CS 101 - Attendance
Spring 2017
Algorithm Due : **April 5th, 2017**
Program Due : **April 9th, 2017**
All work submitted must be your own.
Deliverables : You only need to submit your solution. You should not submit the graphics.py module. You must use functions to modularize your work. You should use exception handling where necessary as well.

Attendance csv

You are going to write a program that can open a csv file (comma separated values file). Each row of the attendance file contains 6 fields all separated by a comma. This row is the information about a given student, class, and date and whether they were attending or not. If you open it in excel you won't see the commas and it will separate the columns into the excel columns. Be careful to not save it in excel as it will change the formatting.

```
CS291,11/07/2016,10:00 AM,"Payne, Michele",863360,Absent
CS101,10/05/2016,3:00 PM,"Snow, Jose",486163,Present
CS191,09/28/2016,3:00 PM,"Sexton, Dora",708846,Present
CS201,08/24/2016,11:00 AM,"Miller, Amber",161920,Present
CS191,08/24/2016,3:00 PM,"Moore, Jerry",175297,Present
CS191,09/07/2016,3:00 PM,"Hand, Benjamin",265451,Present
```

Your boss wants you to read in any csv file and output it into a file that is json formating so it can be displayed in nice HTML graph. Json stands for Javascript Object Notation. There are 2 different files they might want. One is a file that contains the date of attendance in a particular class with the percentage of students that were present that day. The other, is to create a all the dates of attendance with the attendance for each class on that date. Taking input file and outputting data in a different format is pretty common in the work environment.

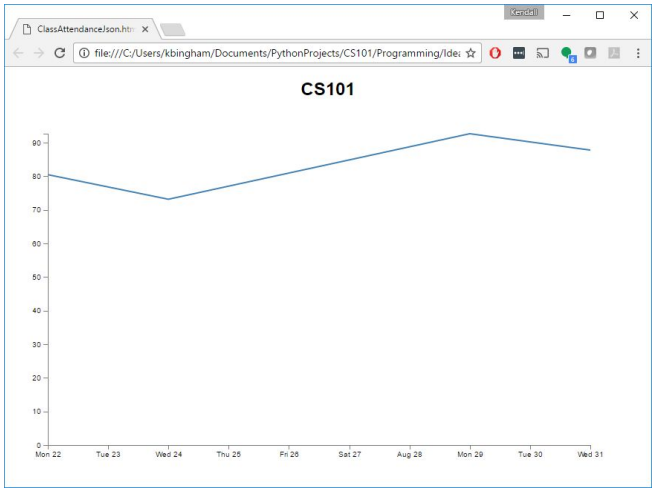
data.js

This javascript file is used by the ClassAttendance.html to show the attendance for a single class. You'll have to aggregate all the information from the attendance file for the specified class and calculate the attendance percentage for each day. Below is an example of the required format.

```
var title = "CS101";
var json_data=
[{"attendance": 80.48780487804879, "date": "08/22/2016"},
 {"attendance": 73.17073170731707, "date": "08/24/2016"},
 {"attendance": 92.6829268292683, "date": "08/29/2016"},
 {"attendance": 87.8048780487805, "date": "08/31/2016"}];
```

The format of this file is very specific and must match it exactly. Title will be set to whichever

class the user selects during the program. Once this datafile is generated you can open the class attendance.html file to see a nice graph of the output.



Multidata.js

When the user wants you to generate the multidata.js file from the input, then you will generate a javascript file that has the aggregate information for all the classes. This means for each class you'll have to aggregate it down to the percentage

```
var json_columns = ["date", "CS101", "CS191", "CS201", "CS281", "CS291"];

var json_data =
[{"CS281": 0.5675675675675675, "CS291": 0.7027027027027027, "CS201":
0.7027027027027027, "CS191": 0.7555555555555555, "CS101": 0.8048780487804879,
"date": "08/22/2016"},
 {"CS281": 0.7027027027027027, "CS291": 0.6756756756756757, "CS201":
0.7027027027027027, "CS191": 0.8, "CS101": 0.7317073170731707, "date": "08/24/2016"},
 {"CS281": 0.6216216216216216, "CS291": 0.6486486486486487, "CS201":
0.7837837837837838, "CS191": 0.8444444444444444, "CS101": 0.926829268292683, "date":
"08/29/2016"}];
```

Run the multipleAttendance.html file to see the results of that javascript data file.

