Homework 5: Regular Expressions

SI 206, FA24

Introduction

In this assignment, you will work as a hacker to retrieve information about users. Your coworker has obtained encrypted information about users from a website. The data is stored in the file "hw5_data.txt". For each function, your task is to use regular expressions to extract useful information.

Sample Data

The data is plain text with the following format:

```
Jane Doe for the @cc0uNT;janeaccount / associatedwith & has the
P455W0RD:janeaccount123 * linked birthday seems to be 06/28/2003
with socials * checkphone 713-123-4567 || checkEMAIL
jan3do3@gmail.com
```

Your Tasks

Please implement the following functions into hw5.py:

- get_user_info(file_name)
  - This function reads the file and returns a list of strings. Each string should contain all the information about one user (username, password, full name, address, and phone number).
  - Sample Output:
    ```
    User_Data:
    ['Jane Doe for the @cc0uNT;janeaccount /
    associatedwith & has the P455W0RD:janeaccount123 *
    linked birthday seems to be 06/28/2003 with socials *
    checkphone 713-123-4567 || checkEMAIL
    jan3do3@gmail.com', 'John Banking in an
    @cc0uNT;johnbanking / there is an included
    P455W0RD:password * born 04/05/2004 includes obtain
    code 3mAil at john@bank.net || mobile for
    313-555-1234', ...]
    ```
- create_age_dict(user_data)
  - This function gets a list of strings, with each string containing a user's information, and returns a dictionary. The dictionary should use the

username as keys and store the birthday (in MM/DD/YYYY format) and their age as of the due date (October 25, 2024).
    - Sample Output:
```
User_Dict: {‘janeaccount’: (‘06/28/2003’, 21),
‘johnbanking’: (‘04/05/2004’, 20), ...}
```
- check_password_strength(user_data)
    - This function gets a list of strings with each user’s information and returns a tuple that contains the password and the password strength. Weak passwords contain at least 6 characters. Medium passwords contain at least 8 characters and have lowercase letters and numbers. A strong password is at least 10 characters and has lowercase letters, uppercase letters, numbers, and special characters (e.g., !@#$%^&*())
    - Sample Output:
```
Password_Strengths = ( ('janeaccount123', 'strong'),
('password', 'weak'), ... )
```
- sort_email_domain(user_data)
    - This function will extract each user's email and return a sorted (descending) dictionary where the domain name (e.g., 'gmail.com' or 'umich.edu') is the key, and the number of times that domain appears in the user data is the value. You will need to parse the email addresses using regular expressions to extract the domain portion (everything after the '@'). Sort the dictionary by the frequency of the domains in descending order.
    - Sample Output:
```
Email_Data = {
    'gmail.com': 5,
    'yahoo.com': 2,
    ...
}
```

Test Cases
There are 4 txt files set up for you in a temporary directory. Make test cases for each txt file for `get_user_info`, `create_user_dict`, `check_password_strength`, and `get_user_initials`. Make sure your tests cover two scenarios: confirming that things that should match indeed do, and ensuring that things that shouldn't match don't.

Extra Credit
- validate_michigan_number()

    - This function checks all of the phone numbers to see if it is a southeast Michigan number (has the area code of 313 734, or 810) and puts all of the valid numbers into a list.
    - Sample Output:

```
Michigan_Numbers = [
    '313-555-1234',
    '734-987-1234'
    ...
]
```

- test_validate_michigan_number()
    - Create test cases for `validate_michigan_number` with the same criteria as indicated above for the other test cases.

Grading Rubric (60 points + 6 points)

This rubric does not show all the ways you can lose points. For each of the functions, you can earn:

- points for correctly implementing each of the 4 functions (44 points)
    - get_user_info (8 points)
    - create_age_dict (10 points)
    - check_password_strength (16 points)
    - sort_email_domain (10 points)
- points for creating tests for each function (16 points)
    - 2 points each * 8 test cases
- points for extra credits and test cases (6 points)
    - correctly implementing validate_michigan_number (4 points)
    - creating tests (2 points)

Submission

Make at least 4 git commits and turn in your GitHub repo URL on Canvas by the due date to receive credit.