Step-by-Step Guide to Creating a Virtual Environment with UV for Windows 11 and WSL2 Ubuntu 24.02

Windows 11

1. **Install UV**:
   - Open PowerShell and run the following command to install UV:

     ```
     powershell -c "irm https://astral.sh/uv/install.ps1 | iex"
     ```

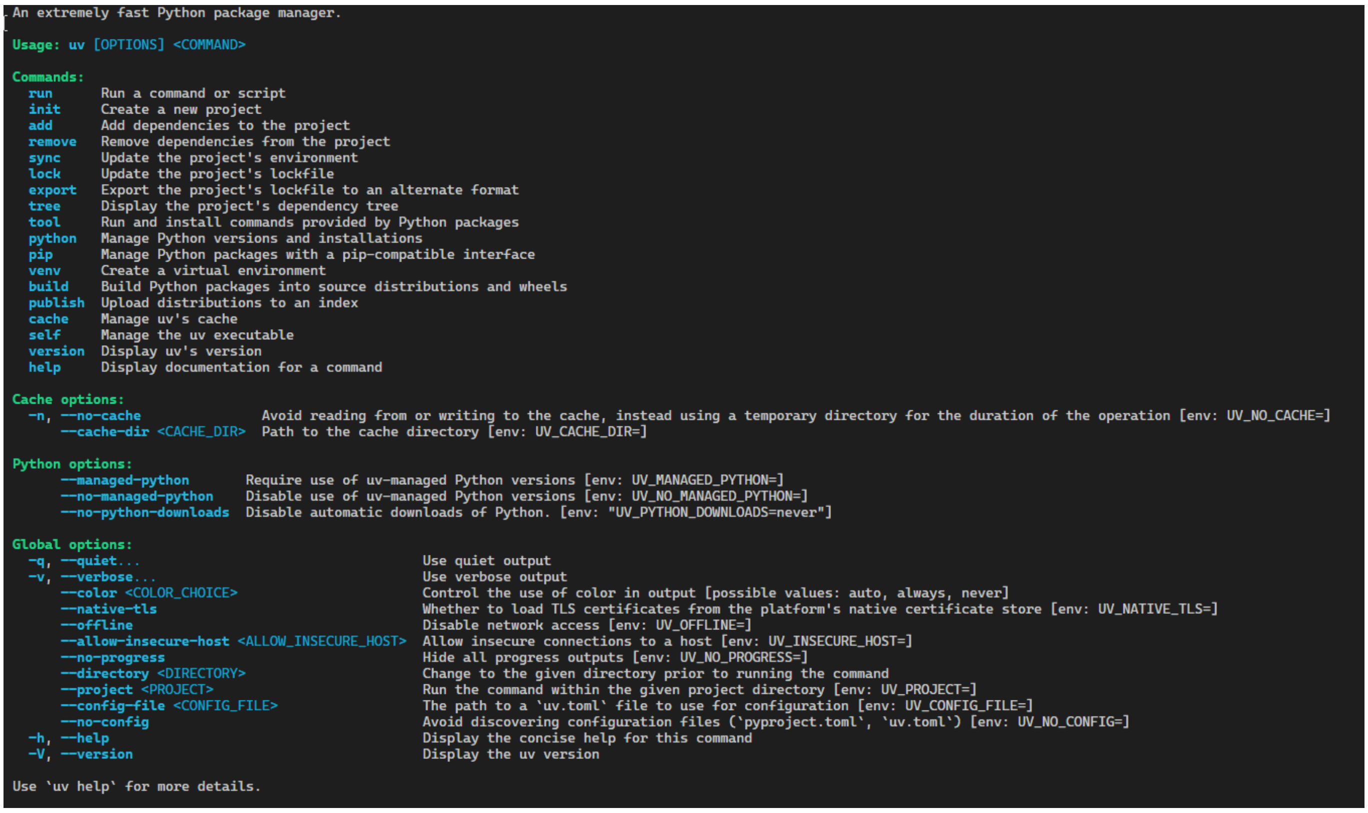
   - Alternatively, you can install UV using pip:

     ```
     pip install uv
     ```

2. **Verify Installation**:
   - Check if UV is installed correctly by running:

     ```
     uv --version
     ```

   - uv help which is constantly evolving below (June 7, 2025) see helpful links at the end:



```
An extremely fast Python package manager.

Usage: uv [OPTIONS] <COMMAND>

Commands:
  run      Run a command or script
  init     Create a new project
  add      Add dependencies to the project
  remove   Remove dependencies from the project
  sync     Update the project's environment
  lock     Update the project's lockfile
  export   Export the project's lockfile to an alternate format
  tree     Display the project's dependency tree
  tool     Run and install commands provided by Python packages
  python   Manage Python versions and installations
  pip      Manage Python packages with a pip-compatible interface
  venv     Create a virtual environment
  build    Build Python packages into source distributions and wheels
  publish  Upload distributions to an index
  cache    Manage uv's cache
  self     Manage the uv executable
  version  Display uv's version
  help     Display documentation for a command

Cache options:
  -n, --no-cache               Avoid reading from or writing to the cache, instead using a temporary directory for the duration of the operation [env: UV_NO_CACHE=]
      --cache-dir <CACHE_DIR>  Path to the cache directory [env: UV_CACHE_DIR=]

Python options:
      --managed-python       Require use of uv-managed Python versions [env: UV_MANAGED_PYTHON=]
      --no-managed-python    Disable use of uv-managed Python versions [env: UV_NO_MANAGED_PYTHON=]
      --no-python-downloads  Disable automatic downloads of Python. [env: "UV_PYTHON_DOWNLOADS=never"]

Global options:
  -q, --quiet...                                   Use quiet output
  -v, --verbose...                                 Use verbose output
      --color <COLOR_CHOICE>                       Control the use of color in output [possible values: auto, always, never]
      --native-tls                                 Whether to load TLS certificates from the platform's native certificate store [env: UV_NATIVE_TLS=]
      --offline                                    Disable network access [env: UV_OFFLINE=]
      --allow-insecure-host <ALLOW_INSECURE_HOST>  Allow insecure connections to a host [env: UV_INSECURE_HOST=]
      --no-progress                                Hide all progress outputs [env: UV_NO_PROGRESS=]
      --directory <DIRECTORY>                      Change to the given directory prior to running the command
      --project <PROJECT>                          Run the command within the given project directory [env: UV_PROJECT=]
      --config-file <CONFIG_FILE>                  The path to a `uv.toml` file to use for configuration [env: UV_CONFIG_FILE=]
      --no-config                                  Avoid discovering configuration files (`pyproject.toml`, `uv.toml`) [env: UV_NO_CONFIG=]
  -h, --help                                       Display the concise help for this command
  -V, --version                                    Display the uv version

Use `uv help` for more details.
```

1. **Initialize the Project**:

- Navigate to your project directory and initialize UV:

```
uv init
```

  - This command creates the necessary configuration files, including `pyproject.toml`.

2.**Create a Virtual Environment**:

- Sample image of the project location

```
![alt text](image.png)
```

- Create a virtual environment:

```
uv venv
```

- To specify a Python version, use:

```
uv venv --python 3.13
```

1. **Activate the Virtual Environment**:

- Activate the virtual environment using:

```
# windows activate
.venv\Scripts\activate

# linux activate
.venv/bin/activate
```

1. **Install Dependencies**:

  - Once the environment is active, install packages using UV `add` command:

```
# uv add <any package_name>, additional packages are space seperated
uv add httpx
```

2. **Sync Packages**:

- To ensure your virtual environment matches the dependencies specified in your `pyproject.toml` and `uv.lock` files, run:

  ```
  uv sync
  ```

3. **Upgrade Packages**:

   - To upgrade a specific package to the latest version:

     ```
     uv lock --upgrade -package_name
     ```

   - To upgrade all packages to their latest versions:

     ```
     uv lock --upgrade
     ```

4. **Force Reinstall Packages**:

   - Explanation of the flags:

     1. install: This is the uv command to install a package.
     2. --upgrade or -U: This flag upgrades the package to the latest version.
     3. --reinstall: This flag forces the reinstallation of the package.

