Hello Python World

～ 楽しく、基礎から実践まで学ぼう ～

Yohei Munesada

about me

宗定 洋平
(むねさだ ようへい)

G’s ACADEMY 講師 & メンター

https://www.yoheim.net

concept

Pythonを楽しみながら学ぶ

基礎〜実践まで盛りだくさんです

演習で手を動かして身につける

pre installed

```
$ python3 --version
Python 3.8.0

$ pip3 --version
pip 18.1 from /Library/Frameworks/Python.framework/
Versions/3.8/lib/python3.8/site-packages (python 3.7)
```

インストールはこちらから →　　https://www.python.org/

course catalog

Basic

- ✓ Pythonとは
- ✓ Python基本編
- ✓ モジュールとパッケージ

Advanced

- ✓ Webスクレイピング（基礎）
- ✓ Webスクレイピング（実践、ビデオのみ）
- ✓ Flaskを用いたWebアプリケーション（基礎）
- ✓ Flaskを用いたWebアプリケーション（実践、ビデオのみ）
- ✓ 手書き文字の判定アプリを作ろう（機械学習）
- ✓ 開発演習（任意提出）

course catalog (per day)

- ✓ Pythonとは
- ✓ Python基本編
- ✓ モジュールとパッケージ
- ✓ Webスクレイピング（基礎）
- ✓ Webスクレイピング（実践、ビデオのみ）
- ✓ Flaskを用いたWebアプリケーション（基礎）
- ✓ Flaskを用いたWebアプリケーション（実践、ビデオのみ） *1st day*

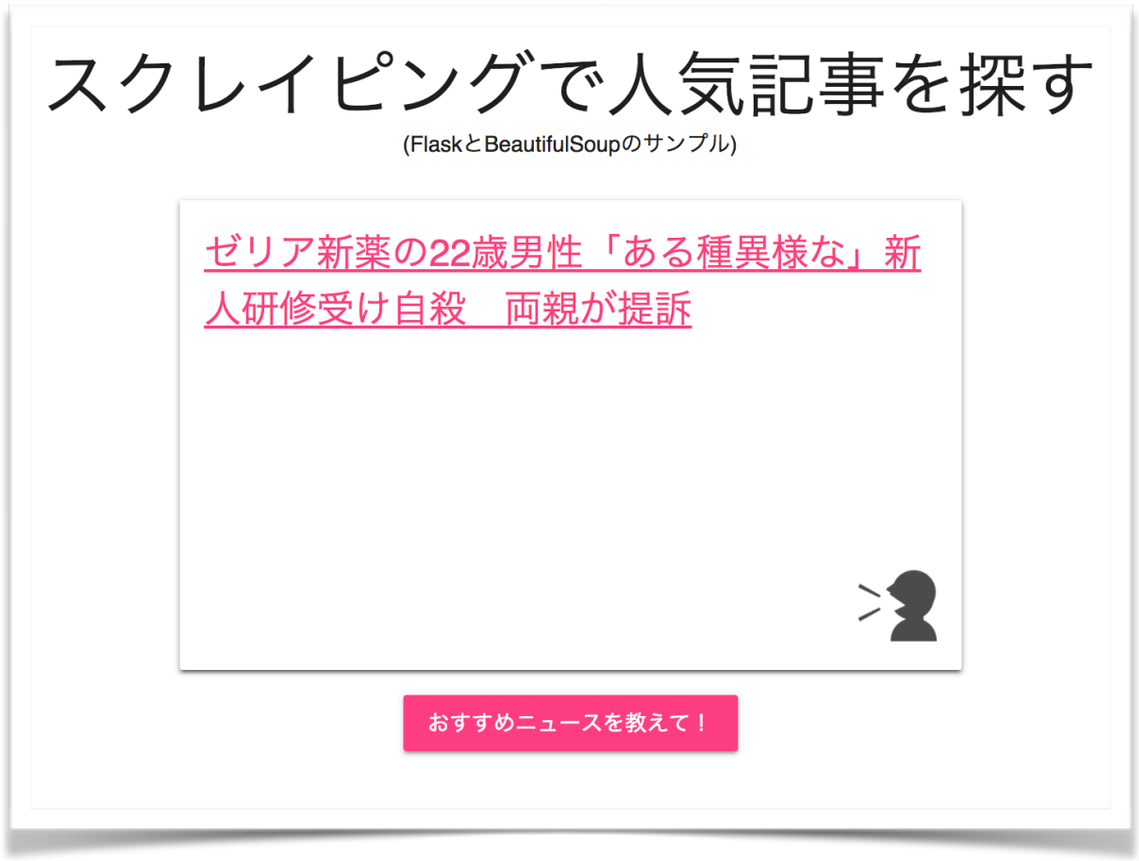
- ✓ 手書き文字の判定アプリを作ろう（機械学習） *2nd day*

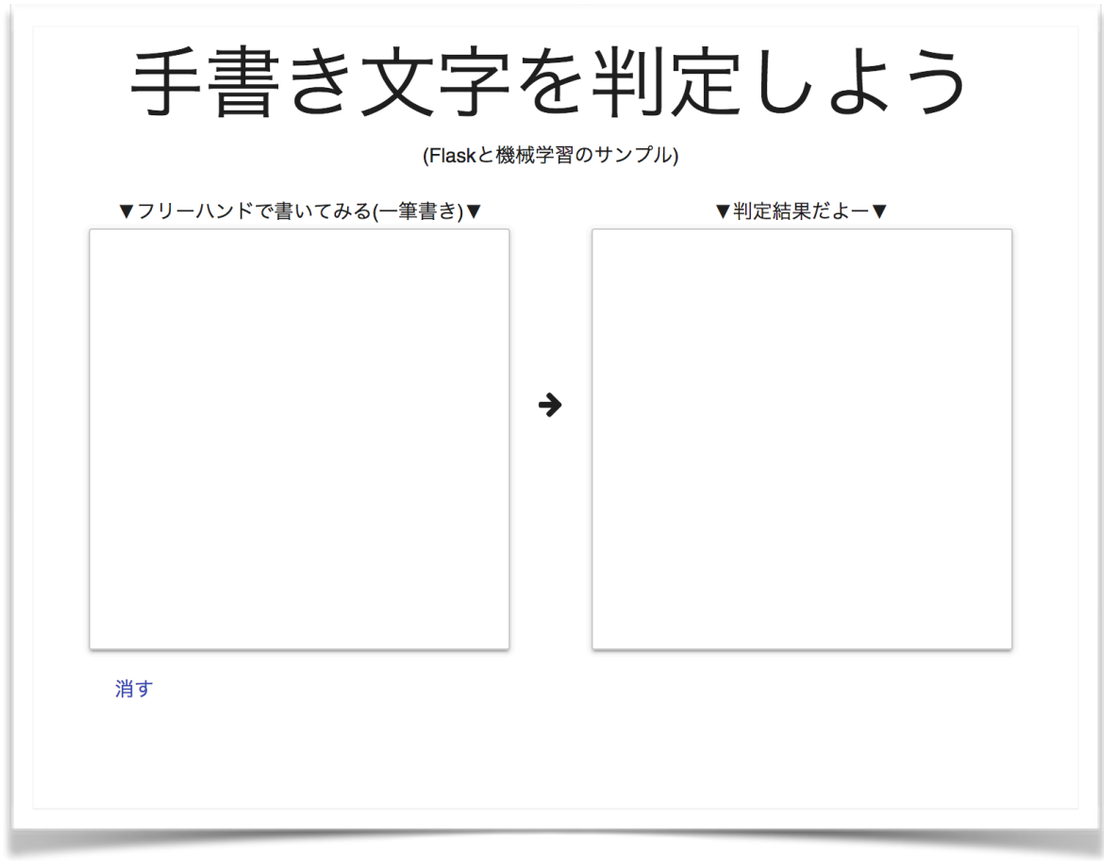
- ✓ 開発演習（任意提出）

course catalog (with video)

