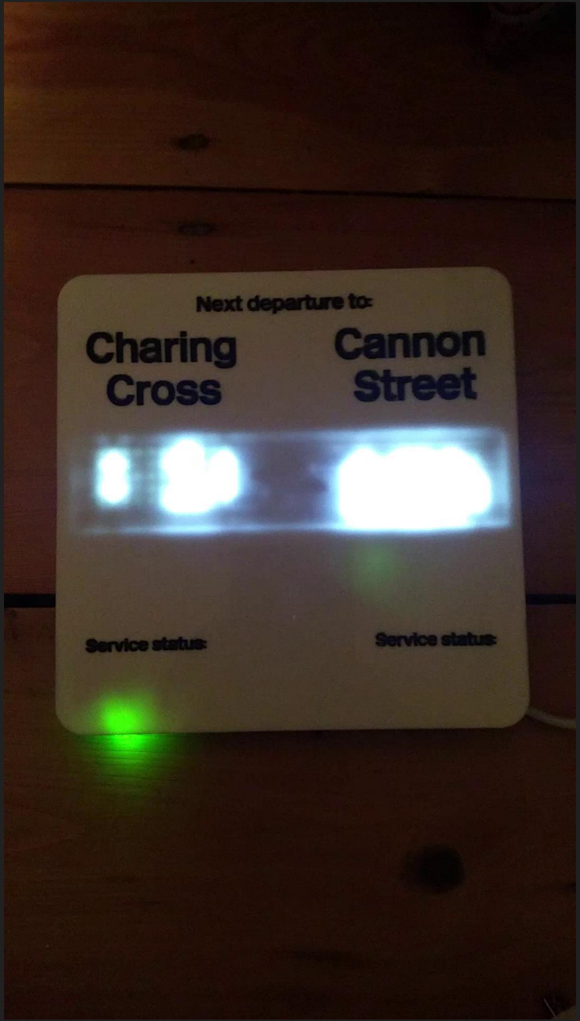
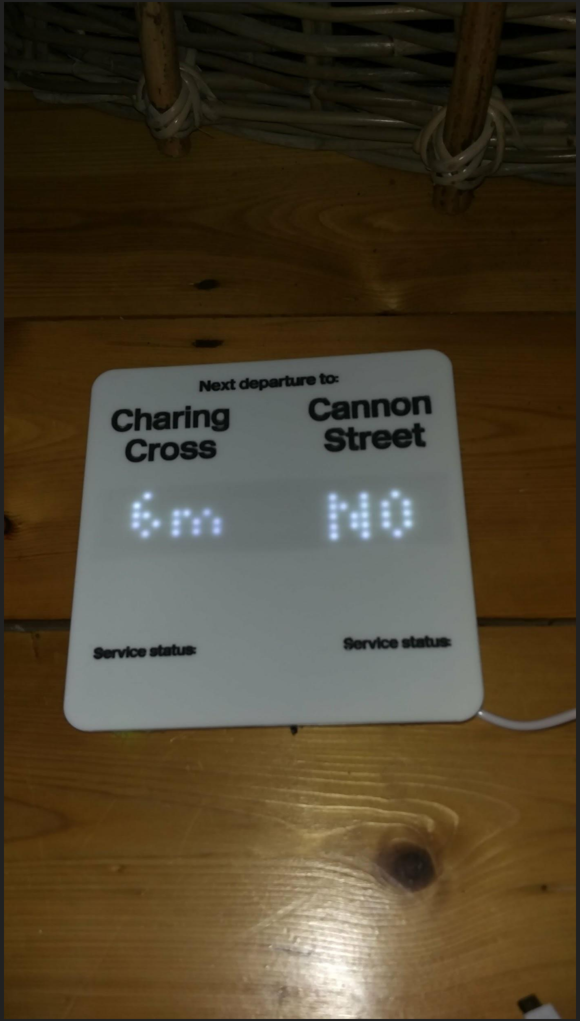
Strikes again

