Week 4 Exercises

This week we've covered alot! A strong grasp of these topics will be essential for you in future as control and logic underpin all program structure

Disclaimer What we're doing here is not what big companies do when they store your data!

Exc. 1 (and): Multiple Users

(This exercise is similar to the *extra* part of last week's summary.)

Write some code which asks a user to "login". This means, they will need to provide a correct username **and** password. * You should have multiple possible username-password combinations which can be "logged into" * Usernames should *not* be sensitive to capital letters

Exc. 2 (lists & in): Bad Password Warning

Let's make our system more slick by using what we know about lists.

Note: Add this functionality to your solution for the previous exercise.

2.a Check for bad passwords

Create a list containing what you'd consider bad passwords (e.g. "password", "1234" etc.). When a user has succesfully logged in, if their password is a poor one, then tell them so

2.b

Store user details in lists. * Create two lists, one containing usernames and the other containing passwords * A user logs in succesfully if they enter a username-password pair corresponding to the i*th* usernames in the username list, and the i*th* password in the passwords list

2.c

Reset bad passwords * If a user logs in and their password is identified as a bad password, give them the **option** to change it (i.e. change the appropriate entry in the passwords list)

Exc. 3 (While):

3.a Have mercy on the user

Using a while loop allow the user to keep entering their password until they get it right * Do this without breaking all the functionality you implemented above!

3.b That was too much mercy

Limit the number of attempts the user gets.

3.c (Extra) Still too much

We don't want russian hackers breaking through! Enforce an increasing time penalty by using time.sleep(time_to_sleep)

[you will need to add **import time** to the top of your code - we'll teach you about this later]

Reading Week Challenges

These challenges are tricky and require lots of clever steps! Don't feel disparaged if you don't manage to finish them - they are fairly mathematical, and it takes a lot of practice to develop the patterns of thought which computer scientists use to tackle such glorified logic puzzles.

However, we **strongly** recommend that you ask friends, or even us (contact us via the facebook hacksoc group) before you look at the worked solutions - sometimes people are closer to the solution than they realise and all they need is a few helpful hints (google "rubber duck debugging" ;))

Challenge 1 (Modulu) - Fizz Buzz

This is a classic problem which you'll probably encounter many times if you learn other programming languages in future.

Problem Statement

Given an integer N (provided by the user), print out all the integers from 0 to N. However, you must replace numbers as follows: * If a number is a multiple of 3, then print **Fizz** instead of the number * If a number is a multiple of 5, then print **Buzz** instead of the number * If a number is a multiple of 3 **and** 5, then print **Fizz Buzz**

Extension

- Instead of using the built-in % operator from python, implement it yourself using another while loop - Can't trust the python developers!
- Make sure that your program can handle **edge cases** - This is extremely important for real programmes

Challenge 2 (While + List Slicing) - Pascal's Triangle

Here you will create a piece of code which generates Pascal's triangle

Base Problem Statement

Given an integer N (provided by the user), using a while loop print out Pascal's Triangle in the format: 1, 121, 1331, 14641,... row_N
[Try doing this without creating more than one list per loop step for more difficulty]

Extra Challenge

- Use *Nested Lists*! Store each row as a list inside of a larger list, e.g.: [[1], [121], [1331],... [row_N]]
- After you've created your pascal's triangle list, print it out so it looks like an actual triangle:
__1__
_1_2_1...
(Note: strings can be multiplied, e.g. "Hiiiiiiii" = "H" + "i" * 8)