The moodle package: generating Moodle quizzes via LaTeX[*]

Anders Hendrickson[†]
anders.o.f.hendrickson AT gmail.com

Matthieu Guerquin-Kern[‡]
guerquin-kern AT crans.org

January 4, 2021

1 Motivation

The acronym Moodle stands for "Modular Object-Oriented Dynamic Learning Environment." It is an open source learning management system (LMS) employed by many universities, colleges, and high schools to provide digital access to course materials, such as notes, video lectures, forums, and the like; see `https://moodle.com/moodle-lms/` for more information. One of the many useful features of Moodle is that mathematical and scientific notation can be entered in LaTeX or TeX code, which will be typeset either through a built-in TeX filter or by invoking MathJax.

For instructors who want to give students frequent feedback, but lack the time to do so, a particularly valuable module in Moodle is the *quiz*. A Moodle quiz can consist of several different types of questions—not only multiple choice or true/false questions, but also questions requiring a short phrase or numerical answer, and even essay questions. All but the essay questions are automatically graded by the system, and the instructor has full control over how often the quiz may be attempted, its duration, and so forth. Feedback can be tailored to specific mistakes the student makes.

All these features make Moodle quizzes very useful tools for instructors who have access to them. Unfortunately, the primary way to create or edit a Moodle quiz is through a web-based interface that can be slow to operate. To users of LaTeX, accustomed to the speed of typing source code on a keyboard alone, the agonizing slowness of switching between mouse and keyboard to navigate a web form with its myriad dropdown boxes, radio buttons, compounded with a perceptible time lag as one's Moodle server responds to requests, can produce a very frustrating experience. Moreover, editing is entirely impossible without network access.

Once the quiz is written, there is no easy way to view and proofread all the information of which it is made. Each question is edited on a separate webpage, which is so full

[*]This document corresponds to moodle.sty v0.8, dated 2021/01/04.

[†]original author of the package (`v0.5`), inactive.

[‡]current maintainer, author of the updates (`v0.6` to `v0.8`), partially funded by ENSEA Graduate School, France.

of options that it cannot be viewed on a single screen. An instructor has to spend much time checking over the newly created quiz in order to be confident there are no errors.

Added to all this is the frustration of managing graphics. If a question requires an image—say, asking a calculus student to interpret the graph of a function—the image must first be produced as a standalone file (e.g., in JPG or PNG format), uploaded to Moodle, and then chosen in a web-based HTML editor. Great is the vexation of the instructor who decided to alter a question, as there are more and more possibilities of error whenever multiple files must be kept synchronized.

Users of LaTeX are also accustomed to the speed and flexibility that comes from defining their own macros, which may be as brief as writing `\R` instead of `\mathbb{R}` or as complex as macros that generate entire paragraphs of text. The Moodle editor, by contrast, requires you to type `\mathbb{R}` every single time you want $\mathbb{R}$.

Finally, there is the question of archiving and reusing one's work. Much, much work goes into creating Moodle quizzes, which then reside on a Moodle server somewhere in the cloud in a format neither easily browsable nor easily modifiable.

LaTeX itself has the power to solve all these difficulties: it is swift to edit and swifter to compile a LaTeX document, and the PDF may be previewed onscreen or printed out for ease of proofreading. Mathematical graphics can be integrated within the main file through Ti*k*Z, and of course LaTeX macros can be customized. Using the present moodle package, a quiz author can type a quiz using familiar LaTeX syntax and document structure. Upon compilation, LaTeX will generate both a well-organized PDF that is easy to proofread and an XML file that can be uploaded directly to Moodle. The entire process is far faster than using Moodle's own web-based editor, makes it easier to catch one's mistakes, and the ultimate source code of one's work is a human-readable `.tex` file that can be archived, versioned, browsed, and edited offline.

Strictly speaking, the moodle package does not generate quizzes: it generates question banks that can be imported in the LMS. The teacher still needs to compose manually a quiz from the question banks. Hopefully, two Moodle features supported by the package make this task easier: categories and tags.

In this documentation the LMS is referred to as Moodle (uppercase M and roman font) while the LaTeX package that is documented here is referred to as moodle (all lower case and sans serif font).

2 Workflow

The process of creating a quiz in Moodle using this package is depicted in Figure 1. It follows a few steps:

1. Write a LaTeX document using `\usepackage{moodle}` as described below.

2. Compile the document to PDF using pdfLaTeX XeLaTeX or LuaLaTeX. This will also produce the file ⟨*jobname*⟩`-moodle.xml`.

3. Open Moodle, navigate to the desired course, and under “Question bank” select “Import.”

4. Select “Moodle XML format,” choose the XML file to upload, and press “Import.”

5. After Moodle verifies that the questions have been imported correctly, you may add them to your quizzes.

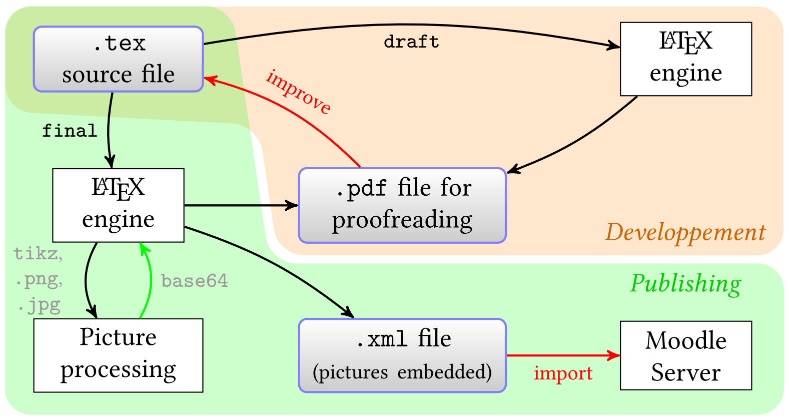

Figure 1: Block diagram describing a typical workflow using the moodle package.

