Module 6 Challenge

New Attempt

Due Monday by 12:59am **Points** 100 **Submitting** a text entry box or a website url

Background

Jack loves the PlanMyTrip app. Beta testers love it too. And, as with any new product, they've recommended a few changes to take the app to the next level. Specifically, they recommend adding the weather description to the weather data you've already retrieved in this module. Then, you'll have the beta testers use input statements to filter the data for their weather preferences, which will be used to identify potential travel destinations and nearby hotels. From the list of potential travel destinations, the beta tester will choose four cities to create a travel itinerary. Finally, using the Google Maps Directions API, you will create a travel route between the four cities as well as a marker layer map.

What You're Creating

This new assignment consists of three technical analyses. You will submit the following deliverables:

- Deliverable 1: Retrieve Weather Data
- Deliverable 2: Create a Customer Travel Destinations Map
- Deliverable 3: Create a Travel Itinerary Map

Files

Use the following links to download the Challenge starter codes.

Download Deliverable 2 starter code (https://2u-data-curriculum-team.s3.amazonaws.com/dataviz-online/module_6/Vacation_Search_starter_code.ipynb)

Download Deliverable 3 starter code (https://2u-data-curriculum-team.s3.amazonaws.com/dataviz-online/module_6/Vacation_Itinerary_starter_code.ipynb)

Deliverable 1: Retrieve Weather Data (25 points)

Deliverable 1 Instructions

Generate a set of 2,000 random latitudes and longitudes, retrieve the nearest city, and perform an API call with the OpenWeatherMap. In addition to the city weather data you gathered in this module, use your API skills to retrieve the current weather description for each city. Then, create a new DataFrame containing the updated weather data.

REWIND

For this deliverable, you've already done the following in this module:

- **Lesson 6.2.3:** Make an API call
- **Lesson 6.2.3:** Create a `config.py` file

- **Lesson 6.2.6:** Retrieve city weather data
- **Lesson 6.2.7:** Add the weather data to a DataFrame
- **Lesson 6.2.7:** Export the DataFrame to a CSV file

1. Create a folder called Weather_Database.
2. Create a new Jupyter Notebook file to retrieve the weather data, and name it `Weather_Database.ipynb`.
3. Create a new set of 2,000 random latitudes and longitudes.
4. Get the nearest city using the `citipy` module.
5. Perform an API call with the OpenWeatherMap.
6. Retrieve the following information from the API call:
    - Latitude and longitude
    - Maximum temperature
    - Percent humidity
    - Percent cloudiness
    - Wind speed
    - Weather description (for example, clouds, fog, light rain, clear sky)
7. Add the data to a new DataFrame.
    - Before exporting your new DataFrame as a CSV file, take a

moment to confirm that it looks similar to the image below:

	City	Country	Lat	Lng	Max Temp	Humidity	Cloudiness	Wind Speed	Current Description
0	Mataura	NZ	-46.19	168.86	39.00	83	2	3.00	clear sky
1	Narsaq	GL	60.92	-46.05	44.60	75	75	2.24	shower rain
2	Aksu	CN	41.12	80.26	82.42	22	6	5.08	clear sky
3	Provideniya	RU	64.38	-173.30	40.84	68	0	0.69	clear sky
4	Bilibino	RU	68.05	166.44	54.82	53	100	4.29	overcast clouds
5	Barrow	US	71.29	-156.79	37.40	89	90	16.11	overcast clouds
6	Ribeira Grande	PT	38.52	-28.70	69.80	73	20	20.80	few clouds
7	Touros	BR	-5.20	-35.46	80.60	74	75	11.41	broken clouds
8	Hasaki	JP	35.73	140.83	66.20	100	75	4.70	light intensity drizzle rain
9	Hoquiam	US	46.98	-123.89	55.00	100	90	0.78	overcast clouds

8. Export the DataFrame as a CSV file, and save it as `WeatherPy_Database.csv` in the Weather_Database folder.

Deliverable 1 Requirements

You will earn a perfect score for Deliverable 1 by completing all requirements below:

- Retrieve all of the following information from the API call: **(15 pt)**
    - Latitude and longitude
    - Maximum temperature
    - Percent humidity
    - Percent cloudiness
    - Wind speed
    - Weather description (for example, clouds, fog, light rain, clear sky)
- Add the weather data to a new DataFrame **(5 pt)**

- Add the weather data to a new DataFrame **(5 pt)**
- Export the DataFrame as `WeatherPy_Database.csv` into the Weather_Database folder **(5 pt)**

Be sure to double-check that you have the following in the Weather_Database folder:

- The `Weather_Database.ipynb` file
- The `WeatherPy_Database.csv` file

Deliverable 2: Create a Customer Travel Destinations Map (35 points)

Deliverable 2 Instructions

Use input statements to retrieve customer weather preferences, then use those preferences to identify potential travel destinations and nearby hotels. Then, show those destinations on a marker layer map with pop-up markers.

