Python Virtual Environment | Introduction

Last Updated : 11 Sep, 2024

A Python Virtual Environment is an isolated space where you can work on your Python projects, separately from your system-installed Python. You can set up your own libraries and dependencies without affecting the system Python. We will use **virtualenv** to create a virtual environment in Python.

What is a Virtual Environment?

A virtual environment is a tool that helps to keep dependencies required by different projects separate by creating isolated Python virtual environments for them. This is one of the most important tools that most Python developers use.

Why do we need a virtual environment?

Imagine a scenario where you are working on two web-based Python projects one of them uses Django 4.0 and the other uses Django 4.1 (check for the latest Django versions and so on). In such situations, we need to create a virtual environment in Python that can be really useful to maintain the dependencies of both projects.

When and where to use a virtual environment?

By default, every project on your system will use these same directories to store and retrieve site packages (third-party libraries).

How does this matter? Now, in the above example of two projects, you have two versions of Django. This is a real problem for Python since it can't differentiate between versions in the "site-packages" directory. So both v1.9 and v1.10 would reside in the same directory with the same name.

This is where virtual environments come into play. To solve this problem, we just need to create two separate virtual environments for both projects.

The great thing about this is that there are no limits to the number of environments you can have since they're just directories containing a few scripts.

A virtual Environment should be used whenever you work on any Python-based project. It is generally good to have one new virtual environment for every Python-based project you work on. So the dependencies of every project are isolated from the system and each other.

Create Virtual Environment in Python

We use a module named virtualenv which is a tool to create virtual environment Python, isolated from the system environment Python.

virtualenv creates a folder that contains all the executables necessary to use the packages that a Python project would need.

Installing virtualenv

```
$ pip install virtualenv
```

Test your installation:

```
$ virtualenv --version
```

Create Python Environment

You can create a virtualenv using the following command:

```
$ virtualenv my_env
```

After running this command, a directory named my_env will be created. This is the directory that contains all the necessary executables to use the packages that a Python project would need.

This is where Python packages will be installed. If you want to specify the Python interpreter of your choice, for example, Python3, it can be done using the following command:

```
$ virtualenv -p /usr/bin/python3 virtualenv_name
```

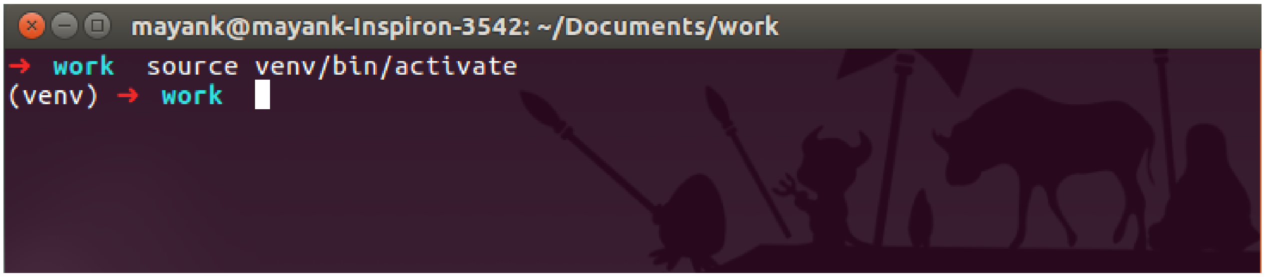
Now after creating a virtual environment, you need to activate it. Remember to activate the relevant virtual environment every time you work on the project. This can be done using the following command:

> ***Activate a Virtual Environment on Windows***
>
> *To activate virtual environment using windows command prompt change directory to your virtual env, Then use the below command*
>
> ```
> $ cd <envname>
> $ Scripts\activate
> ```
>
> ***Note:*** *source is a shell command designed for users running on Linux (or any Posix, but whatever, not Windows).*

Activate a virtual environment on Linux

```
$ source virtualenv_name/bin/activate
```

Once the virtual environment is activated, the name of your virtual environment will appear on the left side of the terminal.



mayank@mayank-Inspiron-3542: ~/Documents/work

```
➜ work  source venv/bin/activate
(venv) ➜ work
```

This will let you know that the virtual environment is currently active.

Installing Dependencies in Virtual Environment Python

In the image below, venv named virtual environment is active. Now you can install dependencies related to the project in this virtual environment.

For example, if you are using Django 1.9 for a project, you can install it like you install other packages.

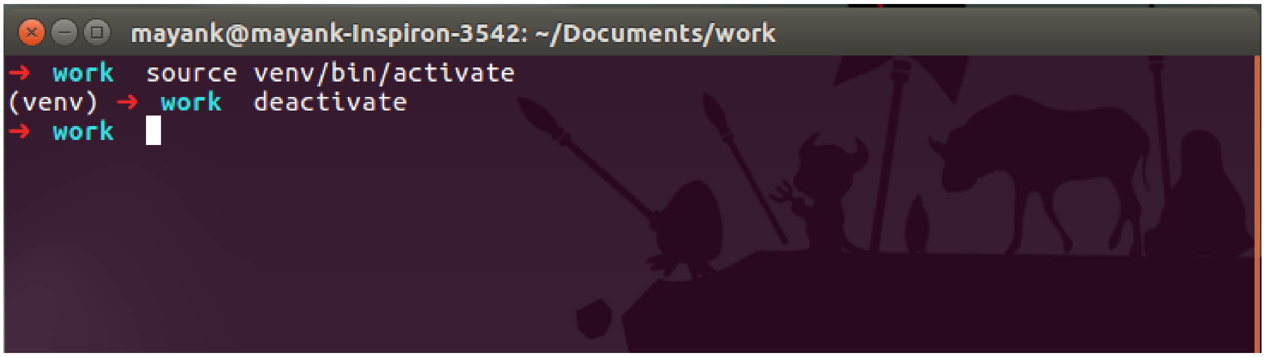
```
(virtualenv_name)$ pip install Django==1.9
```

The Django 1.9 package will be placed in virtualenv_name folder and will be isolated from the complete system.

