.table.find_all('tr')[2].td.text  
'46kg'
```

課題 3

同じく、キズナアイの体重データをスクレイピングしてください。
ただし、体重データがテーブルのどこに格納されているかはわからないもの、
としてください。
([2]のように、配列にベタアクセスするのはNG)

課題3：回答

```
>>> [tr for tr in bs.table.find_all('tr') if '体重' in tr.th.text][0].td.text  
'46kg'
```

リスト内包表記を使って、条件に一致するテーブルを探しています。
ここではレコードのヘッダに「体重」という文字列を含むものを探しています。

リスト内包表記の結果、（今回はヘッダに体重とあるのは一件だけなので）、
配列の先頭が目的のレコードであり、そのデータ(td)のテキストを抽出します。

課題 4

findを使わずに、キズナアイの体重データをスクレイピングしてください。

課題4：回答

```
>>> import re
>>> re.search('[0-9]{2}kg', bs.table.text)
<re.Match object; span=(25, 29), match='46kg'>
```

前項で説明したように、BSによりテーブル全体のテキストが取得できます。
'¥nプロフィール¥n¥n年齢16歳¥n身長156cm¥n体重**46kg**¥n¥n'

ここから**正規表現**により体重データを抽出します。

因みにre.searchの結果は配列なので、以下のようにアクセスできます。
>>> re.search('[0-9]{2}kg', bs.table.text)[0]
'46kg'

スクレイピング：サンプル3

以下のように、二つ目のテーブルにだけアクセスしたい場合はどうすればよいのでしょうか？

VTuber:キズナアイ

プロフィール

年齢	16歳
身長	156cm
体重	46kg

所属事務所

名称	Activ8株式会社
プロジェクト名	upd8(アップデート)

テーブル(table)

スクレイピング：サンプル3

コンテナの利用

VTuber:キズナアイ

プロフィール

年齢	16歳
身長	156cm
体重	46kg

→ コンテナ 1

所属事務所

名称	Activ8株式会社
プロジェクト名	upd8(アップデート)

→ コンテナ 2

Webページでは、「この区画にはこういう情報が入ってるんだよ」ということを示す、コンテナ(divタグ) という目に見えない箱があります。

このコンテナの名前等で、目的のテーブルにだけアクセスすることができます。

スクレイピング : サンプル3

ターミナルで以下のように実行してください。

```
>>>f=open('sample3.html', encoding='utf-8')  
>>>bs = BeautifulSoup(f.read(), 'html.parser')  
>>>bs
```

スクレイピング : サンプル3

```
<div class="profile">
<table bgcolor=…>
<caption><b>プロフィール</b></caption>
<tbody>
<tr><th>年齢</th><td>16歳</td></tr>
<tr><th>身長</th><td>156cm</td></tr>
<tr><th>体重</th><td>46kg</td></tr>
</tbody>
</table>
</div>
<div class="office">
<table bgcolor=…>
<caption><b>所属事務所</b></caption>
<tbody>
<tr><th>名称</th><td>Activ8株式会社</td></tr>
<tr><th>プロジェクト名</th><td>upd8(アップデート)</td></tr>
</tbody>
</table>
</div>
```

スクレイピング : サンプル3

以下のように二つのテーブルが表示されています。

```
<div class="profile">
```

一つ目のテーブル

```
</div>
```

```
<div class="office">
```

二つ目のテーブル

```
</div>
```

アクセスしたいテーブルは、**属性class**が"office"であるコンテナに格納されており、**属性検索**を使ってアクセスできます。

```
>>>>> bs.find('div', {'class':'office'}) ← class属性を指定してdivタグ検索
```

```
<div class="office">
```

...

```
<tr><th>名称</th><td>Activ8株式会社</td></tr>
```

```
<tr><th>プロジェクト名</th><td>upd8(アップデート)</td></tr>
```

...

```
</div>
```

スクレイピング : サンプル3

従って、キズナアイの所属会社は以下のように直接スクレイピングできます。

```
>>> bs.find('div', {'class':'office'}).table.td.text  
'Activ8株式会社'
```

課題 5

キズナアイの所属プロジェクトをスクレイピングしてください。

VTuber:キズナアイ

プロフィール

年齢	16歳
身長	156cm
体重	46kg

所属事務所

名称	Activ8株式会社
プロジェクト名	upd8(アップデート)

課題 5 : 回答

```
>>> bs.find('div', {'class':'office'}).table.find_all('tr')[1].td.text  
'upd8(アップデート)'
```

サンプル2で触れたように、テーブル内には複数のレコードが存在するため、find_allで探し出して、プロジェクト名が格納されている二つ目のレコードにアクセスします。

課題 6

コンテナが存在しない、かつ、二つ目のテーブルにプロジェクトがあると分かっていない場合、どのようにプロジェクトをスクレイピングすればよいでしょうか？

VTuber:キズナアイ

プロフィール

年齢	16歳
身長	156cm
体重	46kg

所属事務所

名称	Activ8株式会社
プロジェクト名	upd8(アップデート)

目的のテーブルを識別するものが
左の画面にあるので、それを使ってください。

課題6：回答

各テーブルにはテーブルのタイトル(caption)が格納されています。
まず、そのタイトルが一致するテーブルを探し出します。




```
>>> target_table = [table for table in bs.find_all('table') if '事務所' in  
table.caption.text]
```

後は今まで同様に、見つけたテーブルから「体重」データを抽出します。

```
>>> [tr for tr in target_table[0].find_all('tr') if 'プロジェクト' in  
tr.th.text][0].td.text  
'upd8(アップデート)'
```


スクレイピング : VTuberランキング

今まで学んだことを使って、Vtuberのランキングサイトから
Vtuber名と順位、そして詳細ページへのリンクをスクレイピングしてみます。

<div> <div>ランキング</div> <div>ライブ・動画</div> <div>事務所別</div> <div>VTuberを検索</div> </div>			
<div> <div>急上昇</div> <div>注目の新人</div> <div>人気の動画</div> <div>ファン数順</div> <div>SHOWROOM</div> <div>TikTok</div> <div>bilibili</div> <div>ランキング掲載申請</div> </div>			
ファン数	ファン数急上昇	総再生回数	総再生回数急
<div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>...</div> </div>			
	1位 キズナアイ A.I.Channel 8 up d8 [順位をツイート]	2,690,000 人 +476 280,415,610 回 +188,351	2016年から活動を開始。はじめYouTuberを自称した。親分訪日促進アンバサダー就任。し [応援メッセージ]
	2位 キズナアイ(ゲーム) A.I.Games 8 up d8 [順位をツイート]	1,420,000 人 +434 138,902,327 回 +101,054	キズナアイのゲーム専門チャ [応援メッセージ]
	3位 輝夜月(かぐやるな) Kaguya Luna Official ✓ VIC [順位をツイート]	999,000 人 -83 98,727,734 回 +20,447	2017年の12月に開始し、話題Mika Pikazo氏デザイン。首絞見るストロングゼロといった [応援メッセージ]

スクレイピング : Vtuberランキング : ページ構造

1	2	4	1位 ナアイ A.I.Channel 8 upd8 [順位をツイート]
3			2位 キズナアイ(ゲーム) A.I.Games 8 upd8 [順位をツイート]
			3位 輝夜月(かぐやるな) Kaguya Luna Official ✓ VIC [順位をツイート]
			4位 ミライアカリ Mira Akari Project 元ENTUM [順位をツイート]
			5位 电脑少女シロ Siro Channel LIVE

```
▼<table class="table table-sm table-hover table-  
common table-ranking table-ranking-yt"> == $0
```

```
▼<tbody>
```

```
▼<tr data-href="/user/  
D780B63C2DEBA9A2_fa95ae">
```

```
▼<td>
```

```
▼<a href="/user/D780B63C2DEBA9A2_fa95ae"  
class="no-propagation">
```

```

```

```
</a>
```

```
</td>
```

```
▼<td class="col-name">
```

```
▼<div>
```

```
<strong>1位</strong>
```

スクレイピング : Vtuberランキング

まずは解析です。

```
>>>import requests  
>>>html = requests.get('https://virtual-  
youtuber.userlocal.jp/document/ranking')  
>>>bs = BeautifulSoup(html.text, 'html.parser')
```

これで準備ができました。

スクレイピング : Vtuberランキング

名前のスクレイピング

テーブル(class属性 = table-ranking-yt) → レコード[0] → aタグ → imgタグ:alt属性

```
>>> bs.find('table', {'class': 'table-ranking-yt'}).find_all('tr')[0].a.img['alt']  
'キズナアイ'
```

順位のスクレイピング

テーブル(class属性 = table-ranking-yt) → レコード[0] → strongタグ

