Assignment 5: An Object-oriented Pokedex

Introduction

Welcome to the last and final assignment for COMP3522 Object Oriented Programming. I can't believe it's almost the end of the semester!

In the last 12 weeks you all have been exposed to almost every facet of writing complex object oriented programs. You have gone from SOLID Principles, Iterators, generators, to complex concepts such as Multiple Inheritance, Decorators, Factories and Threads.

In this assignment you will be gaining first hand experience writing asynchronous code that creates HTTP client sessions and queries an endpoint with GET HTTP Requests. To this end we will be implementing `asyncio` tasks and using the `aiohttp` package. To this end, we will be creating our very own prototype of Professor Oaks `Pokedex`. For those of you who are not into Pokémon, a Pokedex is a device that contains all known information about the Pokémon universe. It is a crucial tool in every Pokémon trainers toolbox.

We will be querying the PokéAPI. This is a **REST**ful API. REST stands for **RE**presentational **S**tate **T**ransfer. This is a web application that provides the client (your application) with a ***Representation*** of the ***State*** of a requested resource.

💡 For more information about RESTful API, check out this link: https://medium.com/extend/what-is-rest-a-simple-explanation-for-beginners-part-1-introduction-b4a072f8740f

So back to PokéAPI. This is a RESTful API that can provide information about the pokemon universe. It has different endpoints (URL) that can return data about pokemon, their attack moves, abilities, stats, evolutions etc. The PokéAPI can be found at https://pokeapi.co/.