REWIND

For this deliverable, you've already done the following in this module:

- **Lesson 6.2.7:** Add the weather data to a DataFrame
- **Lesson 6.2.7:** Add the `config.py` file to the `.gitignore` file

- **Lesson 6.5.3:** Create input statements
- **Lesson 6.5.3:** Filter a DataFrame using the `loc` method
- **Lesson 6.5.4:** Retrieve hotel data
- **Lesson 6.5.4:** Add data to a pop-up marker
- **Lesson 6.5.4:** Create a marker layer map

1. Create a folder called Vacation_Search.

2. Download the `Vacation_Search_starter_code.ipynb` file into your Vacation_Search folder, and rename it `Vacation_Search.ipynb`.

3. In the `Vacation_Search.ipynb` file, make sure the dependencies and API keys are imported correctly.

4. Use the instructions below to add code where indicated by the numbered-step comments in the starter code file.

5. In Step 1, import the `WeatherPy_Database.csv` file from your Weather_Database folder from Deliverable 1 as a DataFrame.

6. In Step 2, write two input statements that prompt the user to enter their minimum and maximum temperature criteria for their vacation.

7. In Step 3, use the `loc` method to filter the `city_data_df` DataFrame for temperature criteria collected in Step 2, then create a new DataFrame.

8. In Steps 4a-b, determine if there are any empty rows, then drop them if necessary and create a new DataFrame.

9. In Steps 5a-b, use the provided code to create a new DataFrame, `hotel_df`, that will hold the hotel names from the hotel search in Steps 6a-6f.

10. In Step 6a, we have supplied the search parameters, which are the same as in this module, that you'll need to use to search for a hotel for each city in Steps 6b-f.

11. In Steps 6b-f, iterate through the `hotel_df` DataFrame, retrieve the latitude and longitude of each city to find the nearest hotel based on the search parameters from Step 6a, then add the hotel name to the `hotel_df` DataFrame. If a hotel isn't found, skip to the next city.

12. In Step 7, drop any rows in the `hotel_df` DataFrame where a hotel name is not found.

    - Before exporting your DataFrame as a CSV, take a moment to confirm that your hotel_df DataFrame looks similar to the image below.

	City	Country	Max Temp	Current Description	Lat	Lng	Hotel Name
2	Aksu	CN	82.42	clear sky	41.12	80.26	Pudong Holiday Hotel
7	Touros	BR	80.60	broken clouds	-5.20	-35.46	INN NEW HORIZON
10	Morehead	US	75.20	clear sky	37.27	-87.18	CCI Express Inn
11	Port Elizabeth	ZA	84.20	clear sky	-33.92	25.57	39 On Nile Guest House
16	Northam	GB	82.40	few clouds	51.03	-4.22	Durrant House Hotel
18	Hyderabad	IN	87.01	haze	17.38	78.47	Taj Krishna, Hyderabad
20	Port Alfred	ZA	82.00	clear sky	-33.59	26.89	The Halyards Hotel

22	Atuona	PF	79.72	clear sky	-9.80	-139.03	Villa Enata
23	Kapaa	US	73.40	heavy intensity rain	22.08	-159.32	Sheraton Kauai Resort at Coconut Beach
25	Nikolskoye	RU	88.00	clear sky	59.70	30.79	Tourist House - Sablino

13. In Steps 8a-b, create an output file to store the `hotel_df` DataFrame as `WeatherPy_vacation.csv` in the Vacation_Search folder.

14. In Step 9, add the city name, the country code, the weather description, and the maximum temperature for the city to the `info_box_template` format template we have provided.

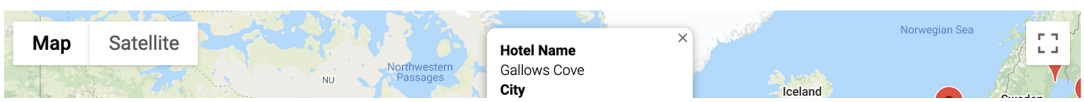
15. In Step 10a, use the provided list comprehension code to retrieve the city data from each row, which will then be added to the formatting template and saved in the `hotel_info` list.

