XML

Data Manipulation in Python

Georgia
Tech

Data Exchange Formats

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

XML Format

people.xml:

```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>
```

Georgia
Tech

Processing XML

- Python's built-in ElementTree API
  - Builds a nested set of objects representing a Document Object Model (DOM) tree
- xmltodict "makes working with XML feel like you are working with JSON"

Georgia
Tech

Parsing XML with ElementTree

```
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
```

Parsing XML with `xmltodict`

```
In [5]: ps = xmltodict.parse(open('people.xml').read())

In [6]: ps
Out[6]:
OrderedDict([('people',
             OrderedDict([('person',
                          [OrderedDict([('firstName', 'Alan'),
                                        ('lastName', 'Turing'),
                                        ('professions',
                                         OrderedDict([('profession',
                                                      ['Computer Scientist',
                                                       'Mathematician',
                                                       'Computer Scientist',
                                                       'Cryptographer'])]))]),
                           OrderedDict([('firstName', 'Stephen'),
                                        ('lastName', 'Hawking'),
                                        ('professions',
                                         OrderedDict([('profession',
                                                      ['Physicist',
                                                       'Comedian'])]))])])]))])
```

JSON Format

Just like Python data structures but you have to use double quotes for strings. 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"]
        }
      }
    ]
  }
}
```

Georgia
Tech

Processing JSON

Use Python's built-in JSON encoder and decoder

- ▶ Loading from a string:

- ▶ Loading from a file:

- ▶ Dumping to a string

- ▶ Dumping to a file:

Georgia
Tech