```
>>> bs.find('table', {'class': 'table-ranking-yt'}).find_all('tr')[0].strong.text  
'1位'
```

詳細ページのスクレイピング

```
>>> 'https://virtual-youtuber.userlocal.jp' + bs.find('table', {'class': 'table-  
ranking-yt'}).find_all('tr')[0].a['href']
```

スクレイピング : Vtuberランキング

ランキングページ全体からスクレイピング

前ページは先頭のテーブル(キズナアイ)のみスクレイピングしましたが、
以下のようにすることでテーブル内全VTuberの名称をスクレイピングできます。

```
>>> [tr.a.img['alt'] for tr in bs.find('table', {'class': 'table-ranking-  
yt'})].find_all('tr')]
```

```
['キズナアイ', 'キズナアイ(ゲーム)', '輝夜月(かぐやるな)', 'ミライアカリ', '電腦少女シ  
ロ', '田中ヒメ 鈴木ヒナ', '猫宮ひなた', '月ノ美兎', 'ヨメミ エトラ', '白上フブキ', '本間ひ  
まわり', '道明寺ここあ', 'YuNi', 'ゲーム部プロジェクト', '夏実萌恵', '笹木咲', '湊あくあ',  
'シフィ(シフィール・エシラー) 旧ベイレーン', '御伽原江良(おとぎばらえら)', '椎名唯華',  
'ときのそら', '樋口楓', '静凜', '花譜(かふ)', '富士葵(ふじあおい)', 'キミノ ミヤ(MIYA  
KIMINO)', '夢月ロア(ゆづきろあ)', 'リゼ・ヘルエスタ', '緑仙(リ्यूシェン)', '鈴鹿詩  
子', 'バーチャルおばあちゃん(VB)', '葛葉', '鈴原るる', 'にじさんじ公式', '犬山たまき',  
'ハロー キティ', 'アンジュ・カトリーナ', '周防パトラ', '天神子兔音(てんじんことね)',  
'マヤ・プトゥリ', '織田信姫', '夏色まつり', 'アルス・アルマル', 'バーチャル番組連盟', '赤  
井はあと', '馬犬', 'ベルモンド・バンデラス', '白銀ノエル(しろがねのえる)', 'ホロライブ  
公式', '竜胆尊(りんどうみこと)']
```

スクレイピング : Vtuberランキング

ランキングページ全体からスクレイピング

従って、ランキングページから「vtuber名、順位」のセットを以下のように抽出できます。

```
>>> vtubers = [tr for tr in bs.find('table', {'class': 'table-ranking-  
yt'})].find_all('tr')]
```

```
>>> profiles = []
```

```
>>> for vtuber in vtubers:
```


```
...     profiles.append((vtuber.a.img['alt'], vtuber.strong.text))
```

```
>>> params
```

```
[('キズナアイ', '1位'), ('キズナアイ(ゲーム)', '2位'), ('輝夜月(かぐやるな)', '3位'), ('ミ  
ライアカリ', '4位'), ('電腦少女シロ', '5位'), ('田中ヒメ 鈴木ヒナ', '6位'), ...]
```

スクレイピング : Vtuber詳細

Vtuberランキングページの各Vtuberをクリックすると、Vtuber詳細ページへジャンプします。このページから、所属オフィスと配信力(ファン数、総再生回数)をスクレイピングします。



ナンジャモンジャをイラスト描いて遊ん...

ナンジャモンジャ

👁 15,611 🕒 約13時間前

A.I.Channel

キズナアイ [ライブ配信予定を登録する](#)


8 upd8

ファン数	269万人
総再生回数	2億8041万5610回


[Twitter](#) @aichan_nel

2016年から活動を開始。はじめてバーチャした。親分と呼ばれる。訪日促進アンバサ属している。


最近の動画



【ASMR】もしもloveちゃんが歴史




学猫叫 Say Meow Meow/covered by



見てたよ。

👁 148,021 🕒 7日前



内緒、だからね。

👁 82,670 🕒 8日前

スクレイピング : Vtuber詳細 : ページ構造 : 所属オフィス



👁 28,073 🕒 約13時間前

A.I.Channel

キズナアイ  ライブ配信予定を登録

8 upd8

ファン数

269万人

総再生回数

2億8057万8251回

 Twitter

@aichan_ne

Elements Console >> 2 33

```
<div class="container">
  <div class="row">
    <div class="col-12 col-lg-6">...</div>
    <div class="col-12 col-lg-6 box-channel-info
      vertical"> == $0
      <div class="vertical">
        <h1 class="channel-subject">A.I.Channel
        </h1>
        <div>...</div>
        <div class="box-office">
          <a href="/document/ranking?office=upd8"
            
            "
            "
            upd8
          </a>
        </div>
```


スクレイピング：Vtuber詳細：所属オフィス

まずは解析です。

```
>>>html = requests.get('https://virtual-  
youtuber.userlocal.jp/user/D780B63C2DEBA9A2_fa95ae')  
>>>bs = BeautifulSoup(html.text, 'html.parser')
```

これで準備ができました。

所属オフィスのスクレイピング

コンテナ(box-office)に直接テキストで格納されています。

```
>>> bs.find('div', {'class': 'box-office'}).text  
'¥n¥n¥n          upd8¥n '
```

stripメソッドを使うと余分な空白や改行コードを取り除いてくれます。

```
>>> bs.find('div', {'class': 'box-office'}).text.strip()  
'upd8'
```

スクレイピング : Vtuber詳細 : ページ構造 : 配信力

にじP 1・2期 にじさんじ(ゲ... にじさんじ



👁 28,073 🕒 約13時間前

A.I.Channel

キズナアイ  ライブ配信予定を登録する

8 upd8

ファン数

269万人

総再生回数

2億8057万8251回

 Twitter

@aichan_nel

```
> <div class="col-12 col-lg-6">...</div>
```

```
▼ <div class="col-12 col-lg-6 box-channel-info
vertical"> == $0
```

```
> <div class="vertical">...</div>
```

```
▼ <div class="vertical my-3">
```

```
▼ <div class="channel-stat my-2">
```

```
<span class="channel-stat-label bg-
success text-white">ファン数</span>
```

```
<span class="channel-stat-value">269万人</span>
```

```
</div>
```

```
▼ <div class="channel-stat my-2">
```

```
<span class="channel-stat-label bg-danger
text-white">総再生回数</span>
```

```
<span class="channel-stat-value">2億8057万
8251回</span>
```

```
</div>
```

```
▼ <div class="channel-stat my-2">
```

```
> <span class="channel-stat-label bg-brand-
tw text-white">...</span>
```

```
▼ <span class="channel-stat-value">
```

```
▼ <span class="text-nowrap">
```

```
<a target="_blank" rel="nofollow
noreferrer" href="https://twitter.com/
```

スクレイピング : Vtuber詳細 : 配信力

ファン数のスクレイピング

コンテナ(box-channel-info)下に、複数のコンテナ('channel-stat')が格納されており、ファン数はそのうちの一つです。

まず、複数コンテナの配列を取得する。

```
>>> bs.find('div', {'class': 'box-channel-info'}).find_all('div', {'class': 'channel-stat'})
```

ファン数はこの配列の一個目なので、

```
>>> bs.find('div', {'class': 'box-channel-info'}).find_all('div', {'class': 'channel-stat'})[0].text
'¥nファン数¥n269万人¥n'
```

改行等、余分なデータを取り払うには、

```
>>> bs.find('div', {'class': 'box-channel-info'}).find_all('div', {'class': 'channel-stat'})[0].text.split('¥n')[2]
'269万人'
```

総再生回数は配列の二個目なので、上記同様にスクレイピングできます。

課題 7

今まで試してきたことを総合して、ランキングページ、詳細ページから以下のようにVtuberプロフィールをスクレイピングしてください。

■ スクレイピング対象

名前、順位、所属オフィス、フォロワー数、総視聴数

■ 出力形式

以下のように配列で出力してください。

```
[('キズナアイ', '1位', 'upd8', '269万人', '2億8368万1807回'),  
 ('キズナアイ(ゲーム)', '2位', 'upd8', '142万人', '1億4051万9110回'),  
 ('輝夜月(かぐやるな)', ...)]
```

課題7 : 回答

