CSE 8A Programming Assignment 7

Due Date: Tue. Dec 1st, 11:59 PM (PST)

Need 1-on-1 Help?

You can get help by submitting a help ticket on the **Autograder**! Please read **Remote Tutoring Procedures For Students** for more instructions. Look at the **Course Calendar** to check when tutors/TAs are available! Look for "tutor hours" or "office hours" on the calendar. We expect that all students will need help at some point in time, so don't hesitate to reach out :)

Learning Goals:

- Understand the need for dictionaries
- Learn how to create a dictionary
- Access/update contents in dictionaries

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:
  - **pa7.py** - Contains your code for Part 1
  - **Pa7-writeup.pdf** - Contains all written portions of Part 2
  - **Pa7-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)

Save all of the code in the python file named pa7.py

Part 1.1: Distribution of Majors (2 Points)

We would like to analyze the distribution of majors of students in our class. Assume that we already collected data from surveys and arranged a list of strings that describe majors such as `['CSE', 'CSE', 'SE', 'CSE', 'NANO']`.

However, strings in the list are not sorted, so we would like to create a dictionary that maps each major with the number of students associated with the major, so that we can readily observe the distribution of majors. Your function will be given a list of majors as a parameter. It has to create and return a dictionary, the keys of which are majors and the values of which are the number of students for each major.

You don't have to worry about the order of key-value pairs in your dictionary, as long as the distribution is accurately computed. Please note that dictionaries are unordered in the sense that two dictionaries are equivalent if they contain the same key-value pairs regardless of in what order the pairs are written. For instance, if two dictionaries are declared as follows,

```
>>> dict1 = {'key1': 1, 'key2': 2}
>>> dict2 = {'key2': 2, 'key1': 1}
```

the two dictionaries are equivalent.

```
>>> dict1 == dict2
True
```

Please define the following function.

Function Name: *distribution_majors*
Parameter: `list_majors` - A list of strings that denote majors of students
Return: A dictionary of majors as its keys and the number of students as its values
Description: Your function is given a list of majors of students. Return a dictionary that represents the distribution of majors

Examples:
`list_majors = ['CSE', 'CSE', 'SE', 'CSE', 'NANO']`
Returns: `{'CSE': 3, 'SE': 1, 'NANO': 1}`
 	
`list_majors = ['CSE', 'CSE', 'SE', 'BENG', 'NANO', 'BENG', 'ECE', 'MAE', 'CSE', 'ECE', 'MAE', 'SE']`
Returns: `{'CSE': 3, 'SE': 2, 'BENG': 2, 'NANO': 1, 'ECE': 2, 'MAE': 2}`
 	
`list_majors = ['BENG', 'CSE', 'CSE', 'MAE', 'CSE', 'ECE', 'MAE', 'BENG', 'BENG', 'CSE', 'MAE', 'CSE', 'ECE', 'ECE', 'MAE', 'ECE']`

Returns: `{'BENG': 3, 'CSE': 5, 'MAE': 4, 'ECE': 4}`

Part 1.2: COVID in the USA (2 Points)

As COVID case numbers are on the rise, one may be curious which states in the USA have been good at keeping cases low compared to other areas. You will be given a dictionary with some state names as *keys*. The *values* of the given dictionary will be the number of total cases per 100,000 residents for each state. Also, you will be given an integer that indicates a threshold for the number of cases. Your function has to create and return a new dictionary that contains only those states that have lower than the threshold of cases per 100,000 residents. The newly arranged dictionary includes those states as *keys* and their number of total covid cases per 100,000 residents as *values*. You don't have to worry about the order of key-value pairs in your dictionary.

Please define the following function.

Function Name: *select_states*

Parameter:
`cases` - A dictionary with state names as keys and the number of COVID cases per 100,000 residents as values
`threshold` - An integer that represents the upper threshold of cases per 100,000 residents

Return: A dictionary that contains states with lower than the threshold of cases per 100,000 residents

Description:
Your function is given a dictionary that contains the information of COVID cases for each state. Return a new dictionary that shows the information of states with COVID cases lower than the threshold.

