# Intégrer le logiciel comme une production scientifique à part entière

Violaine Louvet

Atelier CasuHAL juin 2024

# Contenu de l'atelier

## Atelier d'1h30

- Introduction autour du logiciel
- Appréhender les spécificités à travers des exemples concrets
- Approfondir certaines de ces spécificités
- Du logiciel aux métadonnées

## En pratique

- Des apports « théoriques »
- Du travail collectif
- De la mise en application
- Des échanges et des discussions

## Les objectifs

- Mieux comprendre l'objet logiciel de recherche
- Avoir des éléments pour accompagner les communautés scientifiques
- Bien appréhender les enjeux autour des métadonnées

# Déroulé


1. Le logiciel de recherche : comprendre les spécificités et les enjeux
   - Définitions et contexte
   - Cadre juridique du logiciel
   - Reproductibilité
   - Archivage, signalement et diffusion
2. Le logiciel en pratique
3. La question des métadonnées
4. Conclusions

# Plan


1. Le logiciel de recherche : comprendre les spécificités et les enjeux
   - Définitions et contexte
   - Cadre juridique du logiciel
   - Reproductibilité
   - Archivage, signalement et diffusion
2. Le logiciel en pratique
3. La question des métadonnées
4. Conclusions

# Plan

1 Le logiciel de recherche : comprendre les spécificités et les enjeux
- Définitions et contexte
- Cadre juridique du logiciel
- Reproductibilité
- Archivage, signalement et diffusion

# Algorithme, code source, logiciel, de quoi parle-t-on ?

## Petit historique

- *Software* utilisé dès 1953 pour désigner la partie immatérielle de l'ordinateur par rapport au *hardware*, partie matérielle.
- Le mot *logiciel* apparaît en 1969 (à partir des mots *logique* et *matériel*).

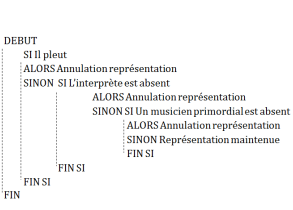
- Algorithme : décrit le déroulé pour la résolution d'un problème posé.
- Code source : mise en oeuvre et formalisation de l'algorithme dans un langage informatique (par exemple python, C++, java ...). C'est un (ou plusieurs) fichier(s) texte.
- Exécutable : traduction du code source (en général via un compilateur ou un interpréteur) en code binaire compréhensible par l'ordinateur.
- Logiciel : en général, l'ensemble global comprenant le code source et/ou l'exécutable, et le plus souvent la documentation, des exemples d'utilisation, éventuellement les dépendances ... et évidemment la licence associée.

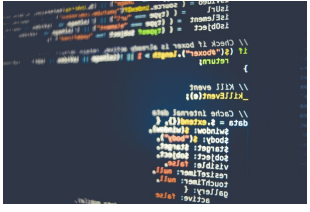
# Ce qu'il faut retenir

- « De manière simplifiée, l'algorithme est une recette de cuisine, et le code source sa réalisation concrète »(cf rapport Bothorel)

- L'« âme »d'un logiciel est dans son code source, c'est-à-dire dans les instructions telles qu'elles sont rédigées pour être lisibles par l'humain.

- Seul le code source permet d'accéder aux informations techniques et scientifiques



# Petite histoire de l'informatique et du logiciel

- Jusque dans les années 1970, seules certaines entreprises avaient les moyens financiers de s'acheter des ordinateurs et donc les logiciels associés.
  - $\longrightarrow$ Ces entreprises avaient un intérêt particulier à laisser les chercheurs et développeurs disposer de ces logiciels notamment pour les étudier et les modifier.
    - A cette période des débuts de l'informatique, le partage de code source entre chercheurs était une pratique universellement admise
    - les programmes informatiques sont alors traités, dans les universités concernées, de la même manière que n'importe quelle information scientifique : mis à la disposition de tout un chacun pour étude, exploitation et amélioration
- Progressivement, avec le développement des ordinateurs personnels, l'importance économique des logiciels apparaît.
  - Bill Gates (création de Microsoft en 1975) publie en 1976 An Open Letter to Hobbiyst dans laquelle il dénonce la faible protection des créations des développeurs.
  - En Octobre 1976, le Copyright Act est adopté aux Etats-Unis, protégeant les logiciels par le droit d'auteur américain.
  - $\longrightarrow$ C'est la naissance des logiciels propriétaires.

