CSE 8A Programming Assignment 4

Due Date: Tue. Nov 10th, 11:59 PM (PDT)

Learning goals:

- Practice using methods such as split, join, and append
- Practice making modification to a list
- Understanding how function calls work

Logistics:

All information for the following can be found on our course syllabus.

- **Pair programming** - This programming assignment can be done individually or with a partner. Make sure to read the guide on pair programming.
- **Academic integrity** - Please adhere to all academic integrity guidelines on the syllabus.

Submission:

- You will submit the following files to Gradescope:
    - **pa4.py** - Contains your code for part 1.1 (and star points, if attempted)
    - **Pa4-writeup.pdf** - Contains all written portions of the assignment
    - **Pa4-video.mp4** - Your recorded video
- You can find the template of the write-up here
- You are also required to complete the weekly reflection
- Instructions on how to submit can be found below at submission instructions.

Part 1: Implementation (6 Points)

Part 1.1: Processing Emails

Save all of the following code in the provided python file named pa4.py.

Every email consists of a user name and a domain name, separated by the @ sign. For example, in gerald@eng.ucsd.edu, gerald is the user name, and eng.ucsd.edu is the domain name.

Let's say that we want to collect all the user names of the email addresses from CSE 8A students to assign groups, and we ask for your help. Your job will be given a list of email addresses, and return a list of user names from those email addresses. The list of user names returned should be in the same order as the list of email addresses. You may assume that the

input parameter will always be a list of email addresses in the correct format (i.e., <user_name>@<domain_name>).

Function Name: *extract_usernames_from_emails*
Parameter: *emails* - A list of strings that contains the email addresses.
Return: A list of strings that contains the user names.
Description: Given a list of email addresses, return a list of user names from those email addresses.
Examples: emails = ["gerald@eng.ucsd.edu", "ruohan@ucsd.edu", "ruohan@gmail.com"] Returns: ["gerald", "ruohan", "ruohan"] emails = ["lerner@eng.ucsd.edu", "joseph@ucsd.edu"] Returns: ["lerner", "joseph"] emails = ["yeohee@ucsd.edu"] Returns: ["yeohee"]

The input parameter (i.e., *emails*) can contain a list of any number of emails. Your function should work as expected irrespective of the number of emails in the list.

Part 1.2: List Mutation

Save all of the following code in the provided python file named pa4.py.

For this part, we would like you to implement the find and replace functionality, which is very common in a text editor. That is given a list of words, a word *word_to_find* and a word *word_to_replace*. Replace all the occurrences of *word_to_find* in the list with the *word_to_replace*. Please notice that the second parameter of the function is *word_to_find*, and the third parameter of the function is *word_to_replace*.

The function should be case sensitive, meaning that, if you look at the example below, the word "JOY" and "Joy" in the list will not be replaced with the word "happy" because we treat "JOY", "Joy", and "joy" as different words. Also, the function will find and replace exact matches. For example, the word "joy" in the word "joyful" will not be replaced when we want to find "joy" and replace it with "happy". See example below.

Function Name: *find_and_replace*

Parameter: *words* - A list of words. *word_to_find* - A word to be found in the list. *word_to_replace* - A word to replace *word_to_find* in the list.
Return: A list of words that replaces all the occurrences of *word_to_find* in the list with the *word_to_replace*.
Description: Given a list of words, a word *word_to_find* and a word *word_to_replace*, replace all the occurrences of *word_to_find* in the list with the *word_to_replace*.
Examples: find_and_replace(['joy', 'anger', 'fear', 'disgust', 'JOY', 'joy', 'Joy', 'joyful'], 'joy', 'happy') Return: ['happy', 'anger', 'fear', 'disgust', 'JOY', 'happy', 'Joy', 'joyful'] find_and_replace(['CSE 8A', 'CSE 8B', 'CSE 11'], 'CSE 12', 'CSE 100') Return: ['CSE 8A', 'CSE 8B', 'CSE 11']

Star Points (optional):

*Save all of the following code in **pa4.py***
Read about star points on our course syllabus.

Part 1.3: Processing Emails, Again :)
(Only for star points; Not required to get full credits on this PA)

This time we would like your help for collecting all the **unique** domain names from the list of email addresses. For example, gerald@eng.ucsd.edu has a domain name of eng.ucsd.edu and asoosairaj@ucsd.edu has a different domain name of ucsd.edu.

