Module 6 Challenge

Start Assignment

- Due Mar 18 by 11:59pm
- Points 100
- Submitting a text entry box or a website url

Background

Data's true power is its ability to definitively answer questions. So, let's take what you've learned about Python requests, APIs, and JSON traversals to answer a fundamental question: "What is the weather like as we approach the equator?"

Now, we know what you may be thinking: "That's obvious. It gets hotter." But, if pressed for more information, how would you prove that?

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 this assignment. Use a folder name that corresponds to the Challenges, such as **WeatherPy**.

4. Inside the folder you just created, add the files called `api_keys.py`, `WeatherPy.ipynb`, and `VacationPy.ipynb` that you will find in the starter code ZIP file provided. These will be the main scripts to run for each analysis.

5. Before you push your changes to GitHub, add a .gitignore file.

Add a .gitignore File

For this assignment, you will need to add a `.gitignore` file to your repo. Doing so will prevent the `api_keys.py` file that contains your API key from being shared with the public. If you skip this step, anyone using GitHub could copy and use your API key, and you may incur charges as a result.

To get started, type `git status` in the command line to see a list of all the untracked files that you have created so far.

To add only the `WeatherPy.ipynb` file to GitHub, for example, type `git add WeatherPy.ipynb`. Keep in mind that you would have to add each file individually when adding or updating a file. A more efficient solution is to add all of the files that you don't want to track to the `.gitignore` file.

Before adding your files to GitHub, add `api_keys.py` to the `.gitignore` file by following these steps:

1. Open your `python-api-challenge` GitHub folder in VS Code.

2. Open the `.gitignore` file and type the following code on the first line:

```
# Adding config.py file.
api_keys.py
```

3. In the command line, type `git status` and press Enter. The output should indicate that the `.gitignore` file has been modified and the `api_keys.py` file is untracked.

4. Use `git add`, `git commit`, and `git push` to commit the modifications to the `.gitignore`, `WeatherPy.ipynb` and `VacationPy.ipynb` files to GitHub.

On GitHub, the only new python files you should find are `WeatherPy.ipynb` and `VacationPy.ipynb`.

Files

Download the following files to help you get started:

Module 6 Challenge files ▣ **(https://static.bc-edx.com/data/dl-1-2/m6/lms/starter/Starter_Code.zip)**

Instructions

This activity is broken down into two deliverables, WeatherPy and VacationPy.

Part 1: WeatherPy

In this deliverable, you'll create a Python script to visualize the weather of over 500 cities of varying distances from the equator. You'll use the **citipy Python library** ▣ **(https://pypi.python.org/pypi/citipy)**, the **OpenWeatherMap API** ▣ **(https://openweathermap.org/api)**, and your problem-solving skills to create a representative model of weather across cities.

For this part, you'll use the `WeatherPy.ipynb` Jupyter notebook provided in the starter code ZIP file. The starter code will guide you through the process of using your Python coding skills to develop a solution to address the required functionalities.

To get started, the code required to generate random geographic coordinates and the nearest city to each latitude and longitude combination is provided.

Requirement 1: Create Plots to Showcase the Relationship Between Weather Variables and Latitude

To fulfill the first requirement, you'll use the OpenWeatherMap API to retrieve weather data from the cities list generated in the starter code. Next, you'll create a series of scatter plots to showcase the following relationships:

- Latitude vs. Temperature

- Latitude vs. Humidity

- Latitude vs. Cloudiness

- Latitude vs. Wind Speed

Requirement 2: Compute Linear Regression for Each Relationship

To fulfill the second requirement, compute the linear regression for each relationship. Separate the plots into Northern Hemisphere (greater than or equal to 0 degrees latitude) and Southern Hemisphere (less than 0 degrees latitude). You may find it helpful to define a function in order to create the linear regression plots.

