

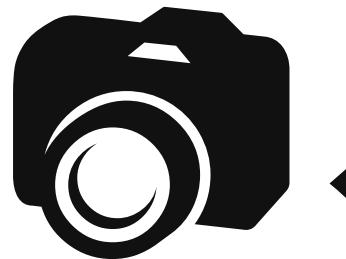
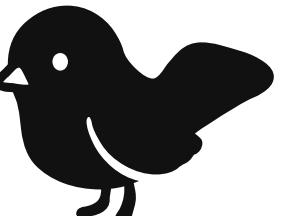
Python開発環境の整え方

2020年10月24日 / OSC Online/Fall

鈴木 たかのり

アジェンダ

- Pythonの環境について
- 開発環境を整える方法
- デバッグ方法

撮影 、ツイート  

#osc20on / @takanory

最初にお願い

- 顔出し推奨
- リアクション大きめで
- どんどん質問してください ■■
 - Zoomのチャット
 - YouTubeのコメント

スライドはこちら

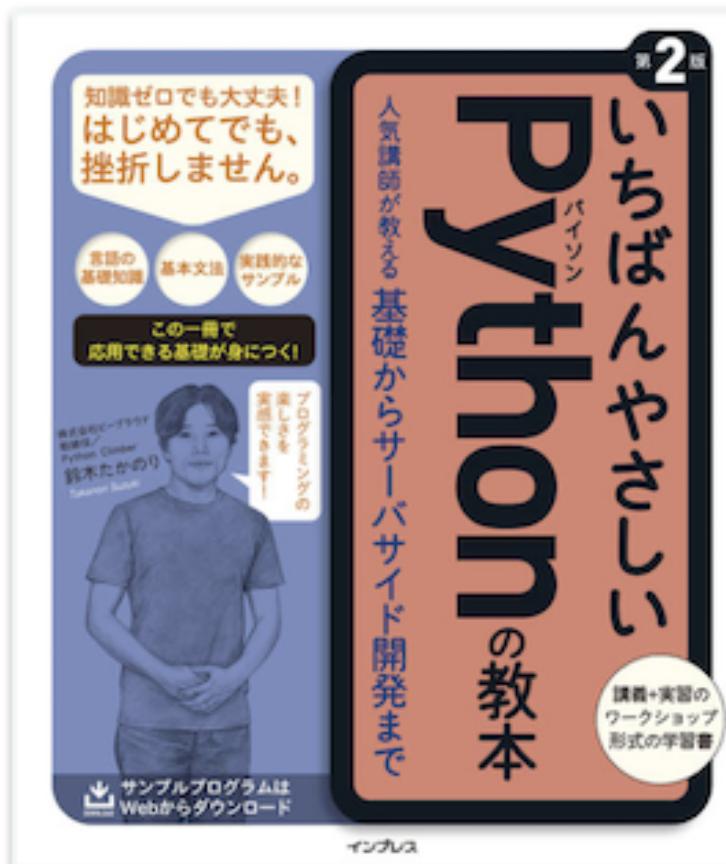
-  github.com/pyconjp/slides

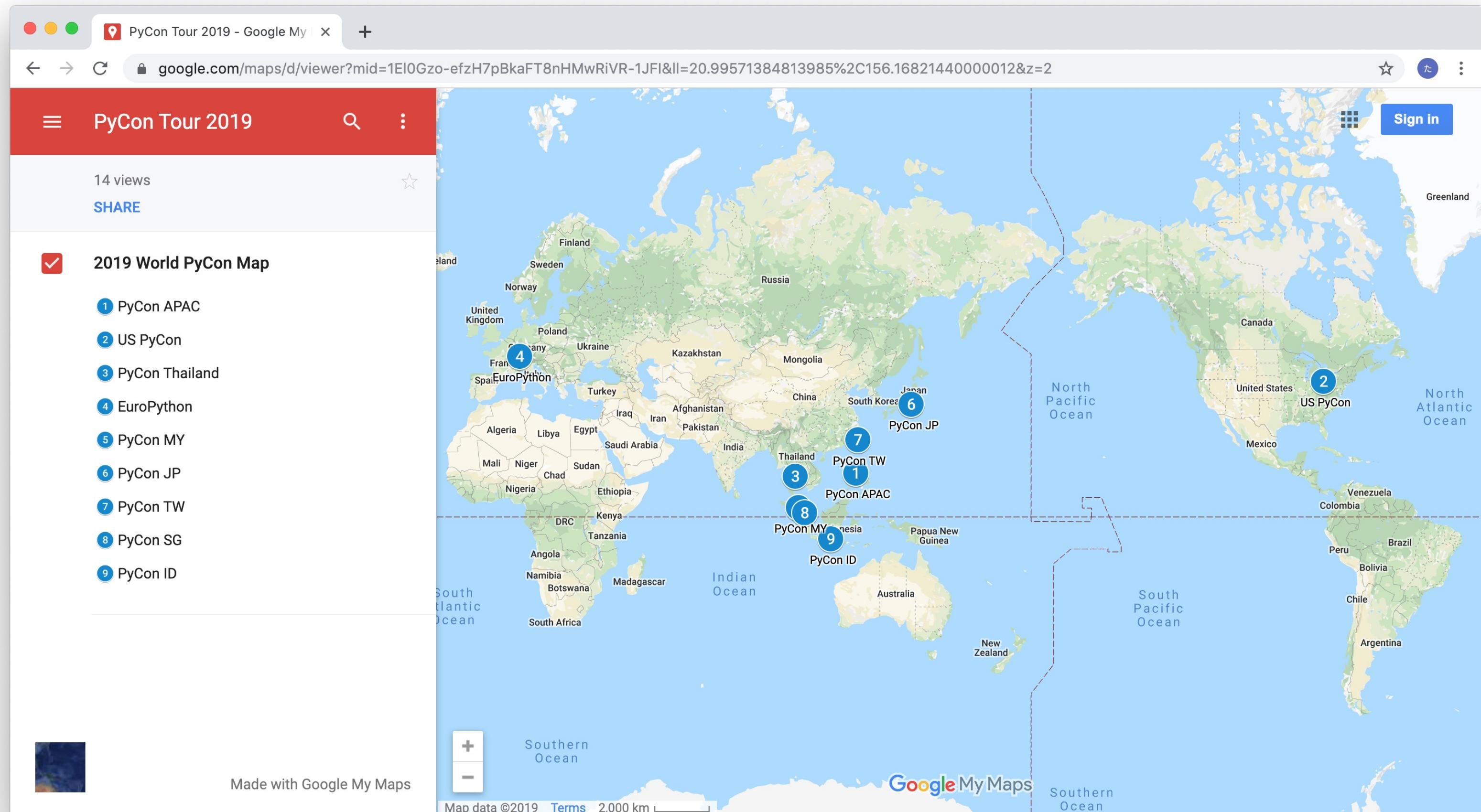
Who am I?(お前誰よ?)

