|  |  |
| --- | --- |
| **Unit 1:** | INTRODUCING FOSS (5 hours) |
| *Introductory text explaining what the unit is about. This should include a description of the content type, the major lesson consents and associated quiz.* [approximately 600 characters] | |
| To be added | |

|  |  |
| --- | --- |
| Lesson Title: 1.1 Defining the concept of "Free and Open Source Software" (FOSS) and the difference with proprietary software (1,3 hours) | |
| Lesson Introduction:  The European Commission has launched the Digital Agenda for Europe, which proposes to better exploit the potential of ICT -including FOSS- in order to foster innovation, economic growth and progress, pointing out the need to promote digital skills of European workforce. The use of FOSS will bring several benefits to European SMEs and will particularly improve their digital performances and competitiveness.  In this lesson we look at the definition of FOSS, its practical implications and the main differences with proprietary software.  **In this lesson we will explore the following:**   * What is the concept behind the definition of FOSS? What are the guiding principles? * What are the key players involved in this field at national and transnational level? * What are the differences between FOSS and proprietary software? | |
| Image 1 | **1.1. Part 1 - The concept behind the definition - A historical background:**  Free Software is 'free' as in freedom and not as in 'free of charge'. There is a variety of terms that are used to refer to “**Free Software**”, but let us see first how it all began.  The **Free Software movement** was started by Richard Stallman in 1983 when he inaugurated [the GNU project](https://www.gnu.org/gnu/about-gnu.html), as an operating system which would be put together by people working together for the freedom of all software users. This was based on the idea that people should have full control over the software they own. Stallman recognised that in order to offer a 100% free software, access to a computer program's source code was a fundamental requirement. The struggle for the survival of Free Software has just began. Source: <https://fsfe.org/freesoftware/basics/summary.en.html>  Historically, Free Software was the first term. The first documented complete definition on “F**ree Software**” appears to be the GNU's Bulletin, vol. 1 no. 1, published February 1986. In particular, the four freedoms that define Free Software are 1. The freedom to run the program, for any purpose; 2. The freedom to study how the program works and adapt it to your needs; 3. The freedom to redistribute copies so you can help your neighbour and 4. The freedom to improve the program, and release your improvements to the public, so that the whole community benefits. This specific term is today officially preferred by the [Free Software Foundation Europe](https://fsfe.org/) (FSFE) as well as by the [GNU Operating System (GNU](https://www.gnu.org/gnu/gnu.en.html)).  Source: <https://www.gnu.org/philosophy/free-sw.html> and <https://www.gnu.org/bulletins/bull1.txt>  In 1997 Debian, a project aiming to create a completely free and community based GNU/Linux distribution, developed the **Debian Free Software Guidelines** ([DFSG](http://www.debian.org/social_contract#guidelines)) as a check-list whether a program can be included in the distribution or not.  Source: <https://fsfe.org/freesoftware/basics/comparison.html>  One year later, on 03.02.1998, in the wake of Netscapes announcement to release their browser as Free Software, a group of people met in Palo Alto in the Silicon Valley and proposed to start a [marketing campaign](http://web.archive.org/web/20021217003716/http:/www.opensource.org/advocacy/faq.html) for Free Software using the term “**Open Source**”.' The goal was to seek fast commercialisation of Free Software and acceptance of Free Software by the companies and venture capitalists of the booming new economy. The group decided to leave aside all long-term issues related to Free Software, feeling these posed obstacles in its acceptance. They proposed to focus on technical advantages only. Often used in good faith, the term "**Open Source Software**" - originally defined to mean the same thing as “Free Software” in terms of licenses and implementation - has seen inflationary usage.  Source: <https://fsfe.org/about/basics/freesoftware.en.html#terminology>  As a result, the term "Open Source Software" was created in 1998 as a term to differentiate "Free and Open Source Software" from software that is free of charge, by a coalition of programmers who eventually became the **Open Source Initiative (OSI)**. The OSI is today a non-profit corporation with global scope formed to educate about and advocate for the benefits of open source and to build bridges among different constituencies in the open source community. (source: Producing Open Source Software: How to Run a Successful Free Software Project, Karl Fogel, © 2005-2017).  Finally, a third term evoke when the European Commission started dealing with Free Software on a regular basis. The Commission sought to avoid the ambiguity of the English word "Free Software" and the misunderstandings of "Open Source" alike. This led to the adoption of a third term which has popped up occasionally since around 1992, "**Libre Software**." This term has proven resistant to inflationary usage and is still used in an identical way to Free Software.  Source: <https://fsfe.org/about/basics/freesoftware.en.html#terminology>  More information on the different terms at: <https://www.gnu.org/philosophy/open-source-misses-the-point.html> |
