PROJECT NAME

By Juee Modi, Charlotte Moss, and Dora Hu

https://github.com/jueemodi/si206final.git

Original Goals

We originally wanted to use three API's that have to do with COVID-19 in different countries. We wanted to see how things like the economy or the air quality in a country correlate with the number of COVID cases. We hoped to use the programming skills we learned in SI206 to collaborate on a project that was interesting to all of us.

Goals Achieved

APIs used: World Bank's indicator API, COVID Data API, Air quality API

Our goals changed as we progressed to better accommodate our APIs and report meaningful data. Instead of connecting the 3 APIs, we found global averages across our APIs, such as the global average recovered from COVID-19 and the global air pollution levels. We collected the number of deaths, cases, recovered, and tests per million from the COVID API, as well as the continents that those countries are located in. From this API, we collected the average deaths per million, average cases per million, and average deaths per million in Europe using a join statement. From the air quality API, we collected the average PM2.5, PM10, NO2, SO2, and CO per country for the year 2020. We then calculated the average global pm2.5 and average global pm10 from the pollutant data we had gathered per country.

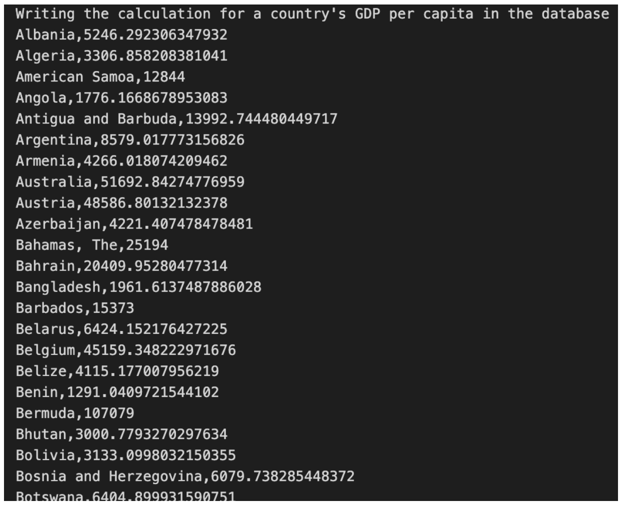
Problems

- For the World Bank API, the only way to get the data is for the user to input the page number of the base url so the API knows to flip the page (since there's so many countries). I tried things like creating a global variable, but it just didn't work as well. I went to office hours and discovered that the most elegant way to do it is by asking the user for the page number they want. Another problem that I ran into was that for some countries, there is population data but no GDP data. My solution was to loop through the GDP json file and if the GDP for a country is None, then I will continue to go on to the next country.

- For the COVID API, I had trouble only downloading 25 items to the database at a time since this API did not have a limit parameter. Since my items (countries) were stored in a list, the way I solved this problem was by using if/elif/else statements to use certain slices of the string depending on how many rows were currently in the database. I created a getCount() function that runs in main before my createCountryTable() function in order to have an accurate row count, which then gets passed into the createCountryTable() function. I also had a problem where certain countries did not have a continent in their data, so my code would stop running after it reached Australia. The way I fixed this was adding an else statement when assigning continent IDs, so the code would not stop if certain countries did not have a continent value.
- For the Air Quality API, I struggled to find which countries within my API had enough data to put into the database. Because each country code had to be manually entered in order to get the data for that country, I did not have a way to figure out which countries in the world had the highest number of pollutant data available, so I had to manually enter them in order to test whether they would show up in the database. Although the PM2.5 and PM10 were available for most countries, NO2, SO2, and CO, the three other major pollutants that I chose, did not have data for several of the countries. I decided to fix this problem by marking the pollutants with their values and adding "NULL" as a default to avoid any errors in code.