There are a lot of new skills to build. So practice many of these things separately to understand how to put them together better.

Csv Module

The csv module allows you to read and write to csv files very easily. It will take care of a lot of the background work for you. You just have to import it. Once you open a file to read, you can pass that file handle to the csv.reader() function and get a csv handle. If you iterate over that, then each line gets split into a list for you automatically.

```
import csv

file = open("test.csv")

csv_file = csv.reader(file)

for line in csv_file:  # each line is a list of strings.
    print(line)
    print(line[0], line[1])

file.close()
```

Datetime module

Working with dates and times can be a huge pain. But, once again Python comes to the rescue with a module to make it easier. The datetime module will help you convert from strings to an actual date object. Since the date in the csv file is in a string format, in order to sort it or manipulate it like a date we need to convert it appropriately. We'll also want to convert a date from the datetime object into a string representation that we like. The sample below shows why

this is important. If we sort a list of strings that are dates, then it doesn't sort it in date order. If we sort a list of datetimes, then it sorts appropriately. We can convert from a string date to a date by calling datetime.datetime.strptime(value, format). Where format is the format of the string we are trying to convert. If the date is in MM/DD/YYYY format then the format passed should be %m/%d/%Y. See references below for datetime module and other formats you can convert.

```
>>> import datetime
>>> dates = ["12/14/2008", "5/2/2000", "5/12/2000", "8/9/2016"]
>>> datetimes = []
>>> for date in dates:
        date_convert = datetime.datetime.strptime(date, "%m/%d/%Y")
        datetimes.append(date_convert)


>>> dates
['12/14/2008', '5/2/2000', '5/12/2000', '8/9/2016']

>>> datetimes
[datetime.datetime(2008, 12, 14, 0, 0), datetime.datetime(2000, 5, 2, 0, 0),
datetime.datetime(2000, 5, 12, 0, 0), datetime.datetime(2016, 8, 9, 0, 0)]

>>> dates.sort()
>>> dates
['12/14/2008', '5/12/2000', '5/2/2000', '8/9/2016']
>>> datetimes.sort()
>>> datetimes
[datetime.datetime(2000, 5, 2, 0, 0), datetime.datetime(2000, 5, 12, 0, 0),
datetime.datetime(2008, 12, 14, 0, 0), datetime.datetime(2016, 8, 9, 0, 0)]
>>>
```

Notice after the string dates are sorted they are in order by the string sort, which isn't correct for dates, but the datetimes list is sorted because it knows how to sort a date.

If we want to convert the datetime value from a date to output it to the user we can call strftime on the date object.

```
import datetime

>>> date_str = "12/08/2017"
>>> date = datetime.datetime.strptime(date_str, "%m/%d/%Y")
>>> date
```

```
datetime.datetime(2017, 12, 8, 0, 0)
>>> print(date.strftime("%m%d%Y"))
12082017
>>> print(date.strftime("%m/%d/%Y"))
12/08/2017
>>>
```

If you try to convert a date that is in the wrong format, you'll get an error. You want to catch those exceptions in the file and warn the user that the data file may be corrupt or have bad data.

```
>>> date = datetime.datetime.strptime("badformat", "%m/%d/%Y")
Traceback (most recent call last):
  File "<pyshell#27>", line 1, in <module>
    date = datetime.datetime.strptime("badformat", "%m/%d/%Y")
  File "C:\Python34\lib\_strptime.py", line 500, in _strptime_datetime
    tt, fraction = _strptime(data_string, format)
  File "C:\Python34\lib\_strptime.py", line 337, in _strptime
    (data_string, format))
ValueError: time data 'badformat' does not match format '%m/%d/%Y'
>>>
```

Json module

The json module helps you create json strings from normal python data structures. This can help you create the strings that you'll want to save out to the file. Doing it manually is extremely difficult, but doable. Make sure you play around with different datatypes in order to figure out how to get it into a string that is in the required format.

