Work with JSON files, Python, and Python dictionaries, to load, alter, and then save data back to disk in this notebook

Serialize JSON from Python

You can serialize to and from JSON in Python using the `json` module

In [1]:
```
# the JSON module can take certain Python data structures like dictionaries ar
import json
```

From Python, convert a dictionary into a JSON string

In [2]:
```
data = {"grape": "Cabernet Franc", "species": "Vitis vinifera", "origin": "Bo
```

In [3]:
```
# Convert Python data to JSON. The `.dumps()` method takes a data structure as
# mnemonic: dumps -> DUMP to String
json.dumps(data)
```

Out[3]:
```
'{"grape": "Cabernet Franc", "species": "Vitis vinifera", "origin": "Bordeau
x, France"}'
```

In [4]:
```
# Convert a JSON string into a Python data structure
# first, define the json data with the string data
json_data = json.dumps(data)
json_data
```

Out[4]:
```
'{"grape": "Cabernet Franc", "species": "Vitis vinifera", "origin": "Bordeau
x, France"}'
```

In [5]:
```
# Now load it into Python
# mnemonic: loads -> LOAD from String
json.loads(json_data)
```

Out[5]:
```
{'grape': 'Cabernet Franc',
 'species': 'Vitis vinifera',
 'origin': 'Bordeaux, France'}
```

In [6]:
```
# Python dictionaries are not the only data structure allowed. Use lists as we
collection = [data, data]
print(collection)
# may look similar in the output, but the difference is that JSON is now a str
json.dumps(collection)
```

```
[{'grape': 'Cabernet Franc', 'species': 'Vitis vinifera', 'origin': 'Bordeau
x, France'}, {'grape': 'Cabernet Franc', 'species': 'Vitis vinifera', 'origi
n': 'Bordeaux, France'}]
```

Out[6]:
```
'[{"grape": "Cabernet Franc", "species": "Vitis vinifera", "origin": "Bordea
ux, France"}, {"grape": "Cabernet Franc", "species": "Vitis vinifera", "orig
in": "Bordeaux, France"}]'
```

JSON Formatting

The `json` module in Python allows more than just loading and parsing JSON. It can be used to format it nicely. Formatting is crucial when dealing with nested data (a dictionary within a dictionary for example).

It is common for HTTP APIs and JSON files to present JSON as a single line. In this section, you will use formatting options in the JSON module to improve the readability of nested information in JSON.

In [9]:
```
# define a nested data structure in a single line
grape_data = {"name": "Cabernet France", "regions": [{"country": "France", "su
# Serialize the Python dictionary to a JSON string, but using extra formattin
# and using 4 spaces for indentation
data_as_json = json.dumps(grape_data, sort_keys=True, indent=4)
print(data_as_json)
```

```
{
    "name": "Cabernet France",
    "regions": [
        {
            "country": "France",
            "sub-regions": [
                "Bordeaux",
                "Loire Valley"
            ]
        },
        {
            "country": "Italy",
            "sub-regions": [
                "Apulia",
                "Tuscany"
            ]
        },
        {
            "country": "Argentina",
            "sub-regions": [
                "Mendoza",
                "Lujan de Cuyo",
                "Salta"
            ]
        }
    ]
}
```

In [10]:
```
# Try other variations like indenting 2 spaces and not sorting keys:
data_as_json = json.dumps(grape_data, sort_keys=False, indent=2)
print(data_as_json)
```

```
{
  "name": "Cabernet France",
  "regions": [
    {
      "country": "France",
      "sub-regions": [
        "Bordeaux",
        "Loire Valley"
      ]
    },
    {
      "country": "Italy",
      "sub-regions": [
        "Apulia",
        "Tuscany"
      ]
    },
    {
      "country": "Argentina",
      "sub-regions": [
        "Mendoza",
        "Lujan de Cuyo",
        "Salta"
      ]
    }
  ]
}
```

Serialize JSON from a file

Python can read JSON files and load them as Python data structures, which can also be saved back to the file system as a valid JSON file. In the next few cells, read a JSON file from the file system, and then use the `json` module to parse the JSON and load it into Python.

The process of reading a foreign format like JSON and loading it into Python is called serializing it.

In [11]:
```
# There are JSON files in the `sample_data/` directory. When working with path
import os
os.path.exists('sample_data/wine-ratings.json')
```

Out[11]: True

In [12]:
```
# read the JSON file and then parse it using the `.load()` method
# note the subtle difference, this is the `.load()` method (no 's'), not `.loa
with open('sample_data/wine-ratings.json') as f:
    loaded_json = json.load(f)
print(loaded_json.keys())
print(f"Number of items: {len(loaded_json['name'])}")
```

```
dict_keys(['name', 'grape', 'region', 'variety', 'rating', 'notes'])
Number of items: 780
```

Serialize from Python to a JSON file

Now that you've loaded JSON from a file into Python, do some data sampling, extract some interesting fields and then save the newly manipulated data to a file on disk as JSON.

In [13]:
```
# sample some items from the json file and then save it as a new file
names = loaded_json['name']
len(names)
```

Out[13]:
```
780
```

In [17]:
```
# these names are using an index, like {"0": "Some Name and Year"}. Update the
names_only = list(names.values())
names_only
```

```
 'Laurenz V Singing Gruner Veltliner 2015',
 'Lava Cap American River Red',
 'Lava Cap Barbera 2010',
 'Lava Cap Battonage Chardonnay 2012',
 'Lava Cap Cabernet Sauvignon 2013',
 'Lava Cap Cabernet Sauvignon 2016',
 'Lava Cap Petite Sirah 2013',
 'Lava Cap Petite Sirah 2014',
 'Lava Cap Petite Sirah 2016',
 'Lava Cap Reserve Chardonnay 2015',
 'Lava Cap Reserve Chardonnay 2018',
 'Lava Cap Reserve Chardonnay 2016',
 'Lava Cap Reserve Merlot 2015',
 'Lava Cap Sauvignon Blanc 2015',
 'Lava Cap Sauvignon Blanc 2017',
 'Lava Cap Syrah 2009',
 'Lava Cap Syrah 2014',
 'Lava Cap Syrah 2013',
 'Lava Vine Winery Knights Valley Reserve Cabernet Sauvignon 2013',
 'Lava Vine Winery Napa Valley Cabernet Sauvignon 2014',
```

In [18]:
```
# now use the `.dump()` JSON method (note no 's'!) to save it to a new JSON f
with open('sample_data/wine_names.json', 'w') as f:
    json.dump(names_only, f)
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
<ipython-input-18-12fc78ed5762> in <module>
      1 # now use the `.dump()` JSON method (note no 's'!) to save it to a n
ew JSON file
----> 2 with open('sample_data/wine_names.json', 'w') as f:
      3     json.dump(names_only, f)

OSError: [Errno 30] Read-only file system: 'sample_data/wine_names.json'
```

In [ ]:
```

```