3 Usage

3.1 Example Document

The following pages presume the reader already has some familiarity with creating and editing Moodle quizzes through the web interface. The `xkeyval` package is used to provide a key-value interface. Here is a simple example document:

```
\documentclass[12pt]{article}
\usepackage[section]{moodle}
\moodleregisternewcommands
\newcommand\monomial[1]{x^{#1}}
\newcommand\sillyanswer{What!?}
\begin{document}
\begin{quiz}{My first quiz}
  \begin{numerical}[points=2]{Basic addition}
    What is $8+3$?
    \item 11
  \end{numerical}
  \begin{shortanswer}[usecase]{Newton's name}
    What was Newton's first name?
    \item Isaac
    \item[fraction=0, feedback={\sillyanswer}] Fig
    \item[fraction=0] Sir
  \end{shortanswer}
  \begin{multi}[points=3]{A first derivative}
    What is the first derivative of $\monomial{3}$?
    \item $\frac{1}{4}\monomial{4}+C$
    \item[feedback={yes!}]* $3\monomial{2}$
    \item[feedback={\sillyanswer}]  $51$
  \end{multi}
\end{quiz}
\end{document}
```

Key features to note in this first example are that a `quiz` environment contains several question environments. Each question takes a name as a mandatory argument, and it may also take optional key-value arguments within brackets. The question environments resemble list environments such as `itemize` or `enumerate`, in that answers are set off by `\item`'s, but the question itself is the text that occurs before the first `\item`.

3.2 Package Options

`draft` `final` If the package option `draft` is invoked, by calling `\usepackage[draft]{moodle}` or `\documentclass[draft]{...}`, then no XML file will be generated. This is especially useful while editing a quiz containing graphics, so as to avoid the time spent converting image files. The package option `final` might be useful if one wants to avoid the option `draft` to be inherited from the `documentclass`.

`handout` If the package option `handout` is invoked (`\usepackage[handout]{moodle}`), the PDF file is generated clean from teacher-only information (answers, points, penalty, feedback, tags) and, hence, can be given to students for classroom work. In particular, as would Moodle do, answers in `matching` questions are shuffled and the option `shuffle` triggers the shuffling of choices offered (`multi` and `matching`). This is achieved thanks to the package `randomlist`, loaded if the option is invoked. The XML file is generated as usual.

`nostamp` By default, the package will output a stamp as a comment in the XML file. This stamp contains information gathered about the TeX engine, the operating system used and the package version. For instance:

```
<!-- This file was generated on 2020-11-30 by LuaLaTeX -->
<!-- running on Linux with the package moodle v0.8 -->
```

The package option `nostamp` prevents this stamp to be written in the XML file.

`section` `section*` `subsection` `subsection*` If the package option `section` is invoked (`\usepackage[section]{moodle}`), then each quiz is represented by a different LaTeX section. Starred variants correspond to unnumbered sections or subsections. To preserve compatibility with Version 0.5 of this package, the default is `subsection*`. Consequently, `\usepackage{moodle}` is equivalent to `\usepackage[subsection*]{moodle}`.

`tikz` The package option `tikz` is described in section 5.3.

`svg` The package option `svg` is described in section 5.5.

3.3 Quiz and Question Environments

`quiz` A `.tex` document to generate Moodle quizzes contains one or more `quiz` environments, within which various question environments are nested. The required argument to the `quiz` environment names a category for Moodles "question bank": after import, the questions defined in this environment will be gathered in this category.

`\begin{quiz}[`⟨*options*⟩`]{`⟨*category name*⟩`}`

There are no quiz-specific options, but any ⟨*options*⟩ set with `\begin{quiz}` will be inherited by all questions contained within that `quiz` environment.

`\moodleset` Options may also be set outside question environments with `\moodleset{`⟨*options*⟩`}`; these changes are local to TeX-groups.

`\setcategory` `\setsubcategory` Although the `quiz` environment defines a category by its own, one can change the current category inside the quiz environment and in between questions, using the macro `\setcategory{`⟨*category name*⟩`}`. A subcategory can also be defined with

Table 1: Valid positive options for the `fraction` key: $100 \cdot (p/q)$.

Denominator q	Numerator p									
	0	1	2	3	4	5	6	7	8	9
20	0	5								
10		10	20	30	40	50	60	70	80	90
9		11.11111								100
8		12.5							100	
7		14.28571						100		
6		16.66667				83.33333	100			
5		20			80	100				
4		25		75	100					
3		33.33333	66.66667	100						
2		50	100							
1	0	100								

`\setsubcategory{`⟨*subcategory name*⟩`}`. The categories and subcategories are reflected in the PDF file as sections, subsections, or subsubsections, in accordance to the package setting `section`, `section*`, `subsection`, or `subsection*`.

The syntax for each question environment is

```
\begin{⟨question type⟩}[⟨question options⟩]{⟨question name⟩}
    ⟨question text⟩
    \item[⟨item options⟩] ⟨item⟩
      ⋮
    \item[⟨item options⟩] ⟨item⟩
\end{⟨question type⟩}
```

The meaning of the ⟨*item*⟩s varies depending on the question type, but they usually are answers to the question. Details will be given below.

The following key-value options may be set for all questions:

`points` `default grade` By default, each question is worth 1 point on the quiz. This may be changed with the `points` key or its synonym, `default grade`; for example, `points=2` makes that question worth two points.

`penalty` The `penalty` is the fraction of points that is taken off for each wrong attempt; it may be set to any value between 0 and 1. The default is `penalty=0.10`.

`fraction` In most question types, it is possible to designate some answers as being worth partial credit—that is, some fraction of a completely correct answer. The `fraction` key may be set to any of the values given in Table 1, from `0` (entirely wrong) to `100` (entirely correct).

In questions where several choices can be selected (see `multi` with the option `multiple`), positive fractions must sum up to exactly 100. It is also possible to set negative fractions (from -100 to 0) for wrong choices, in order to prevent the selection of all choices from leading to a good grade. In this case, the value ranging from -100 to 0 must be the opposite of one of the values listed in Table 1.

`fractiontol` The package tries to match the `fraction` key to one of the admissible values. To this end, the tolerance is controlled by the `fractiontol` key. It defaults to `0.01` but may be changed. When no admissible fraction value is matched, the package throws an error.

`feedback` The `feedback` key sets text that will appear to the student after completing the quiz. For example, one might set

```
feedback={This question might show up in the final exam.}
```

The desired feedback should be included in braces.

Two kinds of feedback can be given. If the `feedback` key is set for a question, then that feedback will appear to each student regardless of the student's answer. Answer-specific feedback (perhaps explaining a common mistake) may also be given by setting the `feedback` key *at the individual answer.*

`tags` The `tags` key sets a keyword for the question that will be taken into account by Moodle for filtering purposes or classification of questions inside the question bank. It is possible for instance to build a quiz with questions cherry-picked among the set of questions holding a particular tag. For example, one might set

```
tags={easy}
```

The desired tag should be included in braces.

