# 1 Installation

To use LaTeXML, you must be using some version of Python 2, and must have `pdflatex` installed on your system. The following Python packages are also required:

- `argparse`
- `xml`

# 2 Writing a Problem

To write a problem using LaTeXML, you must first understand the basics of the XML encoding being used.

## 2.1 Problem XML Specification

Here is a sample problem, for the sake of explanation:

```
<problem>
  <used year="2015">Homework 5</used>
  <used year="2014" private="true">Midterm 1</used>
  <version id="2">
    <author>cjk</author>
    <year>2016</year>
    <topics>number_theory graph_theory</topics>
    <types>proof induction</types>
    <param name="modulus">3</param>
    <dependency>tikz</dependency>
    <body>
      The modulus is \modulus.
    </body>
    <solution>
      TODO
    </solution>
    <rubric>
      TODO
    </rubric>
  </version>
  <version id="1">
    ...
  </version>
</problem>
```

**A Problem XML file MUST have root tag `problem`, which is made up of one or more child `versions`.** Loosely speaking, two 'versions' should be part of the same 'problem' if having the solution to one would make writing the other's solution trivial. For example: adding a part to, theming, or changing the numbers in a previous year's problem would call for a new version of the problem. If several different problems are combined into one, that is generally a new problem (rather than a version of any of them).

**Each version MUST have exactly one year** (in a `<year>` tag), which may be "Unknown". This is the year that the version was written.

**Each version MUST have at least one author, at least one topic, and at least one type.** Each of these tags accepts a comma and/or whitespace separated list of items; thus,

```
<authors>cjk nschank, kl47</authors>
```

would be parsed as three authors: cjk, nschank, and kl47. Additional instances of a tag will append to the list, so e.g. multiple `<author>` tags within a version are allowed.

Each of these fields accepts both the singular and plural of their tagname, purely for convenience. Thus...

**Authors** `<author>` or `<authors>`

**Topic** `<topic>` or `<topics>`

**Type** `<type>` or `<types>`

Note that, while either tagame is accepted, the beginning and ending tags must match.

See the next section for an explanation of acceptable values for topic and type.

**Each version MUST have a body, a solution, and a rubric.** Each of these fields should be filled with arbitrary LaTeX; whatever you would have put between `\begin{document}` and `\end{document}` goes here. It is expected that, if a solution or rubric is not complete, the four letters "TODO" (case sensitive) should appear somewhere within their text. This allows the LaTeXML tools to keep track of any problems that need attention.

As an important note, the characters &, <, and > are special characters within XML. In order for the problem to be parsed, you MUST escape them with the XML sequences `&amp;`, `&lt;`, and `&gt;` respectively. They will be unescaped before being parsed as LaTeX, so should be treated identically to their corresponding characters (e.g. tables will contain many instances of "`&amp;`").

**Each version MUST have an `id` attribute, unique within the problem, set to a positive integer, such that the newest version has the highest ID.**

A version MAY have zero or more `param` or, equivalently, `parameter` tags, which MUST have a name attribute. This is equivalent to temporarily creating a command `\name` which produces the value given in the tag's field. In the provided example, the command "`\modulus`" will evaluate to 3. This field is intended for use within problems that can be easily changed without needing to create a new version. As an example, the name of a person or object within a problem should be refactored into a parameter, so that the problem can be changed easily.

A version MAY have zero or more `dependency` (also allowed: `dep`, `deps`, `dependencies`) tags, each of which should be a comma and/or whitespace-separated list of packages which are required in

order to build the problem. `tikz` is the most commonly included by far. These dependencies will by dynamically imported when building an assignment including this problem.

`usedin` tags should never be created or edited by hand.