| Image 2 | **1.1. Part 2 - The guiding principles/ the four essential freedoms:**  Speaking about Free Software is speaking about freedom. More precisely the four freedoms of free software are:   1. the unlimited use of software for any purpose; 2. the right to study how the program works and understand it. Access to the source code is a precondition for this; 3. the right to share copies of the software and hereby to help other people; 4. the freedom to improve the program and to distribute the improvements to the program, so that everybody profits from it. Access to the source code is a precondition for this.   Like when someone gives away a recipe or when someone even improves it, with free software, people can use or improve the software on their own. Users should not think that since they don’t have the knowledge of programming, these freedoms are of little use for them. They might not be able to do it on their own, but these freedoms give the possibility to those who can program, to solve the problem for all the people. Without the access to the source code however, this would remain impossible.   1. The freedom to run the program means the freedom for any kind of person or organization to use it on any kind of computer system, for any kind of overall job and purpose, without being required to communicate about it with the developer or any other specific entity. it is the user's purpose that matters, not the developer's purpose. 2. The freedom to study the source code and make changes includes the freedom to use your changed version in place of the original. If the program is delivered in a product designed to run someone else's modified versions but refuse to run yours — a practice known as “tivoization”, “lockdown” or “secure boot” — this freedom 1 becomes an empty pretense. These binaries are not free software even if the source code they are compiled from is free. Further, one important way to modify a program is by merging in available free subroutines and modules. If the program's license says that you cannot merge in a suitably licensed existing module — for instance, if it requires you to be the copyright holder of any code you add — then the license is too restrictive to qualify as free. 3. The freedom to redistribute if you wish means you are free to redistribute copies, either with or without modifications, either gratis or charging a fee for distribution, to anyone anywhere, without having to ask or pay for permission to do so. If you do publish your modifications, you should not be required to notify anyone in particular, or in any particular way. 4. The freedom to distribute copies of your modified versions to others includes the freedom to release your modified versions as free software. A free license may also permit other ways of releasing them; in other words, it does not have to be a copyleft license. However, a license that requires modified versions to be nonfree does not qualify as a free license.   Source of the 4 explanations: <https://www.gnu.org/philosophy/free-sw.html>  Copyleft is a general method for making a program (or other work) free (in the sense of freedom, not “zero price”), and requiring all modified and extended versions of the program to be free as well. More: <https://www.gnu.org/copyleft/copyleft.html> |
| No Image | **1.1. Part 3 - The definitions:**  There are two political camps in the free software community, the free software movement and open source. The **free software movement** is a campaign for computer users' freedom, where nonfree programs are considered as an injustice to the users. On the other side, the **open source camp** declines to see the issue as a matter of justice to the users and bases its arguments on practical benefits only.  We saw in a previous lesson how the confusion between the definition of “Free Software” and “Open Source” started when the first term was referred to as "**Open Source**" by the Open Source Initiative (OSI). At the moment, there are companies that prefer and use the one or the other term, which have practically no differences and describe the same set of software.  Another term evoked when a researcher studying practices and methods used by developers in the free software community decided that these questions were independent of the developers' political views and so he used the term “**FLOSS**”. FLOSS was meaning “Free/Libre and Open Source Software” and the idea was to explicitly avoid a preference between the two political camps described above and stay neutral.  The next term of “**FOSS**” stands for “Free and Open Source Software”. FOSS is meant to mean the same thing as “FLOSS,” but for some people it seems less clear. They claim that it fails to explain that “free” refers to freedom, because it presents “open source” prominently and it names free and open source as a single point of view, rather than mentioning the fact that free software and open source are two different political positions that disagree fundamentally. Source: <https://www.gnu.org/philosophy/floss-and-foss.html>  Other people also use terms like "**organic software**" or "**ethical software**", trying to stay out of the terminology debate. These terms tend to cause confusion, because they virtually invite people to look for differences between the terms where actually no differences exist.  Source: <https://fsfe.org/freesoftware/basics/comparison.html>  Further information can be found at:  <https://opensource.com/business/14/11/business-foss-really-community-software>  And  <https://opensource.com/article/17/11/open-source-or-free-software> |
