@saadjamilakhtar

PYTHON VIRTUAL ENVIRONMENT AND DEPENDENCY MANAGEMENT

SWIPE

01

</>

WHY VIRTUAL ENVIRONMENT ?

Virtual environments are isolated spaces where you can develop projects with their own set of dependencies. This isolation prevents conflicts between packages and ensures that each project has access only to the packages it needs

```
# Create a virtual environment named
'myenv'
python -m venv myenv
```

@saadjamilakhtar

02

Activating and Deactivating Virtual Environments

Activating a virtual environment changes your shell's PATH to point to the environment's Python executable and packages meaning any package installations will work within the virtual environment.
Deactivating the virtual environment returns you to the global environment.

```
# Activate on Windows
myenv\Scripts\activate

# Activate on macOS and Linux
source myenv/bin/activate

# Deactivate
deactivate
```

03

</>

Managing Dependencies with Pip

Pip is Python's package manager, and it's essential for installing, upgrading, and managing dependencies within your virtual environment. It ensures that you can easily add and remove packages without affecting other projects.

```
# Install a package
pip install package-name

# Install from a requirements.txt file
pip install -r requirements.txt

# Upgrade a package
pip install --upgrade package-name
```

04

</>

Keeping Track of Dependencies: requirements.txt

A requirements.txt file lists all the packages your project depends on, along with their versions. This file is shareable and helps recreate the exact development environment.

```
# requirements.txt
package-name==1.2.3
another-package>=2.0.0
```

@saadjamilakhtar

05

</>

Creating Reproducible Environments: Pipenv

Pipenv simplifies dependency management by automatically creating and managing a virtual environment for your project. It also generates a Pipfile and a Pipfile.lock file for accurate version management.

```
# Install a package
pipenv install package-name

# Activate the Pipenv shell
pipenv shell
```

@saadjamilakhtar

06

</>

Streamlined Virtual Environment Setup: Poetry

Poetry is a comprehensive tool that combines dependency management and virtual environments. It aims to simplify and enhance the Python packaging experience, making it easy to declare, manage, and install dependencies.

```
# Install a package
poetry add package-name
```

@saadjamilakhtar

07

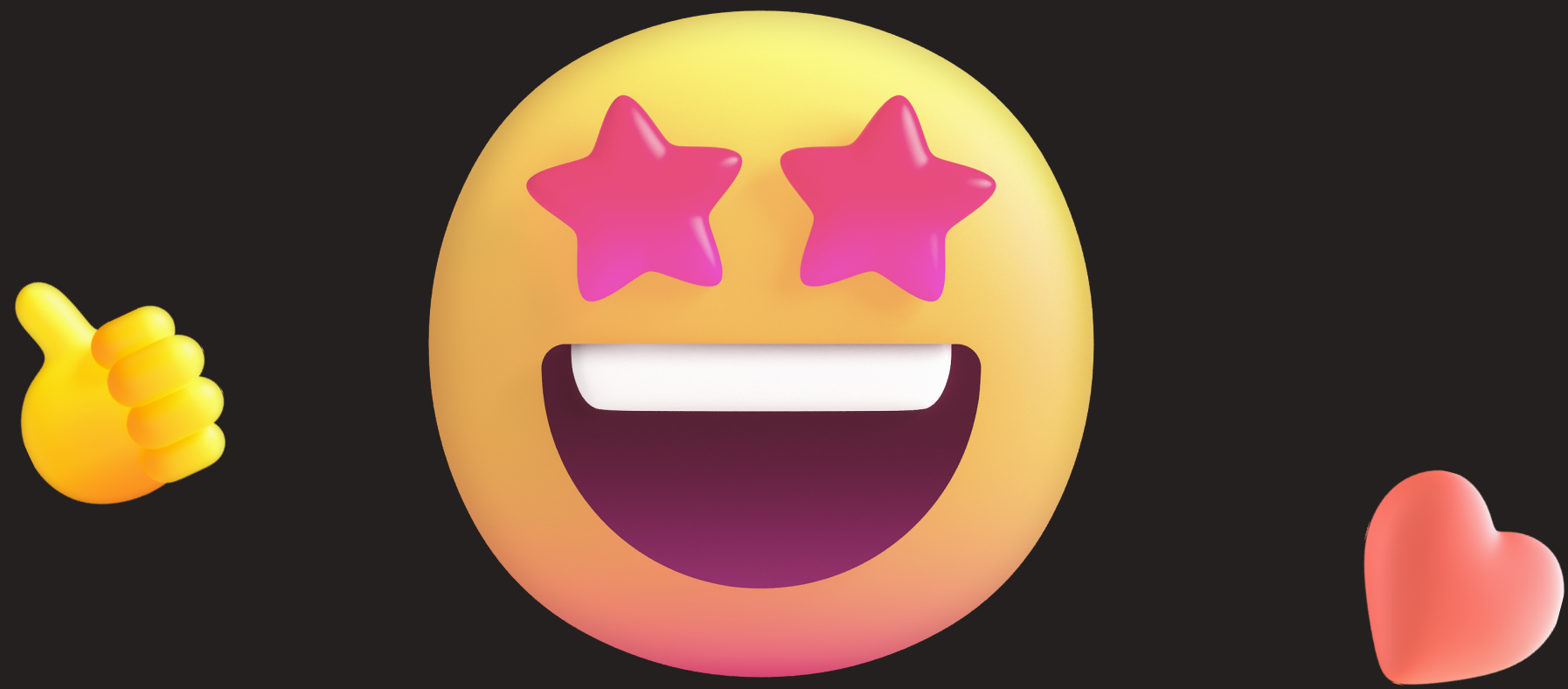
</>

Sharing Your Environment: Version Control

Version control systems like Git help you track and share your project's dependencies. By keeping your virtual environment separate and sharing the requirements.txt file, you ensure a consistent environment across collaborators.

```
# Ignore virtual environment folders
echo "myenv/" >> .gitignore
```

@saadjamilakhtar



Was this post helpful ?
Follow for more

@saadjamilakhtar