INST 326 – Object-Oriented Programming for Information Science

Team Project Annotated Bibliography

December 20, 2020

Team 107: Kristen Wagner, Chandra Tamang, Minsung Kim, Diego Amores

COVID Tracker Application Documentation

****Important!! Please run our pytest first!! Download and save the CSV file (Updated_Report.csv) and txt file (new_data.txt) inside the working directory. Then, make a copy of that CSV file outside the working directory. Once the pytest has been run, delete the CSV file inside the working directory so that the functions can properly run inside the command line of the terminal and for the application to start. If you wish to rerun pytest you must put that CSV, which made a copy of back into the program for it to run our tests script.****

Introduction:

Our team has built a COVID-19 Tracker App to make users' lives little easier during this global pandemic. Our app allows users to get the latest COVID-19 information in various ways. This app has many different functions built for our user. By using an interactive command line, users can get the number of deaths for each state, number of total cases by specific dates, number of cases reported, see Covid-19 data visually, and so on.

Required modules:

- import matplotlib

python -m pip install -U matplotlib For Windows

Python3 pip install matplotlib For Mac OSX

- import re
- import pytest

python -m pip install pytest For Windows

pip install pytest For Mac OSX

- import click

pip install -U click

- import csv
- import pandas

pip install pandas

- import os.path
- import os
- import sys
- import covid_information
- import datetime
- Not Tested for Linux Machines
- If using a Mac OSX, use:
- import ssl
- `ssl._create_default_https_context = ssl._create_unverified_context`

CSV File

Updated_Report.csv: This csv file is a collection of COVID-19 information since January 2020, currently with approximately 16,000 reports. The information is updated every day to report the date, name of the state, FIPS code, total number of cases, and total number of deaths due to coronavirus. This file is read, used by multiple functions, and filters the data to output certain information pertaining to a particular state or several states. This data can then be used to compare among states and have the most recent and updated information about COVID-19 case numbers and deaths in the United States.

TXT File

new_dara.txt: This text file is populated with data from CDC (centers for disease control and prevention), and the data are accurate as of mid-November. The file has a list of fifty states and the number of deaths in each state. This file is read by reg_data function and it returns the data in the file as tuple. The file is also used by graph function to sort the states and number of deaths from highest to lowest. The graph function used the data from sorted lists to make a bar graph that shows ten states with the highest number of deaths.

How to run the scripts: **Important**

Running pytest:

Requirements: updated csv file from repository (Updated_Report.csv) must be in working directory

pytest CovidTrackerPytest.py

Running our python script:

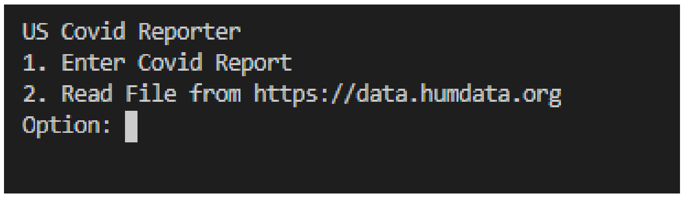
python For Windows or python3 For Mac OSX

ex. python3 CovidTrackerApp.py

Functions:

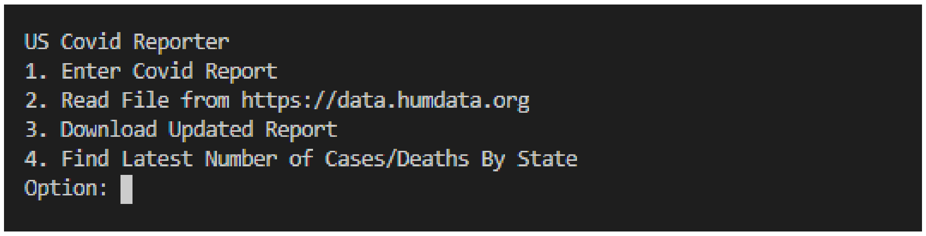
main function: **Important**

When first running the application, you will be prompted with two options. The first option will allow users to enter a COVID report for today's date. The second option will extract data from data.humdata.org's API and store that information in our application.