| Image 3 | **1.1. Part 4 - The key players involved in this field at national and transnational level:**  **1. The Free Software Foundation:**  The Free Software Foundation (FSF) is a nonprofit with a worldwide mission to promote computer user freedom by promoting the development and use of free (as in freedom) software and documentation—particularly the [GNU operating system](https://gnu.org)—and by [campaigning](https://www.fsf.org/campaigns) against threats to computer user freedom like Digital Restrictions Management (DRM) and software patents. The FSF also provides important [resources](https://www.fsf.org/resources/) to the community including the FSF/UNESCO [free software directory.](https://directory.fsf.org/)  Source: <https://directory.fsf.org/wiki/Main_Page>  2. The **Open Source Initiative (OSI):**  The OSI is a non-profit corporation with global scope formed to educate about and advocate for the benefits of open source and to build bridges among different constituencies in the open source community. It also approves the licencing of Open Source Software.  See further: [www.opensource.org](http://www.opensource.org)  **3. Libre Office – The Document Foundation:**  LibreOffice is community-driven and developed software, and a project of the not-for-profit organization, “The Document Foundation”. LibreOffice is free and open source software, originally based on OpenOffice.org (commonly known as OpenOffice), and is the most actively developed OpenOffice.org successor project.  More at: <https://www.libreoffice.org/about-us/who-are-we/>  **4. FSFE:**  Free Software Foundation Europe is a charity that empowers users to control technology. It helps individuals and organisations to understand how [Free Software](https://fsfe.org/about/basics/freesoftware.en.html) contributes to freedom, transparency, and self-determination and enhances users' rights by abolishing barriers to Free Software adoption. It also encourages people to use and develop Free Software and provides resources to enable everyone to further promote Free Software in Europe.  More at: <https://fsfe.org/about/about.en.html>  **5. The European Commission:**  The EC has put efforts in the promotion of open source. In December 2000 it defined a strategy on the internal use of FOSS, which has been continuously updated until the current Action Plan 2014-2017. The Plan proposes to continue adopt formally the use of FOSS technologies & products. The EU is focused on promoting open source within Member States public institutions. The EUPL is the first FOSS license created on the initiative of the EC, available in 22 European languages, open to use by anyone for software distribution. Also, the EC launched the Open Source Observatory.  A list of Organizations that work for freedom in computer development and electronic communications can be found at <https://www.gnu.org/links/links.html#FreedomOrganizations> |
| …. | **1.1. Part 5 - The differences between FOSS and proprietary software:**  1. On first hearing the term "free software" many people mistakenly think it means "zero-cost software". FOSS software is often free of charge, but its main difference to proprietary software is that it is free as in "freedom" — that is,  the freedom to use, study, share and modify a software under the terms of its license. In the case of proprietary software, users are not allowed to copy neither to distribute the software, as this forms an illegal action.  2. At “nonfree” or “proprietary” programs users don't control the program, but the program controls the users, which is controlled by the developer, making it an instrument of unjust and manipulative power. There might be functionalities designed to mistreat, spy on, censor or restrict users. The opposite applies for FOSS, where users control the program, both individually and collectively.  3. Proprietary or closed-source programs are the opposite of "free" or "open source" because they are distributed under traditional, royalty-based licensing terms, where users pay per copy, or under any other terms sufficiently restrictive to prevent open source dynamics from operating. By FOSS the software can be used without paying a license fee, and anyone can modify the software.  4. The source code for proprietary commercial software is usually a closely guarded secret, whereas by FOSS it is freely available.  5. Proprietary software has no security at all in one crucial case — against its developer. As a result, private companies like Apple or Microsoft are enabled to break into government’s computers that use such software and in that way endanger national security.  Sources: Producing Open Source Software: How to Run a Successful Free Software Project, Karl Fogel, © 2005-2017, p.195. and  <https://www.gnu.org/philosophy/free-sw.html>  and  <https://www.greennet.org.uk/support/open-source-vs-proprietary-software>  and  <https://www.gnu.org/philosophy/free-software-even-more-important.en.html> |