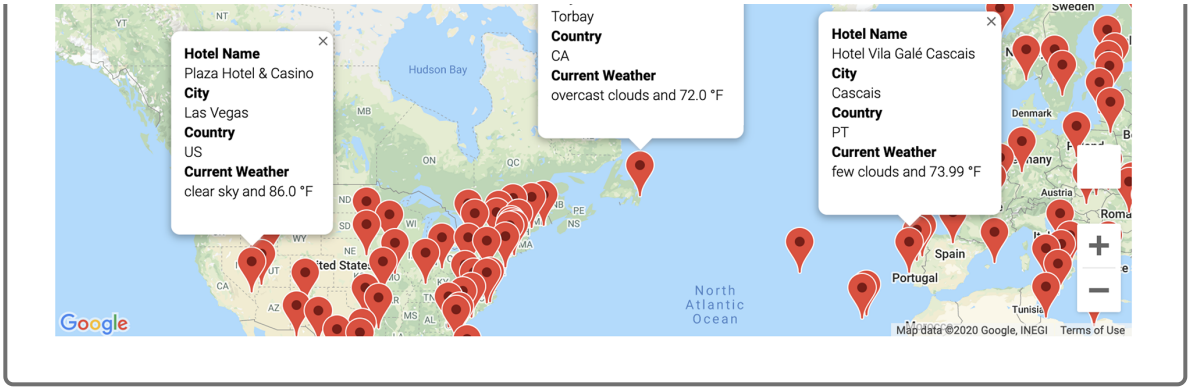
16. In Step 10b, use the provided code snippet to retrieve the latitude and longitude from each row and store them in a new DataFrame.

17. In Steps 11a-b, refactor your previous marker layer map code to create a marker layer map that will have pop-up markers for each city on the map.

18. Take a screenshot of your map and save it to the Vacation_Search folder as `WeatherPy_vacation_map.png`.

    - The marker layer map with a pop-up marker for each city should look similar to the following image:





Deliverable 2 Requirements

You will earn a perfect score for Deliverable 2 by completing all requirements below:

- Input statements are written to prompt the customer for their minimum and maximum temperature preferences. **(5 pt)**
- A new DataFrame is created based on the minimum and maximum temperature, and empty rows are dropped. **(5 pt)**
- The hotel name is retrieved and added to the DataFrame, and the rows that don't have a hotel name are dropped. **(10 pt)**
- The DataFrame is exported as a CSV file into the Vacation_Search folder and is saved as `WeatherPy_vacation.csv`. **(5 pt)**
- A marker layer map with pop-up markers for the cities in the vacation DataFrame is created, and it is uploaded as a PNG. Each marker has the following information: **(5 pt)**
  - Hotel name
  - City
  - Country
  - Current weather description with the maximum temperature
- The marker layer map is saved and uploaded to the Vacation_Search folder as [illegible] **(5 pt)**

folder as `WeatherPy_vacation_map.png`. **(5 pt)**

Be sure to double-check that you have the following in the Vacation_Search folder:

- The `Vacation_Search.ipynb` file
- The `WeatherPy_vacation.csv` file
- The `WeatherPy_vacation_map.png` image

Deliverable 3: Create a Travel Itinerary Map (40 points)