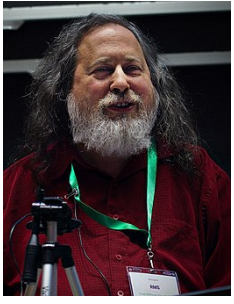
# Philosophie libre

- L'échange libre de logiciels a donc existé depuis les débuts de l'informatique, sans qu'on ait défini les principes de ces partages.
- En particulier dans la communauté des hackers (« bidouilleurs »), désignant originellement ces jeunes étudiants en informatique du MIT, génies de la programmation

  $\longrightarrow$ Un des principes de l'« éthique hacker » est que toute information est par nature libre

**Richard Stallman**

Informaticien au MIT, membre de la communauté des hackers, frustré de ne pas pouvoir modifier le logiciel qui contrôlait l'imprimante de son labo, il est alors convaincu de la nécessité pour les gens de pouvoir librement modifier le logiciel qu'ils utilisent.



# Naissance du logiciel libre

- Richard Stallman fonde la Free Software Foundation (FSF) en 1985.
- Cet organisme formalise les 4 grands principes du libre dans des licences, textes juridiques qui fixent les conditions pour exploiter un logiciel ou une oeuvre intellectuelle.
- La FSF soutient le développement du projet GNU depuis le début.
  - $\longrightarrow$ Le projet GNU concerne le développement d'un système d'exploitation complètement libre.
  - $\longrightarrow$ Sa version fonctionnelle s'appuie sur le noyau linux, développé indépendamment par Linus Torvalds

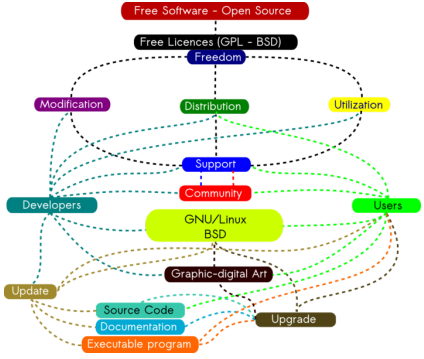
## Les 4 libertés fondamentales du logiciel libre

- possibilité d'utiliser l'oeuvre, pour tous les usages ;
- possibilité d'étudier l'oeuvre ;
- possibilité de redistribuer des copies de l'oeuvre ;
- possibilité de modifier l'oeuvre et de publier ses modifications.

Logiciel libre $\neq$ logiciel gratuit

# Rôle des communautés du logiciel libre

- La communauté d'un logiciel libre désigne le groupe de personnes qui collaborent pour développer, maintenir et promouvoir un logiciel libre.
- Cette communauté est généralement composées de développeurs, de contributeurs et d'utilisateurs.
- Cette action collective assure le partage des connaissances, la transparence des développements, et la flexibilité des contributions.
- Ces contributions peuvent prendre des formes multiples.
  - indiquer un bug,
  - suggérer une nouvelle fonctionnalité,
  - poser une question,
  - corriger un bug,
  - faire de la documentation,
  - ajouter une nouvelle fonctionnalité, ...



# Logiciels développés dans les laboratoires de recherche

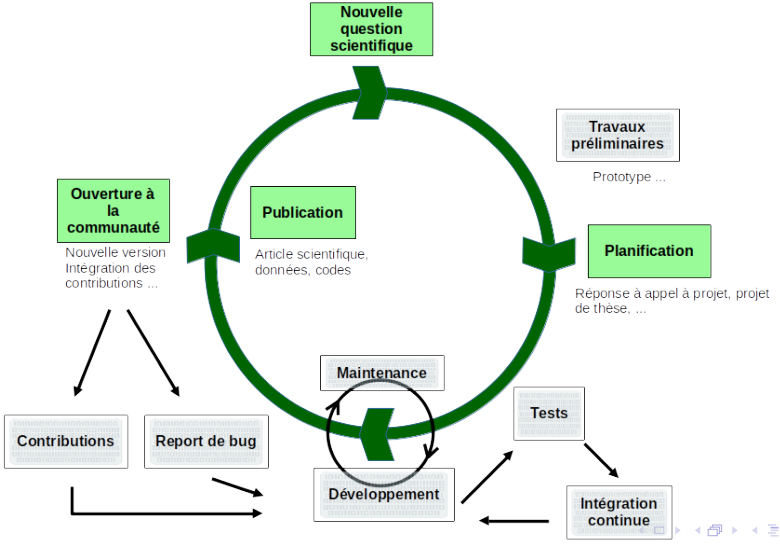
- Individuel : un chercheur / ingénieur (souvent un doctorant) développe seul un code adressant une question de recherche.
- Equipe de recherche : quelques chercheurs / ingénieurs, en général géographiquement (ou thématiquement) proche (même laboratoire) développent et partagent un code sur le sujet sur lequel ils travaillent.
- Communauté : organisation du développement d'un code à l'échelle de toute une communauté scientifique

