MY FIRST OPEN SOURCE CONTRIBUTION

IRENE ARAPOGIORGI – t8160012

MAY 2020

ABSTRACT

This year's course of Software Engineering in Practice set the big challenge to comprehend and contribute to a fully well-developed open source project and, as a result, to familiarize with the real conditions of the software development process. The journey was definitely full of excitement and productive stress, but also with huge relief and satisfaction as everything went out great in the end.

INTRODUCTION

Having been personally interested in software development, the Software Engineering in Practice course played a highly important role in expanding my existing knowledge of software design, development, testing, maintenance and much more. Under the supervision and guidance of professor Diomidis Spinellis and PhD candidate Antonis Gkortzis, I was able to gain more experience and familiarity with the field of software engineering and take a big first step by contributing to the open source project community.

1. PROJECT SELECTION

I knew from the start that I wanted to choose a project that would pique my interest, so I ended up with InstaPy, a pythonic tool for automated Instagram interactions, which is highly used on social media marketing campaigns, a field that I have worked on again in the past and interests me a lot.

At the same time, InstaPy was quite intriguing in mind as it met all the requirements of a friendly open source project that I could easily contribute to. Both of their main communicating channels, GitHub and Discord, are widely used by the community, the project itself is continuously developing and receptive to pull requests, and its structure, code and documentation are easy to comprehend.

So, InstaPy [1] is an upcoming project officially created by a Computer Science student named Tim Grossman (@timgrossmann on GitHub) and aims to help either individuals or small businesses to grow their social media audience and thus achieve their marketing goals by automating social media interactions that are part of their marketing campaigns.

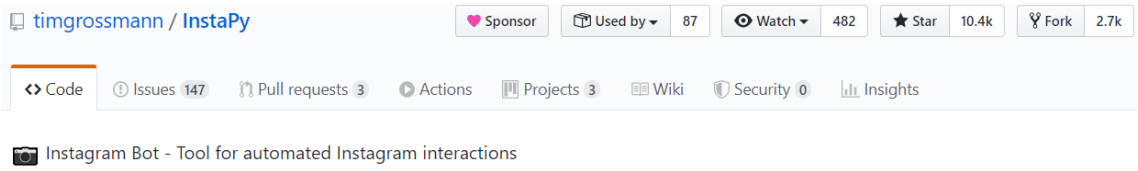


Figure 1: InstaPy repository on GitHub

Using InstaPy is quite easy. Simple users just install it from pip and create their own personalized quickstart.py file which stores their settings for upcoming interactions. Then, running on command line the "python quickstart.py" command just starts the procedure of automated interactions.

Figure 2: InstaPy, a tool for automated Instagram interactions [2]

In order to understand deeper this specific project, I spent a few hours reading each file's code and the entire documentation, and then I started focusing into my own ideas of enhancing the project. The long list of already existing feature requests on GitHub and my own experience with the needs of a social media marketing campaign led to some possible contribution ideas some of which were actually implemented in the end.

2. PROJECT CONTRIBUTION

My overall contribution to InstaPy consists of two separate enhancements of the project. The first one is actually a whole new feature which would be used supplementally with some already existing functions, while the second one is an additional check that was already mentioned as "to-do" from the developing team.

2.1 FIRST CONTRIBUTION

2.1.1 OVERVIEW

When studying InstaPy's documentation, I noticed plenty of functions using lists with usernames, hashtags or comments which are created by the users themselves inside the script in order to target a specific group of people or hashtags and even comment on posts on Instagram.

```
# Like posts based on hashtags
session.like_by_tags(['natgeo', 'world'], amount=10)
```

Figure 3: Like post based on list of hashtags

```
# Follows the followers of each given user
# The usernames can be either a list or a string
# The amount is for each account, in this case 30 users will be followed
# If randomize is false it will pick in a top-down fashion

session.follow_user_followers(['friend1', 'friend2', 'friend3'], amount=10, randomize=False)
```