```
from bs4 import BeautifulSoup
import requests
html = requests.get('https://virtual-youtuber.userlocal.jp/document/ranking')
bs = BeautifulSoup(html.text, 'html.parser')
vtubers = [tr for tr in bs.find('table', {'class': 'table-ranking-yt'}).find_all('tr')]

profiles = []
for vtuber in vtubers:
    name = vtuber.a.img['alt']
    rank = vtuber.strong.text
    html = requests.get('https://virtual-youtuber.userlocal.jp' + vtuber.a['href'])
    bs2 = BeautifulSoup(html.text, 'html.parser')
    follower = bs2.find('div', {'class': 'box-channel-info'}).find_all('div', {'class': 'channel-
stat'})[0].text.split('¥n')[2]
    view = bs2.find('div', {'class': 'box-channel-info'}).find_all('div', {'class': 'channel-
stat'})[1].text.split('¥n')[2]
    try:
        office = bs2.find('div', {'class': 'box-office'}).text.strip()
    except AttributeError:
        print('scraping error : {}'.format(name))
        office = 'Unknown'
    profiles.append((name, rank, office, follower, view))
```

データベース：実装

スクレイピングした結果はデータベース等に記録します。

テーブル構造

名前：テキスト

所属：テキスト

フォロワー数：整数

総視聴数：整数

```
>>>import sqlite3
>>> c = sqlite3.connect('test.db')
>>> c.execute('CREATE TABLE vtuber(name TEXT NOT NULL UNIQUE, office
TEXT, follower INTEGER, view INTEGER)')
>>> c.execute('INSERT INTO vtuber(name, office, follower, view) VALUES("キズ
ナアイ", "upd8", 2690000, 283505794)')
>>> c.execute('COMMIT')
>>> for v in c.execute('SELECT * FROM vtuber'):
...     print(v)
```

```
('キズナアイ', 'upd8', 2690000, 283505794)
```

データベース：コード説明

データベース制御で実施してるのは以下の二つのみです。

データベースに接続(connect)し、SQL文を実行(execute)する。

1. データベースへの接続(新規生成も兼ねる)

```
>>>import sqlite3  
>>> c = sqlite3.connect('test.db')
```

2. データベースにテーブル作成

```
>>>c.execute('CREATE TABLE vtuber(name TEXT NOT NULL UNIQUE,  
office TEXT, follower INTEGER, view INTEGER)')
```

NOT NULL…省略を許さない。UNIQUE…重複データを許可しない。

3. テーブルにデータ追加

```
>>>c.execute('INSERT INTO vtuber(name, office, follower, view)  
VALUES("キズナアイ", "upd8", 2690000, 283505794)')
```

```
>>> c.execute('COMMIT') ← 追加の場合、コミット必要。
```

データベース：コード説明

4. テーブル一覧取得

```
>>> for v in c.execute('SELECT * FROM vtuber'):  
...     print(v)  
...
```


課題 8

以下のデータを追加してみましょう。

名前：白上フブキ

所属：ホロライブ

フォロワー数：381000

総視聴数：35205441

課題 8 : 回答

```
>>>c.execute('INSERT INTO vtuber(name, office, follower, view) VALUES("白  
上フブキ", "ホロライブ", 381000, 35205441)')
```

```
>>> for v in c.execute('SELECT * FROM vtuber'):  
...     print(v)  
...
```

```
('キズナアイ', 'upd8', 2690000, 283505794)  
('白上フブキ', 'ホロライブ', 381000, 35205441)
```

データ集計 : pandas : データフレーム作成

pandasによるデータ集計です。

データベースの項で説明したDBについて、ランキング上位30件分登録したものが
あるので、それを使って pandasの機能を説明します。

まず、pandasにデータストリームを食わせます。

```
>>> import sqlite3
>>> import pandas as pd
>>>
>>> c = sqlite3.connect('vtuber.db')
>>> df = pd.read_sql_query('SELECT * FROM vtuber', c)
>>> pd.options.display.float_format='{:.2f}'.format
```

pandasはデータストリームを食わせると、**データフレーム**という独自の構造を
持ったインスタンスを生成します。このデータフレームを通じて、各種統計値、集計等
を処理できます。

最後はpandasの出力フォーマットを指定します。(指数表記がデフォなので、明示指定)

データ集計 : pandas : 全データdump

データフレームをそのまま出力すると、データフレーム内で管理している全データをダンプします。

```
>>> df
```

	name	office	follower
0	キズナアイ	upd8	2690000
1	輝夜月	VIC	998000
2	ミライアカリ	元ENTUM	736000
3	電腦少女シロ	.LIVE	705000
4	田中ヒメ鈴木ヒナ	Unknown	547000
...			
28	葛葉	にじさんじ	232000
29	犬山たまき	Unknown	230000

データ集計 : pandas : 統計出力

統計出力(describe)

```
>>> df.describe()
```

	follower	view	
count	30.00	30.00	← 件数
mean	446600.00	57213634.97	← 平均値
std	458979.20	54194338.53	← 標準偏差
min	234000.00	563251.00	← 最小値
25%	262250.00	30075874.50	← 1/4分位数
50%	307500.00	41463078.00	← 中央値
75%	411750.00	64993347.00	← 3/4分位数
max	2690000.00	283505794.00	← 最大値

describeは自動的に各種統計量を計算してくれます。
ファン数、総視聴数それぞれについて出力されます。

Vtuber上位30件についていえば、平均フォロワー数が44万人であり、
平均的に累計8000万回、視聴されています。

データ集計 : pandas : ユニーク値集計

ユニーク値の集計(value_counts)

各Vtuberがどのオフィスに所属するのか、その分布を計算します。

所属オフィス('office'カラム)は以下のように抽出できます。

```
>>> df['office']  
0      upd8  
1      VIC  
2      元ENTUM  
...
```

value_countsを使うと、ユニーク値の集計を行ってくれます。

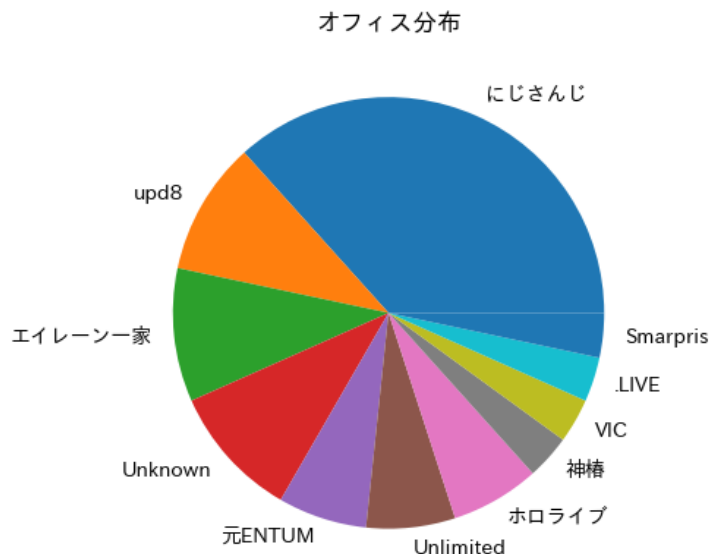
```
>>> df['office'].value_counts()  
にじさんじ      11  
エイレーン一家    3  
Unknown         3  
upd8            3  
...
```

プロット : 円グラフ : オフィス分布

前項で集計したオフィス分布を円グラフにプロットします。

```
>>> import matplotlib.pyplot as plt
>>> import japanize_matplotlib
>>> o_df = df['office'].value_counts()