Tags can be assigned at two levels. If the `tags` key is set at the quiz level, then that tags will be assigned by default to each question of the quiz. Question-specific tags can be assigned by setting the `tags` key *at the question level.* Since only single tag is supported, the tag a the question-level overrides eventual tags specified at the quiz-level.

3.4 Question Types

We next discuss the various question types supported by moodle and the options that may be set.

3.4.1 True/False

`truefalse` The syntax for a True/False question is as follows:

```
\begin{truefalse}[⟨question options⟩]{⟨question name⟩}
    ⟨question text⟩
    \item* ⟨feedback when “true” is chosen⟩
    \item ⟨feedback when “false” is chosen⟩
\end{truefalse}
```

The correct answer is designated by the asterisk `*` after the `\item`; it need not appear first in the list.

Answer-specific feedback can also be defined as an item option, similarly to other types.

```
\begin{truefalse}[⟨question options⟩]{⟨question name⟩}
    ⟨question text⟩
    \item[feedback={⟨When “true” is chosen⟩}]*
\end{truefalse}
```

Note that, in this example, no feedback is defined for the incorrect answer “False”: the corresponding item can be omitted.

With the True/False question type, the `penalty` key has no effect.

3.4.2 Multiple Choice

multi The syntax for a classic multiple choice question, with only one correct answer, is as follows:

```
\begin{multi}[⟨question options⟩]{⟨question name⟩}
    ⟨question text⟩
    \item* ⟨correct answer⟩
    \item[⟨options⟩] ⟨wrong answer⟩
      ⋮
    \item[⟨options⟩] ⟨wrong answer⟩
\end{multi}
```

The correct answer is designated by the asterisk `*` after the `\item`; it need not appear first in the list.

shuffle The boolean key `shuffle` determines whether Moodle will rearrange the possible answers in a random order. Setting `shuffle=false` will guarantee that the answer appear in the order they were typed; the default is `shuffle=true`.

numbering Moodle offers different options for numbering the possible answers. You may set the `numbering` key to any of the following values, which mirror the usual LaTeX syntax: `alph`, `Alph`, `arabic`, `roman`, `Roman`, and `none`. Calling `numbering=none` produces an unnumbered list of answers. The Moodle syntax of `abc`, `ABCD`, `123`, `iii`, and `IIII` is also acceptable, but note that it requires *four* capital letters to obtain upper-case Roman or alphabetic numerals this way.

fraction The `fraction` key can be used to designate some wrong answers as being worth partial credit. For example, a question might read thus:

```
\begin{multi}{my question}
  Compute $\int 4x^3\,dx$.
  \item* $x^4+C$
  \item[fraction=50] $x^4$
  \item $12x^2$
\end{multi}
```

Thus the asterisk `*` is shorthand for `fraction=100`, whereas a bare `\item` sets `fraction=0`.

single By default, the `multi` environment produces a multiple choice question with only one correct answer; this is called `single` mode, and on Moodle it appears with radio buttons.

multiple It is also possible to write questions with possibly more than one correct answer, asking the user to check all correct answers. To do this, use the key `multiple` or `single=false`. The worth of each correct answers in `multiple` mode may be set by `fraction`, but Moodle asks that all the fractions add up to *exactly* 100. If you simply designate each correct answer with `\item*`, then moodle will divide equally among those answers the points lefts for a sum of 100%. Items that are not given a `fraction` are considered incorrect and selecting them results in negative points such that the sum of all incorrect answers is -100%. For example, the following two examples are equivalent:

```
\begin{multi}[multiple]{my question}
  Which numbers are prime?
  \item[fraction=20] 2
  \item* 5
  \item* 7
  \item[fraction=-10] 1
  \item 6
  \item 8
\end{multi}

\begin{multi}[multiple]{my question}
  Which numbers are prime?
  \item[fraction=20] 2
  \item[fraction=40] 5
  \item[fraction=40] 7
  \item[fraction=-10] 1
  \item[fraction=-45] 6
  \item[fraction=-45] 8
\end{multi}
```

Note that, in this example, negative fractions are set for wrong choices. This prevents students selecting all options to obtain a good grade with no merit.

3.4.3 Numerical

A numerical question in Moodle requires the student to input a real number in decimal form. Its typical format is

```
\begin{numerical}[⟨question options⟩]{⟨question name⟩}
    ⟨question text⟩
    \item[⟨options⟩] ⟨correct answer⟩
\end{numerical}
```

If there is more than one correct answer, additional `\item`'s may be included. Because this is not a multiple choice question, there is no need to provide incorrect answers. There may nevertheless be reasons to include incorrect answers. For example, partially correct answers may be specified by setting the `fraction` key. Feedback for a common mistake may be given by including the incorrect answer like this:

```
\item[fraction=0,feedback={You forgot to antidifferentiate!}] ⟨incorrect answer⟩
```

`tolerance` The `tolerance` key can be used to specify the validity of answers within some margin. This key can be set at different levels: quiz, question, item. For example, with the question

```
\begin{numerical}[tolerance=0.01]{my question}
  Approximate value of $\sqrt{2}$?
  \item[tolerance={1e-1}] 1.4142
  \item[fraction=20,feedback={twice this!}] 7.0711e-1
  \item[fraction=0,feedback={Wrong!}] *
\end{numerical}
```

In this example,

- any answer in the range $[1.4042, 1.4242]$ will be validated,
- any answer in the range $[0.69711, 0.71711]$ will get the specific feedback *twice this!* and 20% of points,
- any other answer is incorrect and will get the specific feedback *Wrong!*.

When feedback is to be given for any non-specified answer, one can add a *last* answer item containing the wilcard character `*` only. In this case, the `tolerance` key is irrelevant.

Both answers and tolerance can be specified with the comma (`,`) as a decimal separator. Exponent notation is accepted. After import, Moodle will recognize indifferently `0.000165`, `0,000165`, `1.65E-4`, `1.65e-4`, `1,65E-4`, and `1,65e-4`.

If the siunitx package is loaded, moodle will detect it and numbers will be rendered nicely in the PDF output.

Units, unit-handling and multipliers are currently unsupported.

