\*\*\_(this doc is under construction)\_\*\*

# How to contribute on NPKILL ?

I know that what I am going to say sounds like something typical, but I am sincerely glad that you are reading this, because that means that you are interested in helping to improve Npkill, \_or you may simply be here out of curiosity `cof cof`\_.

Anyway, you are sincerely welcome. I will try to explain the recommended guidelines to contribute.

---

# Common considerations

- Following this protocol helps to avoid working in vain. It would be a shame to dedicate hours to a pull request and have to reject it because there is already someone working on a similar issue.

-Unless they are minor and fast modifications, try to let everyone know that you are modifying something by opening an issue for example, or consulting the [projects](https://github.com/voidcosmos/npkill/projects)

- Change only the necessary lines for your modification. This will help to avoid conflicts, and in case of there being any, it will be easier to solve them.

- Make sure you to run `npm install`, because some development packages are meant to maintain harmony. Prettier, for example, makes sure that in each commit the files are well indented, and Commitlint makes sure that your messages follow the convention.

- Whenever possible, write tests, tests and more tests! tests tests tests tests tests tests tests tests tests tests tests

# New feature

1. If you want to contribute to a new feature, make sure that there isn't a previous issue of someone working on the same feature.

2. Then, open an issue explaining what you want to incorporate, and the files that you think you will need to modify a priori.

3. Wait for the community to give an opinion, and for some member to approve your proposal (a decision that will be taken into the community and future plans).

Yay! Green light to work!

4. Fork this project.

5. Create a new branch following the [recommended conventions]()

6. Write code and create commits regularly following the [recommended convention]()

7. Create a PULL REQUEST using \*\*DEVELOP as the base branch\*\*.

As a title, use the same (or similar) one you used in the creation of the issue, and in the description, any information that you consider relevant next to the link of the issue and "close" text (example: close #issueNumber) [more info](https://help.github.com/en/articles/closing-issues-using-keywords)

# Conventions

## git branch

I recommend using the following nomenclature whenever possible:

- feat/sort-results

- fix/lstat-crash

- docs/improve-readme

## git messages

Be sure to take your time thinking about the message for each commit.

All commits must use a convention similar to `Angular`. [Here all the rules](https://github.com/conventional-changelog/commitlint/tree/master/%40commitlint/config-conventional#type-enum)

- Use the present tense ("add feature" not "added feature")

- Use the imperative mood ("move cursor to..." not "moves cursor to...")

- Limit the first line to 72 characters or less

- Reference issues and pull requests liberally after the first line

\_[Some points extracted from Atom doc](https://github.com/atom/atom/blob/master/CONTRIBUTING.md#git-commit-messages)\_

## code

It is important to apply the principles of clean code.

If you use `VS Code`, there are some add-ons that I recommend:

-TSLint: Lets you know if you are breaking any of the \_coding rules\_ (do not use var, use const if possible, if some type has not been defined etc)

- CodeMetrics: Calculates the complexity of the methods, to ensure that your functions do only 1 thing. (green is ok, yellow is meh, red is oh god why)

If you use a different IDE, there are probably similar add-ons available.