**What you will need**

**Software**

**Software installation**

To install the software you need, run the following command in a terminal:

sudo apt-get install python3-requests

Each Weather Station comes equipped with the sensors shown in the table below:

| **Sensor Name** | **Purpose** |
| --- | --- |
| Rain gauge | Measures the volume of rain falling in millimetres |
| Anemometer | Measures the wind speed in kilometres per hour |
| Weathervane | Measures the wind direction in degrees |
| Soil temperature probe | Measures the soil temperature in degrees Celsius |
| Temperature sensor | Measures the air temperature in degrees Celsius |
| Humidity sensor | Measures the relative humidity of the air as a percentage |
| Pressure sensor | Measures the atmospheric pressure in Pascals |
| Air quality sensor | Measures the air quality as a relative percentage |

The Weather Stations continually monitor the weather and then send their data to an Oracle database, where it is stored and can be accessed.

In this resource you’re going to learn how to find a Weather Station you’re interested in, and then get the latest weather updates from that station.

## Finding a Weather Station

You can get a list of all the Weather Stations that are currently online, using a simple URL. This is because the database that all the Weather Stations upload data to has a RESTful API. This is a method by which you can write code that uses simple HTTP requests (just like a browser) to fetch the data.

Copy and paste the following URL into a web browser:

https://apex.oracle.com/pls/apex/raspberrypi/weatherstation/getallstations

You should see a web page filled with data. This is a little difficult to read, though. Luckily, we can grab this data with a little Python code and then present it in a format that’s easier to read.

Click on Menu > Programming > Python3 (IDLE) to open a new Python shell, then click on File > New File.

The first thing you’ll need is a few Python modules. One of them is not in the standard library, but you can install it using the instructions from the [What you will need section](https://projects.raspberrypi.org/en/projects/fetching-the-weather/What%20you%20will%20need).

from requests import get

import json

from pprint import pprint

The requests module allows you to fetch web pages from the World Wide Web. The json module allows you to easily read JSON data (which is a way of organising data into dictionaries). The pprint module is short for pretty-print, and just makes presenting text a little clearer.

The next thing to do is to save that URL you used earlier as a variable:

url = 'https://apex.oracle.com/pls/apex/raspberrypi/weatherstation/getallstations'

Using get from the requests module you can now fetch the data, and translate it into Python dictionaries using the json module:

stations = get(url).json()['items']

Save and run your code. You can type stations into the Python shell to have a look at the data.

It still looks pretty ugly. Try typing pprint(stations) and see what happens. You should see a huge list of Weather Stations dictionaries. Each dictionary should look something like this:

{'weather\_stn\_id': 1648902,

'weather\_stn\_lat': 52.197834,

'weather\_stn\_long': 0.125366,

'weather\_stn\_name': 'ACRG\_ROOF'}]

What you’re seeing is the unique ID of the station, its location in the world using longitude and latitude (You can learn about this in [later on](https://projects.raspberrypi.org/en/projects/fetching-the-weather/longitude-and-latitude)), and the name of the Weather Station.

For the next part, you’re going to need to pick a Weather Station to fetch the weather from. Scroll up and down the list and pick a weather\_stn\_id that you’d like to have a look at.

**Fetching the latest weather**

Now that you have a Weather Station to look at, you can learn how to fetch the last weather recording from that station.  
This is again handled using the RESTful API of the Weather Station database. This time, the URL you need is made up of two parts. The first tells the database that you’re requesting the latest measurements:

'https://apex.oracle.com/pls/apex/raspberrypi/weatherstation/getlatestmeasurements/'

You need to add the ID of the Weather Station you wish to access to the end of this. For example:

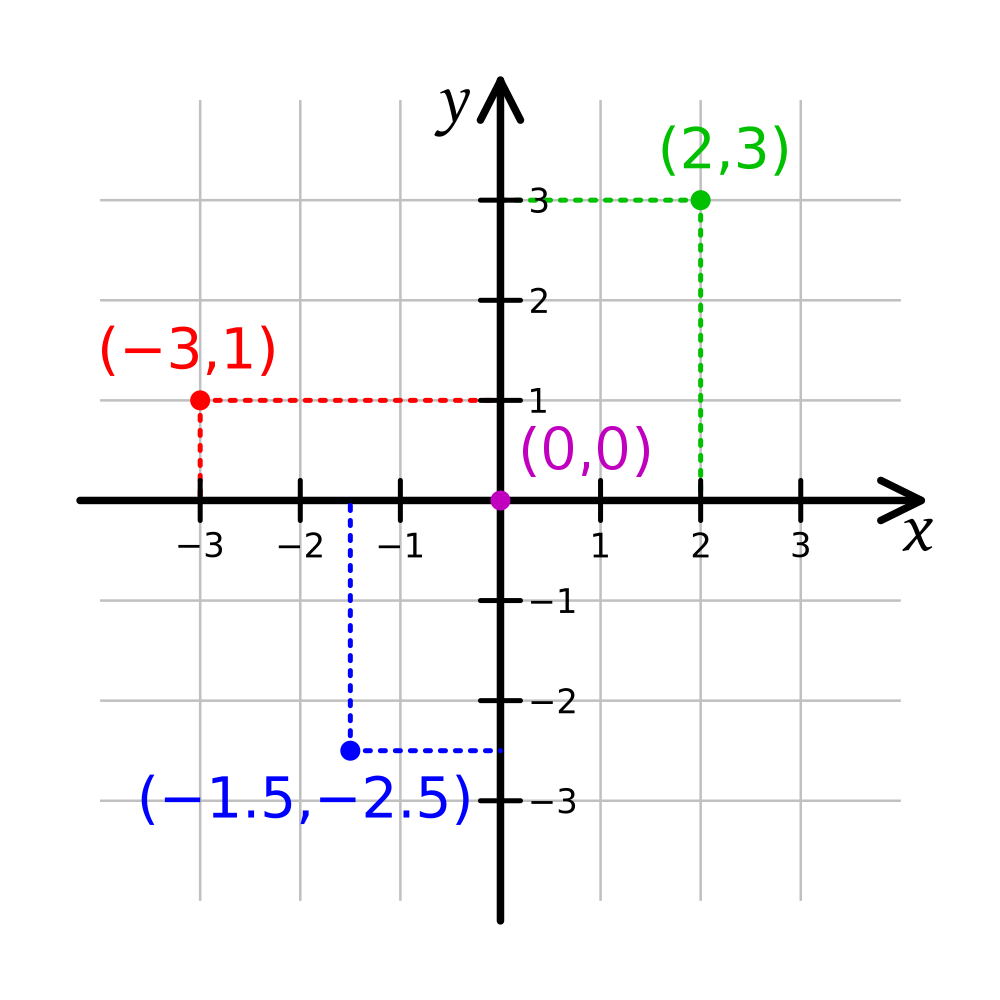
'https://apex.oracle.com/pls/apex/raspberrypi/weatherstation/getlatestmeasurements/1648902'

* Create a new Python file again, by clicking on File > New File.
* Once again, you’ll need the requests and json modules, as well as pprint:
* from requests import get
* import json
* from pprint import pprint
* Now you can define a new url variable, but using the Weather Station ID you’ve chosen:
* url = 'https://apex.oracle.com/pls/apex/raspberrypi/weatherstation/getlatestmeasurements/weather\_stn\_id\_goes\_here'
* To get the latest measurements you need one line of code, but we’ll add a second line to pretty-print it straight away:
* weather = get(url).json()['items']
* pprint(weather)
* You should see something like the following appearing in the shell:
* >>> [{'air\_pressure': 1008.81,
* 'air\_quality': 74.9,
* 'ambient\_temp': 23.58,
* 'created\_by': 'ACRG\_ROOF',
* 'created\_on': '2016-11-16T12:00:01Z',
* 'ground\_temp': 18.69,
* 'humidity': 33.41,
* 'id': 1669238,
* 'rainfall': 0,
* 'reading\_timestamp': '2016-11-16T12:00:01Z',
* 'updated\_by': 'ACRG\_ROOF',
* 'updated\_on': '2016-11-16T12:05:02.437Z',
* 'weather\_stn\_id': 1648902,
* 'wind\_direction': 315,
* 'wind\_gust\_speed': 0,
* 'wind\_speed': 0}]
* If you don’t see any data, it might be because the Weather Station is offline. Just try another Weather Station Id.