- 鈴木たかのり / Takanori Suzuki
-  @takanory
- 一般社団法人 PyCon JP Association 副代表理事
- BeProud 取締役/Python Climber
- Python Boot Camp 講師, Python mini Hack-a-thon 主催, Python Bouldering Club 部長など



Python本書いてます





www.google.com/maps/d/viewer

一般社団法人PyCon JP Association

- ・日本国内のPythonユーザのために、Python の普及及び開発支援を行う為に、継続的にカンファレンス(PyCon)を開くことを目的とした非営利組織
- ・www.pycon.jp



(法人名が、一般社団法人PyCon JPから一般社団法人PyCon JP Associationに変更)

PyCon JP 2020

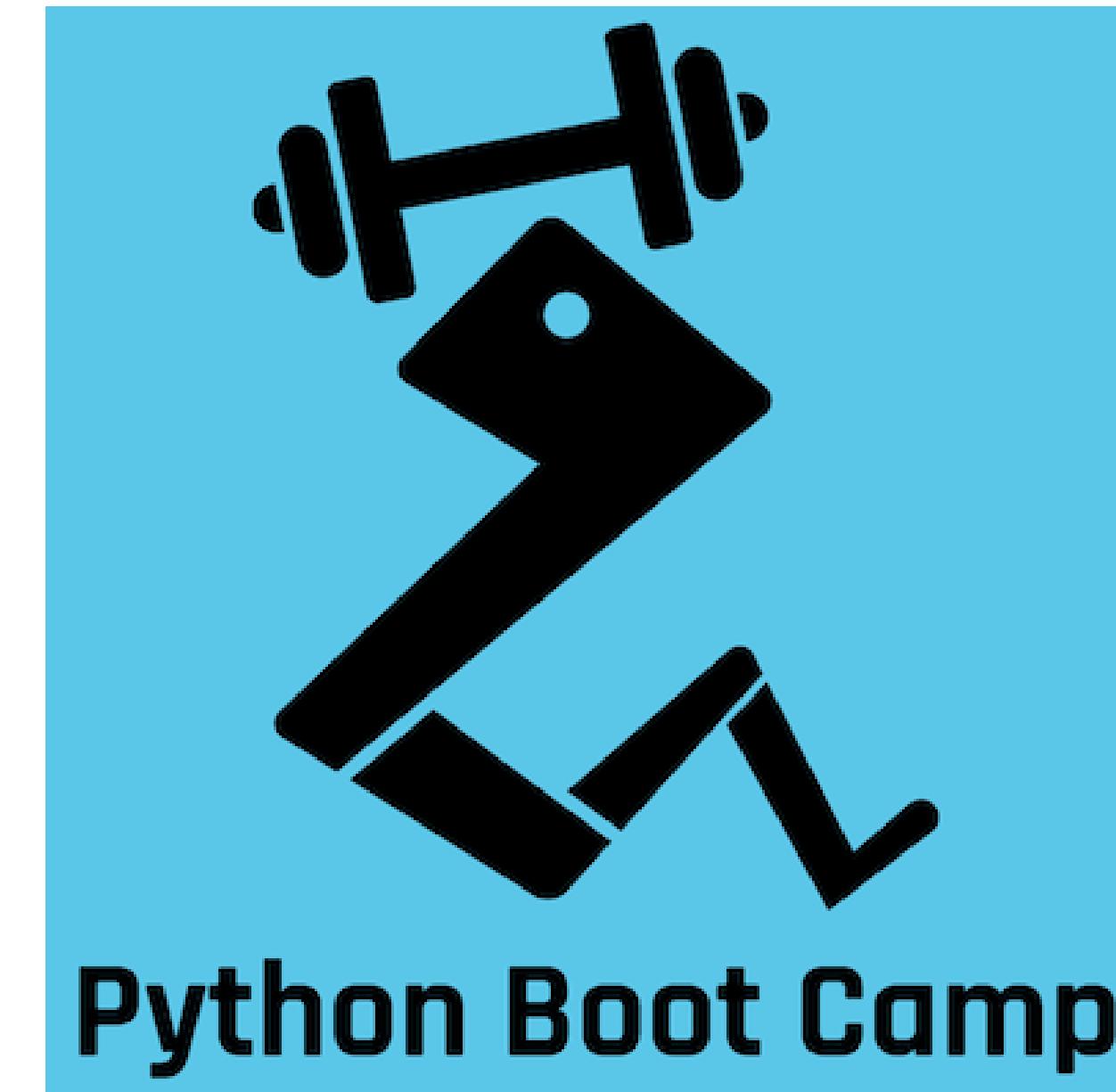
- 国内最大のPythonイベント
- pycon.jp/2020
- カンファレンス: 2020年8月28日(金)、29日(土)
- 会場: オンライン

PyCon JP 2020の様子

- クロージング後の集合写真

Python Boot Camp

- 初心者向け Python チュートリアル
- www.pycon.jp/support/bootcamp



Python Boot Camp(初心者向けP X)

pycon.jp/support/bootcamp.html

40 Python Boot Camp in 京都市 12月7日(土) ピュアイレッジ 照口云館 innovation 那ノ内ガーデン 10名 開催レポート

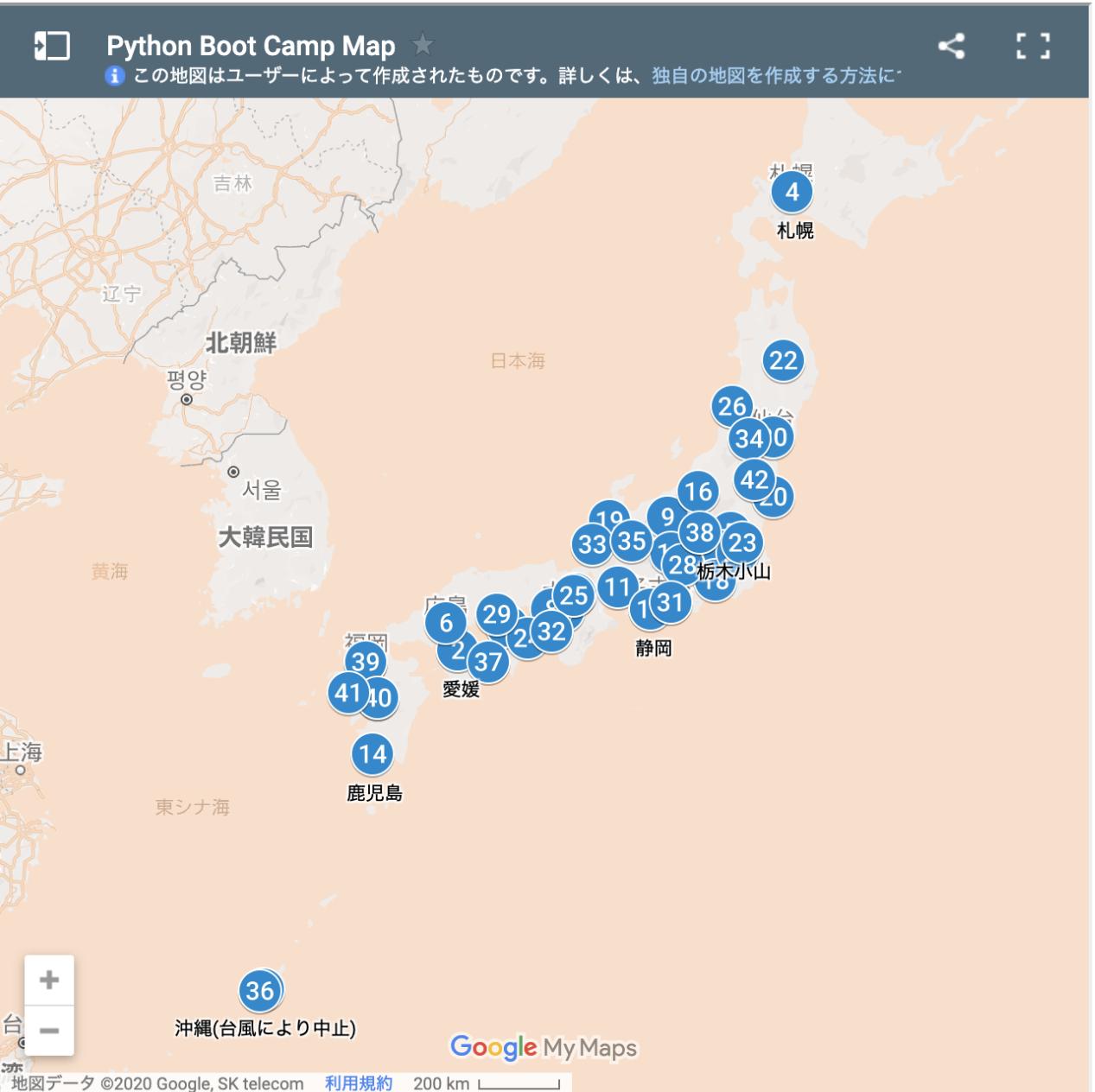
41 Python Boot Camp in 長崎 2月8日(土) ミライON図書館 寺田 学 18名 4 1 開催レポート

