Activity 4: Two virtual environments

Modern DevOps for Systems Biologists

Jun Allard

Last year you measured the brightest pixel in a microscope image, added the background offset your protocol calls for, and wrote the number in your paper. Today you run the same script on the same image, and get a different number.

A. Two answers from one script

i. Check that you have `uv`, the tool that builds and runs Python environments:

```
uv --version
```

If that fails, install it and open a new terminal:

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

ii. Write this into a file called `brightest.py`:

```
import numpy as np

# An 8-bit microscope image. Every pixel is a whole number from 0 to 255.
image = np.array([[12, 250], [180, 7]], dtype=np.uint8)

peak = image.max()      # the brightest pixel
corrected = peak + 100  # the protocol says to add a background offset of 100

print("numpy", np.__version__)
print("brightest pixel:", peak)
print("after offset:   ", corrected)
```

iii. Run it twice, against two different versions of numpy. `uv run --with` builds a throwaway environment containing exactly the version you name, runs your script inside it, and throws it away again:

```
uv run --with 'numpy==1.26.4' brightest.py
uv run --with 'numpy==2.3.2'  brightest.py
```

iv. Write down both answers.

(If `uv` pauses to download a Python as well as a numpy, that is expected: the interpreter is part of the environment too, and `uv` picks one each version of numpy can actually run on.)

B. Which number is wrong, and why

i. The brightest pixel is 250 in both runs, so the image is not the problem. 250 + 100 = 350, and 350 does not fit in the 0-255 range that `uint8` can hold. One version widens the number to make it fit; the other lets it wrap around past 255 and start again from 0. Say which version did which, and check that the wrapped answer is the one you would predict.

ii. Only one of the two runs said anything about this at all. Look again at the output of each — including anything printed before the numbers — and say what warning you got, from which version, and whether you would have noticed it in a script that printed a hundred lines.

iii. This change is documented in NEP 50, which shipped in numpy 2.0.

C. Say which version you meant

Make the version a property of the project.

i. Create a project and pin the version you decided to believe:

```
uv init brightness
cd brightness
uv add 'numpy==1.26.4'      # or 2.3.2 --- your choice, but say which
```

ii. Move `brightest.py` into that directory and run it with no `--with` at all:

```
uv run brightest.py
```

Confirm you get the answer you chose.

iii. Two files now record the environment.

```
cat pyproject.toml
cat uv.lock
```

`pyproject.toml` is what you asked for; `uv.lock` is what you got, down to the exact build of every package numpy itself depends on. The second is the one that makes someone else's run identical to yours. `uv init` wrote a third, `.python-version`, for the same reason.

iv. Commit both, together with the script. A collaborator who clones this and types `uv run brightest.py` now gets your number, on their machine, without asking you anything.

v. Finally, write a `QUICKSTART.md` in that directory: four lines that take a stranger from a fresh clone to the number your script prints. Assume they have nothing installed but `git`.