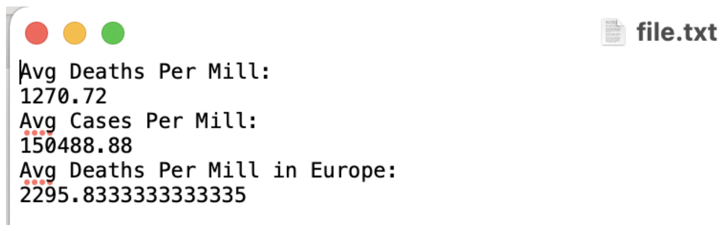
Calculations

World Bank API:



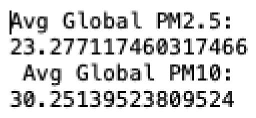
```
Writing the calculation for a country's GDP per capita in the database
Albania,5246.292306347932
Algeria,3306.858208381041
American Samoa,12844
Angola,1776.1668678953083
Antigua and Barbuda,13992.744480449717
Argentina,8579.017773156826
Armenia,4266.018074209462
Australia,51692.84274776959
Austria,48586.80132132378
Azerbaijan,4221.407478478481
Bahamas, The,25194
Bahrain,20409.95280477314
Bangladesh,1961.6137487886028
Barbados,15373
Belarus,6424.152176427225
Belgium,45159.348222971676
Belize,4115.177007956219
Benin,1291.0409721544102
Bermuda,107079
Bhutan,3000.7793270297634
Bolivia,3133.0998032150355
Bosnia and Herzegovina,6079.738285448372
Botswana,6404.899931590751
```

COVID API:



```
Avg Deaths Per Mill:
1270.72
Avg Cases Per Mill:
150488.88
Avg Deaths Per Mill in Europe:
2295.8333333333335
```

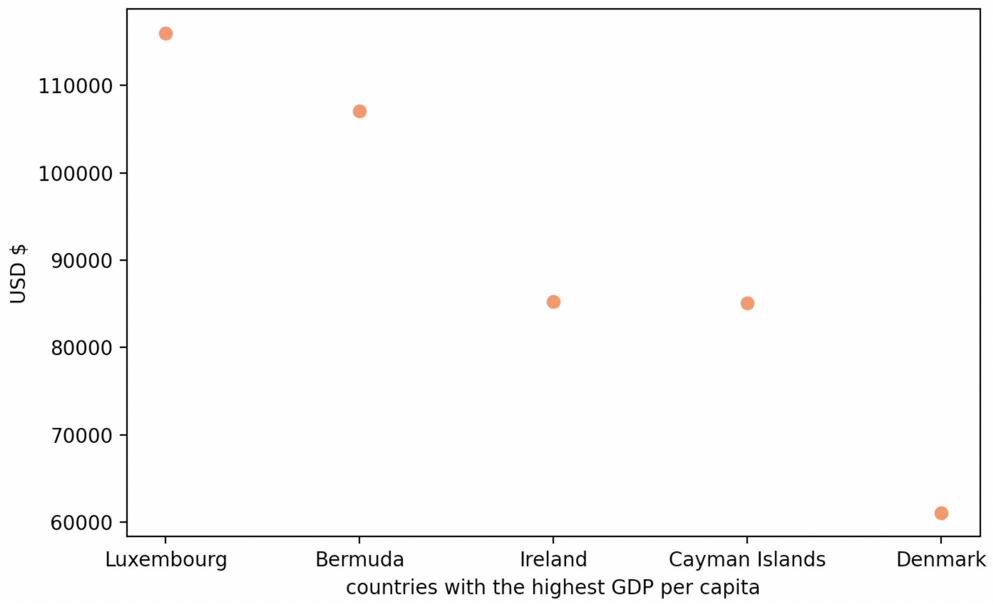
Air Quality API:



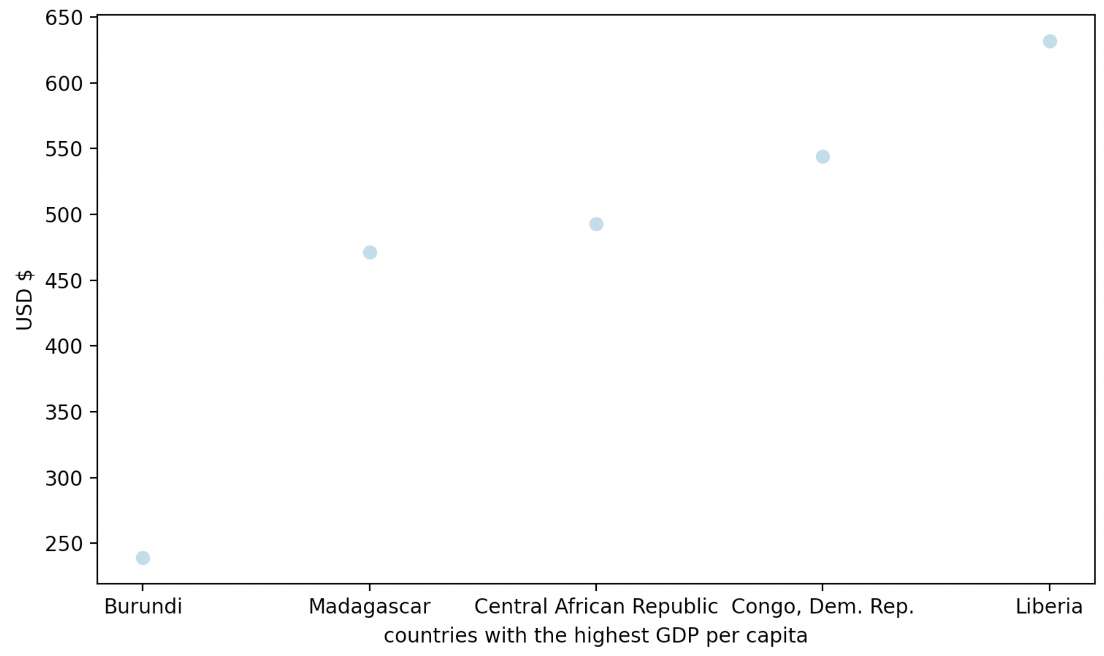
```
Avg Global PM2.5:
23.277117460317466
 Avg Global PM10:
30.25139523809524
```

Visualizations

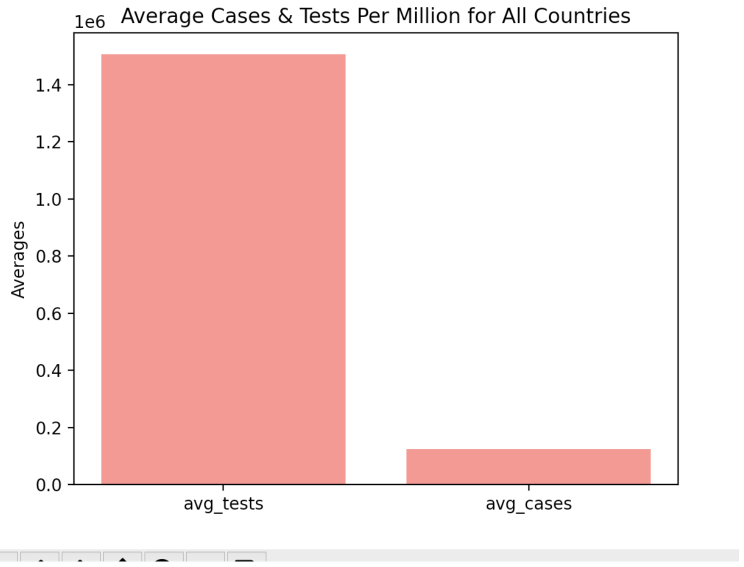
World Bank graph 1:



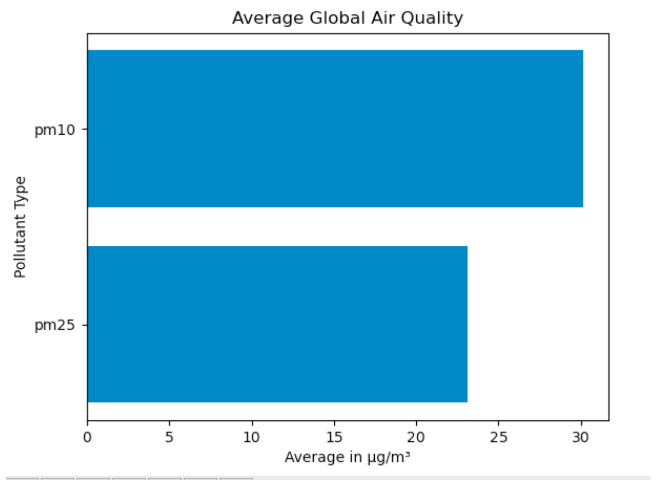
World Bank graph 2:



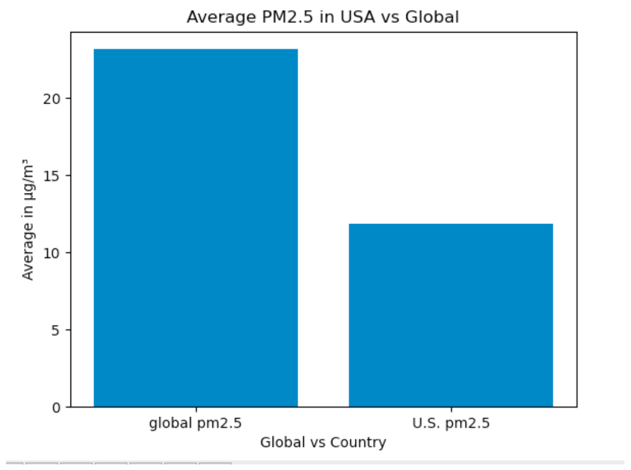
COVID Api Graph: 1



Air Quality Data Graph 1:



Air Quality Data Graph 2:



Instructions for running our codes

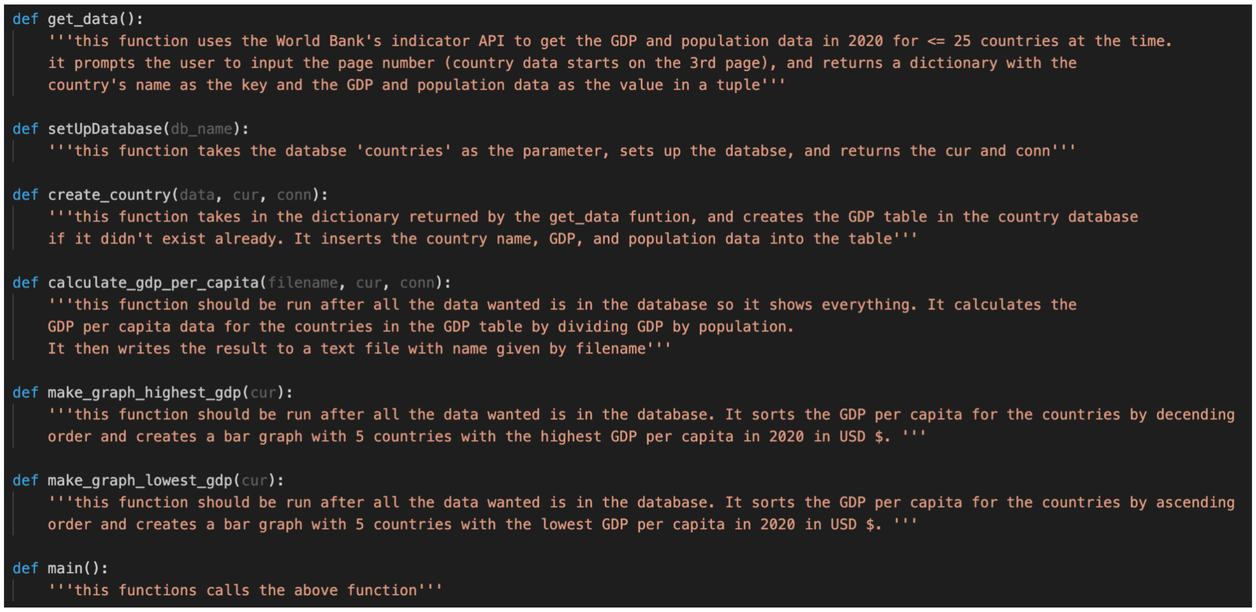
World Bank API: When getting data into the db, it prompts the user to input a page number from which to get the data (each time gets <= 25, I made sure with Prof. Erickson that this is okay). The data for individual countries start on page 3, so input 3 to 7, which will get you a little more than 100 countries worth of data.

COVID Data API: Every time the getdata file is run it will input 25 items into the database: there are about 200 countries so run the file 8 times to get all of the countries in the database. Then, run the readdata file to output the calculation to a file and show the bar chart.

Air Quality Data API: When running the file, 25 countries and their pollutant data will be input into the database. It may take a few runs for all 120ish countries to show up in the database. While calculations and/or potential errors in graphs may show up on the first few runs, once all countries have been inputted in the database, the calculations and graphs should be accurate. There may be minor changes in the calculation data due to the constant updating of the API.