- ✓ Pythonとは
- ✓ Python基本編
- ✓ モジュールとパッケージ
- ✓ Webスクレイピング（基礎）
- ✓ Webスクレイピング（実践、ビデオのみ）
- ✓ Flaskを用いたWebアプリケーション（基礎）
- ✓ Flaskを用いたWebアプリケーション（実践、ビデオのみ）
- ✓ 手書き文字の判定アプリを作ろう（機械学習）
- ✓ 開発演習（任意提出）

apps you'll create





https://goo.gl/0v5dSj

https://goo.gl/KMgK3e

What is Python ?

agenda

- Pythonでできること

- Pythonとは

- Pythonのメリット

- Pythonのデメリット

Planner Web

- 日常的に必要な分析をPythonで自動実行
- 複数のデータソースを扱う分析も自動化し作業効率アップ

Python

Apache

バッチ処理

MySQL

HIVE

Google BigQuery

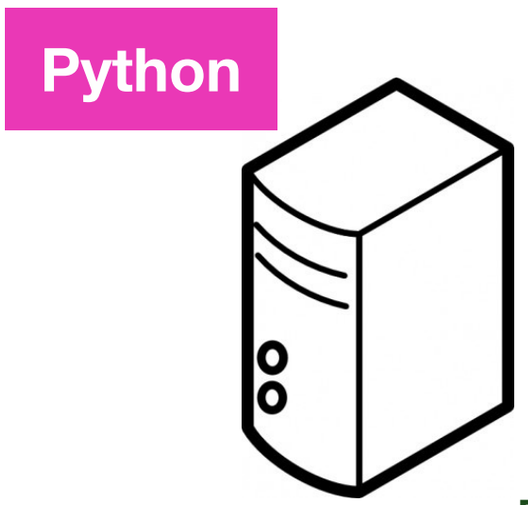
mongoDB.

Google Drive

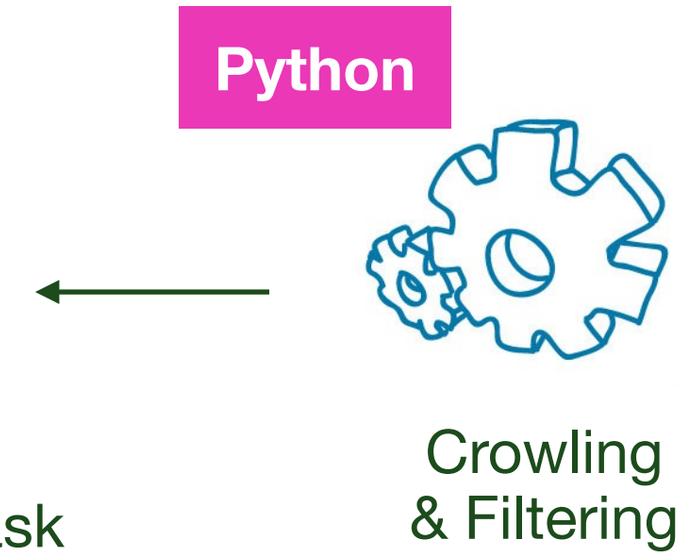
Google Analytics

Machine Learning Now

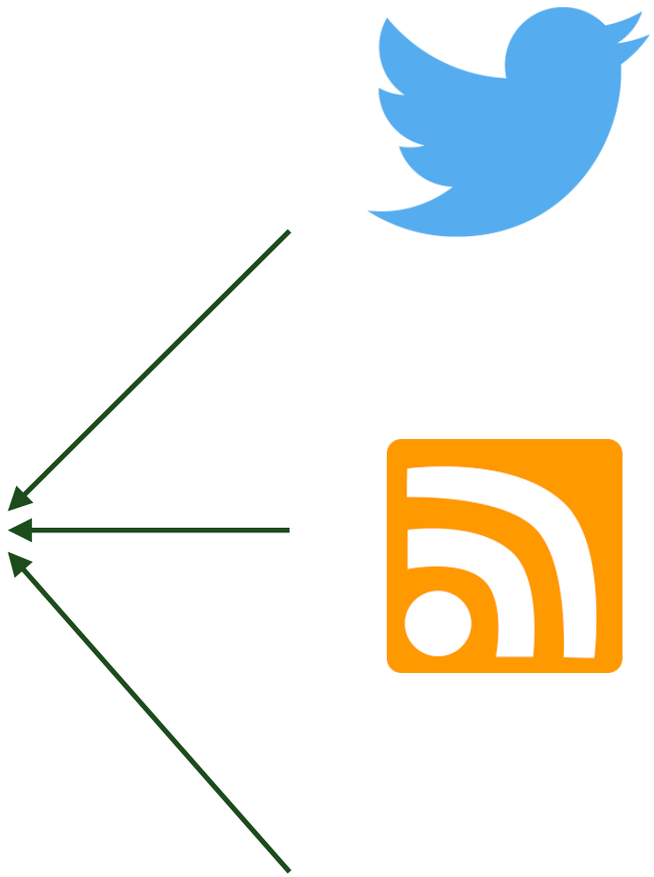
- 日々の情報収集を効率的に行うために仕組み
- 機械学習や人工知能に関する情報が雑多に集まる



