Advanced Python - Query a simple (weather) API

Query a simple (weather) API

Introduction

APIs are very useful, it's a way to share data and informations between applications as well as between a backend and a frontend.

Let's play with the weather API made available by OpenWeatherMap, it will allow us to query the weather for a given city and use the data in our web app.

To query an API we are going to use the Requests HTTP library. Why? Mostly because this library is easy to use and has great documentation.

Let's start

Install the lib

```
pip install requests
```

Code

You can create a new file, say `weather.py` and copy these lines in:

```
import requests

endpoint = 'http://api.openweathermap.org/data/2.5/weather'
payload = { 'q': 'London,UK', 'appid': '44db6a862fba0b067b1930da0d769e98' }

response = requests.get(endpoint, params=payload)
```

```
print response.url
print response.status_code
print response.headers['content-type']
```

So, what will happen?

1. We `import` the library, meaning we allow `requests` to be used in our script.
2. We define 2 variables:
    - `endpoint` that contains the API URL we need to fetch
    - `payload` that contains parameters we pass to the API. *(Note, `appid` is a way for the API owners to identify us as a "customer" of their API, very often while using an API you will need to register and use private ids to identify yourself. This way, API owners can identify you and monitor your usage.)*
3. Using `requests` we query the API and store the response in the variable `response`.
4. We print some of the information we get from the response:
    - `response.url`: URL actually called by `requests`. You can copy it and paste it in your browser, you will get the same response that `requests` got back.
    - `response.status_code`: HTTP status code of the response. In HTTP, when a request succeeds, we expect a `200 Ok` *(More on Wikipedia)*
    - `response.headers['content-type']`: Content type of the response defined in the HTTP header.

Let's run it

To run the previous script, simply save it under `weather.py` and run:

```
python weather.py
```

The output should look something like, corresponding to our print statement:

```
http://api.openweathermap.org/data/2.5/weather?q=London
%2CUK&appid=44db6a862fba0b067b1930da0d769e98
200
```

```
application/json; charset=utf-8
```

Congrats!

So now you are surely wondering how to access the data from the response, from Requests' doc you can access the response content using `response.text`. Try it: add a new print statement at the end of the `weather.py` file:

```
print response.text
```

And run the script again.

You might now wonder what are all these `u` before the 'strings'. These are unicode strings, which can include many characters which are not possible in normal strings such as characters from other alphabets (Ж, □, پش) and emoji (, ,).Unicode encoding of strings is often used in web applications as they may have to handle text input from users all around the world.

Use this data!

Ok so we get the API response, but we would like to print a nice sentence about the weather, not the raw output of the API. Let's do it.

First, we are going to convert the response in JSON and store it in a variable:

```
data = response.json()
```

Note: do not hesitate to print a variable when you need it, it's a good way to see what the data stored inside the variable look like. (Tip: `print data`)

To access a JSON field we can use this notation: `data['main']['temp']`. Try it:

```
print data['main']['temp']
```

Does that look ok?

Don't you think the temperature value `temp` is weird? According to OpenWeatherMap API doc the default unit for temperature is Kelvin. Let's change this by passing a new parameter to our URL: `units=metric` or `units=imperial`, update the `payload` variable:

```
payload = { 'q': 'London,UK', 'units': 'metric', 'appid': '44db6a862fba0b067b1930da0d769e98' }
```

Is it better?

Let's print a nice output for our script now:

```
print u"It's {}°C in {}, and the sky is {}".format(data['main']['temp'], data['name'], data['weather'][0]['main'])
```

And... we are done! Congrats

Bonus / exercise

1. Wouldn't it be cool if when running the script, the user could enter the city she/he is interested in? It could look something like:

```
$ python weather.py
What city are you interested in?
> London,UK
It's 12°C in London, and the sky is clear
```