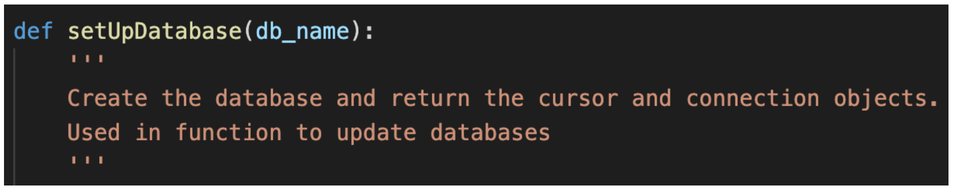
Documentations for our functions

World Bank API

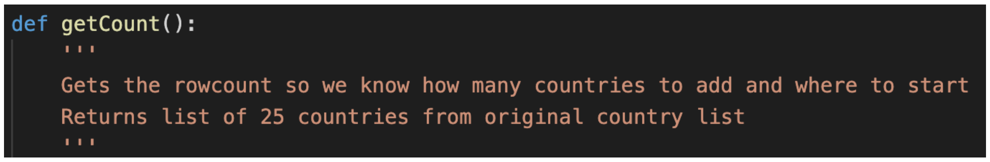


```
def get_data():
    '''this function uses the World Bank's indicator API to get the GDP and population data in 2020 for <= 25 countries at the time.
    it prompts the user to input the page number (country data starts on the 3rd page), and returns a dictionary with the
    country's name as the key and the GDP and population data as the value in a tuple'''

def setUpDatabase(db_name):
    '''this function takes the databse 'countries' as the parameter, sets up the databse, and returns the cur and conn'''

def create_country(data, cur, conn):
    '''this function takes in the dictionary returned by the get_data funtion, and creates the GDP table in the country database
    if it didn't exist already. It inserts the country name, GDP, and population data into the table'''

def calculate_gdp_per_capita(filename, cur, conn):
    '''this function should be run after all the data wanted is in the database so it shows everything. It calculates the
    GDP per capita data for the countries in the GDP table by dividing GDP by population.
    It then writes the result to a text file with name given by filename'''

def make_graph_highest_gdp(cur):
    '''this function should be run after all the data wanted is in the database. It sorts the GDP per capita for the countries by decending
    order and creates a bar graph with 5 countries with the highest GDP per capita in 2020 in USD $. '''

def make_graph_lowest_gdp(cur):
    '''this function should be run after all the data wanted is in the database. It sorts the GDP per capita for the countries by ascending
    order and creates a bar graph with 5 countries with the lowest GDP per capita in 2020 in USD $. '''

def main():
    '''this functions calls the above function'''
```

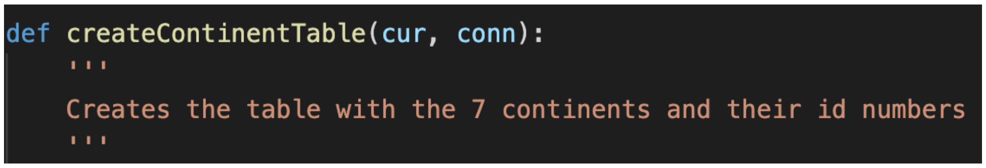
Covid_gettingdata



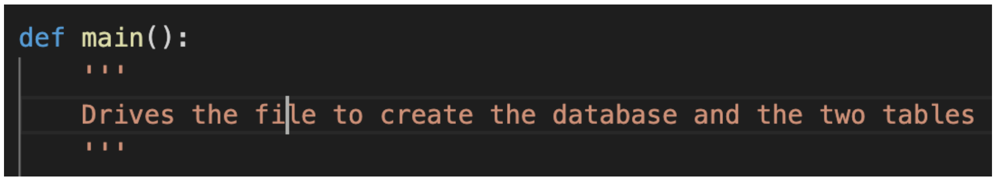
```
def setUpDatabase(db_name):
    '''
    Create the database and return the cursor and connection objects.
    Used in function to update databases
    '''
```



```
def getCount():
    '''
    Gets the rowcount so we know how many countries to add and where to start
    Returns list of 25 countries from original country list
    '''
```

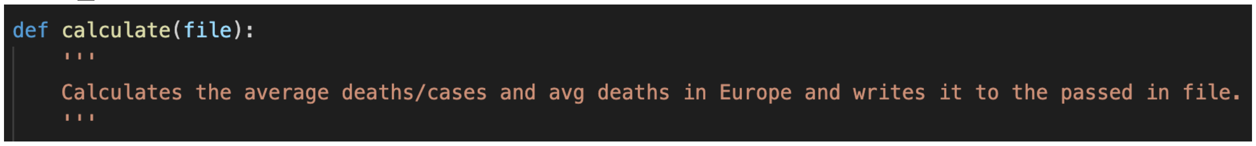


```
def createContinentTable(cur, conn):
    '''
    Creates the table with the 7 continents and their id numbers
    '''
```