   - To force reinstall a specific package:

     ```
     uv install <package_name> --upgrade --reinstall
     ```

   1. **Remove Packages**:

   - To remove a package from your environment:

     ```
     uv remove package_name
     ```

   2.**Freeze Dependencies**:

   - Generate a list of installed dependencies:

     ```
       uv export --format requirements-txt > requirements.txt
     ```

5. **Deactivate and Remove Environments**:

- Deactivate the virtual environment:

```
# windows deactivate
.venv\Scripts\deactivate

# linux deactivate
.venv/bin/deactivate
```

- Remove the virtual environment:

```
  # Remove the current virtual environment
  uv venv --rm

  # Or if you want to remove a specific virtual environment
  uv venv --rm <env_name>
```

6. **Build a Wheel**:

   - To create a wheel distribution for your project:

   ```
   uv build
   ```

   - If you want to build only wheels (not source distributions), you can use:

   ```
   uv build --wheel
   ```

   - Or if you want only source distributions:

   ```
   uv build --sdist
   ```

WSL2 Ubuntu 24.02

1. **Install UV**:

   - Open your WSL2 terminal and run the following command to install UV:

   ```
   curl -LsSf https://astral.sh/uv/install.sh | sh
   ```

2. **Verify Installation**:

   - Check if UV is installed correctly by running:

```
uv --version
```

1.**Initialize the Project**:

- Create and navigate to your project directory:

  ```
  mkdir my-project
  cd my-project
  ```

- Initialize the project with uv:

  ```
  uv init
  ```

- Alternative: Initialize with a specific Python version:

```
uv init --python 3.13
```

- Initialize as a library/package (if you're building a distributable package):

```
uv init --lib
```

1. This will create:

- `pyproject.toml` - Project configuration and dependencies
- `README.md` - Basic project documentation
- `src/ directory` (if using --lib) or main Python files
- `.python-version` file (if specific Python version specified)

1. After initialization, you can:

- `Add dependencies`: uv add package-name
- `Run scripts`: uv run script.py
- `Install the project`: uv sync (installs project in development mode)
- `Create virtual environment`: uv venv (though uv run handles this automatically)

1. The uv init command sets up a modern Python project structure following current best practices.

- `Key fix`: **Replaced uv pip install -e . with uv sync, which is the proper uv command for installing a project in development mode. The uv sync command automatically handles the editable installation without needing to use pip directly.RetryClaude can make mistakes. Please double-check responses.**

1. **Create a Virtual Environment**:
   - Create a virtual environment:
     ```
     uv venv
     ```
   - To specify a Python version, use:
     ```
     uv venv --python 3.13
     ```
2. **Activate the Virtual Environment**:
   - Activate the virtual environment using:

   ```
   .venv/bin/activate
   ```
3. **Install Dependencies**:
   - Once the environment is active, add packages using UV's add command:
     ```
      # uv pip install requests use uv command line syntax below :
      uv add httpx
     ```
   - To add dependencies to your `pyproject.toml` file:
     ```
     uv add requests
     ```
4. **Sync Packages**:
   - To ensure your virtual environment matches the dependencies specified in your `pyproject.toml` and `uv.lock` files, run:
     ```
     uv sync
     ```
5. **Upgrade Packages**:
   - To upgrade a specific package to the latest version:
     ```
     uv add --upgrade package_name
     ```

- To upgrade all packages to their latest versions:

  ```
  uv lock --upgrade
  uv sync
  ```

6. **Force Reinstall Packages**:

   - To force reinstall a specific package:

     ```
     uv export --format requirements-txt > requirements.txt
     ```

7. **Remove Packages**:

   - To remove a package from your environment:

     ```
     uv remove package_name
     ```

8. **Freeze Dependencies**:

   - Generate a list of installed dependencies:

     ```
     uv export --format requirements-txt > requirements.txt
     ```

9. **Deactivate and Remove Environments**:

   - Deactivate the virtual environment:dir

   ```
   .venv/bin/deactivate
   ```

   - Remove the virtual environment:

     ```
     uv remove
     ```

10. Build a Wheel

- To create a wheel distribution for your project:

```
uv build
```

- If you want to build only wheels (not source distributions), you can use:

```
uv build --wheel
```

- Or if you want only source distributions:

```
uv build --sdist
```

- **By following these steps, you can efficiently manage Python virtual environments using UV on both Windows 11 and WSL2 Ubuntu 24.02.**

Documentation

1. UV documentation
2. Youtube UV for Python... (Almost) All Batteries Included
3. Youtube How Much FASTER Is Python 3.13 Without the GIL?