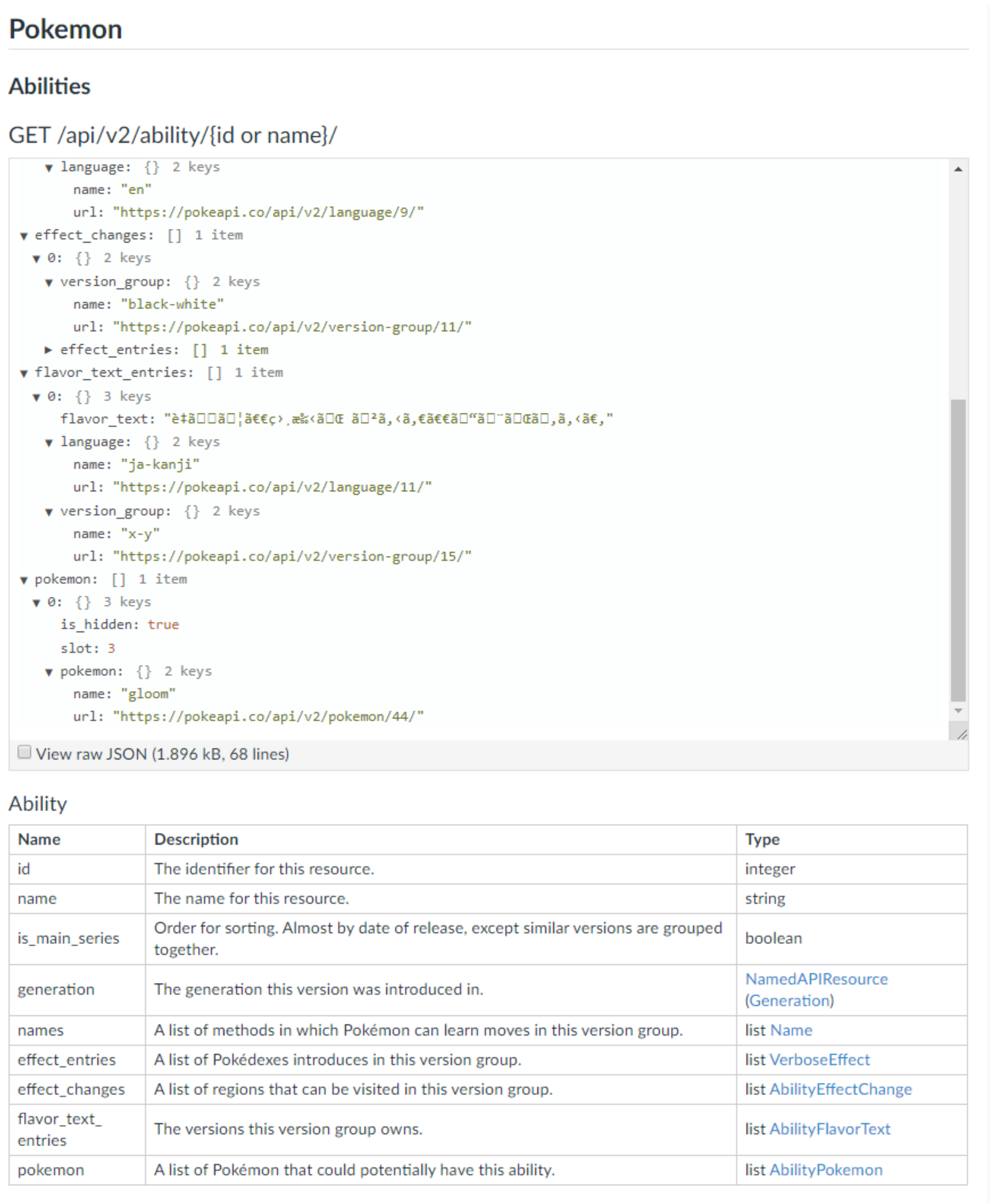
Submission Requirements

1. This assignment is due before **11:59 PM on Friday 6th December 2019. I will not check out any commits past this date and time.**
2. Submit your .py files and UML diagrams in the Assignment 5 folder of your GitHub repository.
3. This is an individual assignment. I strongly encourage you to share ideas and concepts, but sharing code or submitting someone else's work is not allowed.
4. Include a Readme file that describes how your application works. List any errors or edge cases that are not handled/accounted for. Additionally list any behavior or requirement that is not implemented (if any are left out) as well as any peculiar behaviors or extra functionality that I should keep an eye out for.

Reading the Documentation

During the course of this assignment you will find yourself regularly accessing it's documentation. These can be found at https://pokeapi.co/docs/v2.html/ Before we look at the assignment requirements, let's take a look at what the documentation looks like.

Below we have a screenshot of what the documentation for the `Abilities` endpoint looks like. (https://pokeapi.co/docs/v2.html/#abilities)



Pokemon

Abilities

GET /api/v2/ability/{id or name}/

```
    ▼ language: {} 2 keys
        name: "en"
        url: "https://pokeapi.co/api/v2/language/9/"
▼ effect_changes: [] 1 item
  ▼ 0: {} 2 keys
    ▼ version_group: {} 2 keys
        name: "black-white"
        url: "https://pokeapi.co/api/v2/version-group/11/"
    ▶ effect_entries: [] 1 item
▼ flavor_text_entries: [] 1 item
  ▼ 0: {} 3 keys
      flavor_text: "è‡ã□□ã□¦ã€€ç›¸æ‰‹ã□Œ ã□²ã‚‹ã‚€ã€€ã□“ã□¨ã□Œã□‚ã‚‹ã€‚"
    ▼ language: {} 2 keys
        name: "ja-kanji"
        url: "https://pokeapi.co/api/v2/language/11/"
    ▼ version_group: {} 2 keys
        name: "x-y"
        url: "https://pokeapi.co/api/v2/version-group/15/"
▼ pokemon: [] 1 item
  ▼ 0: {} 3 keys
      is_hidden: true
      slot: 3
    ▼ pokemon: {} 2 keys
        name: "gloom"
        url: "https://pokeapi.co/api/v2/pokemon/44/"
```

☐ View raw JSON (1.896 kB, 68 lines)

Ability

Name	Description	Type
id	The identifier for this resource.	integer
name	The name for this resource.	string
is_main_series	Order for sorting. Almost by date of release, except similar versions are grouped together.	boolean
generation	The generation this version was introduced in.	NamedAPIResource (Generation)
names	A list of methods in which Pokémon can learn moves in this version group.	list Name
effect_entries	A list of Pokédexes introduces in this version group.	list VerboseEffect
effect_changes	A list of regions that can be visited in this version group.	list AbilityEffectChange
flavor_text_entries	The versions this version group owns.	list AbilityFlavorText
pokemon	A list of Pokémon that could potentially have this ability.	list AbilityPokemon

First off, right at the top this tells us that this is a `GET` request.