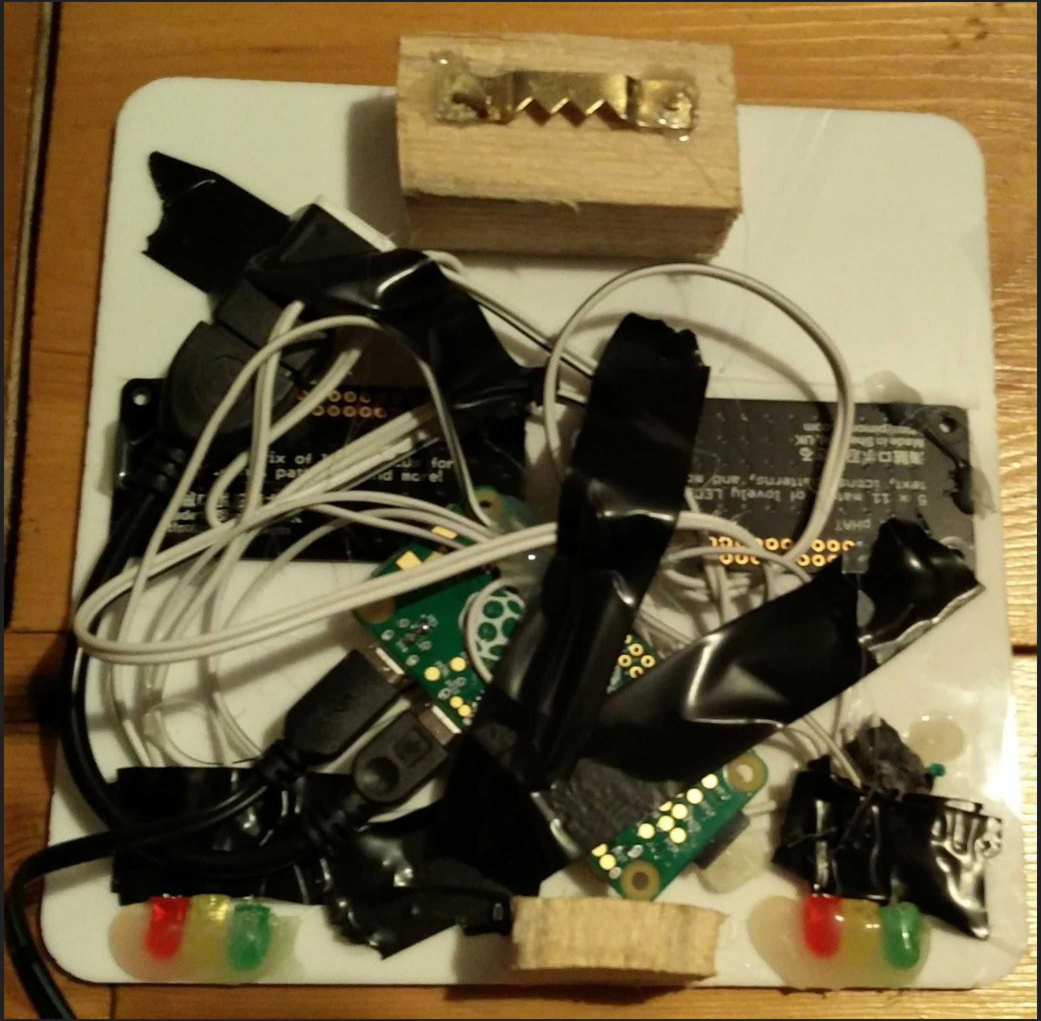
Real time transport data with Raspberry Pi & Python

Next departure to:
Charing Cross
Cannon Street
Service status:
Service status:

Next departure to:
Charing Cross
Cannon Street
Service status:
Service status:

Next departure to:
Charing Cross
Cannon Street
6m
NO
Service status:
Service status:

USB WiFi Dongle - needed to get the train times live

2 x Pimoroni Scroll pHATs - these scroll the train times on them

Red, yellow and green LEDs - these show the status of the train service at the current time



Laser-cut acrylic panel

Raspberry Pi Zero - this gets the data, interprets it and sends it out to the displays. We'll be learning this today!

Lots of hot glue and tape!