>>> plt.title('オフィス分布')
>>> plt.pie(o_df.tolist(), labels=o_df.index.tolist())
>>> plt.show()
```



Vtuberのランキング上位30件において、にじさんじの比率が圧倒的に高いことがわかります。

その他主要事務所についてはおおむね、大きな差分が出ていないことがわかります。

データ集計 : pandas : 条件付き集計

にじさんじに所属するVtuberは以下のように絞り込むことができます。

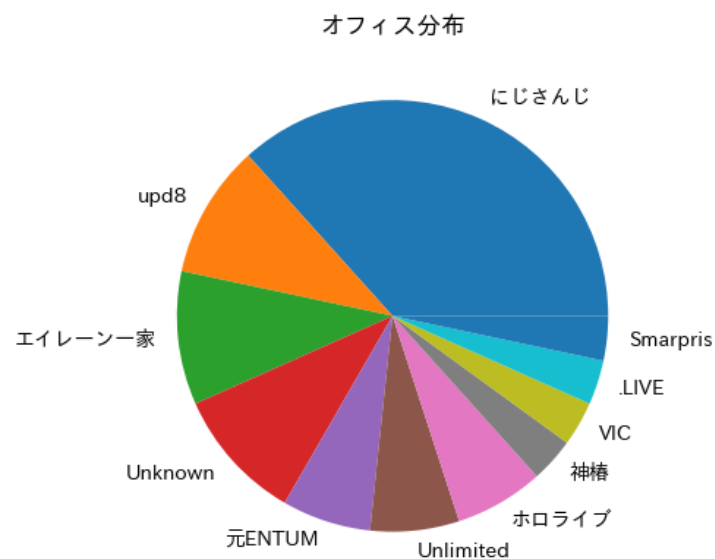
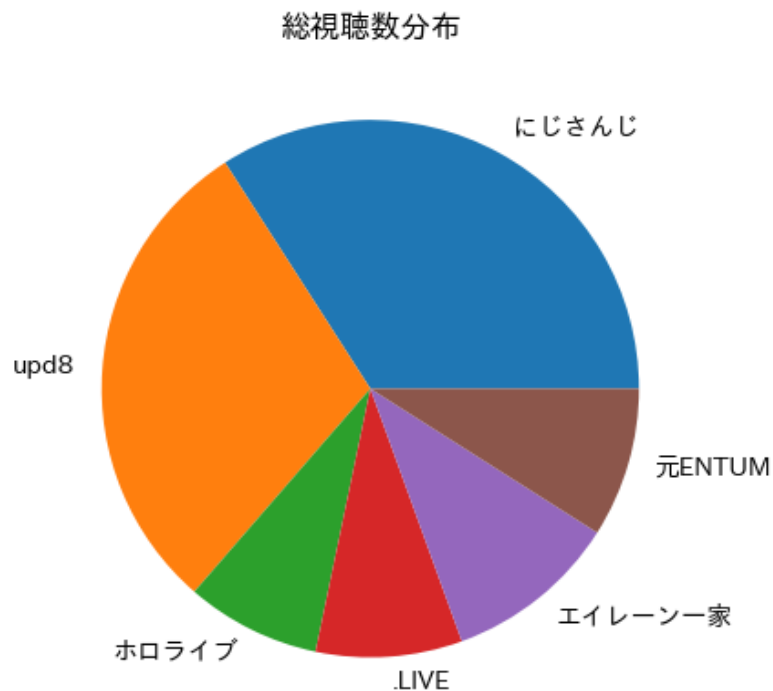
```
>>> df[df['office']=='にじさんじ']
```

ここからにじさんじ全体での総視聴数を以下のように集計できます。

```
>>> df[df['office']=='にじさんじ']['view'].sum()  
405786855
```


プロット：円グラフ：オフィス毎総視聴数

主要オフィスの総視聴数の比率をグラフにプロットすると以下です。



所属オフィス分布ではにじさんじの比率が著しく高かったですが、VTuberの配信力(総視聴数)で言うと、upd8が拮抗しています。

クラウドAPI : Twitter

Twitter APIはTwitter社が外部公開しているWebAPI(クラウドAPI)サービスです。

Twitter社に申請すると、APIを使用するためのアカウント情報が提供されて無償で使用できます。

因みに私はTwitter社と「申請の説明が足りないですよ、どんなことにAPIを使うのですか？」というメールを3回ほどやりとりして承認もらいました。
まあまあ大変。

申請手順は以下のサイト参照

<https://qiita.com/kngsym2018/items/2524d21455aac111cdee>

クラウドAPI : Twitter

TwitterのプロフィールからYoutubeアカウントを抽出したい。



コレ。

クラウドAPI : Twitter

APIはWebのお作法で直接叩けますが、pythonのTwitter APIライブラリが世の中にいくつかあるのでそれを使います。ここでは **tweepy** というライブラリを使います。

以下はミライアカリのTwitterアカウントから、youtube URLを取得するコードです。

```
import tweepy
```

```
auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
api = tweepy.API(auth , wait_on_rate_limit = True)
```

```
user = api.get_user(screen_name='MiraiAkari_prj')
youtube = user.entities['description']['urls'][0]['url']
```

4つのアカウントキーを渡してAPIインスタンスを生成し、APIインスタンスを使って、twitterユーザ情報にアクセスしています。今回はurl配列の先頭に格納されていますが、常にそうだとは限らないため、正規表現等を駆使して探し出す必要があります。

並列実行

ランキングサイト(50人分)をスクレイピングするにあたり、並列実装を試して通常コードと処理時間を比較してみました。

(1) マルチプロセス使用時

```
real    0m9.522s  
user    0m4.188s  
sys     0m0.719s
```

(2) スレッド使用時

```
real    0m16.455s  
user    0m4.016s  
sys     0m0.547s
```

(3) プロセス、スレッド非使用時

```
real    1m10.050s  
user    0m3.875s  
sys     0m0.422s
```

並列実行

マルチプロセス(自宅PC 4 コアのパワー)が最も高速に動作する。
グローバルロックのからみでスレッド処理にやや弱いとされるpythonだが、
マルチスレッドによる効果も高いことがわかる。

スクレイピング処理において、Webページ取り込みの「待ち」が占めている時間が
(予想通りといえは予想通りだが)大きいことがわかる。

また、実行コンテキストAの任意の処理が終わらないと、実行コンテキストBの
任意の処理が停止する、等の実行コンテキスト間の依存関係がゼロなのも、
本結果には効いている。

並列実行 : スレッド:コード例

```
from concurrent.futures import ThreadPoolExecutor
```

```
with ThreadPoolExecutor(max_workers=4) as executor:
```

```
    for n in range(len(vtubers)):                ← 起動スレッド数ループ
```

```
        executor.submit(self._profile, vtubers[n])    ← スレッド関数(引数)呼出し
```

ThreadPoolExecutorモジュールを使うと簡単に実装できる。

max_workersは同時に実行するタスクの数。ここではVTuber4人分の

サイトを同時処理。4 => 8にすると2,3秒短縮されるが、8 => 16にすると

逆に遅くなる。うちのPCは4コア8スレッドなので、上限を超えるとスレッド化の効果が薄くなる、と思われる。

並列実行：マルチプロセス:コード例

```
from concurrent.futures import ProcessPoolExecutor
```

```
r_list = []
```

```
with ProcessPoolExecutor() as executor:
```

```
    for n in range(len(vtubers)): ← 起動プロセス数ループ
```

```
        r_list.append(executor.submit(self._profile, vtubers[n])) ← プロセス起動
```