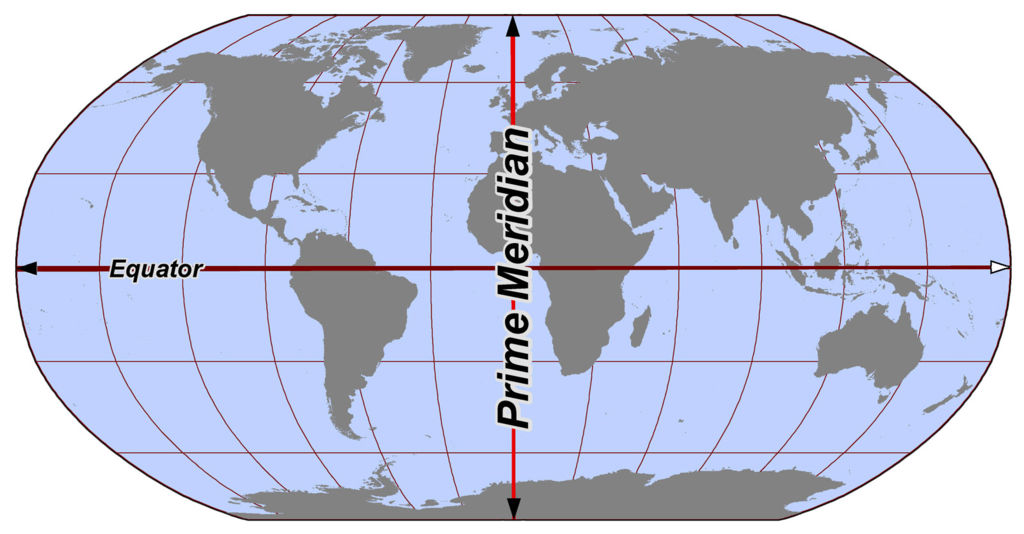
## Longitude and latitude

It would be nice if you could pick a Weather Station that’s close to you to fetch the data from. You can do this, because the database stores the longitude and latitude of all the Weather Stations around the world. Let’s have a look at what we mean by longitude and latitude.

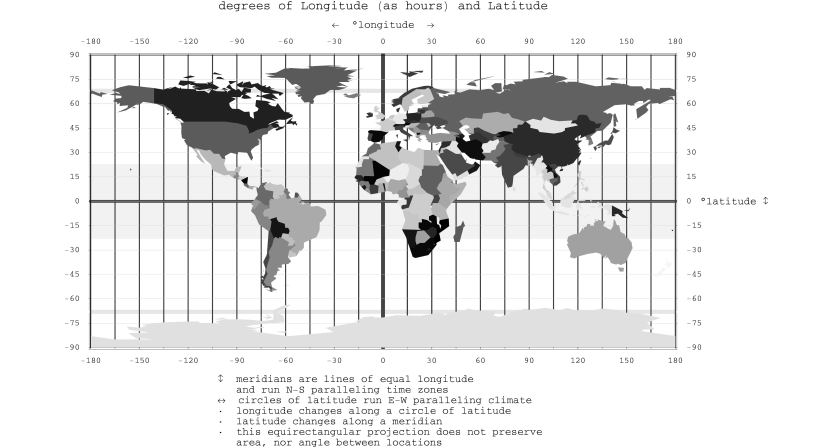
* If you wanted to pinpoint a place on a 2D object like a piece of paper, you could use x and y coordinates. The x coordinate would place the point’s horizontal position, and the y coordinate would place the vertical position. You can see an example of this below.



* Things aren’t so simple when you’re trying to pinpoint a location on a sphere, like the Earth. The vertical and horizontal positions wrap around the sphere, for a start. Also, travelling 5 units of distance along the equator would be a completely different distance to walking 5 units of distance near one of the poles. For this reason we use longitude and latitude when locating items on the Earth’s surface.
* You can draw two imaginary circles around the Earth. The first is called the equator, which you’re probably familiar with. The second is called the prime meridian, which passes through both the North and South Poles and also through Greenwich in London.



* The centre of these two circles is at the centre of the Earth. Imagine you were standing in the centre of the Earth; you would be able to pinpoint any location on the surface by talking about how many degrees you needed to turn within each of these circles. Longitude tells you how many degrees you need to turn east or west from the prime meridian. Latitude tells you how many degrees you need to turn north or south from the equator.



* The easiest way to find your longitude and latitude is to use [Google Maps](https://www.google.co.uk/maps/). You can click on any spot on the map, and your longitude and latitude will be revealed at the bottom of the screen.
* The first number is your latitude and the second is your longitude. Make a note of the values you get, as you’ll need them later.

## Getting ready

In the first worksheet, you fetched all the Weather Stations that are currently registered. The data came in as a huge list of dictionaries. By iterating through this list, you can pick out the longitude and latitude of the Weather Stations, and then run it through your haversine function to find the closest one.

* Create a new Python file (File > New File) and make sure you save it in the same directory as your haversine.py file.
* Start by importing the requests, json, and pprint modules that you used in Worksheet One, but you can now also import your haversine function:
* from requests import get
* import json
* from pprint import pprint
* from haversine import haversine
* In Worksheet One, you used two URLs to get the Weather Stations and the latest weather. You can declare these variables straight away:
* stations = 'https://apex.oracle.com/pls/apex/raspberrypi/weatherstation/getallstations'
* weather = 'https://apex.oracle.com/pls/apex/raspberrypi/weatherstation/getlatestmeasurements/'
* The second URL isn’t complete, as you need to add the weather station ID to the end. You’re going to do that in code.
* Now add in variables for your current longitude and latitude, that you found using Google Maps:
* my\_lat = 52.194504
* my\_lon = 0.134708
* To finish off this section, you can fetch the list of all stations, just like you did in Worksheet One:

all\_stations = get(stations).json()['items']

## Finding the closest station

For this to work, you’re going to need to run the longitude and latitude of all the stations through the haversine function. The trick will be finding the smallest distance to your current longitude and latitude, and saving this as a variable.

