**Static site builder**

This is a project proposed to Python students.

What is a static site?

A static website is a site composed only of files in a folder:

1. HTML files,
2. CSS files,
3. JavaScript files,
4. images,
5. videos…

This is opposed to dynamic websites, where some of these files are generated on the fly by software, for example from data in a database.

**Host a static site.**

Hosting a dynamic site is more complex than for a static site, it is indeed necessary to install the software that will generate the files on the fly. By cons, hosting a static site is relatively simple, just have a small web server that provides the folder containing static files.

**Github.**

Github provides free static site hosting. Just create a git repository with Github, and commit to a specific branch. Your site is then accessible at the following address: https://your\_login.github.io/your\_name\_of\_depot/

More information on:

* <https://pages.github.com/>
* <https://www.christopheducamp.com/2013/12/21/demarrer-avec-pages-github/>
* <https://developer.mozilla.org/fr/docs/Apprendre/Utiliser_les_pages_GitHub>

**Use a web server**

Tools like Apache or Nginx allow to make accessible your site by internet or intranet:

* <https://httpd.apache.org/docs/trunk/fr/getting-started.html>
* <http://sametmax.com/servir-des-fichiers-statiques-avec-nginx/>
* <https://doc.ubuntu-fr.org/nginx>
* <https://nginxconfig.io/>

Python provides you with a minimal web server, for example to go to http: // localhost: 8080 / and see the static site whose files are in ./folder\_of\_my\_site/.

* python -m http.server 8080 --directory ./folder\_of\_my\_site/

**Generate a static site.**

The files understood by an internet browser are in HTML / CSS / JavaScript formats. You may not want to type HTML when writing a blog. It would be convenient to generate web pages from a simple text format, such as the markdown (https://guides.github.com/pdfs/markdown-cheatsheet-online.pdf), the language used to write the document that you please read on (https://raw.githubusercontent.com/vulvereau/site\_static/master/README.md).

Some open-source tools already do this, some known and in Python:

* <https://blog.getpelican.com/>
* <https://www.getlektor.com/>
* <https://www.mkdocs.org/>
* <https://github.com/eudicots/Cactus>
* <http://www.sphinx-doc.org/en/master/>
* <https://www.getnikola.com/>

**Project.**

You will make a tool that converts a folder of markdown files and images into another folder containing the files of a static site. HTML will be generated from the markdown, and this HTML will be mixed with templates of web pages to generate pages all conform to the same model (for example with the same logo, the same summary of website, the same referenced CSS file ... ).

The markdown files can be created:

* with Visual Studio Code
* with <https://github.com/ncornette/Python-Markdown-Editor>
* with <https://pandao.github.io/editor.md/en.html>
* with <https://dillinger.io/>

To give an idea, the most basic version of the project can be done in less than 100 lines.

**Realization of a command line interface.**

You will build a command line tool to generate the static site files. It will take at least as parameters:

* -i ./a\_folder or --input-directory ./un\_folder: the path of the source file folder (containing the markdown files)
* -o ./one\_other\_folder or --output-directory ./one\_other\_folder: the path of the folder where the files generated for the static site will be put
* if the file already exists, you are free to either erase it or write it in for updates (this will be explained in the manual of your tool)
* you can choose the naming convention you like for the generated files, for example you can use as prefix the name of the corresponding markdown file (this will also be explained)
* -t ./other\_folder or --template-directory ./other\_folder: possibly the folder containing templates of web pages to be completed
* -h or --help: to display help to explain command parameters

You can optionally add parameters like:

what you want:

* -k or --kikoo-lol who will add in the text "kikoo", "lol", "mdr", "ptdr" or who repeats letters as in Hellllo, and other deformations of French (https: // fr. wiktionary.org/wiki/kikoolol)
* -a or --achtung to help the Germans read our French blogs. If you apply the rules described here, you will help us speak in the pronounziation of our language and in the non-zekri kompréhenzion.