42 Python Boot Camp in 福島県郡山市 (新型コロナウイルスの影響により中止) 3月14日(土) 市民交流プラザ 第3会議室 清水川 貴之 25名(予定) 7 5

2020年

Python Boot Camp Map

Python Boot Camp Map ★
この地図はユーザーによって作成されたものです。詳しくは、独自の地図を作成する方法について



The map displays numerous blue circular markers, each containing a number representing a different Python Boot Camp event. The events are located across Japan, including major cities like Tokyo, Nagoya, and Sapporo, as well as smaller towns and islands. Some events are also marked outside Japan, such as in Seoul, Pyongyang, and Shanghai. The markers are distributed along coastlines and inland, indicating the variety of locations for these boot camps.

v: latest

コミュニケーション用Slack

- PyCon JP Fellow Slack
- pyconjp-fellow.herokuapp.com

Python Charity Talks

- pyconjp.connpass.com/event/177586
- 2020年7月4日(土) 13:00-18:00
- オンラインイベント
- 全額Python Software Foundation(PSF)に寄付

Jul 4 Python Charity Talks in Japan

Organizing : 一般社団法人PyCon JP Association

python Conference Japan

Hashtag : #pycharity

Registration info

	FCFS
スポンサー(Platinum) ¥300000 (Pre-pay)	1/1
スポンサー(Gold) ¥100000 (Pre-pay)	4/4
スポンサー(Silver) ¥30000 (Pre-pay)	7/10
スポンサー(Patron) ¥5000 (Pre-pay)	24/30
参加者(一般) ¥1000 (Pre-pay)	89/150
参加者(学生) ¥500 (Pre-pay)	9/50
スタッフ ¥1000 (Pre-pay)	8/8
スポンサー(Gold)請求書払い Free	1/1

Group Join Group

PyCon JP
Python Conference Japan

Number of events 182
Members 7468

Public 2020/07/04(Sat)
13:00 ~ 18:00

[Google Calendar](#) [iCal](#)

Please login to register

[Login/Sign Up](#)

Registration Period
2020/05/25(Mon) 20:39 ~
2020/07/04(Sat) 11:00

[Event Organizer Contact](#)

Location オンライン
Privacy - Terms

ミーティングにも来てね

- D会場で14:00～14:45
- セミナーの質疑応答
- Python Boot Campについてなど

Python開発環境の整え方

- Pythonの環境について
- 開発環境を整える方法
- デバッグ方法

Pythonインストール

Pythonインストール

- www.python.org/downloadsから公式版をインストール
 - WindowsはPATH設定のチェックを忘れずに
- 最新バージョンは3.9.0 / 3.8.6
 - 3.9.0: 2020年10月5日リリース
 - サードパーティ製パッケージの対応を確認

Download Python | Python.org +

python.org/downloads/ Guest :

python™

Donate Search GO Socialize

About Downloads Documentation Community Success Stories News Events

Download the latest version for Windows

[Download Python 3.8.5](#)

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [Mac OS X](#), [Other](#)

Want to help test development versions of Python? [Prereleases](#), [Docker images](#)

Looking for Python 2.7? See below for specific releases



A large blue banner at the top features the Python logo and navigation links. Below it, a central section is titled "Download the latest version for Windows" with a yellow button for Python 3.8.5. It also provides links for other operating systems and developer resources. To the right is a cartoon illustration of two cardboard boxes with yellow and white striped parachutes falling through a blue sky with white clouds.

Active Python Releases

For more information visit the Python Developer's Guide.

Python version	Maintenance status	First released	End of support	Release schedule
3.8	bugfix	2019-10-14	2024-10	PEP 569
3.7	security	2018-06-27	2023-06-27	PEP 537
3.6	security	2016-12-23	2021-12-23	PEP 494
3.5	security	2015-09-13	2020-09-13	PEP 478
2.7	end-of-life	2010-07-03	2020-01-01	PEP 373

環境構築ガイド

- www.python.jp/install/install



Home ニュース 求人情報 Contact

環境構築ガイド



Windows



macOS



Ubuntu



CentOS



Anaconda

他の手段

- Anaconda: www.anaconda.com
- pyenv: github.com/pyenv/pyenv
- Homebrew: brew.sh
- Linux:
 - Ubuntu 20.04LTS: Python 3.8.2

Pythonインストール

- 特に理由がなければ公式版を使おう

仮想環境(venv)

仮想環境(venv)

- docs.python.org/ja/3/library/venv
- Pythonに標準でついてくる
- 常に使おう
- プロジェクト単位で使用パッケージを変えられる

venv --- 仮想環境の作成 — Python 3.8.3 documentation

Python » Japanese 3.8.3 Documentation » Python 標準ライブラリ » ソフトウェア・パッケージと配布 » 前へ | 次へ | モジュール | 索引 | クイック検索 | 検索

目次

- venv --- 仮想環境の作成
 - 仮想環境の作成
 - API
 - EnvBuilder を拡張する例

前のトピックへ

[ensurepip --- pip インストーラのブートストラップ](#)

次のトピックへ

[zipapp --- Manage executable Python zip archives](#)

このページ

[バグ報告](#)
[ソースコードを表示](#)

venv --- 仮想環境の作成

バージョン 3.3 で追加。

ソースコード: [Lib/venv/](#)

`venv` モジュールは、軽量な "仮想環境" の作成のサポートを提供します。仮想環境には、仮想環境ごとの `site` ディレクトリがあり、これはシステムの `site` ディレクトリから分離させることができます。それぞれの仮想環境には、それ自身に (この仮想環境を作成するのに使ったバイナリのバージョンに合った) Python バイナリがあり、仮想環境ごとの `site` ディレクトリに独立した Python パッケージ群をインストールできます。

Python 仮想環境に関してより詳しくは [PEP 405](#) を参照してください。

参考: [Python Packaging User Guide: Creating and using virtual environments](#)

仮想環境の作成

仮想環境を作成するには `venv` コマンドを実行します:

```
python3 -m venv /path/to/new/virtual/environment
```

このコマンドを実行すると、ターゲットディレクトリ (および必要なだけの親ディレクトリ) が作成され、その中に `pyvenv.cfg` ファイルが置かれます。そのファイルの `home` キーはこのコマンドを呼び出した Python のインストール場所を指します (よく使われるターゲットディレクトリの名前は `.venv` です)。このコマンドはまた、Python バイナリのコピーまたはシンボリックリンク (のプラットフォームあるいは仮想環境作成時に使われた引数に対して適切な方) を含む `bin` (Windows では `Scripts`) サブディレクトリを作成します。さらに、`lib/pythonX.Y/site-packages` (Windows では `Lib\site-packages`) サブディレクトリも (最初は空の状態で) 作成します。指定したディレクトリが存在している場合は、それが再利用されます。

バージョン 3.6 で非推奨: Python 3.3 と 3.4 では、仮想環境の作成に推奨していたツールは `pyvenv` でしたが、[Python 3.6 では非推奨です](#)。

