National University of Singapore
School of Computing
CS1010X: Programming Methodology
Semester II, 2019/2020

Mission 12 - Side Quest
Facebook Stalkers

Release date: 11 April 2020
Due: 17 May 2020, 23:59

Information

Have you ever wondered how Facebook applications are made? Developers make use of an API (Application Programming Interface) provided by Facebook, which simply put, is a link to retrieve data from Facebook's databases and the data is returned in the JSON (JavaScript Object Notation) format.

In this sidequest, you will do some analysis (read "stalking") on the CS1010X Facebook group page and practise using Python dictionaries.

A template file `sidequest12.1-template.py` has been provided for you. In addition, please ensure that you have the file `cs1010x-fbdata.json` which contains the Facebook data that you will be working with, and `grading.json` which the grader in the template file will use to grade your work.

You should write your solutions in the template file first, and then copy only the required functions onto Coursemology when you are done. Do not copy over the grading script or test code. The variable `member_dict` will be defined for you in Coursemology based on your function in part (d); you may use the variable in the parts after part (d) without defining it.

About the Data

JSON is a format that is a text-based open standard designed for human-readable data interchange. Hence data in JSON format can be parsed by different programming languages such as JavaScript, Python, PHP and many more. Using Python's `json` module which we can use in our code by including `import json`, we can convert JSON data into Python dictionaries and lists. In fact, the JSON syntax should look very familiar to you because it is almost identical to Python syntax for defining dictionaries and lists. An example of data in JSON format is as follows:

```
{
    "first_name": "Ben",
    "last_name": "Leong",
    "age": 21,
    "address": {
        "street": "15 Computing Drive, COM2 Building, #03-20",
        "city": "Singapore",
        "state": "Singapore",
        "postal_code": 117418
    },
    "phone_numbers": [
        {
            "type": "office",
            "number": "6516 4240"
        },
        {
            "type": "fax",
            "number": "6779 4580"
        }
    ]
}
```

If we save the above text in a file called `data.json`, we can open the file to read its contents using the following code:

```
>>> import json
>>> data_file = open('data.json', 'r', encoding='utf-8')
>>> prof_data = json.loads(data_file.read())
```

After doing so, the variable `prof_data` will be initialised as a dictionary with the keys `"first_name"`, `"last_name"`, `"age"`, `"address"`, `"phone_numbers"`. Running the following code in the interpreter gives:

```
>>> print(prof_data['first_name'])
Ben
>>> print(prof_data['address'])
{'street': '15 Computing Drive, COM2 Building, #03-20',
 'city': 'Singapore', 'state': 'Singapore', 'postal_code': 117418}
>>> print(prof_data['address']['city'])
Singapore
>>> print(prof_data['phone_numbers'][0]['number'])
6516 4240
```

For this side quest, we will be analysing a subset of the data that lies in our CS1010X Facebook group. The data has been downloaded as of 29 March 2017, saved in the file `cs1010x-fbdata.json` and can be read in a similar fashion as shown above.

Our CS1010X Facebook group has data in the following format:


```
{
   "members": {
      "data": [
         {
            "name": "Aadit Kamat",
            "id": "1003982836283025"
         },
         {
            "name": "Rakshit Gogia",
            "id": "10204299775189027"
         },
         ...
      ]
   },
   "description": "This is the official FB Group for ...",
   "name": "CS1010X",
   "feed": {
      "data": [
         {
            "message": "Might be useful for the business analytics ...",
            "from": {
               "name": "Ben Leong",
               "id": "10152805891837166"
            },
            "name": "Machine Learning with Python - BDU"
            "id": "409054432560329_1002582839874149",
            "likes": {
               "data": [
                  {
                     "id": "10208170707289199",
                     "name": "Lim Kian Hwee"
                  },
                  {
                     "id": "10204292869386114",
                     "name": "Siidheesh Theivasigamani"
                  },
                  ...
               ]
            },
            ...
         },
         ...
      ]
   },
   "id": "409054432560329"
}
```


Note that the data is large and is difficult to parse without JSON inspection tools. To get around this, we can use an online tool such as `http://www.bodurov.com/JsonFormatter/` to

format our JSON data nicely where we can collapse and expand values. Copy the data from the file `cs1010x-fbdata.json`, paste it in the text field and press "Format". You can then click on the '+' and '-' symbols to expand/collapse the data.

The data has been read and initialised as the `fb_data` variable in the template code. You can make use of the variable directly in your answers. With the data in your hands, let's analyse the data and obtain some statistics about the CS1010X Facebook Group!

This side quest consists of only **one** task BUT **ten** subtasks. (:

Task: Facebook Data Analysis (16 marks)

Note: The following questions require you to inspect the JSON data file with the help of the online inspector to find out where the relevant data resides, so that you can do your own processing.

BEWARE! The data isn't exactly as clean as you might expect; not every dictionary object in the data file has the same keys as the rest. You may have to check whether a key exists in the dictionary before accessing the corresponding value.