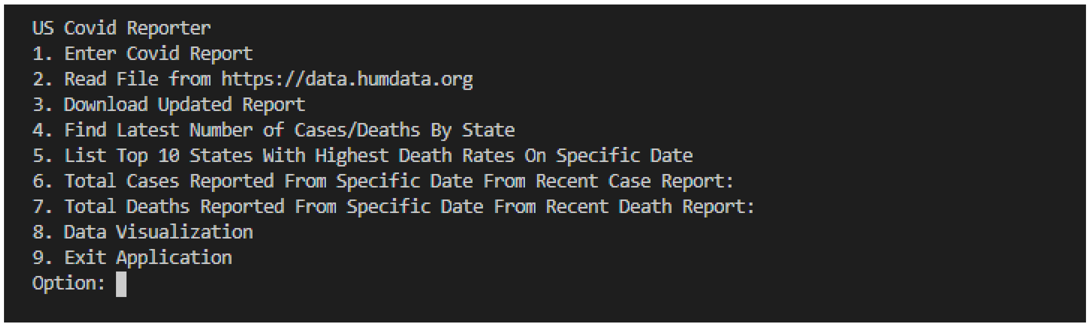
```
US Covid Reporter
1. Enter Covid Report
2. Read File from https://data.humdata.org
Option: 
```

Once a user has inputted at least one COVID-19 report or they have selected to extract data from the API, three new options will become available. Option #3 will allow users to download a csv file of the data extracted from the API, along with any new report that the user has inputted, and it will be sorted in descending order by date, state name. Option #4 will get the latest number of cases/deaths by state. If a user only inputted a new report, that date becomes the latest report date. Therefore, if the user did not import data from API, he/she can only see the latest report for the inputted COVID reports. If user only imported data using option #2 and downloaded the data after (option #3), the user will see the latest information about COVID cases provided by the API. If a user inputted a new COVID-19 report (option #1) AND imported data (option #2) AND downloaded the csv file (option #3), then the latest date is the date of the new inputted report from option #1. Therefore, it will only display data for the new COVID reports inputted.



```
US Covid Reporter
1. Enter Covid Report
2. Read File from https://data.humdata.org
3. Download Updated Report
4. Find Latest Number of Cases/Deaths By State
Option:
```

Option #5, #6, #7, and #8 only display after the user has downloaded the csv file from our program (option #3). Option #5 will display empty data if the csv file downloaded has less than 10 reports from user input or from extracting API data. This option will display the total number of cases and deaths reported from the date that the user inputted and calculate the death rate for that day (total deaths divided by total cases). Option #6 will display the number of cases reported from the date the user inputted to the current date (Total cases for latest date - total cases for inputted date). Option #7 will display the number of deaths reported from the date the user inputted to the current date (Total deaths for latest date - total deaths for inputted date). Option #8 will display a bar graph which contains ten states with the highest number of deaths. This option will also close the program and delete the csv file that was downloaded. Option #9 will exit the application and delete the csv file. For running the program again, you must exit the application from the program, which will delete the CSV but if you kill the terminal you must manually delete the CSV file to run the application properly.



```
US Covid Reporter
1. Enter Covid Report
2. Read File from https://data.humdata.org
3. Download Updated Report
4. Find Latest Number of Cases/Deaths By State
5. List Top 10 States With Highest Death Rates On Specific Date
6. Total Cases Reported From Specific Date From Recent Case Report:
7. Total Deaths Reported From Specific Date From Recent Death Report:
8. Data Visualization
9. Exit Application
Option: 
```

add_reports function:

This function utilizes #1 in the options menu of the terminal. Users can manually input a COVID report for the current date. After the user chooses this function, they will be prompted with a screen that displays the current date and tells users to input the state they wish to report COVID cases and deaths. If a user does not input a valid state, they will be prompted to re-enter the state because they input invalid input. After successfully inputting a valid state, the user will then proceed to enter the number of cases and deaths they wish to report. This screen also shows the fips code for the state they inputted. If the user does not input an integer, they will be prompted to re-enter cases or deaths. After successfully inputting valid values, the user may wish to enter another COVID report for another state, in which they could repeat the process or go back to the main menu. The function is useful because it allows states to report new cases and deaths and keep track of the latest information. This function also does not allow for duplicate report objects to be added to our set. A duplicate COVID report object is when a state enters information twice in a unique day (Python Hash, 1)

read_file function:

This function utilizes #2 in the options menu of the terminal. The purpose of this function is to extract data from data.humdata.org and import it in our program. This would allow us to extract, manipulate, and modify data so it's accessible and readable to users. All the data gets stored in a set of COVID report objects that are not duplicate (Python Hash, 1). There's also a progress bar that users see when the program is extracting data, so they know how long it's going to take to import data to our application (Click Document, 1).

write_file function:

This function utilizes #3 in the options menu of the terminal. The purpose of this function is to download the extracted data we imported from option #2 as well as if a user inputted a new COVID report. This will write to a CSV file named “Updated_Report.csv” all the latest information from the API and the new COVID reports that the user inputted. The CSV will be sorted in descending order by date and state name. This function is useful in that users may wish to download a csv containing the latest information to compare total cases/deaths from various dates until today’s date.

get_latest_report function:

This function is a helper function that helps the program get the latest report date of a COVID report. The side effect of this function is that if a user were to input a new COVID report, they must input data for all states or else only the data for that new report shows for some of our functions. If the user does not input a new report, and only imports data, then it gets the latest date from when the API inputted data. In most cases, the API reports all data for all states, therefore, giving us accurate latest information.

latest_highest_deaths function:

This function utilizes #5 in the option menu of the terminal. Once there are <u>at least 10</u> COVID reports inputted into option #1, this function can be used. Or the csv file can be downloaded and read (option #2) so that the function can extract data from it. After the file is downloaded, select option #3 to complete the download. The file has around 16,000 data points, and the function is going to iterate through the file, searching for the data that has been reported on the date identified by the user in (YYYY-MM-DD) format. After extracting the columns reporting the deaths, number of cases, the date, and the state for all the rows collected, a new column will be added to the data frame that reports the death rate. This number is calculated by dividing the number of deaths for each reported state by the number of cases reported for the state and multiplied by 100. The data frame is then sorted in descending order based on the death rate. The function will print out the top 10 states with the highest death rates based off the most recent reporting and the date inputted by the user. The output will list the name of the 10 states along with the date, the total number of deaths, total number of cases, and the death rate for that given day. This function returns a warning, but the one of the TAs informed us that it is ok because the function does what we want it to do without an error.

find_state function:

This function utilizes #4 in the option menu of the terminal. The most recent csv file will need to be downloaded for it to display the most recent data to the user (option #2), or at least one input in option #1. Once option #4 is selected, the user will then input the name of a state, whichever one they choose, to extract the most recent number of cases and deaths reported on that day, along with the FIPS code for the state. If the user does not make a proper input with the name of a state, there will be an error message, asking for the user to try again and enter a new input. NOTE: there is a small side effect to this function: if the user is already in the application and is to enter a new report, that would be the most recent data to display on that state. After searching for a state that has been inputted by the user, if the user were to search for another state, that state would have to also have been inputted into option #1 first. The function will print out the name of the state and the most recent total cases and total deaths associated with that day. If there is no data reported in the file for the day the user is running the code, it is programmed to get data from the previous day, and if there are still no reports from that date, the day before that will be used to run the function. The data is supposedly updated every day, so we don't expect to run into a problem where there are no reports from more than two days before today's date. The user can use this function to find and compare the most recent COVID data among whichever states he or she chooses. This function is extremely useful because as the data file is being updated every other day, this function will extract and print out the most recent data.

recent_most_case_area function:

The purpose of this function is to calculate and display the number of cases reported from the date(user input) to the most recent date. This function utilizes #6 in the option menu of the terminal. If the user types 6 after downloading the csv file from the website(option #2 and #3), it will display the format of date user input(YYYY-MM-DD). Once the user types the date, it will display the change of reported cases of each state in ascending order. By using this function, the user may see how many new cases were reported during the selected period. This function returns False which breaks the while loop in the main function.

recent_most_death_area function:

This function is like the recent_most_case_area function but it calculates and displays the number of deaths reported from the date(user input) to the most recent date. This function utilizes #7 in the option menu of the terminal. If the user types 7 after downloading the csv file from the website(option #2 and #3), it will display the format of date user input (YYYY-MM-DD). Once the user types the date, it will display the change of reported deaths of each state in ascending order. By using this function, the user may see how many new deaths were reported during the certain period. This function returns the list of change values.

reg_data function:

The purpose of the reg_data function is to return lists of states and the number of deaths in each state as a tuple, which later can be used in a graph function for data visualization. This function reads a text file called new data. A file must be in the same repository to be read. The file has the names of the fifty states and the number of deaths in each state. I created the text file by retrieving the data from centers for disease control and prevention (CDC, 2020) and it is accurate as of mid-November. After reading the file, I am using regular expressions to get the name of the states and number of deaths in each state, then I added those data to two separate lists called states and deaths. At the end of the function, I return two list states and deaths as tuples.

graph function:

After running the script choose option 2 - read file from, afterward choose option 3 that says download updated report, then form the list of options choose option 8 to view the bar graph, maximize the graph for better experience. The purpose of graph function is to display a bar graph that has ten states with the highest number of deaths. This function calls reg_data function and stores the tuple that reg_data function returns in a variable. Then the data in the tuple is converted into a dictionary using zip and dictionary (Java2Novice, 2018). Afterwards the list is sorted by states with highest number of deaths to states with lowest number of deaths. In a new variable called x_values I store all states from sorted list by using the list comprehension and using states from sorted list I get the number of deaths from the dictionary and store it into a variable called y_values. I used matplotlib to make a bar graph that shows ten states with the highest number deaths. I used x_values and y_values list as my x and y variable for the bar graph. I used matplotlib tutorials as guidance (matplotlib, 2020) to make sure I am graphing the values the right way, and to adjust the size of graph windows I used a line of code from python pool (pythonpool, 2020). The outcome of this function is a bar graph, and it can be interpreted as ten states shown in the bar graph have the highest number of deaths as of mid-November.

Pytest:

This test script test various function from COVID tracker app script

test_reg_data function:

This function tests whether the length of tuple returns by reg_data function is 50. If the length is 50 it indicates that lists in tuple have fifty states and number of deaths for each state. The test also makes sure that length of lists is not equal to other numbers, for example length of list is not equal to 100. Since the lists are in alphabetical order as well as in order from 0 to 49, the test makes sure that the name of the state and number of deaths at certain position is accurate, for example name of state at 25th position in the first list has to be Montana and Number of deaths at 25th position in second list has to be 707 because it is the number of deaths in Montana.

test_happy_path_cases_death_rate and test_edge_cases_death_rate:

These functions will test the five different dates for the latest_highest_deaths function for the correct death rate output. This function is an input function where the user inputs a date, and the output shows the number of deaths, cases, and the death rate for that given day. This test function is hardcoded to three specific dates and will check to make sure the death rate output matches that of the latest_highest_deaths function.

test_recent_most_case_area:

This function will test the list of change values that the recent_most_case_area function is returning. It will test three different dates which are the user input and compare if the recent_most_case_area function is returning the correct value.

test_edge_most_recent_case_area:

This function will test two edge cases. One of the cases will input the most recent date. Because the recent_most_case_area function is subtracting the minimum value from the maximum value in the given period, inputting the most recent date will drive the most_recent_case_area function to subtract the same value. The other function will input the date in the future. Since there is no data for the future date, it will return the empty list.

test_happy_cases_add_report:

This function will test to see if the user has put in a valid input for all fields of the COVID report.

test_edge_cases_add_report:

This function will test and see if the user has inputted any invalid inputs into the report, and if they have not, prompt the user with a message to change the input to be valid.

COVID Tracker App: Annotated Bibliography

"CDC COVID Data Tracker." *Centers for Disease Control and Prevention*, Centers for Disease Control and Prevention, covid.cdc.gov/covid-data-tracker/.

For the reg_data function I needed data from reliable sources to do data visualization. The CDC (centers for disease control and prevention) has covid data for all fifty states, so I used those data to create a text file for the reg_data function. I populated the text file with the names of the states and the number of deaths in each state. Then added those data into separate lists called states and deaths using regular expression. Also, these lists are returned as tuples by reg_data so the reg_data function can be called in a graph function and use the data to make a bar graph.

"Datetime - Basic Date and Time Types¶." *Datetime - Basic Date and Time Types - Python 3.9.1 Documentation*, docs.python.org/3/library/datetime.html.

This source provides the information of datetime modules. I utilized the Datetime module for the recent_most_case_area and the recent_most_death_area functions. I was having trouble in comparing the format of the Datetime module and the date format of the csv file. I could figure out how to manually change the format of the date by using this source.

