

MY FIRST OPEN SOURCE CONTRIBUTION

IRENE ARAPOGIORGI – t8160012

JUNE 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.

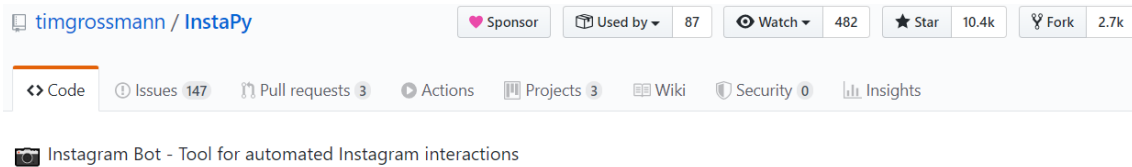


Figure 1: InstaPy repository on GitHub

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.

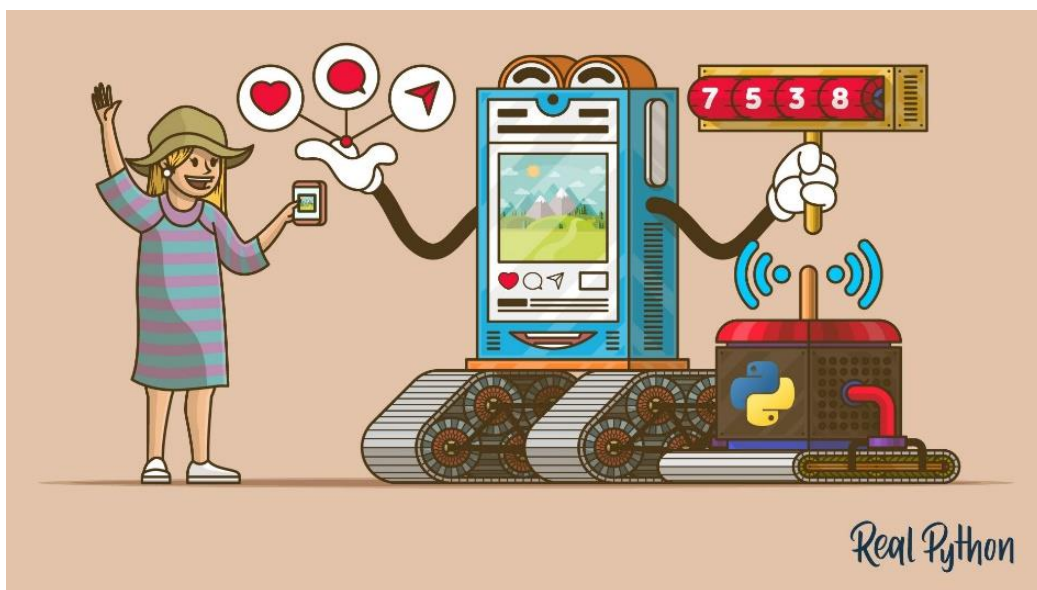


Figure 2: InstaPy, a tool for automated Instagram interactions [2]

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 new feature which would be used supplementally with some already existing functions, while the second one is an additional needed 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 create two new ones - 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:

<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.

```
irene@LAPTOP-SL23CI1H MINGW64 ~/GitRepositories/InstaPy (text_file_reading)
$ black instapy/util.py
All done! 0 0 00 00
1 file reformatted.

irene@LAPTOP-SL23CI1H MINGW64 ~/GitRepositories/InstaPy (text_file_reading)
$ black instapy/instapy.py
reformatted instapy\instapy.py
All done! 0 0 00 00
1 file reformatted.

irene@LAPTOP-SL23CI1H MINGW64 ~/GitRepositories/InstaPy (text_file_reading)
$ black tests/test_file_handling.py
reformatted tests\test_file_handling.py
All done! 0 0 00 00
1 file reformatted.
```

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 possibly be integrated into the existing project.

2.1.4 PULL REQUEST – CODE REVIEW

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!