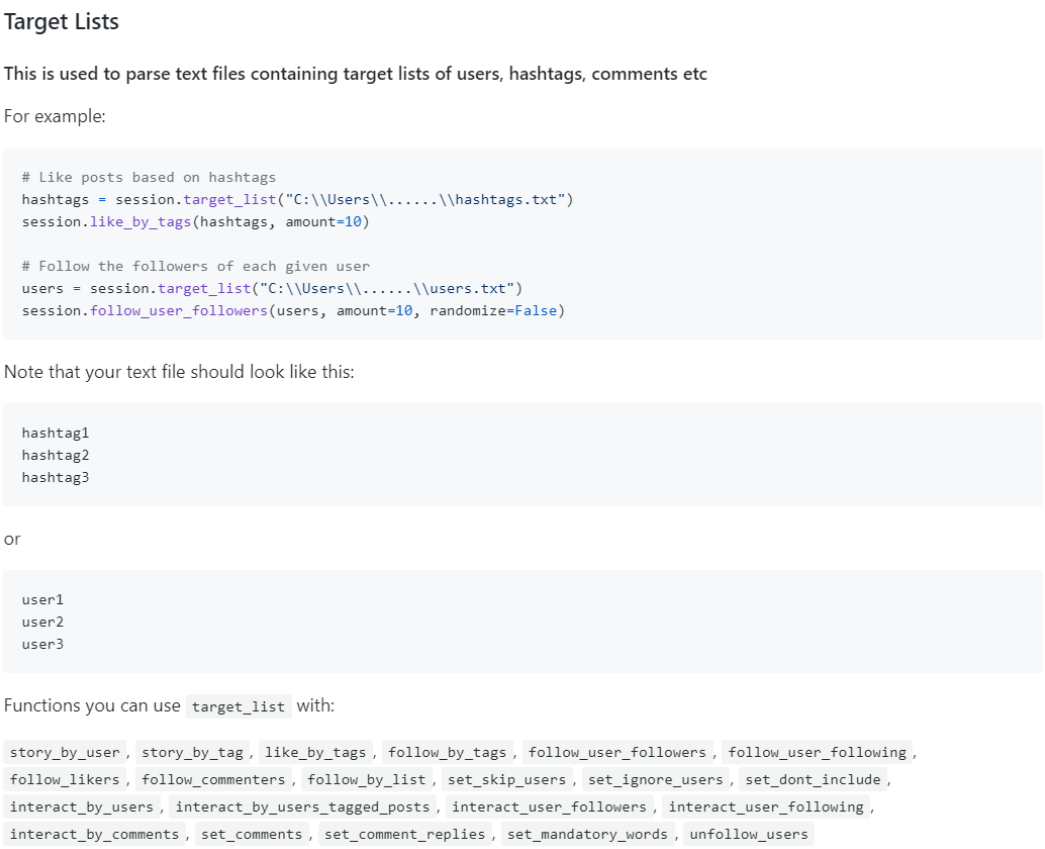
Figure 4: Follow the followers of each user in given list

Immediately I thought, what if someone wants to target different but specific groups of hashtags for specific campaigns and wants to store these lists in a text file and just parse that file instead of having to type all these hashtags again and again every time they run each campaign? The same applies for specific comments to be left on posts or specific users to be reached. The ability to store target elements in reusable text files can satisfy the need for automation and, for me, it was a very appealing idea for my very first contribution.

In total, my contribution affected 7 files, added 94 lines of code, deleted none and introduced two new functions related to the implementation of text files handling.

While implementing my idea, I didn't need to adjust the already existing related functions, but I did created two new ones to be used supplementally - the first one being a helper function and the second one being the function handling the given text file and extracting its elements for further use. The latter function splits each line of the given file containing an element, removes possible leading whitespaces, tabs and newline characters and returns the final elements (hashtags, username or comments) as a list. That list can be parsed into any related function by the user inside the quickstart script.