Deliverable 3 Instructions

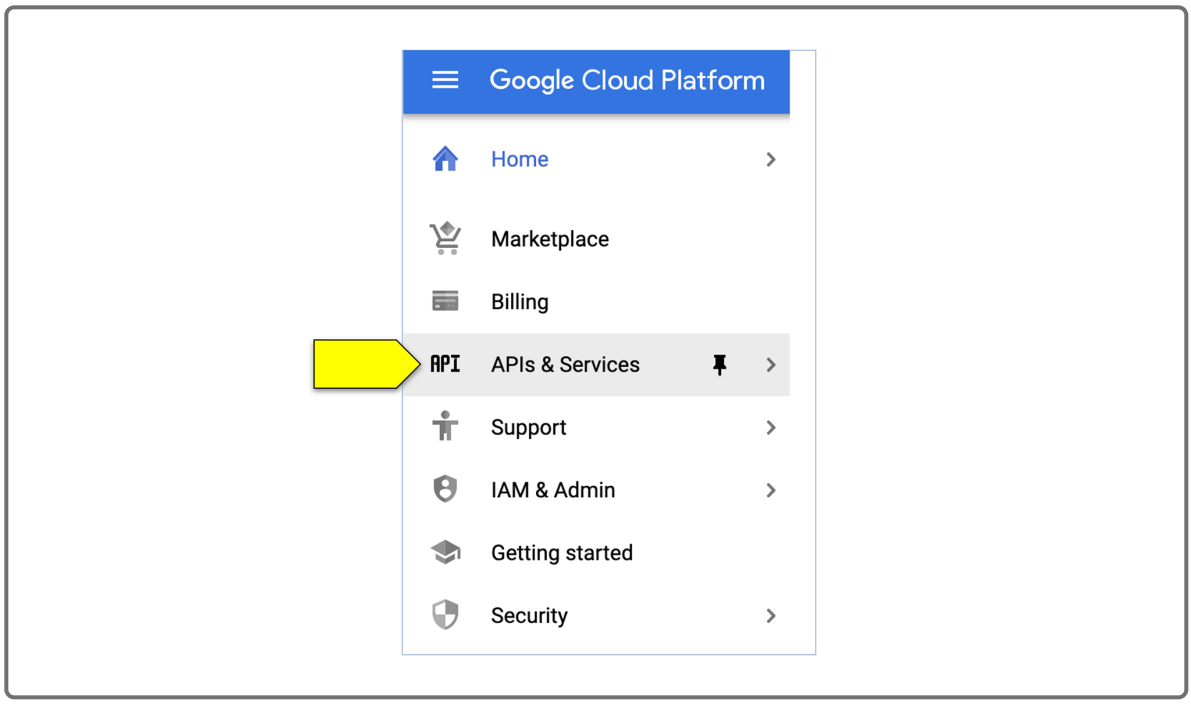
Use the Google Directions API to create a travel itinerary that shows the route between four cities chosen from the customer's possible travel destinations. Then, create a marker layer map with a pop-up marker for each city on the itinerary.

REWIND

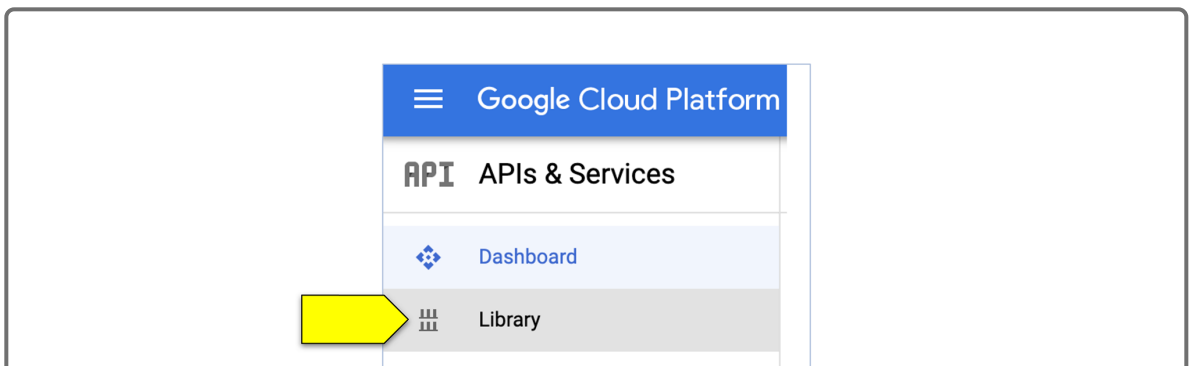
For this deliverable, you've already done the following in this module:

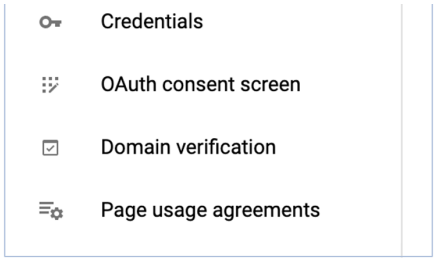
- **Lesson 6.5.4:** Add data to a pop-up marker
- **Lesson 6.5.4:** Create a marker layer map

1. Enable the "Directions API" in your Google account for your API key.
    - On the Google Cloud Platform, select "APIs & Services" from the left-hand side

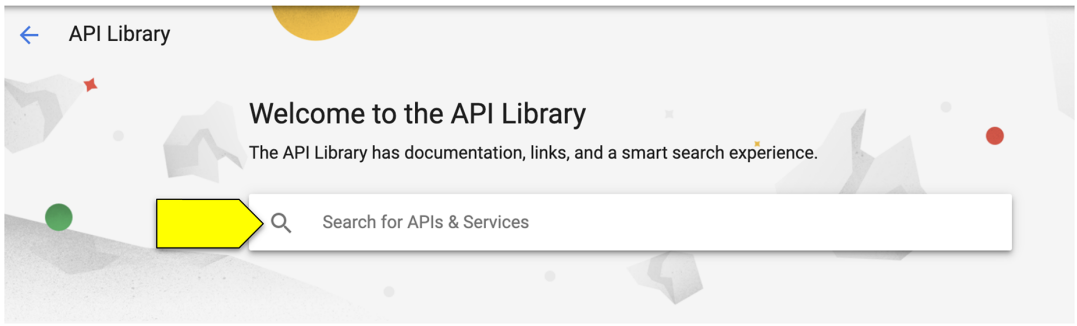


    - Then, select "Library".

