Q1. Create a model in Django with a minimum of 4 attributes. Register that model in Django Admin. Make sure this Model instance is accessible by 1 user (on the admin site) at a time.

Example: User_1 logs in to Django Admin and accesses this model instance. Now User_1 can see all the fields and edit them. Meanwhile, User_2 opens the same model in Django admin. User_2 can't edit this model and all actions are disabled. As soon as User_1 closes this form (navigating away from form) User_2 can edit it.

Q2. Implementing search algorithms that provide relevant results is known to be a challenging task because of the complexity of any language. Please complete the below:

Part 1. Return relevant results considering the following points:

- Different words, sometimes, refer to a common concept. Eg.: men, man ; tree, trees;
- The same word has different meanings given the context. Eg: Free parking (no cost for the parking); Meat free restaurant (no meat dishes served).

    Explain theoretically, (without implementation) how would you use machine learning concepts and techniques to deliver relevant results considering the above points in a search engine context.

Part 2. A user might engage with the insydo website through different kind of activities.

- How to take into consideration his activities to recommend relevant content and a personalized experience.
- What activities should be tracked?

Q3. Find the attached data.txt file which contains a dump JSON for the Business Data and Keyword Data.

1. Read all the data and index that in Elastic Search.
2. Create a GET type Api in Django which takes a query parameter ('q') and search within the index data in Elastic search. e.g. http://localhost:8000/search/?q=cafe
3. Create one simple HTML page with only 1 search text box in it, and hit the above listed api from the search box text.

Q4. Optimize and correct the following Python code snippet.

The following method was implemented to shuffle the elements of a list with some restrictions but it's not working as expected.
Here are the restrictions:

- Elements on the list can move randomly to the left or to the right
- Elements must not move more than one position away from the original one
- The list is non circular.

How would you change the algorithm to work as expected?

Input Value = [21, 22, 23, 24, 25, 26, 27, 28]

Output Value: [22, 21, 24, 23, 26, 25, 28, 27]

```
def _shuffle_list_values(self):

   ids_list = [-1122, 2321, -9023, 2711, 8112, -0912, 2711, 9832]

   random_movements = {}
   for n_id in ids_list:
     movement = random.randrange(-1, 2, 1)
     random_movements[n_id] = movement
   left = -1
   right = 1
   shuffled_list = [x for x in ids_list]
   for n_id in ids_list:
     if not(ids_list.index(n_id) == 0 and random_movements[n_id] == left) or\
            not(ids_list.index(n_id) == len(ids_list) - 1 and random_movements[n_id] ==
right):
        shuffled_list.insert(ids_list.index(n_id) + random_movements[n_id],
shuffled_list.pop(ids_list.index(n_id)))
   return shuffled_list
```

Q5. Share an example of your work (code/repository) that you are proud of and tell us why.

Q6. (Bonus question): Write down the necessary test cases for Q1 and Q3.