Project 1: Password Cracking Program Python

Name: Shefali Athavale

Worked with: Roshan Nellore Prasad

Approach:

For this project, I wrote a simple password cracking program using Python. I used the following libraries for the same - sys, hashlib, time. The program asks the user for a MD5 hash of the password. It then asks the user to enter the method of password cracking. The two methods listed as options are 'Brute Force' Method and the 'Dictionary' Method. Once the user selects the method, the program then executes the code for the respective method and outputs the cracked password, attempts required to crack it and the time required on the screen.

Brute Force Method:

In this method, we have assumed that the maximum length of password is 6 characters consisting of only lowercase letters or uppercase letters or digits and not a combination of any of the two. The program starts iterating through the characters(in our case only lowercase letters) one by one and generates the possible password. The generated password is then hashed using MD5 hash and checked with the hash inputted by the user. If it matches, the program breaks from the loop and outputs the cracked password along with attempts required to crack it and the time required on the screen. The program outputs "Password not found" if the password can't be cracked.

Dictionary Method:

In this method, the program is given a MD5 hash of the password along with a wordlist of all common passwords. The program iterates through the wordlist, calculates the MD5 hash and checks if it matches with the MD5 hash inputted by the user. If it does match, we have cracked the password and the program breaks from the loop. It outputs the cracked password along with attempts required to crack it and the time required on the screen. If a password is not found, the program will output 'Password not found'.

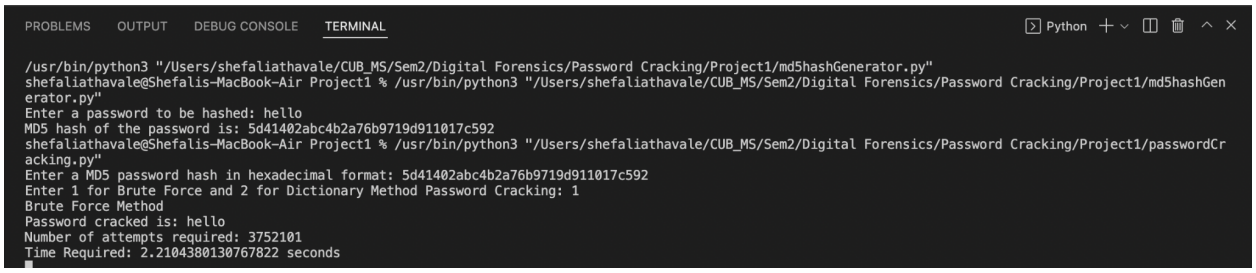
To test the working of this approach, I have used a common wordlist 'rockyou.txt' which is easily available. The program takes the wordlist as a command line argument. I have used only '.txt' wordlists, so I am not sure if it works for other types of files too.

The code is attached below along with the screenshots of the output screen. I have written another program 'md5hashGenerator.py' which just generates the MD5 hashes for the plain text password. These hashes are then inputted to 'passwordCracking.py' which cracks the password. The screenshots have outputs of both the programs - 'md5hashGenerator.py' and 'passwordCracking.py' which is our main program.

Output:

Brute Force Method:

1)



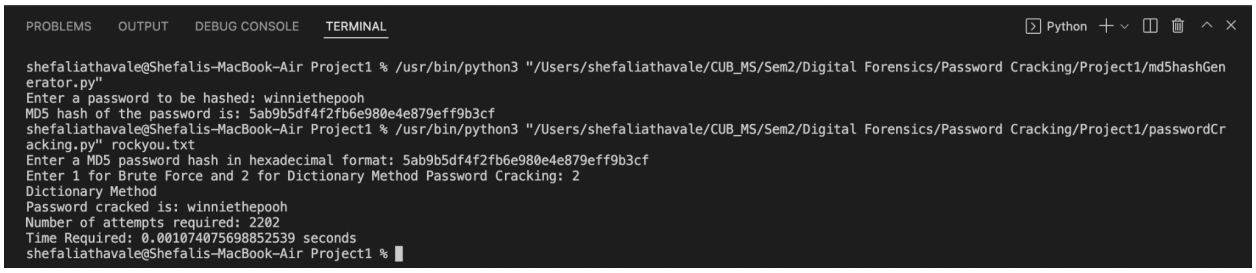
2)

Dictionary Method:

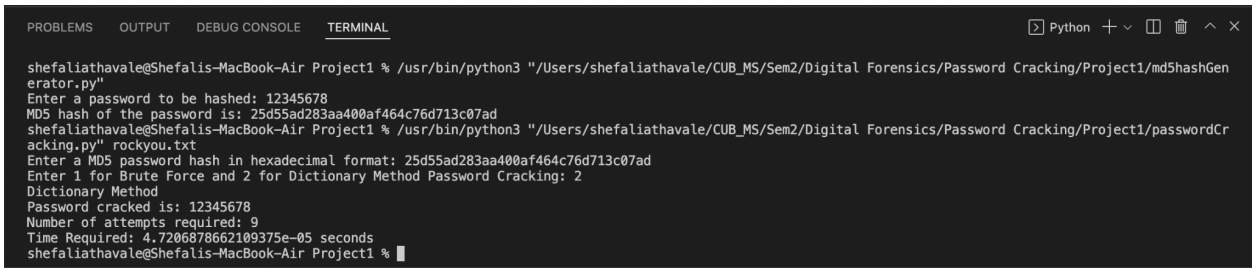
Wordlist used to test: ‘rockyou.txt’

1)

