Module **duplicates**

Functions

def ClusterAllFiles(files: list, sizeThreshold)

Description

Cluster the list of files based on the specified threshold. Then further cluster within each cluster based on file extensions. This reduced the number of fingerprints required to be calculated - since only files that are identical in size and extension could ever be duplicates.

Args:

```
files : list[File or pathlike]
    #: The list of files (or pathlike file objects) to cluster
sizeThreshold : int
    #: The size of the maximum difference between sequential elements in each cluster (in bytes)
```

Outputs

```
allSizeExtClusters: list[list[File]]
    #: List containing all clusters
```

def ClusterByExtension(files: list)

Description

Further cluster files (after size clustering) based on their extension. Any file with an extension that appears only once is dropped.

Args:

```
files : list[File]
    #: The list of files to cluster
```

Outputs

```
extensionClusters: list[list[File]]
    #: List containing all extension clusters (each cluster is a list of files with that extension)
```

def ClusterBySize(files: list, threshold=1000)

Description

Cluster the list of files based on the specified threshold. Files are clustered sequentially - i.e files are added to the cluster if the difference between the next - previous file is less than the threshold.

Args:

```
files : list[File or pathlike]
    #: The list of files (or pathlike file objects) to cluster
threshold : int
    #: The size of the maximum difference between sequential elements in each cluster (in bytes)
```

Outputs

```
fileClusters: list[list[File]]
    #: List containing all size clusters
```

def FindDuplicateDirectories(log: src.fileorganiser.general.Log)

Description:

For all the duplicated files, check if they reside in duplicate directories, and then recursively check their parents if found duplicate directories. Returns the result and summary lines. Args:

```
log : Log
    #: The Log containing the duplicate results
```

Outputs:

```
summaryLines : list[str]
    #: The list of summary lines
folderResultLines : list[str]
    #: The list of folder pair result lines
```

def FingerprintAllFiles(files: list, log: src.fileorganiser.general.Log)

Description

Given a list of files calculate the fingerprint (MD5 hash) for each file. Fingerprints are stored on each file object and within the log.FpFilesDict which stores the fingerprint -> files with that fingerprint.

Args:

```
files : list[File]
    #: The list of all files to be fingerprinted
log : Log
    #: The log object for the duplicate search
```

Outputs

None #: Results are stored in the log object

def FingerprintAllFolders(folders: list, log: src.fileorganiser.general.Log)

Description

Fingerprint all files within the given list of folders (the toplevel of each folder only)

Args:

```
folders : list[Folder]
    #: The list of all folders (subdirectories included as separate folders in the list) to fingerprint files with
in
log : Log
    #: The log object for the duplicate search
```

Outputs

```
log: log
    #: Results are stored in the log object
```

def GetFolderPairs(fpLog: DupeFpLog)

Description:

Given a duplicated file (passed as a DupeFpLog) obtain all possible pairs of folders that contain any of the duplicates. Uses itertools.combinations to find all pairwise combinations of unique folders that contain any of the duplicated files.

Args:

```
fpLog : DupeFpLog
    #: The DupeFpLog of this duplicated file
```

Outputs:

```
folderPairs : list[(Folder, Folder)]
    #: Dict containing list of base pair -> list of recursive parent pair scores
```

`def OrderDupeStorage(log)`

Description:

Sorts the DupeFpLogs based on the duplicated storage size Args:

```
log : Log
    #: The Log containing the duplicate results
```

Outputs:

```
sorted : list[DupeFpLog]
    #: The sorted list of duplicate logs
```

`def SizeClusterGen(iterable, threshold)`

Description

Generator that yields the clusters of files (based on size) within the specified threshold. Adapted from https://stackoverflow.com/a/15801233 (https://stackoverflow.com/a/15801233) Iteratively adds the next element in the list of sizes if the difference between next - previous is less than the threshold. Possible that this can generate large clusters if all sequential files are within the threshold - i.e 1000, 1999, 2998, 3997, 4997 ... would all fall within the same cluster with threshold 1000. In practice, a threshold of 1000bytes is sufficient to avoid this possibility with real file sizes ranging in the kilo to megabyte range.

Args:

```
iterable : list[int]
    #: The list of file sizes to cluster
threshold : int
    #: The size of the maximum difference between sequential elements in each cluster (in bytes)
```

Outputs

```
cluster : list[int]
    #: List of all sizes within the cluster
clusterIdx : int
    #: The index in the original unclustered list of files that correspond to each clustered size (for matching si
ze to file)
```

`def SortPairScores(log: src.fileorganiser.general.Log)`

Description:

Sort the list of folder pairs based on their similarity scores in descending order. Args:

```
log : Log
    #: The Log containing the duplicate results
```

Outputs:

```
sorted : list[(Folder, Folder)]
    #: The sorted list of folder pairs
```

def compareAllPairParents(basePairsToCheck, log: src.fileorganiser.general.Log)

Description:

Given a list of similar base pairs, recursively check the parent similarities of each pair using the recursePairParents function.