バージョン 3.5 で変更: 仮想環境の作成には、`venv` の使用をお勧めします。

Windows では、`venv` コマンドは次のように実行します:

venvの概念

- システムのPython
 - venv1
 - Django 3.0.7
 - venv2
 - Django 2.2.13

venv作成、有効化

- macOS、Linux
 - `python3.8 -m venv 環境名`
 - `source 環境名/bin/activate`

```
$ python3.8 -m venv env
$ source env/bin/activate
(env) $
```

venv作成、有効化

- Windows

- py -3.8 -m venv 環境名
- 環境名\Scripts\activate.ps1
- 環境名\Scripts\activate.bat

```
> py -3.8 -m venv env
> env\Scripts\activate.ps1
(env) >
```

venvとは

- パスを書き換えているだけ
- 削除するときはフォルダーごと削除

venv無効化

- deactivate

```
(env) $ deactivate  
$
```

```
(env) > deactivate  
>
```

仮想環境(venv)

- パッケージをインストールするときは常に使おう

パッケージ管理(pip)

パッケージ管理(pip)

- pip.pypa.io
- サードパーティ製パッケージの管理ツール
- venvの中で使用するのが一般的

pip - The Python Package Insta X +
pip.pypa.io/en/stable/

PyPA » pip 20.1.1 documentation » Quick search Go | next

Table Of Contents

- Quickstart
- Installation
- User Guide
- Reference Guide
- Development
- Changelog

pip – The Python Package Installer

pip is the [package installer](#) for Python. You can use pip to install packages from the [Python Package Index](#) and other indexes.

Please take a look at our documentation for how to install and use pip:

- [Quickstart](#)
- [Installation](#)
- [User Guide](#)
- [Reference Guide](#)
- [Development](#)
- [Changelog](#)

In 2020, we're working on improvements to the heart of pip. Please [learn more and take our survey](#) to help us do it right.

« If you find bugs, need help, or want to talk to the developers, please use our mailing lists or chat rooms:

- [Issue tracking](#)
- [Discourse channel](#)
- [User IRC](#)

If you want to get involved, head over to GitHub to get the source code, and feel free to jump on the developer mailing lists and chat rooms:

- [GitHub page](#)
- [Development mailing list](#)
- [Development IRC](#)

Code of Conduct

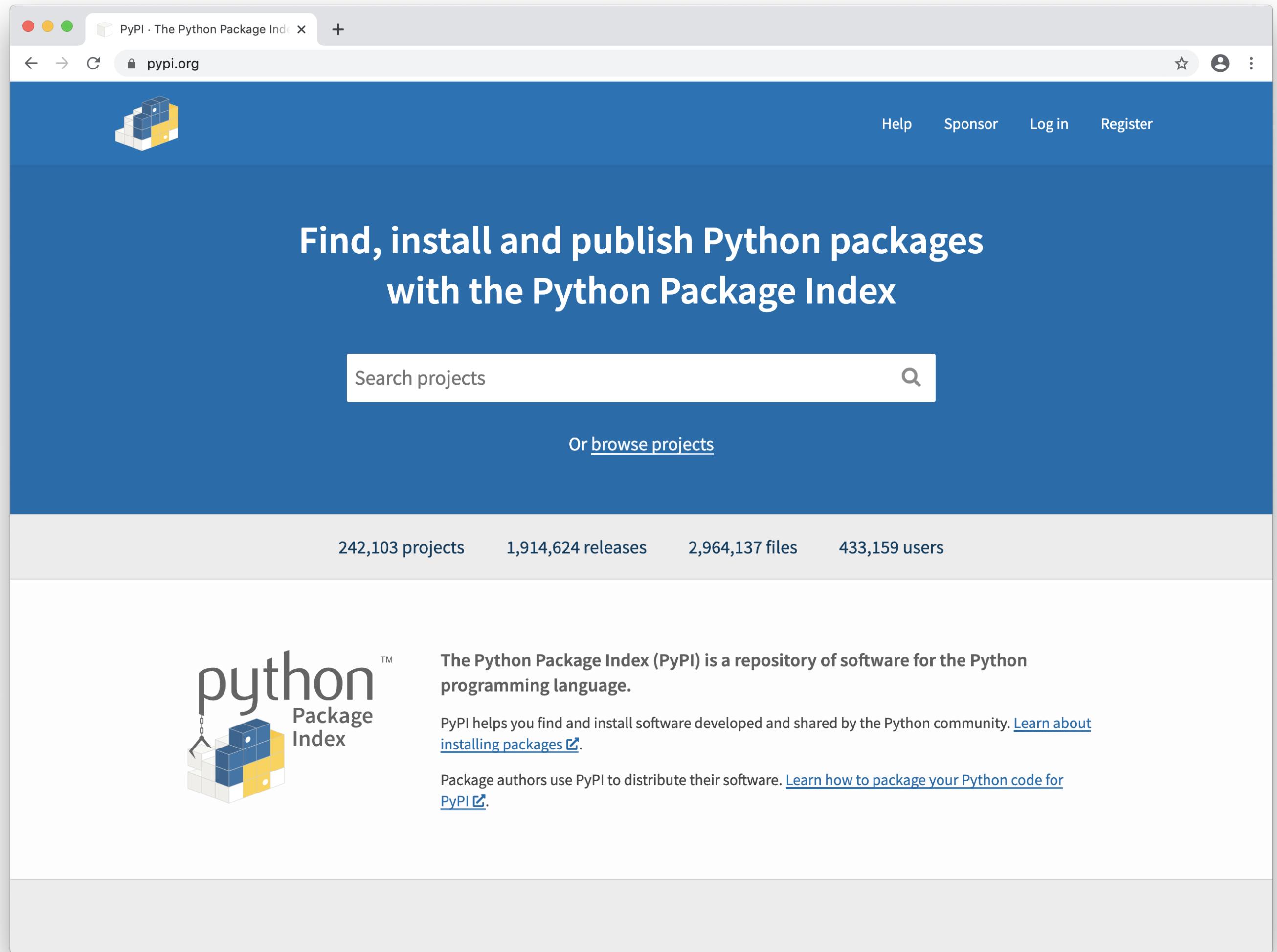
Everyone interacting in the pip project's codebases, issue trackers, chat rooms, and mailing lists is expected to follow the [PyPA Code of Conduct](#).

v: stable ▾

Sponsored Protect your online privacy and access your favorite content with NordVPN ads served ethically
Establishing secure connection...

PyPI

- pypi.org
- サードパーティー製パッケージの配布サイト
- 「パイピーアイ」と読むらしい



The screenshot shows the PyPI (Python Package Index) homepage. At the top, there's a navigation bar with links for Help, Sponsor, Log in, and Register. Below the header is a large blue banner with the text "Find, install and publish Python packages with the Python Package Index". A search bar with the placeholder "Search projects" and a magnifying glass icon is positioned below the banner. Underneath the search bar is a link "Or [browse projects](#)". At the bottom of the main content area, there are statistics: "242,103 projects", "1,914,624 releases", "2,964,137 files", and "433,159 users". The footer contains the PyPI logo and a brief description: "The Python Package Index (PyPI) is a repository of software for the Python programming language. PyPI helps you find and install software developed and shared by the Python community. [Learn about installing packages](#). Package authors use PyPI to distribute their software. [Learn how to package your Python code for PyPI](#)".

PyPI · The Python Package Index

pypi.org

Help Sponsor Log in Register

Find, install and publish Python packages with the Python Package Index

Search projects

Or [browse projects](#)

242,103 projects 1,914,624 releases 2,964,137 files 433,159 users

python™
Package Index

The Python Package Index (PyPI) is a repository of software for the Python programming language.

PyPI helps you find and install software developed and shared by the Python community. [Learn about installing packages](#).