| **Assessment text**  **Are you ready to say what FOSS is?**  Assessment quizzes to be added | |
| **Summary text**  FOSS is based on the four freedoms to use, study, share and change/improve a software in any way, as the source code is open. The main difference between proprietary software and FOSS is that with proprietary users are not allowed to copy neither to distribute the software. | |

|  |  |
| --- | --- |
| Lesson Title: 1.2 FOSS facts and myths related to FOSS theory and applications and the different types of freedom (1,3 hours) | |
| Lesson Introduction:  There are many myths about FOSS being not secure or coming with no technical assistance. These misleading perceptions have been holding users away from FOSS tools for a long time now. In this lesson we discern between facts and myths.  **In this lesson we will explore the following:**   * What are the myths related to FOSS and how can we tell them apart from facts? * Which are the different types of ‘freedom’ implied in the FOSS concept? * Which are the advantages and disadvantages of FOSS tools when related to the common bias and misleading perceptions? | |
| Image 1 | **1.2. Part 1 - The myths and the facts:**  1. **FOSS is less secure since the source code of a software is publicly available**: A code that is openly accessible can be checked by independent parties for security loopholes or installed backdoors. The publication of code can therefore be regarded as a trust-gaining measure. Apart from that, a license has no impact on the security of a software product.  2. **There is no professional technical support available with FOSS**: Many FOSS companies have specialised in support services for clients and users that wish to book professional support packages.  3. **Compatibility of FOSS with other/proprietary software is low**: FOSS can be complementary to other/proprietary software, as it can be modified to run both well on Windows, Apple or Linux. Prominent examples for such tools that are compatible with various systems are the Firefox Browser, Libre Office or the VLC Media Player.  4. **FOSS is not very reliable, because it is always developed by amateurs**: It is clearly a prejudice that only hobbyists contribute code, since many developers are highly paid IT professionals. Large companies invest millions of Euros in FOSS development and code improvement. It is estimated that around 90% of the contributions to Linux come from professional developers, who are employed by different companies.  **5. FOSS is in general less user friendly:** Its long ago since FOSS lacked a proper user interface. For example, the most popular operating system for smartphones (Android) is basically FOSS, the most modern TV screens are operated by FOSS, Wikipedia is fully based on FOSS and websites like WordPress, Drupal or Typo3 are FOSS as well.  6. **FOSS is only a trend, it is not sustainable, and you can’t rely on it for the future**: FOSS is actually a long-term success story. Almost all software prior to the 1980s was Free Software and the first explicit Free Software license was published in the 1980s. Since then, the number of individuals, businesses and institutions using and contributing to FOSS is constantly growing. There are recommendations both on EU and national level, as the European Commission along with many governments encourage public administrations to use FOSS or even implement laws that shall guarantee that FOSS projects funded with public money shall result in public code.  Source: Unpublished text from FSFE  7. **FOSS means “non-commercial”**: A free program must be available for commercial use, development, and distribution. Regardless of how users get their copies (paid or free), they always have the freedom to copy, change the software, or even to sell copies. If a program fails to allow commercial use and commercial distribution, it is not Free Software.  Source: <https://fsfe.org/about/basics/freesoftware.en.html#terminology>  And <https://www.gnu.org/philosophy/free-sw.html>  **Other myths are:**   * Set-up time for FOSS is high / no source found. * If it's open, that means anyone can change our code. * Open source is cheaper. * Tech companies are not able to earn money with FOSS. (For example, Red Hat, one of the biggest players in the market for Free Software services, achieved an annual turnover of 2,9 billion Euro in 2017).   Source: Producing Open Source Software: How to Run a Successful Free Software Project, Karl Fogel, © 2005-2017, p.105-107. |