3.4.4 Short Answer

A short answer question resembles a numerical question: the student is to fill in a text box with a missing word or phrase.

```
\begin{shortanswer}[⟨question options⟩]{⟨question name⟩}
    ⟨question text⟩
    \item[⟨options⟩] ⟨correct answer⟩
      ⋮
    \item[⟨options⟩] ⟨correct answer⟩
\end{shortanswer}
```

You can make the text box appear as part of the question with the control sequence `\blank`. For example, your question might read

```
\begin{shortanswer}{Leibniz}
  Newton's rival was Gottfried Wilhelm \blank.
  \item Leibniz
  \item Leibniz.
\end{shortanswer}
```

Note that as the blank occurred at the end of a sentence, we included two answers, lest students get the question wrong merely by including or omitting a period.

`case sensitive`
`usecase`

The default setting when creating a Short Answer question in Moodle is to ignore the distinction between upper- and lower-case letters when grading a short answer question. This default is preserved by moodle. You can make a question case-sensitive with the key `case sensitive` or its shorter synonym `usecase`.

The wildcard character `*` can used to grab answers that match a specific pattern. For instance:

- "*Sir Isaac Newton*", "*Isaac Newton*" and "*Newton*" will match the pattern `*Newton`,
- "*Gaston*" and "*Wellington*" will match the pattern `*ton`,
- "*Isaac*" and "*Isaac Newton*" will match the pattern `Isaac*`,
- any non empty answer will match the pattern `*` (wildcard alone).

3.4.5 Essay

Instructors may ask essay questions on a Moodle quiz, although Moodle's software is not up to the task of grading them! Instead each essay question answer must be graded manually by the instructor or a teaching assistant.

```
\begin{essay}[⟨question options⟩]{⟨question name⟩}
    ⟨question text⟩
    \item[⟨options⟩] ⟨notes for grader⟩
        ⋮
    \item[⟨options⟩] ⟨notes for grader⟩
\end{essay}
```

Instead of containing answers, the `\item` tags for the `essay` question contain notes that will appear to whoever is grading the question manually.

`response required` Although Moodle cannot grade the content of an essay question, it can at least determine whether the question has been left blank. If the `response required` key is set, Moodle will insist that the student enter something in the blank before accepting the quiz as completed.

`response format` Moodle offers five different ways for students to enter and/or upload their answers to an essay question. You may choose one of these five options:

`html` An editor with the ability to format HTML responses including markup for italics, boldface, etc. This is the default.

`file` A file picker allowing the student to upload a file, such as a PDF or DOC file, containing the essay.

`html+file` The same HTML editor as above, but with the ability to upload files as well. This permits some students to type answers directly into the web form, and others to compose their essays in another program first.

`text` This editor allows only for entering plain text without any markup.

`monospaced` This yields a plain text editor, without any markup, and with a fixed-width font. This could be useful for entering code snippets, for example.

`response field lines` The key `response field lines` controls the height of the input box. The default is `response field lines=15`.

`attachments allowed` The `attachments allowed` key controls *how many* attachments a student is allowed to upload. Permissible values are `0`, `1`, `2`, `3`, or `unlimited`.

`attachments required` You may also require the student to upload a certain number of attachments by setting `attachments required` to `0`, `1`, `2`, or `3`.

`template` Finally, you may preload the essay question with a template that the student will edit and/or type over, with the key `template={`⟨*template*⟩`}`. The ⟨*template*⟩ should be enclosed in braces.

3.4.6 Matching

A matching question offers a series of subquestions and a set of possible answers from which to choose. If there are m questions and $n \geq m$ possible answers, a matching question will look like this:

```
\begin{matching}[⟨question options⟩]{⟨question name⟩}
    ⟨question text⟩
    \item[⟨options⟩] ⟨question 1⟩ \answer ⟨answer 1⟩
    \item[⟨options⟩] ⟨question 2⟩ \answer ⟨answer 2⟩
      ⋮
    \item[⟨options⟩] ⟨question m⟩ \answer ⟨answer m⟩
    \item[⟨options⟩] \answer ⟨answer m + 1⟩
      ⋮
    \item[⟨options⟩] \answer ⟨answer n⟩
\end{matching}
```

Answers 1 through m correspond to questions 1 through m; answers $m+1$ through n are "decoy" answers. If multiple questions should have the same answer, be sure your typed answer match exactly, so that Moodle will not create duplicate copies of the same answer!

`shuffle` The matching question accepts the option of `shuffle` to randomly permute the questions and answers; by default `shuffle=true`.

`drag and drop` `dd` The standard matching question offered by Moodle corresponds to a dropdown box for choosing the answer to each question. There also exists a "drag and drop matching" plugin for Moodle that shows all questions in one column, all answers in a second column, and allows students to drag the correct answer to the question using a mouse. In this package, to enable drag-and-drop matching, use the key '`drag and drop`' or '`dd`' for short. The default is `dd=false`. If you choose the standard format, then due to the limitations of dropdown boxes, no LaTeX or HTML code can be used in the answers.

3.4.7 Cloze Questions and Subquestions

A "cloze question" has one or more subquestions embedded within a passage of text. For example, you might ask students to fill in several missing words within a sentence, or calculate several coefficients of a polynomial. To encode cloze questions in LaTeX using this package is easy: you simply nest one or more `multi`, `shortanswer`, or `numerical` environments within a `cloze` environment, as in the following example:

```
\begin{cloze}{my cloze question}
  Thanks to calculus, invented by Isaac
  \begin{shortanswer}[usecase]
    \item Newton
  \end{shortanswer},
  we know that the derivative of $x^2$ is
  \begin{multi}[horizontal]
    \item $2x$
    \item* $\frac{1}{3} x^3 + C$
    \item $0$
  \end{multi}
  and that $\int_0^2 x^2\,dx$ equals
  \begin{numerical}
    \item[tolerance={4e-4}] 2.667
  \end{numerical}.
  Thanks, Isaac!
\end{cloze}
```

Note that when used as a subquestion within a cloze question, `\begin{multi}` is *not* followed by name in braces; the same is true for the `shortanswer` and `numerical` environments.

`single=true` `single=false` `multiple` Before Moodle version 3.5, within a cloze question, a multiple choice question was necessarily of type `single`, i.e. with a single good answer. If you intend to export your quiz to Moodle 3.5+, the option `multiple` can be used, when multiple good answers are to be found.

`vertical` `horizontal` `inline` Within a cloze question, by default, a multiple choice question is implemented as an `inline` dropdown box. This is visually compact, but it also prevents the use of mathematical or HTML formatting. Adding the option `vertical` displays the subquestion as a vertical column of radio buttons instead; likewise the option `horizontal` creates a horizontal row of radio buttons. The option `inline` is incompatible with `multiple` or `single=false` (dropdown boxes don't let you pick up several answers!).

