**Git and GitHub – youtube video without spyder**

Basic terms:

* Git = version control system that keep track of all your operations
* directory (memory location)
* terminal or command line (navigate through programs, create new file, install etc…)
* CLI = command line interface: some programs require you to interact with them in the command line, is an alternative way to run a program, alternative to more user friendly interfaces
* cd = change directory (follow by a direction)
* Code editor = user friendly interface for coding
* Repository = means typically project, or the folder/place where your project is kept.
* Github = website where we can host your git repository, work in gropu, orginize tour project

Why CLI? Faster and good habit

Git commands

* clone: Bring a repository that is hosted somewhere like Github into a folder on your local machine
* add: after having made a change, this command will track your files or changes in Git
* commit: save your files in git
* push: after having made changes on a file on your pc and having commited it to git, upload git commits to a remote repository, like GitHub
* pull: download changes from remote repo to your local machine, the opposite of push

README file is typically the most basic file you’ll find in a repository: typically contains information on what the project is about, what it does and any other relative information.

We can starts to add files into the repository, the first is the README file, saved in the .md extension. After havnf saved it, github advise us about the last operation, in this case “Create README.md”. If you opne it from the repository and modify it, github will say “Upload README.md”. You can anyway change this advice in the way you prefer.

It’s possible form github (voice “History”) to see all the steps done in the history of your repository. Colors suggest the operation you have done (green: added, red: deleted).

In order to take the project from github to my local machine I’ve to use the terminal, writing the right command.

Using:

> git clone “link found on github”

And your repository will be localized also on your local machine.

Using Spyder, we are able to open and modify .md files. Spyder presents also a terminal from which, using the appropriate commands, it’s possible to manage files from/to local machine to/from GitHub repository, as if it’s the miniconda shell.

To understand what we have to do: <https://docs.spyder-ide.org/current/projects.html>

**Lesson 4**

Note that the last lessons and the actual one (so the third and the fourth lessons) are the most important. Write to the professor in order to have access to the records.

Main topic: testing

Testing code is as much code as the main one.

A code presents a bug when behave differently from what it’s written in the documentation.

In the documentation we include the external documentation (**the manual**) the internal one (**docstrings**) and the implicit one (**name** of the function and of its **arguments**)

Can be considered bugs also **comments** and **variable names** that do not align with what the code is doing, even if less so.

The tips you can leave to let other understand a code are:

- name given to functions

- name given to variables

- comments

Numpy is a great library that you need to know. It has a good documentation that can help you a lot in understanding both its functionality and the way you’ve to write a documentation.

In order to have information about a documentation on sider you have to use the command > help(“name of the function”);, or press ctrl+I with the name of the function written in the command line.

Def: Pure function if:

- deterministic nature (same output for identical inputs)

- its casting do not change the state of the rest of the program

These function are the easiest one to be tested, it’s easy to work with them.

Using global variables or non local ones inside a functions, so this function cannot be pure.

If a function is hard to test, it will ne difficult to use.

Types of test

1. advancedment test
2. regression test

Each time you find a bug (in a particular situation) you try to fix it. After that you have to program a test that put your system in a situation like the one that gave error and analyze the results.

Tests are also ways to have an historycal memory of the problem your code showed.

There are postive and negative tests: positive = the code do what I want at given inputs, negative = code fails as i expect when I give wrong inputs

Strategy of testing

1. informal testing: after having defined a function we always have to check if some parameters give the expected outputs. The assert keyword is used when debugging code.The assert keyword lets you test if a condition in your code returns True, if not, the program will raise an AssertionError. You can write a message to be written if the code returns False, as an error warning.
2. unit testing: we will use the library pytest. pytest is an external program that runs all the function of the form: “test\_”
3. property testing

general rule: if a positive test continuosly give a positive feedback 🡪 you have to be sure that giving it wrong inputs the output will be false, or gives an errore! Same for the negative test clearly!