Comply Vantage

Types of Opensource Licenses:

There are over more than 200 licenses for opensource :

<https://opensource.org/licenses>

Most used open source license:

**Permissive Licenses are:**

MIT,

ISC,

3 clause BSD, (flutter)

2 clause BSD,

Apache 2.0

MIT, ISC, 2 clause BSD are almost equivalent, (as they said, Hey used my code, do what ever you want and don’t sue me and add a notice in your product

JS frameworks: React, Angular, Vue.js uses MIT license

3 clause BSD has the third clause which states that “you cannot endorse or promote the product by using the name of Copywrite holders or contributors”

Go lang uses 3 clause BSD

What permissive license does is: we can modify or change permissive license code but we have to mention that you are using some ones project every where in documentation as well as in GUI

3 clause BSD , says that we cannot promote the application using the name of the developers to whom our code we used

**Weak CopyLeft licenses are:**

LGPLv3

MPLv2 (firebird, Mozilla)

**CopyLeft licenses are:**

What is a copyleft license? A copyleft license makes it mandatory to distribute all the source code of software based on previous copyleft work upon release to the public. Copyleft ensures that any attempt by an organization to use open source work in a commercial venture isn't viable.

AGPLv3 (MONGODB)

GPLv2

GPLv3:

Like the GPL v2, GPL 3 is a strong [copyleft license](https://fossa.com/blog/all-about-copyleft-licenses/), meaning that any copy or modification of the original code must also be released under the GPL v3. In other words, you can take the GPL 3’d code, add to it or make major changes, then distribute your version. However, your version is subject to the same license requirements, meaning that it must be under GPL v3  as well — anyone can see your modified code and install it for their own purposes.

**GPL v3 Requirements**

The license terms of GPL v2 and GPL v3 are similar. They require users of the code to:

1. Include a copy of the full license text
2. State all significant changes made to the original software
3. Make available the original source code when you distribute any binaries based on the licensed work
4. Include a copy of the original copyright notice

In addition, GPL v3 states that anyone who includes the code as part of a consumer device has to include any installation information necessary to update and reinstall the software.

Copywrite

Trademark

Patent

Written piece of code or book or movie all fall in the category of copywrite

But when we make any invention using code, like making a mobile banking system, networking protocols so they are patents

And trademark is like a identification of any product like flutter logo, facebook logo etc

# JAVA CODE STRUCTURE:

Rules for Regex Engine:

.java File

Import statements:

Import java.util.\*

Import java.io.\*

Print statements of java:

System.out.print()

System.out.println()

System.out.printf()

Java Class Structure:

Access modifier? -> static? -> class keyword -> ClassName -> Bracker {

Main statement in Java:

public static void main(String[] args)

Inheritence:

Java uses extends keyword for inheritance

**class child extends Parent{}**

**commonly used datatypes and syntax of declaring and initializing variable in java:**

int myNum = 5; // Integer (whole number)

float myFloatNum = 5.99f; // Floating point number

char myLetter = 'D'; // Character

boolean myBool = true; // Boolean

String myText = "Hello"; // String

Most similar languages as java are following

**C#:**

using System;

namespace HelloWorldApplication {

class HelloWorld {

char varr = 02

static void Main(string[] args) {

/\* my first program in C# \*/

Console.WriteLine("Hello World");

Console.ReadKey();

}

}

}

Using keyword (but java has import keyword)

Namespace keyword (java doesnot have) (we can check using regex if java code has namespace named variable or class/function)

Console.WriteLine as print statement

Extension as .cs

Difference in Inheritence:

class Car : Vehicle // derived class (child)

{

public string modelName = "Mustang"; // Car field

}

**Kotlin:**

**fun** **updatePerson**(firstName: **String**, middleName: **String**, lastName: **String**) {

*// Your code*

}

*// Good*

**fun** **updatePerson**(

firstName: **String**,

middleName: **String**,

lastName: **String**

) {

*// Your code*

}

.kt extension

Difference in declaring a variable and mentioning its datatype

Function keyword is **fun**

Inheritance:

**open** **class** Base(p: Int)

**class** Derived(p: Int) : Base(p)

(use of open keyword)

Commonly Used datatypes and syntax for declaring and initializing variable in kotlin:

val myNum = 5 // Int

val myDoubleNum = 5.99 // Double

val myLetter = 'D' // Char

val myBoolean = true // Boolean

val myText = "Hello" // String

**Key differences of Java and c#:**

1, C# doesn't have checked exception.