Package authors use PyPI to distribute their software. [Learn how to package your Python code for PyPI](#).

pipでインストール

- pip install パッケージ名
- 依存パッケージもまとめてインストール

```
(env) $ pip install beautifulsoup4
Collecting beautifulsoup4
  Downloading beautifulsoup4-4.9.1-py3-none-any
    |████████████████████████████████| 115 kB :
```

```
Collecting soupsieve>1.2
  Downloading soupsieve-2.0.1-py3-none-any.whl
Installing collected packages: soupsieve, beaut
Successfully installed beautifulsoup4-4.9.1 sou
```

pipでインストール

- バージョン指定
 - pip install パッケージ名==バージョン
- 最新にアップグレード
 - pip install -U パッケージ名

```
(env) $ pip install beautifulsoup4==4.8.0
: (省略)
```

```
Successfully uninstalled beautifulsoup4-4
```

```
Successfully installed beautifulsoup4-4.8.0
```

```
(env) $ pip install -U beautifulsoup4
: (省略)
```

```
Successfully installed beautifulsoup4-4.9.1
```

パッケージ一覧を表示

- pip list

```
(env) $ pip install beautifulsoup4==4.8.0
```

: (省略)

```
(env) $ pip list
```

Package	Version
---------	---------

beautifulsoup4	4.8.0
----------------	-------

setuptools	41.2.0
------------	--------

soupsieve	2.0.1
-----------	-------

古いパッケージを確認

- pip list -o

```
(env) $ pip list -o
```

Package	Version	Latest	Type
beautifulsoup4	4.8.0	4.9.1	wheel
setuptools	41.2.0	47.3.1	wheel

パッケージ一覧を再利用 (1/2)

- pip freeze > requirements.txt
- requirements.txtファイルをバージョン管理
 - (env) \$ pip freeze > requirements.txt
 - (env) \$ cat requirements.txt
 - beautifulsoup4==4.8.0
 - soupsieve==2.0.1

パッケージ一覧を再利用 (2/2)

- 別の仮想環境でインストール
 - `pip install -r requirements.txt`

```
$ python3.8 -m venv newenv
$ source newenv/bin/activate
(newenv) $ pip install -r requirements.txt
(newenv) $ pip freeze
beautifulsoup4==4.8.0
soupsieve==2.0.1
```

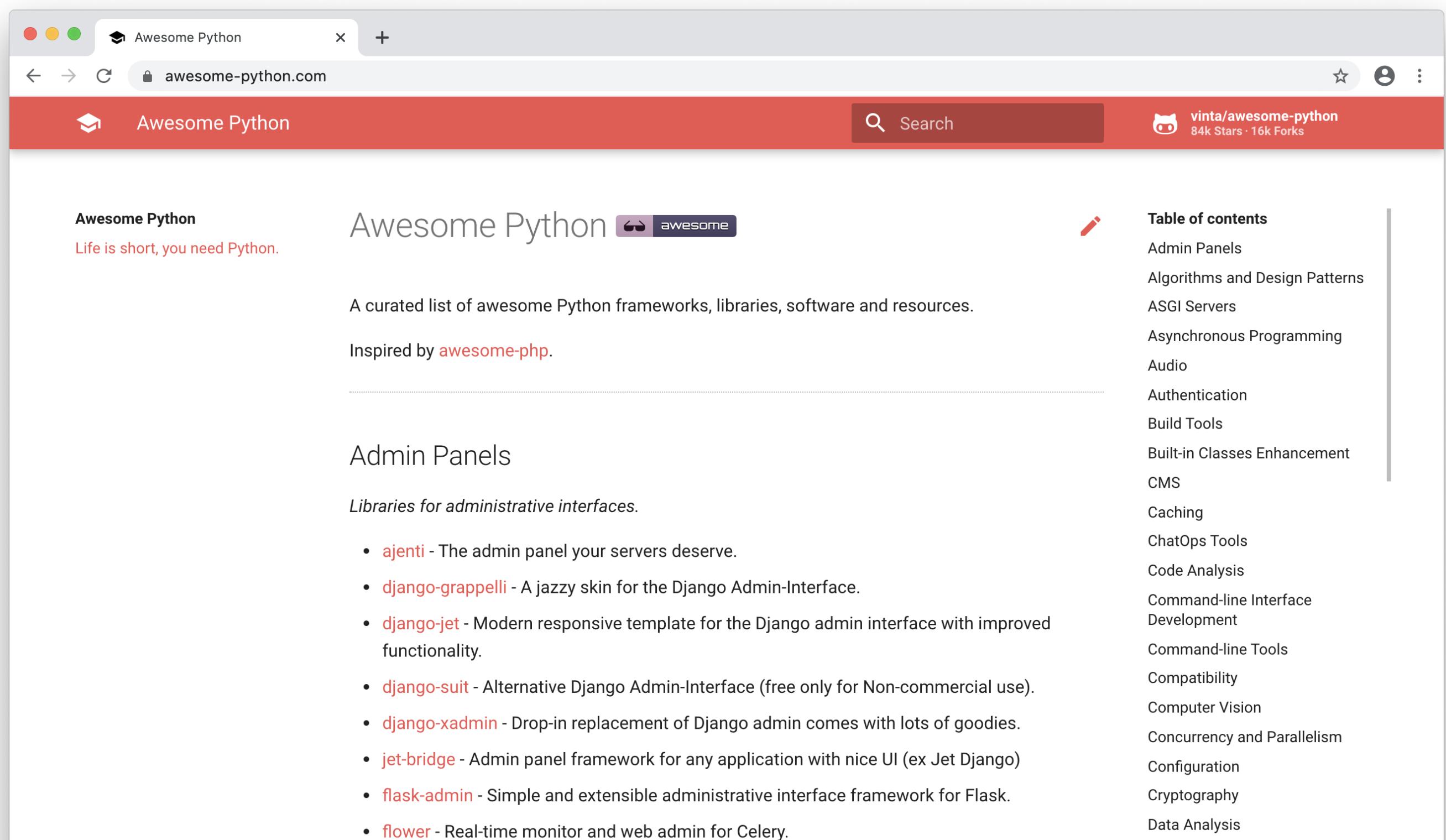
アンインストール

- pip uninstall -y パッケージ名
- 依存パッケージは削除されない

```
(env) $ pip uninstall -y beautifulsoup4
Found existing installation: beautifulsoup4 4.8
Uninstalling beautifulsoup4-4.8.0:
  Successfully uninstalled beautifulsoup4-4.8.0
```

パッケージの探し方

- Awesome Python: awesome-python.com



The screenshot shows a web browser displaying the Awesome Python website. The URL in the address bar is `awesome-python.com`. The page has a red header with the title "Awesome Python" and a search bar. On the right side of the header, there's a GitHub icon and the repository information: `vinta/awesome-python`, 84k Stars · 16k Forks. The main content area features a heading "Admin Panels" under the "Libraries for administrative interfaces." section. A list of packages includes `ajenti`, `django-grappelli`, `django-jet`, `django-suit`, `django-xadmin`, `jet-bridge`, `flask-admin`, and `flower`. To the right of the content, there's a "Table of contents" sidebar listing various Python categories such as Admin Panels, Algorithms and Design Patterns, ASGI Servers, etc.

Awesome Python

Life is short, you need Python.

Awesome Python

A curated list of awesome Python frameworks, libraries, software and resources.

Inspired by [awesome-php](#).

Admin Panels

Libraries for administrative interfaces.

- [ajenti](#) - The admin panel your servers deserve.
- [django-grappelli](#) - A jazzy skin for the Django Admin-Interface.
- [django-jet](#) - Modern responsive template for the Django admin interface with improved functionality.
- [django-suit](#) - Alternative Django Admin-Interface (free only for Non-commercial use).
- [django-xadmin](#) - Drop-in replacement of Django admin comes with lots of goodies.
- [jet-bridge](#) - Admin panel framework for any application with nice UI (ex Jet Django)
- [flask-admin](#) - Simple and extensible administrative interface framework for Flask.
- [flower](#) - Real-time monitor and web admin for Celery.

