CS21 Assignment #9

READ THE ENTIRE DOCUMENT! Use the specified name for your script (shown in parentheses below). Include exception/error handling into all programs.

1. (*make_login.py*) Write a program to that will read in employee information from a file, parse the information, and use that information to create login information for each employee.

Step 1: Write the make_user_name function

Write a function that takes in three parameters, an employee's first name, an employee's last name, and the year the employee was born. Use that information to create the employee's username. The function will not print to to screen or ask for user input. The function will return the created username.

An employee's username will be the first letter of their first name followed by the first 7 letters of their last name followed by the four digit year they were born. Additionally, the following must be true of all usernames:

- Usernames should not contain any capital letters.
- An employee's last name could contain the symbols – or '. The username should not contain these symbols. If one of those symbols appears in the first 7 characters of their last name, it should be removed before finding the first 7 letters. Example: If there was an employee Jim O'Malley who was born in 1972, their username would be `jomalley1972` not `jo'mal1e972`
- If the employee's last name is less than 7 characters in length, additional letters should be used from the first name to ensure that the username will start with 8 letters. For example, if there was an employee named Jacob Ryan who was born in 1984, their username would need to use the first two letters from their first name since their last name is only four letters in length. Jacob Ryan's username would be `jacoryan1984`. If the combined length of the first and last name is still less than 8, then we will not be able to reach the 8 letter minimum. That is acceptable. For example, Joe Day born in 1965 would have the username `joeday1965`

Once you have this function complete, run your script to define the function. Then from the terminal you can call the function yourself to test it out. Here's a test run I did:

```
>>> make_user_name("Frank", "Johnson", 1974)
'fjohnson1974'
>>> make_user_name("Sara", "O'Fly-Hannoran", 1986)
'soflyhan1986'
>>> make_user_name("Bob", "Ess", 1992)
'bobess1992'
>>> make_user_name("Shannon", "LaFluer", 1995)
'slafluer1995'
>>> make_user_name("Elizabeth", "Jefferson-Smith", 1985)
'ejeffers1985'
```

Step 2: write the make_password function:

Write a function that takes in two parameters, the employee's date of birth, and a tuple containing random words that you choose. The function will then create and return the starting password for that user. The password will be the employee's birth month (in MM format) followed by a randomly selected word from the tuple, followed by the employee's birthday (in DD format). The following should be true for all passwords.

- The numbers representing the birth month and birthday should always be two digits in length. If the month or day is less than 10, you will need to add a 0 to the front. For example: if the employee's birthday is 5/7/1991 and the randomly selected word from the tuple is "Otter" the password would be `05Otter07`

Once you have this function complete, run your script to define the function. Then from the terminal you can call the function yourself to test it out. Here's a test run I did (due to randomly selecting a word from the tuple, the output could vary while testing):

```
>>> mywords = ("Otter", "Cow", "Bobcat", "Falcon", "Duck", "Whale", "Owl")

>>> make_password("12/21/1991", mywords)

'12Falcon21'

>>> make_password("4/18/1998", mywords)

'04Cow18'

>>> make_password("11/9/2001", mywords)

'11Duck09'

>>>
```

Step 3: Write the main function:

Now that our functions are written, we want to write the main function for our script. Our main function should ask the user for a name for the input file containing employee information. The main will read in all the employee information in the file and create usernames and passwords for each employee. The program will ask what file the user would like the login information written to and output all the login information to that file.

- The input file will have one employee's information on each line, and each piece of information is separated by a comma, using the format of `FirstName,LastName,DateOfBirth` . You may assume that the file will always follow this format.
- You will need to create a tuple of words to use with your `make_password` function.
- You should output the login information in the following format:
  `<FirstName> <LastName's> username is: <user_name> password is: <password>`
  Examples:
  ```
  Chad Billins's username is: cbillins1989 password is: 05Otter26
  Noah Welch's username is: noawelch1982 password is: 08Bobcat26
  John McCarthy's username is: jmccarth1986 password is: 08Falcon09
  ```
  See my example login_info.txt file for the full output.

It is expected that you will use exception handling and error handling to ensure your programs are robust and will not crash. It is not acceptable to put all of your code in one try suite, your try suites should be specific to one part of the program that could have an exception.

Reminders (not following will result in point deductions):

1. All programs should have a `main()` function as well as additional user-defined function (unless otherwise indicated in the problem). Make sure to utilize value-returning functions now that we've covered them.
2. If a function's task doesn't include output to the user, do not do it!
3. Use constants! No magic numbers!

    Example:
        Correct: GRAVITY = 9.8
        Incorrect: gravity = 9.8
4. It is expected that you will complete the same process of development outlined in the textbook and videos. When you reach the point of having an algorithm (pseudocode), this will become the comments of your program and is the starting point for writing code. Comment first, then code!
5. Be sure to include comments at the top of the program that include your name, class and a short description of the program.
6. All functions should be well documented (include an explanation in comments above the function header / definition stating the task that the function will perform. *Every function must have this!*
7. Be sure all output is formatted. Unless otherwise, specified, displays non-integer values with 2 digits after the decimal point.
8. Your program run must exactly match what is provided as examples in this document for full credit (no modifications to output accepted).
9. Now that exception handling has been covered it is expected that you incorporate exception/error handling into all submitted assignments.
10. Only programming concepts introduced thus far in this course are accepted. While there may be more efficient/elegant solutions to a given problem, the expectation is that you practice the concepts presented. Once a concept is covered, you may use it for the remainder of the course.
11. Create a separate script (Python file) for each program, with the file name as identified above. Submit all files to the Assignment #9Blackboard dropbox (you must attach all files before clicking 'submit').

Any work you submit for this assignment should be authored entirely by yourself. Assistance is permitted from the instructor or teaching assistants only. All submitted programming assignments are subject to originality verification through software designed and used for the Measure Of Software Similarity (MOSS).