2, C# doesn't have anonymous inner class

3, Java doesn't have struct.

4, Java doesn't support delegate.

5, Java doesn't have static class

6, Java doesn't have extension methods

7, Java doesn't have property

8, Java doesn't have out and ref qualifiers

9, Java doesn't have unsigned type

10, Java doesn't have nullable value types

# FOSSOLOGY

Copyright/Email/URL/Author Analysis

ECC Analysis, scanning for text fragments potentially relevant for export control

Keyword Analysis

MIME-type Analysis (Determine mimetype of every file. Not needed for licenses or buckets)

Monk License Analysis, scanning for licenses performing a text comparison

Nomos License Analysis, scanning for licenses using regular expressions

Ojo License Analysis, scanning for licenses using SPDX-License-Identifier

Package Analysis (Parse package headers)

REUSE.Software Analysis (forces \*Ojo License Analysis\*)

Software Heritage Analysis

**Standup 4**

|  |
| --- |
| Features Lacking in Fossology:   1. It doesnot search the licenses of Dependencies 2. It doesnot search the licenses of Third-party packages 3. It doesnot break the code into components 4. It doesnot check the code for the licensing from any knowledge base 5. It is purely a regex-engine based software that works only on the specific text and keywords using PHP regex 6. If we remove the license.txt file and all the other text of License information from the code files, then fossology will not be able to find out the license of codebase. It is purely relies text in the codebase or other files like License.txt or Readme.md 7. It is so slow that it can take up to 2-3 hours to scan on bigger repositories 8. It doesn’t check for language   Working of Black-DucK:   1. It breaks the code and check the chunks of code for the licensing 2. It checks for dependencies and their licensing from knowledge base (but not sure yet) 3. It checks for third party packages and their licensing as well   Current Task:   1. Reviewing code of Fossology |

**Keywords used in licensing extracted through manual testing (by uploading different repos on fossology):**

* Copyright ©2014-2015
* License/Licence
* Redistribute
* Distribute
* Required
* Warranty
* Free software
* Source code
* terms and conditions
* Patent
* Permission
* Granted
* Subject to
* Derivate

**Key words mentioned in the fossology repo file (\_split\_words):**

**\_split\_words**

Fossology>src>nomos>agent>\_split\_words:

* source
* free
* under
* copyright
* grant
* software
* distribut
* licen
* [iu][nst]

## Tasks Assigned on 8th April, Friday, 1st Sprint Review meeting:

**STAND UP 5:**

|  |  |
| --- | --- |
| Tasks | Status: Notes (Optional) |
| Search for the regex patterns in fossology if we can use it in our code or not | No, we cannot use fossology’s regex patterns in our code as, it’s written in PHP and the regex of PHP is different from the Python  To use the fossology’s regex in our code base requires to learn PHP, code structure of fossology and regex, which is not the good option |
| Search keywords from fossology repo that fossology target in the codebase. | * source * free * under * copyright * grant * software * distribut * licen * [iu][nst] |
| Test Guess Lang and write down test cases |  |
| Fuzzy logic python |  |

## 

Search for the regex patterns in fossology: NO

Search keywords fossology check for in fossology code: DONE

Search of the OSS first and then code the validator: DONE

Search regex based java code analyzer in python

Write down test cases for guesslang

Check guesslang if it has functions to unzip files, moving between directories, and read the texts from the from files

Fuzzy logic python

**Stand up 6:**

Testing of Guesslang:

