History of Open Source Software

- In the 1950s and 1960s, computer operating software and compilers were delivered as a part of hardware purchases without separate fees.

- In the early 1970s AT&T distributed early versions of Unix at no cost to government and academic researchers, but these versions did not come with permission to redistribute or to distribute modified versions and were thus not free software in the modern meaning of the phrase. After Unix became more widespread in the early 1980s, AT&T stopped the free distribution and charged for system patches. Most researchers paid for a commercial license because it is quite difficult to switch to another architecture.

- After Unix became more widespread in the early 1980s, AT&T stopped the free distribution and charged for system patches. Most researchers paid for a commercial license because it is quite difficult to switch to another architecture.

- Software was not considered copyrightable before the 1974 US Commission on New Technological Uses of Copyrighted Works (CONTU) decided that "computer programs, to the extent that they embody an author's original creation, are proper subject matter of copyright".

- Therefore, the software had no licenses attached and was shared as public-domain software, typically with source code.

- To increase revenues, a general trend began to no longer distribute source code (easily readable by programmers), and only distribute the executable machine code that was compiled from the source code.
- One person especially distressed by this new practice was Richard Stallman.
- He was concerned that he could no longer study or further modify programs initially written by others. Stallman viewed this practice as ethically wrong.

- In response, he founded the GNU Project in 1983 so that people could use computers using only free software.

- He established a non-profit organization, the Free Software Foundation, in 1985, to more formally organize the project.

- He invented copyleft, a legal mechanism to preserve the "free" status of a work subject to copyright, and implemented this in the GNU General Public License.

- Proprietary means ownership. If anyone wants to use the proprietary software, then a license has to be purchased from the owner for using the software.
- The source code of this kind of software is not visible to anyone except the owner. The owner can be the developer or the organization releasing the software.

- When you pay the fee to obtain the license, you don't have any access and rights to view, copy, distribute or modify the source code of the proprietary software.
- In other words, you simply pay to get a "license to use as per the guidelines given by the owner" and nothing else. As a buyer, you can use it. And the owner can officially distribute the software to any number of buyers. If the ownership is transferred to someone else, then that one gains the authority to view, copy, distribute, modify the proprietary software

Open source or Proprietary software, Which One is Better?

- Open source software may seem to be the best choice since it's usually free and the source code is accessible without any restrictions.
- But that software is not customized or may not be in perfect form for the users. So the community around the world collaborates to keep improving the source code by removing errors, adding features to make the software more efficient and convenient.

Advantages of Proprietary software :

- Stability. It will work perfectly as you expected while purchasing the software.
- Reliable software and, customer service is easier to access.
- Highly user-friendly and the software is customized to specific purposes.
- Unique. Packed with multiple features.

Disadvantages of Proprietary software :

- Even when you purchase the license, you have several restrictions on using the software and you can't modify it.
- The usage of the software may be restricted like 'can only be installed on one computer'. So depending on the agreement, the installation of software on different workstations varies.
- If it has any missing features or bugs, then you have to wait until they are resolved by the owner.

Pragmatism vs Idealism

- Pragmatism is a philosophical approach that evaluates theories or beliefs in terms of the success of their practical application.
- Idealism, refers to any philosophy that states that reality is mentally constructed.

Pragmatism vs Idealism

- The **key difference** between pragmatism and idealism is that**:**

Pragmatism considers the practical consequences of an action as its main component whereas idealism considers mental entities or thoughts and ideas as its main component.

Pragmatism

- According to pragmatists, most philosophical topics such as nature of knowledge, concepts, science, beliefs, and language can be viewed in terms of their practical applications.
- Pragmatism emphasizes on this practical application of thoughts by acting on them to test them in human experiments.

- Charles Sanders Peirce is considered to be the founder of this tradition. William James, George Hubert Mead and John Dewey are also considered as its major proponents.
- For pragmatists, thought is a guide to prediction, problem-solving and action. The practical consequences of an action or thought are the main components of pragmatism.

- **Major Components:**

Pragmatism considers practical consequences of an action as its main component.

Idealism considers mental entities or thoughts and ideas as its main component.

- **Thought:**

Pragmatism considers thought as a guide to prediction, problem-solving and action.

Idealism considers thoughts and ideas as the only real entities.

Example

- When you're idealistic, you dream of perfection, whether in yourself or other people. For example, **you might have the idealistic goal of bringing an end to childhood poverty in the world**.
- A pragmatist can consider something to be true without needing to confirm that it is universally true. For example, **if humans commonly perceive the ocean as beautiful then the ocean is beautiful**.