仮想環境

Python を使って開発や実験を行うときは、用途に応じて専用の実行環境を作成し、切り替えて使用するのが一般的です。こういった、一時的に作成する実行環境を、**「仮想環境」** と言います。

仮想環境は、次のような目的で使われます。

- システム全体で使うPython環境に影響を与えずにモジュールの追加・入れ替えをしたい。
- 異なるバージョンの Python を使いわけたり、同じモジュールの、複数のバージョンを使い分けたい。

  例えば、開発中のWebアプリケーション開発では、Python 3.7 で Webアプリケーションフレームワークとして **Django** の 1.10 を使い、新しいプロジェクトでは Python 3.8 とDjango バージョン 1.11 を使用したい場合など、簡単に切り替えられるようにしたい。

ここでは、ここでは、Python3 の標準ライブラリである **venv** で仮想環境を作成する方法を紹介します。

プロジェクトディレクトリの作成

まず、開発対象のプロジェクトを格納するディレクトリを作成します。

コマンドプロンプトを開き、次のコマンドでディレクトリ `sample1` を作成します。

```
$ mkdir sample1
```

次に、プロジェクトディレクトリに `sample.py` という名前のソースファイルを作成しましょう。

sample.py

```
import requests
print(requests.get("https://www.python.jp").text)
```

`sample.py` で使っている **requestsモジュール** は、Python標準のモジュールではなく、別途インストールしないと使えません。実行するとつぎのようにエラーになります。

```
$ cd sample1
$ python3 sample.py
Traceback (most recent call last):
  File "sample.py", line 1, in <module>
    import requests
ImportError: No module named requests
```

仮想環境の作成

では、最初の仮想環境を作成しましょう。`sample1` ディレクトリで、次のコマンドを実行します。

```
$ python3 -m venv .venv
```

このコマンドは、指定したディレクトリ `~/sample1/.venv` に仮想環境を作成します。仮想環境のディレクトリ名は、 `.venv` 以外でも、好きな名前をつけても大丈夫です。

仮想環境への切り替え

作成した仮想環境 `.venv` ディレクトリにある `bin/activate` を、 `.` または `source` コマンドで実行します

```
$ . .venv/bin/activate
(.venv) $
```

コマンド プロンプトの先頭に `(.venv)` と表示され、仮想環境で実行中であることを示します。

パッケージのインストール

仮想環境を使用中に `pip` モジュールでPyPIからパッケージをインストールすると、仮想環境にインストールされます。
`sample.py` で使っている、 `requests` モジュールをインストールしましょう。

```
(.venv) $ python3 -m pip install requests
```

ここで、さきほど作成した `sample.py` を実行してみましょう。

```
(.venv) $ python3 sample.py
```

こんどは、エラー無しで実行できるはずです。

インストールしたモジュールは、仮想環境内にのみ書き込まれ、元の Pythonや、他の仮想環境からは利用できません。

仮想環境のコピー

違うPCでも開発したり、複数人のチームで開発したりする時は、どこでも同じ仮想環境を使うようにする必要があります。

この場合、仮想環境(`.venv` ディレクトリ) をコピーして共有するのではなく、仮想環境にインストールされているパッケージの一覧を作成して、みんなで共有するようにします。

パッケージの一覧は、つぎのコマンドで作成できます。ここでは、 `requirements.txt` というファイルに一覧を作成しています。

```
(.venv) $ python3 -m pip freeze > requirements.txt
```

作成した `requirements.txt` ファイルは、ソースコードと一緒にgitなどに登録して一元管理しましょう。

新しく仮想環境を作成したり、 `requirements.txt` に新しいパッケージを追加したときには、次のコマンドで一括して仮想環境にパッケージをインストールします。

```
(.venv) $ python3 -m pip install -r requirements.txt
```

仮想環境の終了

仮想環境の使用を終え、通常の状態に復帰するときは、 `deactivate` コマンドを実行します。

```
(.venv) $ deactivate
$
```

仮想環境を終了すると、もう `requests` モジュールは使えなくなってしまいます。

```
$ python sample.py
Traceback (most recent call last):
  File "sample.py", line 1, in <module>
    import requests
ModuleNotFoundError: No module named 'requests'
```

Pythonを指定した仮想環境

複数のバージョンの Python をインストールしている環境では、使用する Python を指定して仮想環境を作成できます。

Python 3.7とPython 3.8がインストールされた環境で、Python 3.7の仮想環境を作成する場合は、`python3.7` を使って次のように指定します。ここでは、仮想環境をディレクトリ `py37env` に作成します。

```
$ python3.7 -m venv py37env
```

ここで作成した `py37env` を使用すると、python 3.7の仮想環境に切り替わります。

```
$ . py37env/bin/activate
$ python3
Python 3.7.8 (default, Jul  8 2020, 14:18:28)
[Clang 11.0.3 (clang-1103.0.32.62)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

同様に、Python 3.8の仮想環境は、つぎのコマンドで作成できます。

```
$ python3.8 -m venv py38env
```

`py38env` を使用すると、python 3.8の仮想環境に切り替わります。

```
$ . py38env/bin/activate
$ python3
Python 3.8.3 (default, Jul  8 2020, 14:27:55)
[Clang 11.0.3 (clang-1103.0.32.62)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

macOS環境のPython
macOS版Pythonのインストール
Pythonの実行
pip
仮想環境
プロジェクトディレクトリの作成
仮想環境の作成
仮想環境への切り替え
パッケージのインストール
仮想環境のコピー
仮想環境の終了
Pythonを指定した仮想環境

最新記事

Python 3.10の新機能(その11) その他の変更: Python3.10の新機能

Python 3.10の新機能(その10) Dataclassでslotsが利用可能に: Python3.10の新機能

Python 3.10の新機能(その9) zip()関数に引数 strict を追加: Python3.10の新機能

Python 3.10の新機能(その8) OpenSSL 1.1.1が必須に: Python3.10の新機能