**Test cases for GUESSLANG:**

|  |  |  |  |
| --- | --- | --- | --- |
| Test case ID | Test case scenario | Predicted out | True output |
| 1 | Java main function with print statement | Java | Java |
| 2 | Java function with print statement inside java class | Java | Java |
| 3 | Signature of main function | Java | Java |
| 4 | Print statement of java | INI | Java |
| 5 | another print statement of java | Batchfile | Java |
| 6 | comments of java | Batch file | Java |
| 7 | declaration and initialization of java variable separately and print it | SQL | Java |
| 8 | declaration and initialization in same line with print statement | Batch File | Java |
| 9 | all datatypes of java variable’s declaration and initialization | Batch File | Java |
| 10 | Type casting in java | Java | Java |
| 11 | arthematic operations | Haskell | Java |
| 12 | java string operations | Batch File | Java |
| 13 | if else in java | Java | Java |
| 14 | switch statements in java | Java | Java |
| 15 | arays in java | Java | Java |
| 16 | multidimensional arrays | Java | Java |
| 17 | multidimensional array, with lops, function and class | Java | Java |
| 18 | method overriding | Java | Java |
| 19 | Java encapsulation | Java | Java |
| 20 | java import statement with | Java | Java |
| 21 | Java I/O exception and file creation | Java | Java |
| 22 | java swing code | Java | Java |
| 23 | java android project | Java | Java |
| 24 | Selecting photo from gallery code in java | Java | Java |
| 25 | java code for camera activity | Java | Java |
| 26 | xml for java gallery screen | XML | XML |
| 27 | POM File | XML | XML |
| 28 | POM File | XML | XML |
| 29 | POM File | XML | XML |

***Code is uploaded on this below link:***

<https://colab.research.google.com/drive/11eIbUdhbofhRAKVtTtJOI-M4mjSeXYZu?usp=sharing>

**Standup 7**

**Done:**

1. Settingup virtual environment
2. Working on the sprint deliverable ( the working unit will accept the file, check it for it’s extension, open a file and read the file, extract all code/text from the file and then pass it to the guesslang for the language detection)
3. Fuzzy logic

# Creating Virtual Environment:

1. Download anaconda:

|  |
| --- |
| <https://docs.anaconda.com/anaconda/install/windows/> |

And follow all the steps in the given link

1. Download Python:

|  |
| --- |
| <https://www.python.org/downloads/> |

And don’t forget to check the checkpoint (Add path variable) after opening the .exe of python

1. Type anaconda in Windows search bar and Open the anaconda prompt
2. Create new virtual environment on anaconda by typing this command:

conda create -n myenv

*myenv: is the name of virtual environment, it can be anything*

1. Activate the Environment using the command as follows.

conda activate myenv

myenv in the name of virtual environment

1. Install the following libraries, by using same commands as below

pip install tensorflow

pip install guesslang

1. Now to go VS code and open the .py file
2. There’s an option in Bottom right corner of VS for selecting the python Interpreter of python virtual environment to execute the code
3. Select the Python interpreter named as same as your Virtual Environment name
4. Click on the Run button and click start without debugging

**Standup 8:**

Writing code for sprint deliverable and will test it.

Will write the step-by-step procedure to setup virtual environment for python

SCAN OSS Database:

ScanOSS makes the finger print of every code to match it with the knowledge base which already has a lot of finger prints

There are many algorithms for generating the finger prints like SHA, MD5

But Scanoss uses its own algorithm to generate fingerprints of code

And it uses winnowing algorithm to compare finger prints with knowledge base

The knowledge base stores source code fingerprints for entire files and code fractions (snippets). These fingerprints are used for matching source code during analysis. Full file fingerprints have been standardized by using cryptographic hash algorithms, such as **MD5** and **SHA**. However, looking at the market for SCA tooling, there are no standards defined for comparing snippets. SCA vendors implement their own proprietary algorithms for calculating snippet fingerprints, which means it is virtually impossible to compare results from different tools. Moreover, the proprietary nature of such algorithms imply that closed binaries are used for performing analysis, which is always a concern given the sensitive nature of corporate source code.

SCANOSS uses an open algorithm known as **Winnowing**, that has been used extensively in academic circles to obtain and compare fingerprints from documents and source code. These fingerprints are used to detect plagiarism against known texts and source code. There are several open source implementations of the Winnowing algorithm available today. Given the wide adoption and broad availability of open source implementations, SCANOSS has adapted this algorithm for indexing and comparing massive amounts of source code.

**ScanOSS Database:**  
SCANOSS designed a database engine specifically for this use case and has already passed the 2 trillion fingerprints mark. SCANOSS’ DatabaseEngine (called: ‘LDB’) organizes data in mapped linked lists, enablinghardware-speed searches. Comparisons are performed in microseconds,which allows the scanning of thousands of files per second while the database footprint is kept to a bare minimum and can be distributed across devices.

The SCANOSS Inventory Engine performs comparison of source code against the Open Source Knowledge Base using the provided LDB Database interfaces. Analysis is performed either on source code files or on pre-calculated Winnowing fingerprints. Output is presented in JSON format.