Flask
Application



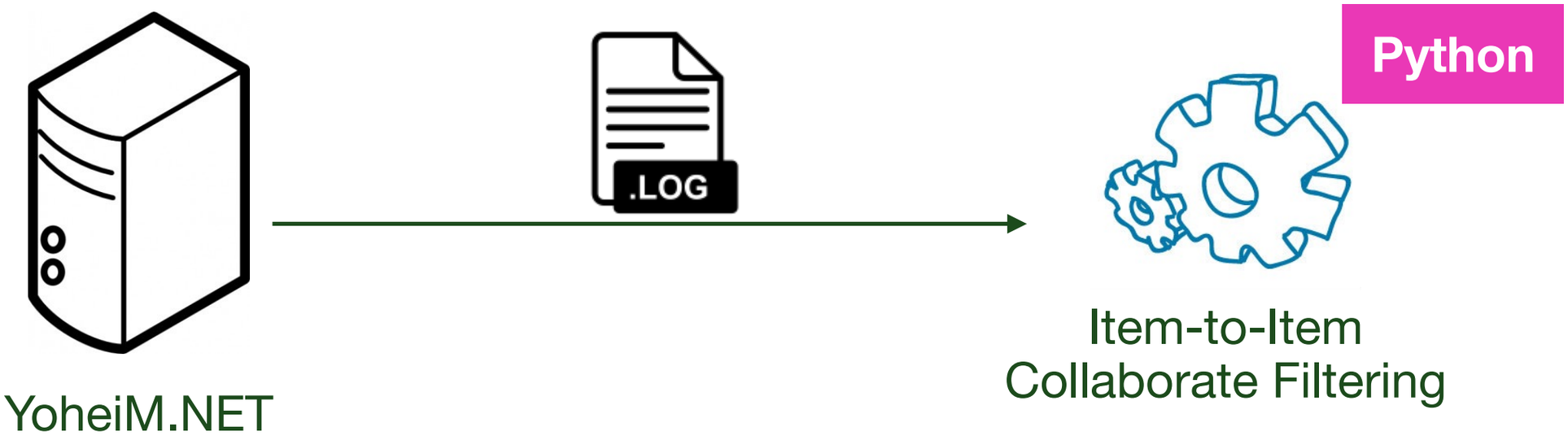
Crowling
& Filtering

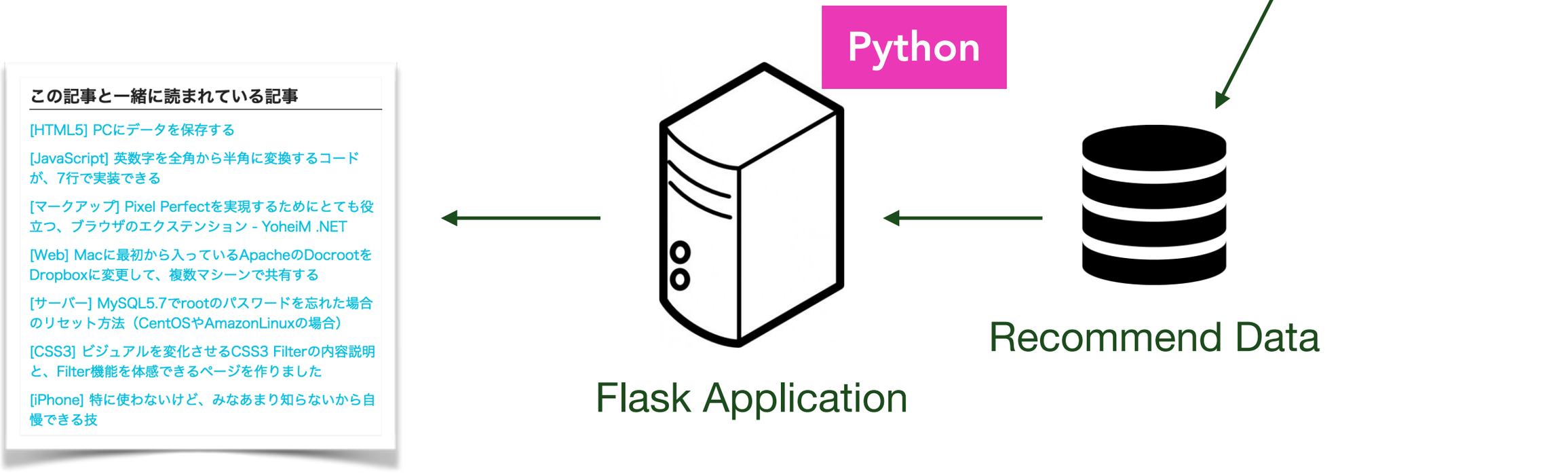


etc

Recommend System for YoheiM.net

- 記事の閲覧ログをもとにレコメンドを提供する





Python used at..

Youtube

Instagram

Gunosy

etc...

Python is

1991年にグイド・ヴァンロッサムにより公開された

イギリスのコメディ番組「空飛ぶモンティ・パイソン」が由来らしい

Python Software Foundation

good points

コードがシンプルで扱いやすい

数行書けば色々とできる

スクリプト言語でコンパイルは不要

マルチプラットフォーム

動的型付け

機械学習のライブラリが充実

not good points

2.x と 3.x の問題

英語のドキュメントに当たることが多い

日本語(マルチバイト)でハマる時がある

動的型付けのためリファクタリングが大変

レンサバで動かすのは大変..

Python Basic

agenda

Basic 1

- プログラムの実行
- インデントスタイル
- 条件分岐とループ処理
- 変数定義とデータ型

Basic 2

- データ構造 (List, Dictionary, Set, Tuple)
- 関数
- ファイルの読み書き
- クラス

agenda

Basic 1

- プログラムの実行
- インデントスタイル
- 条件分岐とループ処理
- 変数定義とデータ型

Basic 2

- データ構造 (List, Dictionary, Set, Tuple)
- 関数
- ファイルの読み書き
- クラス

how to execute

対話型実行

プログラム実行

その他 (jupyter notebookなど)

how to execute

これ以降は、デスクトップにファイルを作成します。

デスクトップへの移動方法は、下記の通りです。

```
Macの場合　　cd ~/Desktop
```

```
Windowsの場合　　cd C:¥Users¥xxx¥Desktop
```

indent style

Pythonはインデントを用いてプログラム構造を表現します

```
def api():
  search_no = get_search_no()
  if not search_no:
    Logger.get().warn("search_no doesn't exist.")
    return None, "error"
  else:
    try:
      result = special_execute_something(search_no)
      return result, None
    except:
      Logger.get().exception(“Oh my god!!!")
```

condition

Pythonの条件分岐とそのポイントを紹介します

```
if hour <= 6:
  print("so sleepy”)

elif 10 <= hour <= 12:
  print("good morning”)

elif hour < 12 and day_of_week == “Sunday" or hour > 20:
  print("good afternoon”)

else:
  print("hi")
```

loop

Pythonの繰り返し処理を紹介します

```python
for i in range(10):
  print(i)
```

```python
my_life = 5
while my_life:
  my_life -= 1
```

```python
for item in ["a", "b", "c"]:
  print(item)
```

```python
idx = 0
while True:
  if idx > 5:
    break
  idx += 1
```

variables

Pythonの変数を紹介します

```
# Define
my_name = "Yohei"
age = 20
is_good = True
unittest = 1
```

```
# Concat
my_name + age  # => Error
my_name + str(age)
```

```
# Check type
type(my_name)
```

agenda

Basic 1

- プログラムの実行
- インデントスタイル
- 条件分岐とループ処理
- 変数定義とデータ型

Basic 2

- データ構造 (List, Dictionary, Set, Tuple)
- 関数
- ファイルの読み書き
- クラス

list

リスト型は配列形式でデータを保持します

```
# Create
users = [
  "Yo", "Ken",
  “Nao”, "Shin", “Lee"
]
```

```
# Index access
users[0] # => ?
users[0:2] # => ?
users[1:] # => ?
users[::2] # => ?
users[::-1] # => ?
```

```
# Add
users.append("Miu")
```

```
# Delete
users.remove("Nao")
```

list comprehension

（リスト内包表記）

```
users = ["Yo", "Ken", "Nao", "Shin", "Lee"]
```

```
users = [u.lower() for u in users if u.find("e") != -1]
```

dictionary

辞書型はkey-valueで値を保持します

```python
# Create
user_dict = {
    "Yohei": 30,
    "John": 35
}
```

```python
# Get all
for k, v in user_dict.items():
  print(k, v)
```

```python
# Get
user_dict["Yohei"]
user_dict.get("Nao", 20)
```

```python
# Update / Delete
user_dict["Yohei"] = 31
del user_dict["John"]
```

set

セット型は重複がなく、順序を持たないデータの集合です

```
# Create
set_ = {
  "Tennis", "Ramen",
  “Programming"
}
```

```
# Add / Delete
set_.add("Gs")
set_.remove("Ramen")
```

```
# Get
set_[0] # => Error!!

for s in set_:
  print(s)
```

```
# Calc
set1 = set([1, 2, 3, 4, 5])
set2 = set([3, 4, 5])
set1 - set2
```

tuple

タプルはイミュータブルなリスト構造です

(変更できない)

```
# Create
nums = "One", "Two", “Three"
nums = (“One", "Two", “Three")
```

```
# Immutable
nums[0] = "Zero" # => Error!
```

```
# Get
nums[0]

for n in nums:
  print(n)
```

```
# Assign
a, b, c = nums
```

