Documentation – pushTimeAnalysis.py

Introduction

The purpose of this program is to get from a database of users and messages the optimal time to send them a push notification.

Usage

This program was designed on Python 3.6.3.

It is designed to working with a database prototyped like the *message_read.csv* file.

Example:

	A	B	C	D	E	F
1	message_read_id,user_id,watermark,timestamp,datetime_created					
2	2436731,42966,1510041364408,1510041390566,2017-11-07 07:56:31.239347					
3	2436734,19771,1510041390166,1510041391151,2017-11-07 07:56:31.388501					
4	2436745,31285,1510041390519,1510041391861,2017-11-07 07:56:32.102914					
5	2436749,12478,1510040633529,1510041391934,2017-11-07 07:56:32.187304					
6	2436758,8034,1510041390507,1510041392480,2017-11-07 07:56:32.699831					
7	2436821,42966,1510041394531,1510041395688,2017-11-07 07:56:35.906531					

To run it, you can execute it on a console with the command:

```
py pushTimeAnalysis <path to the database.csv>
```

Output

The results of this program are displayed like:

user: id of the user **| time:** optimal time to send the push notification

Example:

```
$ py pushTimeAnalysis.py test.csv
pushTimeAnalytics > Database loading...
pushTimeAnalytics > Databases loaded.
pushTimeAnalytics > Analysis in progress...
pushTimeAnalytics > Results:
        user: 42966 | time : 10:46:00
        user: 41541 | time : 14:21:00
        user: 31285 | time : 10:46:00
        user: 17395 | time : 11:43:00
        user: 8034 | time : 10:46:00
        user: 5440 | time : 12:01:00
```

Algorithm

For this program I choose to make, for each user id, the average hour he reads his messages, with the timestamp, and send a push notification 10 minutes before.

For future improvements, it would be better to check at which hour the user is looking at his message per day and send it 10 minutes before.

Code explanation

Main

The main check if the user entered a file name and then calls the above functions one by one.

read_file(filename)

This function is used to read the database. It reads it line by line and store the columns of *user_id* and *timestamp* in a two-dimensional list called datas.

create_users_dic(datas)

This function creates a dictionary with as key the user's id and as values the different *timestamps* for each of them.

A first temporary dictionary is created with list as values which will be converted to tuples in last line for optimisation.

handle_users_timestamps(users)

This function is used to replace the list of *timestamps* in the *users dictionary* to a single time, which is the average hour and minute the user reads his messages minus 10 minutes.

It takes the hour and minute of each *timestamp*, convert it to seconds and then divide the seconds by the number of *timestamps*.

Extra functions

my_print(string)

This function takes a string in parameter and print it with the name of the program before. Then, it forces the program to print it.

error_handler(string)

This function is called when an error occurs, it prints the string in parameter and then closed the program with the exit code 1.