Project Big Data

Assignment 2

You will continue to work with the hue data files supplied for Assignment 1, `hue_upload.csv` and `hue_upload2.csv`. The first four columns represent the *row id*, *user id*, *event id*, and *value*. Any extra columns are irrelevant. For example, the first row of one file reads:

```
"1";"10";"lamp_change_29_mei_2015_19_08_33_984";"OFF"
```

As you can see, the event id encompasses both a description of the event ("lamp_change") and the date/time (May 29, 7:08:33 pm). The following events are considered informative:

String	Description
`lamp_change`	Light control via app
`nudge_time`	Automatic light dim time for people in experimental group
`bedtime_tonight`	Intended bedtime (self-reported)
`risetime`	Rise time (self-reported)
`rise_reason`	Reason for rising (self-reported)
`adherence_importance`	Adherence (self-reported)
`fitness`	Fitness (self-reported)

All self-reported values are entered around noon. Records with other events may be ignored.

(60 points) The first part of this assignment is to write a Python function `read_csv_data` that reads the data into a Pandas DataFrame. The index should be a (date,user) tuple, where date is stored in datetime.datetime format, see the note in the 'Tips' Section. The columns of your Pandas DataFrame should be `bedtime`, `intended_bedtime`, `rise_time`, `rise_reason`, `fitness`, `adherence_importance` and `in_experimental_group` (note: it is important to *stick to this nomenclature*). The way to do this is by going through the csv data line by line, and parsing each line individually, following these requirements:

bedtime should be inferred from the `lamp_change` event. For example, from the row printed above you may infer that the person did not sleep before 7:08:33 pm. As you go through the lines in the csv file, whenever you discover new relevant information, you either update an existing record in the dataframe (if a record for that day and user exists), or you create a new record (see the 'Tips' Section below). For example, if you encounter a line where user 10 turns the light on at 9 pm and another line where he turns it off at 10 pm (still on May 29), you update the record above to change the bedtime to 10 pm. If someone falls asleep past midnight, the bedtime should be stored in the record corresponding to the day before.

intended_bedtime should be filled in based on the `bedtime_tonight` event. Note that `1030` probably means 10:30 in the evening. Again, dates and times should be stored as datetime.datetime.

rise_time The value for the column in your solution should be obtained from the `risetime` event in the csv file.

rise_reason, fitness and adherence_importance values should be copied from the csv file. Note that if multiple distinct values are entered, the last should be assumed to be correct.

in_experimental_group should be boolean (True/False). The default value is False, but should be changed to True if a `nudge_time` event is encountered. If a user is in the experimental group on one day, he is on all days.

(10+20 points) The second part of this assignment is to store the contents of the DataFrame into MongoDB, and to write a function that retrieves data from MongoDB and outputs it in a user-friendly format.

1. The data should be stored in the collection "sleepdata" in the database "BigData". Make sure to use the same column names as specified for the DataFrame, and to define the correct primary key. See below

in the Section 'Tips' for some comments about the primary key. Add the extra columns "date", "user", "sleep_duration" to facilitate sorting the data. Here, "sleep_duration" is the difference between the rise time and the bed time.

2. The following is an example of how the output must be presented.

date	user	intended	actual	rise	fitn	adh	exp
11-06-2015	34	22:30:00	00:51:28	07:00:00	07:10:00	-	47.0
11-06-2015	20	23:00:00	00:28:10	55.0	88.0	x	

Another 10 points are awarded for the cleanliness of your code and the use of idiomatic Python. A total of 100 points can be obtained. The assignment should be done in groups of two students, and must be handed in via Blackboard on June 15 by 23:59h. Your solution should use the provided template (`solution.py`). It is *crucial* that you do not alter the names and arguments of the existing functions, although adding additional functions is recommended. The template contains the file `run_solution.py` that shows how we run your file. Each group should hand in only one solution. If both students submit a solution, only the first submission will be graded. Feedback will be provided to the e-mail addresses you provide in `solution.py`.

Tips

In the assignments, there are some hurdles that are hard to jump without further help.

- It is important to use the datetime.datetime datatype instead of the (seemingly more appropriate) datetime.date type, as the following example shows. The following code *does not* run as expected:

```
idx = (datetime.date(2015,1,1),10)
df = df.append(pd.Series('bedtime':None,'intended_bedtime':None, name=idx))
if idx in df.index:
    print("surprisingly, this line does not run")
```

  The reason is that pandas converts the datetime.date to a pandas datetime type. Since a comparison between a date and a datetime is always false, idx is not found in df.index. So, just stick to the datetime.datetime data type for now (we suspect it to be a bug in Pandas that will get fixed some day, but not soon enough for this course). The following code (the change is in the first line) *does run* as expected:

```
idx = (datetime.datetime(2015,1,1,0,0,0),10)
df = df.append(pd.Series('bedtime':None,'intended_bedtime':None, name=idx))
if idx in df.index:
    print("This gets printed (as expected)")
```

- The next hurdle is that in some configurations, calling `df.set_value` leads to the error "ValueError: could not convert string to float". It turns out to be difficult to pinpoint the exact cause of this error. As a workaround, we advise to use the following function:

```
def insert_if_new(df,idx):
    if idx not in df.index:
        df = df.append(pd.Series({'bedtime' : float('nan'), \
                                  'intended_bedtime' : float('nan'), \
                                  'risetime' : float('nan'), \
                                  'rise_reason' : float('nan'), \
                                  'fitness' : float('nan'), \
                                  'adherence_importance' : float('nan'), \
                                  'in_experimental_group' : False}, \
                                 name=idx))
    return df
```

  Mixing float with `nan` ensures that Pandas does not infer incorrect datatypes. Call this function to add a row, and use `set_value` to modify the dataframe.

- The third hurdle is setting the primary key in mongodb. As mongodb does not support tuples (just dicts and lists), the primary key has to be converted from a tuple to a dict (with strings 'date' and 'user' as keys).
- Good luck! Oh, for those in need, there is an official Pandas cheat sheet.