You can take a deeper look at my implementation in the following link (spoiler alert, my pull request got merged): https://github.com/timgrossmann/InstaPy/pull/5443



Target Lists

This is used to parse text files containing target lists of users, hashtags, comments etc

For example:

```
# Like posts based on hashtags
hashtags = session.target_list("C:\\Users\\......\\hashtags.txt")
session.like_by_tags(hashtags, amount=10)

# Follow the followers of each given user
users = session.target_list("C:\\Users\\......\\users.txt")
session.follow_user_followers(users, amount=10, randomize=False)
```

Note that your text file should look like this:

```
hashtag1
hashtag2
hashtag3
```

or

```
user1
user2
user3
```

Functions you can use `target_list` with:

`story_by_user` , `story_by_tag` , `like_by_tags` , `follow_by_tags` , `follow_user_followers` , `follow_user_following` , `follow_likers` , `follow_commenters` , `follow_by_list` , `set_skip_users` , `set_ignore_users` , `set_dont_include` , `interact_by_users` , `interact_by_users_tagged_posts` , `interact_user_followers` , `interact_user_following` , `interact_by_comments` , `set_comments` , `set_comment_replies` , `set_mandatory_words` , `unfollow_users`

Figure 5: Documentation of text files handling

2.1.2 CODE QUALITY

It is highly suggested by the development team to use a specific Python formatter named Black when contributing to InstaPy. Following that, I reformatted my own code before making a pull request and I also made sure my functions to be well documented according the project's principles. Finally, I checked for any linting errors using pylint, a Python library which checks with coding standards and detects any errors and duplicated code.

Figure 6: Reformatting using Black

2.1.3 CONTINUOUS INTEGRATION & TESTING

Testing my code was a step I couldn't pass over, so I created a testing file containing unit tests of my implementation. Using a Python library named unittest, I tested three different test cases. The first one checked whether a file containing valid entries matches an already initialized list of expected elements, the second one checked the same but when parsing a file with invalid entries (in order to see whether my function actually extracts the correct elements) and the last case checked whether a FileNotFoundError was raised when parsing an invalid file path. Thankfully, my implementation passed all unit tests and also had 100% test coverage, as extracted using Coverage.py, a tool for measuring code coverage of Python programs.

Figure 7: Unit testing and Code Coverage

Moreover, InstaPy uses Travis CI for Continuous Integration, so I was very thrilled to see that my first code contribution passed all build tests and could possible be integrated into the existing project.

2.1.4 PULL REQUEST

The last step before considering myself a contributor to InstaPy was to finally submit my work. Two of the main developers of the project checked my pull request and were very satisfied, so no alterations were asked, but to fill in the documentation and update the changelog file. After that, my first contribution was merged!

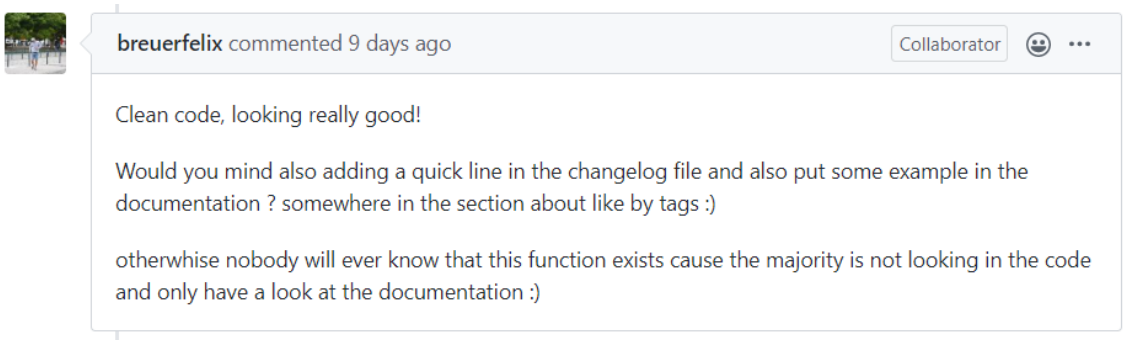


Figure 8: Code Review

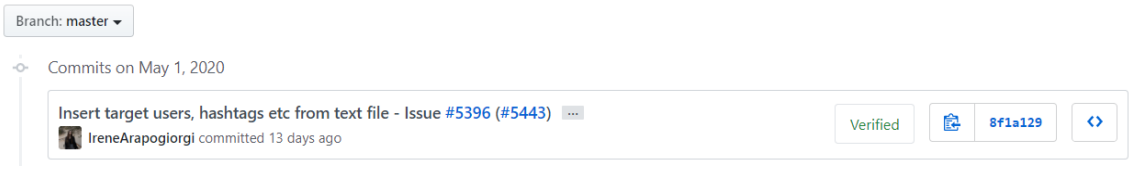


Figure 9: Merged Pull Request