name: Linus R.

period:3

2.1.3 - Passwords

This is a good one. It's about passwords. What's the big deal about making a strong password? Well, it is a big deal, because weak passwords cannot only compromise your own data but also an entire network. Why? because a network is like a chain: and a chain is as strong as its weakest link. You don't want anyone to get into your network.

Let's be clear about the distinction between using brute force and solving a key/encryption algorithm. Think of the algorithm as a function -- an input/output machine. Solving the key is trying to figure out what the function is, so that you can then essentially have access to a working key. Brute force doesn't concern itself with cracking the algorithm. It is more concerned with a more general meaning of the word key. It wants to find the input which produces a certain output; in the case of passwords, this means finding the password which allows access into an account. Brute force tries every possible password until it finds the one that works.

This lesson has some good intro material, so please read it through.

1. Please develop the habit of looking at the code provided in the Trinket to see if you can figure out what the code is going to do before you run it. And also note any questions about how certain lines of code work so that you can ask me and I can give you a response on this doc.

Look through pw_analyzer (which is loaded in the main). What lines of code do you have some uncertainty about? (paste in the whole program and provide the line numbers).

none

Look through pwalgorithms. What lines of code do you have some uncertainty about?

none

2. In the pwalgorithms module:

a) Review the one_word function and describe how it analyzes a one-word password.

It analyzes a one-word password by looping through all of the words in the dictionary file.

b) Review the get_dictionary function. This function returns a list called words. What is in this list?

Every word in the dictionary file

3-4.

Test out the one_word function on koala and three different one-word passwords (maybe one from the beginning of the list of passwords, one from the middle, one from the end, and one not on the list). Record your results here:

1-word password	Guesses	time
koala	15183	0.01333332
apple	400	0.00406122
xylophone	Not found	Not found in 0.00901222
lime	15716	0.00733590

5-9. Paste in your definition of the two_words function you made here:

```
def two_words(password):
    words = get_dictionary()
    guesses = 0
    # get each word from the dictionary file
    for w in words:
        for a in words:
            guesses += 1
            if (w + a == password):
              return True, guesses
    return False, guesses
```

Run your program for the the two-word password for kookykoala and 2 other tests made up of words that would be found in the middle of your words list. Record your results here.

2-word password	Guesses	time
kookykoala	401809261	31.40547228
applebanana	10552120	0.83345890
wateryawn	20918646	1.67088795

10-11. Paste in your definition of the two_words_and_digit function you made here:

```python
def two_words_and_digit(password):
    words = get_dictionary()
    numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
    guesses = 0
    # get each word from the dictionary file
    for w in words:
        for a in words:
            for n in numbers:
                guesses += 1
                if (n + a + w == password):
                    return True, guesses
    return False, guesses
```

Run your program for the the two-word password and digit for 3 tests made up of words that would be found toward the beginning of the words list. Record your results here.

2-word-and-digit password	Guesses	time
1applebanana	43639892	4.55705667

2bananaapple	105521193	10.99258161
3applecarrot	490312834	53.10299659

12-13.

Algorithm	**Average Time**	**Times Longer Than Previous Analysis**
One word	.01 seconds	-
Two words	40 seconds	4,000
Two words and a digit	1,200 seconds or 20 minutes	30
Two words, a digit, and a symbol	36000 seconds	30
Three words (no digits, numbers)	144000000 seconds	4000
Three words and a digit	4320000000 seconds	30

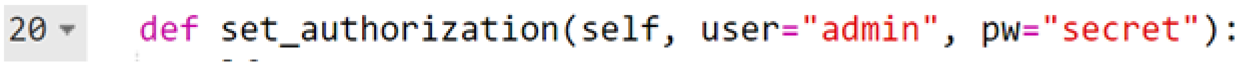
14. Why do they say passphrases better than random characters? Do you agree? Why or why not?

They say they are better than random characters because it is easier to remember. You can remember a really long passphrase made of words, but only a few random characters tied together. I agree, because you can't use a password if you can't remember it.

Multi-factor Authentication

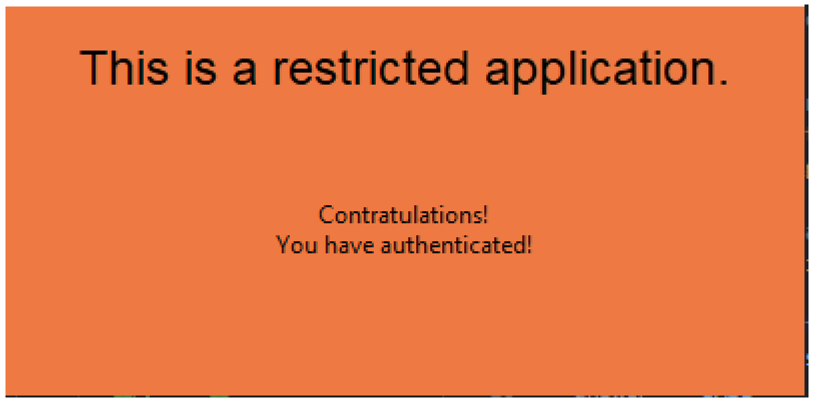
15-19. PLTW points out that in order to be a true module that can be accessed by many different programs, mutlifactorgui.py should not be altered. Make changes to the attributes of your MultiFactorAuth object mfg in main.py. Paste a snip of the lines that did this.

Hint: Notice this line from the gui program. You do NOT want to change this, because you are not to make any changes to the gui program. Notice this has three parameters. `self` is a parameter used in all methods that act on an object -- basically, don't worry about it. Just know that there is not a corresponding argument at the method call for this parameter. `user` and `pw` are both parameters that have default values. If a user chooses not to provide an argument for `user` at the call of this method, "admin" will be assigned to `user`. If a user chooses not to provide an argument for `pw` at the call of this method, "secret" will be assigned to `pw`. So the key to resetting the username and password is to call this method in the multifactor program passing your own username and password as argument values on the call.



```
    def set_authorization(self, user="admin", pw="secret"):
```

Paste a snip of the output that proves to me you authenticated and accessed the restricted application.



This is a restricted application.

Contratulations!
You have authenticated!

Conclusion

1. Why are individual security measures important to the entire internet?

Because a chain is only as strong as it’s weakest link, and if you work at a big company, for example, and you have a bad password, somebody could get into that and hack the company.

2. Give at least two reasons why it is important to have multi-factor authentication as opposed to just having a strong password.

It is important because even if they crack your password, you still have another layer of authentication for them to get through.

Optional Challenges: (only if you have time!)

If you take CS A you will see why these are not very strong algorithms for searching for passwords. It doesn't matter with the one-word search, but when you have to continually look through the words list it makes a very big difference.

If you can improve on the times for the two-word and two-word-and-digit algorithms, run tests using the same test case words you used previously and show the results of the times with your new algorithms. (See if you can make the 2-word search go 1,000 times faster than the one they suggested!)

Also paste the function definitions you created here, as well.