practice for basic

homework

<1> 写経してみよう

<2> 文字列とインデックスアクセス

「じこーだすまあさかんでのみしーゅっみてはたなのんしだいろな」から奇数番目の文字列と、偶数番目の文字列を抜き出してみよう

<3> Setの練習

「銀河鉄道の夜」の一節から、利用されている文字の種類数を取得しよう。

http://www.aozora.gr.jp/cards/000081/files/456_15050.html

<4> リスト内包表記

0〜100の数値を持つ配列を作成し、全ての要素を「(数値)円」にしよう。

function

Pythonの関数は非常に強力で、Pythonの良さの1つです

```
# Basic
def add(a, b):
  return a + b

result = add(10, 20)
```

```
# Keyword arguments
def create_date(
  year=0, month=0, date=0):
  return "..."

create_date(year=2017, date=10)
```

```
# Argument default value
def pow(a, b=2):
  return a ** b

result = pow(10)
```

```
# Multiple return
def get_user():
  return 'Yohei', 30

name, age = get_user()
```

read / write a file

Pythonにおけるファイルの読み書きを説明します。

```
# Write a file.
f = open("python.txt", "w")
f.write(“Hello")
f.close()
```

```
# Append a file.
f = open("python.txt", "a")
f.write(“Hi”)
f.close()
```

```
# Read a file.
f = open("python.txt")
txt = f.read()
f.close()
```

```
# using With
with open("python.txt") as f:
  txt = f.read()
```

class

Pythonにおけるクラスの使い方を説明します。

```
# Class and Constructor.
class User:
  def __init__(self, name):
    self.name = name
```

```
# Method.
class User:
  def say(self, name):
    print("Hello " + name)
```

```
# Instance.
user = User("Yohei")
print(user.name)
```

```
# Use method.
user = User()
user.say("Yohei")
```

practice for basic

homework

<1> 写経してみよう

<2> 関数

引数にxとyを受け付ける関数を作成し「私の名前はxでy歳です」と出力する関数を作成してみよう。「x=Yohei, y=30」を代入して動かしてみよう。年齢が省略された場合に”20”歳と表示するようにしよう。

<3> Python のクラスをさらに学ぼう

Python のクラスの機能はいっぱいです。以下の記事を参考に学んでみましょう！

http://www.yoheim.net/blog.php?q=20160405

practice for basic

homework

<4> その他にも

Python には便利なものがいっぱい。色々と触れてみてください。

- 位置引数のタプル化：http://www.yoheim.net/blog.php?q=20160609
- キーワード引数の辞書化：http://www.yoheim.net/blog.php?q=20160610
- リストのいろいろ：http://www.yoheim.net/blog.php?q=20150801
- コーディング規約：http://www.yoheim.net/blog.php?q=20160612
- デコレーター：http://www.yoheim.net/blog.php?q=20160607
- 演算子の定義：https://goo.gl/S99rvQ
- ラムダ式：http://uxmilk.jp/9426
- など

agenda

Basic 1

- プログラムの実行
- インデントスタイル
- 条件分岐とループ処理
- 変数定義とデータ型

Basic 2

- データ構造 (List, Dictionary, Set, Tuple)
- 関数
- ファイルの読み書き
- クラス

course catalog

- ✓ Pythonとは
- ✓ Python基本編
- ✓ モジュールとパッケージ

Basic

- ✓ Webスクレイピング（基礎）
- ✓ Webスクレイピング（実践、ビデオのみ）

Advanced

- ✓ Flaskを用いたWebアプリケーション（基礎）
- ✓ Flaskを用いたWebアプリケーション（実践、ビデオのみ）
- ✓ 手書き文字の判定アプリを作ろう（機械学習）
- ✓ 開発演習（任意提出）

Python Advance

モジュールとパッケージ

agenda

- モジュールの定義と利用

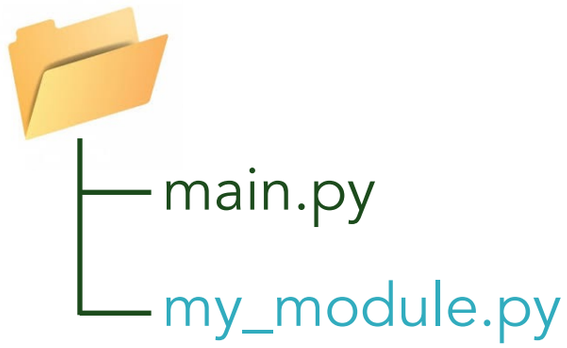
- パッケージを使う

- 標準モジュールを使う

- 外部モジュールを使う

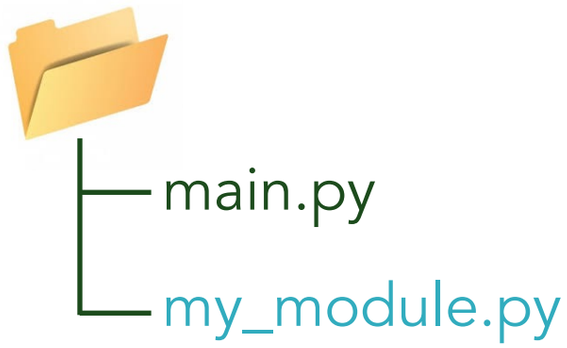
module

複数の処理を1つのファイルにまとめたもの



module

複数の処理を1つのファイルにまとめたもの

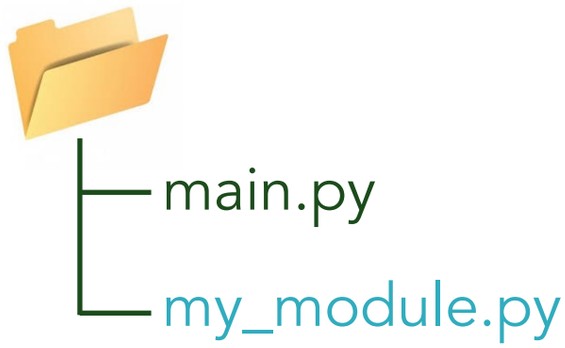



```python
# my_module.py
def get_genius():
  return ["Yamazaki", "Kosuge"]

def get_popular():
  return "Python Language"
```

```python
# main.py
import my_module

my_module.get_genius()
my_module.get_popular()
```

module

複数の処理を1つのファイルにまとめたもの

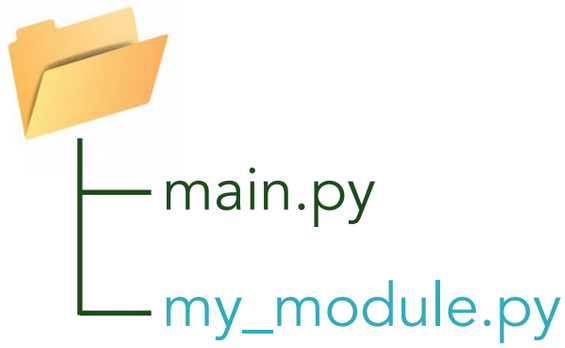



```python
# my_module.py
def get_genius():
  return ["Yamazaki", "Kosuge"]

def get_popular():
  return "Python Language"
```

```python
# main.py
import my_module as mm

mm.get_genius()
mm.get_popular()
```