Webscraping vs APIs

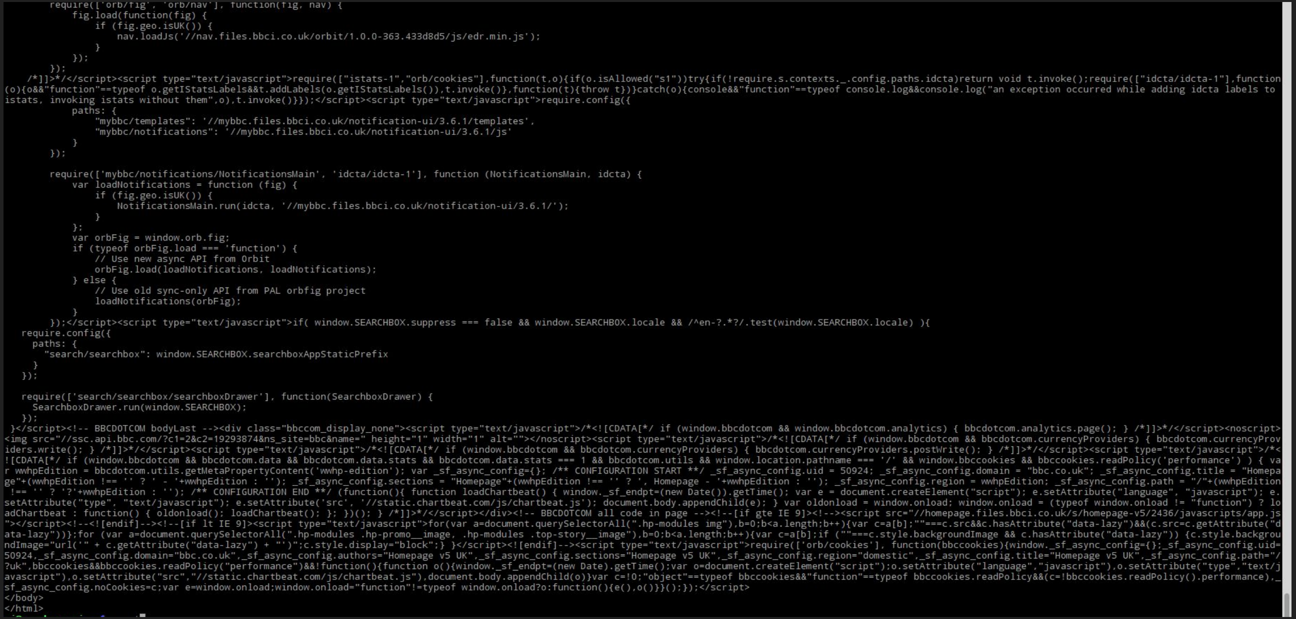
Webscraping: the good

- Methodology is simple
- You get what you want, not what the API wants to give you
- It works for most websites
- If a service doesn’t have an API that’s OK
- No need to sign up for an API

Webscraping: the bad

- If they change the structure of the website, you will need to rewrite your code
- It can be a bit slow and/or require lots of processing power
- It’s hard to interact with websites