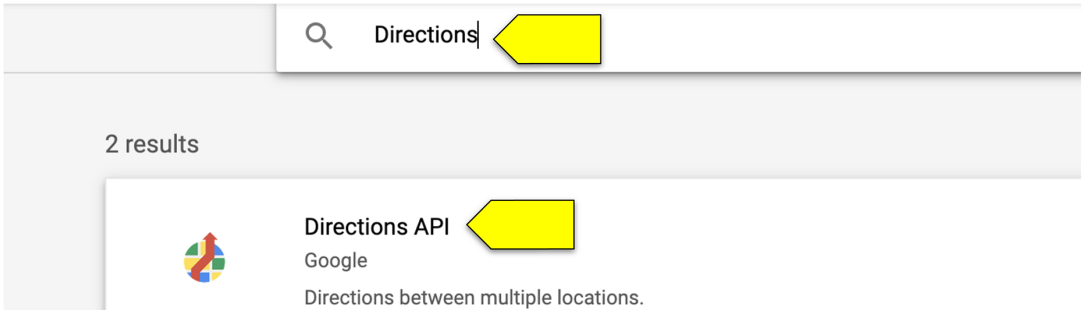


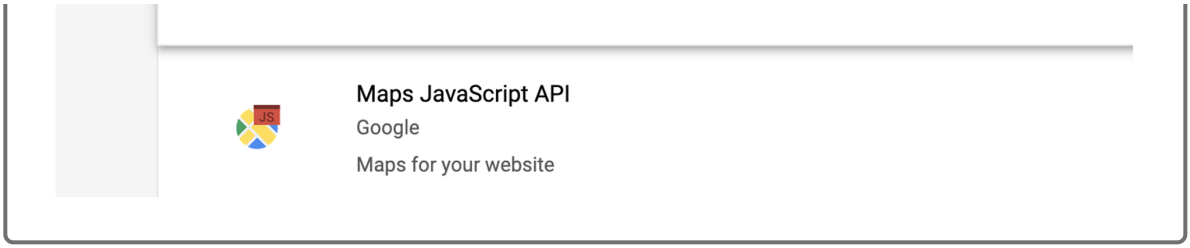


- In the Search field, type "Directions".

