Duplicate File Finder

Subimal Deb

Contents

About this program

Name Duplicate File Finder

Version : 0.1.nogui

Intention Generate a list of duplicate files with MD5 (hex)digests.

Author : Subimal Deb

Shared on `github.com`

Use on the command line.

Tested on Debian 6.0 with Python 2.6.6 (r266:84292, Dec 26 2010, 22:31:48).

Wishlist : Clean up code, add features, optimize, add a wxpython GUI.

Part I

For users

Part II

For developers

Chapter 1

The code

```
import hashlib, os, sys



def FileList(root):
    '''Generates the list of files in the list "root"
       (of directories) by walking through it and its
       subfolders.
    '''

    for root in roots:
        m=os.walk(root)
        for dirpath, subdirs, filenames in m:
            for name in filenames:
                fpath=os.path.join(dirpath,name)
                try:
                    fsize=os.path.getsize(fpath)
                    yield fpath
                except OSError:
                    pass

def GetMD5Sum(filename, chunksize=25600):
    '''Returns the md5 hash for the file "file" using
       a default buffer size of 25600 bytes.
    '''
    f = open(filename,'rb')
    md5sum = hashlib.md5()
    eof = False

    fname=filename.split('/')[-1]
    # TO DO : make this OS independent

    while eof==False:
        oldsum=md5sum.hexdigest()
        data = f.read(chunksize)
        md5sum.update(data)
        if oldsum==md5sum.hexdigest():
            eof = True
    f.close()
    return md5sum.hexdigest()

class FileObject():
    def __init__(self, filename):
        self.name = filename
        self.md5hex=None
    def hash(self):
        if self.md5hex is None:
            self.md5hex = GetMD5Sum(self.name)
```

```
def GetDuplicates(roots):
    # generate the file list
    result=[]
    for i in FileList(roots):
        result.append(FileObject(i))

    # generate the hashes
    map(lambda x: x.hash(), result)

    # generate the hash dictionary
    hashlist={}
    for i in range(len(result)):
        hashlist[result[i].md5hex]=[]

    # populate the hash dictionary with
    #  key   -> md5 hex digest
    #  value -> list of file names with the hex digest in key
    map(lambda x: hashlist[x.md5hex].append(x.name), result)


    # remove all the keys of length 1 (i.e., files with no duplicates)
    for each in hashlist.keys():
        if len(hashlist[each])==1:
            del(hashlist[each])

    # return a dictionary of with
    #   values : the lists of duplicate files
    #   keys   : the md5 hexdigest
    return hashlist

if __name__=='__main__':
    roots = ['/home/subimal/Music/' ]
    print GetDuplicates(roots)
```

Chapter 2

The Classes and Functions

2.1 The classes

2.1.1 FileObject

This class defines an object with the following attributes:

name It contains the file name of a `FileObject`

md5hex It contains the MD5 hex digest of the file. Initial value is `None`. Its value is set through the `hash()` attribute by invoking the function `GetMD5Sum`.

hash() This function will generate the MD5 checksum hexdigest (if the **md5hex** is `None`) by invoking the `GetMD5Sum` function.

What if the file was modified after the MD5 sum was generated? Compare the time when md5hex was assigned and the time of last access of the file?

2.2 The Functions

2.2.1 GetMD5Sum

Arguments

`filename` A *string* containing file's name including the full path.

`chunksize` (Optional). The buffer size for reading the file in parts. Default value is 25600 bytes.

Return value

The MD5 hexdigest for the file.

2.2.2 FileList

2.3