Webscraping: the ugly



```
        require(['orb/fig', 'orb/nav'], function(fig, nav) {
            fig.load(function(fig) {
                if (fig.geo.isUK()) {
                    nav.loadJs('//nav.files.bbci.co.uk/orbit/1.0.0-363.433d8d5/js/edr.min.js');
                }
            });
        });
    /*]]>*/</script><script type="text/javascript">require(["istats-1","orb/cookies"],function(t,o){if(o.isAllowed("s1"))try{if(!require.s.contexts._.config.paths.idcta)return void t.invoke();require(["idcta/idcta-1"],function
(o){o&&"function"==typeof o.getIStatsLabels&&t.addLabels(o.getIStatsLabels()),t.invoke()},function(t){throw t})}catch(o){console&&"function"==typeof console.log&&console.log("an exception occurred while adding idcta labels to
istats, invoking istats without them",o),t.invoke()}});</script><script type="text/javascript">require.config({
            paths: {
                "mybbc/templates": '//mybbc.files.bbci.co.uk/notification-ui/3.6.1/templates',
                "mybbc/notifications": '//mybbc.files.bbci.co.uk/notification-ui/3.6.1/js'
            }
        });

        require(['mybbc/notifications/NotificationsMain', 'idcta/idcta-1'], function (NotificationsMain, idcta) {
            var loadNotifications = function (fig) {
                if (fig.geo.isUK()) {
                    NotificationsMain.run(idcta, '//mybbc.files.bbci.co.uk/notification-ui/3.6.1/');
                }
            };
            var orbFig = window.orb.fig;
            if (typeof orbFig.load === 'function') {
                // Use new async API from Orbit
                orbFig.load(loadNotifications, loadNotifications);
            } else {
                // Use old sync-only API from PAL orbfig project
                loadNotifications(orbFig);
            }
        });</script><script type="text/javascript">if( window.SEARCHBOX.suppress === false && window.SEARCHBOX.locale && /^en-?.*?/.test(window.SEARCHBOX.locale) ){
   require.config({
     paths: {
       "search/searchbox": window.SEARCHBOX.searchboxAppStaticPrefix
     }
   });

   require(['search/searchbox/searchboxDrawer'], function(SearchboxDrawer) {
     SearchboxDrawer.run(window.SEARCHBOX);
   });
 }</script><!-- BBCDOTCOM bodyLast --><div class="bbccom_display_none"><script type="text/javascript">/*<![CDATA[*/ if (window.bbcdotcom && window.bbcdotcom.analytics) { bbcdotcom.analytics.page(); } /*]]>*/</script><noscript>
<img src="//ssc.api.bbc.com/?c1=2&c2=19293874&ns_site=bbc&name=" height="1" width="1" alt=""></noscript><script type="text/javascript">/*<![CDATA[*/ if (window.bbcdotcom && bbcdotcom.currencyProviders) { bbcdotcom.currencyProv
iders.write(); } /*]]>*/</script><script type="text/javascript">/*<![CDATA[*/ if (window.bbcdotcom && bbcdotcom.currencyProviders) { bbcdotcom.currencyProviders.postWrite(); } /*]]>*/</script><script type="text/javascript">/*<
![CDATA[*/ if (window.bbcdotcom && bbcdotcom.data && bbcdotcom.data.stats && bbcdotcom.data.stats === 1 && bbcdotcom.utils && window.location.pathname === '/' && window.bbccookies && bbccookies.readPolicy('performance') ) { va
r wwhpEdition = bbcdotcom.utils.getMetaPropertyContent('wwhp-edition'); var _sf_async_config={}; /** CONFIGURATION START **/ _sf_async_config.uid = 50924; _sf_async_config.domain = "bbc.co.uk"; _sf_async_config.title = "Homepa
ge"+(wwhpEdition !== '' ? ' - '+wwhpEdition : ''); _sf_async_config.sections = "Homepage"+(wwhpEdition !== '' ? ', Homepage - '+wwhpEdition : ''); _sf_async_config.region = wwhpEdition; _sf_async_config.path = "/"+(wwhpEdition
 !== '' ? '?'+wwhpEdition : ''); /** CONFIGURATION END **/ (function(){ function loadChartbeat() { window._sf_endpt=(new Date()).getTime(); var e = document.createElement("script"); e.setAttribute("language", "javascript"); e.
setAttribute("type", "text/javascript"); e.setAttribute('src', '//static.chartbeat.com/js/chartbeat.js'); document.body.appendChild(e); } var oldonload = window.onload; window.onload = (typeof window.onload != "function") ? lo
adChartbeat : function() { oldonload(); loadChartbeat(); }; })(); } /*]]>*/</script></div><!-- BBCDOTCOM all code in page --><!--[if gte IE 9]><!--><script src="//homepage.files.bbci.co.uk/s/homepage-v5/2436/javascripts/app.js
"></script><!--<![endif]--><!--[if lt IE 9]><script type="text/javascript">for(var a=document.querySelectorAll(".hp-modules img"),b=0;b<a.length;b++){var c=a[b];""===c.src&&c.hasAttribute("data-lazy")&&(c.src=c.getAttribute("d
ata-lazy"))};for (var a=document.querySelectorAll(".hp-modules .hp-promo__image, .hp-modules .top-story__image"),b=0;b<a.length;b++){var c=a[b];if (""===c.style.backgroundImage && c.hasAttribute("data-lazy")) {c.style.backgrou
ndImage="url('" + c.getAttribute("data-lazy") + "')";c.style.display="block";} }</script><![endif]--><script type="text/javascript">require(['orb/cookies'], function(bbccookies){window._sf_async_config={};_sf_async_config.uid=
50924,_sf_async_config.domain="bbc.co.uk",_sf_async_config.authors="Homepage v5 UK",_sf_async_config.sections="Homepage v5 UK",_sf_async_config.region="domestic",_sf_async_config.title="Homepage v5 UK",_sf_async_config.path="/
?uk",bbccookies&&bbccookies.readPolicy("performance")&&!function(){function o(){window._sf_endpt=(new Date).getTime();var o=document.createElement("script");o.setAttribute("language","javascript"),o.setAttribute("type","text/j
avascript"),o.setAttribute("src","//static.chartbeat.com/js/chartbeat.js"),document.body.appendChild(o)}var c=!0;"object"==typeof bbccookies&&"function"==typeof bbccookies.readPolicy&&(c=!bbccookies.readPolicy().performance),_
sf_async_config.noCookies=c;var e=window.onload;window.onload="function"!=typeof window.onload?o:function(){e(),o()}}();});</script>
</body>
</html>
```

x10!