Table of contents

- Admin Panels
- Algorithms and Design Patterns
- ASGI Servers
- Asynchronous Programming
- Audio
- Authentication
- Build Tools
- Built-in Classes Enhancement
- CMS
- Caching
- ChatOps Tools
- Code Analysis
- Command-line Interface
- Development
- Command-line Tools
- Compatibility
- Computer Vision
- Concurrency and Parallelism
- Configuration
- Cryptography
- Data Analysis

パッケージ管理(pip)

- pipコマンドを使いこなそう

テキストエディター

テキストエディター

- 好きな物を使いましょう
 - シンタックスハイライト
 - TABでスペース4つ
- とくになければ以下がおすすめ
 - VSCode: code.visualstudio.com
 - PyCharm: www.jetbrains.com/pycharm

Visual Studio Code - Code Edit + |

code.visualstudio.com

This site uses cookies for analytics, personalized content and ads. By continuing to browse this site, you agree to this use. [Learn more](#)

Visual Studio Code Docs Updates Blog API Extensions FAQ [Search Docs](#) [Download](#)

Version 1.46 is now available! Read about the new features and fixes from May.

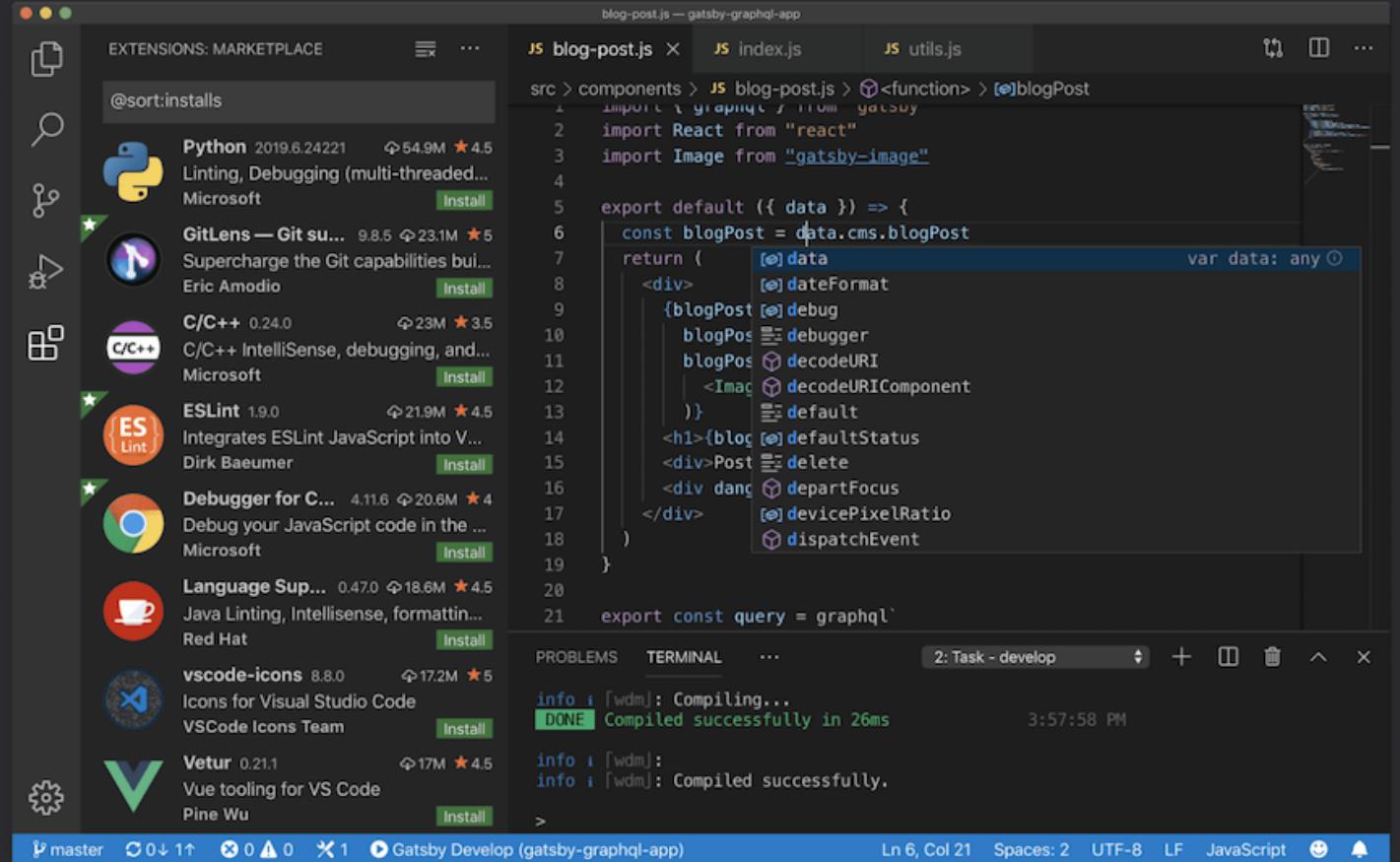
Code editing. Redefined.

Free. Built on open source. Runs everywhere.

[Download for Mac](#)
Stable Build

Other platforms and Insiders Edition

By using VS Code, you agree to its license and privacy statement.



The screenshot shows the Visual Studio Code interface. The main area displays a code editor with a file named 'blog-post.js' containing JavaScript code for a Gatsby blog post component. The code includes imports for React and gatsby-image, and a function that returns a JSX element. To the left of the editor is the 'EXTENSIONS: MARKETPLACE' sidebar, which lists several popular extensions: Python, GitLens, C/C++, ESLint, Debugger for Chrome, Language Support, vscode-icons, and Vetur. The bottom of the screen shows the terminal output, which shows a successful compilation of the code. The status bar at the bottom indicates the current branch is 'master' and the file is 'Gatsby Develop (gatsby-graphql-app)'.

