Python Virtual Environments

Summary: in this tutorial, you'll learn about the Python virtual environments.

Why do you need Python virtual environments

Python stores all system packages in a folder that you specify when installing Python (https://www.pythontutorial.net/getting-started/install-python/) .

Typically, most system packages locate at subfolders of a path specified in the `sys.prefix` .

To find this path, you can import the `sys` module and display it as follows:

```
>>> import sys
>>> sys.prefix
```

It'll show something like this:

```
C:\\Python38
```

When you use pip (https://www.pythontutorial.net/python-basics/python-pip/) to install third-party packages, Python stores these packages in a different folder specified by the `site.getsitepackges()` function:

```
>>> import site
>>> site.getsitepackages()
```

It returns something like:

```
['C:\\Python38',
'C:\\Python38\\lib\\site-packages']
```

If you have several projects that use only standard library, you'll be fine.

However, it'll be a problem when you have some projects that use third-party packages.

Suppose you have two projects that use different versions of a library.

Since there's only one location to store the third-party packages, you cannot store different versions at the same time.

Of course, you can use pip to switch between versions by installing/uninstalling a package. But it will be time-consuming and won't scale.

This is where virtual environments come into play.

What is a virtual environment

Python uses virtual environments to create an isolated environment for every project.

In other words, each project will have its own directory to store the third-party packages.

In case you have multiple projects that use different versions of a package, you can store them in separate folders (or virtual environments).

Python 3 includes the virtual environment module (`venv`) as a standard library. To create a virtual environment for a project, you use the `pipenv (https://www.pythontutorial.net/python-basics/python-pipenv/)` tool.

In the next tutorial, you'll learn how to:

- Install pipenv to manage virtual environments (https://www.pythontutorial.net/python-basics/install-pipenv-windows/) .
- Create a development workflow using virtual environments (https://www.pythontutorial.net/python-basics/python-pipenv/) .

Summary

- A Python virtual environment creates an isolated environment for a Python project.
- Use pipenv tool to manage virtual environments.