# Commit Log Generation

## Project Structure

### Run program:

>>> python3 run\_lmg.py <lang> <fmt> [-n count]

* <lang> is either “java” or “python”
* <fmt> is the project name that will affect output format
* [-n count] controls how many repos to process

### Files:

* run\_lmg.py

Entry point of the project. A bit messy since some repos filtering steps are involved (see *clean\_repo* function and *Utils* class). Leave that aside, the script takes the arguments passed via command line, and spawn one process for each repo by calling the data\_preparation.py file (see *run* and *clone* methods).

* data\_preparation.py
* the self.subject field in *Controller.process*, is the whole unified diff of a repo, which is generated by the command line in the get\_commit.sh file (see below)
* the *main* method calls *Controller.process*; the *Controller.process* method then split the unified diff into individual commit (see *self.commits\_lst*). Next, the Controller employs two groups of rules to filer individual commit: rules apply to Commit Message, and rules apply to the commit (such as filter out merge commit etc.)
* For each commit that pass the filters, a Commit object gets created (see utils.py).

Table

Description automatically generated with medium confidence

* utils.py
  + The *utils.Commit* class represent one commit. It has two subclasses: one for java and one for python. The subclasses are only in charge of saving the results differently for java and python.
  + The *utils.Commit class* contains several filtering rules via its private methods.

Graphical user interface, text, application

Description automatically generated

* get\_commit.sh

a script for generating the unified diff commit log for each Github repository

## Other resources:

1. Pydriller: <https://github.com/ishepard/pydriller>
2. Github REST API: <https://docs.github.com/en/rest>
3. GitPython: https://gitpython.readthedocs.io/en/stable/intro.html