|  | **Some facts concerning FOSS development**  1. Free software is a culture by choice. To operate successfully in it, you have to understand why people choose to be in it in the first place. Coercive techniques don't work. If people are unhappy in one project, they will just wander off to another one.  2. Free software is remarkable even among intentional communities for its lightness of investment. Many of the people involved have never actually met the other participants face-to-face. The normal conduits by which humans bond with each other and form lasting groups are narrowed down to a tiny channel: the written word, carried over electronic wires.  3. Because of this, it can take a long time for a cohesive and dedicated group to form. Conversely, it's quite easy for a project to lose a potential participant in the first five minutes of acquaintanceship. If a project doesn't make a good first impression, newcomers may wait a long time before giving it a second chance.  Source: Producing Open Source Software: How to Run a Successful Free Software Project, Karl Fogel, © 2005-2017, p.10. |
| Image 2 | **1.2. Part 2 - The different types of ‘freedom’ implied in the FOSS concept:**  Is this maybe the same with the prepared text on chapter “1.1. Part 2”? I cant tell the difference here. Sources are needed. |
| No Image | **1.2. Part 3 - The advantages of FOSS tools:**  Any piece of free software that respects users' freedom has a strong inherent advantage over a proprietary competitor that does not. Even if it has other issues, free software always has freedom. The main benefits of using FOSS are: development of digital skills, improved software quality, the ability to customise software for own usage, no vendor-lock in meaning no vendor dependency (including vendor, product, and cloud lock-in), fair competition in the market, long-term sustainability of software, decreasing software costs, increasing security and stability (especially in regard to malware), protecting privacy and giving companies and users more control over their own hardware devices, own IT systems and associated digital artefacts.  Source: <https://www.gnu.org/philosophy/when-free-software-isnt-practically-superior.html>  Video on advantages for SMEs:  <https://www.youtube.com/watch?v=ZtYJoatnHb8> |
| Image 3 | **1.2. Part 4 - The disadvantages of FOSS tools:**  Although we do not often advertise the fact, any user of an early-stage free software project can explain that free software is not always as convenient, in purely practical terms, as its proprietary competitors. Free software is sometimes low quality. It is sometimes unreliable. It is sometimes inflexible. If people take the arguments in favor of open source seriously, they must explain why open source has not lived up to its “promise” and conclude that proprietary tools would be a better choice. There is no reason we should have to do either.  Richard Stallman speaks to this in his article on [Why Open Source Misses the Point](https://www.gnu.org/philosophy/open-source-misses-the-point.html) when he explains, “The idea of open source is that allowing users to change and redistribute the software will make it more powerful and reliable. But this is not guaranteed. Developers of proprietary software are not necessarily incompetent. Sometimes they produce a program that is powerful and reliable, even though it does not respect the users' freedom.”  Secondly, the vast majority of free development projects are at no technical advantage with respect to a proprietary competitor.  Source: <https://www.gnu.org/philosophy/when-free-software-isnt-practically-superior.html> |
| Assessment text  **Are you ready to discern between facts and myths?**  Assessment quizzes to be added | |
| Summary text | |

|  |  |
| --- | --- |
| Lesson Title: 1.3 Rights, licensing and responsibilities | |
| Lesson Introduction:  When we use, study, share or change a software, it is important to be aware of its licence. Licences are affecting our lives giving us not only rights on a software, but also responsibilities. In this lesson we will understand the main implications on licencing and will give them practical application.  **In this lesson we will explore the following:**   * What are the issues related to rights? * What are the issues related to licensing? How to choose a licence? * What are the issues related to responsibilities? | |
| No Image | **1.3. Part 1 - The issues related to rights:**  The main difference between proprietary software and free and open source software is **the granting of rights** to modify and re-use a software product obtained by a customer. FOSS software licenses both rights to the customer and therefore bundles the modifiable source code with the software ("open-source"), while proprietary software typically does not license these rights and therefore keeps the source code hidden ("closed source").  In addition to granting rights and imposing restrictions on the use of copyrighted software, software licenses typically contain provisions which allocate **liability and responsibility between the parties** entering into the license agreement. In enterprise and commercial software transactions, these terms often include limitations of liability, warranties and warranty disclaimers, and indemnity if the software infringes intellectual property rights of anyone.  