`Get` requests are simple requests that just retrieve information from a website or service. Unlike `Post` requests which usually send information and change the state of the data in the web application (eg: Sending a username and password to log in). For this assignment we will only be working with GET requests.

This is followed by the endpoint URL suffix. This is the suffix that will appear in your final URL.

Restful API: `https://pokeapi.co/api/v2/`

Abilities Endpoint: `/api/v2/ability/{id or name}/`

URL Queried: `https://pokeapi.co/api/v2/ability/{id or name}/`

Note: the parameters need to be formatted in the final url string. You can even paste this in the web browser to see the response. Try opening https://pokeapi.co/api/v2/ability/magic-guard/ in your browser now.

> 💡 HINT: Install a browser extension such as `JSON Formatter` to layout the JSON string in a readable manner. this will help A LOT.
>
> For chrome users this is the link to JSON Formatter: https://chrome.google.com/webstore/detail/json-formatter/bcjindcccaagfpapjjmafapmmgkkhgoa?hl=en

This returns a JSON as a response. In our code this will be stored in a `aiohttp.client_reqrep.ClientResponse` object. We can see the structure of the JSON in the table provided below the sample output. The different attributes in the JSON and their types are listed here. These attributes and types are a good look at how objects in your code can be structured. This does NOT have to be a 1-to-1 mapping. You will want to discard attributes that you don't need and perhaps manipulate the information you get to map to the attributes in your class.

Implementation Requirements

Our object oriented Pokedex will be a proof of concept prototype. It will not go into the level of detail that the API models, but will depict some basic information.

The following features and functionality should be part of the Pokedex.

Console Input (and use cases)

Now, the UI artists and UX Designers are hard at work figuring out the look, feel and flow of the application. This is going to take some time. As good developers you know that you can create the core of the application while they are still working on what the front end looks like. For this assignment you will be creating an application that takes it's input from the terminal and can display the output either in the console or print it out to a file as specified by the input.

To achieve this, you need to implement the `argparse` module. You have all encountered this in Lab 08 Chain of Responsibility. You may see it in action by re-visiting the lab and checking out the provided implementation. You should also read the documentation at https://docs.python.org/3.7/howto/argparse.html . You should do this because :

- It explains everything extremely well and has good examples.
- You will need to refer to the documentation to learn how to use various tools and techniques. This is a skill you need to develop irrespective of the language being used.

You should be able to use the following syntax when entering data into the terminal:

```
python3 pokedex.py {"pokemon" | "ability" | "move"} {"filename.txt" | "name or id'} [--expanded] [--output "filename.txt"]
```

Let's break this down:

python3 pokedex.py

Depending on your system, you will want to use `python` or `python3` followed by the name of the python file to execute. In this case, your driver class and main method will be situated in the `pokedex.py` module.

{"pokemon" | "ability" | "move"}

The application **has to** be opened in one of 3 specific modes, Pokemon, Ability, or Move. `mode` is a positional argument, that means it has to be provided. This is the first of the positional arguments.

(Curly braces denote mutually exclusive arguments. That is only one of the parameter sets must be selected).

- In the `pokemon` mode, the input will be an id or the name of a pokemon. The pokedex will query pokemon information.
- In the `ability` mode, the input will be an id or the name of a ability. These are certain effects that pokemon can enable. The pokedex will query the ability information.
- In the `move` mode, the input will be an id or the name of a pokemon move. These are the attacks and actions pokemon can take. The pokedex will query the move information.

{"filename.txt" | "name or id'}

As input, the application can take in either a file name (Text file), or a name/id. The file name must end with a `.txt` extension. The id must be a digit and the name a string. This is the second positional argument.

(Curly braces denote mutually exclusive arguments. That is only one of the parameter sets must be selected).

Filenames allow the user to do bulk queries. For example, if executing the app in `pokemon` mode, then a text file could be a combination of pokemon names and id's, for example:

```
pikachu
Raichu
132
Alakhazam
pidgey
```

Use `asyncio.gather()` to run multiple queries concurrently.

[--expanded]

The `-- expanded` flag is an optional flag. (Square brackets denote optional parameters).