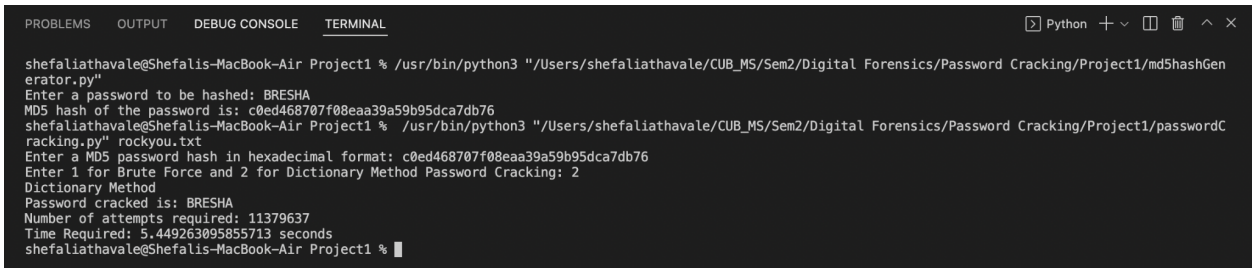
2)



3)



4)



Source Code:

To try Brute Force Method on uppercase letters or digits, comment lines 17-18 and uncomment lines 21-22 or 25-26 respectively.

To try the Dictionary Method with an inbuilt wordlist which is coded in the code itself, comment lines 72, 76-82, 117-120 and uncomment lines 71, 84, 121.

Source code for both the methods is attached below. Copy pasting this code in an editor will keep the indentation intact.

```
# ----Password Cracking using Brute Force method and Dictionary Attack Method----
# To try Brute Force Method on uppercase letters or digits,
# comment lines 17-18 and uncomment lines 21-22 or 25-26 respectively.
# To try the Dictionary Method with an inbuilt wordlist which is coded in the code
itself,
# comment lines 72,76-82,117-120 and uncomment lines 71,84,121. Type
'Project1ShefaliAthavale.py rockyou.txt' in the command line to run the program for
dictionary method

import sys
import time
import hashlib

def bruteForce(pwdHash):
   print("Brute Force Method")
```

```
    n = len(pwdHash)

    # Creates a list of lowercase characters to iterate through
    char = "abcdefghijklmnopqrstuvwxyz"
    chars = list(char)

    # chars1 is the list of uppercase characters to use if needed
    # char1 = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    # chars = list(char1)

    # digits1 is the list of numbers to use if needed
    # digits = "0123456789"
    # chars = list(digits)

    # Counter to keep track of number of attempts
    c = 0
    guessPwdHash = ""
    pwd = ""
    flag = 0
    arr1 = chars.copy()
    start = time.time()
    # Functionality to check passwords for length upto 6
    for i in range(2,7):
        # Temporary list to store passwords generated for ith iteration which can be
used for (i+1)th iteration
        tmp = []
        for j in chars:
            # Checks for password of lenth 1
            if(hashlib.md5(j.encode()).hexdigest()==pwdHash and flag!=1):
                end = time.time()
                print("Password cracked is: "+str(j))
                print("Number of attempts required: "+str(c))
                print("Time Required: "+str(end-start)+" seconds")
                # return (pwd,c,(end-start))
                flag = 1
                break
            else:
                # Checks for passwords of length 2-6
                for k in arr1:
                    pwd = j+k
                    pwd = str(pwd)
                    # Creates a MD5 hash of possible password combination
```

```
                    guessPwdH = hashlib.md5(pwd.encode())
                    guessPwdHash = guessPwdH.hexdigest()
                    c+=1
                    if(guessPwdHash==pwdHash):
                        end = time.time()
                        print("Password cracked is: "+str(pwd))
                        print("Number of attempts required: "+str(c))
                        print("Time Required: "+str(end-start)+" seconds")
                        flag = 1
                        break
                    tmp.append(pwd)
        arr1 = tmp
    if(flag==0):
        print("Password not found")


# def dictMethod(pwdHash):
def dictMethod(dictFile,pwdHash):
    print("Dictionary Method")
    # Counter to keep track of number of attempts
    c = 0
    pwdList1 = []
    # Creates a list from the file taken as command line input from user
    for line in dictFile:
        strippedLine = line.strip()
        lineList = strippedLine.split()
        if(lineList):
            pwdList1.append(lineList[0])
    # Uncomment for inbuilt wordlist
    # pwdList1 = ['password', 'iloveyou', 'princess', '1234567', 'rockyou', '12345678',
'abc123', 'nicole', 'babygirl', 'qwerty']
    start = time.time()
    for pwd in pwdList1:
        # Creates a MD5 hash of possible password combination from the passwords
available in the wordlist
        guessPwdH = hashlib.md5(pwd.encode())
        guessPwdHash = guessPwdH.hexdigest()
        c+=1
        if(guessPwdHash==pwdHash):
            end = time.time()
            print("Password cracked is: "+str(pwd))
            print("Number of attempts required: "+str(c))
```

```
            print("Time Required: "+str(end-start)+" seconds")
            flag = 1
            break
    if(flag==0):
        print("Password not found")


if __name__ == "__main__":

    # Takes MD5 hash of user password from the user
    pwdHash1 = input("Enter a MD5 password hash in hexadecimal format: ")
    # Takes the type of attack the user wants to perform
    typeAttack = input("Enter 1 for Brute Force and 2 for Dictionary Method Password
Cracking: ")
    if(typeAttack=="1"):
        #Brute Force Method
        bruteForce(pwdHash1)

    elif(typeAttack=="2"):
        #Dictionary Method
        # filename = "rockyou.txt"
        # file = open(filename,"r")

        filename = sys.argv[1]
        # File encoding is given because of a decodeError while reading 'rockyou.txt'
        file = open(filename,"r",encoding="ISO-8859-1")
        dictMethod(file,pwdHash1)
        # dictMethod(pwdHash1)

    else:
        print("Invalid Choice")
```