You can use :

* sys.argv (but I do not recommend it, https://docs.python.org/en/3/library/sys.html#sys.argv)
* argparse (https://docs.python.org/en/3/howto/argparse.html)
* click (https://click.palletsprojects.com/en/7.x/)

It is possible that your project is used for example with:

python3.7 generateur.py --input-directory ./dossier\_markdown --output-directory ./dossier\_resultat/ --achtung -k

**Converting markdown to HTML.**

You must at least convert the following syntaxes:

* #, a level 1 title in <h1>
* ##, a level 2 title in <h2>
* ###, a level 3 title in <h3>
* Convert unordered lists to <ul> and <li>
* Convert URLs (http://something.com) to <a href="http://something.com"> http://something.com </a>
* a text \*, an important text in <em> a text </ em>

You can do these conversions using any of the following:

* the basic functions of Python for strings
* regular expressions (https://docs.python.org/en/3/library/re.html)
* a community package

1. <https://github.com/Python-Markdown/markdown>
2. <https://github.com/trentm/python-markdown2>

**Quality of code.**

You will make sure to respect:

* PEP 8: https://www.python.org/dev/peps/pep-0008/ (you can help with https://github.com/ambv/black and https://github.com/hhatto/ autopep8)
* PEP 20: https://www.python.org/dev/peps/pep-0020/
* more details on https://vpoulailleau.wordpress.com/2018/12/04/un-code-pythonic/

**Rendering on Github.**

Your Python personal project will be posted on Github and a link to the public repository will be provided.

At the root of your git repository will be a README.md file that will explain how your project works, how to use it, what its license is .

Open-source project.

You can do a free and open-source project. Many Python projects use the MIT or BSD 3 clauses license, these licenses are easy to read and very permissive. You can also use a stricter license like the GPL which requires that modified versions of your project are also open-source.

You can read the BSD 3 clauses license of the project:

* https://github.com/voulailleau/simplelogging at
* <https://github.com/voulailleau/simplelogging/blob/master/LICENSE>.

You can make your project installable by the Python community by distributing it on the Python Package Index (https://pypi.org/), such as https://pypi.org/project/simplelogging/.

To help you in this adventure, you can use:

* <https://github.com/audreyr/cookiecutter-pypackage>.

**Statement Updates.**

It is possible that, following questions received, the statement will be updated. The latest version of the statement is available here: https://github.com/vpoulailleau/site\_statique. You can see its history on:

* <https://github.com/vpoulailleau/site_statique/commits/master>.

**Objectives.**

It goes without saying that to document, to copy code (in respect of the licenses), to discuss with other coders is strongly recommended to progress. Look at how others are doing and do it your way. Be able to explain what you did.

As a reminder, unlicensed code is by default protected by copyright, so you do not have the right to copy it, except with the agreement of the author.

**Evaluation.**

The project is suitable for all levels, a basic version is feasible, but the project can go up to the realization of an open-source tool made available to the community.

The evaluation criteria are as follows:

* involvement (visible among others by the history of your deposit git)
* respect for PEP 8
* respect for PEP 20
* README.md file quality
* achievement in accordance with the basic functionality of this statement
* bonus points if you go further
* You can get an idea using the following scale: https://github.com/vouleau/site\_static/blob/master/checklist.md.

Good learning, and good project.

**Algorithm** **ideas.**

Command line

Start by setting up the command line that will accept the different parameters mentioned:

* Display of the help message (argparse and click know how to generate it automatically).
* View the name of the folders passed as command line parameters.

**Generating static files.**

**Source file path**

It will be necessary to make a treatment for each file markdown present in the folder containing the files markdown. Pathlib and its global method can help.

**Convert markdown to HTML.**

Markdown files are text files. You have to open them, read them, and generate the corresponding HTML. The result of each conversion is stored in an HTML file in the folder that will contain the static files (folder provided by the command line). The first conversions (titles) can easily be realized with the replace string method. It is possible to go through the regular expressions, you can even get help using cursive\_re presented on:

* <https://vpoulailleau.wordpress.com/2018/11/29/des-expressions-regulieres-lisibles/>.

**Template mechanism.**

The simplest model page mechanism is:

* You create a default HTML page, and without content, just the summary, the logo, the loading of CSS ...
* Instead of the content in the HTML page, you put the text "REPLACE\_ME"
* When you generate an HTML page, you open the template page, and you replace "REPLACE\_ME" with the HTML that you generated from the markdown
* You can use https://getbootstrap.com/ to quickly make more advanced HTML (responsive design, menus, drop-down lists, slide show ...)
* For those who want to make a more advanced mechanism of template pages, you can look at http://jinja.pocoo.org/, https://genshi.edgewall.org/, https://www.makotemplates.org/, https : //opensource.com/resources/python/template-libraries

Saharsh thanks a lot man. The links I have highlighted you can copy and paste it on the web so that you can access it.(I know you already know XD)

Thanks a lot for your help <3 <3 <3