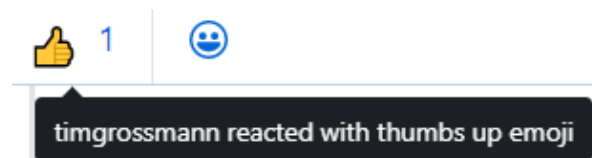


Figure 8: Thumbs up from the creator of InstaPy

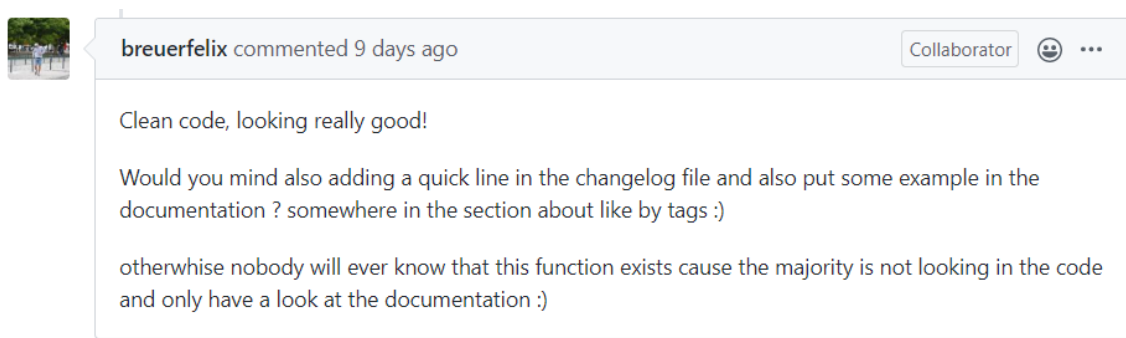


Figure 9: Code Review

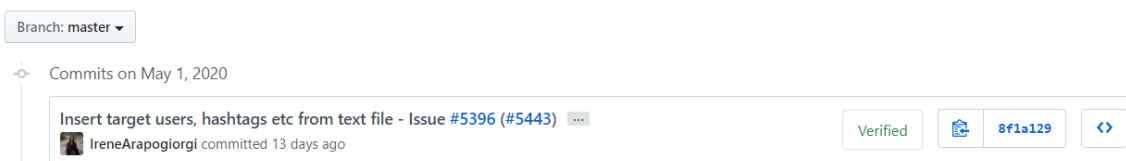


Figure 10: Merged Pull Request

2.1.5 CODE MAINTENANCE

An interesting issue occurred a few days after merging my pull request and I realized I had made a mistake when handling the elements inside text files. That was a case I had missed when testing my implementation, not due to a missing unit test, but because of not having a comment example inside the text files used for testing. Trying to convince myself that I didn't ruin my entire contribution process, I immediately fixed everything and created a new pull request describing what went wrong as seen in the following screenshot. That pull request also got merged.

Fix text file handling for comments #5490

[Edit](#)

Merged breuerfelix merged 2 commits into `tingrossmann:master` from `IreneArapogiorgi:comment_handling` 6 days ago

Conversation 1 Commits 2 Checks 0 Files changed 4 +8 -9

IreneArapogiorgi commented 8 days ago Contributor

Hello, I found out a mistake I made while implementing the `file_handling` function mentioned on issue #5396 and submitted on pull request #5443.

I was removing all spaces found before, after and in-between target words and also transforming all letters to lowercase, which would transform a possible comment from `This is a comment` to `thisisacomment`.

I fixed that and updated the unit tests included.

Reviewers
breuerfelix ✓

Assignees
No one assigned

Labels
None yet

Figure 11: Fix comment handling

2.2 SECOND CONTRIBUTION

2.2.1 OVERVIEW

After finishing my first contribution, I had my eyes on another enhancement of InstaPy which was already mentioned in two to-do comments inside a certain file of the project. I found out about those to-do comments while I was taking a deep look into the project's code at the beginning of my contribution journey.

To be more specific, a file called `relationship_tools.py` stored two functions which grab followers and following users of a given target user and which are both used by other functions for specific Instagram interactions.

```
49     # Get followers count
50     followers_count, _ = get_relationship_counts(browser, username, logger)
51
52     if grab != "full" and grab > followers_count:
53         logger.info(
54             "You have requested higher amount than existing followers count "
55             " ~gonna grab all available"
56         )
57     grab = followers_count
58
59     # TO-DO: Check if user's account is not private
60
```

Figure 12: To-do comment inside `get_followers` function of file `relationship_tools.py`

Both functions `get_followers` and `get_following` needed to check whether the targeted Instagram account is private or not and, in case it is, whether we are already following

it or not. If the account is private and we are not following, the process of finding its followers/followings should stop and a relevant message should warn the user. If we do follow the account or the account is public, the process continues into extracting the data. That was the main idea of my implementation.

The first issue I had to deal with, while working on my implementation, was that I couldn't use an already existing function that checked whether user follows a private account or not. That was because importing that function from its location file `unfollow_util.py` inside `relationship_tools.py` was not allowed because functions of `relationship_tools.py` were already being imported and used inside `unfollow_util.py` and, when trying to import back from `unfollow_util.py` to `relationship_tools.py`, circular (recursive) imports were created which raised an error.

