# JSON

**JAVASCRIPT OBJECT NOTATION (JSON)** = (Jupyter Notebook .ipynb) File format

> Stores data in a way that is easily readable by both humans and machines

> Useful for browser and server to exchange data thus used in web-based applications

> Very similar to Python dictionaries. They contain a key which has a corresponding value

```
{
  "simple-property": "a simple value",

  "object-property": {
      "first-property": "first value",
      "second-property": "second value"
  },

  "array-property": [
      { "item-1-property-1": "one",
        "item-1-property-2": 2 },
      { "item-2-property-1": "three",
        "item-2-property-2": 4}
  ]
}
```

---

### Read JSON Files

> Python has a library called `json` that can read, write, or append elements from or to a JSON file

> Use a context manager, set the mode we want to use, and then use a method. In this case, for reading a file, we use the `load` method

> Can be a dictionary/list/string/numbers

```
import json

with open('JSON_sample.json', mode='r') as f:
  json_dict = json.load(f)

print(json_dict)
```

```
import json

with open('JSON_sample.json', mode='r') as f:
  string_json = f.read()
data = json.loads(f)

print(data)
```

> You don't have to load it directly from a file, it can be a string received over the internet. Use `string_json` to read and `json_loads` to load

---

### Create JSON Files

> Use a context manager, set the mode we want to use, and then use a method. In this case, for reading a file, we use the `load` method

> Create JSON files from dictionaries. The mode of the context manager is `w`. The method is `dump`. It accepts the data and the file to dump the data into.

> WILL NOT WORK WITH SINGLE QUOTES

```
test_dict = {'a': 1, 'b': 2, 'c': 3, 'd': 4}
with open('JSON_test.json', mode='w') as f:
  json.dump(test_dict, f)

# You can also have a string containing a JSON and
# WILL NOT WORK WITH SINGLE QUOTES
x = '{"name": "John", "age": 30, "city": "New York"
y = json.loads(x)
print(y)
print(type(y))

# Dictionary to JSON
test_dict = {'a': 3, 'b': 4}
new_json = json.dumps(test_dict)
print(new_json)
```

---

### Pandas

```
test = pd.read_json('data/modified.json', orient='records', lines=True)
df_nice = pd.json_normalize(df["Employees"]) # Normalize so each column is each value
df.to_json('data/modified.json', orient='records', lines=True)
```