Complete Python Virtual Environment Guide: Setup & Management

It is best to run Python programmes in a virtual environment to avoid package incompatibilities and maintain project isolation.

1. Setting up your virtual environment (one-time setup)

Windows

```
# Open Command Prompt (cmd) or PowerShell
# Navigate to the root directory of your project
cd path\to\your\project

# Create virtual environment
python -m venv .venv

# Activate the virtual environment
.venv\Scripts\activate

# Verify activation (you should see (.venv) in your prompt)
# Upgrade pip (recommended)
python -m pip install --upgrade pip

# Install project dependencies
python -m pip install -r requirements.txt

# Deactivate when finished
deactivate
```

macOS/Linux

```
# Open Terminal
# Navigate to the root directory of your project
cd /path/to/your/project

# Create virtual environment
python3 -m venv .venv

# Activate the virtual environment
source .venv/bin/activate

# Verify activation (you should see (.venv) in your prompt)
# Upgrade pip (recommended)
python -m pip install --upgrade pip
```

```
# Install project dependencies
python -m pip install -r requirements.txt

# Deactivate when finished
deactivate
```

2. Daily usage of your virtual environment

Windows

```
# Navigate to your project directory
cd path\to\your\project

# Activate virtual environment
.venv\Scripts\activate

# Verify activation - your prompt should show (.venv)
# Run your application (check README.md for specific instructions)
python your-app-name.py

# When finished working, deactivate
deactivate
```

macOS/Linux

```
# Navigate to your project directory
cd /path/to/your/project

# Activate virtual environment
source .venv/bin/activate

# Verify activation - your prompt should show (.venv)
# Run your application (check README.md for specific instructions)
python your-app-name.py

# When finished working, deactivate
deactivate
```

3. Git integration

Essential: Add the virtual environment directory to your `.gitignore` file:

```
# Virtual environment
.venv/
venv/
env/
```

```
# Python cache files
__pycache__/
*.pyc
*.pyo
*.pyd
.Python
```

4. Troubleshooting

Common issues:

- **"python/python3 not found"**: Ensure Python is installed and added to your system PATH
- **Permission denied (macOS/Linux)**: You may need to install Python via homebrew or your package manager
- **Virtual environment not activating**: Check you're in the correct directory and using the right activation script
- **Packages not installing**: Ensure your virtual environment is activated before running pip commands

Verification commands:

```
# Check which Python you're using (should show .venv path when activated)
where python     # Windows
which python     # macOS/Linux

# Check installed packages
pip list

# Check pip version
pip --version
```

5. Best practices

- Always activate your virtual environment before working on your project
- Keep `requirements.txt` updated: `pip freeze > requirements.txt`
- Use descriptive names for virtual environments if not using `.venv`
- Never commit virtual environment directories to version control
- Create a new virtual environment for each project to maintain isolation