Python templates considered mostly harmless

I have had difficulty letting templating languages into my life.

Way back when I was first working with the web I ran into PHP and ASP – both of which I considered really awful. Why would you want to put your code in the middle of a web page? I suppose that both ASP and PHP have both improved architecturally since then, but I have not wanted to get involved. Pretty much stayed away from anything that looked remotely PHP-like. That included templating systems.

Last week I got involved in a quick optimization task. I was looking at reducing the time it takes to process around 200,000 records of AANRO data for ingest into a Fedora repository. Our first try used techniques we developed as part of the RUBRIC data migration toolkit, but I ended up replacing the whole process, because things kept failing on me and running multiple XSLT stylesheets just wasn't getting the job done in a reasonable time (did you know that on Linux you can only have 32000 files in a directory? True, at least our default Ubuntu and Red Hat installs. And that did you know that no matter how careful you are cleaning up after yourself the Python bindings for libxslt have horrible memory leaks?)

I finished up with a prototype spike solution that could churn through the whole lot of data in a few minutes, rather than some hours, and not turn my MacBook into a space heater, either.

The program sucked up the data record by record into a Python dict and used variable interpolation to format it, like this (yes I know there are better ways to make XML):

```
for author in authors:

  authors += "<dc:creator>%s</dc:creator>" % author
```

I ended up with half a dozen little functions to format bits and pieces and add them to a dictionary. Then I realised that the whole thing would be lot clearer and easy to maintain with a templating language. Obvious, but not to me.

I picked Cheetah and spent half an hour or so seriously reducing the complexity of my code.

I ended up with one simple template file with all the FOXML for a single record. Not much of this is reusable for other jobs, because it's specific to AANRO so why complicate it with lots of different parts? Here's the bit that does some of the MODS metadata:

```
    #for name, affiliation in $affiliations

    <name type="personal">

            <namePart>$name</namePart>

            <affiliation>$affiliation</affiliation>

            <role>

                <roleTerm authority="marcrelator"
type="text">creator</roleTerm>

            </role>
```

```
    </name>
    #end for
    <typeOfResource>text</typeOfResource>
    <genre authority="AANRO">$doctype</genre>
    <originInfo>
      <dateIssued>$pubyear</dateIssued>
    </originInfo>
    <abstract>$abstract</abstract>
    <note>$notes</note>
    #for $subject in $subjects
     <subject>
            <topic>$subject</topic>
    </subject>
    #end for
```

The advantage of this approach is that a metadata specialist like RUBRIC's Neil Godfrey can maintain the mapping directly, if the programmers explain the data structure.