- Look at all this data! This is what you’ll need to wade through to get the single piece of info you want.

Let’s try it!

Click the terminal icon in the bar at the top:

In the window that appears, type this line:

```
curl http://www.bbc.co.uk
```

Hello, world!

Python

Click the icon, and then go to programming > Python 2 (IDLE).

It’s important you use Python 2 here, as some of the code won’t work in Python 3.

Press ctrl+N to open up a new Python file. Let’s get started! Type the following:

```
import urllib2
page = urllib2.urlopen(‘http://www.bbc.co.uk’)
print(page)
```

And run the code by hitting F5. Nonsense! Why are we getting this?

Soup

To install the BeautifulSoup library, go back to the terminal and run:

```
sudo pip install BeautifulSoup4
```

Note the capitals there. If you don’t use them, it won’t install the right package! (package names are case-sensitive)

Close python, then open it again for the files to take effect.

Let’s scrape

Add the following lines to your code:

```
from bs4 import BeautifulSoup
```

At the top, and then after the other code:

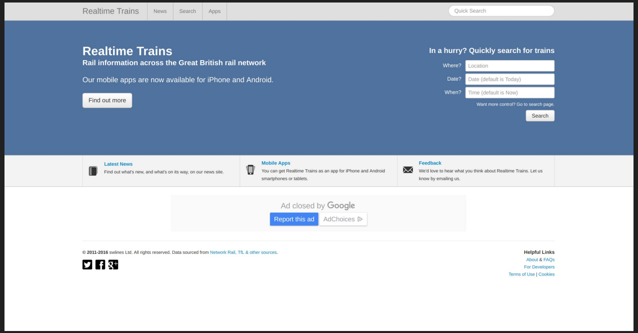
```
scraped = BeautifulSoup(page)
print(scraped)
```

and run!

Train times

Open up the web browser (the icon next to the terminal one from earlier)

Navigate to http://www.realtimetrains.co.uk



Realtime Trains
Rail information across the Great British rail network

Our mobile apps are now available for iPhone and Android.

In a hurry? Quickly search for trains

Want more control? Go to search page.

Latest News
Find out what's new, and what's on its way, on our news site.

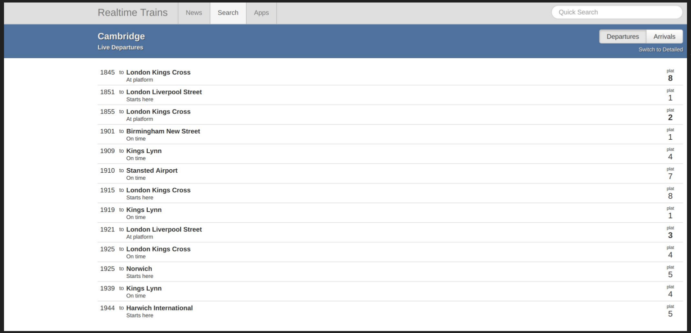
Mobile Apps
You can get Realtime Trains as an app for iPhone and Android smartphones or tablets.

Feedback
We'd love to hear what you think about Realtime Trains. Let us know by emailing us.

Ad closed by Google

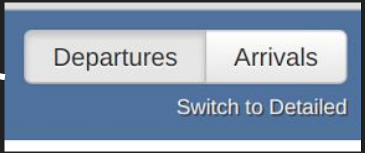
© 2011-2016 swlines Ltd. All rights reserved. Data sourced from Network Rail, TfL & other sources.