→ Tous sont liés au processus de recherche et orientés vers un même objectif : la production de connaissances scientifiques

**Définition du collège logiciels du COSO**

Les logiciels de recherche sont développés pour répondre à des besoins spécifiques de la science. Ils sont conçus, maintenus, et utilisés par des scientifiques (chercheurs et ingénieurs) et institutions de recherche, éventuellement dans une dimension internationale. Ils peuvent découler de travaux de recherche comme ils peuvent les favoriser, notamment par des publications avant/sur/autour/avec le logiciel. Ceux-ci peuvent se formaliser de différentes façons (une plateforme, un intergiciel, un workflow ou une bibliothèque, module ou greffon d'un autre logiciel) et être ainsi en interaction dans un écosystème ou au contraire plus autonomes.

# Cycle de vie d'un code de recherche



# La vie d'un code de recherche

- Les questions scientifiques orientent le développement qui est complètement intégré au processus de recherche
- Les cycles et sous-cycles sont itératifs et imbriqués
- Selon le contexte, certaines étapes peuvent ne pas exister ou être juste esquissées (par exemple les tests et l'intégration continue)
- La temporalité est très variable : le cycle peut s'interrompre sur une durée plus ou moins longue (code dormant, voir mort) et reprendre si l'intérêt scientifique renaît

# Code de recherche $\neq$ données de recherche

- Les données de recherche sont plutôt passives, les codes sont intrinsèquement vivants
  - On ne change en général pas les données, collectées dans un contexte bien défini
  - On change éventuellement la façon dont on les traite et on les analyse (grâce à des codes)
  - Les codes sont associés à une (ou des) action(s) : création de connaissances, transformation d'informations, visualisation, ...
  - Le code peut être réutilisé tel que, en reproduisant son environnement et toutes ses dépendances mais on a surtout envie de le modifier pour l'adapter à nos besoins propres ou l'enrichir de nouvelles fonctionnalités
- Les codes s'appuient sur des dépendances et tout un environnement logiciel et matériel qui évolue sans cesse
  - Cela complexifie les questions de reproductibilité
- Les codes représentent un travail de création, et correspondent à un cadre juridique différent de celui des données

# Forges logicielles : au coeur des développements

- Au delà de la gestion de versions, une forge logicielle est un système complet en ligne de gestion, de partage et de maintenance collaborative de textes (et donc en particulier de codes sources)
- Intègre de nombreux outils :
  - système de gestion des versions (par exemple, via Git)
  - outil de suivi des bugs
  - gestionnaire de documentation
  - gestion des tâches
  - intégration continue ...

## Une ressource essentielle

- Favorise les contributions et facilite leur intégration (pull request)
- Simplifie les interactions entre les développeurs (tickets) et les utilisateurs (forums)
- Facilite la gestion des tests automatiques (intégration continue)

# Plan

1 Le logiciel de recherche : comprendre les spécificités et les enjeux
- Définitions et contexte
- Cadre juridique du logiciel
- Reproductibilité
- Archivage, signalement et diffusion

# Le cadre juridique du logiciel

Le logiciel est protégé par le droit d'auteur avec des règles spécifiques :

- Droits moraux attachés à l'auteur
- Droits patrimoniaux, qui régissent les modalités d'exploitation, sont propriétés de ou des institutions qui emploient le ou les auteurs
  - Contrairement aux autres droits d'auteurs, il y a une « dévolution automatique des droits patrimoniaux à l'employeur »
  - Depuis le 15 décembre 2021, c'est vrai aussi pour les stagiaires
- L'algorithme, considéré comme une suite d'idées, ou le modèle mathématique ne peuvent pas être soumis au droit d'auteur
- La documentation est protégée par le droit commun du droit d'auteur