Unlicensed software outside the scope of copyright protection is either **public domain software** (PD) or software which is non-distributed, non-licensed and handled as **internal business trade secret**. *(Public domain is explained as having no copyright holder, meaning that there is no one who has the right to restrict copying of the work. When a work is in the public domain, material from it can be incorporated into a copyrighted work, and the derivative is thus under the same overall copyright as the original copyrighted work. Source: Producing Open Source Software: How to Run a Successful Free Software Project, Karl Fogel, © 2005-2017, p.195.)* Contrary to popular belief, distributed unlicensed software (not in the public domain) is fully copyright protected, and therefore legally unusable (as no usage rights at all are granted by a license) until it passes into public domain after the copyright term has expired. Examples of this are unauthorized software leaks or software projects which are placed on public software repositories like GitHub without a specified license. As voluntarily handing software into the public domain (before reaching the copyright term) is problematic in some international law domains (for instance the Law of Germany), there are also licenses granting PD-like rights, for instance the [CC0](http://www.creativecommons.org) or WTFPL.  Source: <https://en.wikipedia.org/wiki/Software_license> |
|  | **1.3. Part 1b - The issues related to rights: Ownership and licencing**  Concerning the ownership rights of a user, it is important to keep in mind that a copyright holder cannot oppose the resale of a digitally sold software. According to the rule of copyright exhaustion on first sale, ownership is transferred and users can sale such software. After the fight of a European company for this in court, the EU Directive 2009/24/EC expressly **permits trading used computer programs**.  This is good to know, since at proprietary software licenses the software publisher grants the use of one or more copies of software under the EULA ([end-user license agreement](https://en.wikipedia.org/wiki/End-user_license_agreement)), but ownership of those copies **remains with the software publisher**. This means that certain rights regarding the software are reserved by the software publisher, for instance the number of installations allowed or the terms of distribution. As a result, the end-user has no alternative but to accept the software license, since without it s/he may not be able to use the software at all, like for instance with the license for Microsoft Windows. Αctivities that are usually restricted are reverse engineering, simultaneous use of the software by multiple users, and publication of benchmarks or performance tests.  Source: <https://en.wikipedia.org/wiki/Software_license> |
| Image 1 | **1.3. Part 2 – Licensing issues:**  A [**software license**](https://en.wikipedia.org/wiki/Software_license) is a legal instrument (usually by way of contract law, with or without printed material) governing the use or redistribution of software. To make a new developed software free, it is necessary to release it under a free software license. There are three widely recognized entities in the Free Software movement that regularly evaluate free and open-source software licenses: The [Free Software Foundation](http://www.fsf.org), the [Debian project](http://www.debian.org) and the [Open Source Initiative](http://www.opensource.org). To decide whether a specific software license qualifies as a free software license, licensing experts in the staff and the board of directors of these entities review a licence and its conditions. They judge it based on [specific criteria](https://www.gnu.org/philosophy/applying-free-sw-criteria.html) (which are frequently updated) to determine whether it fits their spirit as well as the precise words. If a license includes unconscionable restrictions, it is rejected. Source: <https://www.gnu.org/philosophy/free-sw.html>  The goal is to see whether a particular license gives software users the freedom to use, study, share and improve a program. For instance, a free license may not require compliance with the license of a nonfree program. Further, it is acceptable for a free license to specify which jurisdiction's law applies, or where litigation must be done, or both.  Most free software licenses are based on copyright, and there are limits on what kinds of requirements can be imposed through copyright. However, some free software licenses are based on contracts, and contracts can impose a much larger range of possible unacceptable restrictions making the software nonfree.  **Free and Open Source Software licenses** are divided into two main categories: **protective share-alike or Copyleft** licenses and **non-protective** or **permissive or non-copyleft** licenses. Copyleft licenses are designed to protect the rights to use, study, share and improve the software. Furthermore, they require that those freedoms apply to any derivative works. An example of a copyleft free software license is the often-used GNU General Public License (GPL), also the first copyleft license. Non-protective licenses allow to distribute the software without those rights, like for instance with proprietary software where the developers use copyright to take away the users' freedom. Permissive licences have minimal requirements about how the software can be redistributed and do not have a clause requiring  that they apply to derivative works as well. Examples of permissive free software licenses are the [BSD license](https://opensource.org/licenses/BSD-2-Clause) and the [MIT license](https://opensource.org/licenses/MIT), which give unlimited permission to use, study, and privately modify the software, and include minimal requirements on redistribution.  Source: <https://fsfe.org/freesoftware/basics/comparison.html>  And  <https://en.wikipedia.org/wiki/Software_license>  A list of available free and non-free licences can be found at:  <https://www.gnu.org/licenses/license-list.html>  and  <https://spdx.org/licenses/> |
