6.1.5

Generate Random World Cities

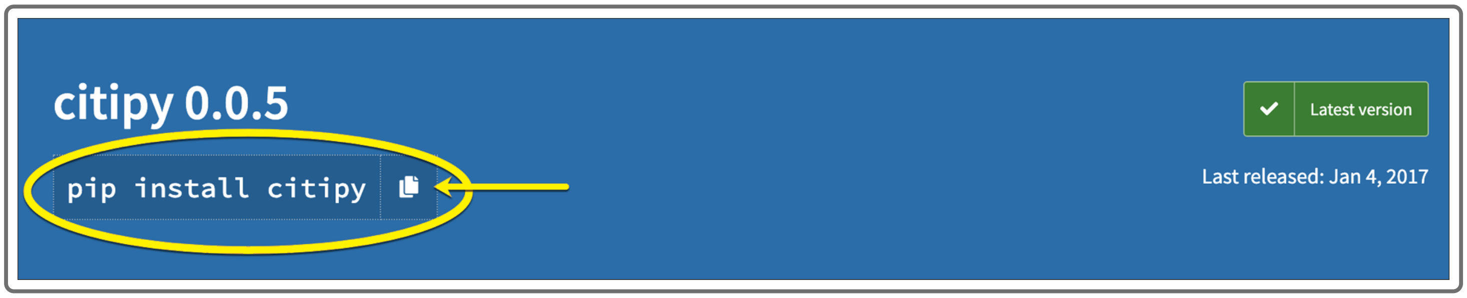
You lean back from your desk and reflect on the project thus far. Not only have you successfully navigated a quick return to the geographic coordinate system and the Earth's geography, you also wrote code to generate 1,500 latitudes and longitudes. Now, as cool as this is, your clients aren't going to want information or suggestions provided to them in this format. So, it's time to get started on the next step in your project plan: match those coordinates up with cities.

We are making great progress. With our list of random latitudes and longitudes, we'll use the coordinates in our `lat_lngs` tuple to find the nearest city using Python's citipy module.

Since we haven't worked with the citipy module yet, let's import and test it. Citipy doesn't come with the Anaconda module, so we'll install it in our PythonData environment.

The citypy documentation (https://pypi.org/project/citipy/) instructs us to install the citipy module by typing `pip install citipy`. To complete the installation, follow the instructions for your operating system.

Check out the macOS instructions below, or jump to the **Windows instructions**.



```
pip install citipy
```

macOS

To install the citipy module on macOS, complete the following steps:

1. Click the clipboard icon to copy `pip install citipy`.
2. Launch the command line and make sure you are in your PythonData environment. You should see the following in your terminal:

```
(PythonData) your_computer_name:~ your_home_directory$
```

3. Paste `pip install citipy` and press Enter.

The citipy module will probably take a few minutes to download into your PythonData environment.

Windows

To install the citipy module on Windows, complete the following steps:

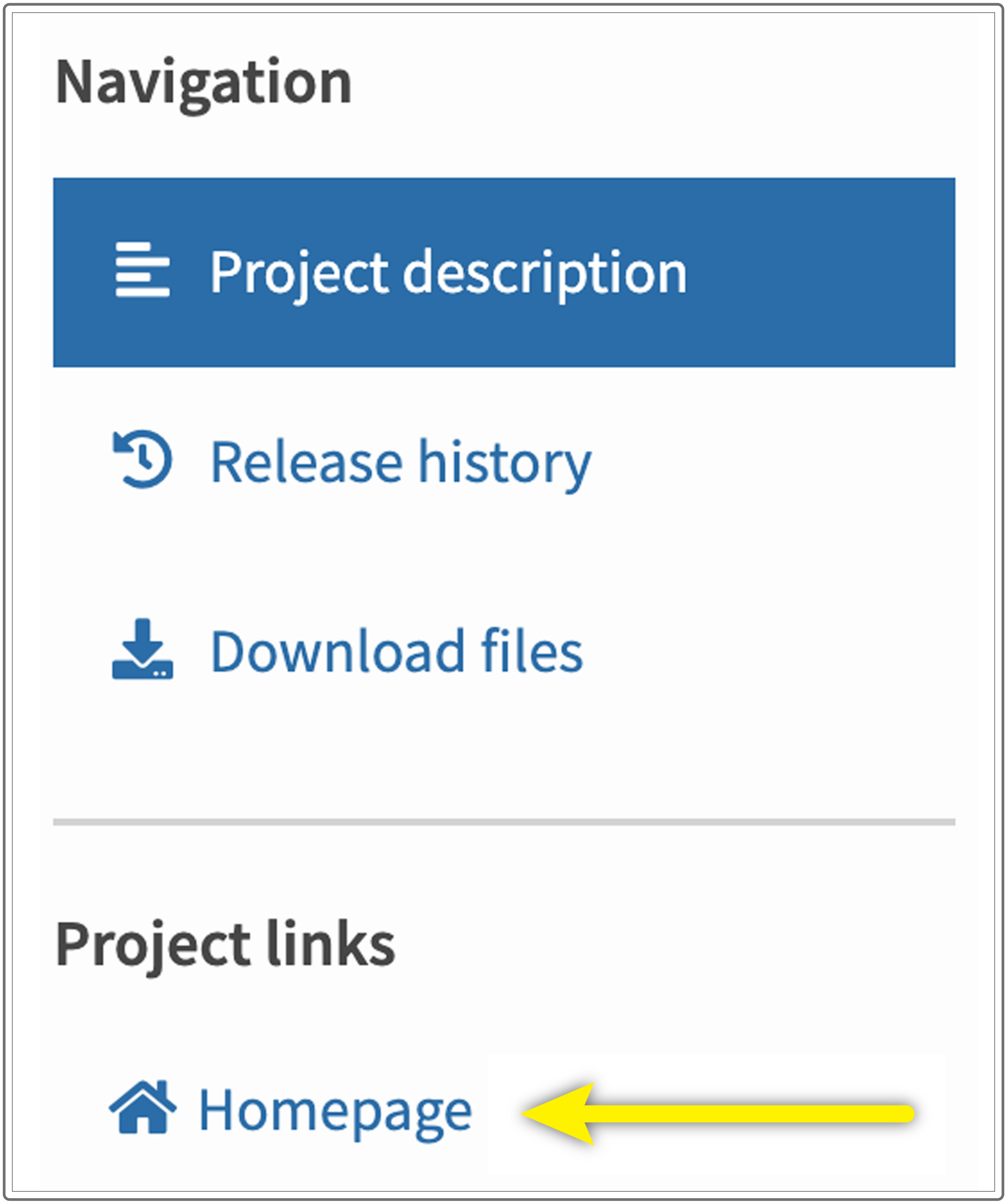
1. Click the clipboard to copy `pip install citipy`.
2. Launch your PythonData Anaconda Prompt. You should see the following:

```
(PythonData) C:\Users\your_computer_name>
```

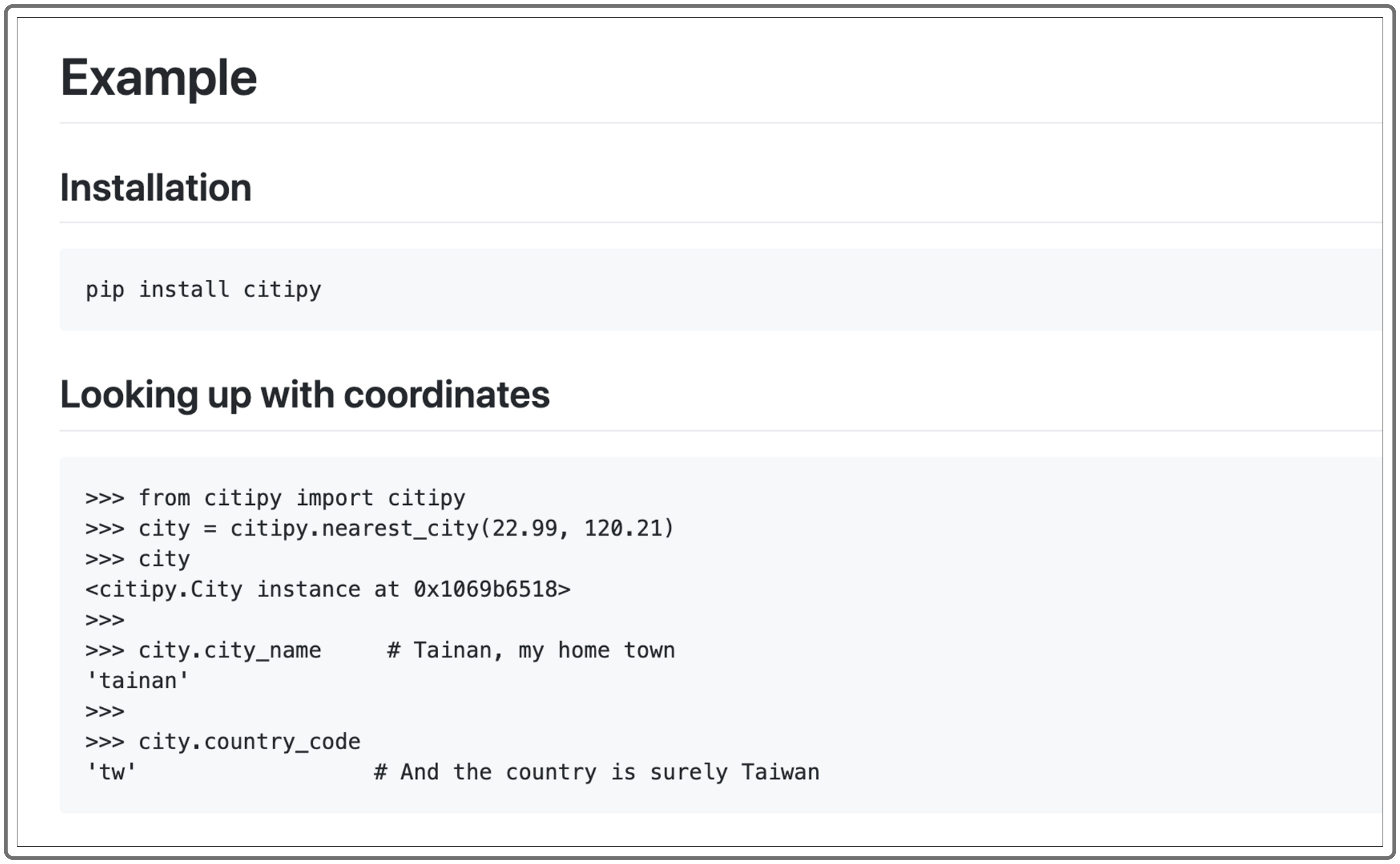
3. Paste `pip install citipy` and press Enter.

