FULL TUTORIAL

PyInstaller + venv

Linux & Windows

BUILD .EXE & ELF — NO SYSTEM CLUTTER

LINUX WINDOWS PYTHON VENV

MWU

MyWebUniversity

@MyWebUniversity

دری

MyWebUniversity

درس‌های دری

@MyWebUniversity-Dari

PYTHON TUTORIAL

PyInstaller

with venv

Build Standalone Executables for Linux & Windows
Using Virtual Environments — Clean, Isolated, Manageable

 Linux

 Windows

 Python

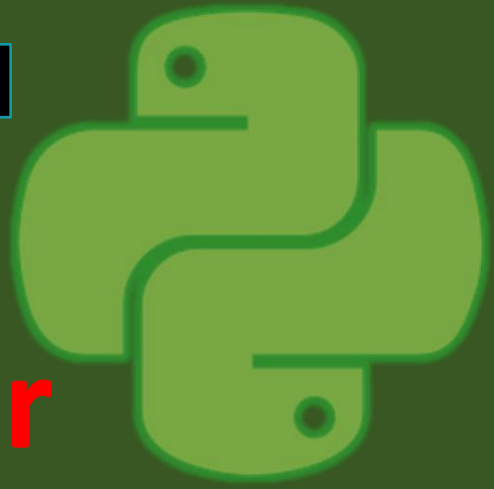
PYTHON TUTORIAL

Create Python Executables on Windows using PyInstaller with venv

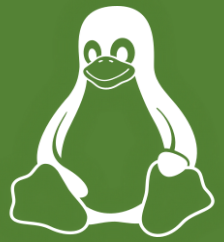
Build Standalone Executables for Windows
Using Virtual Environments — Clean, Isolated, Manageable

LINUX Python Executables With Venv & PyInstaller

PYTHON TUTORIAL



Build Standalone Executables for Windows Using Virtual Environments — Clean, Isolated, Manageable



Linux

Python

PYTHON TUTORIAL

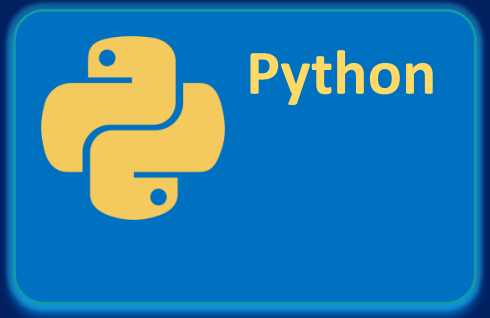
LINUX Python Executables With Venv & PyInstaller

Linux

Python

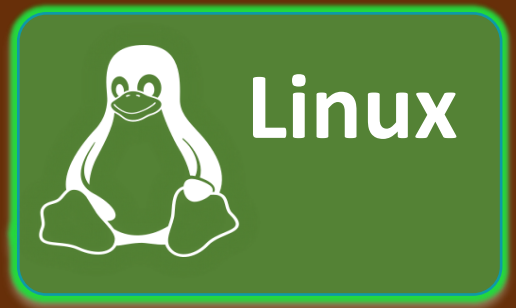
PYTHON TUTORIAL

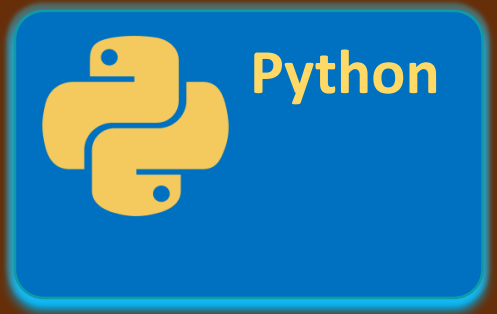
Windows Python Executables With Venv & PyInstaller



Lecture Only

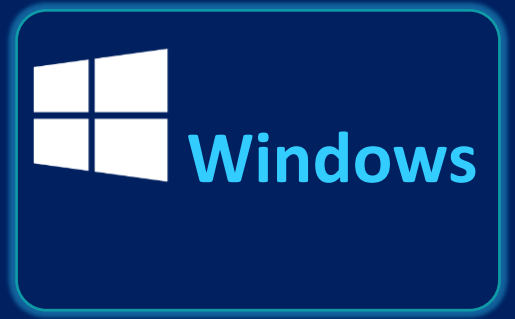
Windows & LINUX Python Executables With Venv & PyInstaller

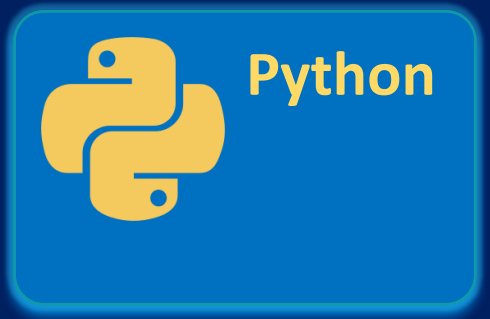


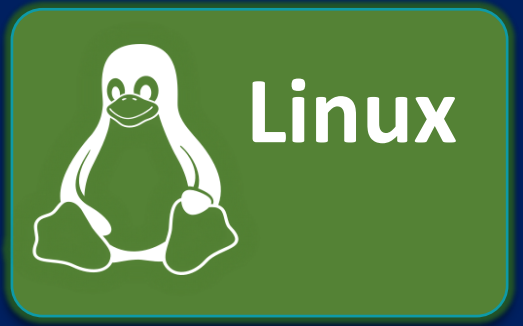


PYTHON TUTORIAL

Windows & LINUX Python Executables With Venv & PyInstaller







What We'll Cover Today

01

Why venv?

Keep your system clean, isolate dependencies per project

02

Setting Up the venv

Create and activate a virtual environment

03

Installing PyInstaller

Install only inside venv — no system pollution

04

Build on Linux

Generate an ELF executable for Linux systems

05

Build on Windows

Generate a .exe executable for Windows

06

Options & Tips

One-file, icon, hidden imports, troubleshooting

Why Use a Virtual Environment?

No System Clutter

Packages install only inside the venv folder — your global Python stays untouched.

Per-Project Control

Each project gets its own exact dependency versions. No more version conflicts.

Reproducibility

Share requirements.txt and anyone can recreate the exact same environment.

Safe to Delete

Just delete the venv folder to completely remove all project packages.

Prerequisites

Python 3.6+

```
python3 --version
```

Comes with venv module built-in

Available from python.org or package manager

Linux

No extra steps needed

```
python3-venv may need install:
```

```
sudo apt install python3-venv
```

Windows

Download Python installer from python.org

Check ✅ Add Python to PATH

venv is included automatically

Step 1 — Create & Activate the venv

Linux / macOS

```
# Install python3.12-venv first
sudo apt install python3.12-venv

# Navigate to you project folder
cd ~/my_project

# Create the virtual environment

python3 -m venv venv


# Activate it

source venv/bin/activate


# Your prompt now shows (venv)

(venv) user@machine:~/my_project$
```

Windows (CMD / PowerShell)

```
# Navigate to your project folder

cd C:\Users\You\my_project


# Create the virtual environment

python -m venv venv


# Activate it (CMD)

venv\Scripts\activate.bat


# Activate it (PowerShell)

.\venv\Scripts\Activate.ps1
```

Step 2 — Install PyInstaller Inside the venv

```
(venv) $ pip install pyinstaller
```

Always activate first

Make sure your prompt shows (venv) before running pip install. This ensures PyInstaller goes into the venv, not your system Python.

Verify installation

Run: pyinstaller --version
You should see something like: 6.x.x

Install your app deps

Also install your app's own dependencies here:
pip install requests numpy …

Step 3 — Build the Executable on Linux

One-file executable

→ Produces a single binary in dist/

```
pyinstaller --onefile my_app.py
```

With console window

→ Default — shows terminal output

```
pyinstaller --onefile --console my_app.py
```

No console (GUI apps)

→ Silent executable, no terminal pop-up

```
pyinstaller --onefile --noconsole my_app.py
```

Custom output name

→ Renames the binary to 'MyApp'

```
pyinstaller --onefile --name MyApp my_app.py
```

```
Output: ./dist/my_app   (ELF binary — run with: ./dist/my_app)
```

Step 4 — Build the Executable on Windows

Basic one-file .exe

→ Creates my_app.exe in dist\

```
pyinstaller --onefile my_app.py
```

Windowed app (no CMD popup)

→ Ideal for GUI apps like tkinter

```
pyinstaller --onefile --windowed my_app.py
```

With a custom icon

→ Attach a .ico file to the .exe

```
pyinstaller --onefile --icon=app.ico my_app.py
```

All together

→ Full real-world example

```
pyinstaller --onefile --windowed --icon=app.ico --name MyApp my_app.py
```

```
Output:  dist\my_app.exe   (Double-click to run — no Python installation needed!)
```

Recommended Project Structure

```
my_project/
├── venv/
│   └── (isolated packages)
├── my_app.py
├── app.ico
├── requirements.txt
├── dist/
│   └── my_app  (or .exe)
├── build/
│   └── (temp files)
└── my_app.spec
```

Never commit venv/

Add it to .gitignore. It's huge and user-specific.

Commit requirements.txt

Run: pip freeze > requirements.txt

The .spec file

PyInstaller generates this. Edit for advanced configs.

dist/ is your output

Only share the binary in dist/ with users.

Useful Options & Troubleshooting

--hidden-import

PyInstaller misses dynamic imports.
pyinstaller --hidden-import=pkg my_app.py

Test before building

Run your app from the activated venv first to catch any import errors early.

--add-data

Bundle extra files (images, configs):
--add-data 'src:dst' (Linux)
--add-data 'src;dst' (Windows)

Antivirus false positives

PyInstaller .exe files sometimes trigger AV. Sign your executable or advise users to whitelist it.

--clean

Force a fresh rebuild, removes cached .spec artifacts:
pyinstaller --clean --onefile my_app.py

Deactivate the venv

When done working:
deactivate
Your prompt returns to normal.

Quick Reference Cheat Sheet

Linux	Windows
Create venv `python3 -m venv venv`	Create venv `python -m venv venv`
Activate `source venv/bin/activate`	Activate CMD `venv\Scripts\activate.bat`
Install `pip install pyinstaller`	Activate PS `.\venv\Scripts\Activate.ps1`
Build `pyinstaller --onefile my_app.py`	Install `pip install pyinstaller`
Deactivate `deactivate`	Build `pyinstaller --onefile my_app.py`

You're Ready to Build!

Create a venv → Install PyInstaller inside it → Build & distribute your app

- venv keeps it clean
- --onefile bundles everything
- Build on target OS
- dist/ is for users

👍 Like & Subscribe for more Python tutorials!