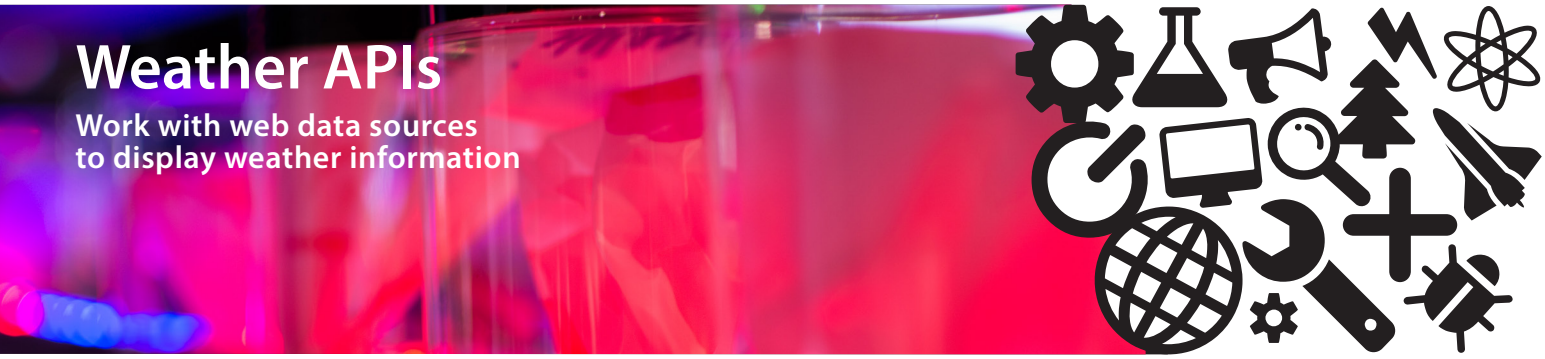
nustem



Weather APIs

Work with web data sources to display weather information

Ingredients

Platform:	Raspberry Pi / Raspbian Python3
Components:	ScrollpHAT HD
Libraries:	pyowm PyQuery

Glossary

API	"Application Program Interface" – a documented set of functions and methods which allows one program to access data from another.
object	A packaging of data and ways of accessing that data: an object likely has methods you can call which prompt it to output the data it holds in a range of ways.

Getting the weather – the easy way

Look out of the window.

Getting the weather – the slightly harder way

All the code for this worksheet is in `Digital-Making/examples/weatherapi/`. You can start by loading `weather.py` in Thonny, or just with a new script.

We're going to use a Python module called PyOWM (Open Weather Map) to access current weather reports. OWM requires you to sign up for an *API Key* in order to access their services. We've handily included one in the file `clientsecrets.py`, so:

```
import pyowm
from clientsecrets import owmkey
```

Now we have to set up an object to query the OWM service, and retrieve the current weather observations:

```
owm = pyowm.OWM(owmkey)
observation = owm.weather_at_place('Newcastle upon Tyne,GB')
```

The first line here sets up an OWM object associated with our API key, the second asks it for a current report from Newcastle, storing the response in the new object `observation`.

`observation` stores all manner of things, so we first ask for just the weather bit:

```
w = observation.get_weather()
```

...then we can print things like the current temperature:

```
print(w.get_temperature())
```

See `weather.py` for a longer list of information you can extract from OWM.

PyOWM is a nice example of a clean Python module which wraps up calls to a web API for you. The documentation is good, with lots of examples: `https://pyowm.readthedocs.io/en/latest/`

Displaying stuff on the ScrollpHAT HD

This Pi has a Pimoroni ScrollpHAT HD attached – a decent-sized array of LED lights. The program `scrollphat-test.py` scrolls a short message for a few seconds, then shows the time, and repeats.

Challenge: get the ScrollpHAT HD to show some weather data, by pasting in bits of the `weather.py` code.

