ENG 101 – Python Homework #2

Due Tuesday, December 3 at 11:59 pm (Canvas submission)

You are to submit only **ONE** file named "**HW2.py**" with four functions in the file for each of the problems. Each function should have a docstring with help on the function and the assignment header information.

1. Pretend you work for an online email site. Your job is to develop a function called "**`password`**" with one input and one output that asks for user names and passwords. If the input value is "True" it set up new users. If the input value is "False" it checks for a current user name and password. The output is "True" if the function successfully created a new user or if the user entered the valid user name with the correct password for a current user, otherwise, the output is "False". The function should prompt the users for a user name and password. If it is a new user, it should append the new user name and password to a file called "userpass.csv" with a comma used as the delimiter. If it is a current user, it only needs to check if the user name is in the file and if the password is correct for that user.
   a. A valid **user name** should be at least 6 characters long and less than 17 characters and check that it has not been used before in userpass.csv.
   b. The **password** is valid only if the following conditions are met:
      - At least 1 lower case letter [a-z]
      - At least 1 upper case letter between [A-Z].
      - At least 1 number between [0-9].
      - At least 1 of the following special characters: ~, !, @, #, $, %, ^ , &, *, <, or >
      - Any other characters cannot be used in a password.
      - Minimum length of 6 characters.
      - Maximum length of 16 characters.

   The input commands should not be in a loop but only asked for the user name and password once. If the conditions for a valid password are not met, the function should tell the user what is wrong with the password. The function should give an error if it is a new user and the user name has already been selected. For a current user, it should give an error only if both the user name and password do not match what is in the "userpass.csv" file.
   Hint: Use the **`ord`** function to check the ASCII value of individual characters.

2. Use the **grocery.py** file on Canvas to start your program for this problem. It contains two dictionaries of stock and price. Create a function called "**`grocery_cost`**" that has one input (dictionary with food items and quantities). The function will have two outputs of the total cost of a shopping list and the dictionary with food items and quantities sold. The function should check each item in the input dictionary and determine if the item is something that is listed in the stock dictionary. If it is not stocked the function should display a message to the user that it is not an item that is stocked. If it is stocked it should check to see if the quantity requested is available in the stock dictionary. If the requested quantity is greater than the available quantity, it should ask if the user if they would like the quantity that is available listing the stock dictionary. The "**`grocery_cost`**" function also needs to update the number of items available in the stock dictionary after the quantity of each item is finalized. After determining the items that are available in stock, the function should calculate and display the subtotal of the cost, tax of 5.5% and the total cost (subtotal plus tax) of all items. The displayed values should all line up at the decimal point. The total cost and final grocery list are returned from the function.

3. Write a Python function called "**`prime`**". It has one input and one output. The input and outputs are integers. For an input integer n, the function will output the n^{th} prime number. If n is less than 1 the function output should be zero. Example output is shown in Figure 1.

```
========== RESTART: E:/ENG 101/HW203.py =====
>>> prime(7)
17
>>> prime(2000)
17389
>>>
```

Figure 1: Test Output from prime function

4. Modify your in-class "**`fileRW.py`**" assignment to be a function called "**`IO`**". The function will have two input values for the input file and the output file. If the input file does not exist it should give a descriptive error message about the input file. IO should also do the following:
   a. The user should be asked (with input command) for the number of data points to average together. The input text should be clear and descriptive. It should keep asking the user for input until a valid integer is obtained. If an appropriate response is not entered it should give a statement about what is wrong with the input and then ask for new input.
   b. The file should automatically determine the number of data columns and the number of rows that contain header information. Assume the header information is finished if the string in the first column is a floating-point value. Store the header information and write it back to the output file.