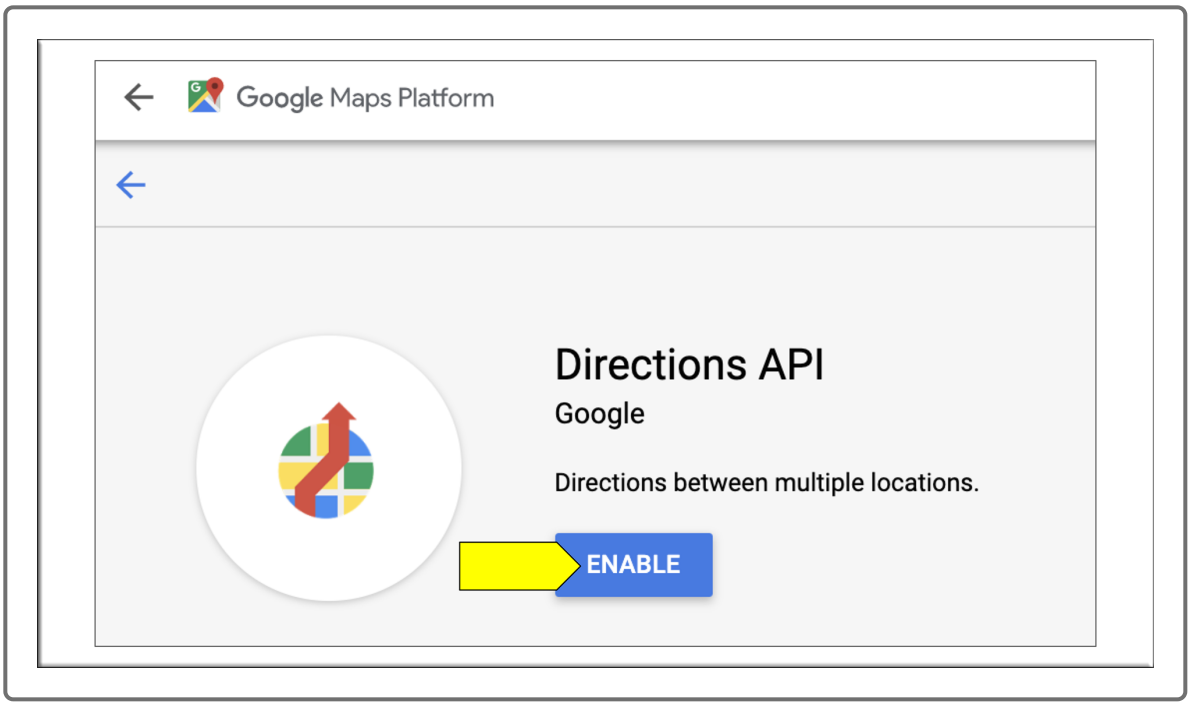


- Select "Directions API".





- Click "Enable" to activate the Directions API.



2. Create a folder called Vacation_Itinerary.

3. Download the `Vacation_Itinerary_starter_code.ipynb` file into your Vacation_Itinerary folder and rename it `Vacation_Itinerary.ipynb`.

4. In the `Vacation_Itinerary.ipynb` file, make sure the dependencies and API keys are imported.

5. Use the instructions below to add code where indicated by the numbered-step comments in the starter code file.

6. In Step 1, import the `WeatherPy_vacation.csv` file from your Vacation_Search folder from Deliverable 2 as a DataFrame.

7. In Steps 2-4, copy or refactor the code from Steps 11a-b of Deliverable 2 to create a marker layer map of the vacation search results.

8. From the vacation search map, *choose four cities* that a customer might want to visit. They should be close together and in the same country.

    - You may have to refine your search with different weather criteria to get cities that are close together.

9. In Step 5, use the variables we have provided and the `loc` method to create separate DataFrames for each city on the travel route.

Hint: You will start and end the route in the same city, so the `vacation_start` and `vacation_end` DataFrames will be the same city.

10. In Step 6, use the `to_numpy()` function and list indexing to write code to retrieve the latitude-longitude pairs as tuples from each city DataFrame.

If you'd like a hint on using the `to_numpy()` function with list indexing, that's totally okay. If not, that's great too. You can always revisit this later if you change your mind.

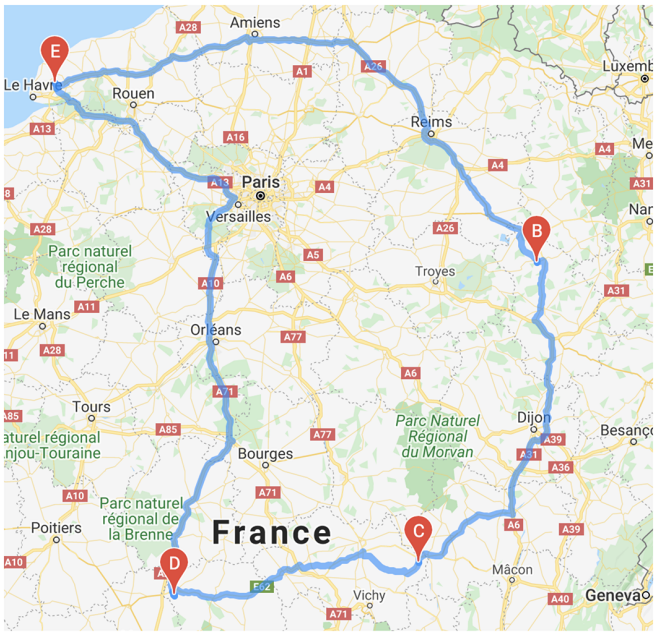
SHOW HINT

11. In Step 7, use the **gmaps documentation (https://jupyter-gmaps.readthedocs.io/en/latest/tutorial.html#directions-layer)** to create a

gmaps.readthedocs.io/en/latest/tutorial.html#directions-layer) to create a directions layer map using the variables from Step 6, where the starting and ending city are the same city, the `waypoints` are the three other cities, and the `travel_mode` is either `"DRIVING"`, `"BICYCLING"`, or `"WALKING"`.

12. Take a screenshot of your map from Step 7 and save it to the Vacation_Itinerary folder as `WeatherPy_travel_map.png`.

