Accessing JSON Data using Pythons Requests Library

These following are important details to understand when working with Pythons **Requests** library. One of the most popular methods used in the Requests library is response.json(). This method belonging to the Requests library loads JSON data into Python objects. Once this is complete, for example we can then write code to access different areas of the Python Dictionary parsed from the JSON object obtained through the API request body.

The programming paradigm **Object Orientated Programming** is an important concept to grasp in computer science. Python is considered an (OOP) language which essentially means everything in Python is an object. Objects will have specific methods attributed to them. Understanding this concept is important in this use case working with the Request library when we want to access Python objects.

Working with JSON data loaded into Python objects can be in the popular format of Python dictionaries and lists. When we load the JSON data into Python object initially using the above method this will be converted into Pythons in built data structure the Dictionary now acting as the outer layer of the programming object.

We will focus on two of Pythons in built data structures for explanatory purposes. Python dictionaries can contain for example a list of dictionaries or vice versa to note which can increase coding complexity to access data. This is important to consider when building and engineering data. What is important here is that we understand based on the concept (OOP) that a Python list is a separate data structure to a Python dictionary essentially being two different objects. This is the case for all Python in-built data structures including lists, dictionaries, tuples and sets. These objects will have specific methods attached for differing functionality and will have differing attributes pre-defined such as being ordered and changeable.

- List is a collection which is ordered and changeable. Allows duplicate members.
- Tuple is a collection which is ordered and unchangeable. Allows duplicate members.
- Set is a collection which is unordered, unchangeable*, and unindexed. No duplicate members.
- Dictionary is a collection which is ordered** and changeable. No duplicate members.

NOTE
It can be useful when trying to understand OOP to visualize an empty object (box or rectangle) and on the outside this will have pre-defined methods attached that will govern the functionality of the object in question.

To grasp an understanding of the above to be able to work efficiently with the Requests library researching Python data structures is advisable. A good reference point is the following website w3schools which covers the basics on Python and many other languages and concepts.

Moving on from data structures to data types can be considered as a classification of data which tells the compiler or interpreter how the programmer intends to use the data. Essentially, data types will govern the type of data that can be held in a variable and act as a placeholder for example an integer value. It is Important to remember data types (OOP) are also objects with specific built in methods. On the other hand, data structures will hold collections of data which can be off different data types depending on the object in question (ex. Dictionary).

So to conclude the above and highlight once again the importance of understanding the above concepts in depth will essentially determine our ability to access Python data structures and manipulate this to case specific requirement and needs.

Accessing JSON object using Python

Python Data Structure & Data Types	**JSON Data Structures & Data Types**
Dictionary	Object
List	Array
Tuple	Array

Python Data Types	**JSON Data Types**
str	String
int	Number
float	Number
True	true
False	false
None	null

Response json()

Python requests simplified are generally used to gather the content from a particular resource URL. Whenever we make a request to a specified URL through Python, it returns a response object. Now, this response object would be used to access certain features such as content, headers, body etc. This article revolves around how to check the response.json() out of a response object. It is one of the most used methods in the Requests module.

The response.json() loads JSON data into Python objects as we have discussed above.

Parsing Python requests Response JSON Content

In the below code, firstly we imported the requests module and then fetch the data from an API using requests.get() method and store in the response variable. When we print the response this should print '<Response [200]>' which is the HTTP code that indicates success. This may not always be the case, but we won't go into this for the moment.

To print the JSON data fetched we have used json() method which prints the JSON data in the Python dictionary format as seen in the output. In this way, we can parse JSON responses in Python.

```
# import requests module

import requests

# Making a get request

response = requests.get('https://api.github.com')

# print response
```

```
print(response)

# print json content

print(response.json())
```

response.text – Python requests

response.text returns the content of the response, in Unicode basically referring to Binary Response content.

```
# import requests module

import requests

# Making a get request

response = requests.get('https://api.github.com')

# printing request text

print(response.text)
```