# Attribution des droits

- Il est essentiel de pouvoir tracer les différentes contributions pour identifier à qui appartiennent les droits
- En général, différents rôles à considérer :
  - Participation essentielle au développement
  - Participation anecdotique (codage d'exemple, corrections, ...)
  - Participation à la diffusion (script d'installation, création d'un paquet ...)
- L'identification des auteurs doit se faire en amont et de façon concertée

# Les types de licences

Deux grands types de licence :

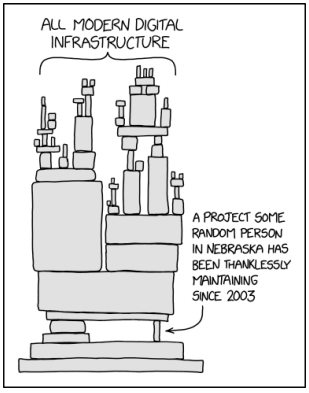
- Licences libres ou Open Source, termes plus ou moins similaires qui définit 4 types de liberté :
  - Liberté d'exécuter le programme, pour tous les usages.
  - Liberté d'étudier le fonctionnement du programme, et de l'adapter à vos besoins. Accès au code source condition requise.
  - Liberté de redistribuer des copies.
  - Liberté d'améliorer le programme et de publier vos améliorations, pour en faire profiter toute la communauté. Accès au code source condition requise.
- Il existe différents types de licences libres :
  - sans copyleft : la licence initiale ne s'impose pas. Permission de redistribuer et de modifier, mais aussi d'y ajouter des restrictions (ex : BSD).
  - copyleft faible : la licence initiale reste, des ajouts peuvent avoir une autre licence (ex : LGPL).
  - copyleft fort : la licence initiale s'impose sur tout. Licence dite contaminante (ex : GNU GPL).
- Licences propriétaires, donc non libre c'est-à-dire que seul l'auteur ou l'ayant droit du logiciel peut le modifier.

# Plan

1 Le logiciel de recherche : comprendre les spécificités et les enjeux
- Définitions et contexte
- Cadre juridique du logiciel
- Reproductibilité
- Archivage, signalement et diffusion

# Vraie et fausse reproductibilité [1]

- Le code ouvert permet et facilite l'inspection critique du source
- Mais il ne suffit pas de rendre le code public
- Il faut aussi ouvrir son environnement
- Et évaluer la fiabilité des dépendances
- Sans parler des dépendances possibles au hardware




a

a. https://xkcd.com/2347/

1. Open Science Days UGA 2022, K. Hinsen

# Complexité liée au logiciel

Chaque résultat numérique d'un code dépend de :

- les données d'entrée
- le code source
- les bibliothèques (dépendances) utilisées par le code
- les compilateurs / interpréteurs
- les options de compilation
- le système d'exploitation de l'ordinateur
- le hardware de la machine

Tout un environnement non stable, pas toujours documenté, qui évolue en permanence et qu'on ne contrôle pas.

# Codes de recherche et FAIR

Les principes FAIR sont-ils adaptables au logiciel ?

- Les objectifs des principes FAIR sont de rendre les objets de recherche réutilisables
- Problématique proche de la reproductibilité
- Or la reproductibilité en matière de logiciel est un idéal difficile à atteindre
- Questions ouvertes :
  - Comment définir les contributions et donc les citations ? En particulier quand il y a un grand nombre d'auteurs. Et des types de contributions très différents.
  - Comment intégrer l'environnement, les dépendances ?
  - Comment considérer la problématique de l'installation qui peut être très complexe ?
  - Comment prendre en compte la dynamique du code dans un identifiant pérenne et unique ?
  - ...

# Plan

1 Le logiciel de recherche : comprendre les spécificités et les enjeux
- Définitions et contexte
- Cadre juridique du logiciel
- Reproductibilité
- Archivage, signalement et diffusion

# Préserver sur le long terme

- Le code source est fragile :
  - Obsolescence des formats, problème matériel, dépendances à des outils (forge par exemple) qui peuvent disparaître ...
  - La perte des codes ayant été utilisés pour de la production scientifique arrive malheureusement **très** régulièrement

- Les logiciels sont un des piliers des processus de recherche, au côté des publications et des données et il est essentiel de les préserver

**Quelques exemples**

- Gitorious : forge logicielle communautaire, racheté en 2013 par une société norvégienne et polonaise, puis par Gitlab qui ferme le site en 2015.
- Google Code : site web destiné aux développeurs. Fermé en 2016.