```
    u_vtubers = []
```

```
    for r in r_list:
```

```
        u_vtubers.append(r.result()) ← プロセス間データ通信による受取り。
```

ProcessPoolExecutorモジュールは、ThreadPoolExecutorと全く同じインタフェースを持っている。つまり、マルチスレッドかマルチプロセスなのか、その違いを**ほぼ意識せずに**実装可能。

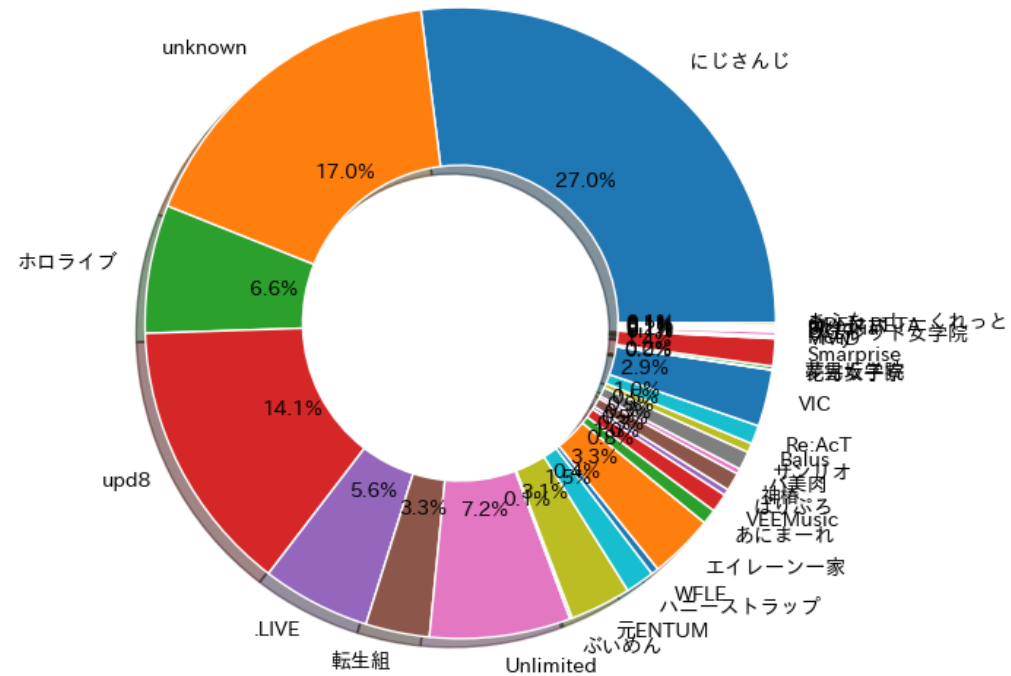
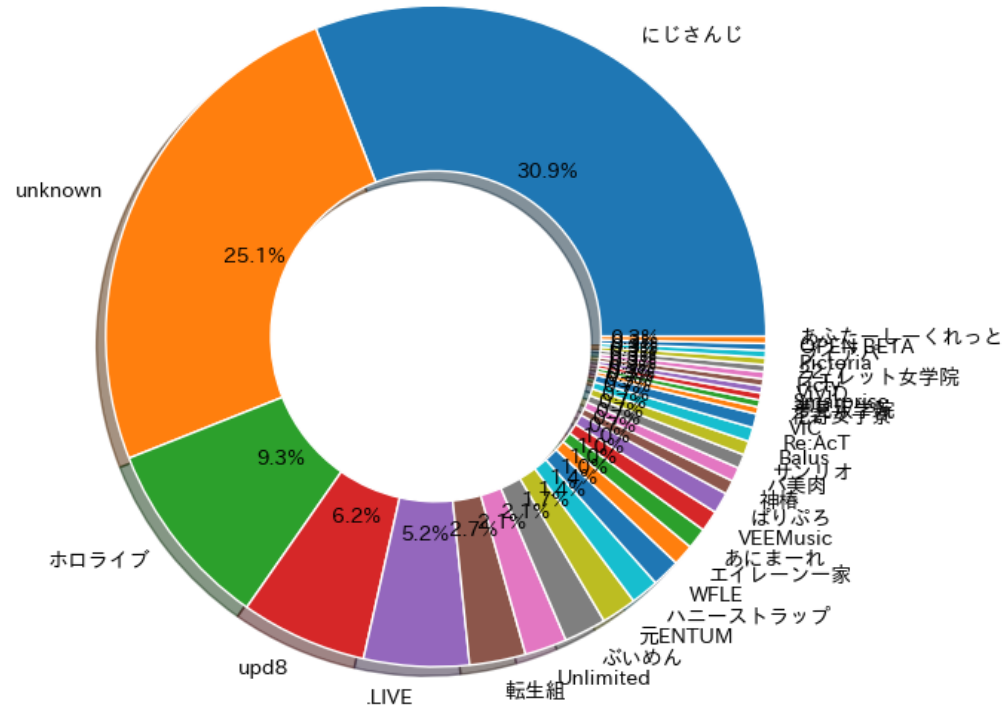
並列実行：マルチプロセス:コード例

ただし、実行したい関数は複数の異なるプロセス空間で並列実行されるので、マルチスレッドのようににはデータ共有できない。

引数で渡しているvtubers[n]は**プロセス間通信でコピーが転送**されるので、self._profile関数内で編集しても元のvtubersには反映されない。

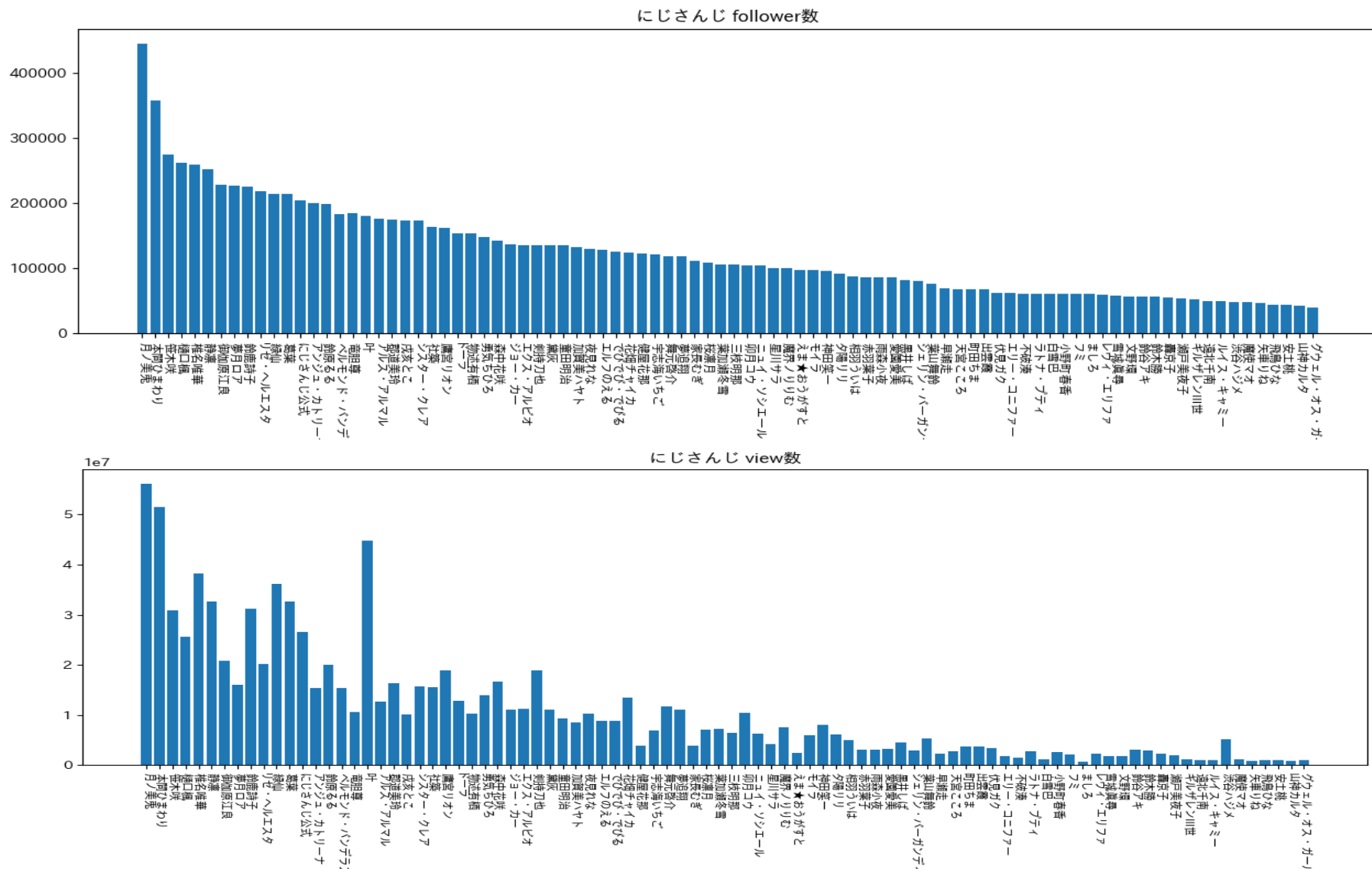
ProcessPoolExecutorは引数同様、関数の戻り値もプロセス間通信で転送してくれるので、この値を参照するために、戻り値を**result()**で抽出する。これが唯一、マルチプロセスを意識したコードとなるし、場合によってはこれがネックとなり、マルチプロセスの効果が弱まる。

view by office (総計 : 3,606,353,413)



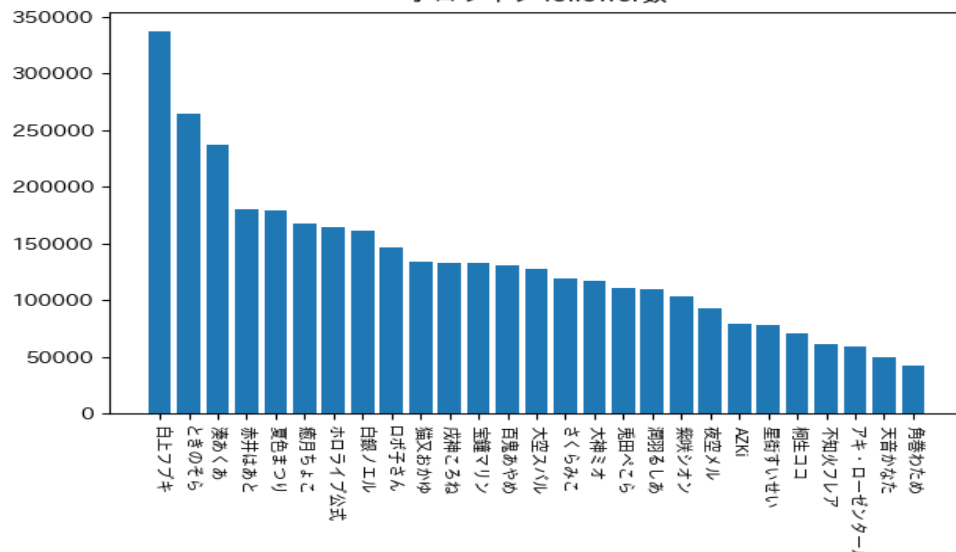
おまけ：Vtuber毎の配信力(にじさんじ)

フォロワー数で降順にプロットした時の、総視聴数の伸び。

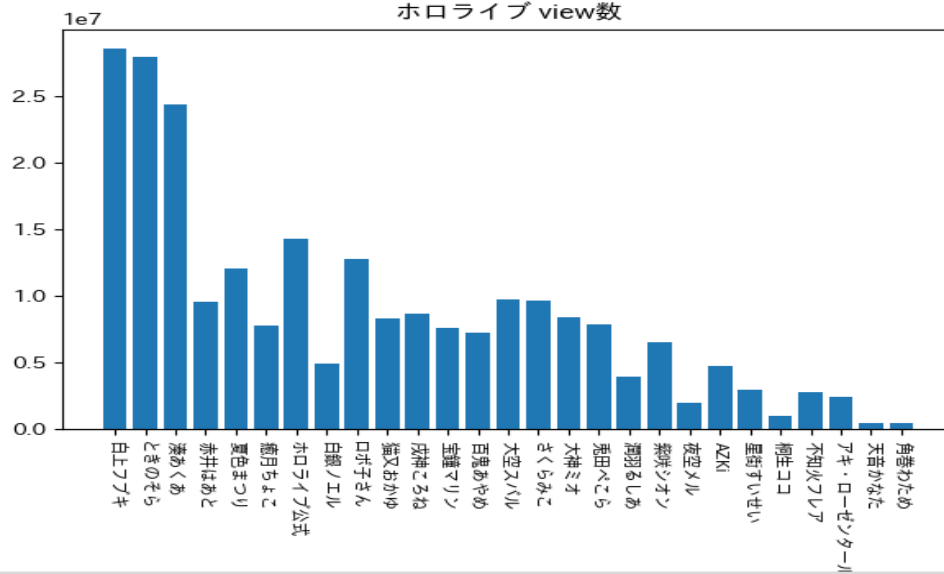


おまけ：Vtuber毎の配信力(ホロライブ)

ホロライブ follower数

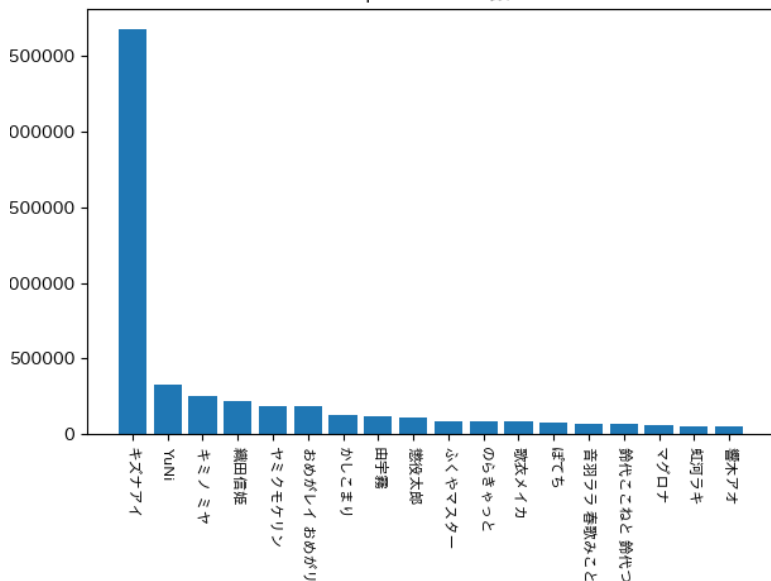


ホロライブ view数

