Development guidelines

Starting a new project

- work under **source control**
- provide a way to be **built and run in one step**
- provide a way to **run tests in one step**
- have a `README.md` file in the root folder of the project that explain
    - what the application does
    - how to install the required development environment
    - how to run the application
    - how to run the tests

Making decisions

- Take the time to discover framework / tools / libraries.

- **Make small prototypes** using different library / framework before choosing one.

- Take the time to learn **all what your library / framwork can do**. Use the library / framework whenever you can, don’t reinvent the whell and don’t add a library if there is no clear benefit adding it.

- Use the right tool for the job. Don’t try to bend something on doing something it is not supposed to.

Coding

- **Do not repeat yourself (DRY):** Make sure the code doesn’t have too much duplication.

- **Clean code:** Make sure you delete unused code. You can find it in your source control system.

- **Keep it simple and stupid (KISS):** Simpler is always better. Reduce complexity as much as possible. Less code equals less bugs equals easier maintenance.

- **Boy scout rule:** Always leave the campground cleaner than you found it.

- **Follow standard coding conventions:** Check the standard coding convention for you language, the libraries you're using …
- **Root cause analysis:** Always look for the root cause of a problem. Otherwise, it will get you again and again.
- **Work your abstractions:** Each class should abstract one and only one thing (SRP). The abstraction should not be too thick nor too thin. Each method should do only one thing, should be documented and should not be too big.
- **Consistency:** If you do something a certain way, do all similar things in the same way. Same variable name for same concepts, same naming pattern for corresponding concepts.
- **Pick good names:** Take the time to choose good descriptive names that reflect the level of abstraction of the class or method you are working on. The name of a method should describe what is done, not how it is done.
- **Documentation:** Every function/method/class should have a standard documentation string that describe what it does according the abstraction it belongs to. Technical details about the implementation should go into comments, not into documentation.
- **Comments:** Everytime it's not clear what some lines of code are doing, those lines of code should have a comment that explains what they are doing and how.
- **Language:** Everything should be written in english and interface strings should be translated straight away.

Testing

- **Learn to test!** You should know about parametrized test, test context, mocks, testing frameworks, testing libraries … Every time you learn how to build something, you should also learn how to test what you built.
- Almost everything should be tested using unittests. They should be fast, isolated and repeatable.
- Your test suite should be strong enough so that you have enough confidence to:
    - Add new features
    - Upgrade rapidly a framework/library
    - Refactor a lot your code