Saving Data to a JSON File

JSON looks remarkably like a data structure you might define manually in JavaScript. This is no accident; you can actually use the JavaScript `eval()` function to "decode" json-serialized data. (The usual caveats about untrusted input apply, but the point is that JSON *is* valid JavaScript.) As such, JSON may already look familiar to you.

```
shell = 1

basic_entry = {}                                        #①
basic_entry['id'] = 256
basic_entry['title'] = 'Dive into history, 2009 edition'
basic_entry['tags'] = ('diveintopython', 'docbook', 'html')
basic_entry['published'] = True
basic_entry['comments_link'] = None
import json
with open('basic.json', mode='w', encoding='utf-8') as f:  #②
    json.dump(basic_entry, f)                           #③
```

① We're going to create a new data structure instead of re-using the existing `entry` data structure. Later in this chapter, we'll see what happens when we try to encode the more complex data structure in JSON.

② JSON is a text-based format, which means you need to open this file in text mode and specify a character encoding. You can never go wrong with UTF-8.

③ Like the `pickle` module, the `json` module defines a `dump()` function which takes a Python data structure and a writeable stream object. The `dump()` function serializes the Python data structure and writes it to the stream object. Doing this inside a `with` statement will ensure that the file is closed properly when we're done.

So what does the resulting JSON serialization look like?

```
you@localhost:~/diveintopython3/examples$ cat basic.json
{"published": true, "tags": ["diveintopython", "docbook", "html"], "comments_link": null,
```

```
"id": 256, "title": "Dive into history, 2009 edition"}
```

That's certainly more readable than a pickle file. But JSON can contain arbitrary whitespace between values, and the `json` module provides an easy way to take advantage of this to create even more readable JSON files.

```
shell = 1

import json
with open('basic-pretty.json', mode='w', encoding='utf-8') as f:
     json.dump(basic_entry, f, indent=2)                              #①
```

① If you pass an `indent` parameter to the `json.dump()` function, it will make the resulting JSON file more readable, at the expense of larger file size. The `indent` parameter is an integer. 0 means "put each value on its own line." A number greater than 0 means "put each value on its own line, and use this number of spaces to indent nested data structures." And this is the result:

```
you@localhost:~/diveintopython3/examples$ cat basic-pretty.json
{
  "published": true,
  "tags": [
    "diveintopython",
    "docbook",
    "html"
  ],
  "comments_link": null,
  "id": 256,
  "title": "Dive into history, 2009 edition"
}
```