json.dumps(object) This function takes a python object, and converts it into a json string.

```
>>> import json
>>> my_list = ["Hello", 2, "World"]
>>> s1 = json.dumps(my_list)
>>> print(s1)
["Hello", 2, "World"]
>>> my_dict = {"Key":2, "Key2":"Value", "Key3":["A", "List", "as", "a", "value"]}
>>> s1 = json.dumps(my_dict)
>>> print(s1)
{"Key3": ["A", "List", "as", "a", "value"], "Key": 2, "Key2": "Value"}
>>>
```

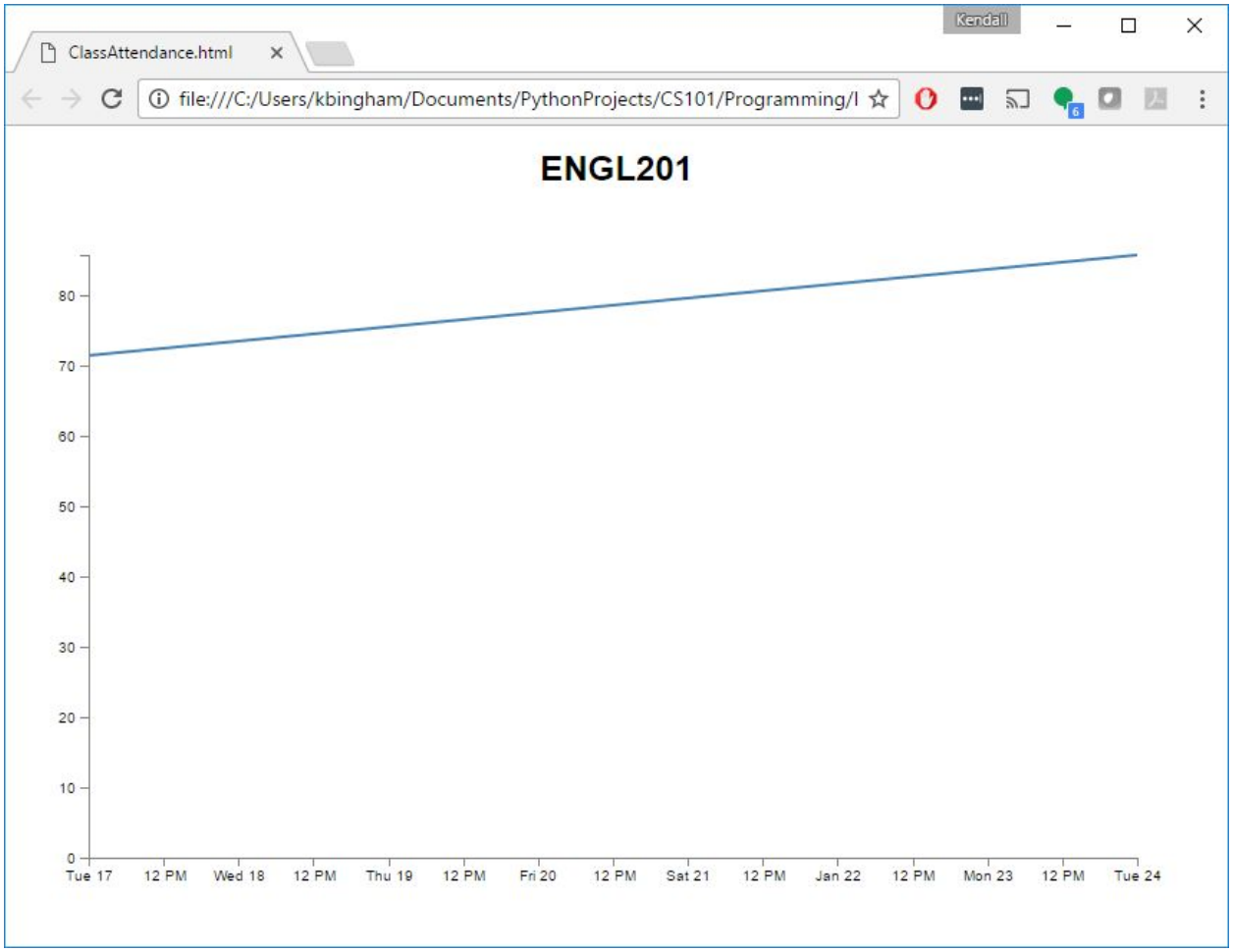
webbrowser module

After you generate the js file for a given html page, you can use the webbrowser module to automatically open the browser window to the URL. If the html file is in your current directory just give the file name for the url.

```
>>> import webbrowser
>>> webbrowser.open("ClassAttendanceJson.html")
True
>>> webbrowser.open("http://sce.umkc.edu")
True
>>>
```

