Engineering Challenge - Python Developer

Goal:
To evaluate the candidate based on:

- Ability to understand a new product
- assess coding skills
- assess knowledge of basic Data structures

Both the 2 challenges are mandatory.
Challenge 1
Refer to the file api-response.json you can find at this link **[1]**. This is a sample response from one of the omni:us APIs which provide metadata about the documents.

Problem 1: Write a function to capture the **status** vise distribution of documents. The function should return an appropriate data structure which will help to identify how many documents belong to each status.
Consider the example below:

```
[{
    "document_id": "d95a099b-b275-4b5f-9a61-1a79d6549705",
    "collection_id": "1a6b5bf1-d1b2-489b-99b1-fc3863d8b9cb",
    "status": "REOPENED",
    "file_name": "10.pdf",
    "created_date": "2019-09-18T11:34:07",
    "revision_number": 0
  },
  {
    "document_id": "851b8766-170a-43f6-ba80-7eb5d0000541",
    "collection_id": "1a6b5bf1-d1b2-489b-99b1-fc3863d8b9cb",
    "status": "VALIDATED",
    "file_name": "77.pdf",
    "created_date": "2019-09-18T11:34:13",
    "revision_number": 0
  },
  {
    "document_id": "6d149bc8-8274-4479-a03a-dc95fba39b20",
    "collection_id": "1a6b5bf1-d1b2-489b-99b1-fc3863d8b9cb",
    "status": "REOPENED",
    "file_name": "44.pdf",
    "created_date": "2019-09-18T11:34:59",
    "revision_number": 0
  }]
```

The function should indicate that:

- 2 documents are in **REOPENED** state
- 1 document is in **VALIDATED** state

Problem 2: Use the function written as part of **Problem 1**, and write a function that accepts **status** as an argument and returns all the documents details (document_id, collection_id, file_name, created_date, revision_number) belonging to that status.

Problem 3: Write a function that accepts **file_name** as an argument and returns all the documents details (document_id, collection_id, status, created_date, revision_number) belonging to that file_name.

Please Note :

Do not **print** anything as part of the above functions. **Identify the appropriate return type** for each of the above functions and implement these functions to return the same.

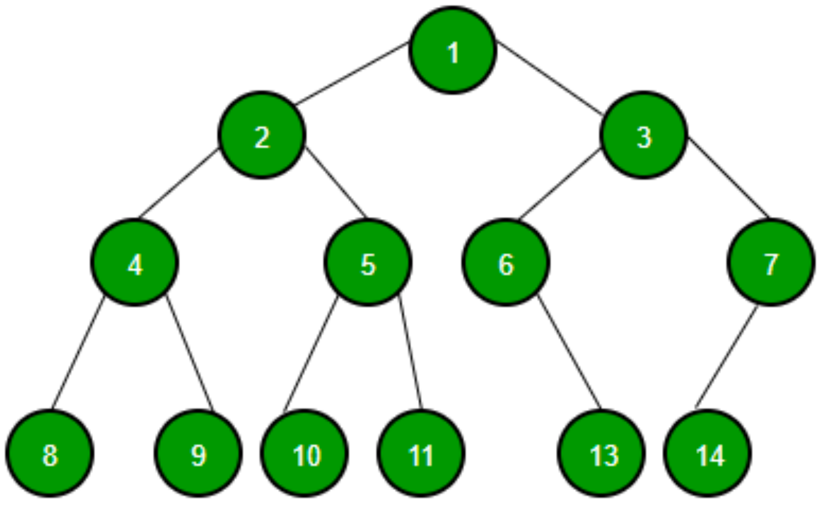
Challenge 2

Objective: Write a python function that flattens a tree and convert it into a dictionary.

Input: A json file whose content follows a tree structure. See sample-1.json**[2]**, sample-2.json**[3]**, sample-3.json**[4]** for reference and test samples.

Expected Output: A dictionary that will appropriately represent the given input. Note: If required, we should be able to recreate the original tree using this dictionary.

Let's see how to achieve this considering the below tree as an example:



Defining the key: For a given node, consider **parent_key + '__' + 'node_key'** as the node key.

Flattened output dictionary for the above tree:

Do BFS(breadth-first search) traversal of the tree and capture the details as a dictionary.

Key	Value
1	[1__2, 1__3]
1__2	[1__2__4, 1__2__5]
1__3	[1__3__6, 1__3__7]
1__2__4	[1__2__4_8, 1__2__4__9]
1__2__5	[1__2__5__10, 1__2__5__11]
1__3__6	[1__3__6__13]
1__3__7	[1__3__7__14]
1__2__4_8	[]
1__2__4__9	[]
1__2__5__10	[]
1__2__5__11	[]
1__3__6__13	[]
1__3__6__14	[]

Assignment Deliverables

Deliver the solution as source code located in a git repository. The Readme must contain a short description of the repo and Build and Run instructions

→ For any help please sen an email to backend@omnius.com ←

Resources

[1] https://raw.githubusercontent.com/omni-us/coding-challenges-resources/master/python-developer/api-response.json

[2] https://raw.githubusercontent.com/omni-us/coding-challenges-resources/master/python-developer/sample-1.json

[3] https://raw.githubusercontent.com/omni-us/coding-challenges-resources/master/python-developer/sample-2.json

[4] https://raw.githubusercontent.com/omni-us/coding-challenges-resources/master/python-developer/sample-3.json