| Image 2 | **1.3. Part 3 - How to choose a licence:**  When choosing a license to apply to a project or a work that someone creates -whether that's a modification of an existing work, or a new original work-, the user should keep the following in mind:  - It is better to use an **existing license** instead of making up a new one, - it is better to use one of the **widely-used**, well-recognized existing licenses, (Source: Producing Open Source Software: How to Run a Successful Free Software Project, Karl Fogel, © 2005-2017, p.24 and 198.)  - According to the recommendations of the Free Software Foundation's Licensing and Compliance Lab, when someone contributes to an existing project, s/he should usually release his/her modified versions **under the same license as the original work,**  - Users can use a different license, when they make major changes to a work under a non-copyleft license and if their newly created version is considerably more useful than the original,  - Concerning software, the experts suggest using different licenses for different projects, depending mostly on the software's purpose,  - For tutorials, reference manuals and other large works of documentation, it is recommended to use the [**GNU Free Documentation License (GFDL)**](https://www.gnu.org/licenses/fdl.html). Source: <https://www.gnu.org/licenses/license-recommendations.html>  The most commonly used licenses:  - If you want a non-copyleft license and you're comfortable with your project's code potentially being used in proprietary programs, then use an **MIT-style license**. It is the simplest of several minimal licenses that do little more than assert nominal copyright (without actually restricting copying) and specify that the code comes with no warranty. - If you want a copyleft license and you don't want your code to be used in proprietary programs, use the **GNU General Public License, version 3** (https://www.gnu.org/licenses/gpl.html). The GPL is probably the most widely recognized free software license in the world today. Source: Producing Open Source Software: How to Run a Successful Free Software Project, Karl Fogel, © 2005-2017, p.24, 196, 198.  Worth mentioning here is the project of FSFE “**REUSE**” on how to add copyright and license information to a project in ways which allow for more automation. The project provides users with easy help on how to release some software under a FOSS license and make it easily reusable. Available at: <https://reuse.software/about/> |
|  | **Good to know: "free software license" and "open source license"**  The terms "**free software license**" and "**open source license**" are essentially synonymous. Technically, the former term refers to licenses confirmed by the Free Software Foundation as offering the "four freedoms" necessary for free software (see https://www.gnu.org/philosophy/freesw.html), while the latter term refers to licenses approved by the Open Source Initiative as meeting the Open Source Definition (https://opensource.org/osd). However, if someone reads the FSF's definition of free software, and the OSI's definition of open source software, it becomes obvious that the two definitions delineate the same freedoms. The inevitable, and in some sense deliberate, result is that the two organizations have approved the same set of licenses.  Source: Producing Open Source Software: How to Run a Successful Free Software Project, Karl Fogel, © 2005-2017, p.24 and 196. |
| Image 3 | **1.3. Part 4 - The issues related to responsibilities:**  What should be written here? |
| Assessment text  **Are you ready to say how licencing applies at FOSS products?**  Assessment quizzes to be added | |
| **Summary text**  FOSS licences grant users the rights to modify and re-use a software product they obtained. Free and Open Source Software licenses are divided into two main categories: protective share-alike or Copyleft licenses and non-protective or permissive or non-copyleft licenses.  The most commonly used licenses are the MIT-style license or the GNU General Public License, version 3. | |

|  |  |
| --- | --- |
| Lesson Title: 1.4 FOSS communities and their ways of collaboration | |
| Lesson Introduction:  Introduction to …  In this lesson we will explore the following:   * How can someone search and find different existing FOSS communities? * Which are the main existing FOSS communities in national or international level? * How do the members of FOSS communities communicate and collaborate at local, national and transnational level? | |
