Radio Community Tools Tutorials Documentation

Release 0.1

The Radio Community

Oct 15, 2020

Contents

CHAPTER 1

Getting your environment set up

CHAPTER 2

Using community tools

CHAPTER 3

How to contribute to tools and tutorials

3.1 Contributing code through the browser (easy)

radioscripts_contrib is a collection of user-created radio astronomy python scripts. This code is offered with minimal vetting process and without any guarantee of reliability or accuracy. We encourage interested visitors to contribute by adding their own useful python routines. The radioscripts_contrib collection operates out of a GitHub repository. Below is a tutorial of a very simple browser based method to contribute your own code to the repository. If you are interested in a slightly more advanced, command-line based method, which will generally be more convenient for contributing multiple files and editing as you test them on your own system, you may want to check out the other tutorial on contributing code *via command line*

1. Get a GitHub account

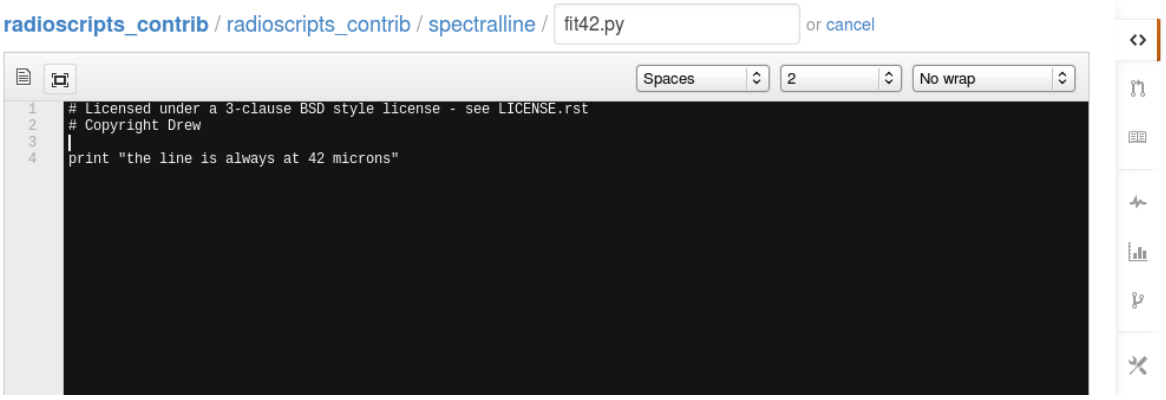
   If you don't already have a GitHub account, the first step is to go to https://github.com and set one up. GitHub also describes how to install a command line interface on your machine to interact with the online repository, but for our purposes you don't need to worry about this. All you need is the GitHub account and a web browser.

2. Create your own personal "fork" and add a file to it.

   Within radioscripts_contrib scripts are organized into subfolders by topic. If there is a subfolder that describes the topic your script fits into, navigate to that location (for instance, if you wrote a routine to fit spectral lines, this might belong under the spectralline path.) If no folder adequately describes the topic of your script, just use the main radioscripts_contrib directory. From the appropriate directory screen on GitHub, click on the icon of a document containing a plus sign.

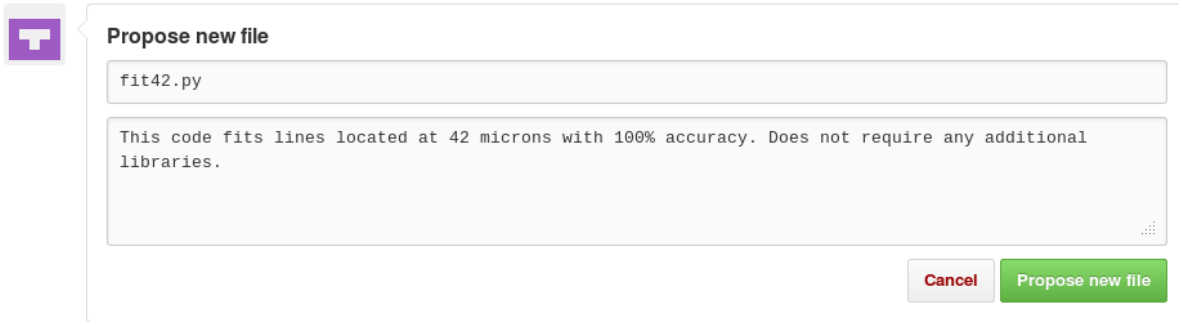
   This simultaneously creates a "fork" (a remote copy) and brings you to an editor where you can type your code (or paste it from a file you already have) and name the file.