The only way to overcome that issue was to create a new file named `follow_util.py`, in which I moved the related function and then imported and used it inside both `relationship_tools.py` and `unfollow_util.py`.

Furthermore, besides adding a few lines of code to check private accounts and creating a new file as already mentioned, even more adjustments needed to be made as there were multiple functions affected by the addition of that necessary check. I added all extra parameters needed to the related functions, fixed all related imports and added some extra checks to all functions using the enhanced `get_followers` and `get_following` ones, as their actions should also be stopped in case of a private not following account.

In total, my second contribution affected 5 files, added 200 and deleted 116 lines of code, modified 12 functions and introduced a new file.

You can take a deeper look at my implementation in the following link:

<https://github.com/timgrossmann/InstaPy/pull/5470>

2.2.2 CODE QUALITY

For my second contribution, I used again the suggested Python formatter named Black as well as the library called pylint to check for possible linting errors. I reformatted my own code before making a pull request and I also made sure my code to be well documented according the project's principles.

2.2.3 CONTINUOUS INTEGRATION & TESTING

Testing this particular implementation was a difficult thing to do. Both functions `get_followers` and `get_following`, where I added the needed checks, have a very complicated functionality of several lines of code and use many more functions, so they could not be easily tested. In addition to that, in order to create any unit tests, I would have to parse personal data (a password included) and create an Instagram session which is not a good practice and, for those tests to pass all build tests, my credentials would have to be published on GitHub.

So, I asked for any advice about unit testing my code and I was suggested to test its functionality manually instead. For that, I created a test script to check all possible cases regarding all functions affected by my contribution.

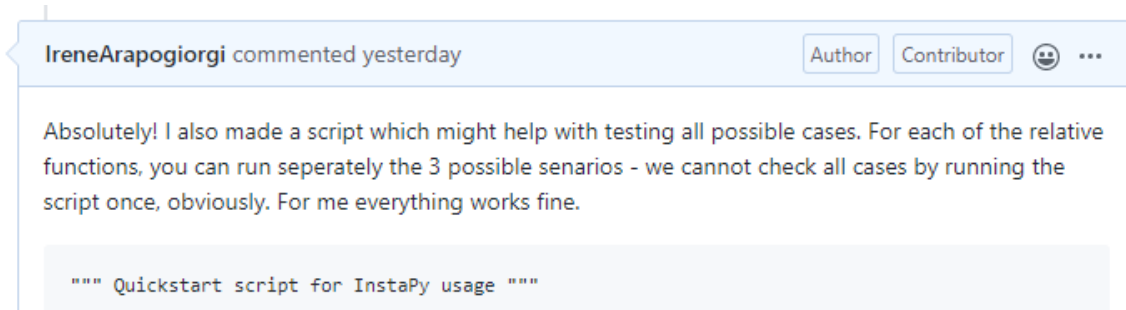


Figure 13: Test script for functionality check

Regarding Continuous Integration, InstaPy uses Travis CI, as we have already mentioned, so I was again thrilled to see that my second contribution passed all build tests and could possibly be integrated into the existing project.

2.2.4 PULL REQUEST – CODE REVIEW

My overall second contribution was specifically reviewed by one of the development team members who had also reviewed my previous one. After pointing out some issues needed to be addressed for my contribution to be correct and complete, he agreed that everything seems great and we only need to check the functionality of all affected functions before merging my pull request. As I'm writing this report, I'm waiting for him to run the test script I made and check the functions himself and, as my own tests have already passed and everything works fine, my second contribution to InstaPy is very close to be integrated to the rest of the project.

3. COMMUNICATION WITH THE TEAM

From the very beginning, I reached out to InstaPy's Discord channel where I had an excellent communication and cooperation with Felix Breuer (@breuerfelix on GitHub), who was the one that mainly reviewed all my pull requests. He was instantly responding and was very helpful giving me small tips, as well as solving any questions I had about the process of my very first contribution. In addition to that, Felix was always willing to give feedback for my work on GitHub and helped me improve my code. I was very lucky to have that excellent cooperation, which concluded in having a great overall experience while contributing to an open source project for the first time.