$\longrightarrow$ nombreux codes perdus et beaucoup de liens cassés (en particulier dans certaines publications)

# Forges logicielles $\neq$ archives

## La forge est le coeur névralgique de tout développement

- Facilite la mise en oeuvre de bonnes pratiques de développement
- Absolument indispensable quand on développe à plus de un mais particulièrement utile aussi pour les développements individuels
- Simplifie aussi la diffusion du logiciel, son référencement, son archivage ...
  - En particulier, c'est le seul endroit où les informations sont constamment à jour

**MAIS**

- Ce n'est pas une archive de codes
- Cela ne sert pas à pérenniser sur le long terme

# Software Heritage : archive des codes sources

- Collecter l'intégralité des logiciels disponibles publiquement sous forme de code source
- Préserver : les codes sont stockés et accessibles sur le long terme
- Partager : les codes collectés sont organisés et indexés de façon à être référencés de manière optimale

Initiative à but non lucratif lancée en 20216 par INRIA et soutenu par de nombreux organismes, dont l'UNESCO
Membres fondateurs : Roberto Di Cosmo et Stefano Zacchiroli

# Archivage et référencement

- Une collaboration entre Sotware Heritage et l'archive ouverte HAL a permis de développer le dépôt de logiciels pour favoriser le référencement et la description des codes sources.
  - Le dépôt y est simplifié en utilisant l'identifiant SWHID si le logiciel est déjà archivé dans Software Heritage
  - Il est possible de déposer un fichier sous format compressé du code qui sera ensuite pérennisé dans SWH si ce n'est pas le cas

- La notive HAL permet d'assurer la description du logiciel
  - Avec des métadonnées de qualité et vérifiées

# Plan

1 Le logiciel de recherche : comprendre les spécificités et les enjeux
- Définitions et contexte
- Cadre juridique du logiciel
- Reproductibilité
- Archivage, signalement et diffusion

2 Le logiciel en pratique

3 La question des métadonnées

4 Conclusions

# Comprendre le logiciel à travers un premier exemple

## L'exemple : Pianolib

https://hal.science/hal-04571401v1

## Lien vers le quizz

https://api.socrative.com/rc/ASrrh7

# Le quizz

**Quelle est la licence du logiciel ?**

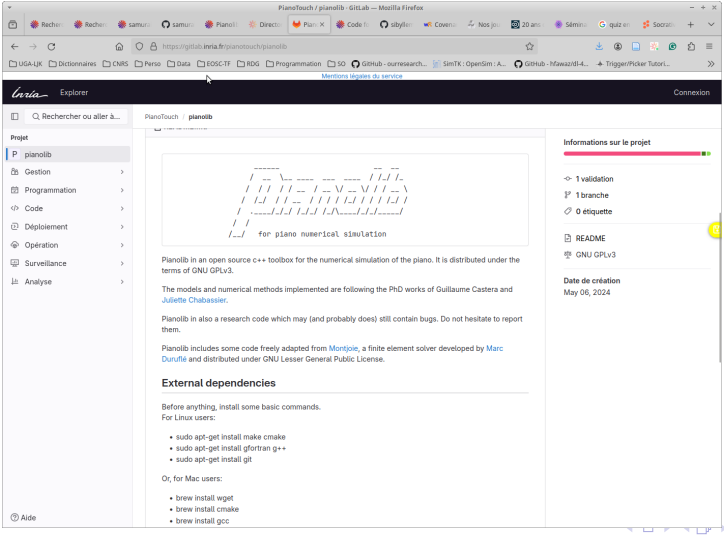
- CeCILL 2
- CC-BY-NC 4
- GPL V3
- BSD 2

