UV Python Package Manager

Watch this video:
Stop Using Pip - This New Tool is 100x Faster (UV Tutorial)
https://www.youtube.com/watch?v=6pttmsBSi8M

- Can install packages and also python versions
- Very fast
- Auto creates the virtual environment. The virtual environment can also be created and activated manually. UV always uses a virtual environment by default. It gets created (.venv) when the project folder is created using ($ uv init).

UV can be used for a single script and for a project folder. The steps below are for a project folder only.

UV auto creates some files (like .venv) that are not visible in the terminal. This is the command to view hidden files in the terminal: (These files can be viewed in an IDE.)

```
$ ls -a
```

Check if uv is working.
This will give some output if uv is installed and working.

```
$ uv
```

When you already have a folder created

```
$ cd into the folder
$ uv init
```

To create a new project folder

```
$ uv init your_folder_name
```

To create a project folder for a specific python version. UV will download that version of python if needed. The –python flag can be used with many commands e.g. when creating a virtual environment manually.

```
$ uv init your_folder_name --python 3.11.6
```

List python versions that can be downloaded via uv. Also shows currently downloaded versions.
$ uv python list

Install a version of python
$ uv install python 3.8

Uninstall a python version
$ uv python uninstall python 3.8

Create a virtual environment
$ uv venv
Activate the environment
$ source .venv/bin/activate
Deactivate the environment
$ deactivate

Sync all packages and the python version
$ uv sync

Download a package (here it's the requests package)
$ uv add mcp (add one package only)
$ uv add mcp arxiv (add more than one package)

Uninstall a package
(This command behaves like pip uninstall, but it's powered by uv's faster resolver and environment management.)
$ uv pip uninstall <package-name>

How to run a Jupyter notebook from a uv project folder:
Ref: https://docs.astral.sh/uv/guides/integration/jupyter/
Within a notebook, you can import your project's modules as you would in any other file in the project. For example, if your project depends on requests, **import requests** will import requests from the project's virtual environment. To keep things simple don’t install packages from the notebook. Install them using **uv add** from the terminal.
$ uv run --with jupyter jupyter lab

How to install the mcp inspector

Install node. This is not done with uv.
$ brew install node

Check that node is installed.
Output should be something like: /opt/homebrew/bin/node
$ which node

cd into the uv project folder
$ cd mcp_project

Launch the virtual environment
$ source .venv/bin/activate

Launch the inspector. The server file is research_server.py
If asked to install @modelcontextprotocol/inspector@0.13.0
type **yes.**
It's better to use this command to start the server:
$ npx @modelcontextprotocol/inspector uv run research_server.py

If you use this command the server does not start when the
filename is changed:
$ mcp dev research_server.py

Launch the inspector.
You will be given a url in the terminal. Type this url in the
browser to launch the inspector.
Click **Connect** in the left panel.

Simple step-by-step process for running the mcp inspector

1. Create the server file e.g. server.py

2. Put the server file in a project folder e.g. test_server_folder

3. Cd into test_server_folder
```
$ cd test_server_folder
```

4. Initialize uv in a python 10 environment. The mcp package needs python 10 or newer.
```
$ uv init --python 3.10
```

5. Install mcp and any other packages that your code needs
```
$ uv add mcp
```

6. Launch the virtual environment
```
$ source .venv/bin/activate
```

7. Launch the inspector
```
$ npx @modelcontextprotocol/inspector uv run server.py
```

8. Click **Connect**, Select **Tools** at the top, Click **List Tools**, Click on a listed tool

9. On the far right type in the input for that tool and click **Run Tool**