Assignment - 3: Python

Software Systems Development

Deadline: 11:55 pm, 4 November 2020

Commits made on GitHub before the assignment deadline will be considered for grading.

Submission Instructions on Moodle:

- A README file is mandatory.
- Any/all questions/clarifications regarding Assignment 3 must be Via Moodle. Please use the Assignment 3 thread on the News Forum for the same.

Submission Instructions on Github:

- Create a GitHub repository. [IGNORE IF ALREADY DONE]
- Push all three questions to this repository as given in the Moodle submission format and add a readme.md as well.

NOTE:

Reduce the cyclomatic complexity of your code to preferred grades of A(ideal) or B (Accepted) as outputted by radon tool.

Question 1.

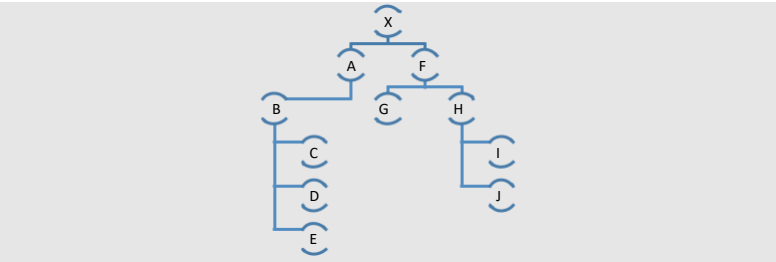
Write a python program to find common leader of any three to five employees in an organization.

For example, when inputted the organization chart as in figure below, we can conclude **F** is common leader of **employee G, employee H** and **employee I**

F is common leader for **employee G, employee H, employee I** and **employee J**

X is common Leader for **employee A, employee B, employee J, employee G** and **employee E**

You need to create a JSON file `org.json` in given format (Use Employee number in place of A, B, C……):



The above org chart will be inputted as a JSON like below

```
{
        "L0": [
                { "name": "X" }
        ],
        "L1": [
                { "name": "A", "parent": "X" },
                { "name": "F", "parent": "X" }
        ],
        "L2": [
                { "name": "B", "parent": "A" },
                { "name": "G", "parent": "F" },
                { "name": "H", "parent": "F" }
        ],
        "L3": [
                { "name": "C", "parent": "B" },
                { "name": "D", "parent": "B" },
                { "name": "E", "parent": "B" },
                { "name": "I", "parent": "H" },
                { "name": "J", "parent": "H" },
         ]
}
```

User can input any three employees' Emp No. (use input() function) and he should get common leader (Emp ID) as an output.

Also show the level of the leader from employee. For example: leader A is two levels above employee C in the above tree.

#Employee Number format should strictly be of type integer for e.g.: [001-999]

NOTE: Code should be able to handle different sizes of input org tree ie. It should be able to handle increase and decrease of org size.

==========

Question 2.

Write a python program to find difference between given dates.
For example

If Date1 is '**10th September, 2020**' or '**10th Sep, 2020**' or '**10/9/2020**' or '**10-9-2020**' and Date2 is '**11th September, 2020**' or '**11th Sep, 2020**' or '**11/9/2020**' or '**11-9-2020**', then your output should be below

Date Difference: 1 day

Input file date_calculator.txt should be in following format with two rows:

Date1: **10th September, 2020**
Date2: **11th September, 2020**

Output file output.txt will have output in number of days as below Date Difference: 1 Day

Note: you cannot use any date library.

Change your code so that user can input following date formats:

mm/dd/yyyy

mm-dd-yyyy

mm.dd.yyyy

Note: In future the formats might change further so try to make your code as generic/adaptable as possible.

==========

Question 3:

You have three to 5 text files Employee1.txt, Employee2.txt, Employee3.txt... Each file contains the busy slots of each employee in a dictionary format.
NOTE: Your code should be adaptable to 3 to 5 text files. So, make the changes accordingly.

Key: Employee Name (Assuming the names are all unique)
Value: a dictionary with **key** being the date and **value** – a list of employee's busy slots.

You need to write a python program that will search for all free slots for the given two employees and reserve the first available common slot. (Slot duration to be inputted by the user). If no common slot available, then just print no slot available.

Workday to be considered from 9AM – 5PM on any given day.

Output file must contain the following,

- All Available slots before blocking the slot,
- Time of the blocked slot for the both employees.

Example of Input files:

You can create Employee1.txt and Employee2.txt in the given format and ask the user to input meeting slot duration.

Employee1.txt
{'Employee1': {'5/10/2020':['10:00AM - 11:00AM', '12:30PM - 1:00PM', '4:00PM - 5:00PM']}}

Employee2.txt
{'Employee2': {'5/10/2020':['10:30AM - 11:30AM', '12:00PM - 1:00PM', '1:00PM - 1:30PM', '3:30PM - 4:30PM']}}

Output file (output.txt)

Available slot
Employee1: ['9:00AM - 10:00AM', '11:00AM - 12:30PM', '1:00PM - 4:00PM']
Employee2: ['9:00AM - 10:30AM', '11:30AM - 12:00PM', '1:30PM - 3:30PM', '4:30PM - 5:00PM']

Slot Duration: ½ hour
{'5/10/2020' : ['9:00AM - 9:30AM']}

Note: The timeslots can be different than the ones that are in the example file.