LECTURE 15

JSON

MCS 260 Fall 2021

David Dumas

REMINDERS

- We'll talk about Project 2 a bit today
- Homework 5 due tomorrow at 10am
- Worksheet 6 posted

HINTS

- In the terminal:
  - Tab = autocomplete
  - ↑ = previous command
  - `ls` lists files in current directory
- You can download sample scripts from Github using the "Raw" button, then File>Save.

JSON

JSON stands for **JavaScript object notation**. It is a format for storing various types of data in text files. Many languages can read and write this format.

Python has a module for reading and writing JSON.

By using that module, you can move various types of data in and out of files cleanly, without needing to handle the read/write details yourself.

JSON TYPES

Supported basic types:

- **string** — must use double quotes.
- **number** — float, int, other? Up to reader.
- **boolean** — lower case names `true`, `false`.
- **null** — similar to Python's `None`.

JSON TYPES

Supported composite types:

- **array** — ordered sequence of values like Python `list`. Surrounded by square brackets, values separated by commas.
- **object** — associative array like Python `dict`. Surrounded by curly braces, comma separator. Keys must be strings.

A JSON file must contain a **single value**. Most often it is an object or array.

WRITING JSON

Import the module `json` to make JSON function available.

You'll need to open a file yourself; JSON functions expect a file object.

Use `json.dump(val,f)` to write `val` to file object `f` as JSON.

Python to JSON type conversion table:

- `dict` → `object`
- `list` or `tuple` → `array`
- `int` or `float` → `number`
- `bool` → `boolean`
- `None` → `null`

JSON STRING

If you want to handle writing yourself, you can ask the JSON module to convert a value to a JSON string, e.g.

```
json.dumps(val) # make JSON string out of val
```

READING JSON

Use `json.load(f)` to interpret contents of file object `f` as JSON and return the decoded result.

`json.loads(text)` will instead process string `text` as JSON.

NOT SUPPORTED IN JSON

- Complex numbers
- Date/time types
- Distinctions between:
  - `int` and `float`*
  - `tuple` and `list`
- Comments

* But Python's `json` module will try to guess when reading.

EXAMPLE

Let's modify `wordstats3file.py` so that it writes a dictionary of statistics to a JSON file.

That way, another program can read and use the report easily without concern for formatting details.

EXAMPLE

A list of "awesome" JSON datasets.

One of the items in that list is a data file listing episodes of the Netflix TV show Stranger Things. The link to it is actually broken, but the list is available at:

https://api.tvmaze.com/shows/2993/episodes

Let's download it and poke around using Python.

REFERENCES

- The `json` module documentation is good and has some helpful examples.
- data.gov has a directory of many US government APIs and data portals. Many of these require you to first sign up for a free access key.
- A list of some "awesome" JSON datasets

REVISION HISTORY

- 2021-09-27 Initial publication
- 2021-09-27 Add episode list link