ScanOSS:

There’s only 1 repo on the ScanOSS github, which is purely written in the Python, and all the other repos are written in C language

I Have explored the Python ones Repo and study all the available documentation on the internet about the ScanOSS

Finger print

Algorithm for the finger print

Finger print matcher

CLI setup

Generating Json Output using CLI

Python Module setup

Understand the code of Scanner.py

Java file testing

Generating Output

Found the API (URL) from the code

[***https://osskb.org/api/scan/direct***](https://osskb.org/api/scan/direct)

LBD and engine comes under GPL2.0 ,

Shows the website where repository exists

their mechanism to search the code from LDB,

finger print algorithm MD5

Matching algorithms Winnowing

they have 3 criterias to check the licenses of the code

1st is ScanOSS look for the whole code files and check the license using License Text in the comments

2nd Extracts the code and check the code in its Local Database

3rd Breakdown the code and again check the code in it's local database

ScanOSS done this task in no time as they are using the hash algorithms to check the simi

And also I have set up the CLI and will explore the ScanOSS using its CLI today

Document everything

CLI, Command for uploading project (For validation Module)

UML

scanoss-py scan -o scan-output.json D:\DependencyCheck-main

**Validator:**

.ZIP, .RAR, .ARJ, .TAR.GZ, and .TGZ commonly used compressed formats.

ScanOSS, Files

FILTERED\_DIRS = { # Folders to skip

"nbproject", "nbbuild", "nbdist", "\_\_pycache\_\_", "venv", "\_yardoc", "eggs", "wheels", "htmlcov",

"\_\_pypackages\_\_"

}

FILTERED\_DIR\_EXT = { # Folder endings to skip

".egg-info"

}

FILTERED\_EXT = { # File extensions to skip

".1", ".2", ".3", ".4", ".5", ".6", ".7", ".8", ".9", ".ac", ".adoc", ".am",

".asciidoc", ".bmp", ".build", ".cfg", ".chm", ".class", ".cmake", ".cnf",

".conf", ".config", ".contributors", ".copying", ".crt", ".csproj", ".css",

".csv", ".dat", ".data", ".doc", ".docx", ".dtd", ".dts", ".iws", ".c9", ".c9revisions",

".dtsi", ".dump", ".eot", ".eps", ".geojson", ".gdoc", ".gif",

".glif", ".gmo", ".gradle", ".guess", ".hex", ".htm", ".html", ".ico", ".iml",

".in", ".inc", ".info", ".ini", ".ipynb", ".jpeg", ".jpg", ".json", ".jsonld", ".lock",

".log", ".m4", ".map", ".markdown", ".md", ".md5", ".meta", ".mk", ".mxml",

".o", ".otf", ".out", ".pbtxt", ".pdf", ".pem", ".phtml", ".plist", ".png",

".po", ".ppt", ".prefs", ".properties", ".pyc", ".qdoc", ".result", ".rgb",

".rst", ".scss", ".sha", ".sha1", ".sha2", ".sha256", ".sln", ".spec", ".sql",

".sub", ".svg", ".svn-base", ".tab", ".template", ".test", ".tex", ".tiff",

".toml", ".ttf", ".txt", ".utf-8", ".vim", ".wav", ".whl", ".woff", ".xht",

".xhtml", ".xls", ".xlsx", ".xml", ".xpm", ".xsd", ".xul", ".yaml", ".yml", ".wfp",

".editorconfig", ".dotcover", ".pid", ".lcov", ".egg", ".manifest", ".cache", ".coverage", ".cover",

".gem", ".lst", ".pickle", ".pdb", ".gml", ".pot", ".plt",

# File endings

"-doc", "changelog", "config", "copying", "license", "authors", "news",

"licenses", "notice",

"readme", "swiftdoc", "texidoc", "todo", "version", "ignore", "manifest", "sqlite", "sqlite3"

}

FILTERED\_FILES = { # Files to skip

"gradlew", "gradlew.bat", "mvnw", "mvnw.cmd", "gradle-wrapper.jar", "maven-wrapper.jar",

"thumbs.db", "babel.config.js",

"license.txt", "license.md", "copying.lib", "makefile"

}

WFP\_FILE\_START = "file="