module

複数の処理を1つのファイルにまとめたもの

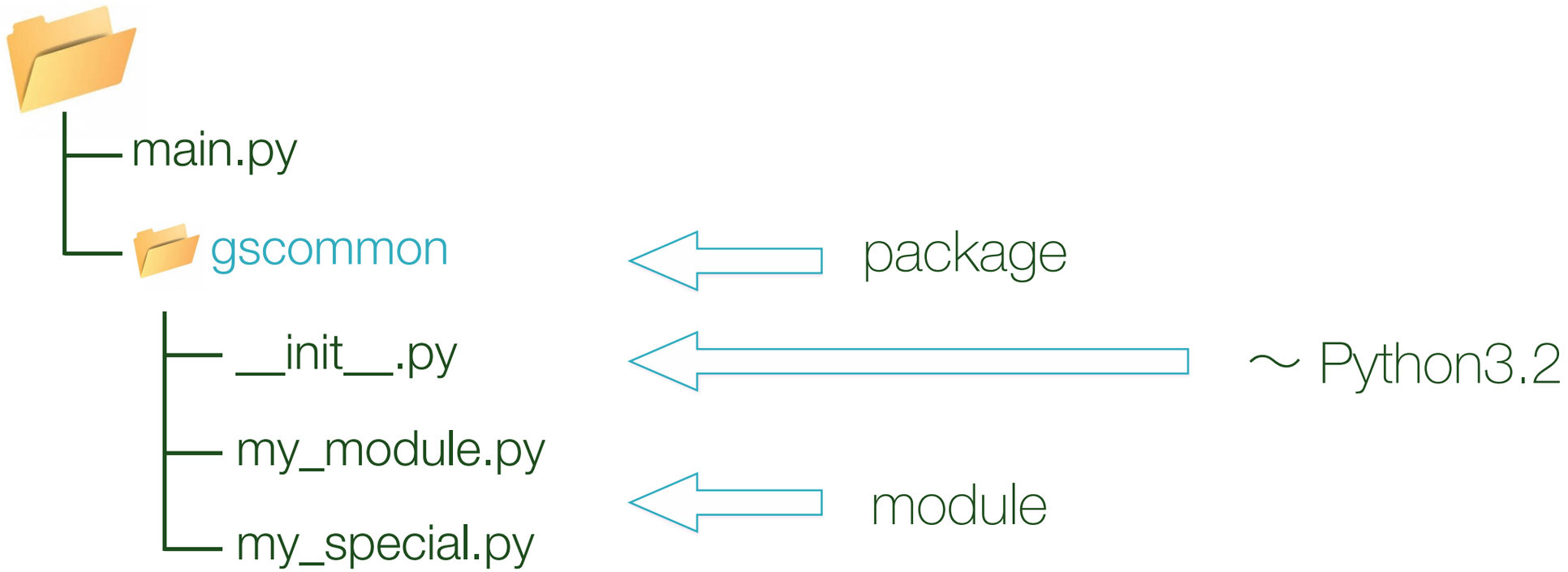



```python
# my_module.py
def get_genius():
  return ["Yamazaki", "Kosuge"]

def get_popular():
  return "Python Language"
```

```python
# main.py
from my_module import get_genius

get_genius()
```

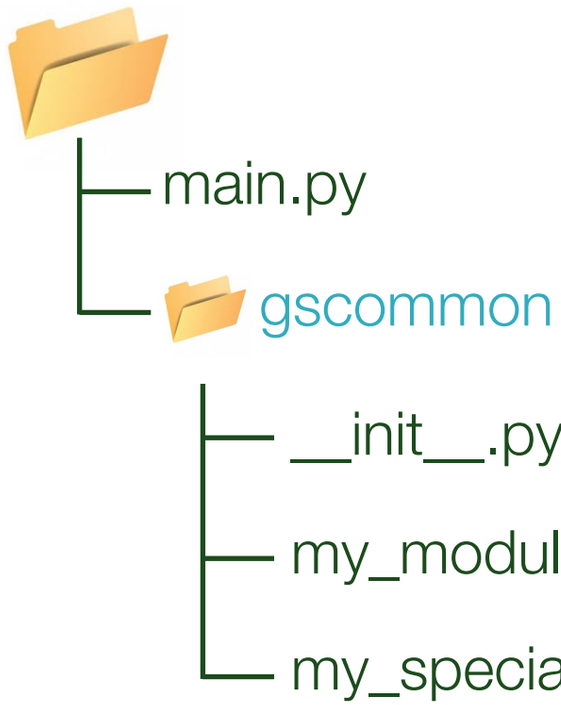
package

複数のモジュールをまとめたもの



package

複数のモジュールをまとめたもの

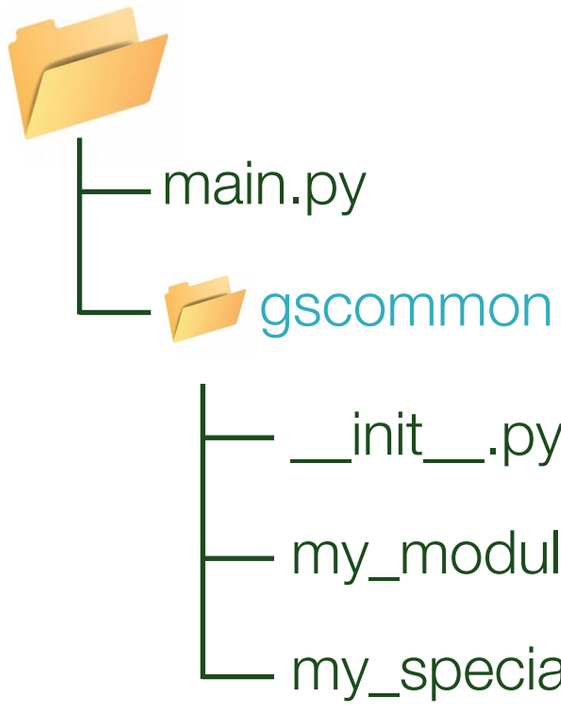



```python
# main.py
import gscommon
gscommon.my_module.get_genius()
```

```python
# gscommon/my_module.py
def get_genius():
    return ["Yamazaki", "Kosuge"]
```

package

複数のモジュールをまとめたもの

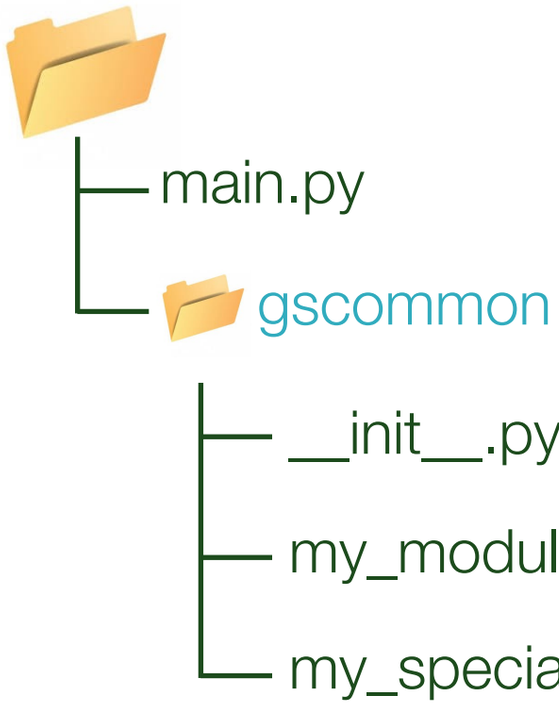



```python
# main.py
from gscommon import my_module
my_module.get_genius()
```

```python
# gscommon/my_module.py
def get_genius():
    return ["Yamazaki", "Kosuge"]
```

