**Python API Homework - What's the Weather Like?**

**Background**

Whether financial, political, or social -- data's true power lies in its ability to answer questions definitively. So let's take what you've learned about Python requests, APIs, and JSON traversals to answer a fundamental question: "What's the weather like as we approach the equator?"

Now, we know what you may be thinking: *"Duh. It gets hotter..."*

But, if pressed, how would you **prove** it?

**Before You Begin**

1. Create a new repository for this project called python-api-challenge. **Do not add this homework to an existing repository**.
2. Clone the new repository to your computer.
3. Inside your local git repository, create a directory for both of the Python Challenges. Use a folder name that corresponds to the challenges, such as: **WeatherPy**.
4. Inside the folder you just created, add new files called WeatherPy.ipynb and VacationPy.ipynb. These will be the main scripts to run for each analysis.
5. Push the above changes to GitHub.

**Adding A .gitignore File**

We don't want the api\_keys.py file containing the API key to be exposed to the public on GitHub, as this would mean anyone could copy and use our API key, possibly causing us to incur charges.

When we type git status in the command line, we can see all the files we have created so far that are untracked.

If we only wanted to add the WeatherPy.ipynb file to GitHub we could type git add WeatherPy.ipynb. However, every time we want to add a new file or update current files to the repository, we have to add each file individually, which is time-consuming and cumbersome. Instead, we can add the files we don't want to track to the .gitignore file.

Before we add our files to GitHub, let's add api\_keys.py to the .gitignore file. Follow these steps:

1. Open your python-api-challenge GitHub folder in VS Code.
2. Open the .gitignore file, and on the first line type the following:

# Adding config.py file.

api\_keys.py

1. While the .gitignore file is open, add the API\_practice.ipynb and random\_numbers.ipynb files and save the file.
2. In the command line, type git status and press Enter. The output should say the .gitignore file has been modified and the WeatherPy.ipynb file is untracked.
3. Use git add, git commit, and git push to commit the modifications to .gitignore and the WeatherPy.ipynb file to GitHub.

On GitHub, the only new file you should see is the WeatherPy.ipynb file.

**Part I - WeatherPy**

In this example, you'll be creating a Python script to visualize the weather of 500+ cities across the world of varying distance from the equator. To accomplish this, you'll be utilizing a [simple Python library](https://pypi.python.org/pypi/citipy), the [OpenWeatherMap API](https://openweathermap.org/api), and a little common sense to create a representative model of weather across world cities.

The first requirement is to create a series of scatter plots to showcase the following relationships:

* Temperature (F) vs. Latitude
* Humidity (%) vs. Latitude
* Cloudiness (%) vs. Latitude
* Wind Speed (mph) vs. Latitude

After each plot, add a sentence or two explaining what the code is analyzing.

The second requirement is to run linear regression on each relationship. This time, separate the plots into Northern Hemisphere (greater than or equal to 0 degrees latitude) and Southern Hemisphere (less than 0 degrees latitude):

* Northern Hemisphere - Temperature (F) vs. Latitude
* Southern Hemisphere - Temperature (F) vs. Latitude
* Northern Hemisphere - Humidity (%) vs. Latitude
* Southern Hemisphere - Humidity (%) vs. Latitude
* Northern Hemisphere - Cloudiness (%) vs. Latitude
* Southern Hemisphere - Cloudiness (%) vs. Latitude
* Northern Hemisphere - Wind Speed (mph) vs. Latitude
* Southern Hemisphere - Wind Speed (mph) vs. Latitude

After each pair of plots, take the time to explain what the linear regression is modeling. For example, describe any relationships you notice and any other analysis you may have.

Your final notebook must:

* Randomly select **at least** 500 unique (non-repeat) cities based on latitude and longitude.
* Perform a weather check on each of the cities using a series of successive API calls.
* Include a print log of each city as it's being processed with the city number and city name.
* Save a CSV of all retrieved data and a PNG image for each scatter plot.

**Part II - VacationPy**

Now let's use your skills in working with weather data to plan future vacations. Use jupyter-gmaps and the Google Places API for this part of the assignment.

