Quick reference guide to `Git`

Castillo, M. Ezequiel

This mini-tutorial will help you on basic `Git` commands for working with remotes. This mini-guide can, in principle be updated periodically as I find more useful commands.

1. Working with *your* Remotes

1.1. Cloning your Remotes

First, `cd` to your project directory. Then type the following:

```
git clone https://github.com/ezitoc/MySimpleScripts.git
```

This will create a new folder at the current working directory called `MySimpleScripts`. This is similar to `checkout` command from Subversion. Now if you run:

```
git remote -v
```

this will show:

```
origin  https://github.com/ezitoc/MySimpleScripts.git
```

which is the URL that Git has stored for the shortname to be expanded to:

1.2. Adding Remote Repositories

Let's say you want to add a Remote but this time with a different shortname, for example, `ezitoc` (be aware, you are creating a new project, if you just want to rename, there is a special command for this). You should type:

```
git remote add ezitoc https://github.com/ezitoc/MySimpleScripts.
   git
```

Now, by running:

```
git remote -v
```

you will get:

```
ezitoc  https://github.com/ezitoc/MySimpleScripts.git
```

1.3. Renaming and removing a Remote

For renaming you will have to enter the following:

```
git remote rename origin ezitoc
```

For removing just type:

```
git remote rm ezitoc
```

1.4. Fetching and pulling

Some definitions:

- **Fetch:** pulls the data to your local repository but it doesn't automatically merge it with any of your work or modify what you're currently working on.
- **Pull:** if you have a branch set up, automatically fetch and merge a remote branch into your current branch.

To get data from your remote projects, you can run:

```
git fetch ezitoc
```

To pull data from your remote projects, you can run:

```
git pull ezitoc
```

1.5. Creating new files

In order to start tracking new files you can write:

```
git add filename
```

Once you have modified `filename` you will need to stag it, before you push changes.

1.6. Staging files

To change a file that was already tracked you have to use the multi-purpose command `add` to stag files.

```
git add filename
```

1.7. Commiting

Once you have set up your stagging area, you will need to run:

```
git commit
```

and you will be redirected to your editor. Before exiting you must specify a commit message. You can also set the `-v` flag to which the diff will be included in your commit message. Besides this, you can pass the inline flag `-m` and a message between quotes to be included in the commit message, which will result in no redirection to your editor.

Once you have commit your changes, you will need push the changes to the server repository.

1.8. Pushing

When you want to share your work or back up, you will have to run:

```
git push ezitoc master
```

where `master` is your master-branch (which is set by default when using the `clone` command). This command works if you cloned from a server to which you have write access and nobody have push in the meantime.

2. Reverting changes

2.1. Revert a commit

Supose you commit files you didn't intended to commit. For example say you ran the following command:

```
git commit calcE.py -m 'Update␣calcE'
```

Then you must find the commit name with:

```
git log
```

We will find something like:

```
commit 3fee482e587ed65acbafb46adfa953dbcd9a6e61
Author: M. Ezequiel Castillo <ezecastillo@gmail.com>
Date:   Wed Apr 24 20:09:03 2013 -0300

    Update calcE.py

commit e36950110dda5c906cf36f54626b739783db00ae
Author: M. Ezequiel Castillo <ezecastillo@gmail.com>
Date:   Wed Apr 24 20:07:07 2013 -0300

    Revert ``Update?''

origin  https://github.com/ezitoc/MySimpleScripts.git
```

Then we can revert the commit "*Update calcE.py*" with

```
git revert 3fee482e587ed65acbafb46adfa953dbcd9a6e61
```