`shuffle` Starting from Moodle version 3.0, within a cloze question, the items of a multiple choice question can be shuffled. Setting `shuffle=false` will guarantee that the answer appear in the order they were typed; the default is `shuffle=true`.

`case sensitive` `usecase` Within a cloze question, the short answer question can be made case sensitive. This option, disabled by default, is selected with `case sensitive` or `usecase`.

3.4.8 Description

The Moodle description type is not really a question. It is more like a label. One can set a `feedback` that the student gets when reviewing the submission. Tags can be set as well.

For descriptions, moodle redefines the existing `description` environment.

The syntax for a Description question is as follows:

```
\begin{description}[⟨question options⟩]{⟨question name⟩}
    ⟨question text⟩
\end{description}
```

3.5 Summary of the Key Options

Table 2 summarizes the key options available at the question and answer levels depending on the question type. For the essay questions, please refer to section 3.4.5.

4 Conversion to HTML

Questions should be typed as usual for LaTeX, including `\$` to obtain dollar signs, `$`'s or `\(...\)` for math shifts, `$$`'s or `\[...\]` for display math, et cetera. The package moodle.sty automatically converts this LaTeX code into HTML for web display.

Table 3 lists LaTeX macros, commands, and environments that are specifically converted to HTML. Single and double quotation marks, french quotation marks, inverted exclamation and question marks, and the diacritical commands `\^`, `\'`, `` \` ``, `\"`, `\~`, `\c`, `\H`, `\u` and `\v` are also handled, as are the characters `\aa`, `\ae`, `\l`, `\oe`, `\o`, `\ss`, and their capitalizations. See Tables 6, 7, and 8 for more details.

Table 2: Options offered at the question and answer levels for each question type.

	Question										Answer		
Question type	points	penalty	feedback	tags	shuffle	numbering	multiple	usecase	tolerance	dd	fraction	feedback	tolerance
Multichoice	●	●	●	●	●	●	●				●	●	
Numerical	●	●	●	●					●		●	●	●
Short Answer	●	●	●	●				●			●	●	
Matching	●	●	●	●	●					●	●		
True/False	●		●	●								●	
Description			●	●									
Cloze	●	●	●	●									
Numerical	●								●		●	●	●
Short Answer	●							●			●	●	
Multi (regular)	●				●		●				●	●	
Multi (horizontal)	●				●		●				●	●	
Multi (vertical)	●				●						●	●	

Table 3: Conversion of LaTeX material to HTML.

Macros		Commands	Environnments
`~`	`\#`	`\emph{}`	`\begin{center}`
`\$`	`\&`	`\textbf{}`	`\begin{enumerate}`
`\\`	`\par`	`\textit{}`	`\begin{itemize}`
`\&`	`\S`	`\texttt{}`	`\begin{tikzpicture}`
`\{`	`\}`	`\textsc{}`	
`\␣`	`\relax`	`\underline{}`	
`\,`	`\thinspace`	`\textsuperscript{}`	
`\dots`	`\ldots`	`\up{}`	
`\euro`	`\texteuro`	`\fup{}`	
`\TeX`	`\LaTeX`	`\textsubscript{}`	
`\_`	`\textbackslash`	`\url{}`	
		`\href{}{}`	
		`\tikz[]{}`	
		`\includegraphics[]{}`	
		`\verbatiminput{}`	
		`\VerbatimInput[]{}`	
		`\LVerbatimInput[]{}`	
		`\BVerbatimInput[]{}`	
		`\inputminted[]{}{}`	

In addition, `<` and `>` will be converted to `&lt;` and `&gt;` *within math mode only.* If they should be typed outside of math mode, they will be passed as typed to the HTML, and probably interpreted by students' browsers as HTML tags or other unpredicated results.

Be aware that *moodle does not know how to convert any other TeX or LaTeX commands to HTML.* If other sequences are used, they may be passed verbatim to the XML file or may

`\htmlregister`

cause unpredicted results. The `\htmlregister` command lets you specify the macros that must be expanded in the XML file. It works only when no optional argument is used.

`\moodleregisternewcommands`

When the list of macros is long, it becomes cumbersome to record them individually for expansion. Calling `\moodleregisternewcommands` triggers the automatic expansion of macros defined subsequently using `\newcommand`, `\renewcommand`, `\providecommand`, or their starred variants. Again, this works only if the macros are defined *without* optional argument.

If you think of another LaTeX command that should be changed to an HTML equivalent, please contact the maintainer at `guerquin-kernATcrans.org` so that it may be added to a future revision of the package.

5 Graphics

The moodle package can handle two kinds of graphics seamlessly. External graphics files may be included with the `\includegraphics` command from the `graphicx` package, and graphics may be generated internally using Ti*k*Z. In either case, the graphics will be embedded in base-64 encoding directly within the Moodle XML produced. This prevents the hassle of managing separate graphics files on the Moodle server, as Moodle will store the picture within the question in the question bank.

5.1 Default `includegraphics`

`\includegraphics`
`height`
`width`

When using `\includegraphics`, the only options currently supported are `height` and `width`. Attempts to use other `\includegraphics` options, such as `scale` or `angle`, will affect the PDF but not the XML output. The dimensions set by `height` and `width` are TeX dimensions such as `4 in` or `2.3 cm`. In order to prepare the image for web viewing, this

`ppi`

package converts those dimensions to pixels using a default of 103 pixels per inch.[1] That value may be changed by setting the `ppi` key (e.g., `ppi=72`); this is probably best done for the entire document with a `\moodleset` command, rather than image-by-image.

`\graphicspath`

You can use `\graphicspath{{`*path*`}}` to specify a directory where the pictures to be included are located.

A special rule was added for the inclusion of GIF pictures (`.gif` extension). These files are passed as-is to the XML, preserving potential animations. However, as pdfTeX engines do not support the GIF format, the picture is passed to the PDF output after a conversion to the PNG format. When the GIF file is animated, only its first frame is passed to the PDF.

[1] This number was selected because an image with `<IMG HEIGHT=103 WIDTH=103 SRC="...">` showed up as almost exactly 1 inch tall and 1 inch wide on several of this author's devices and browsers as of January 2016.