This exercise is based on a Pi I have sitting on a shelf at home in one of Pimoroni's kits, as above. It displays a rolling update of the time, pollen forecast, air pressure, and current temperature, with the weather data updated every ten minutes. If you're interested, the code is in `databot.py`. You'll spot some new ideas in there – see you can work out what's going on.

Getting pollen data

I'd originally intended for this to be an integral part of the worksheet. But on reflection (and with Joe giving me long hard stares of "*Really?*") I've taken it out. But then left it here as sort-of a worked example. Getting data from web pages by scraping them and parsing their contents is an important and useful technique, but it's also fiddly.

Open Weather Map unfortunately doesn't hold pollen report or forecast data for the UK. There's a Python module called `pypollen`, but it's not very reliable... so we're going to do this the hard way.

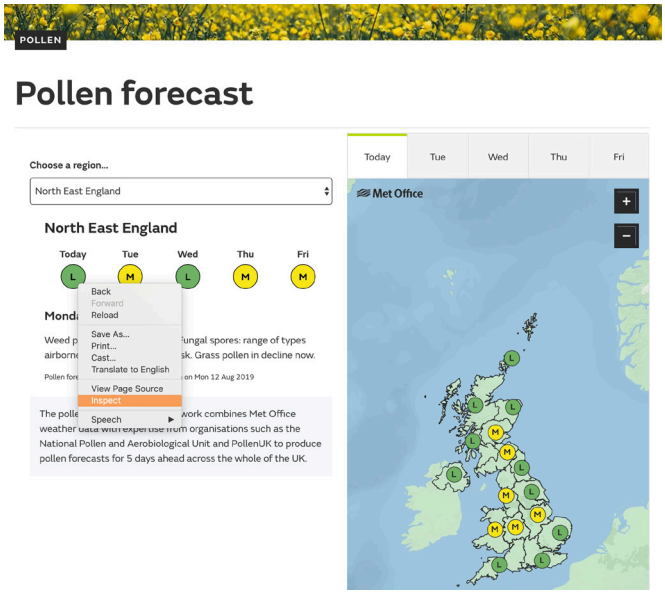
The Met. Office posts pollen forecasts here:

`https://www.metoffice.gov.uk/weather/warnings-and-advice/seasonal-advice/pollen-forecast`

There are several Python modules which can read a web page and drill into the HTML code to find specific bits. The one we're going to use here is called PyQuery. This example code is in `pollenscrape.py`:

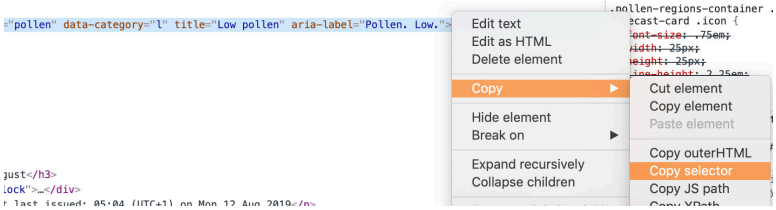
```
from pyquery import PyQuery as pq

data = pq(url="https://www.metoffice.gov.uk/weather/warnings-and-advice/seasonal-advice/pollen-forecast")

pollen = data("#ne table tbody tr td div span")
print(pollen.html())
```

That's... it. Run the code, and it'll print out 'L' (low), 'M' (moderate), 'H' (high), and there may be 'V' (very high), I'm not sure.



The line which starts `pollen = data(…)` is the key one. How to find the right incantation? In Chrome, load the target web page then pull up the Inspector by right-clicking (see left).

From there, you can right-click on the specific bit of the page you want (in this case, the 'L' which indicates low pollen for today), go to Copy, and ask Chrome for the selector path.



In this example, that looks like:

```
#ne > table > tbody > tr > td:nth-child(1) > div > span
```

All I've done, then, is reformat that set of selectors and pass them to the PyQuery object. The `nth-child` bit I just ignored: a guessed based on a bit of knowledge of HTML.

Hopefully, you can see how scraping web pages to extract data is possible. The big catch, of course, is that your code will tend to break if the web page is redesigned.