    - The directions layer map should look similar to the following image:



13. In Step 8, use the provided `concat()` function code snippet to combine the four separate city DataFrames into one DataFrame.

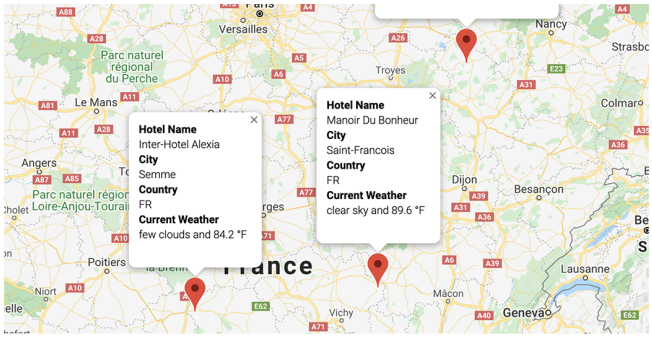
If you'd like a hint on combining DataFrames into one DataFrame using the `concat()` function, that's totally okay. If not, that's great too. You can always revisit this later if you change your mind.

SHOW HINT

14. In Steps 9-11, refactor the code from Steps 2-4 to create a new marker layer map of the cities on the travel route.

15. Take a screenshot of your map and save it to the Vacation_Itinerary folder as `WeatherPy_travel_map_markers.png`.

    - The marker layer map with a pop-up marker for each city should look similar to the following image:



Deliverable 3 Requirements

You will earn a perfect score for Deliverable 3 by completing all requirements below:

- Four DataFrames are created, one for each city on the itinerary. **(10 pt)**
- The latitude and longitude pairs for each of the four cities are retrieved. **(5 pt)**
- A directions layer map between the cities and the travel map is created and uploaded as `WeatherPy_travel_map.png`. **(10 pt)**
- A DataFrame that contains the **four** cities on the itinerary is created. **(10 pt)**
- A marker layer map with a pop-up marker for the cities on the itinerary is created, and it is uploaded as `WeatherPy_travel_map_markers.png`. Each marker has the following information: **(5 pt)**
  - Hotel name
  - City
  - Country
  - Current weather description with the maximum temperature

Be sure to double-check that you have the following in the Vacation_Itinerary folder:

Vacation_Itinerary folder:

- The `Vacation_Itinerary.ipynb` file
- The `WeatherPy_travel_map.png` image
- The `WeatherPy_travel_map_markers.png` image

Submission

Once you're ready to submit, make sure to check your work against the rubric to ensure you are meeting the requirements for this Challenge one final time. It's easy to overlook items when you're in the zone!

As a reminder, the deliverables for this Challenge are as follows:

- Deliverable 1: Retrieve Weather Data
- Deliverable 2: Create a Customer Travel Destinations Map
- Deliverable 3: Create a Travel Itinerary Map

Upload the following to your WeatherPy GitHub repository:

- The Weather_Database folder with the following:
  - The `Weather_Database.ipynb` file
  - The `WeatherPy_Database.csv` file
- The Vacation_Search folder with the following:
  - The `Vacation_Search.ipynb` file
  - The `WeatherPy_vacation.csv` file
  - The `WeatherPy_vacation_map.png` image

- The Vacation_Itinerary folder with the following:
  - The `Vacation_Itinerary.ipynb` file
  - The `WeatherPy_travel_map.png` image
  - The `WeatherPy_travel_map_markers.png` image
- A README.md that describes the purpose of the repository. Although there is no graded written analysis for this challenge, it is encouraged and good practice to add a brief description of your project.

IMPORTANT

> Do not include your `config.py` file in your submission.

If you'd like a hint on how to not include the `config.py` file when adding your files to your GitHub repository, that's totally okay. If not, that's great too. You can always revisit this later if you change your mind.

SHOW HINT

To submit your challenge assignment in Canvas, click Submit, then provide the URL of your WeatherPy GitHub repository for grading. Comments are disabled for graded submissions in BootCampSpot. If you have questions about your feedback, please notify your instructional staff or the Student Success Manager. If you would like to resubmit your work for an improved grade, you can use the **Re-Submit Assignment** button to upload new links.

You may resubmit up to 3 times for a total of 4 submissions.

IMPORTANT

> Once you receive feedback on your Challenge, make any suggested updates or adjustments to your work. Then, add this week's Challenge to your professional portfolio.

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.