Helpful Links
About & FAQs
For Developers
Terms of Use | Cookies

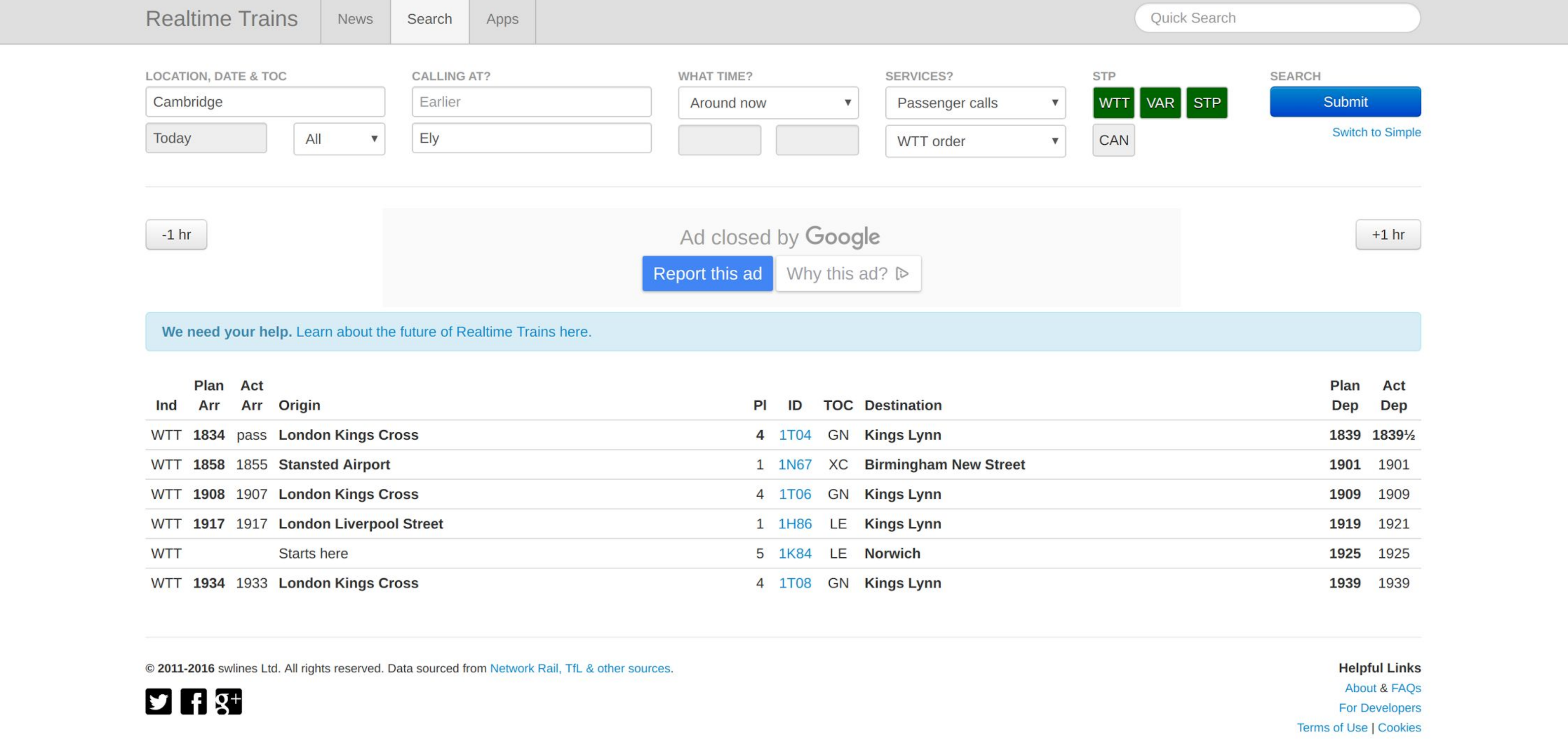


Cambridge
Live Departures

1845	to	**London Kings Cross** At platform	plat **8**
1851	to	**London Liverpool Street** Starts here	plat 1
1855	to	**London Kings Cross** At platform	plat **2**
1901	to	**Birmingham New Street** On time	plat 1
1909	to	**Kings Lynn** On time	plat 4
1910	to	**Stansted Airport** On time	plat 7
1915	to	**London Kings Cross** Starts here	plat 8
1919	to	**Kings Lynn** On time	plat 1
1921	to	**London Liverpool Street** At platform	plat **3**
1925	to	**London Kings Cross** On time	plat 4
1925	to	**Norwich** Starts here	plat 5
1939	to	**Kings Lynn** On time	plat 4
1944	to	**Harwich International** Starts here	plat 5



Enter your data, press enter, then click switch to detailed



LOCATION, DATE & TOC: Cambridge, Today, All
CALLING AT?: Earlier, Ely
WHAT TIME?: Around now
SERVICES?: Passenger calls, WTT order
STP: WTT VAR STP CAN
SEARCH: Submit
Switch to Simple

We need your help. Learn about the future of Realtime Trains here.