Sample Program

```
>>> ================================ RESTART ================================
>>>

                Attendance Data File Generator

You can create a single data file of attendance for a single class or for multiple

1. Read an attendance csv file and create a single class output. ( data.js)
2. Read an attendance csv file and create a multiple class aggregate output  (multidata.js)
Q. Quit

==> e
You must choose a valid option from (1,2,Q)

                Attendance Data File Generator

You can create a single data file of attendance for a single class or for multiple

1. Read an attendance csv file and create a single class output. ( data.js)
2. Read an attendance csv file and create a multiple class aggregate output  (multidata.js)
Q. Quit

==> 1
Enter the csv file name ==> badfile.csv
Could not open the file, please try again
Enter the csv file name ==> attendance_bad.csv
The file given has an invalid date column.  Please enter another

Enter the csv file name ==> attendance_math.csv
Enter the class you want exported to data.js ==> noclass
Class noclass is not in the attendance file.  You must choose from ENGL101,ENGL201.

Enter the class you want exported to data.js ==> engl201
```




```
                Attendance Data File Generator

You can create a single data file of attendance for a single class or for multiple

1. Read an attendance csv file and create a single class output. ( data.js)
2. Read an attendance csv file and create a multiple class aggregate output  (multidata.js)
Q. Quit

==> 2
Enter the csv file name ==> attendance.csv
```


```
                Attendance Data File Generator

You can create a single data file of attendance for a single class or for multiple

1. Read an attendance csv file and create a single class output. ( data.js)
2. Read an attendance csv file and create a multiple class aggregate output  (multidata.js)
Q. Quit

==> e
You must choose a valid option from (1,2,Q)

                Attendance Data File Generator

You can create a single data file of attendance for a single class or for multiple

1. Read an attendance csv file and create a single class output. ( data.js)
2. Read an attendance csv file and create a multiple class aggregate output  (multidata.js)
Q. Quit

==> q
```

Specification

- The user must choose a valid menu item. They can choose to output to the single class js file (data.js) or the aggregate output file called multidata.js.
- If the user chooses quit from the menu the program ends
- If the user chooses 1 or 2 then it asks for the attendance csv file to open.

    - If the file doesn't exist it warns the user and asks again.
    - If the file has a bad date format, the user is warned and it prompts again.
- If the user chooses 1,
    - The program will ask for a valid class from the attendance file. If the class specified doesn't exist, it will prompt the user for the class again.
    - The program will output the properly formatted aggregate of the attendance out for that class to data.js
    - Once the file is output it will open the ClassAttendance.html file in the web browser.
- If the user chooses 2
    - The program will output the aggregate attendance to multidata.js
    - Once the file is output it will open the MultipleAttendance.html file in the web browser.

Point Breakdown - May be modified as needed

Points	Requirement
5	Header
10	Readability, variable naming, comments
5	Data Structures
5	Proper functions
4	Show Menu and get only valid choice
7	Ask for properly read attendance file from user. Handle errors.
5	If input file has an invalid date, then warn user and ask for input again.
7	If the user chooses 1, then output properly to the data.js file.
7	If the user chooses 2, then output properly to the multidata.js
5	Sort output data by date.

30 points off for programs that crash on expected input.

References

1. Csv Module https://docs.python.org/3/library/csv.html
2. Datetime module https://docs.python.org/3/library/datetime.html
3. Datetime strptime() behavior https://docs.python.org/3/library/datetime.html#strftime-strptime-behavior
4. Json module https://docs.python.org/3/library/json.html
5. Webbrowser Module https://docs.python.org/3.6/library/webbrowser.html