Next, create a series of scatter plots. Be sure to include the linear regression line, the model's formula, and the r values as you can see in the following image
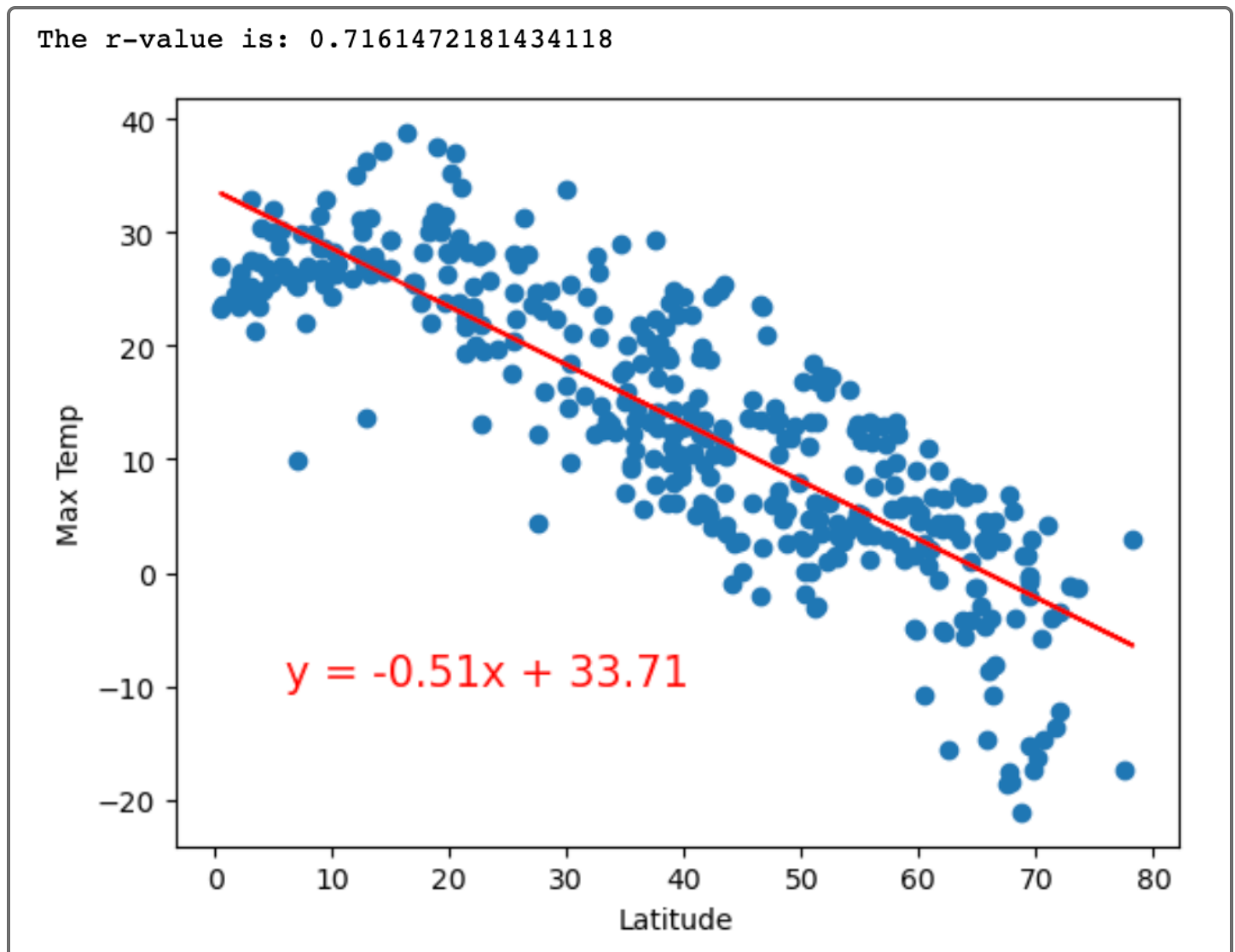
You should create the following plots:

- Northern Hemisphere: Temperature vs. Latitude

- Southern Hemisphere: Temperature 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 vs. Latitude

- Southern Hemisphere: Wind Speed vs. Latitude

After each pair of plots, explain what the linear regression is modeling. Describe any relationships that you notice and any other findings you may uncover.

Part 2: VacationPy

In this deliverable, you'll use your weather data skills to plan future vacations. Also, you'll use Jupyter notebooks, the geoViews Python library, and the Geoapify API.

The code needed to import the required libraries and load the CSV file with the weather and coordinates data for each city created in Part 1 is provided to help you get started.

Your main tasks will be to use the Geoapify API and the geoViews Python library and employ your Python skills to create map visualizations.

To succeed on this deliverable of the assignment, open the `VacationPy.ipynb` starter code and complete the following steps:

1. Create a map that displays a point for every city in the `city_data_df` DataFrame as shown in the following image. The size of the point should be the humidity in each city.
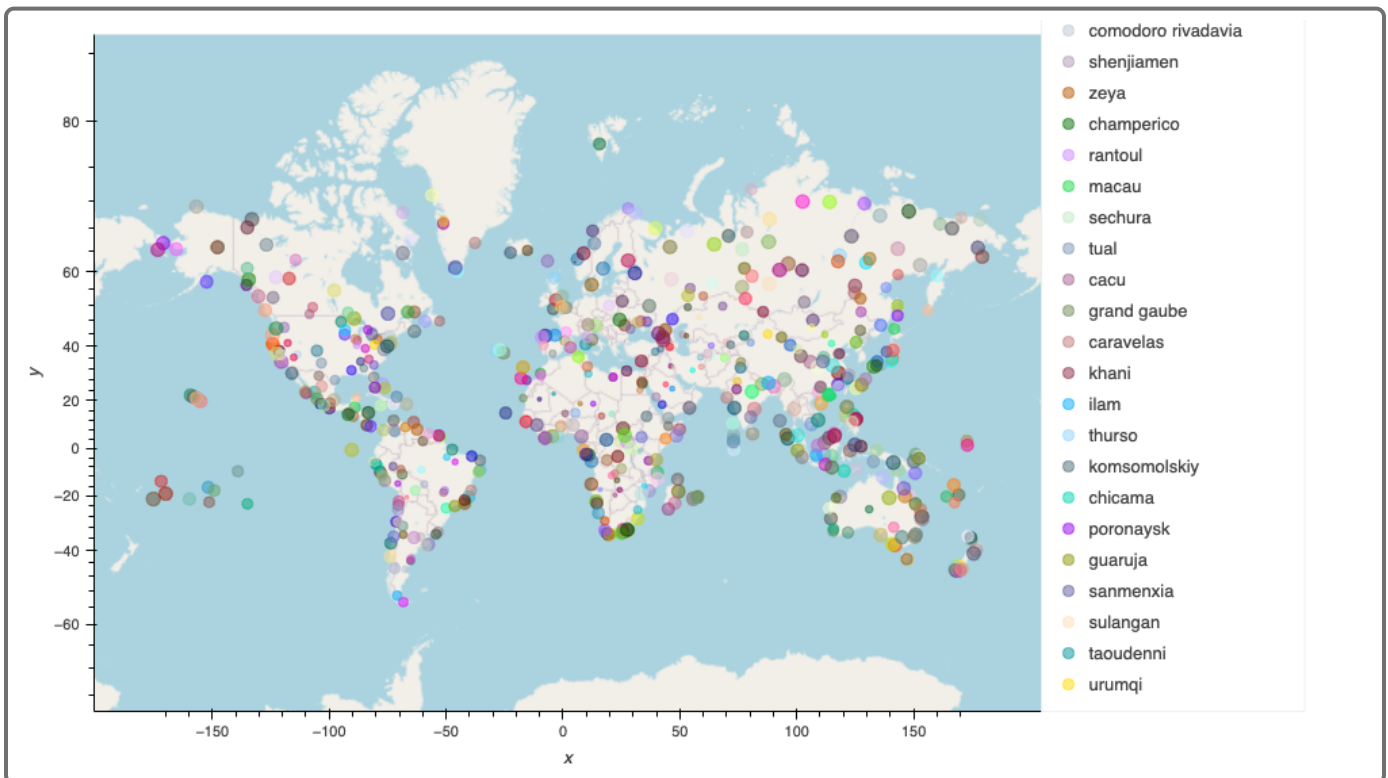
2. Narrow down the `city_data_df` DataFrame to find your ideal weather condition. For example:

   - A max temperature lower than 27 degrees but higher than 21

   - Wind speed less than 4.5 m/s

   - Zero cloudiness

   NOTE

   Feel free to adjust your specifications but make sure to set a reasonable limit to the number of rows returned by your API requests.