# Le quizz

**Quelle est la licence du logiciel ?**

- CeCILL 2
- CC-BY-NC 4
- GPL V3
- BSD 2

# Quelques compléments : la question de la licence



# Le quizz

**Sur quelle forge est hébergée le code ?**

- gitlab.com
- INRIA
- Sourcesup
- github.com

# Le quizz

**Sur quelle forge est hébergée le code ?**

- gitlab.com
- INRIA
- Sourcesup
- github.com

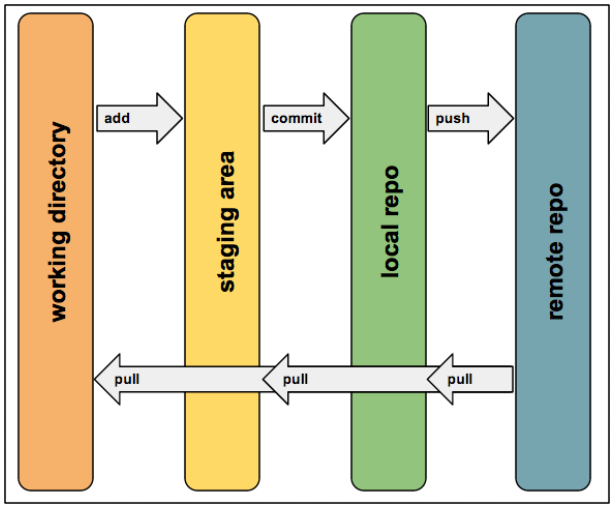
# Quelques compléments : Git et gitlab en deux slides

## Git n'est pas gitlab

- Git est un logiciel de gestion de versions décentralisé (logiciel libre sous GPLv2)
- GitLab est un logiciel libre de forge basé sur git. A noter que gitlab est scindé en deux versions : l'une libre, l'autre propriétaire. Gitlab est une plateforme qui peut être déployée dans les établissements ou laboratoires.

- Le système de gestion de versions a comme objectif de pouvoir retracer des modifications physiques dans la mémoire de façon intelligente
- Ces changements sont stockés dans un dépôt (repository)
- Git fonctionne en créant plusieurs dépôts des changements :
  - Le premier se trouve sur la même machine que les fichiers de travail
  - Le deuxième (si il existe) se trouve dans une autre machine, souvent un serveur ou une plateforme cloud comme GitLab, qui s'occupe de centraliser les changements

# Git et gitlab en deux slides



# Forges disponibles

Rapport du collège logiciels et codes sources du COSO sur les forges

- Forges de l'ESR
  - De nombreux établissements, organismes voir laboratoires ont déployé leur propre forge (en général gitlab mais il en existe quelques autres)
  - Liste non exhaustive : CNRS, INRIA, Huma-Num, INRAE, UGA, U Bordeaux, ...
  - SourceSup est la forge nationale hébergée par Renater. Elle s'appuie sur FusionForge (et pas gitlab)
  - La plupart de ces forges ont un accès restreint à leur communauté / personnels avec un degré d'ouverture plus ou moins important
- Forges commerciales (github, gitlab.com ...)
  - Très utilisées dans le cadre de collaborations internationales ou pour assurer plus de visibilité
  - Pas de contraintes d'accès comparé à la plupart des forges ESR
  - Mais attention aux conditions d'utilisation ! Et à la pérennité de ces plateformes

# La question du choix de la forge

- La plupart des logiciels libres de l'ESR à périmètre international et/ou sociétal utilisent une forge commerciale.
- Facilite tous les types de contributions par tout le monde.
- **MAIS**
  - Pas de garantie de pérennité.
  - Souvent soumises à des juridictions extra-européennes.
  - Utilisation des sources sans réel contrôle pour l'apprentissage des outils d'aide à l'écriture de code (OpenAI Codex/GitHub copilot, GitLab suggestions, etc.).
  - Flou juridique sur la question du respect des licences pour ces outils.
- Il y a un enjeu majeur de **souveraineté**.
- Le collège logiciels et codes sources du COSO travaille actuellement sur des préconisations pour répondre à ces enjeux.

# Le quizz

## Le code est-il archivé sur Software Heritage ?

- Oui
- Non

## Quel est le langage de programmation du code ?

- Python
- Java
- Fortran
- C++

# Le quizz

## Le code est-il archivé sur Software Heritage ?

- Oui
- Non

## Quel est le langage de programmation du code ?

- Python
- Java
- Fortran
- C++

# Quelques compléments : langage de programmation

- Permet la formalisation d'un algorithme en un code source
- Il en existe plusieurs centaines, mais les plus courants se comptent plutôt en dizaines
- Un langage de programmation est compréhensible par l'humain mais pas par l'ordinateur ! Il faut une étape de « traduction » : via un compilateur ou un interpréteur
- Selon le type de logiciel, on va choisir le langage :
  - majoritairement utilisé dans sa communauté
  - portable et disponible sur les systèmes d'exploitation les plus courants
  - selon les besoins, performant et modulaire

**Classement des langages les plus utilisés**

https://www.tiobe.com/tiobe-index/

# Langage compilé et interprété

## Langage compilé

- Nécessite un compilateur qui va transformer le code source en exécutable
- Exemples : C, C++, Fortran
- Avantages : performance, vérification d'erreurs lors de l'étape de compilation
- Inconvénients : l'exécutable dépend de la machine (OS, matériel) sur laquelle il a été créé

## Langage interprété

- Nécessite un interprèteur qui va exécuter ligne par ligne le programme
- Exemple : Python, Javascript, PHP
- Avantages : portabilité, rapidité de développement
- Inconvénients : lenteur d'exécution

# Comprendre le logiciel à travers un deuxième exemple

**L'exemple : Samurai**

https://hal.science/hal-04545389v1

**Lien vers le quizz**

Le même !