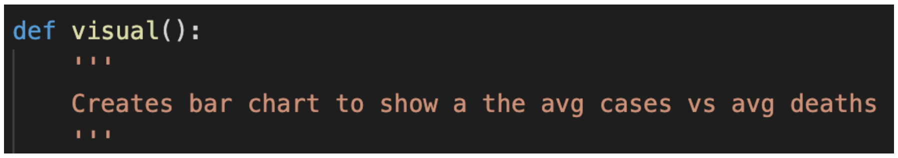


```
def main():
    '''
    Drives the file to create the database and the two tables
    '''
```

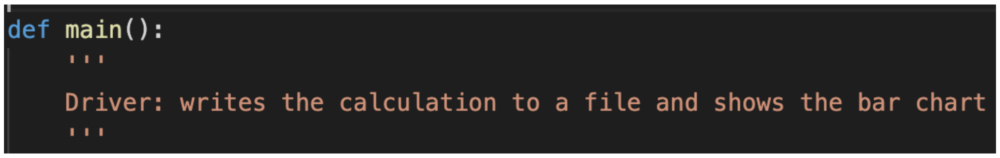
Covid_readdata



```
def calculate(file):
    '''
    Calculates the average deaths/cases and avg deaths in Europe and writes it to the passed in file.
    '''
```

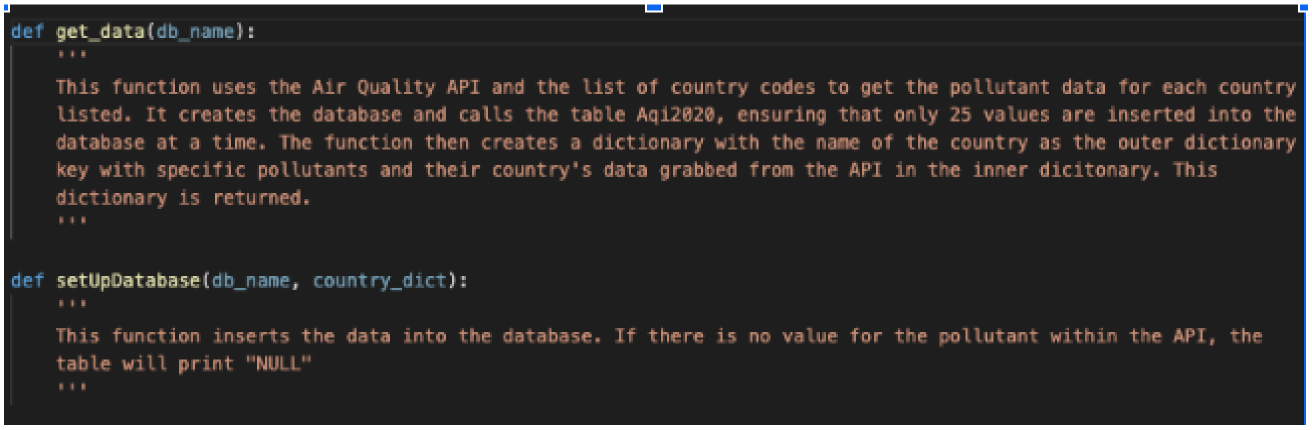


```
def visual():
    '''
    Creates bar chart to show a the avg cases vs avg deaths
    '''
```