| Image 1 | **1.4. Part 1 - Ways to search and find different existing FOSS communities:**  Communities are diverse and highly motivated. It is easy to find someone to either get or give help to no matter what your interests are. Communities are supported by initiatives (f.e. the OSI), foundations (f.e. the FSF) or associations (like the Drupal Association) to contribute to a common effort, to maintain communication with each other and to efficiently organise themselves. As private companies (like Intel, IBM or Samsung) use open source code to build their own commercial products and services, they also see a strategic value in contributing back to those projects and support such communities as well.  Looking in the pages of the Free Software Foundation and the GNU Operating System, we see Richard Stallman explaining in his article that today there is a large community of users who run GNU, Linux and other free software and there are even more people who would like to extend this work, and who have adopted the goal of convincing more computer users to “use free software”. Since 2005, over 13,500 developers from over 1,300 different companies have contributed only to the Linux kernel project.  Source: <https://www.gnu.org/philosophy/use-free-software.en.html>  And <https://www.linuxfoundation.org/resources/open-source-guides/participating-open-source-communities/>  Looking across Social Media pages like Facebook, Twitter, Google+, LinkedIn or Reddit there are numerous official and unofficial accounts that share each community's news and views. This is an easy way to find all relevant international or national communities of Wordpress, Joomla, NextCloud, OwnCloud, Drupal, Mozilla or the Software Libre Society. |
| Image 2 | **1.4. Part 2 - The main FOSS communities in national and international level:**   * <https://directory.fsf.org/wiki/Free_Software_Directory:Participate/Project_Team> * <https://nextcloud.com/> * ...   Beside individuals there are also many well known organisations in the Free Software ecosystem. Many of them play an important role and emphasize different aspects of Free Software. For example, some organisations focus on the technical direction of Free Software projects, some on legal aspects, some on political, social and ethical aspects and some focus on license evaluation. These organisations typically have decided to use one or another term and stick to it. But this should not lead to the conclusion that the term they use is the critical factor regarding their motivations. Source: <https://fsfe.org/freesoftware/basics/comparison.html>  The [Drupal community](https://www.drupal.org/community) is one of the largest open source communities in the world. We're more than 1,000,000 passionate developers, designers, trainers, strategists, coordinators, editors, and sponsors working together. We build Drupal, provide support, create documentation, share networking opportunities, and more. Our shared commitment to the open source spirit pushes the Drupal project forward.  The [Open Source Maintainers](https://github.com/maintainers) on GitHub is an official community that started in the beginning of the year 2017 and continued its journey towards 2018. It brings in maintainers from all over the GitHub community to discuss and talk about anything and have a 600+ member community. |
| No Image | **1.4. Part 3 - The ways in which members of FOSS communities communicate and collaborate at local, national and transnational level:**  **Live communication:**  The IRC Live Chat is commonly used for user support and general discussion. For example, the members of the Free Software Foundation can join each other each Friday 12:00-15:00 EDT, in the IRC Live Chat channel <irc://chat.freenode.net/fsf>, for their scheduled meetings. In the [Free Software Directory blog](https://www.fsf.org/blogs/directory) users can find more information, like the meeting topics. Similarly, on [**IRC at freenode.net**](http://webchat.freenode.net/), the #osi channel is the live chat of the Open Source Initiative.  Source: <https://directory.fsf.org/wiki/Main_Page>  **Mailing lists:**  Mailing lists is a usual way to communicate with subscribed members. At the Open Source Initiative for instance, they have many [**global and regional mailing lists**](https://opensource.org/lists), for announcements, discussion, licensing and other topics and one can read their mailing lists on the web via [**Mailman**](https://lists.opensource.org/cgi-bin/mailman/listinfo).  **Working groups:**  Many open source projects, like Drupal, have created Community Working Groups with the mission to uphold each community’s Code of Conduct in order to maintain a friendly and welcoming community for members and newcomers.  **Forums:**  Forums offer communities a huge knowledge bank that supports both developers and users. It is another way for them to contribute and help others.  **Live events, Meetups and Conferences:**  Community members can meet each other at community **events,** where they can meet face to face, swap tips, and get inspiration for their ongoing and next projects, making friends along the way. |
| Assessment text  **Are you ready to say how communities work with each other?**  Assessment quizzes follow | |
| Summary text | |