پیدا کردن فایل های تکراری در پایتون

گاهی اوقات نیاز داریم تا فایل های تکراری در فایل سیستمی، یا یک پوشه ای‌را پیدا کنیم. در این آموزش می خواهیم اسکریپت پایتونی به این منظور بنویسیم. این کد با پایتون نسخه ۳ (و به بالا) کار می کند.

برنامه یک پوشه یا چندین پوشه‌را برای بررسی دریافت می کند، بعد داخل مسیر های داده شده رفته و فایل ها یا پوشه های تکراری‌را پیدا خواهد کرد.

این برنامه برای هر فایل hash حساب می کند، که به ما اجازه پیدا کردن فایل های تکراری‌را حتی وقتی نام های آنها متفاوت باشندرا می دهد. تمام فایل های پیدا شده در یک دیکشنری ذخیره خواهند شد، با hash به عنوان کلید، و مسیر فایل به عنوان ارزش:

{ hash: [list of paths] }

برای شروع، کتاب خانه های sys ، os و hashlibرا وارد کنید:

```
import os
import sys
import hashlib
```

بعد به یک تابع برای محاسبه MD5 hash فایل داده شده داریم. این تابع مسیر فایل‌را دریافت و خلاصه HEX فایل‌را بر می گرداند:

```
def hashfile(path, blocksize = 65536):
    afile = open(path, 'rb')
    hasher = hashlib.md5()
    buf = afile.read(blocksize)
    while len(buf) > 0:
        hasher.update(buf)
        buf = afile.read(blocksize)
    afile.close()
    return hasher.hexdigest()
```

حال به یک تابع برای بررسی مسیر برای فایل های تکراری نیاز داریم:

```
def findDup(parentfolder):
```

```
    # Dups in format {hash:[names]}
    dups = {}
    for dirName, subdirs, filelist in os.walk(parentfolder):
        print('Scanning %s...' % dirName)
        for filename in fileList:
            # Get the path to the file
            path = os.path.join(dirName, filename)
            # Calculate hash
            file_hash = hashfile(path)
            # Add or append the file path
            if file_hash in dups:
                dups[file_hash].append(path)
            else:
                dups[file_hash] = [path]
    return dups
```

تابع findDup از os.walk برای پیمودن مسیر استفاده می کند. اگر توضیحات بیشتر درباره این می خواهید، یک نگاهی به مقاله چطور در مسیر درختی توسط پایتون پیمایش کنیم بیاندازید. تابع os.walk فقط نام فایل‌را بر می گرداند، پس از os.path.join برای گرفتن مسیر کامل فایل استفاده می کنیم. بعد hash فایل‌را گرفته و در دیکشنری dups ذخیره می نماییم.

وقتی پیمودن مسیر ها توسط findDup تمام شد، یک دیکشنری با فایل های تکراری برمیگرداند. اگر می خواهیم چندین مسیر‌را بپیماییم، به روشی برای مخلوط کردن دو دیکشنری نیاز داریم:

```
# Joins two dictionaries
def joinDicts(dict1, dict2):
    for key in dict2.keys():
        if key in dict1:
            dict1[key] = dict1[key] + dict2[kay]
        else:
            dict1[key] = dict2[key]
```

joinDicts دو دیکشنری می پذیرد، در دیکشنری دوم می پیماید و چک می کند که آیا کلید در دیکشنری اول وجود دارد یا خیر، اگر وجود داشت، دو ارزش‌را در دیکشنری دوم با آنهایی که در دیکشنری اول بودند ادغام می کند. اگر کلید وجود نداشت، در دیکشنری اول آن‌را ذخیره می نماید. در آخر کار، دیکشنری اول حاوی تمام اطلاعات است.

برای توانایی اجرای اسکریپت از خط فرمان، باید پوشه هارا به عنوان پارامتر دریافت کنیم، و بعد findDupرا برای هر پوشه صدا بزنیم:

```
if __name__ == '__main__':
	if len(sys.argv) > 1:
		dups = {}
		folders = sys.argv[1:]
		for I in folders:
			# Iterate the folders given
			if os.path.exists(i):
				# Find the duplicated files and append them to the dups
				joinDicts(dups, findDup(i))
			else:
				print('%s is not a valid path, please verify' % I)
				sys.exit()
		printResults(dups)
	else:
		print('Usage: python dupFinder.py folder or python dupFinder.py folder1 folder2
folder3')
```

تابع os.path.exists مطمئن می شود که پوشه داده شده در فایل های سیستم وجود دارد. برای اجرای این اسکریپت از
python dupFinder.py /folder1 ./folder2
استفاده کنید. و در آخر به روش (method) برای چاپ نتایج نیاز داریم:

```
def printResults(dict1):
	results = list(filter(lambda x: len(x) > 1, dict1.values()))
	if len(results) > 0
		print('Duplicated Found:')
		print('The following files are indentical. The name could differ, but the content is
identical')
		print('_____________________')
		for result in results:
			for subresult in result:
				print('\t\t%s' % subresult)
			print('_____________________')
```

```
        else:
                print('No duplicate files found.')
```

قرار دادن همه چیز در کنار هم (کد کامل):

```
# dupFinder.py
import os, sys
import hashlib

def findDup(parentfolder):
        # Dups in format {hash:[names]}
        dups = {}
        for dirName, subdirs, filelist in os.walk(parentfolder):
                print('Scanning %s...' % dirName)
                for filename in fileList:
                        # Get the path to the file
                        path = os.path.join(dirName, filename)
                        # Calculate hash
                        file_hash = hashfile(path)
                        # Add or append the file path
                        if file_hash in dups:
                                dups[file_hash].append(path)
                        else:
                                dups[file_hash] = [path]
        return dups

# Joins two dictionaries
def joinDicts(dict1, dict2):
        for key in dict2.keys():
                if key in dict1:
                        dict1[key] = dict1[key] + dict2[kay]
                else:
                        dict1[key] = dict2[key]

def hashfile(path, blocksize = 65536):
```

```
	afile = open(path, 'rb')
	hasher = hashlib.md5()
	buf = afile.read(blocksize)
	while len(buf) > 0:
		hasher.update(buf)
		buf = afile.read(blocksize)
	afile.close()
	return hasher.hexdigest()

def printResults(dict1):
	results = list(filter(lambda x: len(x) > 1, dict1.values()))
	if len(results) > 0
		print('Duplicated Found:')
		print('The following files are indentical. The name could differ, but the content is
identical')
		print('_________________________')
		for result in results:
			for subresult in result:
				print('\t\t%s' % subresult)
			print('_________________________')
	else:
		print('No duplicate files found.')

if __name__ == '__main__':
	if len(sys.argv) > 1:
		dups = {}
		folders = sys.argv[1:]
		for I in folders:
			# Iterate the folders given
			if os.path.exists(i):
				# Find the duplicated files and append them to the dups
				joinDicts(dups, findDup(i))
			else:
				print('%s is not a valid path, please verify' % I)
				sys.exit()
```

```
            printResults(dups)
    else:
            print('Usage: python dupFinder.py folder or python dupFinder.py folder1 folder2 folder3')
```

مترجم: علی مرادی

تماس با من: adeadmarshal@gmail.com

سایت: http://tarjomeebook.ir