Ind	Plan Arr	Act Arr	Origin	Pl	ID	TOC	Destination	Plan Dep	Act Dep
WTT	1834	pass	London Kings Cross	4	1T04	GN	Kings Lynn	1839	1839½
WTT	1858	1855	Stansted Airport	1	1N67	XC	Birmingham New Street	1901	1901
WTT	1908	1907	London Kings Cross	4	1T06	GN	Kings Lynn	1909	1909
WTT	1917	1917	London Liverpool Street	1	1H86	LE	Kings Lynn	1919	1921
WTT			Starts here	5	1K84	LE	Norwich	1925	1925
WTT	1934	1933	London Kings Cross	4	1T08	GN	Kings Lynn	1939	1939

Helpful Links
About & FAQs
For Developers
Terms of Use | Cookies

Copy the web address of this page!

Back to code

Go to your python code again

Change the line:

```
page = urllib2.urlopen(‘http://www.bbc.co.uk’)
```

to:

```
page = urllib2.urlopen(‘paste_web_address_here’)
```

Getting the format all nice

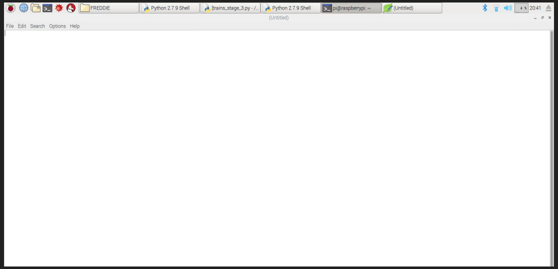
That’s what we’re going to use to get our data. But, there are a few problems. There are quite a lot of returns and spaces - we don’t need them and it just makes stuff harder. So add the following lines before your `print` statement:

```
scraped = str(scraped)
scraped = "".join(scraped.split())
```

The first line makes doubly-sure it’s a *string*, and the second one gets rid of all *whitespace*.

Run it again!

Interpreting our data



Copy all of that text it’s spat out.

Now, open a text editor by going to raspberry>Accessories>Text Editor

Scroll through - can you see any times? Look for a time the webpage showed us (so 0900 for example).

Find an identifier - something which appears shortly before the time, but nowhere else. Use ctrl + F and crtl + G to find or find again a value in the text. Trial and error is your friend!

Make sure to use the *actual* time here, rather than the *timetabled* time since we want to know when the train will really arrive!

Finding the time

For this example, we’re using ‘realtimeactual’ as the phrase before the time.

Add the following line to your code:

```
index = scraped.find(‘your_identifier’)
```

That will give us the position of the first character of the phrase. Go back to the text editor, and count how many characters after this the train’s time actually begins. For *realtimeactual*, it’s 16. Add the line:

```
index = index + your_number_goes_here
```

Extracting the time

So now we know where the time is. Great! Now let’s get the actual time itself.

Add the line:

```
traintime = scraped[index:(index+4)]
```

- The square brackets are used to cut out a certain bit of the string.
- The thing before the : is the start number, and the thing after is the end
- As we know that the time is 4 digits long, we can just add 4 to the index to find the end.

Change the print statement to `print(traintime)`

Try it!

Does it work - does the code give you a number which is a time in the near future (or recent past)?

If you get something a bit nonsensy, like ‘>090’, then that probably means your index is a bit wrong - tinker with it and see.

Finding the time (again)

Now we know when the train’s coming. But currently, we don’t know what the tine is now. Let’s fix that! At the top of the code, add the line:

```
import time
```

Then, before the `print` statements, add the line:

```
currenttime = time.strftime(‘%H%M’, time.gmtime())
```

The strftime formats the time nicely, while the time.gmtime() tells it to get GMT.

How many minutes away is the train?

Now it’s just a case of subtracting the train’s time from the time at the moment, to get how far it is away. Maths in Python is simple:

```
timeuntiltrain = int(traintime) - int(currenttime)
```

We will convert the traintime and current time to integers, as by default they are strings. We can’t do maths with strings, as Python sees them as words not numbers!

Simple maths is easy in python - just use +, -, * or / between two numbers or variables to add, subtract, multiply or divide them.