package

複数のモジュールをまとめたもの

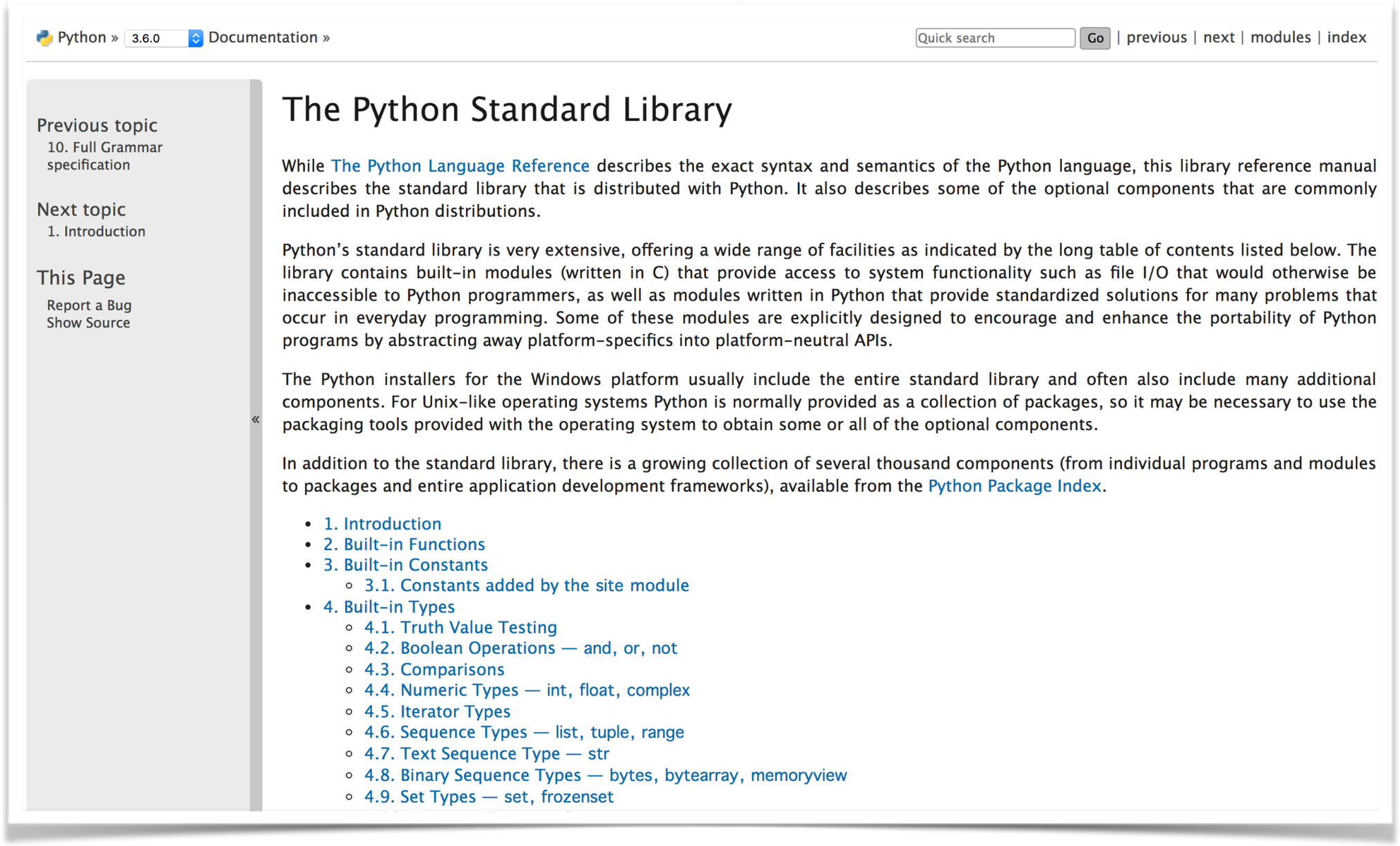



```python
# main.py
from gscommon import my_module as mm
mm.get_genius()
```

```python
# gscommon/my_module.py
def get_genius():
    return ["Yamazaki", "Kosuge"]
```

python standard libraries

標準ライブラリは非常に充実しています



https://docs.python.org/3/library/index.html

python standard libraries

例えばJSONを扱うライブラリ

```
# A example of `json`.
import json

s = """
{
    "name": "G's Academy",
    "members": 500
}
"""

school = json.loads(s)
```

https://docs.python.org/3/library/json.html

python standard libraries

例えばHTTP通信を扱うライブラリ

```
# A example of `urllib.request`.
import urllib.request

url = “http://www.yoheim.net”
with urllib.request.urlopen(url) as res:
  html = res.read().decode('utf-8')
```

https://docs.python.org/3/library/urllib.request.html

3rd party libraries

外部ライブラリは pip でインストールできます

```
# HTTPリクエストを発行するライブラリの例

$ pip3 install requests
```

https://github.com/requests/requests

practice for module and package

homework

<1> 写経してみよう

<2> モジュールの作成

「my_special.py」というモジュールを作成して使ってみよう

<3> パッケージの作成

「myutils」というパッケージを作成して使ってみよう

<4> 標準ライブラリの利用

以下のURLを参考に、datetimeモジュールとreモジュールを使ってみよう。

datetime : http://www.yoheim.net/blog.php?q=20150902

re : http://www.yoheim.net/blog.php?q=20160203

practice more

homework

<5> Pythonの仮想環境に挑戦！

実際の開発では、プロジェクトごとにPythonの環境を分けて行います。実現方法は、「pyenv/pyenv-virtualenv」を使う方法と「venv(virtualenv)」を使う方法があります。試してみましょう！

pyenv：http://www.yoheim.net/blog.php?q=20170204
pyenv-virtualenv：http://www.yoheim.net/blog.php?q=20170207
venv：https://github.com/yoheiMune/python-playground/tree/master/40_venv

<6> その他にも

他にも便利なものがいっぱいありますので、触れてみてください。

- 引数を扱う①：http://www.yoheim.net/blog.php?q=20151002
- 引数を扱う②：http://www.yoheim.net/blog.php?q=20160509
- ユニットテスト：http://www.yoheim.net/blog.php?q=20160902
- Google Analyticsを扱う：http://www.yoheim.net/blog.php?q=20151001
- など

practice more

homework

<7> その他にチェックすべき事項

・相対インポート：https://goo.gl/ZcA3RV
・デフォルトのパスと追加：http://blog.bonprosoft.com/684
・pip freeze と requirements.txt：https://goo.gl/A4Naru

course catalog

- ✓ Pythonとは
- ✓ Python基本編
- ✓ モジュールとパッケージ

Basic

- ✓ Webスクレイピング（基礎）
- ✓ Webスクレイピング（実践、ビデオのみ）

Advanced

- ✓ Flaskを用いたWebアプリケーション（基礎）
- ✓ Flaskを用いたWebアプリケーション（実践、ビデオのみ）
- ✓ 手書き文字の判定アプリを作ろう（機械学習）
- ✓ 開発演習（任意提出）

web scraping

BeautifulSoup4 を用いた Web スクレイピング

install

BeautifulSoup は pip 経由でインストールすることができます.

```
$ pip3 install beautifulsoup4
```

web scraping

steps ??

web scraping

urllib.request と beautifulSoup4 を使うと簡単にスクレイピングができます

3 steps

1. HTML をサーバーから取得する
2. BeautifulSoup で HTML を読み込む
3. DOM から情報を取り出す

web scraping

```
from urllib.request import urlopen
from bs4 import BeautifulSoup

# 1. Get a html.
with urlopen("http://www.yoheim.net") as res:
    html = res.read().decode("utf-8")

# 2. Load a html by BeautifulSoup.
soup = BeautifulSoup(html, "html.parser")

# 3. Get items you want.
titles = soup.select(".articleListItem h2")
titles = [t.string for t in titles]
```

web scraping

```
# Check results.
print(titles[:4])
```

```
['[Linux] 最終更新日や最終アクセス日を指定して、ファイルを検索/削除する',
 '[フロントエンド] Yamlというデータ構造に入門する',
 '[Docker] DockerのインストールとLinux起動まで',
 '[Javascript] 絵文字(サロゲートペア)を含んだ文字列の文字数を正しく取得する']
```

