17.3. Processing JSON results

JSON stands for JavaScript Object Notation. It looks a lot like the representation of nested dictionaries and lists in python when we write them out as literals in a program, but with a few small differences (e.g., the word null instead of None). When your program receives a JSON-formatted string, generally you will want to convert it into a python object, a list or a dictionary.

Again, python provides a module for doing this. The module is called json. We will be using two functions in this module, `loads` and `dumps`.

`json.loads()` takes a string as input and produces a python object (a dictionary or a list) as output.

Consider, for example, some data that we might get from Apple's iTunes, in the JSON format:

Save & Run Original - 1 of 1 Show in CodeLens

```
1 import json
2 a_string = '\n\n\n{\n "resultCount":25,\n "results": [\n{"wrapperType":"track", "ki
3 print(a_string)
4 d = json.loads(a_string)
5 print("------")
6 print(type(d))
7 print(d.keys())
8 print(d['resultCount'])
9 # print(a_string['resultCount'])
10
```

```
{
 "resultCount":25,
 "results": [
{"wrapperType":"track", "kind":"podcast", "collectionId":10892}]}
------
<class 'dict'>
['resultCount', 'results']
25
```

Activity: 1 -- ActiveCode (ac17_3_1)

The other function we will use is `dumps`. It does the inverse of `loads`. It takes a python object, typically a dictionary or a list, and returns a string, in JSON format. It has a few other parameters. Two useful parameters are sort_keys and indent. When the value True is passed for the sort_keys parameter, the keys of dictionaries are output in alphabetic order with their values. The indent parameter expects an integer. When it is provided, dumps generates a string suitable for displaying to people, with newlines and indentation for nested lists or dictionaries. For example, the following function uses json.dumps to make a human-readable printout of a nested data structure.

Save & Run Original - 1 of 1 Show in CodeLens

```
1 import json
2 def pretty(obj):
3     return json.dumps(obj, sort_keys=True, indent=2)
4
5 d = {'key1': {'c': True, 'a': 90, '5': 50}, 'key2':{'b': 3, 'c': "yes"}}
6
7 print(d)
8 print('--------')
9 print(pretty(d))
10
```

```
{'key1': {'c': True, 'a': 90, '5': 50}, 'key2': {'c': 'yes', 'b': 3}}
--------
{"key1":{"5":50,"a":90,"c":true},"key2":{"b":3,"c":"yes"}}
```

Activity: 2 -- ActiveCode (ac17_3_2)

Check Your Understanding

nested-9-1: Because we can only write strings into a file, if we wanted to convert a dictionary d into a json-formatted string so that we could store it in a file, what would we use?

- A. json.loads(d)
- B. json.dumps(d)
- C. d.json()

Check me Compare me

✔ dumps turns a list or dictionary into a json-formatted string

Activity: 3 -- Multiple Choice (question17_3_1)

nested-9-2: Say we had a JSON string in the following format. How would you convert it so that it is a python list?

```
entertainment = """[{"Library Data": {"count": 3500, "rows": 10, "locations": 3}}, {"Mo
vie Theater Data": {"count": 8, "rows": 25, "locations": 2}}]"""
```

- [ ] A. entertainment.json()
- [ ] B. json.dumps(entertainment)
- [x] C. json.loads(entertainment)

Check me | Compare me

✔ Correct.
C. loads (load from string) turns a json-formatted string into a list or dictionary

Activity: 4 -- Multiple Choice (question17_3_2)

You have attempted 5 of 4 activities on this page

17.2. Nested Dictionaries">
ested Dictionaries">
17.4. Nested Iteration">
✔ Completed. Well Done!
17.4. Nested Iteration">Next Section - 17.4. Nested Iteration

| Back to top