(a) Write a function `count_comments` that takes in an argument, the Facebook data, and returns the number of comments in the Facebook group.

```
>>> print(count_comments(fb_data))
469
```

(b) Write a function `count_likes` that takes in an argument, the Facebook data, and returns the number of likes in the Facebook group. Note that both messages and comments can have likes.

```
>>> print(count_likes(fb_data))
1539
```

(c) Write a function `create_member_dict` that takes in an argument, the Facebook data, and returns a dictionary of the Facebook group's members where the key is the `id` and the value is the member data object without the id.

```
>>> member_dict = create_member_dict(fb_data)
>>> print(member_dict["10205702832196255"])
{'gender': 'male', 'name': 'Louie Tan Yi Jie'}
# Note that the order of the keys in a dictionary does not matter
```

Questions:

(i) Why did we choose the `id` of the member data object to be the key?

(ii) It is inappropriate to use the `name` as the key. What will happen if we use `name` as the key of `member_dict`?

(d) Write a function `posts_freq` that takes in an argument, the Facebook data, and returns a dictionary of key-value pairs where the key is the id of a user and the value is the number of wall posts that user has made in the Facebook group. Members who have not posted before should not be included.

An example of a wall post would be `fb_data['feed']['data'][<index>]['message']`.

```
>>> print(posts_freq(fb_data))
{'10152805891837166': 158, '803419996347741': 8, ...}
```

(e) Write a function `comments_freq` that takes in an argument, the Facebook data, and returns a dictionary of key-value pairs where the key is the id of a user and the value is the number of wall comments that user has made in the Facebook group. Members who have not commented before should not be included.

```
>>> print(comments_freq(fb_data))
{'10152805891837166': 158, '803419996347741': 8, ...}
```

(f) Write a function `likes_freq` that takes in an argument, the Facebook data, and returns a dictionary of key-value pairs where the key is the id of user and the value is the number of items that user has liked in the Facebook group. Members who have not liked anything before should not be included.

```
>>> print(likes_freq(fb_data))
{'10208170707289199': 7, '10153237758294588': 25, ...}
```

(g) Write a function `popularity_score` that takes in an argument, the Facebook data, and returns a dictionary of key-value pairs where the key is the id of members and the value is the number of likes their wall posts and/or comments have received. Members who have not received any likes before should not be included.

```
>>> print(popularity_score(fb_data))
{'10152805891837166': 1106, '803419996347741': 35, ...}
```

(h) Write a function `member_stats` that takes in an argument, the Facebook data. The function returns a dictionary which extends `member_dict`, adding in the properties `posts_count`, `comments_count` and `likes_count`, into each member data object. If a member has not posted / commented / liked anything before, the count for that action should be set to 0.

```
>>> stats = member_stats(fb_data)
>>> print(stats["10152805891837166"])
{'name': 'Ben Leong', 'posts_count': 158, 'comments_count': 133,
 'likes_count': 8}
```

(i) Write a function `activity_score` that takes in an argument, the Facebook data, and returns a list of dictionary of key-value pairs where the key is the id of members and the value is a weighted score that is computed based on the member's activity in the Facebook group. The weighted score is computed using the following formula: each wall post is worth 3 points, each comment is worth 2 points and each like is worth 1 point. Members who have not posted / commented / liked anything before should have 0 points.

```
>>> scores = activity_score(fb_data)
>>> print(scores["10153020766393769"])
30
>>> print(scores["857756387629369"])
8
```

(j) Lastly, write a higher order function `active_members_of_type` that takes in three arguments, the Facebook data, a positive integer k and a frequency function, and returns a sorted list (sorted in descending order according to frequency) of two-element lists where the first element is the name of the member and the second element is the value generated by the frequency function for the member. The return result only includes elements with frequencies that are $\geq k$, where $k \geq 1$. In case of a tie, the names are sorted in alphabetical order. You need not list non-members.

Note: You should make use of `create_member_dict` to help you get the name of a member from his/her id in $O(1)$ time.

```
>>> print(active_members_of_type(fb_data, 2, posts_freq))
[['Ben Leong', 158], ['Raj Joshi', 8], ['Sean Tay', 7],
 ['Mak Qi En', 4], ['Yu Ke Lim', 4], ['Peck Ying', 3],
```

```
 ['Brian Koh', 2], ['Louie Tan Yi Jie', 2], ['Nicholas Lui', 2],
 ['Rahul Vinod', 2], ['Renxing Liu', 2], ['Ryan Tal Teo', 2],
 ['Shamantha Yan', 2]]

>>> print(active_members_of_type(fb_data, 20, comments_freq))
[['Ben Leong', 133], ['Sean Tay', 28], ['Yu Ke Lim', 26],
 ['Tay Yang Shun', 25]]

>>> print(active_members_of_type(fb_data, 40, likes_freq))
[['Tay Yang Shun', 63]]

>>> print(active_members_of_type(fb_data, 20, popularity_score))
[['Ben Leong', 1106], ['Peck Ying', 43], ['Raj Joshi', 35],
 ['Shamantha Yan', 25], ['Matthew Chan', 23], ['Brian Koh', 22],
 ['Yu Ke Lim', 22]]

>>> print(active_members_of_type(fb_data, 80, activity_score))
[['Ben Leong', 748], ['Tay Yang Shun', 116], ['Sean Tay', 92],
 ['Yu Ke Lim', 89]]
```