# Le quizz

**Le développement de ce code est-il**

- Très actif
- Pas vraiment actif

**Tous les auteurs sont des développeurs**

- Vrai
- Faux

**Rien n'est prévu pour faciliter les contributions**

- Vrai
- Faux

# Le quizz

**Le développement de ce code est-il**

- Très actif
- Pas vraiment actif

**Tous les auteurs sont des développeurs**

- Vrai
- Faux

**Rien n'est prévu pour faciliter les contributions**

- Vrai
- Faux

Vérifions tout ça : `https://hal.science/hal-04545389v1`

# Plan

1 Le logiciel de recherche : comprendre les spécificités et les enjeux
- Définitions et contexte
- Cadre juridique du logiciel
- Reproductibilité
- Archivage, signalement et diffusion

2 Le logiciel en pratique

3 La question des métadonnées

4 Conclusions

# Métadonnées

## Métadonnées intrinsèques

Font référence aux informations contenues dans le code source lui-même, telles que les fichiers README, les licences et les fichiers de gestion des paquets.

## Métadonnées extrinsèques

Métadonnées sur des entités externes telles que des plateformes d'hébergement de code, des registres ou des référentiels scientifiques contenant des informations supplémentaires sur le logiciel.
Elles renseignent aussi sur la relation entre le logiciel et dautres produits de recherche : des publications, des jeux de données. Ces métadonnées ne sont pas incluses dans le code source.

# L'exemple de CodeMeta

- Format de métadonnées qui joue un rôle de pivot / table de concordance entre différents vocabulaires disponibles
- Format privilégié pour HAL + SWH
- La présence d'un fichier codemeta.json dans le dépôt du projet sur SWH est détecté par HAL qui charge automatiquement les métadonnées
- Existence d'un outil en ligne facilitant la création du fichier json : `https://codemeta.github.io/codemeta-generator/` (mais aussi d'outils spécifiques pour les paquets R et Python).

Remarque : dans le dépôt du logiciel Samurai, on constate la présence d'un fichier codemeta.json

# CodeMeta en pratique

## Travail sur le code SMILEI

https://smileipic.github.io/Smilei/index.html

- Travail en groupe :
- Chaque groupe va travailler sur un bloc de CodeMeta pour le remplir, sur une durée de 10 à 15 mn :
  - Groupe 1 : The software itself
  - Groupe 2 : Discoverability and citation
  - Groupe 3 : Development community / tools
  - Groupe 4 : Run-time environment
  - Groupe 5 : Current version of the software
  - Groupe 6 : Editorial review
  - Groupe 7 : Authors + Contributors
- Pas de panique si vous n'avez pas une des entrées attendues !
- Un rapporteur par groupe pour expliquer la façon dont les champs du CodeMeta generator ont été remplis pendant une phase de restitution à l'issue du travail collectif

# Plan

1 Le logiciel de recherche : comprendre les spécificités et les enjeux
- Définitions et contexte
- Cadre juridique du logiciel
- Reproductibilité
- Archivage, signalement et diffusion

2 Le logiciel en pratique

3 La question des métadonnées

4 Conclusions

# Conclusions

- La plupart des développements réalisés dans les laboratoires de recherche le sont sous licence libre
  - Le logiciel libre est né dans la pratique universitaire, précurseur du mouvement de la science ouverte
- Le logiciel est différent des données et doit être considéré dans toute sa spécificité
  - complexité des questions de reproductibilité
  - inadéquation de certains principes FAIR
  - ...
- Le code source est un objet fragile qui nécessite des infrastructures adaptées pour sa pérennisation
- Software Heritage propose l'archivage des codes sources publics
  - Le lien avec HAL permet d'assurer une description s'appuyant sur des métadonnées vérifiées et complètes grâce en particulier au vocabulaire Codemeta.
  - La question du lien publications / données / logiciels est à adresser à partir de cet existant.