Printing out the final information

Delete the print statement, and replace it with:

```
message = ‘Train arriving in ‘ + timeuntiltrain + ‘ minutes!’

print(message)
```

This will form the message: ‘train arriving in’, then our variable we calculated, then the word ‘minutes’ to finish off the sentence. You could add lots of strings or variables in here and it would just stick them all together.

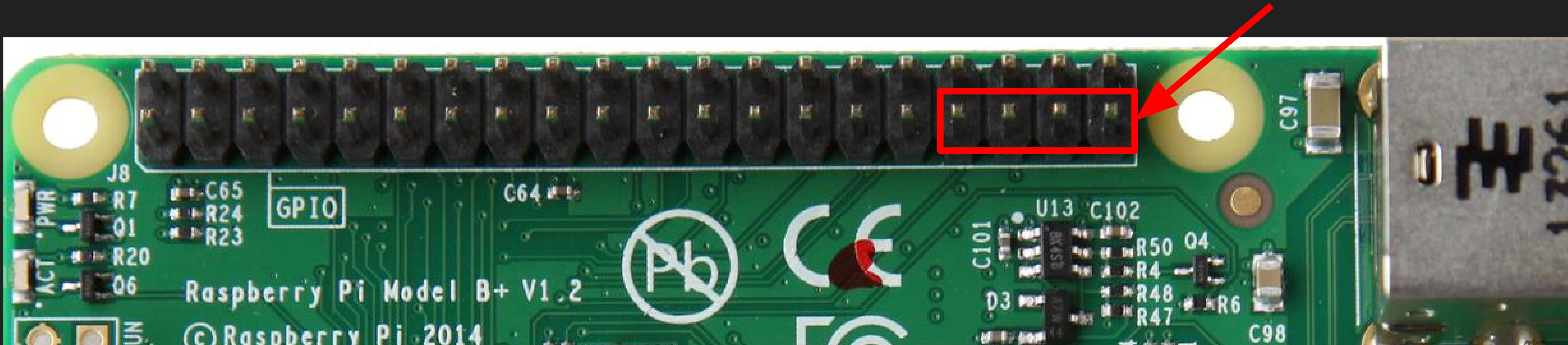
Note the spaces after *in* and before *minutes* - the code won’t automatically add any spaces so we need to put them in text!

Adding hardware output

That’s pretty fun. You know when to get the next train!

But wait, there’s more. Grab a Pi-Stop and fit it to your pi as shown below.

GND Pin goes this end



Light those LEDs

We’ll need a new script here, just to test those LEDs really work. Crtl+N for that

First up, we need to import the lovely gpiozero library (which makes it really easy to control those LEDs - thanks Ben!)

```
import gpiozero
```

Then, setup the LEDs:

```
red = gpiozero.LED(13)
amber = gpiozero.LED(19)
green = gpiozero.LED(26)
```

Turn them on!

To turn the red LED on, you need to run:

```
red.on()
```

And to turn it off:

```
red.off()
```

Have a play!

Adding this to the train times code

Add those setup lines to the top of your train time code. Then add:

```
if timeuntiltrain > 15:
    red.on()
elif traintime >= 5:
    amber.on()
elif traintime <5:
    green.on()
```

Run it and see what happens! When is the next train going to be? Try altering the numbers to see what happens.

Making it update

Before the *‘page =...’* line, add this:

```
while True:
```

Select all the code after, that, and do alt +] to indent all the code (move it in a bit) so it’s inside.

At the end, add this line (make sure it’s indented too):

```
time.sleep(60)
```

So that the code waits a minute between each refresh.

Turn off the LEDs again

If we don’t turn the LEDs off, they will stay on and eventually they’ll all just become on and it won’t tell us everything.

So in the same way we did earlier, add these lines just after the *while True* (so they turn off at the start)

```
red.off()
amber.off()
green.off()
```

Going further

If you want to add to this, why not try:

- Finding the scheduled time and the actual time, and working out if the train’s running late
- Finding the trains for a specific date and/or time
- Finding only trains that start at your station (because these will be nice and empty)
- Adding different messages/remarks
- Adding animations to the LEDs

Going even further (at home)

If you want to take this forward at home, why not try:

- Interfacing with different hardware (scrolling messages, multicolour LEDs… your imagination is the limit!)
- Using different websites - could you make a news-watcher, weather-forecast monitor, or traffic indicator. Just find a website that will give you the details!
- Try using the APIs - TfL has a really big API covering all its services, and there are a few tutorials using it. APIs can be really powerful when you have the right tools!