UE SAR INFO SI-SSG FISE A1 S2

Session 2 - Practice with tools and libraries

remous-aris.koutsiamanis@imt-atlantique.fr

Remous-Aris Koutsiamanis

DAPI / IMT Atlantique, Nantes

STACK (LS2N / INRIA)

2021-04-20

Overview

In this session we will:

- See how to find information about a Python library
- See how to read library documentation
- Point out a pitfall regarding git and .ipynb files
- Instructions for changelog management
- See brief examples of multiple libraries

Finding a library

- Pick a library
- For example the first library we will work with, named `pandas`. Two options:

  1. Do a general search for that name
     - If the name is common it might be harder to find the correct one
     - Certain libraries are poorly named (e.g. `mesa`: search)
     - Try searching for `libraryname python` or `libraryname pypi`
     - Look for links within the PyPI `pypi.org` site
  2. Go to the Python library repository PyPI
     - Contains most available Python libraries and links to documentation

General library information

- When at the library's PyPI site, focus on:
  - Latest version
  - How to install (with `pip`)
  - Link to project homepage
    - `pandas` has one, others do not or it's just the GitHub repository
  - Link to documentation
  - Statistics
  - Requirements

Reading Documentation (1)

- Go to the pandas documentation
  - Depending on the size and especially the popularity of the library, there will be various sections
- General information (e.g. purpose)
- Quick startup, often including
  - Installation instructions
- User guide, often including
  - Basic concepts
  - Simple examples
  - More complicated examples and concepts

Reading Documentation (2)

- API reference, i.e., how to get the "official" specification for each class/method/function
- Developer guide, i.e., how to modify/augment the library's code
  - How to contribute, i.e., how to participate in the community

Looking up functions and parameters

- Example, **get** the **maximum** value of a **column** of data in **pandas**
  - It is very helpful to mention the library and potentially also the language
  - *English* generally gives more results (and often more accurate)
  - Try to use terms which are already used in the library (`maximum`, `column`)
    - May require some initial familiarity with the library
  - Search for `get maximum column pandas`: search
    - Third-party site (kite.com):
      - How to find the max value of a pandas DataFrame column in Python
    - Official documentation:
      - pandas.DataFrame.max on pandas.pydata.org/docs

Using code from other sources

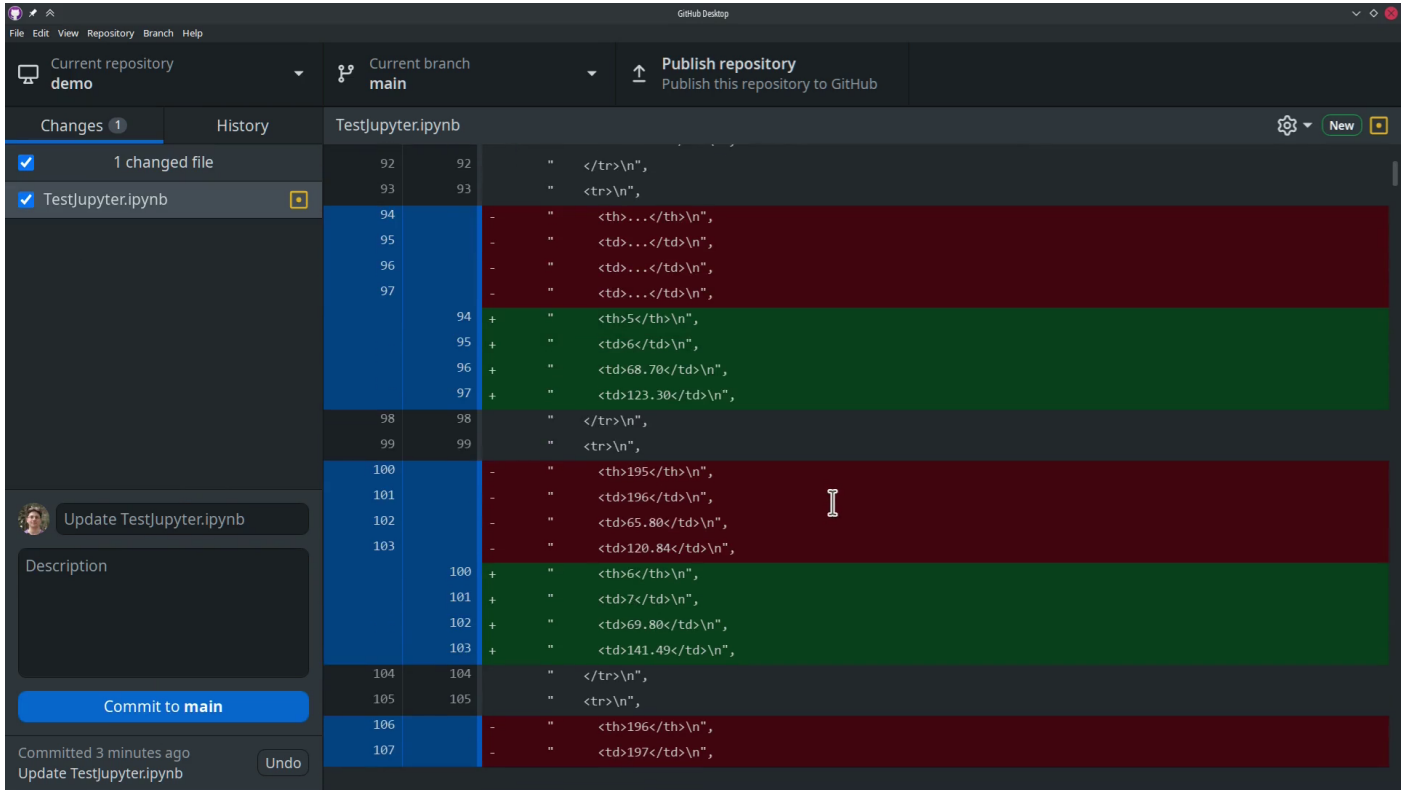
Note on git and `.ipynb` files

- Git works with both *textual* and *binary* files
- Jupyter Notebook files (`.ipynb`) are *textual*, **but**...
  - Temp files in `.ipynb_checkpoints`: **Ignore them by adding them to .gitignore**
  - Outputs from cells are saved inside the file !
- Git is useful for finding what has changed and comparing versions
  - Much harder to do when the cell output is present

In *Jupyter Lab*: Go to menu *Kernel → Restart Kernel and Clear All Outputs…*
Then, save the file and commit. *Do not re-run any cells before saving and committing*.

Video on committing .ipynb files

- Committing .ipynb files



How to record your progress

- I have pushed a file named `CHANGELOG.md` into your repositories.
  - The `.md` file extension indicates a MarkDown format
  - The way to write in it is described in KeepAChangelog
- Before continuing your work, make sure to `git pull`
- Modify and commit your `CHANGELOG.md` when you have made some progress
  - As a guideline, minimum once per session (at the end)
  - Maybe more often, for example after every few exercises

Next steps

Go to the **Moodle** course and look for the **Session 2 folder**