Bonus Assignment Week 4

This week we are going to write a program that guesses passwords.

Two files are provided:

- `words.txt` **- this file contains a list of English words, one word per line**
- `passwordfile.txt` **- this file contais a list of username:passwordhash combinations, one per line.**

The password file

A well-known way to authenticate users is through the use of a username and password. If the user can provide the correct password, the user is granted access.

To check the password, the system uses a *password file*, in which known username, password (both strings) combinations are stored, one combination per line. However, for security reasons, passwords are not stored in plain-text, but an encrypted version (called a *hash*) of the password string is stored instead. Hashes are computed using the function `encrypt_password()`, which is provided below. This way, if the password file is stolen, the passwords are not immediately visible.

For example, if username "John" has password "secret", then instead of:

```
John:secret
```

the following is stored in the password file (see also below):

```
John:2bb80d537b1da3e38bd30361aa855686bde0eacd7162fef6a25fe97bf527a25b
```

When John wants to login into the system, then John provides his username and plaintext password. Then the system uses the `encrypt_password()` function to compute the hash of the provided password and compares the result to what it finds in the password file for the user John. If both strings match each other, John is granted access, otherwise not.

In [1]:

```
##
## The encrypt_password function.
## You can use this function in your answer.
##

import hashlib

def encrypt_password( passwd ):
    """Encrypt a plaintext password (a string). It returns the result.
    This encryption is one-way only, meaning it is not easy (impossible) to decrypt
    the encrypted password to find out the original plaintext password again."""
    return hashlib.sha256( passwd.encode() ).hexdigest()
```

In [2]:

```
## Some examples of using the encrypt_password function
encrypt_password("secret")
```

Out[2]:

```
'2bb80d537b1da3e38bd30361aa855686bde0eacd7162fef6a25fe97bf527a25b'
```

In [ ]:

```
encrypt_password("assignment")
```

In [ ]:

```
encrypt_password("#@SS1GNMENT!")
```

Hacking (finding) the passwords

Unfortunately, some users use passwords that are simply existing words or simple variations on them. A simple variations on words are to:

- **replace each instance of the letters i and I (small and capital i) with the number 1 (one)**
- **replace each instance of the letters a and A (small and capital a) with the character @**
- **add a hash sign (#) at the beginning of the word**
- **add an exclamation mark (!) at the end of the word**
- **capitalize all letters**

For example, a user might use the password "#@SS1GNMENT!" which is a variation on the word "assignment".

You are going to write a program that uses a file containing a long list of possible passwords to try to find the plaintext passwords of users given a password file with encrypted passwords.

***Hint*: Make sure to break up your code in smaller parts (functions) and to first think about the solution (use paper!) before you start coding!**

***Hint*: You may assume that a user either uses a word unaltered from the words file as a password or uses *all* 5 ways to alter an existing word from the file simultaneously as a password (so either "assignment" or "#@SS1GNMENT!").**

***Hint*: You can use the provided The *encrypt_password()* function in your answer.**

Assignment 4.1

Write a function that takes as argument a string and returns a new string in which the argument string has been altered using the 5 mentioned alterations. For example, "assignment" should be changed into "#@SS1GNMENT!".

Hint: You migth want to use the [string.replace()] method in your answer.

In [3]:

```
def alter_string(wrd):
    for i in range(len(wrd)):
        if wrd[i] in ["a","A"]:
            wrd=wrd[:i]+"@"+wrd[i+1:]
        if wrd[i] in ["i","I"]:
            wrd=wrd[:i]+"1"+wrd[i+1:]
    wrd="#"+wrd.upper()+"!"
    return (wrd)

alter_string("assignment")
```

Out[3]:

```
'#@SS1GNMENT!'
```

Assignment 4.2

Write a program that reads the file `words.txt` and that prompts a user for an encrypted (hashed) password and that finds and prints the plaintext password (if it can be found) of the encrypted password. It should work for encripted versions of words from the read file, as well as for their altered version. For example:

```
Input encrypted password:
b46352779bececb48e5a8d31e58684e8da139a944715bd44f0b930f3ac46bbca
Plaintext password is: #@SS1GNMENT!
```

Another example is:

```
Input encrypted password:
ce7a7c10b0dfd96808cca64c88cf5c5e13b7775283bdc924767887bfa32c8fa1
Plaintext password is: assignment
```

Note: make sure to use a dictionary data-type in your solution

In [4]:

```
f=open("words.txt","r")
fline=f.readlines()

lst=[]
for line in fline:
    words = line.split()
    lst = lst + words

code_dictionary=dict()
for word in lst:
    code_dictionary[encrypt_password(word)] = word
    code_dictionary[encrypt_password(alter_string(word))] = alter_string(word)
```

In [5]:

```
a=input("Input encrypted password:")

if a not in code_dictionary:
    print ("There is no matching plaintext password.")
else:
    for word in code_dictionary.values():
        if code_dictionary[a] == word:
            print ("Plaintext password is:",word)
```

```
Input encrypted
password:b46352779bececb48e5a8d31e58684e8da139a944715bd44f0b930f3ac46bbca
Plaintext password is: #@SS1GNMENT!
```

Assignment 4.3

Write a program that reads the password file `passwordfile.txt` and for each line in the password file prints out the name and corresponding plaintext password, if you can find it (if not, print something appropriately). For example, the output could be:

```
John    secret
Mary    #@SS1GNMENT!
Bob     Cannot find password, too complex!
Jane    python
```

```
    --
Peter   #PYTHON!
...
```

In [6]:

```
f2=open("passwordfile.txt","r")
fline2=f2.readlines()

lst2=[]
for line in fline2:
    words = line.rstrip().split(":")
    lst2 = lst2 + words

for i in range(0,len(lst2),2):
    if lst2[i+1] not in code_dictionary:
        print (lst2[i],"\t","Cannot find password, too complex!")
    else:
        print (lst2[i],"\t",code_dictionary[lst2[i+1]])
```

```
John    #@@RDV@RK!
Mary    #FLEDGL1NG!
Bob    Cannot find password, too complex!
Jane    python
Peter    #PYTHON!
Julia    programmer
Mike    #UN1VERS1TY!
Alice    #ELECT1ONS!
Zach    zombies
Vicky    #TUTOR1@L!
```

In [ ]:

```
```