3. Create a new DataFrame called `hotel_df` to store the city, country, coordinates, and humidity.

4. For each city, use the Geoapify API to find the first hotel located within 10,000 meters of your coordinates.

5. Add the hotel name and the country as additional information in the hover message for each city on the map as in the following image:
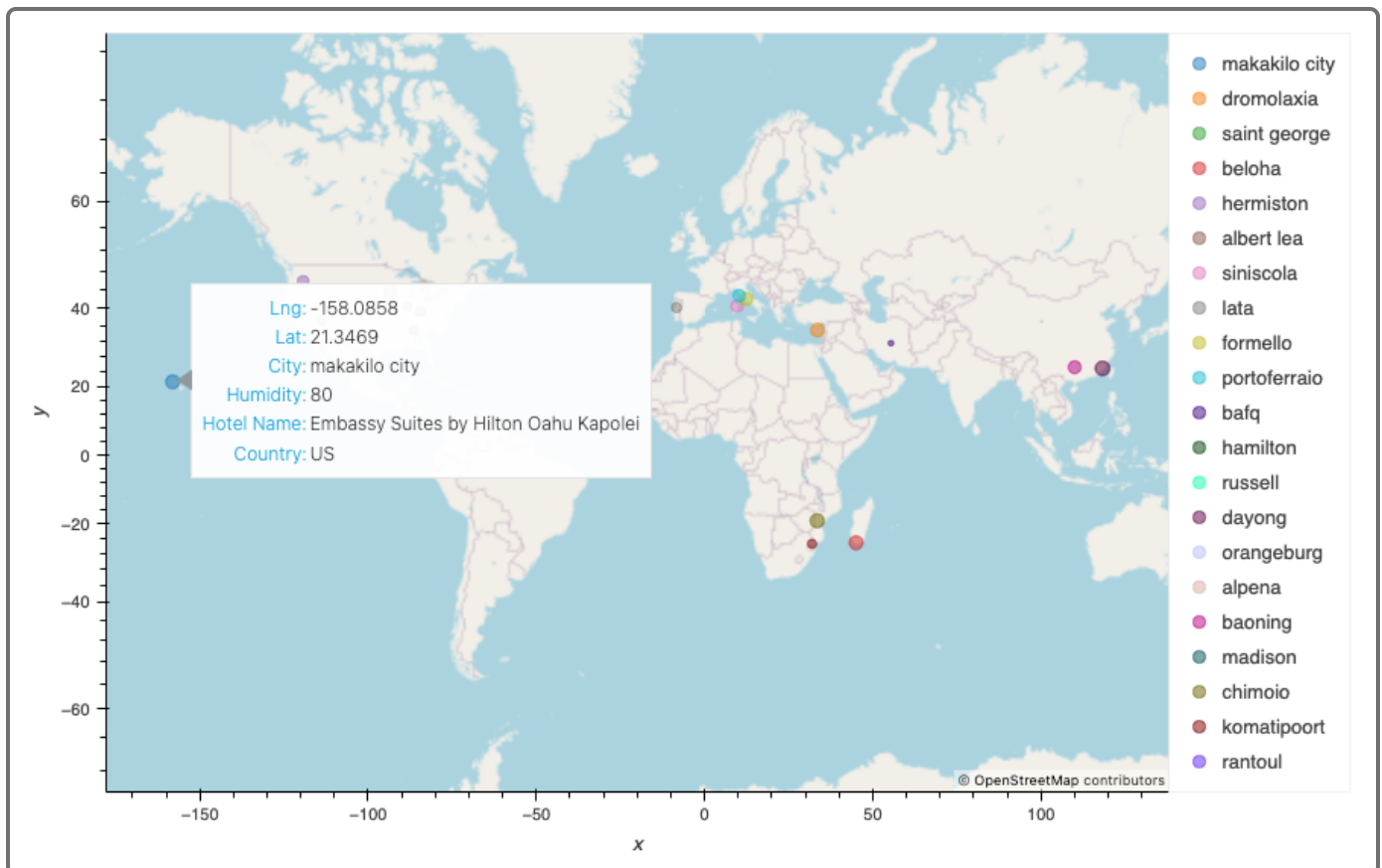
Hints and Considerations

- The city data that you generate is based on random coordinates and different query times, so 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.

- Take some time 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 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 by using the **citipy Python library** ⤷ **(https://pypi.python.org/pypi/citipy)** . Before you try to incorporate the library in your analysis, start with simple test cases outside of your main script to confirm that you are using it correctly. Often, when introduced to a new library, learners spend hours trying to figure out errors in their code when a simple test case can save you a lot of time and frustration.

- You will need to apply your critical thinking skills to understand how and why we're recommending these tools. What is citipy used for? Why would you use it in conjunction with the OpenWeatherMap API? How would you do so?

- While building your script, pay attention to the cities you are using in your query pool. Are you covering the full range of latitudes and longitudes? Or are you choosing 500 cities from one region of the world? Even if you were a geography genius, simply listing 500 cities based on your personal selection would create a biased dataset. Try to think of ways that you can counter these selection issues.

  - **Hint:** Consider the full range of latitudes.