IntelliSense

Run and Debug

Built-in Git

Extensions

50 / 87

PyCharm: the Python IDE for Professional Developers

jetbrains.com/pycharm/

JET BRAINS **20**
years

Tools Languages Solutions Support Company Store  

Coming in 2020.2 What's New Features Learning Center Buy **Download**

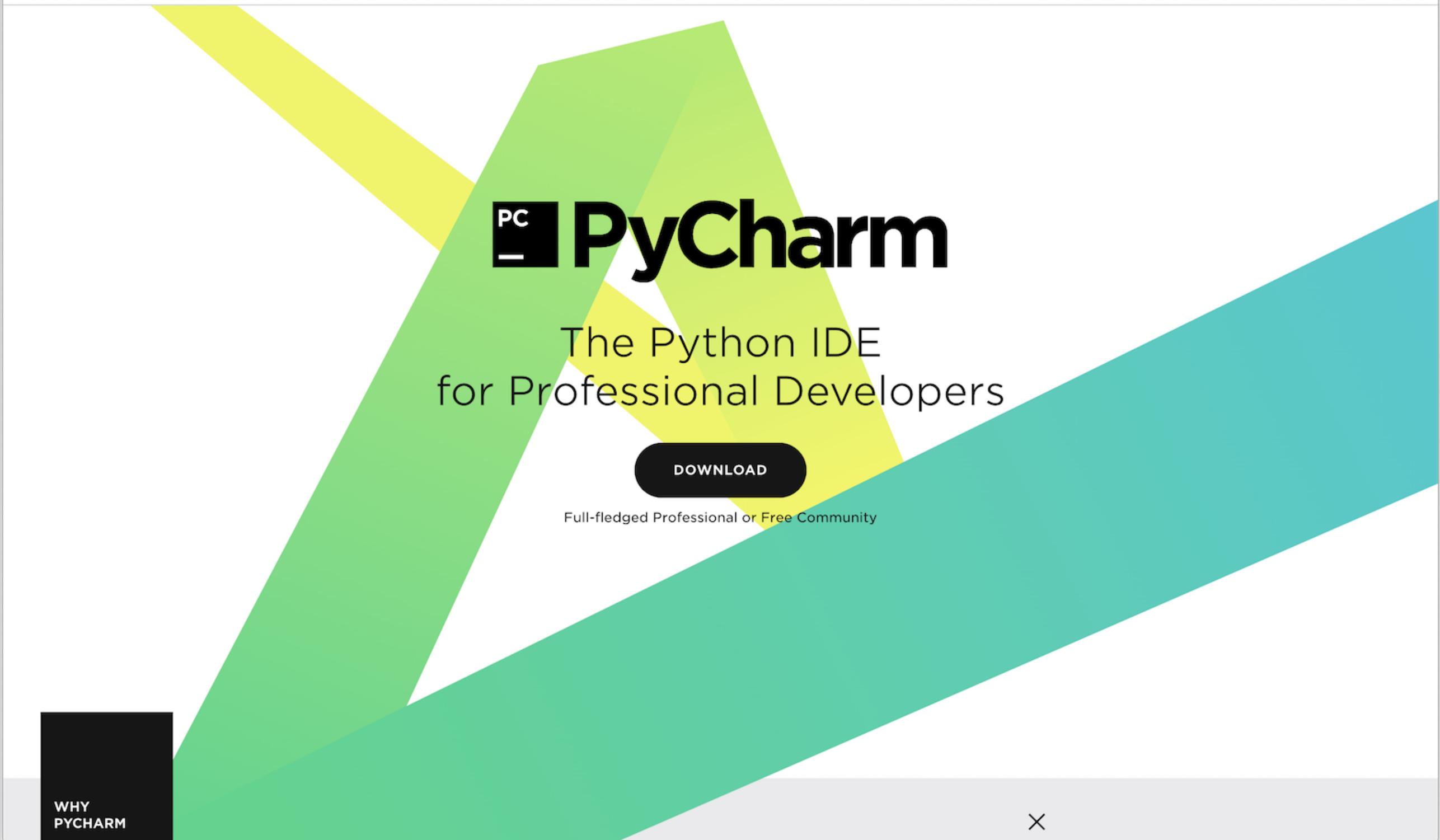
PC PyCharm

The Python IDE
for Professional Developers

DOWNLOAD

Full-fledged Professional or Free Community

WHY PYCHARM



X

静的チェック(Flake8)

静的チェック(Flake8)

- flake8.pycqa.org
- コードを静的にチェックするコマンド
 - `pycodestyle`: PEP8のチェック
 - `pyflakes`: エラーが発生しそうなコードをチェック
 - `mccabe`: 循環的複雑度のチェック

Flake8: Your Tool For Style Guide Enforcement [Edit on GitHub](#)

Flake8: Your Tool For Style Guide Enforcement

Quickstart

Installation

To install **Flake8**, open an interactive shell and run:

```
python<version> -m pip install flake8
```

If you want **Flake8** to be installed for your default Python installation, you can instead use:

```
python -m pip install flake8
```

Note
It is very important to install **Flake8** on the correct version of Python for your needs. If you want **Flake8** to properly parse new language features in Python 3.5 (for example), you need it to be installed on 3.5 for **Flake8** to understand those features. In many ways, **Flake8** is tied to the version of Python on which it runs.

Using Flake8

To start using **Flake8**, open an interactive shell and run:

```
flake8 path/to/code/to/check.py  
# or  
flake8 path/to/code/
```

Note

Read the Docs v: latest ▾

PEP8

- PEP 8 -- Style Guide for Python Code
 - www.python.org/dev/peps/pep-0008/
 - pep8-jp.readthedocs.io: 日本語訳
- Python標準のスタイルガイド
- 主なルール
 - インデントはスペース4つ
 - 1行の最大文字数は79文字
 - 空行の入れ方
 - スペースの入れ方

PEP 8 -- Style Guide for Python X +

python.org/dev/peps/pep-0008/

Python PSF Docs PyPI Jobs Community

 python™

Donate  Search GO Socialize

About Downloads Documentation Community Success Stories News Events

Tweets by @ThePSF

Python Software Foundation ✅ @ThePSF

2020 Python Software Foundation Board of Directors Election Retrospective and Next Steps ift.tt/2YGpAjn

4h

Python Software Foundation ✅ @ThePSF

We've opened a thread over at discuss.python.org/t/psf-board-el... regarding @thepsf Board of Directors Election Reform and we'd love to hear from you!

Jun 26, 2020

PSF Board Election Refo... Greetings PSF community... discuss.python.org

Embed View on Twitter

Python »» Python Developer's Guide »» PEP Index »» PEP 8 -- Style Guide for Python Code

PEP 8 -- Style Guide for Python Code

PEP:	8
Title:	Style Guide for Python Code
Author:	Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com>
Status:	Active
Type:	Process
Created:	05-Jul-2001
Post-History:	05-Jul-2001, 01-Aug-2013

Contents

- [Introduction](#)
- [A Foolish Consistency is the Hobgoblin of Little Minds](#)
- [Code Lay-out](#)
 - [Indentation](#)

はじめ — pep8-ja 1.0 ドキュメント +
← → ⌛ 🔒 pep8-ja.readthedocs.io/ja/latest/ ⌂ ☆ ⌂ ⌂

Docs » はじめに [Edit on GitHub](#)

PEP: 8

Title: Python コードのスタイルガイド

Version: \$Revision\$

Last-Modified: \$Date\$

Author: Guido van Rossum <guido@python.org>,
Barry Warsaw <barry@python.org>,
Nick Coghlan <ncoghlan@gmail.com>

Status: Active

Type: Process

Content-Type: text/x-rst

Created: 05-Jul-2001

Post-History: 05-Jul-2001, 01-Aug-2013

X-Original-Text: <https://hg.python.org/peps/file/tip/pep-0008.txt>

X-Translator: Yoshinari Takaoka <[reversethis -> gro tod umumum ta umumum](mailto:reversethis@gro.tod.umumum.tumumum)>

はじめに

この文書は Python の標準ライブラリに含まれている Python コードのコーディング規約です。CPython に含まれる C 言語のコード [1] については、対応する C 言語のスタイルガイドを記した PEP を参照してください。

この文書と PEP 257 (Docstring 規約) は、Guido が書いたオリジナルの Python スタイルガイドのエッセイと、Barry のスタイルガイドに少し追記したものをまとめたものです。[2]

このスタイルガイドは、追加の規約が必要だとわかったり、Python の言語自体が変更されることで過去の規約が時代遅れになった時に徐々に改訂されてゆきます。

多くのプロジェクトには、自分たちのコーディングスタイルに関するガイドラインがあります。それとこの文書の規約の内容が矛盾した場合は、そのプロジェクトのガイドラインが優先します。

一貫性にこだわりすぎるのは、狭い心の現れである


Troubleshoot your Python app performance to help identify bottlenecks. Try Datadog APM free today.

