Calling an API

Go here: https://rickandmortyapi.com/documentation/#rest

For each, make a request to the API to...

1. Get the first set of characters (page 1)
2. Get the second set of characters, using a filter query
3. Get one character using a path
4. Get multiple characters using a path
5. Find the character with the name Rick, using a filter query
6. Find the character with the name `Baby Legs` and is still alive, using a filter query

Create an API in python

Part 1 Setting up REST API

1. Create a new folder with a new virtual environment in it and activate it
2. Install Flask `pip install Flask`. If this doesn't work for you there is a guide on installing Flask here (https://flask.palletsprojects.com/en/1.1.x/installation/#installation)
3. Make a `hello world` API by flowing the Flask quick start guide (https://flask.palletsprojects.com/en/1.1.x/quickstart/)
4. Call your API
5. Play around with your API, make it return something different

Part 2 Implementing REST API services for a website

This part of the exercise will get you to build the backend serveries (REST APIs) for a cafe website. The website will work as intended if you build the correct APIs. If your API implementation deviates from what the front end is expecting then an error message will be returned.

Opening the website

1. Save the `rest-ful-cafe.html` file somewhere in your computer
2. Open the HTML file in your browser by double clicking it. You should see a very basic website
3. Make sure your API server is running on http://127.0.0.1:5000/ and try out the website. Note that none of the APIs will work for two reasons, you have not built them yet and your CORS policy is not set up

Setting up a CORS policy

1. If you open your browser's developer tools you will see an error message to do with CORS like this:

```
Access to fetch at 'http://127.0.0.1:5000/order' from origin 'null'
 has been blocked by CORS policy: No 'Access-Control-Allow-Origin'
 header is present on the requested resource. If an opaque response
 serves your needs, set the request's mode to 'no-cors' to fetch the
 resource with CORS disabled.
```

CORS (Cross-origin resource sharing) allows you to restrict what websites an API can be used on. This error is caused by not having a CORS policy set up.

2. To set up a CORS policy we must return a header called `Access-Control-Allow-Origin` to the browser. This will tell your browser that its safe to communicate with it. You can do this by pasting the code below into your API module.

```
 # After a request is made, add a header called "Access-Control-Allow-Origin" to the response.
# This will tell the browser that the API can supply data to any URL.
@app.after_request
def add_header(response):
    response.headers["Access-Control-Allow-Origin"] = "*"
    return response
```

3. After making any changes to your API code to make them take effect you need to restart your server. To restart your server, stop your server with `Ctrl + C` and start it again with `flask run` or `Up-Arrow + Enter`. Then check the CORS error has disappeared

Implementing the API

1. GET Product

Start by implementing the `GET /product` API. This API should return a list of products that looks like

```
[
  {
    "id": 1,
    "name": "Tea",
    "price": 1.2,
    "imageUrl":
    "https://www.twinings.co.uk/app_/responsive/TwiningsUKI/media/content/About-Tea/shutterstock_126275183.jpg?w=1260"
  },
  {
    "id": 2,
    "name": "Coffee",
    "price": 3.5,
    "imageUrl":
    "https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTHm0-W876ppWBpCak_UzC9eRz8X5CeI1KPYw&usqp=CAU"
  }
]
```

Note you might have to JSON encode the python object to turn it into a string for the API.

2. Get Product Ingredients

Implement the `GET /product/product_id/ingredients` API. This should return a response that looks like this:

```
[
  { "name": "Rice flower", "content": 20, "isCommonAllergen": True },
  { "name": "Fish", "content": 50, "isCommonAllergen": False },
  { "name": "Egg", "content": 30, "isCommonAllergen": True }
]
```

3. Create Order

Implement the `POST /order` API. This should return an integer orderId like:

```
  { "orderId": "123" }
```

4. Challenge calculate the total price of the order

When an order is created the `POST /order` API receives a request which contains a JSON list of objects containing the `productId` and the `quantity`.

Use these to return a `totalPrice` field to the website like:

```
{
    "orderId": 123,
    "totalPrice": 14.5
}
```

Hint: If you want a hint on getting the request body inside the API then keep reading... You can get the raw request body like this

```
from flask import request
request_body = request.data
```

See docs https://flask.palletsprojects.com/en/1.1.x/api/#flask.Request.data