Examples:

```
cases = {'Vermont': 497, 'Alabama': 4504, 'Washington': 1849,
'California': 2670}
threshold = 4000
```
Returns: `{'Vermont': 497, 'Washington': 1849, 'California': 2670}`

```
cases = {'Maine': 697, 'Wyoming': 4225, 'Pennsylvania': 2195,
'Montana': 4631, 'Massachusetts': 2800}
threshold = 2500
```
Returns: `{'Maine': 697, 'Pennsylvania': 2195}`

```
cases = {'Illinois': 4730, 'Colorado': 2999, 'Iowa': 6068,
'Delaware': 3035, 'Utah': 4958, 'Michigan': 2968, 'California':
2670}
```

```
threshold = 1000
```
Returns: `{}`

Part 1.3: NFL Standings

In the middle of an NFL season, we are given a dictionary that shows the number of wins until last week for some teams. Based on sports news during the past few days, we would like to update the dictionary with the recent number of wins this week. A dictionary is given to your function that has NFL teams as keys and their number of wins till last week as values. Also, a list of teams that won this week is given to your function. No team has won more than once for the past week. Then, your function has to return a dictionary which shows the updated numbers of wins for each team.

Please define the following function.

Function Name: *update_standing*

Parameter:
`win_record` - A dictionary that has NFL teams as keys and their numbers of wins as values
`new_winners` - A list of teams as strings that won for this week

Return: A dictionary that has updated information about the numbers of wins for each team

Description:
Your function is given a dictionary that shows NFL standing until last week and a list of teams which won a game this week. The function updates the NFL standing.

Examples:
```
win_record = {'Patriots': 4, 'Chiefs': 8, 'Eagles': 3, '49ers': 4}
new_winners = ['Patriots', '49ers']
```
Returns: `{'Patriots': 5, 'Chiefs': 8, 'Eagles': 3, '49ers': 5}`

```
win_record = {'Giants': 3, 'Seahawks': 6, 'Bears': 5, 'Falcons': 3,
'Panthers': 3}
new_winners = ['Giants', 'Panthers']
```
Returns: `{'Giants': 4, 'Seahawks': 6, 'Bears': 5, 'Falcons': 3,
'Panthers': 4}`

It is guaranteed that each team in the `new_winners` list is what already exists in the `win_record` dictionary. No team that is not in `win_record` will appear in `new_winners`.

Part 1.4: Star Points (optional)

Create your own useful dictionary and a function that manipulates your dictionary. You can create any dictionary of your choice as long as it has some meaning to it. Your function should do some useful process or computation on the dictionary you created. Your dictionary and function should be nontrivial. For example, don't create a meaningless dictionary and a function that returns the same dictionary without any useful process. We will not grant any star point for a trivial answer. If you are unsure whether your dictionary and function are trivial, ask a teaching staff. This question is completely open-ended, be creative!

The function you write should meet the following minimum requirements:

- Takes at least one **dictionary as an input parameter**
    - Your function may optionally take any additional parameters as needed
- **Uses the dictionary** (and other parameters) to perform some **non-trivial computation**
- **Returns** some value
    - Your function may return any data of the following types:
        - int
        - float
        - string
        - boolean
        - list
        - tuple
        - dictionary
- Your function has a **2 - 3 lines of comments** explaining the purpose of your function

Add your star point code to *pa7.py* and complete part C on the write up.

Part 2: Write up (3 Points)

You must report how you tested your code as well as answer a few short questions in **Pa7-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. Explain similarities and differences between lists and dictionaries.

2. The following dictionary represents information about a student.
```
student_info = {'first name': 'Yeohee', 'last name':
'Im', 'ID': 'A00000000', 'age': 20, 'academic year': 2}
```
Write code with a for loop to have the output printed in the form of tuples as follows:

```
('first name', 'Yeohee')
('last name', 'Im')
('ID', 'A00000000')
('age', 20)
('academic year', 2)
```

Hint: look into different methods that can be used on dictionaries

C. **Star Point (Optional)**
If you attempted the star point question. Explain what dictionary and function you created and why they are useful.

Part 3: Video (2 Points)

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 *distribution_major*, line by line, with the input
   `list_majors = ['CSE', 'CSE', 'SE', 'CSE', 'NANO']`

2. Follow the execution of the function *update_standing*, line by line, with the input
   `wins = {'Patriots': 4, 'Chiefs': 8, 'Eagles': 3, '49ers': 4},`
   `teams = ['Patriots', '49ers']`

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**)

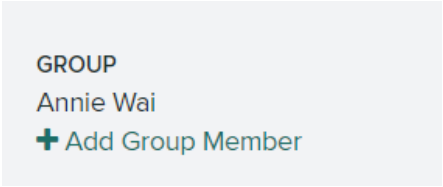
Part 4: Weekly Reflection (1 Point)

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 7 anytime before **11:59pm on Friday Dec 4th**.

Submission Instructions

Read all instructions carefully before submitting.

- You will need to submit **pa7.py, Pa7-writeup.pdf, and Pa7-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 Pa7-video.mp4
- Sign into Gradescope and submit all files to PA 7. 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 7.
    - 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!