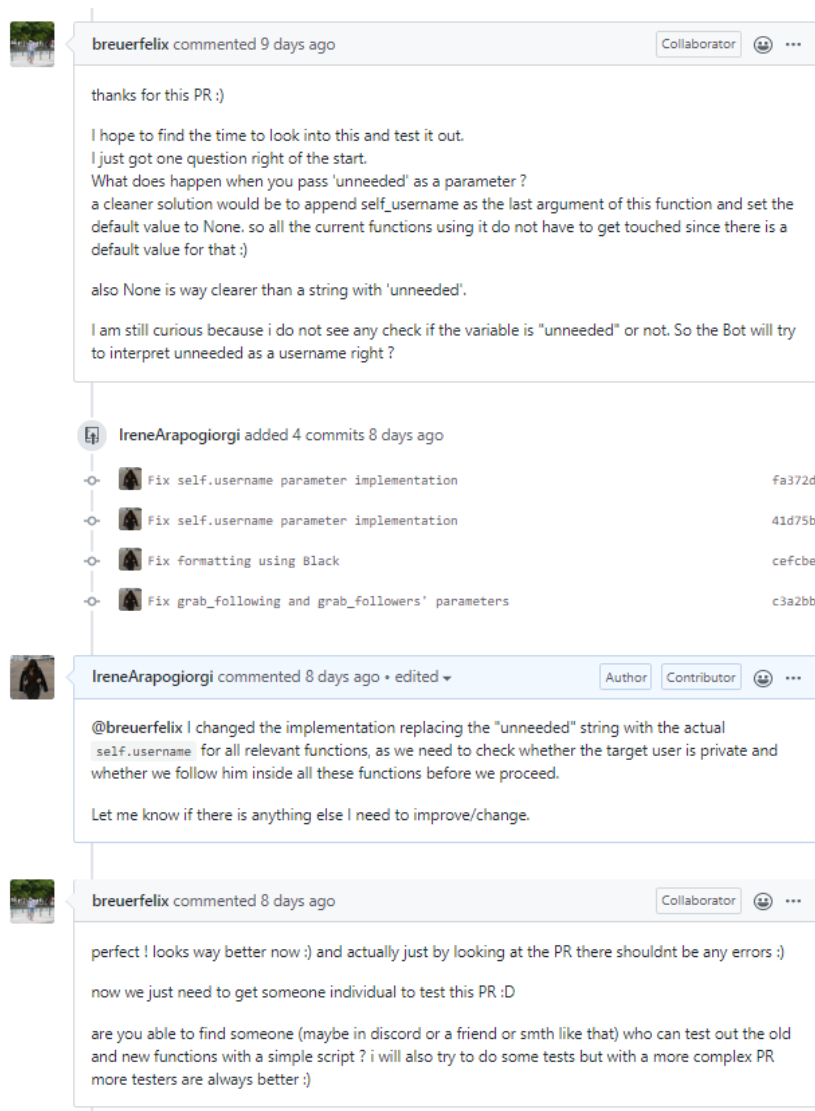


Figure 14: Discussing about my second contribution

4. SOURCE CODE MANAGEMENT

Using git and GitHub was my only hobby while working on my overall contribution to InstaPy. I had used both again in the past, but not as extensively as I needed to for my contribution. In the end, I've surely gained more knowledge on Version Control Systems and managed to comprehend and practice them even more.

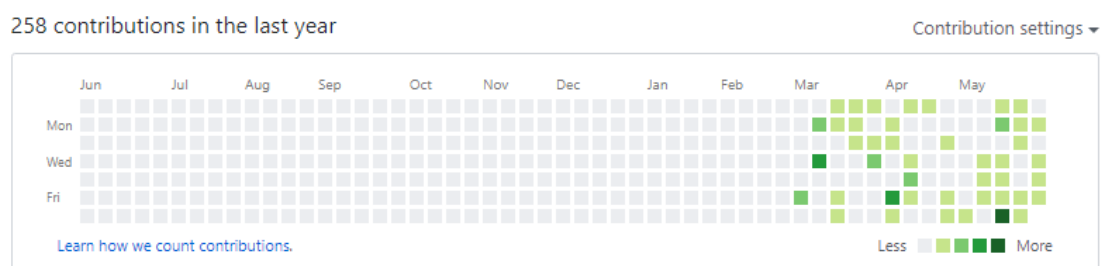




Figure 15: GitHub commits in the last year

Before starting both of my contributions, I opened relevant issues on GitHub in order to get assigned to work on my suggested enhancements and get some feedback.

Insert target users, hashtags etc from text file #5396


 Closed

IreneArapogiorgi opened this issue on 6 Apr · 4 comments



IreneArapogiorgi commented on 6 Apr

Contributor



...

Hello, I would like to contribute to InstaPy as part of a university course of mine, called Software Engineering in Practice, in which we are supposed to contribute to an open source project of our choice.

Expected Behavior

Amongst other feature ideas, I thought it would be useful for most users to have the ability to import a text file with their target users, hashtags etc. This implementation will especially benefit businesses using InstaPy as they could store users/hashtags etc in already-made target lists as text files and parse each one of them depending on their campaigns.