Function Name: *extract_domains_from_emails*
Parameter: *emails* - A list of strings that contains the email addresses.
Return: A list of strings that contains the unique domain names.
Description: Given a list of email addresses, return a list of **unique** domain names from those email addresses.
Examples: emails = ["gerald@eng.ucsd.edu", "ruohan@ucsd.edu", "yeohee@ucsd.edu"]

```
Returns: ["eng.ucsd.edu",  "ucsd.edu"]

emails = ["lerner@eng.ucsd.edu",   "joseph@ucsd.edu",  "ruohan@bio.ucsd.edu"]
Returns: ["eng.ucsd.edu",  "ucsd.edu", "bio.ucsd.edu"]

emails = ["apwai@ucsd.edu", "jwarmus@ucsd.edu", "pranand@ucsd.edu", "r8hu@ucsd.edu",
"sasriniv@ucsd.edu", "wiwilson@ucsd.edu", "yim@ucsd.edu", "dverzhbi@ucsd.edu",
"rsbhusha@ucsd.edu", "usinha@ucsd.edu"]
Returns: ["ucsd.edu"]
```

The input parameter (i.e., *emails*) can contain a list of any number of emails. Your function should work as expected irrespective of the number of emails in the list.

Part 2: Write up (3 Points)

You must report how you tested your code as well as answer a few short questions in **Pa4-writeup.pdf.** A template has been provided. See submission instructions below on how to make a copy of this template to your own drive. In particular, you must provide:

A. Report Bugs and Issues
Include in your report any known bugs or issues with your program.

B. Questions
Answer the following questions:

1. In Part 1.1, you probably used the split method. There is another method called join. Explain what join method does and give an example to illustrate.

2. In Part 1.2, we are doing list mutation. Can you do the same thing to a string in python (for example, given a string and replace a certain character with another character)? If so, please write down the approach; if can not, please explain why you could not do so.

Part 3: Video

For this part, you will create a video recording explaining the code you have written. Your video should answer the questions below. If you are working with a partner, both partners should be in the video and each partner should have some speaking point. Example of following code execution: https://youtu.be/7uw_Vi_F-dY

1. Follow the execution of the function *extract_usernames_from_emails*, line by line, with an input of [“gerald@gmail.com”, “sorin@ucsd.edu”, “csestudent@ucsd.edu”].

2. Follow the execution of the function *find_and_replace*, line by line, with an input of [“apple”, “orange”, “APPLE”, “apple”], “apple” (this is the *word_to_find*), and “pear” (this is the *word_to_replace*).

The following things will be checked in your video while grading:

1. The student(s) **code is clearly visible** in the video. [Hint: Increase your font size to 18
   Windows: "Options" → "Configure Idle" → Size
   Mac: "IDLE"(in top menu bar) → "Preferences" → "Fonts/Tabs" → "Size"]
2. The student(s) clearly **answer(s) the questions**.
   a. Students submitting individually answer both questions.
   b. Students submitting in pairs each answer one of the questions.
3. Video is within **time limit** (max: **2 mins**)

Part 4: Weekly Reflection (1 Point)

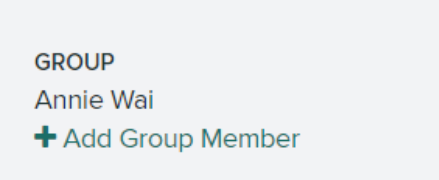
Fill out the reflection form (link here). This weekly reflection form is not optional, it counts towards 1 point of your assignment. All students have to **individually** submit their own weekly reflection regardless if you’re working with a partner. Weekly reflections are not due when the PA is due. You may submit your weekly reflection for PA 4 anytime before **11:59pm on Friday Nov 13th**.

Submission Instructions

Read all instructions carefully before submitting.

- You will need to submit **pa4.py, Pa4-writeup.pdf, and Pa4-video.mp4** on Gradescope, fill out the weekly reflection, and complete the degree planning assignment.
- To copy the writeup template and export as a pdf:
  - Click here to see a copy of the assignment format.
  - Click on "File" -> "Make a copy", and you will get a local copy of this Google Doc.
  - Fill in the Google Doc, making sure you keep the headings in about the same places.
  - Once you are done, in Google Docs, click on "File"-> "Download" -> "PDF Document", which will export it to a pdf.
- To record a video on zoom:
  - Start a zoom video meeting.
  - Choose "Join with Computer Audio"
  - If you’re working with a partner, invite your partner into the meeting.
  - Share your computer’s screen using "Share Screen"
  - Show your code on your computer’s screen
  - Click "Record" > "Record on your computer".
  - Answer the required questions.
  - Once you are done, click "stop recording" and "End meeting".
  - Save your video file on your computer and name it as Pa3-video.mp4

- Sign into Gradescope and submit all files to PA 4. You should be able to drag and drop multiple files into the upload files window. Ask a teaching staff for help if you are unsure whether you've submitted properly.
- If you are working with a partner, **only one member will need to submit the files** in Gradescope. Do not both submit the files individually to Gradescope. It will be your responsibility to ensure both members are added in Gradescope.
- To add a group member on Gradescope:
    - First submit all files to PA 4.
    - This should take you to your submissions page. Otherwise, you can view your submission by clicking on the assignment.
    - Click on **"Add Group Member"** on the top right under your name.

    - Confirm you have added your partner. You should see both you and your partner's name under "Group" in the top right after submitting.
- You may submit multiple times until the deadline. We will be grading only your latest submission. So, please make sure that your latest submission is your best version!