When this flag is provided, certain attributes are expanded, that is the pokedex will do sub-queries to get more information about a particular attribute. If this flag is not provided, the app will not get the extra information and just print what's provided.

For this prototype, only `pokemon` queries support the expanded mode. Refer to the pokemon query details below in the document for more information.

[--output "filename.txt"]

The `--output` flag is optional. If provided, a filename (with a `.txt` extension) must also be provided. and the query result should be printed to the specified text file. If this flag is not provided, then print the result to the console. Be sure to print the file in a nicely formatted, readable manner.

Get Pokémon Data

The user should be able to provide the name or id of a Pokémon and query its information. You want to create a `Pokemon` Class that can store (and ultimately be used to report) the following data:

Pokemon

Name	Type	Exapandable	Notes

Name	Type	Expandable	Notes
Name	string	☐	
ID	int	☐	
Height	int	☐	If printing height, be sure to print it as decimetres. For example, a height value of 100 will be printed as "100 decimetres"
Weight	int	☐	If printing weight, be sure to print it as hectograms. For example, a weight value of 100 will be printed as "100 hectograms"
Generation	string	☐	
Stats	PokemonStat list	☑	A pokemon query by default returns the name of the stat and it's base value alongside a URL. If the app is run with the `--expanded` flag, then the URL that is provided must also be queried to provide more detailed stat information (see the entry for the stats query below). If the app is not run with the `--expanded` flag then only store the name and the base value.
Types	list string	☐	
Ablities	PokemonAbility list	☑	A pokemon query by default returns a list of ability names and corresponding URL's. If the app is run with the `--expanded` flag, then the URL that is provided must also be queried to provide more detailed ability information (see the entry for the ability query below). If the app is not run with the `--expanded` flag then only store the ability name.
Moves	PokemonMove list	☑	A pokemon query by default returns a list of move names, the level at which the move is learnt and corresponding URL's. If the app is run with the `--expanded` flag, then the URL that is provided must also be queried to provide more detailed move information (see the entry for the ability query below). If the app is not run with the `--expanded` flag then only store the move name and the level it is acquired.

Get Ability Data

The user should be able to provide the name or id of an Ability and query its information. You want to create an `Ability` Class that can store (and ultimately be used to report) the following data:

Ability

Name	Type	Expandable	Notes
Name	string	☐	
ID	int	☐	
Generation	string	☐	
Effect	string	☐	Only take the entry for english. (language name "en")
Effect (Short)	string	☐	Only take the entry for english. (language name "en")
Pokemon	list string	☐	This would be a simple list of pokemon names. We won't be expanding these.

Get Moves Data

The user should be able to provide the name or id of an Ability and query its information. You want to create an `Move` Class that can store (and ultimately be used to report) the following data:

Move

Name	Type	Expandable	Notes
Name	string	☐	
ID	int	☐	
Generation	string	☐	
Accuracy	int	☐	
PP	int	☐	
Power	int	☐	
Type	string	☐	
Damage Class	string	☐	
Effect (Short)		☐	Only take the entry for english. (language name "en")

Printing a Report

When a query has finished, it's results must be printed. This "Report" would either be printed to a file, or to the console (depending on whether the `--output` flag was provided or not).

In both cases the app must print the information in a nicely formatted readable manner. If printing a pokemon query and if the `--expanded` flag was provided, make sure to append Stat, Move and Ability information from the sub-queries as well.

Grading

The assignment is marked out of 20. For full marks, you must:

1. Correctly implement the features mentioned above and meet all the requirements set out in this assignment. Handle user input and avoid crashes. - **10 Marks.**
2. Draw a UML Class Diagram that accurately represents your system. - **3 Marks**
3. Correctly format and comment your code. Follow **all** the PEP-8 Guidelines and good coding practices. - **2 Marks**
4. Good coding practices. No unnecessary for loops, use appropriate data types, meaningful identifiers, proper use of SOLID principles where applicable etc. - **4 Marks**
5. Format your output. Make sure your messages display the correct information in a pleasant, readable manner. You could even use ASCII art if you dare! - **1 Mark**

Please remember that this is an individual assignment. I strongly encourage you to share ideas and concepts, but sharing code or submitting someone else's work is not allowed and will result in a grade of **ZERO!** .

Good luck, be creative and have fun!