5.2 Ti*k*Z Pictures

When Ti*k*Z is loaded and used to define pictures, moodle invokes the `external` Ti*k*Z library, so that each `tikzpicture` environment is compiled to a freestanding PDF file.

5.3 Package Option `tikz`

`tikz` The moodle package admits a `tikz` option which has the following effects:

- the package `tikz` is loaded.
- `includegraphics` is embedded in a Ti*k*Z picture. Consequences are that
  - the pictures encoded in the XML file are resampled. This prevents encoding images at a higher resolution than rendered by Moodle.
  - the full set of `includegraphics` options is accessible, e.g. `scale=.5`, `angle=90`, or `width=.2\textwidth`.
- `\embedaspict` a macro `\embedaspict{...}` is provided for the inclusion of inline LaTeX material as images. This can serve as a workaround to overcome limitations of this package—like the conversion of tabulars to HTML— or limitations of Moodle itself. For the definition of this macro, the package `varwidth` is loaded.
- optimizations of the Ti*k*Z-external library are disabled. Compilation might get sensibly slower.

5.4 External Tools

The mechanisms used for handling graphics are somewhat fragile and rely upon three free external programs.

1. GhostScript (`www.ghostscript.com`) is used to convert the PDF output from Ti*k*Z into a PNG raster graphics file. The default command line is presumed to be `gswin64c.exe` (if `\ifwindows` from the `ifplatform` package returns true) or `gs` (if `\ifwindows` returns false). If your system requires a different command line to invoke Ghostscript, you may change it by invoking: `\ghostscriptcommand`

   `\ghostscriptcommand{`⟨*executable filename*⟩`}`

2. When external graphics files such as PDF are included, the open-source ImageMagick software (`www.imagemagick.org`) converts each file to PNG format. The command line for ImageMagick is the nondescript word `convert`, but may be changed by invoking `\imagemagickcommand{`⟨*executable filename*⟩`}`. `\imagemagickcommand`
3. OptiPNG (`http://optipng.sourceforge.net/`) is used to optimize the PNG images. The command line is presumed to be `optipng`, but can be changed with `\optipngcommand{`⟨*executable filename*⟩`}`. `\optipngcommand`

Please note the following vital points to make the graphics handling work:

- As of now, graphics are only supported when compiling directly to a PDF with `pdflatex`. Including PS graphics or using Ti*k*Z with the DVI→PS workflow is not yet supported.

- You must have Ghostscript and ImageMagick installed on your system to fully use the graphics-handling capabilities of moodle.
- If OptiPNG is not installed, the corresponding system calls will fail with otherwise no impact on the compilation process: PNG files are passed unoptimized to the XML output.
- LaTeX must be able to call system commands; that is, `\write18` must be enabled. For MikTeX, this means adding `--enable-write18` to the command line of `pdflatex`; for TeXLive, this means adding `--shell-escape=true`.
- Due to security issues with old versions of Ghostscript, some systems default to a policy that prevents the conversion of PDF and PS to PNG. Assuming that, as a user of moodle which requires shell escape capabilities, you either use a sandboxed environment or trust the files handled at the system-level, you may want to disable this over-zelous security policy. For example, see this.
- Users of the `circuitikz` package must enclose their circuits' Ti*k*Z code in the `tikzpicture` environment instead of `circuitikz`. That is required, as of Ti*k*Z 2.1, by the `external` library.

5.5 Package Option `svg`

Important Notice *The svg option is an experimental feature introduced in moodle v0.8. It has been tested exclusively under Linux, with TeXLive 2020, Inkscape v1.0.1 and Scour 0.38.2.*

`svg` The moodle package admits an experimental `svg` option which has the following effects:

- `\includegraphics` can be used to import SVG graphic files directly (extension `.svg` or `.SVG`). In this case, the SVG file is passed as-is to the XML output and is converted using Inkscape (must be installed) for inclusion in the PDF output.
- the graphic files in PDF format are converted to the SVG format using Inkscape (must be installed), rather than beeing rasterized. Before inclusion to the XML output, the SVG file is optimized using the Scour utility. This optimization step is optional in the sense that, if the Scour call fails, the unoptimized SVG file will be passed to the XML output. Two processes benefit from this PDF→SVG conversion:
  - inclusion of PDF graphics with `\includegraphics`, and
  - Tikz pictures that are externalized.

`\PDFtoSVGcommand`
`\SVGtoPDFcommand`
`\optiSVGcommand`

The call of external tools can be changed using the macros `\PDFtoSVGcommand{`⟨...⟩`}`, `\SVGtoPDFcommand{`⟨...⟩`}` and `\optiSVGcommand{`⟨...⟩`}`.

6 Verbatim Code

Because, for HTML translation, moodle parses the body of questions, the use of verbatim code results in compilation errors. This is why the use of `\verb`, `\begin{verbatim}` and other standard utilities is not supported.

However, using the following three utilities, verbatim code can be imported from an external file:

Table 4: Options and corresponding values considered for XML generation of verbatim material with `VerbatimInput` and `inputminted`.

Option keys	Possible values
`gobble`	⟨*integer*⟩
`autogobble`[1]	`true` or `false`
`tabsize`	⟨*integer*⟩
`numbers`	`none`, `left`, `right`, or `both`[2]
`firstnumber`	`auto`, `last`, or ⟨*integer*⟩
`firstline`	⟨*integer*⟩
`lastline`	⟨*integer*⟩
`numberblanklines`	`true` or `false`
`highlightlines`[2]	`{`⟨*coma-separated list of integers or ranges*⟩`}`
`style`[1]	⟨*string*⟩

[1] `autogobble`, `numbers=both`, and `style` are offered only by minted.

[2] line highlighting is offered only with fvextra or minted loaded.

`\verbatiminput`

1. `\verbatiminput{`⟨*filename*⟩`}` from the verbatim package inserts verbatim code in both the PDF and the XML for moodle, without fancy additions.

`\VerbatimInput` `\BVerbatimInput` `\LVerbatimInput`

2. The macro `\VerbatimInput{`⟨*options*⟩`}{`⟨*filename*⟩`}` from fancyvrb or fvextra does more, with several options and settings offered (see below). The variants `\VBerbatimInput` and `\LVerbatimInput` are also supported, with identical effect on the XML output. The variants with a star are unsupported and result in errors when used.

`\inputminted`

3. On top of that `\inputminted[`⟨*options*⟩`]{`⟨*lang*⟩`} {`⟨*filename*⟩`}`from the minted package offers syntaxic highlighting tailored to the specified language.

