Data Exchange Formats

Data Manipulation in Python

Data Exchange Formats

- XML
  - A verbose textual representation of trees
- JSON
  - JavaScript Object notation – like a Python `dict`

XML Format

people.xml:

```
<?xml version="1.0"?>

<people>
  <person>
    <firstName>Alan</firstName>
    <lastName>Turing</lastName>
    <professions>
      <profession>Computer Scientist</profession>
      <profession>Mathematician</profession>
      <profession>Computer Scientist</profession>
      <profession>Cryptographer</profession>
    </professions>
  </person>
  <person>
    <firstName>Stephen</firstName>
    <lastName>Hawking</lastName>
    <professions>
      <profession>Physicist</profession>
      <profession>Comedian</profession>
    </professions>
  </person>
</people>
```

Parsing XML with ElementTree

Use Python's built-in ElementTree API

```
In [17]: import xml.etree.ElementTree as ET

In [18]: root = ET.parse('people.xml')

In [21]: persons = root.findall("person")

In [24]: for person in persons:
    ...:     print(person.find("firstName").text, end="")
    ...:     print(person.find("lastName").text)
    ...:     for profession in person.find("professions"):
    ...:         print("\t", profession.text)
    ...:
AlanTuring
     Computer Scientist
     Mathematician
     Computer Scientist
     Cryptographer
StephenHawking
     Physicist
     Comedian
```

JSON Format

Just like Python data structures except:

- double quotes – " – for all strings
- `true` and `false` boolean literals instead of `True` and `False`
- `null` instead of `None`

Here's the XML people example represented as JSON:

```
{
  "people": {
    "person": [
      {
        "firstName": "Alan",
        "lastName": "Turing",
        "professions": {
          "profession": ["Computer Scientist", "Mathematician",
                         "Computer Scientist", "Cryptographer"]
        }
      },
      {
        "firstName": "Stephen",
        "lastName": "Hawking",
        "professions": {
          "profession": ["Physicist", "Comedian"]
        }
      }
    ]
  }
}
```

Reading JSON

Use Python's built-in JSON encoder and decoder

- Loading from a string:

```
In [2]: json.loads('{"CS4400": ["CS1301","CS1315","CS1371"],
     "CS3600":["CS1332"]}')
Out[2]: {'CS3600': ['CS1332'], 'CS4400': ['CS1301', 'CS1315', 'CS1371']}
```

- Loading from a file (notice that you must provide a file object, not just a file name):

```
In [8]: cat fall2017-breaks.json
{
    "2017-09-04": "Labor Day",
    "2017-10-09": "Fall Student Recess",
    "2017-10-09": "Fall Student Recess",
    "2017-11-22": "Student Recess",
    "2017-11-23": "Thanksgiving Break",
    "2017-11-24": "Thanksgiving Break"
}

In [9]: json.load(open('fall2017-breaks.json'))
Out[9]:
{'2017-09-04': 'Labor Day',
 '2017-10-09': 'Fall Student Recess',
 '2017-11-22': 'Student Recess',
 '2017-11-23': 'Thanksgiving Break',
 '2017-11-24': 'Thanksgiving Break'}
```

Writing JSON

- Dumping to a string

```
In [11]: prereqs = {'CS3600': ['CS1332'], 'CS4400': ['CS1301', 'CS1315',
     'CS1371']}

In [12]: json.dumps(prereqs)
Out[12]: '{"CS3600": ["CS1332"], "CS4400": ["CS1301", "CS1315", "CS1371"]}'
```

- Dumping to a file (notice the write-mode file object):

```
In [14]: json.dump(prereqs, open('prereqs.json', 'wt'))

In [15]: cat prereqs.json
{"CS3600": ["CS1332"], "CS4400": ["CS1301", "CS1315", "CS1371"]}
```