Current Behavior

For all existing functions accepting lists as given parameters, simple user is obliged to edit those lists inside the script, which complicates code comprehension & requires time.


Possible Solution

Implement the option of reading a text file for each function which accepts lists as given parameters.

I will be thrilled if you accept my upcoming contribution & I hope we'll have a great collaboration.

PS: Any advice is more than welcomed.

Assignees

 IreneArapogiorgi —unassign me

Labels

Feature

enhancement

in progress

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

Notifications

Customize




 Unsubscribe

Figure 16: Issue about my first contribution

Check if user's account is private #5451


 Open

IreneArapogiorgi opened this issue 19 days ago · 3 comments



IreneArapogiorgi commented 19 days ago · edited

Contributor



...

Hello, I would like to make another contribution to InstaPy as part of the same university course of mine, in which we are supposed to contribute to an open source project of our choice.

Expected Behavior

When using `get_followers()` and `get_followings()` we should check if user's account is private or not in order to proceed into finding his followers/followings.

Current Behavior


Both functions search for given user's followers and followings regardless of whether the account is private or not. So, in case of a private account, they return 0 followers/followings without informing the user that this happened because target account was private. Note that there are already two to-do comments related to this implementation inside these functions.

Possible Solution

Use `is_private_profile()` inside `get_followers()` and `get_followings()`.

PS: Any advice is more than welcomed.

Assignees

 IreneArapogiorgi —unassign me

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

Notifications

Customize


 Unsubscribe

Figure 17: Issue about my second contribution

Starting my contribution, I forked and cloned the InstaPy repository to my local machine. I also worked on different branches and not directly on the forked master branch and made sure my commits were concise and well documented with the proper commit message.

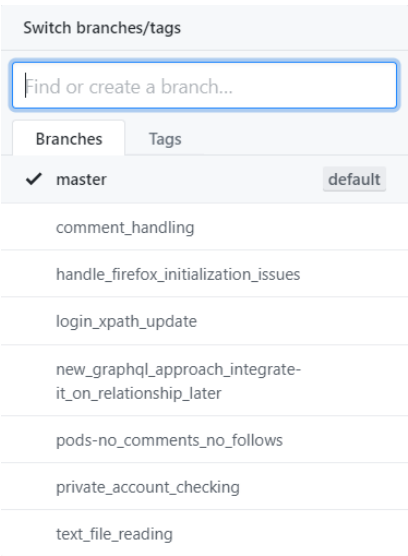


Figure 18: My personal branches `text_file_handling`, `comment_handling` & `private_account_checking`

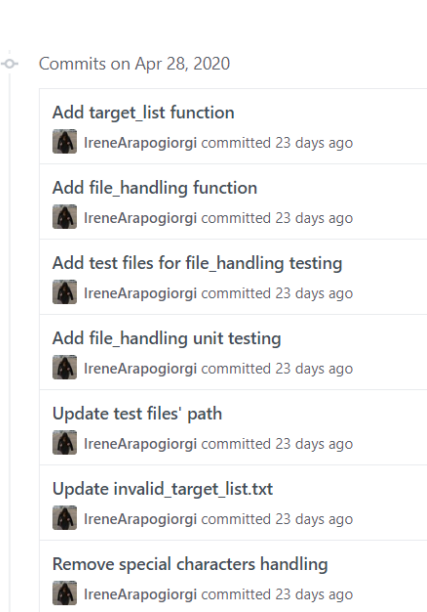


Figure 19: Example commits

5. PRESENTATIONS

You can find my PowerPoint presentations regarding the entire process of my first open source contribution journey on the following GitHub repository:

[IreneArapogiorgi/MyFirstOpenSourceContribution](#)

CONCLUSION

My first contribution to an open source project ended up being a journey that I will always remember and recall, as it taught me, first of all, that I should believe in myself and in what I'm able to achieve, regardless of how difficult anything may seem. Being able to observe the work of others and find little spots to fill in with my personal ideas was the most creative part during the entire process of contributing, while actual coding was the most fulfilling. During the past few months, I learned how to cooperate and exchange ideas with people around the world, I gained useful knowledge on actual software development and I faced the fear of many errors (literally coding errors) but also earned huge satisfaction when all my work finally paid off. The world of real software engineering was even more exciting than I had in mind and it certainly is a world I would like to contribute more to.

REFERENCES

- [1] <https://github.com/timgrossmann/InstaPy>
- [2] <https://realpython.com/instagram-bot-python-instapy/>