The moodle package handles these three commands to pass the code in the output XML.

With `\VerbatimInput` and `\inputminted`, the options that are taken care of for XML generation are listed in Table 4. Using `\fvset{`⟨*key=value,...*⟩`}`, options can be set globally. Equivalently, with minted, `\setminted[`⟨*lang*⟩`]{`⟨*key=value,...*⟩`}` is available.

In order to define the verbatim code from the LaTeX document itself, it is still possible to use, outside the scope of the moodle questions, the environments `filecontents*` (from the filecontents package or LaTeX kernel itself since 2019) or `VerbatimOut` (from the fancyvrb and fvextra packages).

When code decorated with left-side line numbers is placed in question items, the output PDF could show a collision between numbers of the item and the first line. To avoid this, `\LVerbatimInput` or `\BVerbatimInput` can be used. Instead, when minted is used, the "left-right" mode can be enforced with the LaTeX command:

```
\RecustomVerbatimEnvironment{Verbatim}{LVerbatim}{}
```

When using utilies from fancyvrb, fvextra, or minted, moodle sets framing options for the display of code in the output PDF:

```
\fvset{frame=lines,label={[Beginning of code]End of code},
       framesep=3mm,numbersep=9pt}
```

These settings can be overidden using `\fvset` after the preamble.

7 Calculated Questions

Moodle's calculated questions are not supported by this package.

However, as demonstrated by A. Vohns, an advanced scripting language may be used to generate a series of questions sharing the same prototype.

We suggest to apply a specific tag to these questions. After import in Moodle, when creating a quizz, this tag can be selected to narrow down a random selection of question. This would mimic the behavior of calculated questions while bringing the flexibility of your favorite scripting language.

Here are two examples inspired from the work of A. Wohns. The first one relies on the native Lua capabilities of LuaLaTeX.

```
\begin{quiz}[tags={calculated}]{Example Quiz}
\directlua{

  function clozenum_print(pair,op,result)
    tex.print("\\begin{numerical}$"..pair[1].." "..op.." "..pair[2].."
    =$".."\\item ",result,"\\end{numerical}")
  end
  function cloze_print(pair,points)
    tex.print("\\begin{cloze}[points="..points.."]{Arithmetic Quiz
    ("..pair[1]..", "..pair[2]..")}Solve the following tasks!\\\\")
    clozenum_print(pair,"+",pair[1]+pair[2])
    clozenum_print(pair,"-",pair[1]-pair[2])
    clozenum_print(pair,"*",pair[1]*pair[2])
    if pair[1]/pair[2]==math.floor(pair[1]/pair[2]) then
      clozenum_print(pair,":",math.floor(pair[1]/pair[2]))
    end
    tex.print("\\end{cloze}")
  end
  for x = 2,4 do
    for y = 2,4 do
      if x>y then
        if x/y==math.floor(x/y) then points=4 else points=3 end
        cloze_print({x,y},points)
      end
    end
  end

}
\end{quiz}
```

The second example makes use of the python package (\usepackage{python}).

```
\begin{quiz}[tags={calculated}]{Example Quiz}
\begin{python}

  def clozenum_print(pair,op,result):
    print(rf"""\begin{{numerical}}
  ${pair[0]} {op} {pair[1]} =$\item {result}
  \end{{numerical}}""")
  def cloze_print(pair,points):
    print(rf"""\begin{{cloze}}[points={points}]{{Arithmetic Quiz
    {(pair[0],pair[1])}}}Solve the following tasks!\\""")
    clozenum_print([x,y],"+",x+y)
    clozenum_print([x,y],"-",x-y)
    clozenum_print([x,y],"*",x*y)
    if pair[0]/pair[1] == pair[0]//pair[1]:
      clozenum_print([x,y],":",x//y)
    print("\end{cloze}")
  for x in range(2,5):
    for y in range(2,5):
      if x > y:
        if x/y == x//y:
          points=4
        else:
          points=3
        cloze_print([x,y],points)

\end{python}
\end{quiz}
```

These two codes yield the same XML content.

8 Known Limitations and Call for Bug Reports

Table 5 lists some different features supported, limitations, and bugs.

Tables 6, 7, and 8 describe the current support for some special characters, accents and other diacritical marks.

Some features of Moodle quizzes have not yet been implemented in moodle. Here is a non-exhaustive list.

- Moodle's feature of designating feedback for correct, partially correct, and incorrect answers.
- Hints
- Multiple keywords (tags) for questions

The authors have used this package together with a limited number of colleagues for a few semesters of teaching. If other users adopt this package, we fully expect them to find bugs. *Please* send all bugs you find to `guerquin-kern@crans.org`, so that we can fix them for subsequent versions.

9 Compatibility

This package has been originally written for and tested with the implementation of Moodle 2.9 run by Moodlerooms for St. Norbert College in January 2016. Since then, it has been successfully combined with Moodle 3.5. Future versions of this package might include some support for specifying your version of Moodle in the `.tex` file to help ensure compatibility.

Table 5: Content enrichment (pictures, equations) support after XML import in Moodle v3.5.7, depending on the question type.

	XML rendering in...		
Question type	Question	Answer	Feedback
Multichoice	yes	yes	yes
Numerical	yes	no[1]	yes
Short Answer	yes	no[1]	yes
Matching (std)	yes	no[2]	no[3]
Matching (dd)	yes	yes[4]	no[3]
Essay	yes	yes[5,6]	yes[5]
True/False	yes	no	yes
Description	yes	∅	yes
Cloze	yes	∅	∅
Numerical	yes	no[1]	yes
Short Answer	yes	no[1]	yes[7]
Multi (regular)	yes	no[2]	yes[7]
Multi (horizontal)	yes	yes	yes
Multi (vertical)	yes	yes	yes

[1] Moodle prompts the student for an answer and then compares it to the solutions provided. This is text-only.

[2] Moodle uses a dropdown list to let one choose among the possible answers. This forbids either picture inclusion and LaTeX rendering.

[3] Not supported by Moodle (in this context, answer-specific feedback represents lots of possible combinations).

[4] The drag-and-drop-matching plugin seems broken before version 1.6 20190409. Moodle's XML import fails with a dmlwriteexception when the field content exceeds few hundreds characters. This prevents the inclusion of most base64 images and maybe some complicated equations.