- Once you have computed the linear regression for one relationship, you will follow a similar process for all other charts. Optionally, try to create a function that will create these charts based on different parameters. (Note: there will be no extra points for completing this.)

- 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 that your repository has regular commits and a thorough README.md file.

- Lastly, remember that this is a challenging activity. Push yourself! If you complete this task, you can safely say that you've gained a strong understanding of the core foundations of data analytics, and it will only get better from here. Good luck!

Requirements

The requirements for "Part 1: WeatherPy" are the following

Create Plots to Showcase the Relationship Between Weather Variables and Latitude (30 points)

- Use the OpenWeatherMap API to retrieve weather data from the cities list generated in the started code (10 points)

- Create a scatter plot to showcase the relationship between Latitude vs. Temperature (5 points)

- Create a scatter plot to showcase the relationship between Latitude vs. Humidity (5 points)

- Create a scatter plot to showcase the relationship between Latitude vs. Cloudiness (5 points)

- Create a scatter plot to showcase the relationship between Latitude vs. Wind Speed (5 points)

Compute Linear Regression for Each Relationship (40 points)

- Linear regression scatter plot for Northern Hemisphere: Temperature (C) vs. Latitude (5 points)

- Linear regression scatter plot for Southern Hemisphere: Temperature (C) vs. Latitude (5 points)

- Linear regression scatter plot for Northern Hemisphere: Humidity (%) vs. Latitude (5 points)

- Linear regression scatter plot for Southern Hemisphere: Humidity (%) vs. Latitude (5 points)

- Linear regression scatter plot for Northern Hemisphere: Cloudiness (%) vs. Latitude (5 points)

- Linear regression scatter plot for Southern Hemisphere: Cloudiness (%) vs. Latitude (5 points)

- Linear regression scatter plot for Northern Hemisphere: Wind Speed (m/s) vs. Latitude (5 points)

- Linear regression scatter plot for Southern Hemisphere: Wind Speed (m/s) vs. Latitude (5 points)

The requirements for "Part 2: VacationPy" are the following (30 points)

- Create a map that displays a point for every city in the `city_data_df` DataFrame (5 points)

- Narrow down the `city_data_df` DataFrame to find your ideal weather condition (5 points)

- For each city in the `hotel_df` DataFrame, use the Geoapify API to find the first hotel located within 10,000 metres of your coordinates (10 points)

- Add the hotel name and the country as additional information in the hover message for each city in the map. (10 points)

Grading

This assignment will be evaluated against the requirements and assigned a grade according to the following table:

Grade	Points
A (+/-)	90+
B (+/-)	80–89
C (+/-)	70–79
D (+/-)	60–69
F (+/-)	< 60

Submission

To submit your Challenge assignment, click Submit, and then provide the URL of your GitHub repository for grading.

NOTE

You are allowed to miss up to two Challenge assignments and still earn your certificate. If you complete all Challenge assignments, your lowest two grades will be dropped. If you wish to skip this assignment, click Next, and move on to the next module.

Comments are disabled for graded submissions in Bootcamp Spot. If you have questions about your feedback, please notify your instructional staff or your Student Success Advisor. If you would like to resubmit your work for an additional review, you can use the Resubmit Assignment button to upload new links. You may resubmit up to three times for a total of four submissions.

IMPORTANT

It is your responsibility to include a note in the README section of your repo specifying code source and its location within your repo. This applies if you have worked with a peer on an assignment, used code in which you did not author or create sourced from a forum such as Stack Overflow, or you received code outside curriculum content from support staff such as an Instructor, TA, Tutor, or Learning Assistant. This will provide visibility to grading staff of your circumstance in order to avoid flagging your work as plagiarized.

If you are struggling with a challenge assignment or any aspect of the academic curriculum, please remember that there are student support services available for you:

1. Ask the class Slack channel/peer support.

2. AskBCS Learning Assistants exists in your class Slack application.

3. Office hours facilitated by your instructional staff before and after each class session.

4. **Tutoring Guidelines** ⤴ **(https://docs.google.com/document/d/ 1hTldEfWhX21B_Vz9ZentkPeziu4pPfnwiZbwQB27E90/edit?usp=sharing)** - schedule a tutor session in the Tutor Sessions section of Bootcampspot - Canvas

5. If the above resources are not applicable and you have a need, please reach out to a member of your instructional team, your Student Success Advisor, or submit a support ticket in the Student Support section of your BCS application.