Homework 1

January 23, 2021

Homework should be submitted by email, as a single Jupyter notebook file with name hw1_lastname.ipynb. The code you write should be compatible with Python version 3.7.0 Partial credit may be given, so write your code clearly with as many comments as needed to understand what you are attempting to do and why. The homework is due before the the start of class on Monday, February 1.

1. Write a program to determine the prime factors of a user supplied number. The script should contain a function *prime_factors* that takes a single argument that is an integer. It should return a sorted list of the prime factors. You may want to test your code on some sample numbers using web calculators such as `https://www.calculatorsoup.com/calculators/math/prime-factors.php`.

   Be careful to handle cases where a number's divisors include multiple occurrences of the same prime factor. Your program should handle incorrect user input and not crash, but rather output some kind of complaining message to the user asking him/her to try again. You can accomplish this by checking the input or, more elegantly, using the python **try**...**except** statements for exception handling, see `http://docs.python.org/3/tutorial/errors.html`. If you have time you should try both ways (but you do not have to), since there is something to learn in both approaches.

2. Many websites require users to input a username and password to register. Write a function to generate a random password that meets the following criteria:

   (a) At least one lower case letter

(b) At least one upper case letter

(c) At least one number

(d) At least one character that is not a letter or number

(e) Whitespace characters are not allowed

(f) At least 6 characters long

(g) No more than 20 characters long

The function should be called *generate_password*. It should take no arguments and return a string with the randomly generated password. You should also write a second function called *check_password*, which takes a single argument that is a proposed password and returns a boolean indicating whether or not the password is valid. Hint: you may want to build your solution using existing python modules such as **random** or **string**.

3. Write a function called *get_directory_contents*. The function should take two arguments, the first a path and the second a boolean that has default value **False**. The function should run the shell command */bin/ls* on the user specified path, return the contents of the directory as a list and, if the boolean is true, print a comma-separated list of the contents. Write a second function called *get_list_of_files*. This function should take three arguments: the first a path, the second an (optional) file extension that should default to any extension, and an optional third boolean argument that has the default value **False**. The function should return a list of all files in the input path that match the input file extension. If the boolean input is set to **True**, it should also print a formatted list to the screen. You may find the python **os** or **subprocess** modules helpful. Something like this could be useful because the output of the shell command would then be available to python for further processing. Use the functions you've written to determine the number of perl scripts in the path */usr*, i.e. files that end in '.pl'.

4. Write a function called *hangman* that allows the user to play hangman. It should prompt the user to guess and write feedback as needed to the screen. The program should randomly select a word and, at each step, indicate to the user all relevant information, e.g. which letters have been guessed (correct and incorrectly) and the number of guesses

remaining. The game should end when the word is completed or the player runs out of guesses (after 6 incorrect guesses). Hint: you may want to consider the **nltk** and **random** modules.