```
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.
    '''
    progressq="-\|/-"
    f = open(filename,'rb')
    md5sum = hashlib.md5()
    eof = False
    count = 0
    fname=filename.split('/')[-1]
    while eof==False:
        oldsum=md5sum.hexdigest()
        data = f.read(chunksize)
        md5sum.update(data)
        if oldsum==md5sum.hexdigest():
            eof = True
        count = count+1
    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)


roots = ['/home/subimal/Music/' ]

# 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)):
    if result[i].md5hex not in hashlist.keys():
        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])

# delete the list of "FileObject"s - no longer needed
del(result)

print hashlist
```