Args:

```
basePairsToCheck : list[(Folder, Folder)]
    #: The list of all starting pairs of folders
log : Log
    #: The log object with the duplicate file results
```

Outputs:

```
allPairsRecursiveSimDict : dict[(Folder, Folder), list[((Folder, Folder), int)]]
    #: Dict containing list of base pair -> list of recursive parent pair scores
```

def compareFolderPairs(folderPairs: list, log: src.fileorganiser.general.Log)

Description:

Given a list of folder pairs, calculate their similarity based on the intersection/union of the number of duplicated files.

Args:

```
folderPairs : list[(Folder, Folder)]
    #: List of pairs of folders
log : Log
    #: The log object with the duplicate file results
```

Outputs:

```
pairScores : list[((Folder, Folder), int)]
    #: List of pairs that contains the folder pair and their respective scores
```

def findDuplicatesMain(config, writeFile=False, cluster=True, wholeTree=True)

Description:

The main function used to search for duplicates, requires a config object, writeFile specifies whether the output should generate a file (old version) or simply return a list of lines to display elsewhere (django)

Args:

```
config : Config
    #: Duplicate search configuration object
```

Kwargs:

```
writeFile : bool : default=False
    #: Whether to output the results to a file or simply store as list of lines
cluster : bool : default=True
    #: Whether to apply pre size and extension clustering to reduce number of fingerprinted files
wholeTree : bool : default=True
    #: If true, only search within the specified included folders from the config/
```

Outputs:

```
fileSummaryLines : list[str]
    #: The lines for the summary of the duplicate search
fileResultLines : list[str]
    #: The lines with the list of duplicate file results
folderResultLines : list[str]
    #: The lines with the similar folder results
```

`def find_duplicate_fingerprints(log)`

Description

Given a list of all fingerprints this identifies any that appear more than once. A DupeFpLog instance is constructed for any duplicated fingerprints, using the fingerprintFiles dict to map the duplicated fingerprint to the corresponding files

Args:

```
log : Log
    #: The log object for this search run
```

Outputs

```
log : Log
    #: The log with populated results
```

`def fingerprintFile(file, log)`

Description

Fingerprint an individual file by calculating its 128-bit MD5 hash (represented as a 32-byte hex string). Fingerprints are stored on each file object and within the log.FpFilesDict which stores the fingerprint -> files with that fingerprint.

Args:

```
file : File
    #: The file to fingerprint
log : Log
    #: The log object for the duplicate search
```

Outputs

```
fileHash: str
    #: 32-byte hex string of the file fingerprint
```

`def fingerprintFolder(folder, log)`

Description

Fingerprint all files within the toplevel of the given folder. Fingerprints are stored on each file object and within the log.FpFilesDict which stores the fingerprint -> files with that fingerprint.

Args:

```
folder : Folder
    #: The folder to fingerprint files within
log : Log
    #: The log object for the duplicate search
```

Outputs

```
fpList: list[str]
    #: List of fingerprint strings within this folder
```

def recursePairParents(basePair, parentScoresList, log, threshold=1)

Description:

Given a similar base pair, recursively check the parent folder pair similarities. This function is called recursively and appends the next level up of parent scores up to the previous list of parent scores.

Args:

```
basePair : (Folder, Folder)
    #: The starting pair of folders
parentScoresList : list[((Folder, Folder), int)]
    #: The list of already calculated pair scores to append to
log : Log
    #: The log object with the duplicate file results
```

Outputs:

```
parentScoresList : list[((Folder, Folder), int)]
    #: The list of already calculated pair scores with the newly appended parent pair score
```

def store_duplicate_results(lines, outPath, iteration)

def writeDuplicateDirsResult(log: src.fileorganiser.general.Log)

Description:

Write the results of the folder similarity duplicate search. Writes list of folder pairs + similarity scores and a short summary.

Args:

```
log : Log
    #: The log object holding the results
```

Outputs:

```
summaryLines : list[str]
    #: List of summary lines
resultLines : list[str]
    List of result lines
```

def write_duplicate_result_lines(log, iteration)

Description:

Write the results of the file duplicate search. Writes list of duplicated files (from each DupeFpLog) and the duplicated storage space.

Args:

```
log : Log
    #: The log object holding the results
```

Outputs:

```
summaryLines : list[str]
    #: List of summary lines
resultLines : list[str]
    List of result lines
```

Classes

class DupeFpLog (Fp, log)

Class:

This class contains the information about a single duplicate fingerprint. Contains the fingerprint, a list of all files that have this fingerprint and the corresponding file sizes (+ duplicated storage occupied)

Description

Constructor for the DupeFpLog class that takes the duplicated fingerprint string and a reference to the main log object. Stores a list of all files with this duplicated fingerprint. Args:

```
fp : str
    #: The duplicated fingerprint
log : Log
#: The log object for the duplicate search
```

Outputs

```
allSizeExtClusters: list[list[File]]
    #: List containing all clusters
```

Index

Super-module

fileorganiser

Functions

Classes