Contribute to the Open Source Movement

Open source provides a framework for building and maintaining systems that are transparent, community-driven, and resilient against central points of failure or control. This framework empowers users and developers, ensuring that the control and evolution of digital tools are distributed. Open source stands as a fundamental pillar in the pursuit of a more decentralised and participatory digital future.

This workshop will guide you through the fundamentals of creating and participating to open source projects.

Autonomy

While this sheet provides good general guidance on how to successfully complete the objectives of this workshop, it is not an exhaustive step by step recipe. If you find yourself lacking informations to move forward, you will find the answers online.

Being an Open Source contributor requires autonomy so this workshop purposefully asks you to read some documentation, be creative and take ownership of your work.

A. Creating an Open Source project

1. The project

You must choose a project from the following list. Each project should be a console project that is run in the terminal. Feel free to innovate in your implementation of each project.

The project list:

1. Game of life

The famous game by mathematician John Horton Conway. To make it console based, you can use spaces for dead cells and # for live cells.

2. Text-based Adventure Game

A text based adventure where the player's decisions define the outcome of the story.

3. Todo list application

A console based application to help keep track of tasks. You should be able to add, delete, complete tasks.

4. Quiz game

The player answers a number of questions to try and win the game.

5. ASCII Art Generator

Provide simple text to the program and watch it output beautiful ASCII art that represent your text. For example:

`Open` can become,

```
. /$$$$$$
 /$$__  $$
| $$  \ $$  /$$$$$$   /$$$$$$  /$$$$$$$
| $$  | $$ /$$__  $$ /$$__  $$| $$__  $$
| $$  | $$| $$  \ $$| $$$$$$$$| $$  \ $$
| $$  | $$| $$  | $$| $$_____/| $$  | $$
|  $$$$$$/| $$$$$$$/|  $$$$$$$| $$  | $$
 \______/ | $$____/  \_______/|__/  |__/
          | $$
          | $$
          |__/
```

(Optional)

Have a cool idea ? Develop it !

If you have an idea for a project of a similar size as the ones described above and have a peer in your TD who is willing to contribute to the project, you can go with this option.

2. Create a public Github repository

2.1 Git

Git is an open source version control system to facilitate collaboration. It allows multiple people to work on the same project simultaneously by tracking changes and coordinating work among multiple contributors.

The majority of open source projects use Git.

2.2 Github

Github is a platform used to host and manage Git repositories.

It is one of the most common and easy to use solution but it is not the only one. Alternatives like GitLab, Bitbucket, SourceForge, Gitea and Phabricator exist.

Github, being owned by Microsoft is not ideal when we analyse it through the lens of decentralisation. The centralisation of power on the platform, the data privacy and the potential for corporate influence go against the core ethos of the Open Source movement. Although this issues are present, a lot of open source projects actively use the Github platform.

This workshop is using Github as a learning tool to easily discover how to run and contribute to open source projects, what you will learn here can very easily be transferred to other platforms.

2.3 Create a Github account

If you don't already have one, follow this quick guide.

2.4 Create a repository for your project

Create a Github repository with the following properties:

- **Name:** {name_of_your_project}-{your_name}-{your_td_number}
- **Description:** {description_of_your_project}
- **Public:** yes
- **Initialise a README file:** yes
- **Add .gitignore**: no
- **License:** MIT License

You can follow this guide for more precisions.

3. Clone the repository on your machine

Congratulations you are officially the owner of an open source project ! It's now time to kickstart your adventure by being it's first contributor.

Cloning your project is the action of downloading your project's files with Git. Once downloaded on your machine you'll be able to start developing.

You can follow this guide to clone your newly created repository on your computer.

4. Implement the project

4.1 Choose a programming language between the ones below:

- Python
- Javascript (NodeJS)

4.2 Setup the license

In the root of your project, you should create a setup file which will hold the project's name, version, license and other project informations.