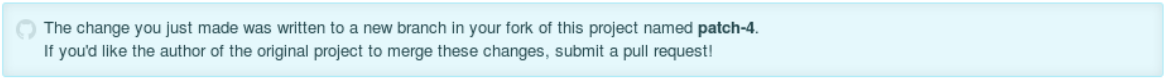
We request that contributed code be prefaced with a two line license statement at the top:

```
# Licensed under a 3-clause BSD style license - see LICENSE.rst
# Copyright [authorname]
```

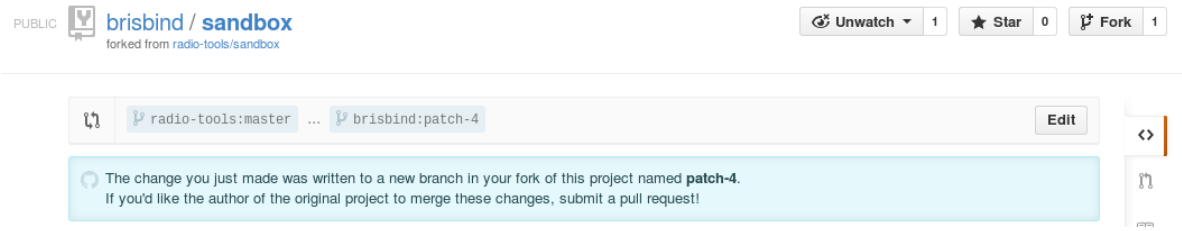
Further down the page you can describe your file or put any extra comments that aren't part of the code and then click "propose new file"



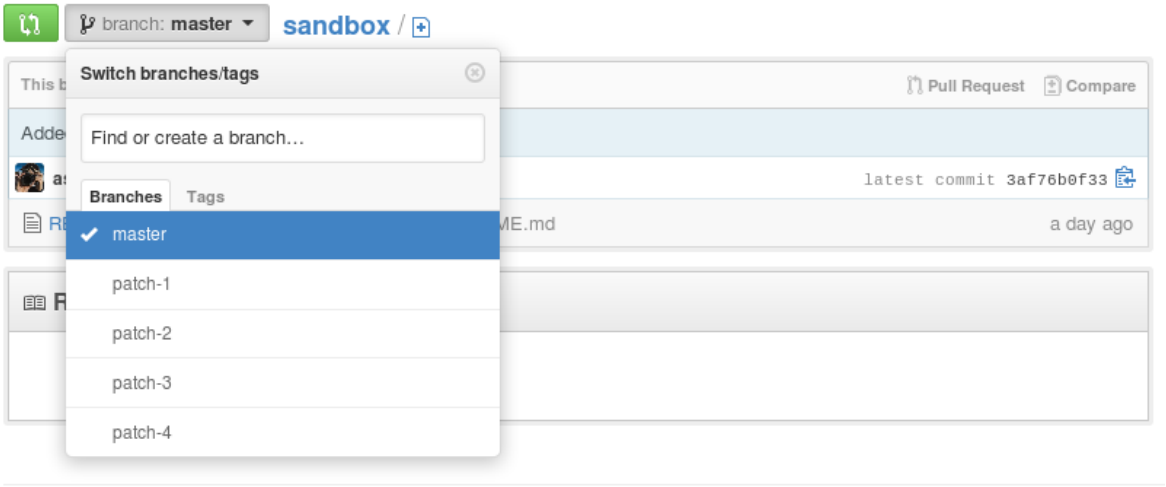
GitHub should now tell you the name of the fork where your file has been placed. In this case it's patch-4.



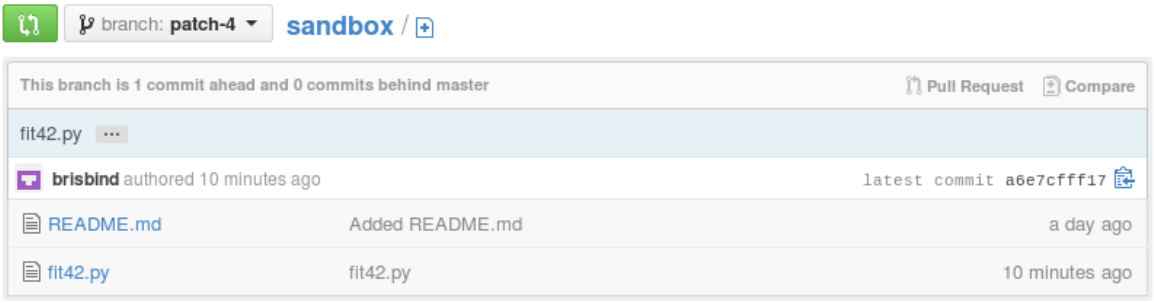
Before trying to merge your code (which currently exists in your own fork) in with the main repository, take a moment to update the README in the folder where your code is to give a one or two line description of your new code. Near the top of the current page click the name of the repository where you just put your code. In this example I'm working in a repository named "sandbox" but in your case it will probably be radioscripts_contrib or one of its subfolders.