selectors in BeautifulSoup

beautifulSoup4 には主に2つの要素選択があります

select

```
soup.select(".articleListItem h2")
```

select_one

```
soup.select_one(“.articleListItem h2")
```

※ 他にも find, find_all などもあります。

extractors in BeautifulSoup

DOM要素から値を取得する方法は主に2つあります

text

```
# <h1>My Special App</h1>
elm.string
```

attribute

```
# <img src="/my_secret.png"/>
elm["src"]
```

practice for web scraping

homework

<1> 写経してみよう

<2> メンターの一覧を取得してみよう

G’sアカデミーのサイトから、メンター名を抜き出してみましょう。さらに余裕があればメンターの画像も抜き出してみましょう！

https://gsacademy.tokyo/mentor/

course catalog

- ✓ Pythonとは
- ✓ Python基本編
- ✓ モジュールとパッケージ

Basic

- ✓ Webスクレイピング（基礎）
- ✓ Webスクレイピング（実践、ビデオのみ）

Advanced

- ✓ Flaskを用いたWebアプリケーション（基礎）
- ✓ Flaskを用いたWebアプリケーション（実践、ビデオのみ）
- ✓ 手書き文字の判定アプリを作ろう（機械学習）
- ✓ 開発演習（任意提出）

web application

Flask を用いた Web アプリケーション

agenda

- 主要な WebApp ライブラリ
- Flask とは
- Flask のインストール
- スモールスタート
- ルーティング
- GET と POST
- テンプレート と static ファイル

major web app libraries

web apps ライブラリでは Django と Flask の2強です

django

重量級なライブラリで大規模開発に使われる。

flask

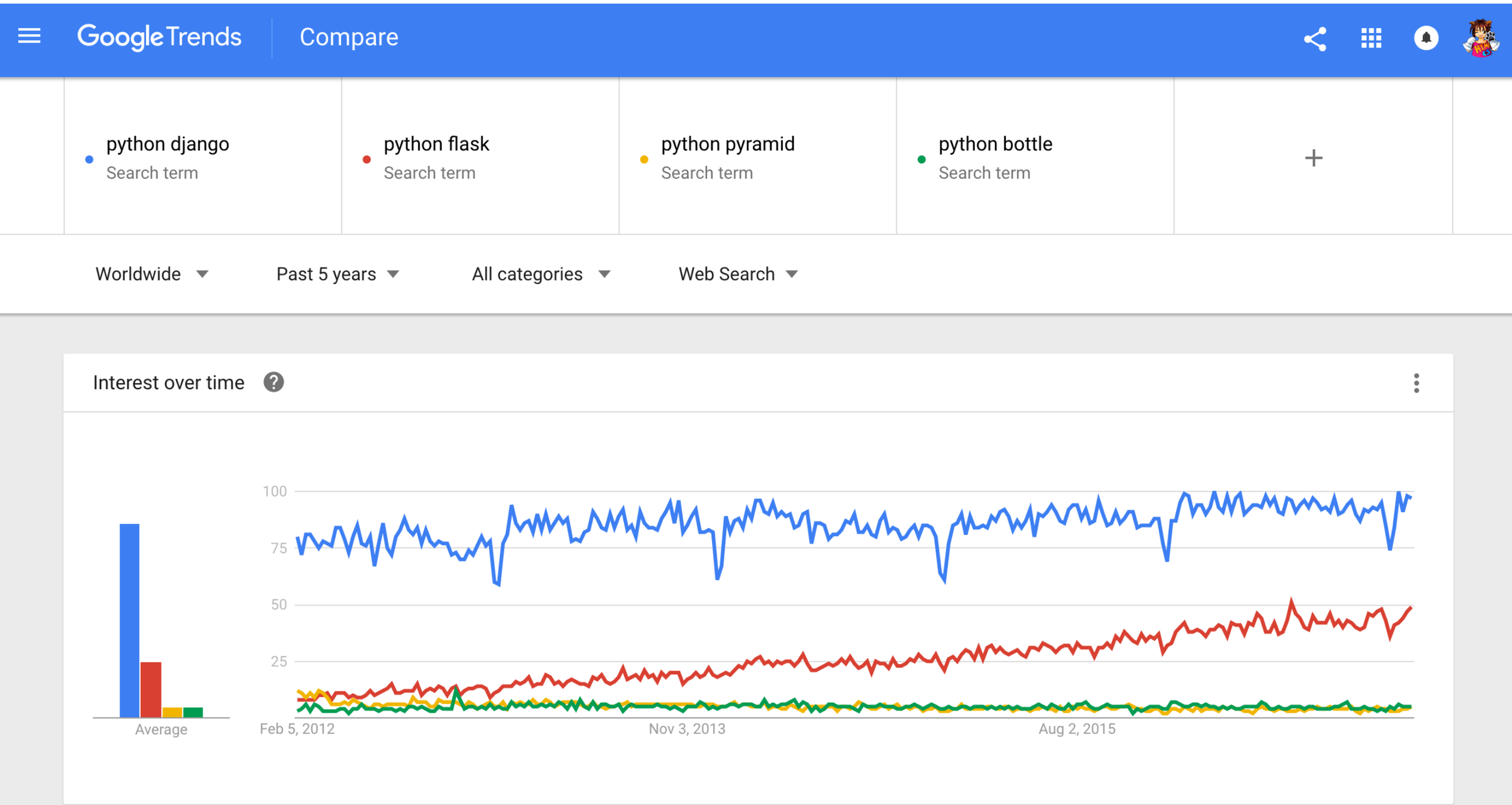
超軽量なライブラリで非常にお手軽に使える。

pyramid

bottle

...

major web app libraries



https://goo.gl/DVu9SE

major web app libraries

web apps ライブラリでは django と flask の2強です

django

重量級なライブラリで大規模開発に使われる。

flask

超軽量なライブラリで非常にお手軽に使える。

pyramid

bottle

...

Flask is

マイクロフレームワーク

request / response 周りの機能が中心

本番向けアプリケーションの利用実績も多数

アプリケーション設計(ディレクトリ構成など)は自分で行う

Database アクセスも自前で用意する

install

flask は pip 経由でインストールすることができます.

```
$ pip3 install Flask
```

small start

flask は簡単にサーバーを起動することができます.

```
from flask import Flask

app = Flask("app")

@app.route("/")
def index():
    return "Hello from flask"

app.run()
```

```
$ python3 app.py
```

routing

homework

ルーティングにはデコレーター（@app.route）を使います

```python
@app.route("/")
def index():
    return "Hello from flask"
```

```python
@app.route("/api/hello")
def api_hello():
    return "api_hello"
```

```python
@app.route("/api/items/<int:item_id>")
def api2(item_id):
    return "item_id is %d" % item_id
```

デコレーターとは：http://www.yoheim.net/blog.php?q=20160607

GET and POST

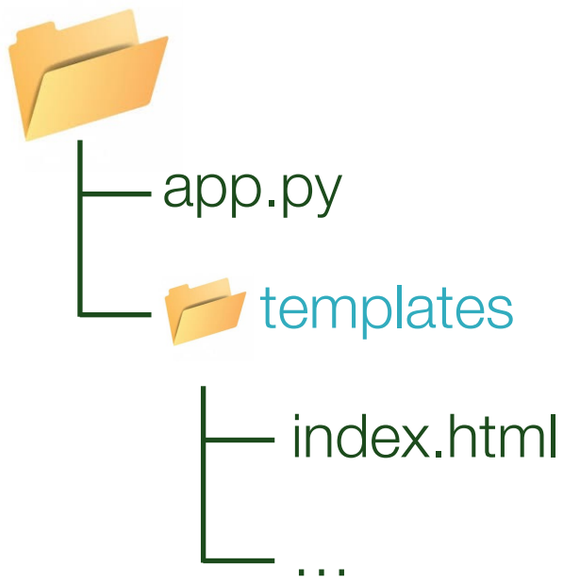
HTTPメソッドの指定方法と、値の取り出し方を説明します

```
from flask import Flask, request

@app.route("/api/users", methods=["GET"])
def api_users_get():
    search_key = request.args.get("user_id")
    return “user_id is %s" % user_id

@app.route("/api/users", methods=["POST"])
def api_users_update():
    user_name = request.form.get("user_name")
    return “user_name=%s" % (user_name)
```

template and static files

テンプレート機能 と static ファイルの配信を説明します

template




```python
from flask import Flask, render_template

@app.route(“/top")
def mypage():
    return render_template(
        "index.html",
        title=“Dear G's members”,
        message=“Congratulation!!!”
    )
```

```html
<html><body>
  <h1>{{ title }}</h1>
  <p>{{ message }}</p>
</body></html>
```

template and static files

テンプレート機能 と static ファイルの配信を説明します

static files

app.py

static

main.js

...