4.2.1 If you chose Python:

Create a `setup.py` file in the root of your project and add the following content.

```
#!/usr/bin/env python

from distutils.core import setup

setup(
  name='{name-of-your-project}',
  version='1.0',
  author='{your-full-name-or-your-github-username}',
  license='MIT',
  long_description=open('README.md').read(),
)
```

4.2.2 If you chose Javascript:

(If you don't have npm installed, follow this guide)

Run `npm init` and provide the following informations:

- **package name:** {the_name_of_your_project}
- **version:** 1.0.0
- **description:** a short description of your project
- **entry point:** index.js
- **test command:** *blank*
- **git repository:** the link of the public git repository of the project

- **key words:** *blank*
- **author:** your full name or your github username
- **license:** MIT

When asked, type `yes`

4.3 Implement

It's now time to implement the first version of your project by following the specifications given in section A.1.

The project should be functionnal and the setup should allow an other developer to run the project on his machine in less than 5 minutes.

4.4 Push your implementation to Github

Every time you make a new feature, fix a few bugs or make any substantial changes to your code base, you should commit your changes and push your commit to Github. This will make your changes visible to other contributors.

Follow this guide to learn how to commit and push your changes.

> 💡 Git commits should always contain a message, this message should describe what the commit's changes are doing to the project. For more informations, see this guide.

4.5 Document your project

You could think that your project is ready to receive contributions but we're lacking something important.

If a developer wishes to contribute to your project, we need to make his life as easy as possible by creating a documentation for the project.

The documentation should at least contain the following informations:

- A description of the project
- How to run the project
- A guide on how to contribute

This documentation should be located in a README.md file at the root of your project.

md is short for markdown, an easy to use markdown language. You can learn the basics here.

You can use this template to get started with your documentation.

Don't forget to commit and push your documentation changes !

B. Receive contributions

The goal is for you to create a community of developers who will help you develop your project. For now a community of 2 people is sufficient (you and another developer) but if you can get more people to contribute it's even better !

1. Suggest improvements to your project

Open source contributions may come from users of the software who have a very clear feature in mind, but it's also common for a developer to wish to contribute without any clear idea of what they want to bring to the project. Basically, they want to help in any way they can.

A good way to ease the contribution on your project is, yourself, to suggest potential improvements you may need help with.

After all, you have created the first implementation so you must have countless improvement ideas.

The best way to suggest improvements is to create a Github issue. Issues are little threads which allow you to discuss any topics about your project.

Write 3 Github issues to suggest improvements to your project.

Follow this guide to learn how to create a Github issue.

Your issues should be clear to understand and well documented. If a developer asks questions in your issues' thread, you should answer their question so keep an eye on your project's notifications.

2. Receive an issue

If a developer finds an opportunity to work on, he will first create an issue. This issue will be in the form of an explanation of what he wishes to contribute. You should answer these issues by giving your opinion on the proposed improvement.

If the work doesn't align with your project, as the project's owner, you can refuse the contribution.

If the work would be a good feature for your project, you should let the person know and ask him to suggest a Github pull request.

3. Receive Pull Requests

After some time, you will receive your first contribution.

Amazing 🎉

This will be in the form of a pull request. A pull request is a proposal to merge changes from one branch into another. In our case it's more specifically, a proposal to merge a contribution to the project. Pull requests should mention and be linked to their associated Github issue to give contest on what feature or bug fix is being proposed.

When receiving a pull request, you can access it in the `Pull Requests` tab of your Github repository page. When you click on a pull request, it allows you to review the commits associated with the pull request to make sure the changes correctly implement the said features and to make sure no bugs are introduced in the project.

If you find any shortcomings to the contribution, you should create an answer in the pull request's thread to notify the contributor of your desired modifications or improvements. You will then see the updated work in the same pull request once the contributor will have pushed his work.

Finally, when you are happy with the contribution you received, you can merge the pull request.

Your project just got a new update !

C. Contributing to Open Source

Open source would be nothing without contributions, it's now time for you to create your own !

1. Find and analyse a project

Find a project from your class that has no contributions and analyse its code, documentation and open issues. (you can also give the project a Github star !)

2. Suggest a contribution

From your code analysis and from the open issues on the Github repository, you should contribute in three ways.

(1) Improve documentation

Maintainers don't always do the best job when it comes to documenting a project. It's difficult to put yourself in the shoes of someone who doesn't know the project so documentation often lack some key informations to more easily understand the inner workings of the project.