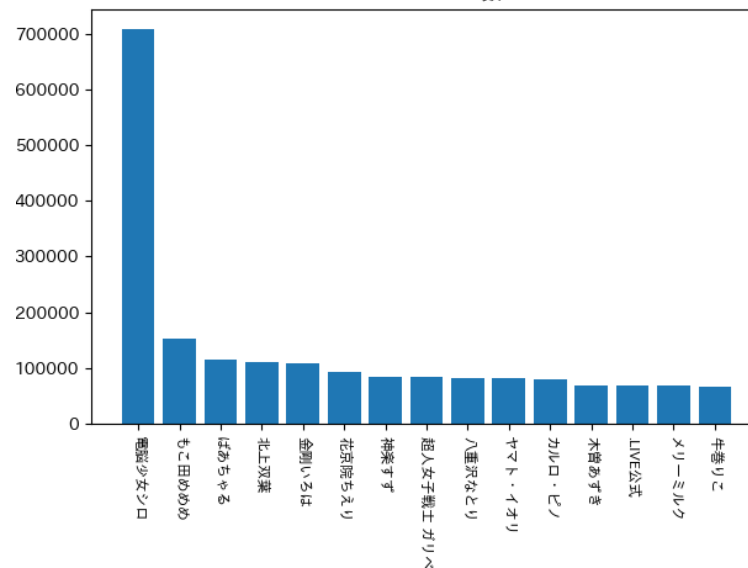


おまけ：Vtuber毎の配信力(upd8、.LIVE)

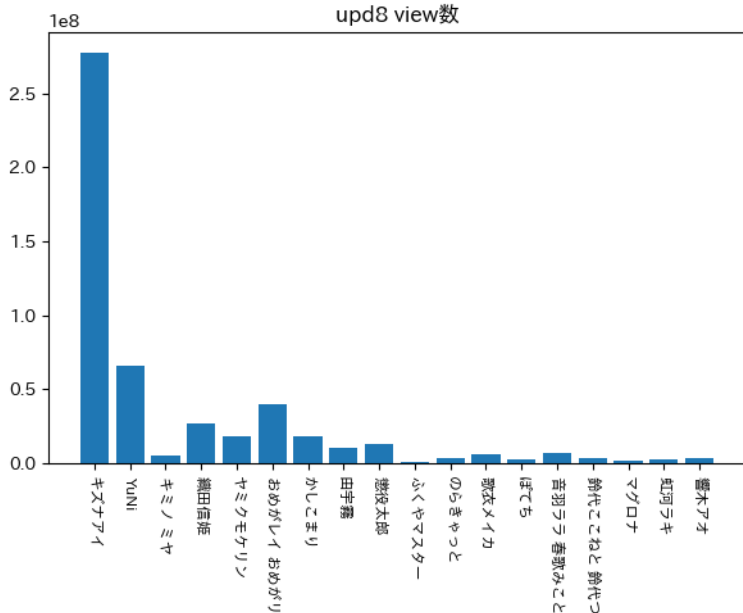
upd8 follower数



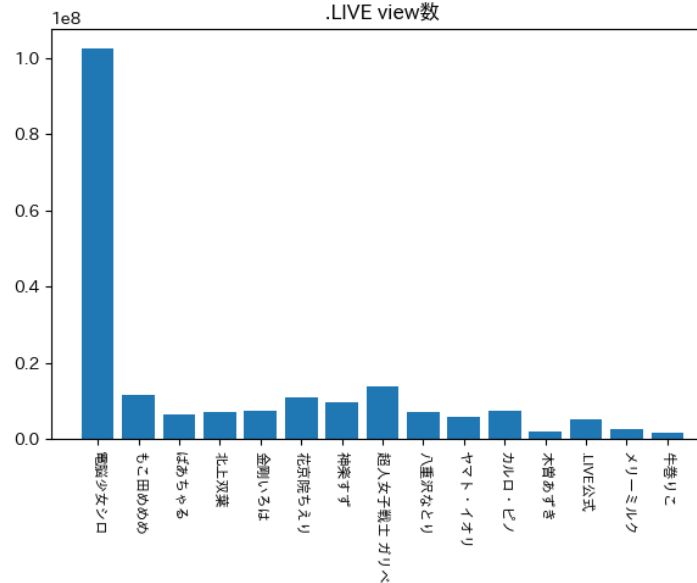
.LIVE follower数



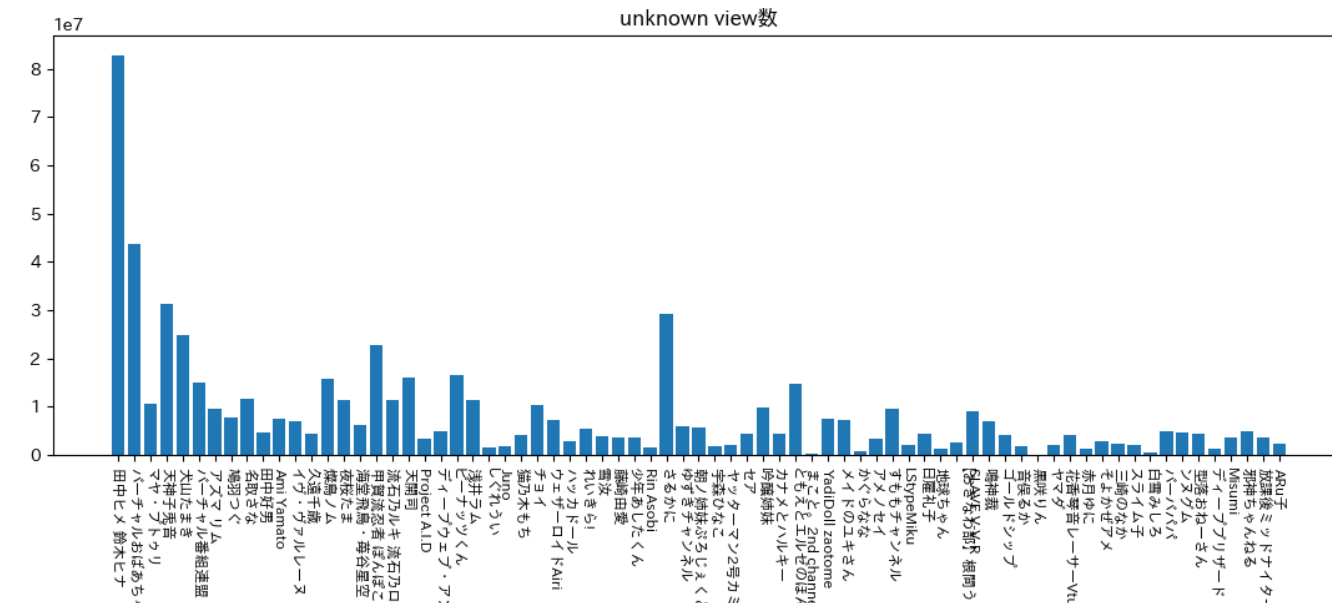
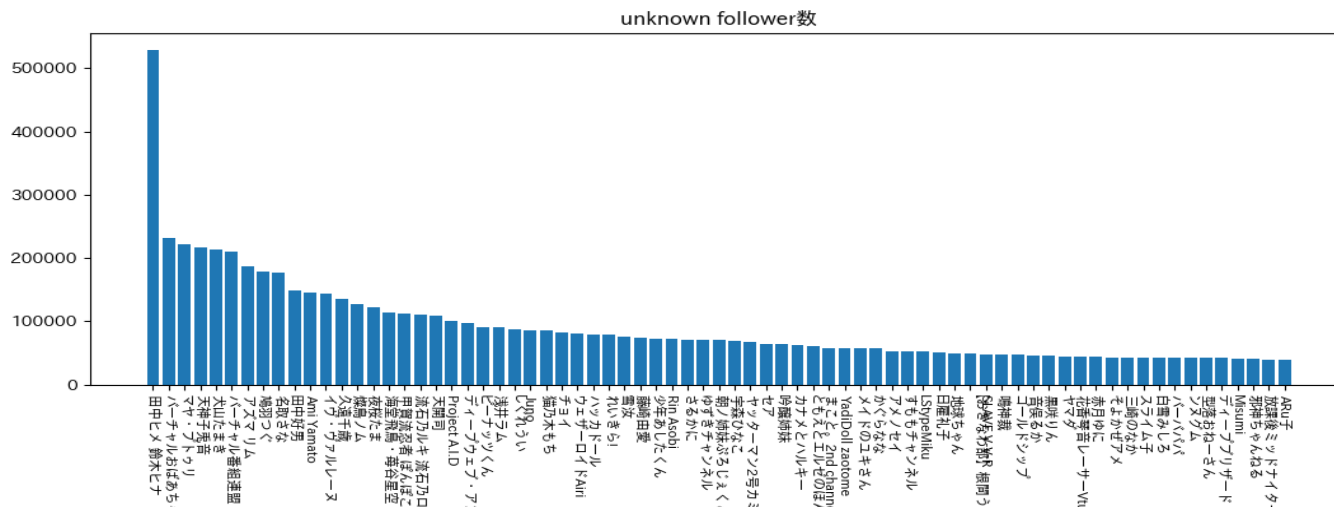
upd8 view数



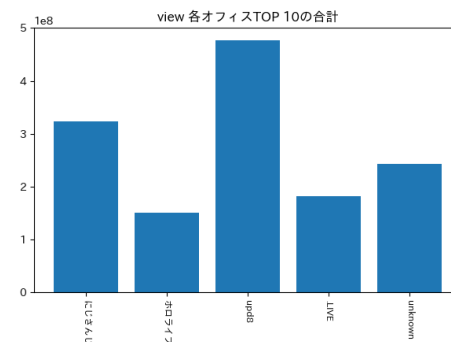
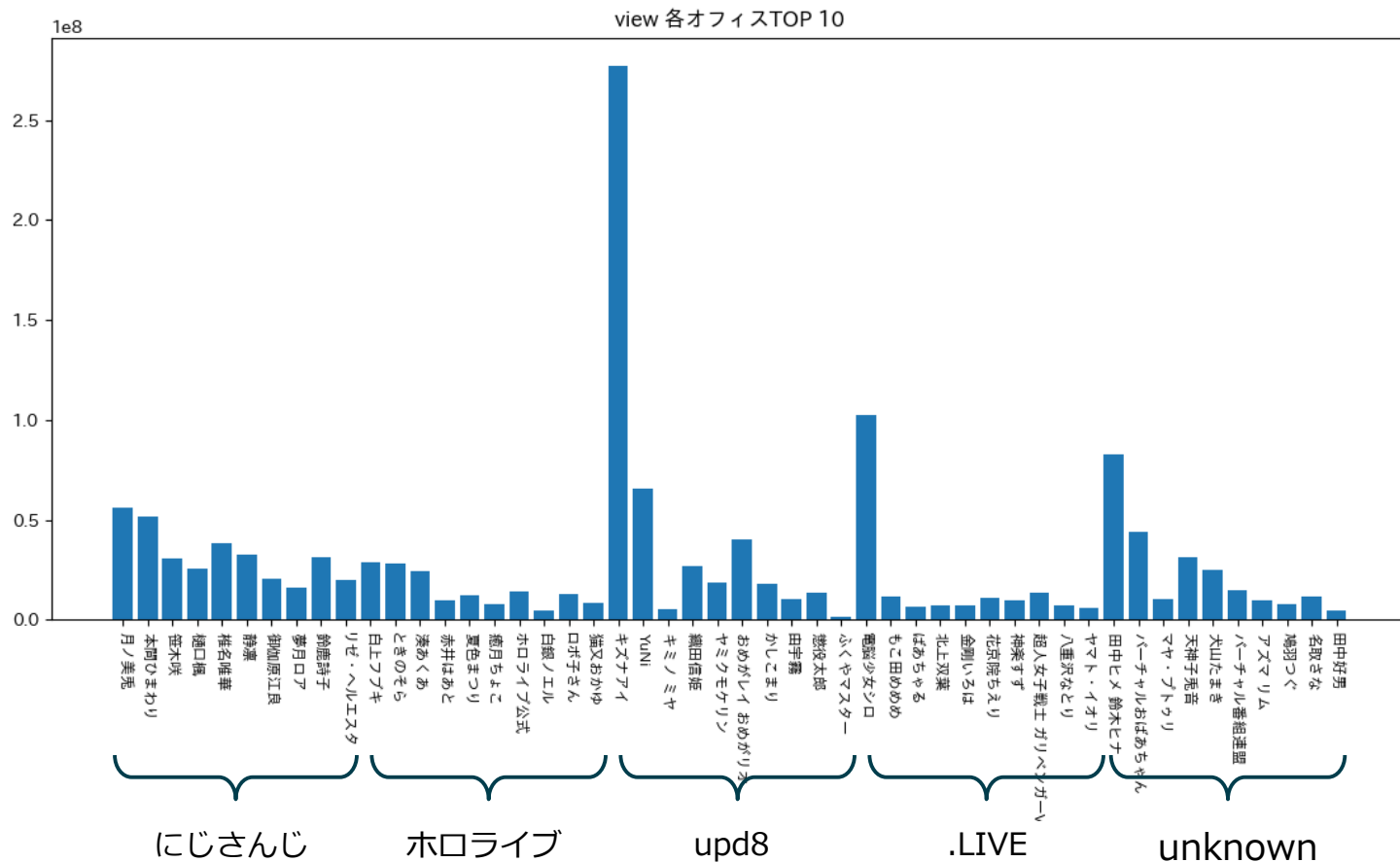
.LIVE view数



おまけ：Vtuber毎の配信力(unknown)

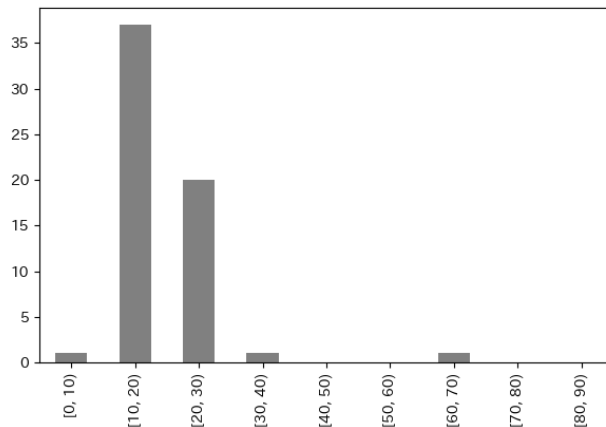


おまけ：各オフィスTOP10比較



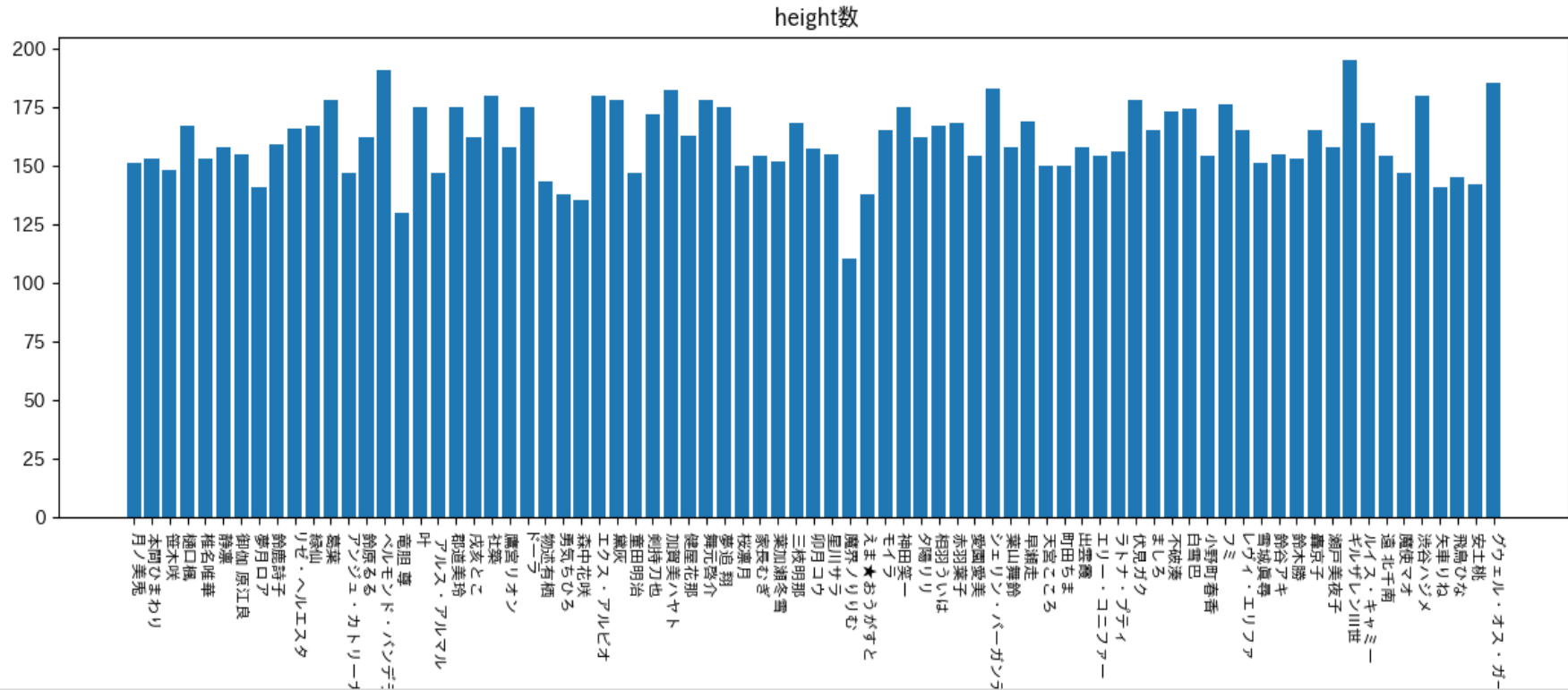
10人の合計でも特定
Vtuberのおかげで
他オフィスがupd8を凌
駕できない。

age数



Copyright © PERSOL RESEARCH & DEVELOPMENT CO., LTD. All Rights Reserved.

おまけ：身長分布(にじさんじ)



付録：python命名規則



コーディングの際の命名規則は以下に沿っています。
参考：<https://qiita.com/naomi7325/items/4eb1d2a40277361e898b>

 @naomi7325 2019年05月25日に更新

Python命名規則一覧

Python コーディング規約 命名規則

対象	ルール	例
パッケージ	全小文字 なるべく短くアンダースコア非推奨	tqdm, requests ...
モジュール	全小文字 なるべく短くアンダースコア可	sys, os, ...
クラス	最初大文字 + 大文字区切り	MyFavoriteClass