Module-6 Rubric

Criteria	Ratings					Pts
Deliverable 1: Retrieve Weather Data	**25 to >21.0 pts Demonstrating Proficiency** ✓The deliverable fulfills the “Emerging” criteria AND the following: ✓All the weather data is added to a new DataFrame. ✓The DataFrame is exported and saved as a CSV.	**21 to >18.0 pts Approaching Proficiency** ✓The deliverable fulfills the “Emerging” criteria AND the following: ✓All the weather data is added to a new DataFrame. ✓Code is written to export the DataFrame as a CSV, but there is an error to save it.	**18 to >15.0 pts Developing Proficiency** ✓The deliverable fulfills the “Emerging” criteria AND the following: ✓Most of the weather data is added to a new DataFrame.	**15 to >0.0 pts Emerging** ✓All of the required data from the API is retrieved.	**0 pts Incomplete**	25 pts

Criteria	Ratings					Pts
Deliverable 2: Create a Customer Travel Destinations Map	**35 to >32.0 pts Demonstrating Proficiency** ✓Input statements are written to get the minimum and maximum temperature. ✓A new DataFrame is created based on the weather criteria, and empty rows are dropped. ✓The hotel name is added to the DataFrame, and the empty rows are dropped. ✓The DataFrame is exported and saved as a CSV file. ✓A marker layer map is created with a pop-up marker for each city that has all the correct data. ✓The marker layer map is saved as a PNG	**32 to >27.0 pts Approaching Proficiency** ✓Input statements are written to get the minimum and maximum temperature. ✓A new DataFrame is created based on the weather criteria, and empty rows are dropped. ✓The hotel name is added to the DataFrame, and the empty rows are dropped. ✓The DataFrame is exported and saved as a CSV file. ✓A marker layer map is created with a pop-up marker for each city, but some cities don't have all the data. ✓The marker layer map is saved as a PNG	**27 to >24.0 pts Developing Proficiency** ✓Input statements are written to get the minimum and maximum temperature. ✓A new DataFrame is created based on the weather criteria, and empty rows are dropped. ✓The hotel name is added to the DataFrame, but the empty rows are not dropped. ✓The DataFrame is exported and saved as a CSV file. ✓A marker layer map is created with a pop-up marker for each city, but some cities don't have all the data. ✓The marker layer map is saved as a PNG	**24 to >0.0 pts Emerging** ✓Input statements are written to get the minimum and maximum temperature. ✓A new DataFrame is created based on the weather criteria, but the empty rows are not dropped. ✓The hotel name is added to the DataFrame, but the empty rows are not dropped. ✓The DataFrame is exported and saved as a CSV file. ✓A marker layer map is created with a pop-up marker for each city, but some cities don't have all the data. ✓The marker layer map is saved as a PNG	**0 pts Incomplete**	35 pts

Criteria	Ratings					Pts
Deliverable 3: Create a Travel Itinerary Map	**40 to >36.0 pts Demonstrating Proficiency** ✓Four DataFrames are created, one for each city in the itinerary. ✓The latitude and longitude pairs for each city are retrieved to create the directions layer map. ✓A directions layer map between the cities and the travel map is uploaded as a PNG. ✓A DataFrame that contains the four cities on the itinerary is created. ✓A marker layer map with a pop-up marker for the cities in the itinerary is created, and is uploaded as a PNG.	**36 to >34.0 pts Approaching Proficiency** ✓Four DataFrames are created, one for each city in the itinerary. ✓The latitude and longitude pairs for each city are retrieved to create the directions layer map. ✓There is a directions layer map between THREE of the FOUR cities, and the travel map is uploaded as a PNG. ✓A DataFrame that contains the four cities on the itinerary is created. ✓A marker layer map with a pop-up marker for the cities in the itinerary is created, and is uploaded as a PNG.	**34 to >31.0 pts Developing Proficiency** ✓Four DataFrames are created, one for each city in the itinerary. ✓The latitude and longitude pairs for each city are retrieved to create the directions layer map. ✓There is a directions layer map between TWO of the FOUR cities, and the travel map is uploaded as a PNG. ✓A DataFrame that contains the four cities on the itinerary is created. ✓A marker layer map with a pop-up marker for the cities in the itinerary is created, and is uploaded as a PNG.	**31 to >0.0 pts Emerging** ✓Four DataFrames are created, one for each city in the itinerary. ✓Code is written to retrieve the latitude and longitude pairs for each of the four cities. ✓Code is written but a directions layer map isn't created between the cities. ✓A DataFrame that contains the four cities on the itinerary is created. ✓A marker layer map with a pop-up marker for the cities in the itinerary is created, and is uploaded as a PNG.	**0 pts Incomplete**	40 pts
						Total Points: 100