Title: How to keep your Python code tidy with 4 easy tips.

Subtitle: Defining functions, classes, modules and config files to standardize your workflow in python.

Featured image:

Intro:

Those of us using python for data science without a solid programming background may agree that working code is good-enough code. But good enough may depend on how many times you are going to use it, and how much effort you will have to put into deciphering it in the future. Your perfectly working code can get very long very quickly and loosing track of what part of your code was doing what is only natural. I will share 4 easy tips that will help keep your code clean. Let’s get started.

Body:

When parts of your code are repeated and used several times, it is useful to wrap these sections up as **functions**. These functions have to be defined once at the beginning and can be later called repeatedly in your code in a single, very informative line.

If these functions get somewhat bigger, and especially when they need several arguments, it may be useful to **define a class** instead, with specific attributes and methods.

A further problem with an unclear script structure appears when you need to input hardcoded parameters directly into the source code (e.g., file paths). A very handy trick to make your life easier in the future, and also help out colleagues that may use your code, is to bundle and outsource hardcoded parameters to a separate **config file** (e.g., config.yaml). This file will contain all parameters that need to be adapted and will be human readable (for those not used to read code).

Last but not least, you can also outsource classes and functions to other python files and load them as **modules** in your tidy and manageable script.

Let’s have an example.

Usecase:

Building a frame of body part coordinates from a csv data file.

We will start off with a functional but unstructured python script to (1) read the csv file, (2) build a skeleton for each time frame connecting given body parts, and (3) calculate basic kinematics such as speed, trajectory etc. [Python script here](https://github.com/Guillermo-Hidalgo-Gadea/UQOAB/tree/main/Pose%20Analysis)

Then, I will show how to rearrange the structure of your script by defining functions.

In the next step, we will create a standalone class with attributes and methods, and in our code, we will only have to create an object from that class and apply its methods.

In cases in which class attributes or function arguments need to be

Conclusion:

* define functions
* define classes
* outsource functions and classes to separate scripts (modules)
* outsource hardcoded parameters to separate config files (e.g., yaml)