Secured and protected content

If no

If yes

delete

Check for duplicate file

If no

If yes

Corrupt files

File1

File2

User Authentication

repair Corrupted  Module

Modules

1. Corrupted  Module: A corrupted file is one that is damaged, and does not perform properly. This can apply to any type of file, from program files to system files and all types of documents. Just about everybody has probably had an issue with a corrupted file at some point in time.

There are many causes of file corruption, such as a software bug, a virus, a computer crash, or bad sectors on a hard drive.

How Are Files Corrupted?

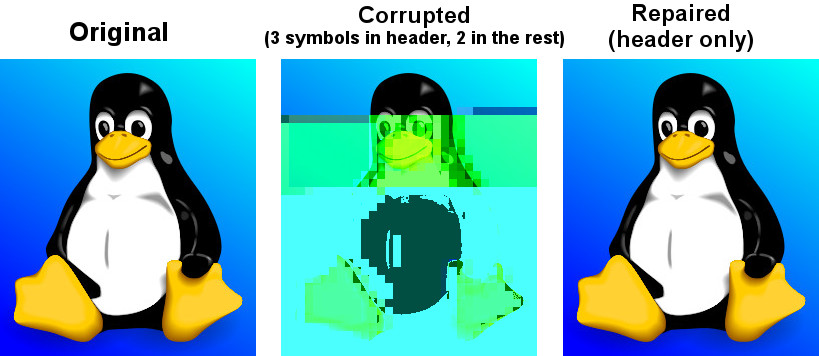
There are many ways a file can become corrupted. Data corruption can occur when there's a problem during the saving process. For example, if your computer crashes while you're saving a file, there's a chance the file might become corrupted. A power outage can affect multiple open files.

An application might experience an error while saving or creating a file, corrupting the file in the process. A browser might run into problems when downloading a file, resulting in file corruption. Viruses can corrupt data files, as can an interruption in normal computer processes.

Physical problems can cause data corruption, as well. For example, if a hard drive is dropped, shaken, or shifted rapidly too many times, electrical and mechanical malfunctions could result in corrupted files. A corrupted file might be a sign of a bad physical sector, which is a larger problem.

This project aims to provide a set of open source, cross-platform, easy to use and easy to maintain (readable code) to protect and manage data for long term storage. The project is done in pure-Python to meet those criteria.

Here is an example of what pyFileFixity can do:



On the left, this is the original image.

At the center, the same image but with a few symbols corrupted (only 3 in header and 2 in the rest of the file, which equals to 5 bytes corrupted in total, over 19KB which is the total file size). Only a few corrupted bytes are enough to make the image looks like totally unrecoverable, and yet we are lucky, because the image could be unreadable at all if any of the “magic bytes” were to be corrupted!

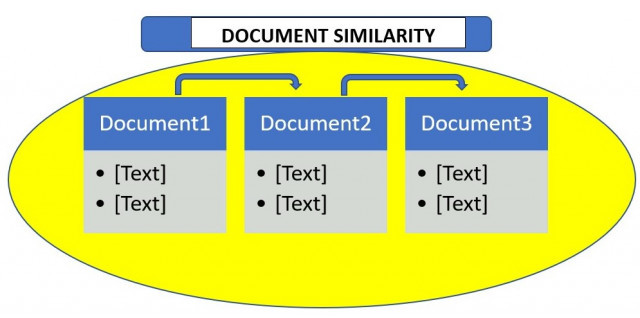
At the right, the corrupted image was repaired using header\_ecc.py of pyFileFixity. This repaired only the image header (ie, the first part of the file), so only the first 3 corrupted bytes were repaired, not the 2 bytes in the rest of the file, but we can see the image looks like it’s totally repaired! And the best thing is that it only costed the generation of a “ecc repair file”, which size is only 3.3KB (17% of the original file)!

This works because most files will store the most important information to read them at their beginning, also called “file’s header”, so repairing this part will almost always ensure the possibility to read the file (even if the rest of the file is still corrupted, if the header is safe, you can read it).

1. Duplicate File Finding Process

* through traversing the folder it will find duplicate files. this application will compute a hash for every file present in the folder regardless of their name and are stored in a dictionary manner with hash being the key and path to the file as value. Then application iterates over the files and calls algorithms to find duplicates.

1. Accuracy check Document similarity, as the name suggests determines how similar are the two given documents.



Document similarity is calculated by calculating document distance. Document distance is a concept where words(documents) are treated as vectors and is calculated as the angle between two given document vectors. Document vectors are the frequency of occurrences of words in a given document.