Start by defining a new function, and setting a variable within it for the smallest distance. The longest possible distance between two points on the Earth’s surface is 20036km, so this would be a good place to start the variable:

def find\_closest():

smallest = 20036

Now you can use a for loop to iterate through all the stations. Let’s start by printing the data for each:

for station in all\_stations:

print(station)

To get the list of stations you need to run your function, so type the following into the shell:

find\_closest()

You should see a large list of dictionaries, with each dictionary looking something like this:

{'weather\_stn\_name': 'ACRG\_ROOF', 'weather\_stn\_lat': 52.197834, 'weather\_stn\_id': 1648902, 'weather\_stn\_long': 0.125366}

The data we’re interested in is the 'weather\_stn\_lat' and 'weather\_stn\_long'. These are the values we want to use in the haversine function.

Go back to your script; you can now get those values in your function. Remove the print(station) line and then add the following:

station\_lon = station['weather\_stn\_long']

station\_lat = station['weather\_stn\_lat']

Now that you have all the data, it can be run through the haversine function to find the station’s distance to you:

distance = haversine(my\_lon, my\_lat, station\_lon, station\_lat)

print(distance)

Run the code again and type find\_closest() in the shell again.

That’s a long list of distances. Next, you need to find the smallest one and then save that station’s ID. If the distance is smaller than the smallest variable it can be saved, and then next time around the loop it can be checked again.

if distance < smallest:

smallest = distance

closest\_station = station['weather\_stn\_id']

return closest\_station

Your find\_closest function should now look like this:

def find\_closest():

smallest = 20036

for station in all\_stations:

station\_lon = station['weather\_stn\_long']

station\_lat = station['weather\_stn\_lat']

distance = haversine(my\_lon, my\_lat, station\_lon, station\_lat)

if distance < smallest:

smallest = distance

closest\_station = station['weather\_stn\_id']

return closest\_station

## Getting the weather data

Now that you can get the closest Weather Station to you, getting the data is just as easy as it was in Worksheet One.

Start by calling your newly created function and saving the weather station ID:

closest\_stn = find\_closest()

Now this can be added to the end of the weather variable that stores the URL. It’s an integer at the moment though, so it needs to be changed to a string:

weather = weather + str(closest\_stn)

Finally, you can use requests to get the data and then pretty-print it:

my\_weather = get(weather).json()['items']

pprint(my\_weather)

Run your code and you should see the weather data for the station nearest you, printed out in the shell.

POST

HTTP GET and POST requests can be used to communicate with web servers ,api etc. Using this method you can develop some cool apps with your Raspberry Pi. In this tutorial we are going to see how to make HTTP GET and POST requests with your Raspberry Pi.

### Making HTTP GET requests

In Raspberry Pi the HTTP GET requests can be made using the package **urllib**. The code is very simple. By using the method **urllib.urlopen()** you can get the contents of the page. In this example I am making HTTP GET request to the url “<http://api.learn2crack.com/rpi/rpi_get.php>”. It will print the output as “Hello Raspberry Pi”.

**Example**

import urllib

url = "<http://api.learn2crack.com/rpi/rpi_get.php>"

response = urllib.urlopen(url).read()

print response

### Making HTTP POST requests

Http POST requests can be made using the package **urllib** and **urllib2**. The post parameters should be in the format of { ‘data’:’data’ }. The post request can be made using the method **urllib2.Request()**. By using the method **urllib2.urlopen()** you can get the contents of the page. In this example I am making HTTP POST request to the url “<http://api.learn2crack.com/rpi/rpi_post.php>”. It will print the output as “State is : Active”.

**Example**

import urllib2

import urllib

query\_args = { 'RPI':'Active' }

url = '<http://api.learn2crack.com/rpi/rpi_post.php>'

data = urllib.urlencode(query\_args)

request = urllib2.Request(url, data)

response = urllib2.urlopen(request).read()

print response

### Steps to execute the code

1. Open the Raspberry Pi terminal.

2. Then type **nano program.py**

3. Now type the python program and save it.

4. Execute using the command **python program.py**

References

<https://stackoverflow.com/questions/30459762/using-python-script-to-post-data-to-web-server>

<https://requestb.in/1nxhwu71?inspect>

<https://www.learn2crack.com/2014/03/raspberry-pi-post-get.html>

<https://projects.raspberrypi.org/en/projects/mapping-the-weather>