FIND AND PROMPT FOR REMOVAL OF DUPLICATE FILES

The find_dup_files.py program, could really use a more cleaver name. But it does what its name says. It is a simple, but useful program that scans a set of directory trees looking for duplicate files. When duplicate files are found, it lists them and prompts the user asking how to handle the set of files. The user may select to skip the set of duplicates, keep one of the files and delete the rest, delete all of the files, or exit the program.

This program will not likely be needed very often, but it can really help with cleaning up and organizing files that have accumulated over many years. I find the program especially useful with digital pictures, which have been maintained by several people over several years. Duplicate files can also crop up as files are copied between computers.

For the student wanting to learn Python programming via example, this program demonstrates the use dictionaries, lists, and various file and directory operations. The combined use of dictionaries and lists demonstrated here is a common strategy to search for the frequency of occurrence of values in a set of data.

USAGE

The program needs one or more directory names as arguments to begin searching for duplicates.

HOW IT WORKS

The program begins by evaluating each file by its file size. A dictionary containing lists of potential duplicates is built as the file system is scanned. From this dictionary, a list of lists is built for each set of one or more files that are the same size.

Next, the sets of potential duplicates are evaluated twice by calculating md5 hash values. In the first pass, we only process the first 1024 bytes of each file. The reason for this abbreviated pass is that many files may have the same size and calculating hash values is fairly slow. So, the first pass makes a quick determination to eliminate many non-duplicates. To be certain that the files are duplicates, the second pass compares each set by calculating the md5 hash value for the whole file.

AUTO GENERATED DOCUMENTATION

`find_dup_files.`**`first_hash`***`(file)`*

Returns md5 hash value for first 1024 bytes of file

`find_dup_files.`**`full_hash`***`(file)`*

Returns md5 hash value for all of the file contents

`find_dup_files.`**`scan_by_size`***`(directories)`*

Scan directories building a dictionary of lists from which a list of lists is extracted. Each sublist contains a set of file names with the same length, which is returned.

`find_dup_files.`**`filter_dupes`***`(potentialDupes, hash_func)`*

Loop through a list of lists containly potential duplicate files. Use the specified hash function to determine if they do appear to be duplicates. If a list of potential duplicates contains multiple sets of duplicates, a new list is generated for each.

find_dup_files.py CODE

```
#!/usr/bin/env python3
#
# Find and prompt for deletion of duplicate files in a file
system.
# This is especially useful when multiple people managing
digital pictures
# over several years on multiple computes has lead to duplicate
files.
#
# Original version by Bill Bumgarner. Refactored and ported to
Python 3
# by Tim Bower, May 19, 2010.
# Apache open source license
#

import os
import sys
import stat
import hashlib

def first_hash(file):
    "Returns md5 hash value for first 1024 bytes of file"
    try:
        aFile = open(file, 'rb')
    except IOError:
        return None
    hasher = hashlib.md5()
    hasher.update(aFile.read(1024))
    aFile.close()
    return hasher.digest()

def full_hash(file):
    "Returns md5 hash value for all of the file contents"
    try:
        aFile = open(file, 'rb')
    except IOError:
        return None
    hasher = hashlib.md5()
```

```
    while True:
        r = aFile.read(4096)
        if not len(r):
            break
        hasher.update(r)
    aFile.close()
    return hasher.digest()

def scan_by_size(directories):
    """ Scan directories building a dictionary of lists from
which a list of
    lists is extracted. Each sublist contains a set of file
names with the
    same length, which is returned.

    """
    dupes = []
    filesBySize = {}
    for dir in directories:
        print('Scanning directory "%s"....' % dir)
        for r,d,files in os.walk(dir):
            for f in files:
                file = os.path.join(r, f)
                if not os.path.isfile(file):
                    continue
                size = os.path.getsize(file)
                if size < 100:
                    continue
                if size in filesBySize:
                    filesBySize[size].append(file)
                else:
                    filesBySize[size] = [file]

    for file_set in list(filesBySize.values()):
        if len(file_set) > 1:
            dupes.append(file_set)
    del filesBySize
    return dupes
```

```
def filter_dupes(potentialDupes, hash_func):
    """ Loop through a list of lists containly potential
duplicate
    files.  Use the specified hash function to determine if
they do appear
    to be duplicates.  If a list of potential duplicates
contains multiple
    sets of duplicates, a new list is generated for each.

    """
    dupes = []
    for aSet in potentialDupes:
        hashes = {}
        for fileName in aSet:
            #print('Scanning file "%s"...' % fileName)
            hash_value = hash_func(fileName)
            if hash_value in hashes:
                hashes[hash_value].append(fileName)
            else:
                hashes[hash_value] = [fileName]

        for outFiles in list(hashes.values()):
            if len(outFiles) > 1:
                dupes.append(outFiles)
        del hashes
    return dupes

if __name__ == '__main__':
    if len(sys.argv) < 2:
        print('Usage: {} directory1 [other
directories]'.format(sys.argv[0]))
        sys.exit()
    print('Finding potential duplicates by file size...')
    dupes = scan_by_size(sys.argv[1:])
    print('Found %d sets of potential duplicate files' %
len(dupes))
    print('Filter with md5 hash of first 1024 bytes...')
    dupes = filter_dupes(dupes, first_hash)
    print('Found %d sets of potential duplicate files' %
len(dupes))
```

```
    print('Filter with full file md5 hash...')
    dupes = filter_dupes(dupes, full_hash)
    print('Found %d sets of duplicate files' % len(dupes))
    #
    # Pick files to keep
    #
    dupes.sort()
    for f_set in dupes:
        f_set.sort()
        print("Duplicate set:")
        cnt = 0
        for f in f_set:
            print("\t{} {}".format(cnt,f))
            cnt += 1

        pick = input("pick which to keep or n for none, s to
skip, q to quit: ")
        if pick.isdigit():
            p = int(pick)
            if p < cnt:
                del f_set[p]
            else:
                continue
        else:
            p = pick[0].lower()
            if p ==  'q':
                break
            elif p ==  's':
                continue
            elif p ==  'n':
                pass
            else:
                continue

        for f in f_set:
            print('Deleting %s' % f)
            try:
                os.remove(f)
            except OSError:
                print('Delete failed: {}'.format(f))
        print()
```