Deactivate Python Virtual Environment

Once you are done with the work, you can deactivate the virtual environment by the following command:

```
(virtualenv_name)$ deactivate
```



```
➜ work  source venv/bin/activate
(venv) ➜ work  deactivate
➜ work  
```

Now you will be back to the system's default Python installation.

We have covered virtual environment in Python, How to create virtual environment in Python?, how to activate and deactivate virtual environment? and now to install dependencies.

This covers all the basic concepts of Python virtual environment and you can use it on your personal PC.

Also Read:

- Creating Python Virtual Environment in Windows and Linux
- Create virtual environment using venv | Python
- How to leave/exit/deactivate a Python virtualenv
- Set up virtual environment for Python using Anaconda

Looking to dive into the world of programming or sharpen your Python skills? Our **Master Python: Complete Beginner to Advanced Course** is your ultimate guide to becoming proficient in Python. This course covers everything you need to build a solid foundation from fundamental programming concepts to

M Mayank Agrawal 86

Previous Article **Next Article**

Jython - Introduction and Installation

Similar Reads

Creating Python Virtual Environment in Windows and Linux

A Virtual Environment is a Python environment, that is an isolated working copy of Python that allows you to work on a specific project without affecting other...

2 min read

Using mkvirtualenv to create new Virtual Environment - Python

Virtual Environment are used If you already have a python version installed and you want to use a different version for a project without bothering the older one...

2 min read

Create virtual environment using venv | Python

Managing multiple Python projects that have conflicting dependencies can be a daunting task. However, virtual environments can help you solve this problem....

6 min read

How to Set Up a Python Virtual Environment in Visual Studio on Windows

Creating a Python virtual environment is a fundamental practice for managing dependencies and ensuring project isolation. This guide will walk us through th...

3 min read

Using Jupyter Notebook in Virtual Environment

In this article, we are going to see how to set Virtual Environment in Jupyter. Sometimes we want to use the Jupyter notebook in a virtual environment so tha...

2 min read

Setting Up a Virtual Environment in Django

Setting up a virtual environment in Django is essential for isolating your project's dependencies and ensuring consistent behavior across different environments. ...

2 min read

Add packages to Anaconda environment in Python

Let's see some methods that can be used to install packages in the Anaconda environment. There are many ways one can add pre-built packages to an...

2 min read

Python - Setting up the Bokeh Environment

Bokeh is supported on CPython versions 3.6+ only both with Standard distribution and Anaconda distribution. Other Python versions or implementatio...

1 min read

PYTHONPATH Environment Variable in Python

Python's behavior is greatly influenced by its environment variables. One of those variables is PYTHONPATH. It is used to set the path for the user-defined modul...

2 min read

Environment Variables in Python

In Python, its behavior is highly influenced by the setup of the environment variables. There is a fixed number of environment variables that Python...

4 min read

Article Tags : Python

Practice Tags : python

Corporate & Communications Address:-
A-143, 9th Floor, Sovereign Corporate
Tower, Sector- 136, Noida, Uttar Pradesh
(201305) | Registered Address:- K 061,
Tower K, Gulshan Vivante Apartment,
Sector 137, Noida, Gautam Buddh
Nagar, Uttar Pradesh, 201305

Company

About Us
Legal
In Media
Contact Us
Advertise with us
GFG Corporate Solution
Placement Training Program
GeeksforGeeks Community

DSA

Data Structures
Algorithms
DSA for Beginners
Basic DSA Problems
DSA Roadmap
Top 100 DSA Interview Problems
DSA Roadmap by Sandeep Jain
All Cheat Sheets

Web Technologies

HTML
CSS
JavaScript
TypeScript
ReactJS
NextJS
Bootstrap
Web Design

Computer Science

Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths
Software Development
Software Testing

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns

Languages

Python
Java
C++
PHP
GoLang
SQL
R Language
Android Tutorial
Tutorials Archive

Data Science & ML

Data Science With Python
Data Science For Beginner
Machine Learning
ML Maths
Data Visualisation
Pandas
NumPy
NLP
Deep Learning

Python Tutorial

Python Programming Examples
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question
Django

DevOps

Git
Linux
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

Inteview Preparation

Competitive Programming
Top DS or Algo for CP
Company-Wise Recruitment Process
Company-Wise Preparation
Aptitude Preparation

OOAD

System Design Bootcamp

Interview Questions

School Subjects

Mathematics

Physics

Chemistry

Biology

Social Science

English Grammar

Commerce

World GK

Puzzles

GeeksforGeeks Videos

DSA

Python

Java

C++

Web Development

Data Science

CS Subjects