The citipy module will probably take a few minutes to download into your PythonData environment.

To learn how to use citypy, click "Homepage" on the module webpage or see the **GitHub citipy repository (https://github.com/wingchen/citipy)** .



Select the `README.md` file on the citipy GitHub page for an example of how to use citipy to locate the nearest city and its country code from a pair of latitude and longitude coordinates.



Example

Installation

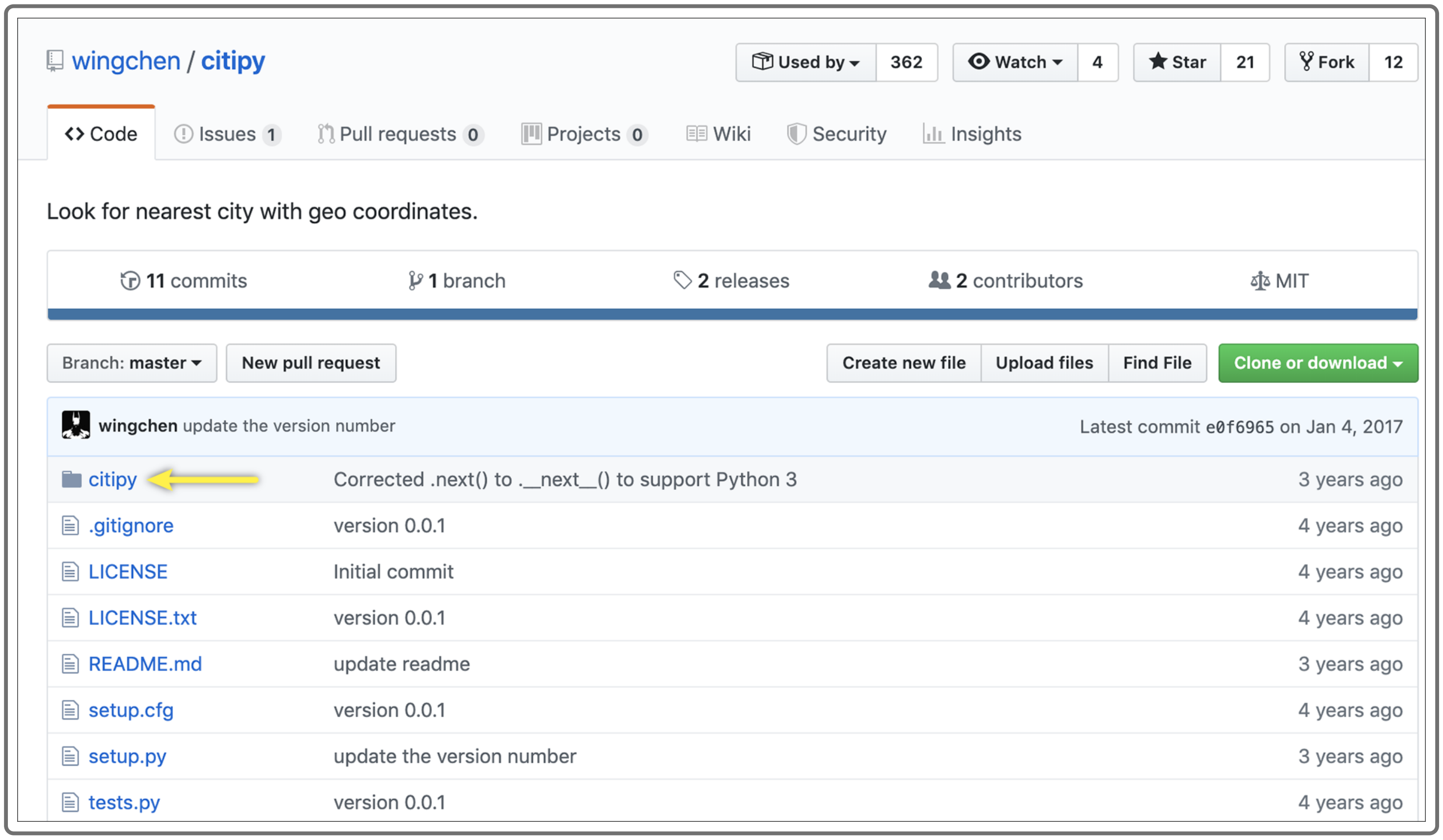
```
pip install citipy
```

Looking up with coordinates

```
>>> from citipy import citipy
>>> city = citipy.nearest_city(22.99, 120.21)
>>> city
<citipy.City instance at 0x1069b6518>
>>>
>>> city.city_name     # Tainan, my home town
'tainan'
>>>
>>> city.country_code
'tw'                  # And the country is surely Taiwan
```

Under "Looking up with coordinates," the first line says `from citipy import citipy`, meaning we'll import the `citipy` script from the citipy module.

This script is located in the GitHub repository as indicated by the arrow in the following image.



Look for nearest city with geo coordinates.

11 commits | 1 branch | 2 releases | 2 contributors | MIT

wingchen update the version number — Latest commit e0f6965 on Jan 4, 2017

citipy	Corrected .next() to .__next__() to support Python 3	3 years ago
.gitignore	version 0.0.1	4 years ago
LICENSE	Initial commit	4 years ago
LICENSE.txt	version 0.0.1	4 years ago
README.md	update readme	3 years ago
setup.cfg	version 0.0.1	4 years ago
setup.py	update the version number	3 years ago
tests.py	version 0.0.1	4 years ago

NOTE

When a Python file containing a script is imported to use in another Python script, the `.py` extension does not need to be added to the name of the file when using the import statement.

Let's import the `citipy` script and practice using it. In our `API_practice` file, add a new cell and import the `citipy.py` script from the citipy module.

```
# Use the citipy module to determine city based on latitude and longitude.
from citipy import citipy
```

Next, use the five pairs of latitudes and longitudes we used from our zip practice to get a city and country code from the citipy module.

In a new cell, create a `for` loop that will do the following:

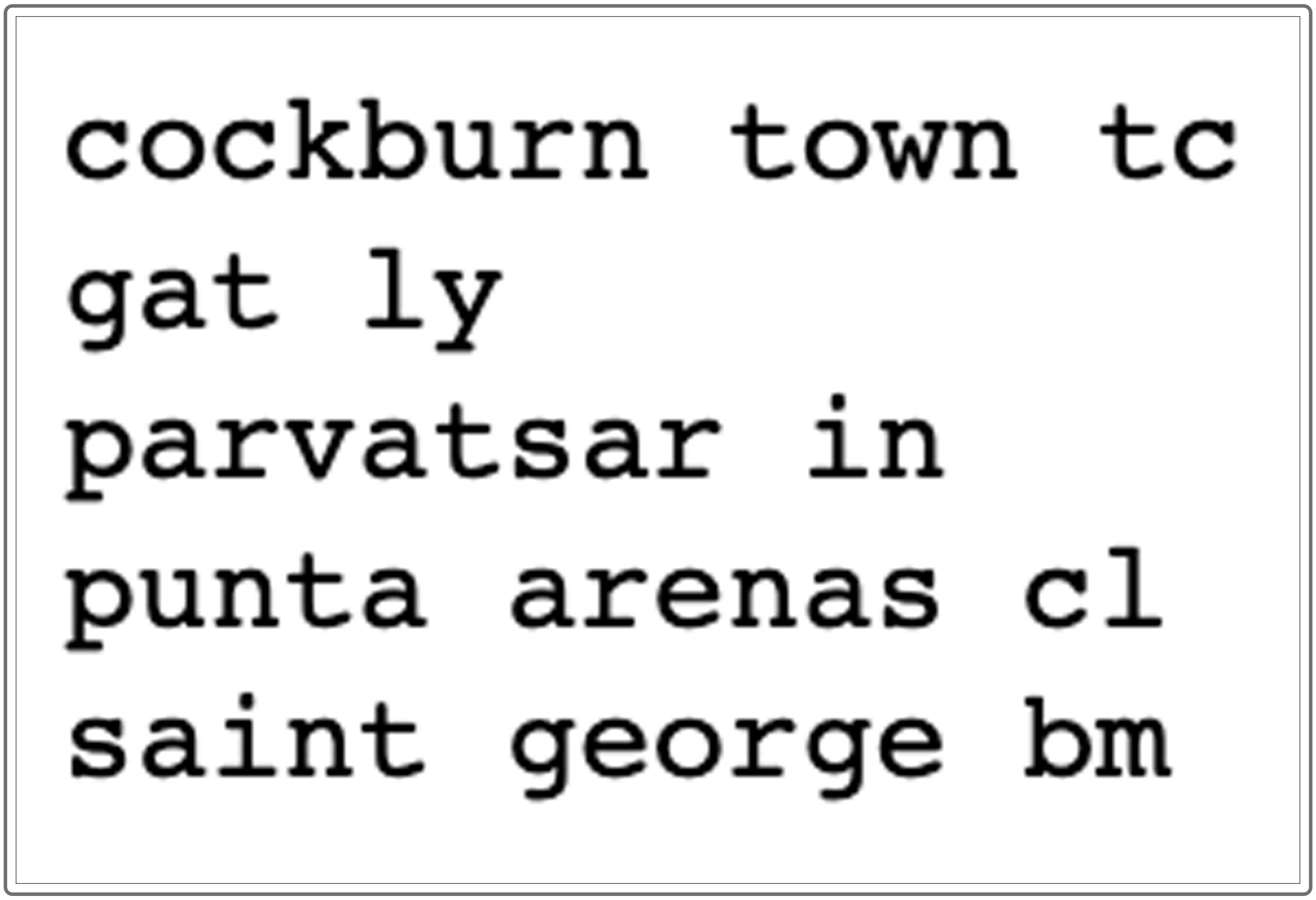
1. Iterate through the coordinates' unzipped tuple.

2. Use `citipy.nearest_city()` and inside the parentheses of `nearest_city()`, add the latitude and longitude in this format: `coordinate[0], coordinate[1]`.
3. To print the city name, chain the `city_name` to the `nearest_city()` function.
4. To print the country name, chain the `country_code` to the `nearest_city()` function.

Your code should look like the following. Add the code to a new cell and run the cell.

```
# Use the print() function to display the latitude and longitude combinati
for coordinate in coordinates:
    print(citipy.nearest_city(coordinate[0], coordinate[1]).city_name,
          citipy.nearest_city(coordinate[0], coordinate[1]).country_code)
```

When we run this cell, the output will show five cities with their associated country codes.



```
cockburn town tc
gat ly
parvatsar in
punta arenas cl
saint george bm
```

Now that we are familiar with using the citipy module, we can iterate through our zipped `lat_lngs` tuple and find the nearest city. When we find a city, we'll need to add it to a list so that we can use the cities to get the weather data.

First, import the `citipy` module in our `WeatherPy` file.

```
from citipy import citipy
```

Then, in a new cell, add the following code:

```
# Create a list for holding the cities.
cities = []
# Identify the nearest city for each latitude and longitude combination.
for coordinate in coordinates:
    city = citipy.nearest_city(coordinate[0], coordinate[1]).city_name

    # If the city is unique, then we will add it to the cities list.
    if city not in cities:
        cities.append(city)
# Print the city count to confirm sufficient count.
len(cities)
```

Some of this code should look familiar, but let's break it down:

1. We create a `cities` list to store city names.
2. We iterate through the `coordinates`, as in our practice, and retrieve the nearest city using the latitude and longitude pair.
3. We add a decision statement with the logical operator `not in` to determine whether the found city is already in the `cities` list. If not, then we'll use the `append()` function to add it. We are doing this because among the 1,500 latitudes and longitudes, there might be duplicates, which will retrieve duplicate cities, and we want to be sure we capture only the unique cities.

NOTE

The citipy module finds the nearest city to the latitude and longitude pair with a population of 500 or more.

FINDING

When you run the code block, you should get slightly more than 500 unique cities. If you get fewer than 500, increase your `size` limit on the `np.random.uniform()` function.

ADD/COMMIT/PUSH

Update or add your code to your GitHub repository for this module.