* **Note:** Remember that any API usage beyond the $200 credit will be charged to your personal account. You can set quotas and limits to your daily requests to be sure you can't be charged. Check out [Google Maps Platform Billing](https://developers.google.com/maps/billing/gmp-billing#monitor-and-restrict-consumption) and [Manage your cost of use](https://developers.google.com/maps/documentation/javascript/usage-and-billing#set-caps) for more information.
* **Note:** if you having trouble displaying the maps, try running jupyter nbextension enable --py gmaps in your environment and retry.

To complete this part of the assignment,you will need to do the following:

* Create a heat map that displays the humidity for every city from Part I.
* Narrow down the DataFrame to find your ideal weather condition. For example:
  + A max temperature lower than 80 degrees but higher than 70.
  + Wind speed less than 10 mph.
  + Zero cloudiness.
  + Drop any rows that don't contain all three conditions. You want to be sure the weather is ideal.
  + **Note:** Feel free to adjust to your specifications but be sure to limit the number of rows returned by your API requests to a reasonable number.
* Using Google Places API to find the first hotel for each city located within 5000 meters of your coordinates.
* Plot the hotels on top of the humidity heatmap with each pin containing the **Hotel Name**, **City**, and **Country**.

As final considerations:

* You must complete your analysis using a Jupyter notebook.
* You must use the Matplotlib or Pandas plotting libraries.
* For Part I, you must include a written description of three observable trends based on the data.
* For Part II, you must include a screenshot of the heatmap you create and include it in your submission.
* You must use proper labeling of your plots, including aspects like: Plot Titles (with date of analysis) and Axes Labels.
* For max intensity in the heat map, try setting it to the highest humidity found in the data set.

**Hints and Considerations**

* The city data you generate is based on random coordinates as well as different query times. As such, your outputs will not be an exact match to the provided starter notebook.
* If you'd like a refresher on the geographic coordinate system, [this site](http://desktop.arcgis.com/en/arcmap/10.3/guide-books/map-projections/about-geographic-coordinate-systems.htm) has great information.
* Next, spend the requisite time necessary to study the OpenWeatherMap API. Based on your initial study, you should be able to answer basic questions about the API: Where do you request the API key? Which Weather API in particular will you need? What URL endpoints does it expect? What JSON structure does it respond with? Before you write a line of code, you should be aiming to have a crystal clear understanding of your intended outcome.
* A starter code for Citipy has been provided. However, if you're craving an extra challenge, push yourself to learn how it works: [citipy Python library](https://pypi.python.org/pypi/citipy). Before you try to incorporate the library into your analysis, start by creating simple test cases outside your main script to confirm that you are using it correctly. Too often, when introduced to a new library, students get bogged down by the most minor of errors -- spending hours investigating their entire code -- when, in fact, a simple and focused test would have shown their basic utilization of the library was wrong from the start. Don't let this be you!
* Part of our expectation in this challenge is that you will use critical thinking skills to understand how and why we're recommending the tools we are. What is Citipy for? Why would you use it in conjunction with the OpenWeatherMap API? How would you do so?
* In building your script, pay attention to the cities you are using in your query pool. Are you getting coverage of the full gamut of latitudes and longitudes? Or are you simply choosing 500 cities concentrated in one region of the world? Even if you were a geographic genius, simply rattling 500 cities based on your human selection would create a biased dataset. Be thinking of how you should counter this. (Hint: Consider the full range of latitudes).
* Once you have computed the linear regression for one chart, the process will be similar for all others. As a bonus, try to create a function that will create these charts based on different parameters.
* Remember that each coordinate will trigger a separate call to the Google API. If you're creating your own criteria to plan your vacation, try to reduce the results in your DataFrame to 10 or fewer cities.
* Ensure your repository has regular commits and a thorough README.md file.
* Lastly, remember -- this is a challenging activity. Push yourself! If you complete this task, then you can safely say that you've gained a strong mastery of the core foundations of data analytics and it will only go better from here. Good luck!

**Rubric**

[Unit 6 Rubric - Python API Homework - What's the Weather Like?](https://docs.google.com/document/d/1Y17QYjs0KMeEPPGd_1BpMjnqXiTaJVeFwqea5ReMdeU/edit?usp=sharing)

© 2021 Trilogy Education Services, LLC, a 2U, Inc. brand. Confidential and Proprietary. All Rights Reserved.