# Intro to production code

1. Considerations

* Testability & Maintainability
* Scalability & performance
* Reproducibility (version control, tracking, release version, etc.)

1. Python packages

* Module: a file which contains various Python functions and global variables. It is just a file with a .py extension which has python executable code
* Package: a collection of modules

1. Package structure

Parent/

* Regression\_model/
* Config/
* Datasets/
* Processing/
* Trained\_models/
* Config.yml
* Pipeline.py
* Predict.py
* Train\_pipeline.py
* VERSION
* Requirements/
* Requirements.txt
* Test\_requirements.txt
* Setup.py
* MANIFEST.in
* Pyproject.toml
* Tox.ini
* Tests/

# Reasoning behind the Prod code

1. How do we know how to structure our own projects?

* Conventions
* Versioning
* Config
* PEP8/ linting tools
* Packaging mandatory files
* setup.py
* MANIFEST
* Software engineering best-practices
* For packages
  + Separating train & predict code
* General
  + Testability: can we write unit tests for one functionality? E.g., data\_manager function
  + Separation of concerns, i.e. modularity: data manager + data validation
  + SOLID: what is the role of the, e.g., train\_pipeline module
  + Maintainability: imagine a PR?

1. Notes

* Pick a convention (e.g., for config) – choose one your friends/colleagues use. Established conventions \*usually\* adhere to good practices (and learning when they do not take practice)
* Follow a standard where possible (e.g., Python packaging)
* Optimize your code for readability – this includes structure. The ‘principle of least astonishment’ is useful
* Remember: the one who maintains your code afterwards is a psychopath who knows where you live 😊

# Package requirements files