Quickstart for contributors

Edit New Page

Kyle P. Johnson edited this page on 11 Oct · 10 revisions

New contributors are welcome to the CLTK. Please do not be shy about opening an issue with your project idea or emailing the administrator. For more advanced advice, see CLTK code style and workflow.

Learn Git basics

Git is an essential coordination tool the CLTK Organization uses in several places, namely for source code and also corpus version control. The GitHub website offers many features which makes using Git easier. The following tools and tutorials are useful if you're brand new to this technology.

- Hello World
- Good Resources for Learning Git and GitHub
- Github Desktop

Communicate

It is best practice to reach out the team before embarking on a fix, especially if what you plan on doing is not trivial. Those who have more acquaintance with the toolkit and its history might have helpful insights about how best to get the job done. Also, if your task may take a little while, you'll have "dibs" on it, so others will know to work on something else. Post on the issues tracker or reach out to the admin.

The lifecycle of an idea is usually along the following lines:

1. publicly propose them on the Issues page
2. a project maintainer asks questions, approves, rejects, or refers the topic to a subject matter expert
3. if an affirmative decision is reached, a work plan semi-formally agreed upon, including delivery date
4. a pull request is made by the contributor
5. the PR is checked on the build server; failed builds are always rejected
6. a project maintainer comments on the code, either rejecting, approving, or sending the PR back
7. a new tagged release is made and the code is available on PyPI

Tips

- Follow the steps in Example Git and Python workflow.
- If you do not have a preferred Python IDE, use PyCharm
- Check style. It is strongly advised to use the Pylint linter on anything that you write, to make sure that it is reasonably "Pythonic". There is no minimum score requirement, but we generally like to see code with a score of 5 and above.
- Write tests (`cltk/tests`). A minimal goal is to write at least one test for each new function.
- Write docs (`docs`). Note that the template syntax here is reStructuredText, not Markdown.
- Run all tests (with `$ nosetests`) to confirm you didn't break anything.

Clone this wiki locally

```
https://github.com/cltk/clt
```

Clone in Desktop