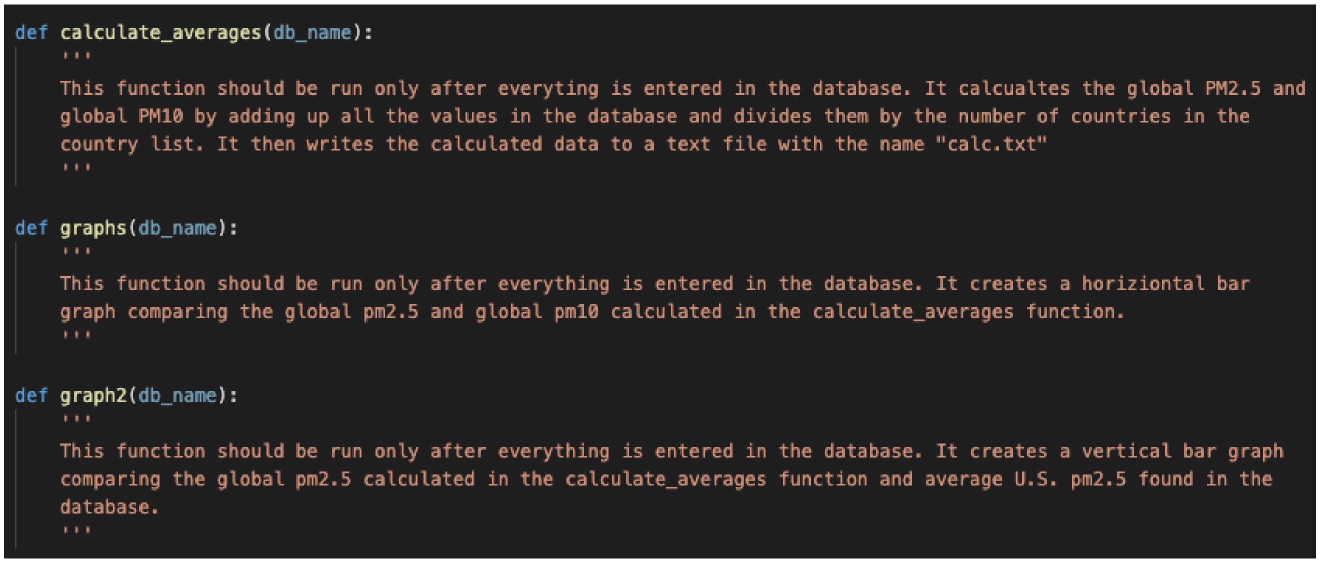


```
def main():
    '''
    Driver: writes the calculation to a file and shows the bar chart
    '''
```

Air Quality API



```
def get_data(db_name):
    '''
    This function uses the Air Quality API and the list of country codes to get the pollutant data for each country
    listed. It creates the database and calls the table Aqi2020, ensuring that only 25 values are inserted into the
    database at a time. The function then creates a dictionary with the name of the country as the outer dictionary
    key with specific pollutants and their country's data grabbed from the API in the inner dicitonary. This
    dictionary is returned.
    '''

def setUpDatabase(db_name, country_dict):
    '''
    This function inserts the data into the database. If there is no value for the pollutant within the API, the
    table will print "NULL"
    '''
```



```
def calculate_averages(db_name):
    '''
    This function should be run only after everyting is entered in the database. It calcualtes the global PM2.5 and
    global PM10 by adding up all the values in the database and divides them by the number of countries in the
    country list. It then writes the calculated data to a text file with the name "calc.txt"
    '''

def graphs(db_name):
    '''
    This function should be run only after everything is entered in the database. It creates a horiziontal bar
    graph comparing the global pm2.5 and global pm10 calculated in the calculate_averages function.
    '''

def graph2(db_name):
    '''
    This function should be run only after everything is entered in the database. It creates a vertical bar graph
    comparing the global pm2.5 calculated in the calculate_averages function and average U.S. pm2.5 found in the
    database.
    '''
```

Resource used

Date	**Issues Description**	**Location of Resources**	**Result (Did it solve the issue?)**
4/9	I was having trouble finding how many times the get-data() function was run to determine what page number to pass in as part of the request url.	https://stackoverflow.com/questions/423379/using-global-variables-in-a-function	Yes, I declared TIMES as a global variable and increment it everytime get_data() runs.
4/18	Related to the previous issue. I realized that without a user input page_number, I cannot put <= 25 data into the database each time	Office Hours	Yes, I got rid of the TIMES global variable and prompted the user for a page_number each time
4/14(juee)	Also having trouble with the 25 limit	Office hours	Solved: the instructor helped me use slices of my country list in order to add 25 each time depending on how many rows there

			were already in the database
4/14	I was having trouble getting the GDP data and the population data for a country at the same time. Some countries have the population data but not the GDP data.	I just tried different things with my code	Yes. I was able to have a variable to record which country I am getting. If it doesn’t have GDP data, I used continue to skip it.
4/20	Returning tuple rather than string or integer	https://www.reddit.com/r/Python/comments/dps1g7/python_mysql_query_returns_tuple_and_not_string/	Solved.