**Hot Weather Home Work**

Today you will be working with live weather data. We will collect temperature data of our three campuses, Budoora, Clayton, and Caulfield. I have written the code for you to collect the live data.

/// Python program to collect live weather data.

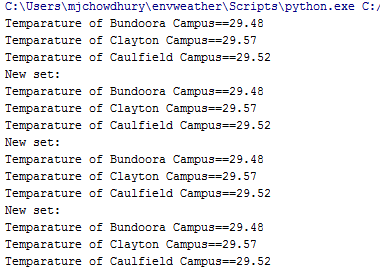
**import** requests  
**import** time  
  
latrobe\_bundoora=**'http://api.openweathermap.org/data/2.5/weather?zip=3086,au&units=metric&APPID=d69a4b6015c26ca2ef20c16aecdeaee8'**monash\_clayton=**'http://api.openweathermap.org/data/2.5/weather?zip=3800,au&units=metric&APPID=d69a4b6015c26ca2ef20c16aecdeaee8'**monash\_caulfield=**'http://api.openweathermap.org/data/2.5/weather?zip=3145,au&units=metric&APPID=d69a4b6015c26ca2ef20c16aecdeaee8'** *#make a request to the collect the dat*res\_bundoora = requests.get(latrobe\_bundoora)  
res\_clayton = requests.get(monash\_clayton)  
res\_caulfield = requests.get(monash\_caulfield)  
  
*#get data as json format*data\_bundoora =res\_bundoora.json()  
data\_clayton = res\_clayton.json()  
data\_caulfield = res\_caulfield.json()

#collect the temperature data  
temperature\_bundoora = data\_bundoora[**'main'**][**'temp'**]  
temperature\_clayton = data\_clayton[**'main'**][**'temp'**]  
temperature\_caulfield = data\_caulfield[**'main'**][**'temp'**]

# printing the temperature data in every 10 minutes 5 second.

#The website publishes new data in every 10 minutes  
**while True**:  
 print(**'Temparature of Bundoora Campus=='** + str(temperature\_bundoora))  
 print(**'Temparature of Clayton Campus=='** + str(temperature\_clayton))  
 print(**'Temparature of Caulfield Campus=='** + str(temperature\_caulfield))  
 time.sleep(605)  
 print(**'New set:'**)

Output of this program will look like this,



Now your job will be to extend this program.

Note: you have to install requests (pip3 install requests) to run this code.

**Kafka Group:**

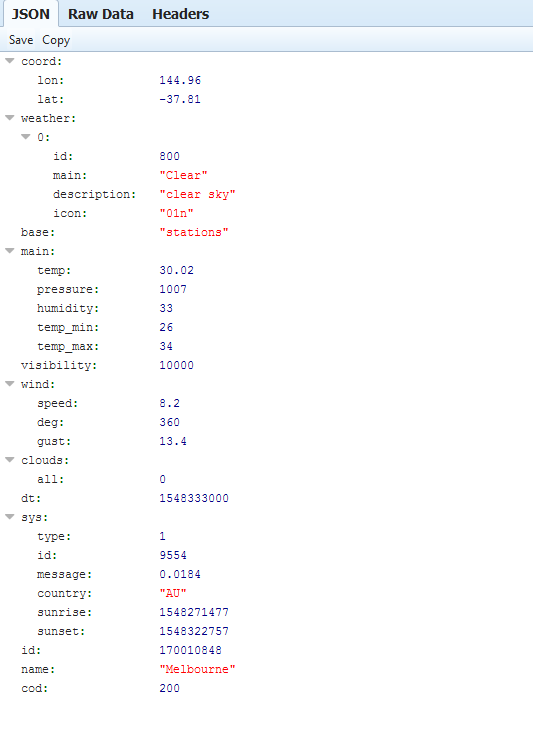
1. Draw graph from these three data series. As the data point may be very similar, so find out the appropriate scaling (y-axis) so that it looks interesting.
2. In your window, it will always show the three data point max, min, and current for each campus. But the max and min temperature may not be from the current window, rather it will be the max and min temperature till the program is running.
3. Your program will store your max and min temperature and current time into the mongoDb database whenever it happens. If the value changes then you have to update the value in the database. The schema of the database can be like,

temp {{campus: xxx},{max: value,time:value},{{min:value,time:value}}

1. Your program will show the max and min value and when that happen at 6PM for every campus and terminate (e.g, while condition).
2. Finally, you will print which campus was the coolest by printing the minimum of three campuses.

**Spark Group:**

1. Draw diagram out of these three data series.
2. If you run the Print res\_bundoora.json() you will see many types of data other than temperature. You can use pprint to see json respone. To use pprint install pprint.



You have to store all the data in the mongodb database. In addition to these data you also have to store the current time of every insert.

1. From the data, you have to find max and min and time(of max and min) using query to the database. For all three campuses.
2. Plot a heat map of your data at 6PM for all three campuses.

(Resource: <https://matplotlib.org/gallery/images_contours_and_fields/image_annotated_heatmap.html>)

1. Finally, you will print which campus was the coolest by calculating the average value. In addition, you will also print the minimum temperature campus and maximum temperature campus at 6 PM.