例外	最初大文字 + 大文字区切り	MyFuckingError
型変数	最初大文字 + 大文字区切り	MyFavoriteType
メソッド	全小文字 + アンダースコア区切り	my_favorite_method
関数	全小文字 + アンダースコア区切り	my_favorite_funcion
変数	全小文字 + アンダースコア区切り	my_favorite_instance
定数	全大文字 + アンダースコア区切り	MY_FAVORITE_CONST

- C/C++ のモジュールはアンダースコアで開始
- 自クラス内でのみ使用する内部変数と内部メソッドはアンダースコアで開始

PEP8で定められています。

環境構築

Pythonに関する様々なライブラリをまとめて入れてくれるAnacondaを使います。

1. Anacondaのダウンロードページへ行く
<https://www.anaconda.com/distribution/>
2. Python3.7バージョンをDLする。
OSによって32ビット、64ビットを間違えずに。
3. インストール
オプションはすべて表示されたままで進む。
5. Anaconda Promptの起動
いわゆるターミナル。Windowsの左下の検索窓からanaconda promptと打つと候補の中に現れる。
6. Matplotlib日本語パッチを当てる。
5のターミナルの中で以下のように打つ。
`pip install japanize_matplotlib`

環境構築(WSL版)

Windows10の新機能、WSLを使います。(WSL: Windows Subsystem for Linux)

1. Windows上にUbuntu用のWSL構築
元々インストールされているWSLを有効にし、
Microsoft StoreでUbuntuをインストールする。
参考 : <https://www.pc-koubou.jp/magazine/21475>

2. Ubuntuアップデート&日本語環境整備
Ubuntuを起動して以下の操作を行う。

【アップデート】