Since you just came up to the project with a fresh mind, it was probably easy for you to spot some opportunities to improve the documentation !

Improving documentation can either be improving the README.md file or adding documentation directly to the code itself.

Find some improvements and create a Github issue that describes clearly what you wish to improve.

(2) Fix a bug

It is said that a developer creates, on average, 70 bugs per 1000 lines of code. There is very probably a bug in the project you chose. It's not always easy to find them but with your outside perspective you should be able to find at least one.

Create an issue that describes in details the bug(s) you found.

(3) Improve the project

Every project has room for new features. Use your imagination to find an interesting improvement of the project.

Clearly describe this new feature in an issue.

3. Get feedback from the owner of the repository

After submitting your issues, you will get answers from the project's owner (or potentially from other contributors / users of the project). Discuss the issue until the project's owner gives you the approval to submit a pull request.

4. Implement

You should fork the repository to your personal Github account by following this guide.

Then you should clone the repository on your computer by following this guide.

In Git you work on a branch. The `main` branch is, as its name implies, the principal branch where the latest stable version of the software lies.

When contributing, you should not push your changes to the main branch. This branch is usually protected and only the repository's owner can push changes to it.

The preferred way to push code to a project is to create a personal branch in your forked repository. You should create one branch for each contribution you wish to make and you should name the branch in a short and descriptive way. Follow this guide for more informations.

Now that your branch is created you should commit and push the changes to your branch. Each commit should clearly describe the changes it brings to the project.

5. Propose a Pull Request

Once your contribution is finished and pushed on the Github branch you created, you should create a pull request. To do so, go to your forked repository, click on the `Pull Request` tab and click on `New Pull Request`.

Then you should open a pull request where the base repository is the original repository you wish to contribute to, the base branch should be main, the head repository should be your forked repository and the compare branch should be the branch where you made your contribution. Once this is setup right, click on `Create Pull Request`.

To learn more about pull requests, see this guide.

Don't forget to follow the status of your pull request, the owner of the open source project you are contributing to might have some comment or request changes to your contribution. If the pull request is not merged yet, you can make new commits to your local branch, push the changes and they will be automatically added to the pull request.

(5. Bonus)

The best time to contribute to an open source project was yesterday, the second best time is right now !

If you contribute to a real open source project with a legit contribution history and at least 20 github stars, you will be awarded bonus points. The contribution can be as small as finding a typo or improving documentation.

- Opening an issue and getting an answer: 2 bonus points
- Creating a pull request: 2 bonus points
- Getting a pull request merged: 3 bonus points

- Project has more than 1k Github stars: 5 bonus points

Warning: Do not spam any Open Source project, make sure your contributions add value to the project you choose. A good rule to follow is to alredy be a user of the projects you are contributing to or to know the project very well before creating an issue or suggesting a pull request.

Grade

After finishing your project, complete the form in *De Vinci Learning* with proofs of completion of each step. **You should submit the form before the start of the next TD.**

Here is the grading system used for this workshop:

A. Creating an Open Source project

Objective	Points
Creating an open source repository with the right license and setup	1
Implementing a working project	2
Documenting the project so it can be understood easily and be run in ~5 minutes	2
Creating at least 3 issues for your repository	1

B. Receive contributions

Objective	Points
Answering to a contributor in an issue to let him know you'd like a contribution	1
Merging a pull request for an improvement of the documentation	1
Merging a pull request for a bug fix	1
Merging a pull request for an improvement of the project	2
(Bonus) After finding issues in the proposed modifications of a pull request, you asked the contributor to make changes before merging	1
(Bonus) After receiving an issue which was not relevant, you informed the contributor and closed the issue	1
(Bonus) Receive contributions from more than 1 developer	1

C. Contributing to Open Source

Objective	Points
Getting a pull request merged to improve the documentation of a project	1
Getting a pull request merged to fix a bug of a project	2
Getting a pull request merged to improve a project	3
All proposed pull requests explain clearly what changes are being made	1
All proposed pull requests are made on a different branch	1
All proposed pull requests are linked to a Github issue	1
Contributing to a well known open source project	Up to 12