CASE STUDY

5

Creating a Hash File (or a message digest of a file)

Hashing is the process of transforming a string of characters of arbitrary length into a usually shorter fixed-length string that represents the original string. The output of the hash function is called *message digest*. Hashing is used in many encryption algorithms.

The hash or message digest is generated by a mathematical formula in such a way that it is extremely unlikely that some other text will produce the same hash value. Hashes are very important in security systems. They ensure that transmitted messages have not been tampered with. Consider the steps given below which helps to ensure that the message is not modified during transmission.

Step 1: The sender creates a hash of the message.

Step 2: The hash is encrypted using an encryption algorithm.

Step 3: The sender transmits the hash as well as the original message.

Step 4: The receiver receives the message and hash and decrypts both the message and the hash.

Step 5: The sender takes original message and again generates another hash from the received message.

Step 6: If the received hash is same as hash generated, indicates that the message was transmitted intact.

Thus, we see that hashing is widely used in cryptography. There are many hashing functions like `MD5`, `SHA-1`, etc. However, we will create hash using the `SHA-1` hashing algorithm which generates a hash value which is `160` bits long.

In the program given below, instead of taking the entire file all at once, we have taken chunks of data. This is especially important when files are very large to fit in memory all at once. Processing data in small chunks (of `1024` bytes, here) makes efficient utilization of memory. The file to be hashed has been opened in the read in binary mode as binary files are more efficient than text files. In Python, hash functions are available in the `hashlib` module. The file is read in the `while` loop and on reaching the end, an empty byte is obtained. The program finally prints the message digest in hexadecimal representation using the `hexdigest()` method.

```
# Program to create a hash file

import hashlib
def hash_file(filename):
   hash = hashlib.sha1()        # make a hash object
   with open(filename,'rb') as file:
       chunk = 0
       while chunk != b'':
           chunk = file.read(1024)
           hash.update(chunk)
    return hash.hexdigest()
```

```
text = hash_file("Body.txt")
print("Hash of file is : ",text)
```

OUTPUT

```
Hash of file is :  24134bdf497ce78a0903dfdb69d0019283faa8c3
```

CASE STUDY

6

Mail Merge Program

You must have already tried the Mail Merge feature of MS Word which is used to send the same letter to a large number of people. With this feature, you just have to type the contents that has to be sent to a number of people in one file. In another file, type the names of all the receivers of the email. Then merge both these files in such a way as if it was specifically written for an individual. Consider the files given below which have been used to illustrate how mail merge is practically realized through Python.

Note that to perform mail merge, we have created three files—`Names.txt` that stores names of the receivers, `Body.txt` that stores the content or body of the mail (or message) to be sent and the main program file which is basically a Python script that opens both the files and merges them.

In the code for the mail merge, we open both the files in reading mode and iterate over each name using a `for` loop. New files with name `"[Name].txt"` are created, where `[Name]` is the name of the receiver as specified in the file storing all the names of the receivers. The `strip()` method has been used to clean up leading and trailing whitespaces. This is especially important because while reading a line from the file, the newline `'\n'` character is also read. Finally, the `write()` method is used to write the body (contents of the mail) into the new `[Name].txt` files.

```
# Contents of Names.txt
Reema
Goransh

# Contents of Body.txt
Greetings !!!
This is to invite you to attend the National Conference at IIT Delhi on 29th August
2017.

Looking forward for your participation.

Registration Fess : Rs. 1000

Thanks and Regards,

Conference Convener

# Program for Mail Merge

with open("Names.txt",'r') as Names:
    with open("Body.txt",'r') as Body:
        text = Body.read()
        for name in Names:
```

```
        msg = "Hello "+ name + text
        with open(name.strip()+".txt",'w') as File:
            File.write(msg)
```

OUTPUT

```
#Contents of Reema.txt
Hello Reema
Greetings !!!
This is to invite you to attend the National Conference at IIT Delhi on 29th August
2017.

Looking forward for your participation.
Registration Fees : Rs. 1000
Thanks and Regards,
Conference Convener

#Contents of Goransh.txt
Hello Goransh
Greetings !!!
This is to invite you to attend the National Conference at IIT Delhi on 29th August
2017.
Looking forward for your participation.
Registration Fees : Rs. 1000

Thanks and Regards,

Conference Convener
```

CASE STUDY

7

Finding Resolution of an Image

JPEG (Joint Photographic Experts Group) is one of the most widely used compression techniques for image compression. Most of the image file formats have headers stored in the initial few bytes to retain some useful information about the file.

In the following program, we will find out the resolution of JPEG image by reading the information stored in the header.

```
# Program to find the resolution of an image

def find_res(filename):

    with open(filename,'rb') as img_file:     # open image in binary mode
        # height of image is at 164th position
        img_file.seek(163)
        # read the 2 bytes
        a = img_file.read(2)
        # calculate height
        height = (a[0] << 8) + a[1]
        # read next 2 bytes which stores the width
        a = img_file.read(2)
        # calculate width
        width = (a[0] << 8) + a[1]
    print("IMAGE RESOLUTION IS : ",width,"x",height)
find_res("C:\Python34\Icon.jpg")
```

OUTPUT

```
IMAGE RESOLUTION IS :  4352 x 769
```

In this program, we opened the image in binary mode as all non-text files must be open in this mode. In a JPEG file, the height of the image is stored in the header at 164th position followed by width of the image. Both this information are two bytes long. This two bytes information is converted into a number using the bitwise shift operator (<<) and finally, the resolution of the image is displayed.

Note The above program will run only for JPEG images as every file format uses a slightly different way to store the same information.