Erstellen und Aktivieren von Python-Umgebungen mit verschiedenen Tools

Das Ziel dieses Dokuments ist es, die verschiedenen Möglichkeiten aufzuzeigen, wie Python-Umgebungen mit verschiedenen Tools erstellt und aktiviert werden können.

Grundgedanke

Es gibt verschiedene Tools, die verwendet werden können, um Python-Umgebungen zu erstellen und zu aktivieren. Jedes Tool hat seine eigenen Vor- und Nachteile, und die Wahl des richtigen Tools hängt von den spezifischen Anforderungen des Projekts und den persönlichen Vorlieben ab.

Durchführungsschritte

Verwendung von venv

Keine Installation notwendig. `venv` ist in Python 3.3 und höher enthalten.

Aktivieren der virtuellen Umgebung mit venv

1. Verwenden Sie den Befehl `python3 -m venv myenv`, um eine virtuelle Umgebung mit `venv` zu erstellen.
2. Um die virtuelle Umgebung zu aktivieren:
   - Unter Windows verwenden Sie den Befehl `myenv\Scripts\activate`.
   - Unter Unix oder MacOS verwenden Sie den Befehl `source myenv/bin/activate`.

Deaktivieren der virtuellen Umgebung mit venv Um die virtuelle Umgebung, die mit `venv` erstellt wurde, zu deaktivieren, kann der Befehl `deactivate` verwendet werden. Dieser Befehl funktioniert sowohl unter Windows als auch unter Unix oder MacOS.

Beispiel:

```
deactivate
```

Verwendung von virtualenv

Virtualenv muss installiert werden. Befehl: `pip install virtualenv`.

Aktivieren der virtuellen Umgebung mit virtualenv

1. Verwenden Sie den Befehl `virtualenv myenv`, um eine virtuelle Umgebung mit `virtualenv` zu erstellen.
2. Um die virtuelle Umgebung zu aktivieren:
   - Unter Windows verwenden Sie den Befehl `myenv\Scripts\activate`.

- Unter Unix oder MacOS verwenden Sie den Befehl `source myenv/bin/activate`.

Deaktivieren der virtuellen Umgebung mit virtualenv Um die virtuelle Umgebung, die mit `virtualenv` erstellt wurde, zu deaktivieren, kann ebenfalls der Befehl `deactivate` verwendet werden. Dieser Befehl funktioniert sowohl unter Windows als auch unter Unix oder MacOS.

Beispiel:

```
deactivate
```

Verwendung von

Pipenv muss installiert werden. Befehl: `pip install pipenv`.

Aktivieren der virtuellen Umgebung mit pipenv

1. Verwenden Sie den Befehl `pipenv install`, um eine virtuelle Umgebung mit `pipenv` zu erstellen.
2. Um die virtuelle Umgebung zu aktivieren, verwenden Sie den Befehl `pipenv shell`.

Deaktivieren der virtuellen Umgebung mit pipenv Um die virtuelle Umgebung, die mit `pipenv` erstellt wurde, zu deaktivieren, kann der Befehl `exit` verwendet werden.

Beispiel:

```
exit
```

Verwendung von conda

Aktivieren der virtuellen Umgebung mit conda

1. Verwenden Sie den Befehl `conda create --name myenv`, um eine virtuelle Umgebung mit `conda` zu erstellen.
2. Um die virtuelle Umgebung zu aktivieren, verwenden Sie den Befehl `conda activate myenv`.

Deaktivieren der virtuellen Umgebung mit conda Um die virtuelle Umgebung, die mit `conda` erstellt wurde, zu deaktivieren, kann der Befehl `conda deactivate` verwendet werden.

Beispiel:

```
conda deactivate
```

Anforderungen

- Python muss auf dem System installiert sein.
- Die entsprechenden Tools (`venv`, `virtualenv`, `pipenv`, `conda`) müssen installiert sein.

Beispiel

Ein Beispiel für die Verwendung von `venv`:

```
python3 -m venv myenv
```

Um die virtuelle Umgebung zu aktivieren:

- Unter Windows:

```
myenv\Scripts\activate
```

- Unter Unix oder MacOS:

```
source myenv/bin/activate
```

Pros und Cons von den verschiedenen virtuellen Umgebungen

conda

- Pro
  - Einfache Bereitstellung von Python-Umgebungen
  - Einfache Bereitstellung von Python-Umgebungen in verschiedenen Python-Versionen
  - Einfache Verwendung
  - Einfache Verwaltung von Bibliotheken
  - Einfache Verwaltung von Umgebungen
  - Einfache Verwaltung von Abhängigkeiten
- Cons
  - Die Daten werden in einem zentralen Ordner gespeichert, nicht im Projektordner
  - Die Daten werden in einem proprietären Format gespeichert

pipenv

- Pro
  - Einfache Verwendung
  - Einfache Verwaltung von Bibliotheken
  - Einfache Verwaltung von Umgebungen
  - Einfache Verwaltung von Abhängigkeiten
  - In der Pipfile werden die Bibliotheken und die Python-Version gespeichert
- Cons

  - Die Daten werden in einem zentralen Ordner gespeichert, nicht im Projektordner
  - Manchmal Probleme mit der Installation von Bibliotheken (Pipfile.lock)

venv

- Pro
  - Einfache Verwendung
  - Einfache Verwaltung von Bibliotheken
  - Einfache Verwaltung von Umgebungen
  - Einfache Verwaltung von Abhängigkeiten
  - Die Daten werden im Projektordner gespeichert
- Cons
  - Die Daten werden im Projektordner gespeichert. Muss bei Weitergabe bedacht werden.

virtualenv

- Pro
  - Einfache Verwendung
  - Einfache Verwaltung von Bibliotheken
  - Einfache Verwaltung von Umgebungen
  - Einfache Verwaltung von Abhängigkeiten
  - Die Daten werden im Projektordner gespeichert
- Cons
  - Die Daten werden im Projektordner gespeichert. Muss bei Weitergabe bedacht werden.

Vergleich venv und virtualenv

- venv ist in Python 3.3 eingeführt worden
- virtualenv ist für Python 2.7 entwickelt worden

Unterschiede zwischen venv und virtualenv die zu beachten sind bei der Verwendung

- virtualenv kann auch mit Python 3 verwendet werden
- virtualenv kann auch mit Python 2 verwendet werden
- venv kann nur mit Python 3 verwendet werden
- venv kann nicht mit Python 2 verwendet werden

Best Practices für die Verwendung von venv und virtualenv

- virtualenv sollte nur mit Python 2 verwendet werden
- venv sollte nur mit Python 3 verwendet werden