Java2Novice, https://youtu.be/KKcF6r9RM9E

When I called reg_data function in a graph function two lists were returned as tuples. I wanted to access data in tuples and convert them into the dictionary so I can sort the dictionary by its value from highest number of deaths to lowest number of deaths. I watched a python tutorial video from Java2Novice, where the presenter converts the two lists into dictionaries using dict and zip. First, I access lists in tuple then using help from Java2Novice I was able to convert two lists into the dictionary.

"Matplotlib Figsize: Change the Size of Graph Using Figsize." *Python Pool*, 24 Aug. 2020, www.pythonpool.com/matplotlib-figsize/.

When my teammates run my functions, the graph appears smaller, so I was suggested to change the size of graph windows. I used a line of code from python pool to make these changes. I used "figure(figsize=(15,4))" code and changed the number to make graph windows bigger.

"Pandas.DataFrame¶." *Pandas.DataFrame - Pandas 1.1.5 Documentation*, pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html.

This module was used to learn how to collect data from a csv file and read it into a pandas dataframe. The site provided valuable information regarding the max, min, and

group by functions which allowed us to extract information from the csv file and output the exact data we wanted. We used groupby in several of our functions to group the file by state so that our output was clean and organized.

“Parametrize Pytest for List and Pandas Dataframe Fixture.” *Stack Overflow*, 1 Feb. 1969, stackoverflow.com/questions/59653813/parametrize-pytest-for-list-and-pandas-dataframe-fixture.

This source provided some guidance to create the pytest for the latest_highest_deaths_rates function. The same basic format was used as shown in the example, but the column names were changed to match that of our function. The death rate percentage that was being tested was also changed from the original example in this site. Since pandas are still a new concept, and we have not written a pytest for this topic before, this site was extremely useful to gain a better insight into how to go about writing a test for this function that utilizes pandas.

“Python Hash().” *Programiz*, www.programiz.com/python-programming/methods/built-in/hash.

I came across the problem of having duplicate new COVID reports. After researching online, I found that to make an object unique they must implement a hashcode for that object. For a COVID report to be unique, they must have the same date and state name. So, after understanding the concept of a hashcode, I implemented the __hash__ to make my object unique using this documentation on the concept being applied to an example.

Real Python. “Reading and Writing CSV Files in Python.” *Real Python*, Real Python, 21 Nov. 2020, realpython.com/python-csv/.

This module was used for writing data to a CSV file. Since the professor never gave a lecture on writing to CSV files. I decided to research online about writing data to CSV files and implement it in our application. I used this documentation to understand how I would go about writing to CSV files by using the writerow function on a writer object.

“The New York Times Coronavirus (Covid-19) Cases and Deaths in the United States - Us-States.csv - Humanitarian Data Exchange.” *The New York Times Coronavirus (Covid-19) Cases and Deaths in the United States* , Humanitarian Data Exchange, data.humdata.org/dataset/nyt-covid-19-data/resource/34450bc6-76e5-49a5-879e-26edfa7b3b27.

This site is the basis of information for many of our functions in the program. The site updates every few days, and reports the date, the name of the state, the FIPS code, the number of cases, and number of deaths. The file is a repository of data since January of 2020 and is updated to report the most recent information about coronavirus. Our main function utilizes the csv file from this site for the user to download in the command line. Each one of our functions then extracts different information from the file to filter the data a certain way and print the output. The data on this site is extremely useful for using and reporting the most recent data for whomever runs our program.

"Tutorials¶." *Tutorials - Matplotlib 3.3.3 Documentation*, matplotlib.org/3.3.3/tutorials/index.html.

I am using matplotlib to do data visualization. I do have some matplotlib experience from INST 126 class, but I still wanted to use tutorials from matplotlib to make sure I am graphing everything correctly. I used matplotlib tutorials as a guidance as well as reference while I was plotting x and y variables, naming labels, and naming title of the graph.

"Utilities¶." *Utilities - Click Documentation (5.x)*, click.palletsprojects.com/en/5.x/utils/.

Another problem that I came across when I was trying to implement the read_file function was that it did not show anything in the terminal for two minutes. However, the program was working fine. After researching online about possible implementation on a progress bar, I came across the Click module documentation. Within this module, they had useful code for clearing the terminal and a progress bar that displayed the amount of time it would take to extract data within a nice progress bar animation.