Python Project Structure

No jupyter

We’re not going to be using jupyter on this course.

It’s a useful tool and it’s great for interactive work, scripts etc.

However, you will find it very hard to make your code reproducible and reliable if you develop code like this.

Additionally, if you end up running your code on the rescomp cluster, you will find that you are not able to do this using tools like jupyter.

Packaging python projects

In order to work without jupyter, we will have to learn a couple of basics of working with python projects:

- Packaging (setup.py) - this is used to create an installable package from your code. This will allow you to import your library code flexibly and easily.
- Dependency management (pip) - this is how we keep track of the libraries our code depends on so that other people can install them easily and run the code.

Packaging python projects

The example project will give you a starting place for your future projects.

If you want to learn more:

A great resource on packaging is here:

https://the-hitchhikers-guide-to-packaging.readthedocs.io/en/latest/quickstart.html

There's a guide on pip here:

https://realpython.com/what-is-pip/

Project structure

The way we lay out a project affects how easy it is for others to use your code.

There should be a clear and obvious place for each part of your code, and you shouldn't have to hunt to find stuff.

The key principles here are:

- Be consistent
- Keep it simple (avoid endless nesting of directories for example)

Directory structure - Bad

I feel bad for picking on an example. This is from a peer reviewed paper however and, in my opinion, the code is low quality.

https://github.com/lfranzen/scwat-st

It will be very difficult for another person to extend this work and reproduce results. There is no assurance that the results are accurate and finding and fixing bugs in it will be extremely challenging.

Project structure - Good

Here is an example of a very well written project in the data science space:

https://github.com/facebookresearch/detectron2

It’s a relatively complex code base, but don’t be afraid of it. If you are ever wondering ‘where should this go’, this project is not a bad point of reference.

Command Line Interface (CLI)

We advocate creating a basic CLI interface to your code. This makes it very easy for others to reproduce results, because you can just give them commands to run.

It also allows you to easily automate running experiments with different parameters and so on.

We will use the Typer library for this in this course. It is a beautiful library, and will save you a lot of pain over argparse:

https://github.com/tiangolo/typer