This brings you to a page showing the contents of this repository. Click the button that says "branch:master" and a drop down list will appear with all the different versions of the repository you have access to. from the drop down list select the name of the fork where your file was put (patch-4 in my case).



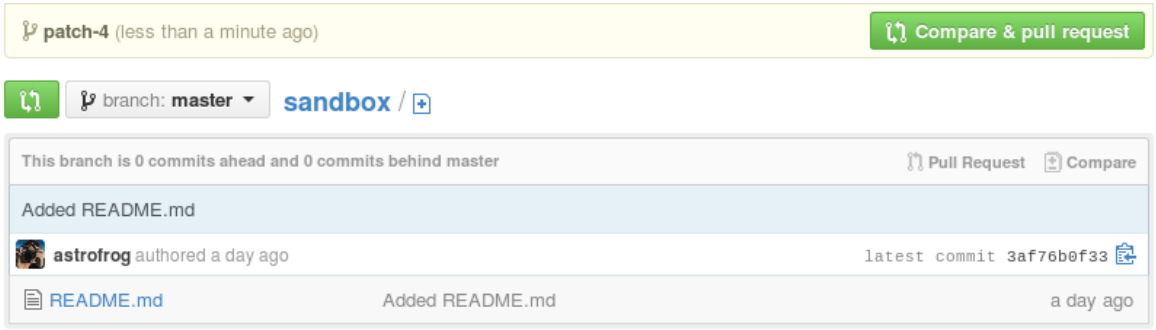
You should now see all the files (including README and your new file) that exist in your newly created fork.



Click the README file and click the "Edit" button.

In the editor window with the README file, insert the name of your code and a line or two describing it.

Once more click the name of the subfolder where you put your code. Next to the repository with your code (patch-4 here,) click the green button , "Compare & pull request"



Describe your updates and hit "Send pull request"

And you're done!

3.2 Contributing code with git (intermediate)

radioscripts_contrib is a collection of user-created radio astronomy python scripts. This code is offered with minimal vetting process and without any guarantee of reliability or accuracy. We encourage interested visitors to contribute by adding their own useful python routines. The radioscripts_contrib collection operates out of a GitHub repository. Below is a very quick and very dirty tutorial on how to contribute your own code to the repository using the command line Git interface — very convenient for sharing multiple files and editing them while simultaneously testing them on your own computer. This will not make you a Git master, but it will let you get the job done. If you're interested in a more in depth look at coordinating public contributed code in a nice controlled manner you may want to check out the Astropy tutorial on contributing code: http://docs.astropy.org/en/latest/development/workflow/development_workflow.html

If you don't like working with a command line and simply want the quickest way to add a single file to the repository, you can check out the *tutorial on contributing code via the GitHub browser interface*

Briefly, here are the steps we'll be going through (if these don't make sense right now, don't worry we've got you covered below!): install and configure Git; create a personal fork of the radioscripts_contrib repository; clone your fork onto your machine; add your code as a new file in the appropriate directory; add, commit, and push your changes back onto your online repository; and finally do a pull request to ask moderators to merge the changes in your fork into the main version.

1. Get Git

   If you already know your way around Git you can skip down to step 2.

   a. If you don't already have a GitHub account, the first step is to go to https://github.com and set one up.

   b. Next you need to make sure Git is installed on your computer. If you're working on a linux machine there's a good chance you already have it. To check, open a terminal and type:

   ```
   git --version
   ```

   If git is installed you should see a message telling you what version you have. Something along these lines:

   ```
   git version 1.7.1
   ```

   If it's not installed head over to http://git-scm.com/downloads and download the version appropriate for your OS.

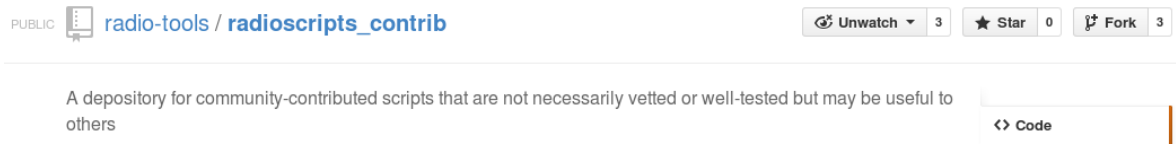
   c. You need to configure Git on your machine to let it know who you are. You can read through the Git guide on doing this here: https://help.github.com/articles/set-up-git#set-up-git But in the end this only requires two simple lines of typing in terminal:

   ```
   git config --global user.name "Your Name Here"
   git config --global user.email "your_email@example.com"
   ```

   The Git guide also describes how to set up password caching (and there's the alternative of using SSH keys instead.) These steps are *generally* only useful to save yourself time later from typing in a password when intereacting with your online Git repository. Don't worry about that unless you want to. *However,* if you have an old version of git (this definitely applies to version 1.7.1 and probably any version <1.7.10) there are a few extra steps involved. Head over to https://help.github.com/articles/generating-sshkeys and follow along with their walkthrough and explanation of setting up an SSH key.

