HW4

HW4-HW5-HW6 Overview

Data source

In this project you will write code to process data from 311 Service Requests data set available via OpenData Buffalo. This data set (which you will work with by the time you get to HW6) has over 800,000 data records.

For HW4 I suggest you work with an abbreviated version of this data set, containing a little over 1,000 data records, and available here: 311_Service_Requests_Abbreviated.csv

In fact, as you develop your own test cases for the required functions below you may wish to create even smaller versions of this data file for your own use.

In the complete homework progression you will have built a web app that can process this dataset and visualize things like:

> A histogram of the amount of time it takes to resolve a request
> How many requests there are per 'subject' (city department)
> How many requests there are per street
> How many requests there are per street per year
> How many requests there are per department per year
> How many requests there are per street per department per year
> etc.

You are encouraged to explore the dataset (you can open it in any spreadsheet) to get a feel for what it looks like.

Requirements

You must define, in Python, each of the following functions. They must all be defined in a file named `App.py`. Each function must be named exactly as given, have the inputs and output exactly as described, and the functionality indicated.

> **Added note (11-1-2020)**
>
> Note that we are requiring that you put the code in `App.py`. Since repl.it will run the code in `main.py`, how do you test the code in `App.py`? Put your testing code in

main.py, and also `import App` in `main.py`; call function `f` from `App.py` in `main.py` as `App.f`

For example, if `App.py` contains:

```
def f(x):
    return 2 * x + 1;
```

then `main.py` can use this function as follows:

```
import App

print(App.f(3))
```

readCSV

Define the function `readCSV(`*filename*`)` to so it reads the contents of the CSV file named *filename* and returns a list of dictionaries corresponding to its contents. The CSV file will have a header file with field names. Each dictionary in the returned list of dictionaries must consist of key-value pairs where the key is a field name. The field name must be paired with the corresponding data value from the record.

For example, if `sample.csv` contains:

```
a,b,c
1,2,3
4,5,6
7,8,9
```

then `readCSV('sample.csv')` must return this list:

```
[ { 'a':'1', 'b':'2', 'c':'3' },
  { 'a':'4', 'b':'5', 'c':'6' },
  { 'a':'7', 'b':'8', 'c':'9' }
]
```

writeCSV

Define the function `writeCSV(`*filename, list_of_dictionaries*`)` to so it writes data contained in *list_of_dictionaries* as a CSV file named *filename* and returns `None`. If the file already exists it must be overwritten with its new content. If the file does not already exist it must be created

with the new content. *list_of_dictionaries* will contain a list of dictionaries in the format returned by `readCSV`. The CSV file written to *filename* must have a header row containing the field names, followed by all the data rows represented by the values in each dictionary.

For example, if `lod` is:

```
[ { 'abby':'51', 'bell':'62', 'dee':'33', 'river':'97' },
  { 'abby':'54', 'bell':'65', 'dee':'26', 'river':'71' },
  { 'abby':'45', 'bell':'39', 'dee':'88', 'river':'22' },
  { 'abby':'57', 'bell':'68', 'dee':'39', 'river':'62' }
]
```

then `writeCSV('output.csv', lod)` must ensure that a file name *output.csv* has the following contents:

```
abby,bell,dee,river
51,62,33,97
54,65,26,71
45,39,88,22
57,68,39,62
```

keepOnly

Define the function `keepOnly(`*list_of_dictionaries*`, `*key*`, `*value*`)` so it returns a list of dictionaries which contains only those dictionaries from *list_of_dictionaries* which contain the *key:value* pairing.

For example, if `lod` is:

```
[ { 'abby':'51', 'bell':'62', 'dee':'33', 'river':'97' },
  { 'abby':'54', 'bell':'65', 'dee':'26', 'river':'71' },
  { 'abby':'45', 'bell':'39', 'dee':'88', 'river':'22' },
  { 'abby':'57', 'bell':'68', 'dee':'26', 'river':'62' }
]
```

then `keepOnly(lod, 'dee', '26')` must return

```
[ { 'abby':'54', 'bell':'65', 'dee':'26', 'river':'71' },
  { 'abby':'57', 'bell':'68', 'dee':'26', 'river':'62' }
]
```

discardOnly

Define the function `discardOnly(`*list_of_dictionaries*`, `*key*`, `*value*`)` so it returns a list of dictionaries which contains only those dictionaries from *list_of_dictionaries* which do not contain the *key:value* pairing.

For example, if `lod` is:

```
[ { 'abby':'51', 'bell':'62', 'dee':'33', 'river':'97' },
  { 'abby':'54', 'bell':'65', 'dee':'26', 'river':'71' },
  { 'abby':'45', 'bell':'39', 'dee':'88', 'river':'22' },
  { 'abby':'57', 'bell':'68', 'dee':'26', 'river':'62' }
]
```

then `discardOnly(lod, 'dee', '26')` must return

```
[ { 'abby':'51', 'bell':'62', 'dee':'33', 'river':'97' },
  { 'abby':'45', 'bell':'39', 'dee':'88', 'river':'22' }
]
```

filterRange

Define the function `filterRange(`*list_of_dictionaries*`, `*key*`, `*low*`, `*high*`)` so it returns a list of dictionaries which contains only those dictionaries *d* from *list_of_dictionaries* for which *low* <= *d[key]* < *high*

For example, if `lod` is:

```
[ { 'abby':'51', 'bell':'62', 'dee':'33', 'river':'97' },
  { 'abby':'54', 'bell':'65', 'dee':'26', 'river':'71' },
  { 'abby':'45', 'bell':'39', 'dee':'88', 'river':'22' },
  { 'abby':'57', 'bell':'68', 'dee':'26', 'river':'62' }
]
```

then `filterRange(lod, 'bell', '39', '68')` must return

```
[ { 'abby':'51', 'bell':'62', 'dee':'33', 'river':'97' },
  { 'abby':'54', 'bell':'65', 'dee':'26', 'river':'71' },
  { 'abby':'45', 'bell':'39', 'dee':'88', 'river':'22' }
]
```

duration

The datetime Python module makes it easy to do computations with dates. For this project you will need to compute the duration of a request (each record in the CSV file, and therefore each dictionary in the list of dictionaries, corresponds to one request).

You can read the documentation for the datetime module, but what you need to know is how to create a date object, how to compute the difference between two dates (called a timedelta), and how to get the number of days in a timedelta. This short example should give you the basic idea:

```
> import datetime
> d1 = datetime.date(2016,5,3)
> d2 = datetime.date(2016,11,21)
> diff = d2 - d1
> diff
datetime.timedelta(days=202)
> diff.days
202
```

Define the function `duration(`*date1*`, `*date2*`)` so that it accepts two datetime.date objects as arguments and return the number of days between the two dates, as an integer.