algorithmica

Programming Language Learning Series
Mastery of Python Language
(Password File Cracker)

Over the past several decades computer systems have relied on the concept of a password in order to authenticate users and grant access to secure systems. I hope you all are using several different passwords for different systems and websites as that is currently the best security practice. However even when requiring users to use “strong passwords” and change them frequently, passwords inevitably get leaked and people unfortunately use the same password for their bank account as for Facebook.

There are a vast array of methods for cracking passwords but in this project, you will be introduced to 2 of the possible methods(Brute force & Dictionary driven methods) and you have to implement those methods.

BruteForce Password Cracker: A brute force attack is a computer program will generate every possible combination of input then try all possibilities. Take a smartphone as an example, most have a 4 digit pass code so a brute force attack would start by first trying 0000, 0001, 0002, 0003, 0004, so on and so forth, until the correct passcode is found. In the case of a 4-digit password, there are 10,000 combinations. When using all the characters on your keyboard, the possible combinations quickly climb into the millions.

This function takes your password protected zip file as arguments and tries to crack the password using dictionary. In this function, you have to write a brute force logic that generates a strings containing a-z (all lowercase) of length 8 or less as the password to attempt (Hint: start small trying small passwords first). For example, your string should start out with “a”, then “b”, then “c”, so on and so forth. When you get to “z” change the first character back to “a” and append another letter so your string becomes “aa” then “ab”. Once the password is cracked you should display the password to the user.

Dictionary Based Password Cracker: A dictionary attack relies on a source in order to carry out an attack. A source file is simply just a text file containing passwords. Sometimes, when malicious hackers manage to break into a “secure” system they release a *password dump* which is a file containing all the passwords found on that system. This is a good way of seeing the most common passwords. This function takes your zip file and dictionary file as arguments and tries to crack the password using dictionary. Once the password is cracked you should display the password to the user. If you have finished trying out all the passwords in the source file, display an appropriate error message.

algorithmica

Programming Language Learning Series
Mastery of Python Language
(Password File Cracker)

Implementation Hints

1. ***Working with Zip files:*** To properly open zip files in python you need to import the `zipfile` module. Use the following function to open zip file:
   ```
   zip_file = zipfile.ZipFile(filename)
   ```

   To apply your `password` for opening and extraction of opened zipfile, use the following function:
   ```
   zip_file.extractall(pwd=password.encode())
   ```

   The type of the variable `password` is string and `encode()` is a string method that you must apply to the `password` variable when used as an argument to `extractall`. Basically, you try to unzip the file using `extractall`: if it generates an error, the password didn't work; if it doesn't generate an error, the password succeeded. Therefore, you want to put the call to `extractall` within *try-except*. Unfortunately, a variety of errors get generated by `extractall` so you must use `except` with no error type specified (that is generally not good practice, but necessary in this case). Therefore, your `except` line is simply `except:`

2. ***Working with itertools:*** To brute force attack passwords you want to try all permutations of characters. The `itertools` has a function named `product` that provides exactly that functionality. However, that function generates tuples of characters, e.g. `('a','b','c')`, whereas we need a string, e.g. `'abc'`. There is a string method named `join` that allows you to join characters together with a specified character between each one. For example, `'-'.join(('a','b','c'))` yields the string `'a-b-c'`, but we don't want any characters in between so we use an empty string as `''.join(('a','b','c'))` to yield `'abc'`. For example, if we want to generate strings (passwords in this project) of length 3 from a string of characters `'abcdef'` we use the following:
   ```
   from itertools import product
   for items in product('abcdef',repeat=3):
       print(''.join(items))
   ```

3. Use `strip()` on the word read from the dictionary file before trying it as a password