2. Set up your personal version of radioscripts_contrib

   a. You need to create a "fork" (remote copy) of the radioscripts_contrib repository to have a personal version you can muck around with. From the online Git repository (https://github.com/radio-tools/radioscripts_contrib) click the button labelled "Fork" located near the upper right hand corner.



b. You now need to create a "clone" (local copy) of the radioscripts_contrib repository. Navigate to a path in terminal where you want to work and type:

```
git clone https://github.com/your-user-name/radioscripts_contrib.git
```

c. This will create a subdirectory named radioscripts_contrib/ which contains the full repository. Within radioscripts_contrib/radioscripts_contrib/ scripts are organized into subfolders by topic. If there is a subfolder that describes the topic your script fits into, copy your code into that path (for instance, if you wrote a routine to fit spectral lines, this might belong under the spectralline path.) If no folder adequately describes the topic of your script, just copy it into the radioscripts_contrib/radioscripts_contrib/ path. We request that contributed code be prefaced with a two line license statement at the top:

```
# Licensed under a 3-clause BSD style license - see LICENSE.rst
# Copyright [authorname]
```

We also request that you add a one or two line description of your code in the README file in the subdirectory where you place your code.

Note that if you're just trying this tutorial out to get the hang of it, there's a file named helloworld.py which you should feel free to edit.

3. Tell the online GitHub about your edited version of the repository

   At this point, if you're using Git version < 1.7.10 there's one more step to make things run properly. Head here to check it out.

   a. At this point on your machine you should have a version of the radioscripts_contrib repository that is nearly identical to the one you started with, except with one or two new or edited files (the README file and your new script.)

   If you ask Git about your files by typing:

```
git status
```

   Git will point out these new and edited files to you. You should get some lines of informational text along with the notice that you have "Untracked" files. These are the files which are newly updated since the last time you told Git about the important files you want to add. Along with the one or two files you actually care about there might be some junk files you don't care about (autosaved files ending in a ~ for instance.) It will look something like this:

```
# On branch master
# Your branch is ahead of 'origin/master' by 1 commit.
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
# helloworld.py
#       README.md
# helloworld.py~
nothing added to commit but untracked files present (use "git add" to track)
```

   Stage the important files for uploading by typing:

```
git add your-first-file.name
git add your-second-file.name
```

Ask git about the status again:

```
git status
```

And you should now see your important files listed under "# Changes to be committed"

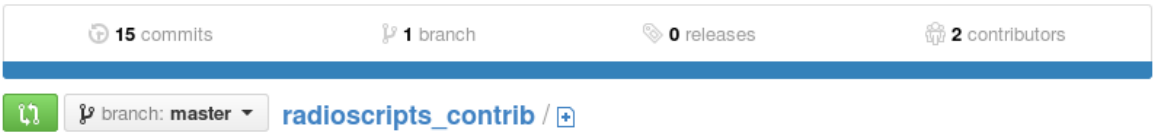
b. You are now ready to commit these changes. As you do this include a brief message saying what changes you've made in your Git repository:

```
git commit -m "Added my python script to fit spectral lines and updated README"
```

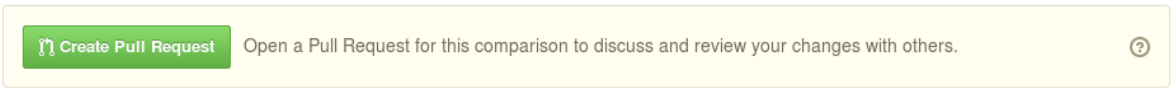
c. Now you need to push this version of the repository back online:

```
git push
```

d. Almost done, now you just need to request to get your contributions merged into the main radioscripts_contrib repository by performing a "pull request". To do this, go back to your online forked version of the radioscripts_contrib repository. If you want to check, you can browse to the appropriate path and you should now see your newly added file(s). On the left side of the page, just above the list of files there is a green button with two arrows.



Click it to go to a page that will summarize your changes and ask for a title. If all looks good then click the green button on that page and your pull request will be processed



As long as everything looks good in your code, your pull request will be accepted and the code will be merged into the main repository. If it turns out there *is* something that needs to be changed, you'll receive an email with comments from the moderators asking for changes. Once you make those changes in your code on your local directory, just add it, stage it, and commit it again:

```
git add your-first-file.name
git commit -m "Made the changes to the whatsit you requested"
git push
```

You're done!