Exercise 1

1. create a JSON file looking like that

Entrée [ ]:

```
{"users":[
    {
        "username": "David",
        "password": "12345",
        "email": "david@super.com",
        "message_received":[],
        "message_sent": []
    },
    {
        "username": "Mark",
        "password": "23412",
        "email": "mark@super.com",
        "message_received":[],
        "message_sent": []
    },
    {
        "username": "Jacob",
        "password": "23821",
        "email": "jacob@super.com",
        "message_received":[],
        "message_sent": []
    },
    {
        "username": "Joseph",
        "password": "62797",
        "email": "joseph@super.com",
        "message_received":[],
        "message_sent": []
    }
]
}
```

Part 1

1. create a function called log_in:
   - this function is going to look for every user in the 'database' (the json file)
   - it will do the same thing as the first part of exercise 6 of day2
     A. ask the user to give his credentials (username +password).
     B. Find him in the list and check if both of the username and the password are correct.
     C. if the password is not correct tell him to enter it again (give him 3 trials).
   - If the user is found then return it
   - if the username is not right, call a function called create_new_user()
2. the function create_new_user will ask the user to enter his credentials and create an account.
   - after the account is created append it to the network

        - this is what a account looks like : { "username": "Joseph", "password": "62797", "email": "joseph@super.com (mailto:joseph@super.com)", "message_received":[], "message_sent": [] }
    - this function should return the user that was just created

Part 2

1. create a function main that is going to do that:
    A. if the function log_in return a user then you should ask this user the following menu:
        a. Ask the connected user if he wants to read his messages or send one.
        b. if he want's to read them show all the content of his 'message_reveived'.
        c. if he wants to send a message first ask him to who.
        d. then ask him what he wants to write
        e. then complete the dictionary of a message ('from' is the connected user, 'to' is the one he wants to send it to)
        f. finally append this message to the 'message_sent' of the connected user and to the message received of the receiver.
2. create a function called save to json

    - this function will save the network list to the json file
    - you should call this function as soon as you make change in the network list(for exemple when we created a new user)

Exercise 2

Part 1

1. We are trying to recreate the rolling of dice.
2. Your code should keep throwing 2 dice until they both land on the same number.
3. It should keep throwing 2 dice (using your throw_dice function) until they both land on the same number (until we reach doubles). For example: (1, 2), (3, 1), (5,5) → then stop throwing, because doubles were reached.
4. We also want to keep track of the number of throws we had to do to get a double.

Part 2

1. Ask the user how many times does he want to throw dice.
2. Create a variable dictionary called keep_track. It should have three keys: number_of_throw, number_of_double, average_double
3. When we get to a double we want to keep throwing dice and add +1 to the number_of_doubles in our dictionary.
4. At the end calculate the average of double. (number_of_double/number_of_throw)
5. Show the results to the user. The output would show something like this:

```
    - Total throws: 8
```

Part 3

1. Save each 'keep_track' dictionary to a JSON file. It should look like that:

Entrée [ ]:

```
{
    'data':[
        {
            'number_of_throw':12,
            'number_of_double': 2,
            'average_double':0.16666
        },
        {
            'number_of_throw':12,
            'number_of_double': 2,
            'average_double':0.16666
        },
    ]
}
```

Part 4

1. Create a new python file
2. it should read the data from the previous JSON file
3. Calculate the average of double for all the throws (total of double/total of throws)

Entrée [ ]:

Entrée [ ]: