Installing `uv` on a Windows machine without administrative rights

The following instructions are intended as a *workaround* for those of you who do not have administrative rights on their machine. If you do, in fact, have administrative rights, it is recommended that you follow the instructions here!

Main workaround: install `uv` in a subfolder of the user directory

Since you usually have sufficient rights inside your user directory, we install the `uv` tool manually in that directory. `uv` normally creates a `cache` folder in a system directory. We reroute that `cache` to a subfolder in the user directory as well.

1. Open a PowerShell terminal, then run:

```
# Define locations
# this is where we download the uv binary
$uvUrl = "https://github.com/astral-sh/uv/releases/latest/download/uv-x86_64-pc-windows-
# this is where we temporarily store the zip file from the web
$uvZip = "$env:TEMP\uv.zip"
# this is where we install the `uv` tool (in the `uv` subfolder in your user directory)
$uvDir = "$env:USERPROFILE\uv"
# the cahce will be a subfolder of that directory (therefore also in your user directory
$uvCache = "$uvDir\.cache"

# Create install and cache directories
New-Item -ItemType Directory -Force -Path $uvDir | Out-Null
New-Item -ItemType Directory -Force -Path $uvCache | Out-Null

# Download and extract uv
Invoke-WebRequest -Uri $uvUrl -OutFile $uvZip -UseBasicParsing
Expand-Archive -Path $uvZip -DestinationPath $uvDir -Force

# Add uv to PATH for this session
$env:PATH = "$uvDir;$env:PATH"

# Set UV cache directory for this session
$env:UV_CACHE_DIR = $uvCache
$env:UV_CONFIG_DIR = $uvCache

# Persist PATH and cache location for future sessions (user-level, no admin rights neede
[Environment]::SetEnvironmentVariable("PATH", "$uvDir;$([Environment]::GetEnvironmentVar
[Environment]::SetEnvironmentVariable("UV_CACHE_DIR", $uvCache, "User")
[Environment]::SetEnvironmentVariable("UV_CONFIG_DIR", $uvCache, "User")

# Verify installation -> should produce a version like "uv 0.8.14" or similar
uv --version

# Verify cache directory -> should point to C:\Users\<you>\uv\.cache
uv cache dir
```

2. Run the following to test everything

> **Note:** If you are using conda, you should first deactivate any active conda environment (use `conda deactivate`).

The following series of terminal commands temporarily creates a project directory, creates a virtual environment, and installs a package, then removes everything again and goes back to the original directory you started in.

In the PowerShell, run:

```
# store the current working directory in a variable:
$env:OLDPWD = Get-Location
# create a (temporary) project directory in the user home
mkdir $HOME/test___uv -f
# change into the project directory
cd $HOME/test___uv
# initialize the project
uv init --python=3.13
# create a virtual environment
uv venv
# enable script execution for the current terminal process (required for venv activation
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
# activate the virtual environment
.\.venv\Scripts\activate
# add the ipykernel package
uv add ipykernel
# synchronize the environment
uv sync
# delete the (temporary) project
cd $HOME
Remove-Item -Recurse -Force test___uv
# go back to the original working directory
cd $env:OLDPWD
```

3. Restart your computer

You may have to restart your computer to make `uv` available when you open a new terminal.

4. Check if you can use `uv` in a new terminal session after the restart

The following should work in a new PowerShell terminal, opened **after a restart of your computer**:

```
uv --version
```

5. If `uv` is not recognized after a restart, do this (each time you open a terminal):

It seems that your computer does not allow you to change the `PATH` environment variable. Therefore, you may have to set the settings each time you open a new terminal. You can use this snippet:

```
$uvDir = "$env:USERPROFILE\uv"
$uvCache = "$uvDir\.cache"
$env:PATH = "$uvDir;$env:PATH"
$env:UV_CACHE_DIR = $uvCache
$env:UV_CONFIG_DIR = $uvCache
[Environment]::SetEnvironmentVariable("PATH", "$uvDir;$([Environment]::GetEnvironmentVar
[Environment]::SetEnvironmentVariable("UV_CACHE_DIR", $uvCache, "User")
[Environment]::SetEnvironmentVariable("UV_CONFIG_DIR", $uvCache, "User")
uv --version
```

Fallback solution, if the *workaround* does not work for you either:

If you cannot use `uv` you can still do everything manually by creating and managing virtual environments using `venv` and installing packages with `pip`.

1. Download and install Python 3.13 from https://python.org/downloads

During the installation, choose a **custom installation** and make sure to select the option to "Add Python to environment variables".

2. Verify your python installation

Open a powershell terminal and run:

```
python --version
```

The output should indicate the version you installed earlier (3.13), e.g.:

```
Python 3.13.7
```

3. Run the following series of commands to check if everything works

The following commands will take a while, because a major advantage of using `uv` is its speed. Nonetheless, after a few minutes the following should finish without error:

```
# store the current working directory in a variable:
$env:OLDPWD = Get-Location
# create a (temporary) project directory in the user home
mkdir $HOME/test___venv -f
# change into the project directory
cd $HOME/test___venv
# create a virtual environment
python3 -m venv .venv
# enable script execution for the current terminal process (required for venv activation
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
# activate the virtual environment
.\.venv\Scripts\activate
# install a package
pip install pandas
# check if we can import the package
python -c "import pandas; print(f'successfully loaded pandas {pandas.__version__}')"
# delete the (temporary) project
cd $HOME
Remove-Item -Recurse -Force test___venv
# go back to the original working directory
cd $env:OLDPWD
```