Task 2

Learn About Web APIs

# What are APIs?

[APIs](https://www.ibm.com/think/topics/api) are just a set of rules and definitions that allow different pieces of software to communicate with each other. Think of it like the menu in a restaurant, it tells you what you can order, and how much it costs. But it doesn’t make the order for you, it just tells you how to do it.

# An example

This task will specifically focus on [web apis](https://en.wikipedia.org/wiki/Web_API). We will focus on this API for this task [PokéAPI](https://pokeapi.co/).

Open the documentation, it lists a bunch of URLs, these are the endpoints.

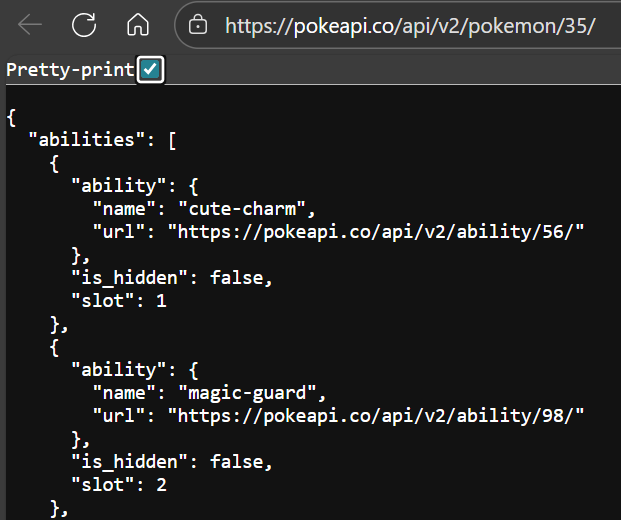
Example resource URL: <https://pokeapi.co/api/v2/pokemon/35/>

Sample output: (the output might be present in some other order)

| id:35  name:"clefairy"  base\_experience:113  height:6  is\_default:true  order:56  weight:75  ..  ..  ..  .. |
| --- |

So what happens if you visit this URL?

(It may look different on a different browser, this is edge)



Well, the URL is not a website, instead it’s some data in a format called [JSON](https://stackoverflow.blog/2022/06/02/a-beginners-guide-to-json-the-data-format-for-the-internet/). This data is the response.

To see what this is, we read the documentation about the endpoint.

In the documentation search for the endpoint: https://pokeapi.co/api/v2/pokemon/{id or name}/

So, we accessed the endpoint by going to that URL, and it gave us a response of the stats, abilities of the Pokémon searched. It gave us the name, some sort of id and a lot of other fields that can be used as we please.

Take a look at the other endpoints of the API, enter the URLs in the browser and see what you get, modify the URLs and see what happens. Think of the endpoints as the different types of information that the API can give you.

What if we need to pass some parameters to the API? We just modify the URL, look at the other endpoints for more details.

# How do we use APIs?

Now we have been accessing the API through the browser, but this is highly unusual. Normally APIs are accessed using programs.

For example: for python you will need the [requests](https://realpython.com/python-requests/) module.

**API PARAMETERS**

API parameters are extra pieces of information you send along with an API request. They help define or change the data you’ll be working with.

There are 2 types of parameters that go directly into the URL of the API:

1. path parameters
2. query parameters

Path parameters are written in the form {url}/param. Here, "param" is a placeholder for some value (for example, a product id or username). The API will return results based on this value. Path parameters are usually used to point to a specific resource, like a single product, user, or item.

Query parameters are written in the form {url}?param=value. Here "param" is the name of the parameter, and "value" is its value. Query parameters are often used when you want to filter, sort, or paginate data. For example, you might use ?page=2 to get results from the second page. You can add multiple parameters by joining them with an &. The ‘?’ marks the start of the query parameters.

Now coming to the actual task,

**Write programs to do the following (separate program for each):**

**TASK A: Pokémon Data Extraction and JSON Creation**

1. Input Data

* You will be provided with a file named pokemons.txt.
* This file contains a list of Pokémon names or IDs.

2. Retrieve Pokémon Information

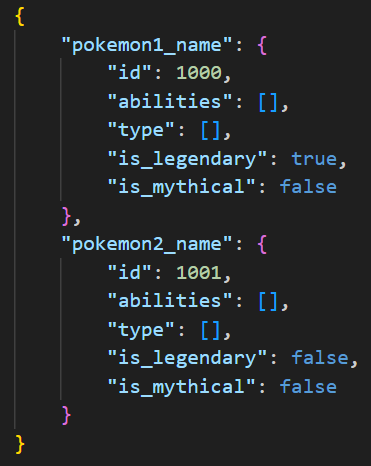
* For each Pokémon listed in pokemons.txt, use PokéAPI’s available endpoints to retrieve the complete data about that Pokémon.
* The API will return a large amount of data, but only a small portion is relevant for this task.