1. Check if compressed path is valid or not
2. If valid then we go for further processing(to line 4) and if not (to line 3)
3. It will throw exception as “No such Directory Exists to unzip!”
4. If no argument passed (unzipping folders)to extract files at desired location then files will be extracted at the default location of terminal
5. If argument given to extract files at desired location then first Python will first check if it is a valid path(check if directory exists) or not
6. If path is valid it will extract all the files there otherwise
7. It will raise exception that “No such Directory exists to extract the Files!”
8. Zipfile python library only accepts .zip formats
9. CHECK what if all the files already exist in the folder (what will be the name of new extracted files)
10. If files already exists in the folder then it will over ride the files
11. CHECKED: if we create our own .zip file using windows, zipFile library of python works on it charmingly
12. Research on Shutil python library
13. Checking os.path.dir() either it checks for only directory or zip and other files too
14. Making a new directory and extract all the files there
15. Before making a new directory we get the folder name by the file path using dirname() method
16. DOINGS: parsing through file, checking folders if exist and also parse through it, and checking zip file and also uncompress and parse through it
17. CHECK implemented : if there is a directory, parser will parser through it and get all the files from there as well

**Patool library:**

patool supports 7z (.7z), ACE (.ace), ADF (.adf), ALZIP (.alz), APE (.ape), AR (.a), ARC (.arc), ARJ (.arj), BZIP2 (.bz2), CAB (.cab), COMPRESS (.Z), CPIO (.cpio), DEB (.deb), DMS (.dms), FLAC (.flac), GZIP (.gz), ISO (.iso), LRZIP (.lrz), LZH (.lha, .lzh), LZIP (.lz), LZMA (.lzma), LZOP (.lzo), RPM (.rpm), RAR (.rar), RZIP (.rz), SHN (.shn), TAR (.tar), XZ (.xz), ZIP (.zip, .jar), ZOO (.zoo) and ZPAQ (.zpaq) formats. It relies on helper applications to handle those archive formats (for example bzip2 for BZIP2 archives).

CANNOT USE PATOOL because it has GPL license:

|  |  |
| --- | --- |
| 'r' | open for reading (default) |
| 'w' | open for writing, truncating the file first |
| 'x' | create a new file and open it for writing |
| 'a' | open for writing, appending to the end of the file if it exists |
| 'b' | binary mode |
| 't' | text mode (default) |
| '+' | open a disk file for updating (reading and writing) |
| 'U' | universal newline mode (deprecated) |

# Test cases for Validator:

|  |  |  |
| --- | --- | --- |
| Test Case ID | Test Case Description | Results |
| 1 | Getting Paths of different Directories | Checked |
| 2 | Checking Extensions of different files (.zip etc) | Checked |
| 3 | Validating The extensions | Checked |
| 4 | Accepting Zip files | Checked |
| 5 | Accepting Rar files | Not Accepting |
| 6 | If path of zip file is not valid | Raise Exception |
| 7 | Unzipping the .Zip folder | Checked |
| 8 | Checking all the files if any corrupt or not | Checked |
| 9 | Getting the user provided directory for extraction | Checked |
| 10 | Validating the user provided if it does not exist | Raise Exception |
| 11 | Extract all the Files in user provided Directory | Checked |
| 12 | Making a new folder named at user provided location as ComplyVantage in which all the files to be extracted | Checked |
| 13 | if folder already exists as ComplyVantage | Raise Exception |
| 14 | If user does not provide any Path | Extract at Deafult folder path |
| 15 | Parsing through all the files | Checked |
| 16 | Storing name of all the extracted files in the list | Checked |
| 17 | Storing root address of all the extracted files in the list | Checked |
| 18 | Storing complete Address of all the Extracted files in the list | Checked |
| 19 | Checking extensions of all the files | Checked |
| 20 | Storing extension of files in the list | Checked |
| 21 | Joining root directory and files name to get the complete path of the file to be extracted | Checked |
| 22 | Parse all the files and read content of it | Checked |
| 23 | Pass content/code to GuessLang | Checked |
| 24 | Store all the outputs of Guesslang in the list | Checked |
| 25 | Making Dictionary of   * File Name * Predictions of GuessLang * Extensions * Root Path * File Path * IsJava | Checked |
| 26 | Creating DataFrame of Pandas | Checked |
| 27 | Making CSV using Pandas DataFrame | Checked |
| 28 | If CSV already exist as same name | Overwrite the CSV |