no code !

practice for flask application

homework

<1> 写経して、動かしてみよう

<2> Get と Post の対応

自分で好きなように拡張してみて、動くことを体験してください。

<3> ドキュメントを読んでみよう

実装を進めるにあたりドキュメントを読む必要があります。どのようなことが書かれているのかをざっと把握してみましょう。

http://flask.pocoo.org/

<4> コードリーディングに挑戦！

GsのSwift講義で使うサーバーサイドが Flask で開発してあります。

https://github.com/gs-swift/gsnap-server

course catalog

- ✓ Pythonとは
- ✓ Python基本編
- ✓ モジュールとパッケージ

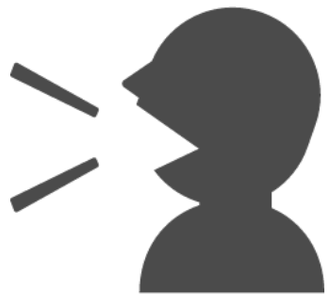
Basic

- ✓ Webスクレイピング（基礎）
- ✓ Webスクレイピング（実践、ビデオのみ）
- ✓ Flaskを用いたWebアプリケーション（基礎）
- ✓ Flaskを用いたWebアプリケーション（実践、ビデオのみ）
- ✓ 手書き文字の判定アプリを作ろう（機械学習）
- ✓ 開発演習（任意提出）

Advanced

1st demo app

Webチャットボットを作ろう



today's demo

音声で人気記事を探す

(FlaskとBeautifulSoupのサンプル)

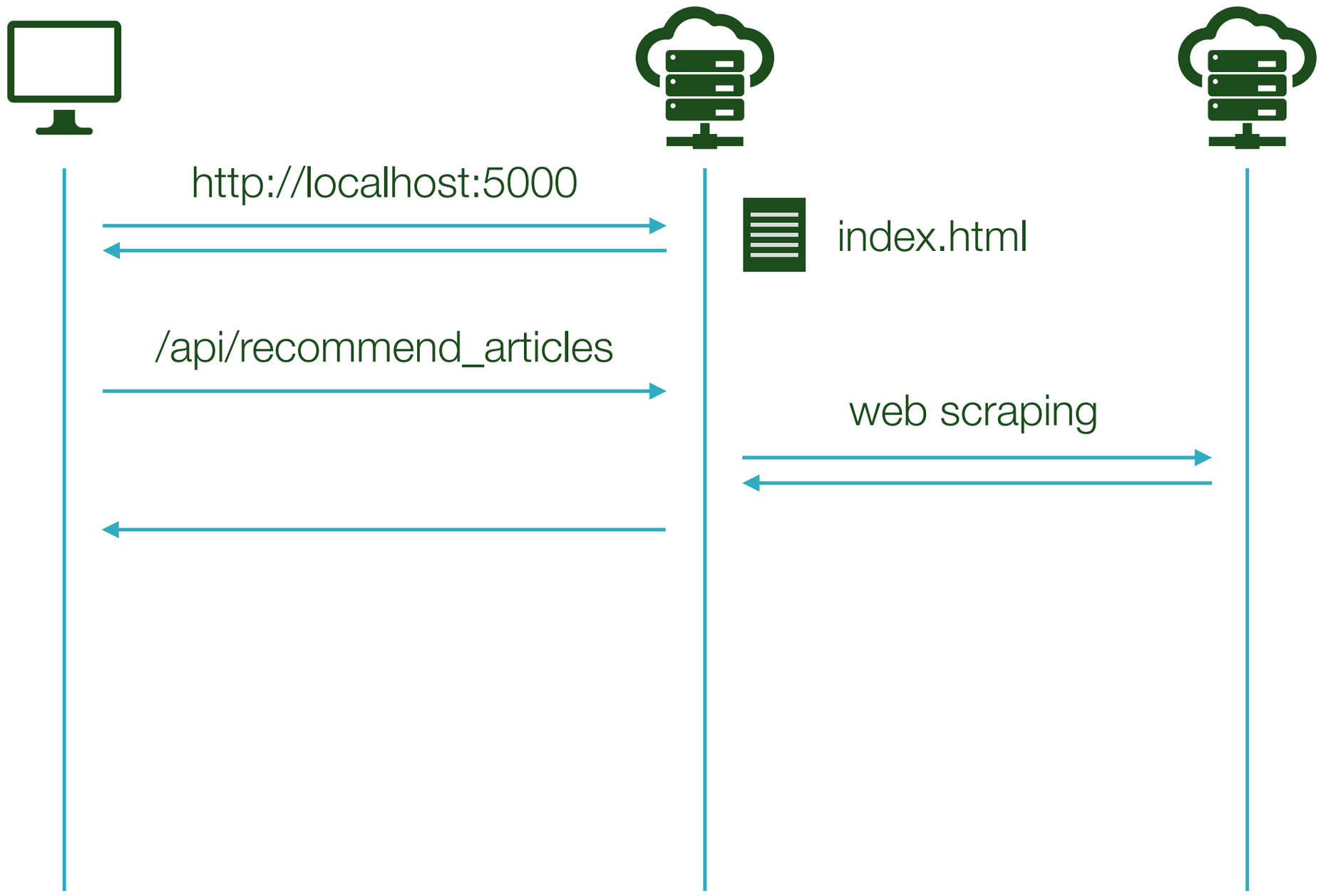
「おすすめニュースを教えて」と聞いてみてください。

音声認識を始める

https://goo.gl/0v5dSj

how it works

Web スクレイピングと Flask アプリケーションを組み合わせています



how to create

1. レポジトリをクローンする

2. ライブラリー覧を読み込む

3. 起動してみる

4. Web スクレイピングの処理を実装する

5. 動作テストをする

https://github.com/yoheimune-python-lecture/chatbot-news

extends the app

1.「今日の天気は？」に応えてみよう

2.「オススメのレシピは？」に応えてみよう

3. その他、自由に改造してみよう

course catalog

- ✓ Pythonとは
- ✓ Python基本編
- ✓ モジュールとパッケージ

Basic

- ✓ Webスクレイピング（基礎）
- ✓ Webスクレイピング（実践、ビデオのみ）
- ✓ Flaskを用いたWebアプリケーション（基礎）
- ✓ Flaskを用いたWebアプリケーション（実践、ビデオのみ）
- ✓ 手書き文字の判定アプリを作ろう（機械学習）
- ✓ 開発演習（任意提出）

Advanced

homework

1. Practice をやろう（特にコーディング）

2. デコレーターについて調べてみよう（Flaskにて登場）

3. Web チャットボットを完成させよう

4. ビデオ講義を確認しよう（任意）

5. 2日目の予習をしよう

Finish !