```
sudo apt update  
sudo apt upgrade
```

【日本語環境設定】

```
sudo apt -y install language-pack-ja  
sudo update-locale LANG=ja_JP.UTF8
```

ここでUbuntu再起動

【タイムゾーンを「東京」にセット】

```
sudo dpkg-reconfigure tzdata
```

【Ubuntuフォント設定】

Ubuntu画面の左上のアイコンをクリック→メニューからプロパティを選択し、
フォントをMSゴシックに設定

環境構築(WSL版)

3. 仮想Xウィンドウセットアップ
WSLはあくまでシェルサブシステムであり、linuxカーネル自体をエミュレートしているわけではないので、Windows上で動作する仮想Xウィンドウを起動し、WSLと接続する必要がある。

参考 : <https://qiita.com/ryoi084/items/c4339996c50c0cf39df4>

4. 各種パッケージインストール

※python3はデフォルトでUbuntuに入ってる。

【pythonパッケージャー、pip】
`sudo apt install python3-pip`

【スクレイピング用Lib】
`pip3 install beautifulsoup4`

【スクレイピング結果格納DB】
`sudo apt install sqlite3`

【スクレイピングデータ分析Lib】
`pip3 install pandas`

【Twitter APIライブラリ】
`pip3 install tweepy`

【スクレイピングデータプロット(グラフ描画)Lib】
`pip3 install matplotlib`
`pip3 install japanize_matplotlib (※)`
`sudo apt install python3-tk`

(※) <https://yolo.love/matplotlib/japanese> 参照