3. Filter the Data

* From the API’s response, extract and retain only the following attributes:
  + Abilities
  + Types
  + Is Legendary (whether the Pokémon is legendary)
  + Is Mythical (whether the Pokémon is mythical)

4. Reformat into JSON

* Using the filtered data, create a structured JSON file with the following format:



* Each Pokémon from the pokemons.txt file should appear as a key in the JSON object.
* The Pokémon’s data should be structured exactly as shown above.

5. Output File

* Take filename as the first command line argument of the program (If not present, create it).

**TASK B: Type Chart Generator**

1. Access Pokémon Type Data

* Use PokéAPI’s `types` endpoint to retrieve information about all Pokémon types:  
  “https://pokeapi.co/api/v2/type/{id or name}/”
* From this data, extract how each type interacts with others in terms of damage multipliers (immunities, weaknesses, and resistances).

2. Damage Multiplier Representation

* Construct a 2D array (matrix) that represents the effectiveness of each type against every other type.
* In this matrix:
  + Rows represent the defending type (the type receiving damage).
  + Columns represent the attacking type (the type dealing damage).
  + Each cell contains a damage multiplier (e.g., 0 for immunity, 0.5 for resistance, 2 for weakness, 1 for neutral).

3. Example Explanation

* Suppose the row is Grass type:
  + The entries across that row show how much damage Grass takes from Fire, Water, Electric, Flying, etc.
* Suppose the column is Fire type:
  + The entries down that column show how much damage Fire deals to Grass, Rock, Steel, Bug, etc.

4. Final Output

* The completed matrix should cover all types and clearly represent type effectiveness in both directions (attack vs. defence).
* Ensure that the numbers correctly reflect Pokémon mechanics (immunity, resistance, weakness, neutral).
  1. Imagine your program is building something like the chart shown below (the classic Pokémon Type Effectiveness chart). You do not need to add labels like *Attacker*, *Defender*, or color codes. Your job is only to build the matrix that contains the same information.



* 1. You are required to host your own local HTTP server that serves this 2d array.  
     Let’s say the server is hosted on “localhost:8000”. Let’s denote this by {{base\_url}}

You are expected to read the data from this matrix and display it on various endpoints specified below:

1. Endpoint: {{base\_url}}?attacker={attacker\_type}&defender={defender\_type}

Description**:** Returns the damage multiplier when the given attacker\_type attacks the given defender\_type

Example: “GET localhost:8000?attacker=Fire&defender=Grass”

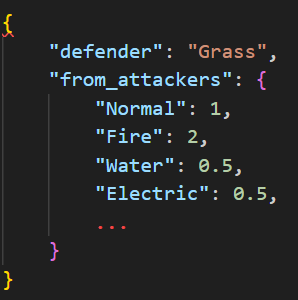
Response: { "attacker": "Fire", "defender": "Grass", "multiplier": 2 }

1. Endpoint: {{base\_url}}?defender={defender\_type}

Description**:** Returns the entire row of the matrix, i.e. how much damage the defender\_type takes from all attacker types.

Example: “GET localhost:8000?defender=Grass”

Response:



1. Endpoint: {{base\_url}}?attacker={attacker\_type}

Description**:** Returns the entire column of the matrix, i.e. how much damage the attacker\_type deals to all defender types.

Example: “GET localhost:8000?attacker=Fire”

Response:



**Brownie points:**1. User Input Pokémon

* Allow the user to enter the name of any Pokémon (for testing, you may hardcode one Pokémon of your choice).
* Use PokéAPI to extract the Pokémon’s type(s).
* You should use the locally hosted server that you have set up to retrieve the damage multipliers.

2. Single-Type Pokémon

* If the Pokémon has only one type, display a list of types it is:
  + 2× Weak to
  + Immune to

3. Dual-Type Pokémon

* If the Pokémon has two types, you must calculate the combined weaknesses and immunities.
* This is done by multiplying the damage multipliers of both types at each column index in the matrix.

Example Calculation

* Consider a Grass/Bug Pokémon.
  + Grass is 2× weak to Fire.
  + Bug is also 2× weak to Fire.
  + Combined: 2 × 2 = 4×, meaning 4× weakness to Fire.
* If one type is weak (2×) and the other resists (0.5×):
  + 2 × 0.5 = 1×, meaning neutral damage.
* If one type is immune (0×):
  + 0 × anything = 0, meaning complete immunity.

4. Output Format (Important)

* When displaying the result for the chosen Pokémon, clearly separate:
  + 2× Weaknesses (list of types that deal 2× damage)
  + 4× Weaknesses (list of types that deal 4× damage)
  + Immunities (list of types that deal 0× damage)

**NOTE:**

* You can go to `http://localhost:port` where port is the port number your server is running on in a browser to test your code or use command line tools like `curl` if you're on linux or `Invoke-WebRequest` if you're on Windows… Another great tool is [HTTPie](https://httpie.io/) which is cross-platform and really user friendly. For web based solutions there is of course [Postman](https://www.postman.com/).
* **Submissions need to be made on Github latest by 4th September EOD.**
* A Google Form will be shared later for submitting the links to the repositories where you have uploaded your code.