Sponsored · Ads served ethically

Read the Docs v: latest

Flake8をインストール

- pip install flake8

だめなコード

```
from random import *

def add(a,b):
    c = choice([a, b]) # 意味のない処理
    return a + b
```

Flake8を実行

- flake8 ファイル名

```
$ flake8 sample.py
sample.py:1:1: F403 'from random import *' used
sample.py:3:1: E302 expected 2 blank lines, fou
sample.py:3:10: E231 missing whitespace after '
sample.py:4:5: F841 local variable 'c' is assig
sample.py:4:9: F405 'choice' may be undefined,
sample.py:4:23: E261 at least two spaces before
```

コードをきれいにする

```
from random import choice
```

```
def add(a, b):  
    c = choice([a, b])    # 意味のない処理  
    return a + b, c
```

Flake8を再実行

- エラーメッセージが出なくなった!!

```
$ flake8 sample.py  
$
```

Flake8とエディター

- 多くのエディターから直接チェックできる
- PyCharm: External Toolsで設定
- VSCode: Python Extension→設定変更
 - デフォルトのPylintは制限が強い
- Emacs: flycheck, flymake
- Vim: vim-flake8

静的チェック(Flake8)

- Flake8を通るコードにしよう

コード整形(Black)

コード整形(Black)



コード整形(Black)

- black.readthedocs.io
- The uncompromising code formatter
 - 頑固なコードフォーマッター
- PEP8準拠のコードにしてくれる
- 設定はほぼなし
 - Blackの流儀に合わせる

The uncompromising code formatter

By using *Black*, you agree to cede control over minutiae of hand-formatting. In return, *Black* gives you speed, determinism, and freedom from *pycodestyle* nagging about formatting. You will save time and mental energy for more important matters.

Black makes code review faster by producing the smallest diffs possible. Blackened code looks the same regardless of the project you're reading. Formatting becomes transparent after a while and you can focus on the content instead.

Try it out now using the [Black Playground](#).

Note:

[Black is beta.](#)

Testimonials

Dusty Phillips, writer:

Black is opinionated so you don't have to be.

Hynek Schlawack, creator of `attrs`, core developer of Twisted and CPython:

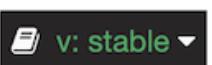
An auto-formatter that doesn't suck is all I want for Xmas!

Carl Meyer, Django core developer:

At least the name is good.

Kenneth Reitz, creator of `requests` and `pipenv`:

This vastly improves the formatting of our code. Thanks a ton!

 v: stable ▾

Contents



"Any color you like."

 Star 16,524

 build passing

Navigation

- [Installation and usage](#)
- [The *Black* code style](#)
- [pyproject.toml](#)
- [Editor integration](#)
- [blackd](#)
- [Version control](#)
- [integration](#)
- [Ignoring unmodified files](#)
- [Contributing to *Black*](#)
- [Show your style](#)
- [Change Log](#)
- [Developer reference](#)
- [Authors](#)

This Page

[Show Source](#)

Quick search

Go



Blackをインストール

- pip install black

だめなコード

```
from random import *

def add(a,b):
    c = choice([a, b]) # 意味のない処理
    return a + b
```

```
$ black --diff sample.py
--- sample.py      2020-06-26 11:20:43.268988 +00
+++ sample.py      2020-06-26 11:22:20.716557 +00
@@ -1,6 +1,7 @@
 from random import *

-def add(a,b):
-    c = choice([a, b]) # 意味のない処理
+
+def add(a, b):
+    c = choice([a, b]) # 意味のない処理
```

Blackでファイルを更新

- black ファイル名

```
$ black sample.py
$ cat sample.py
from random import *
```

```
def add(a, b):
    c = choice([a, b])    # 意味のない処理
    return a + b
```

コード整形(Black)

- Blackでコードを整形しよう

デバッグ(pdb)

デバッグ(pdb)

- [docs.python.org/ja/3/library/pdb](https://docs.python.org/ja/3/library/pdb.html)
- Pythonの標準ライブラリ
- 対話型のデバッガ

pdb --- Python デバッガ — Pyt

docs.python.org/ja/3/library/pdb.html

Python » Japanese 3.8.3 Documentation » Python 標準ライブラリ » デバッグとプロファイル » クイック検索 検索 | 前へ | 次へ | モジュール | 索引

目次

- [pdb --- Python デバッガ](#)
 - デバッガコマンド

前のトピックへ

- [faulthandler --- Python tracebackのダンプ](#)

次のトピックへ

- [Python プロファイル](#)

このページ

- [バグ報告](#)
- [ソースコードを表示](#)

pdb --- Python デバッガ

ソースコード: [Lib/pdb.py](#)

モジュール `pdb` は Python プログラム用の対話型ソースコードデバッガを定義します。 (条件付き)ブレークポイントの設定やソース行レベルでのシングルステップ実行、スタックフレームのインスペクション、ソースコードリストティングおよびあらゆるスタックフレームのコンテキストにおける任意の Python コードの評価をサポートしています。事後解析デバッギングもサポートし、プログラムの制御下で呼び出すことができます。

デバッガーは拡張可能です -- 実際にはクラス `Pdb` として定義されています。現在これについてのドキュメントはありませんが、ソースを読めば簡単に理解できます。拡張インターフェースはモジュール `bdb` と `cmd` を使っています。

デバッガのプロンプトは (`Pdb`) です。デバッガに制御された状態でプログラムを実行するための典型的な使い方は以下のようになります:

```
<<
>>> import pdb
>>> import mymodule
>>> pdb.run('mymodule.test()')
> <string>(0)?()
(Pdb) continue
> <string>(1)?()
(Pdb) continue
NameError: 'spam'
> <string>(1)?()
(Pdb)
```

バージョン 3.3 で変更: `readline` モジュールによるコマンドおよびコマンド引数のタブ補完が利用できます。たとえば、`p` コマンドの引数では現在のグローバルおよびローカル名が候補として表示されます。

他のスクリプトをデバッグするために、 `pdb.py` をスクリプトとして呼び出さることもできます。例えば:

```
python3 -m pdb myscript.py
```

スクリプトとして `pdb` を起動すると、デバッグ中のプログラムが異常終了したときに `pdb` が自動的に事後デバッグモードに入ります。事後デバッグ後（またはプログラムの正常終了後）、`pdb` はプログラムを再起動します。自動再起動を行った場合、`pdb` の状態（ブレークポイントなど）はそのまま維持されるので、たいていの場合、プログラム終了時にデバッガーも終了させるよりも便利なはずです。

pdbを起動

- `python3 -m pdb プログラム.py`
- `breakpoint()` 関数

```
$ python3 -m pdb sample.py
/path/to/sample.py(1)<module>()
-> from random import choice
(Pdb) n
/path/to/sample.py(4)<module>()
-> def add(a, b):
```

pdbの主なコマンド

- **h(elp)**: ヘルプを表示
- **b(reak) 行番号**: ブレイクポイントを設定
- **n(ext)**: 次の行に移動
- **c(ont(inue))**: ブレイクポイントまで実行
- **l(ist)**: コードを表示
- **p 式**: 式を評価した結果を出力
- **q(uit)**: デバッガを終了

デバッグ(pdb)

- `print()` デバッグより効率的

その他

その他

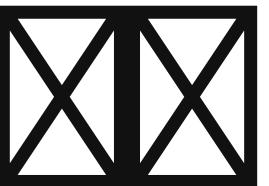
- テストコード: [unittest](#), [pytest](#)
- テスト環境: [tox](#)
- 型ヒント: [typing](#)
- 型ヒントチェック: [mypy](#)

まとめ

まとめ

- Python公式版
- venvで仮想環境
- pipでパッケージ管理
- テキストエディター
- Flake8で静的チェック
- Blackでコード整形
- pdbでデバッグ

ありがとうございました



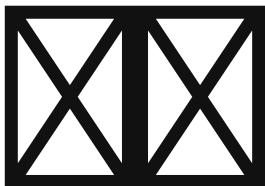
-  [@takanory](https://twitter.com/takanory)
-  github.com/pyconjp/slides

ミーティングにも来てね

- D会場で14:00～14:45
- セミナーの質疑応答など

Question?

ありがとうございました



-  [@takanory](https://twitter.com/takanory)
-  github.com/pyconjp/slides