[5] For this question type and in the context of XML generation, the Answer column represents the "template" while the Feedback column represents the "notes for the grader". Obviously, the grading process is not automatic and there is no answer-specific feedback.

[6] Picture and LaTeX rendering could be done, but only after submission and only if the keyval "response format" is set to "html".

[7] Moodle only reveals the feedback when hovering the checkmark or X mark with the mouse.

Table 6: Support for diacritical marks in a UTF8-coded TeX document compiled with (pdf)LaTeX (packages `inputenc` with option `utf8` and `fontenc` with option `T1`), LuaLaTeX or XeLaTeX (package `fontspec`).

Input type				Engine support	
Unicode		LaTeX		XeLaTeX or LuaLaTeX	(pdf)LaTeX
å	Å	`\aa`	`\AA`	Unicode and LaTeX	Unicode and LaTeX
à	À	`` \`a ``	`` \`A ``	Unicode and LaTeX	Unicode and LaTeX
â	Â	`\^a`	`\^A`	Unicode and LaTeX	Unicode and LaTeX
ã	Ã	`\~a`	`\~A`	Unicode and LaTeX	Unicode and LaTeX
é	É	`\'e`	`\'E`	Unicode and LaTeX	Unicode and LaTeX
è	È	`` \`e ``	`` \`E ``	Unicode and LaTeX	Unicode and LaTeX
ë	Ë	`\"e`	`\"E`	Unicode and LaTeX	Unicode and LaTeX
ê	Ê	`\^e`	`\^E`	Unicode and LaTeX	Unicode and LaTeX
î	Î	`\^i`	`\^I`	Unicode and LaTeX	Unicode and LaTeX
ï	Ï	`\"i`	`\"I`	Unicode and LaTeX	Unicode and LaTeX
ñ	Ñ	`\~n`	`\~N`	Unicode and LaTeX	Unicode and LaTeX
õ	Õ	`\~o`	`\~O`	Unicode and LaTeX	Unicode and LaTeX
ö	Ö	`\"o`	`\"O`	Unicode and LaTeX	Unicode and LaTeX
ô	Ô	`\^o`	`\^O`	Unicode and LaTeX	Unicode and LaTeX
ù	Ù	`` \`u ``	`` \`U ``	Unicode and LaTeX	Unicode and LaTeX
ü	Ü	`\"u`	`\"U`	Unicode and LaTeX	Unicode and LaTeX
û	Û	`\^u`	`\^U`	Unicode and LaTeX	Unicode and LaTeX
ç	Ç	`\c{c}`	`\c{C}`	Unicode and LaTeX	Unicode and LaTeX
ş	Ş	`\c{s}`	`\c{S}`	Unicode and LaTeX	LaTeX only
ţ	Ţ	`\c{t}`	`\c{T}`	Unicode and LaTeX	LaTeX only
ő	Ő	`\H{o}`	`\H{O}`	Unicode and LaTeX	LaTeX only
ű	Ű	`\H{u}`	`\H{U}`	Unicode and LaTeX	LaTeX only
ÿ	Ÿ	`\"y`	`\"Y`	Unicode and LaTeX	LaTeXonly
ă	Ă	`\u{a}`	`\u{A}`	Unicode and LaTeX	LaTeX only
ĕ	Ĕ	`\u{e}`	`\u{E}`	Unicode and LaTeX	LaTeX only
ğ	Ğ	`\u{g}`	`\u{G}`	Unicode and LaTeX	LaTeX only
ĭ	Ĭ	`\u{\i}`	`\u{I}`	Unicode and LaTeX	LaTeX only[1]
ŏ	Ŏ	`\u{o}`	`\u{O}`	Unicode and LaTeX	LaTeX only
č	Č	`\v{c}`	`\v{C}`	Unicode and LaTeX	LaTeX only
ď	Ď	`\v{d}`	`\v{D}`	Unicode and LaTeX	LaTeX only
ě	Ě	`\v{e}`	`\v{E}`	Unicode and LaTeX	LaTeX only
ľ	Ľ	`\v{l}`	`\v{L}`	Unicode and LaTeX	LaTeX only
ň	Ň	`\v{n}`	`\v{N}`	Unicode and LaTeX	LaTeX only
ř	Ř	`\v{r}`	`\v{R}`	Unicode and LaTeX	LaTeX only
š	Š	`\v{s}`	`\v{S}`	Unicode and LaTeX	LaTeX only
ť	Ť	`\v{t}`	`\v{T}`	Unicode and LaTeX	LaTeX only
ž	Ž	`\v{z}`	`\v{Z}`	Unicode and LaTeX	LaTeX only

[1] XeLaTeX renders correctly `\u{i}`, that is, without a superscript dot. Instead, with (pdf)LaTeX the rendering of `\u{i}` is flawed by the superposition of the superscript dot and the breve diacritical mark. Both engines render `\u{\i}` as expected.

Table 7: Support for ligatures in a UTF8-coded TEX document compiled with (pdf)LATEX (packages `inputenc` with option `utf8`, `fontenc` with option `T1`), LuaLATEX or XƎLATEX (package `fontspec`).

Input type				Engine support	
Unicode		LATEX		XƎLATEX or LuaLATEX	(pdf)LATEX
æ	Æ	`\ae`	`\AE`	Unicode and LATEX	LATEX only
œ	Œ	`\oe`	`\OE`	Unicode and LATEX	LATEX only
ß	ẞ	`\ss`	`\SS`	Unicode and LATEX[1]	LATEX only[2]

[1] the Libertine font, used in this documentation and available for instance via the package `libertine`, defines the glyph ẞ. Most fonts do not define this glyph.

[2] LATEX defines the `\SS` macro but (pdf)LATEX renders it as a doubled capital S.

Table 8: Support for other glyphs and punctuation marks in a UTF8-coded TEX document compiled with (pdf)LATEX (packages `inputenc` with option `utf8`, `fontenc` with option `T1`), LuaLATEX or XƎLATEX (package `fontspec`).

Input type				Engine support	
Unicode		LATEX		XƎLATEX or LuaLATEX	(pdf)LATEX
ł	Ł	`\l`	`\L`	Unicode and LATEX	LATEX only
ø	Ø	`\o`	`\O`	Unicode and LATEX	LATEX only
«		`\guillemotleft`[1]		Unicode and LATEX	LATEX only
»		`\guillemotright`[1]		Unicode and LATEX	LATEX only
¿		`\textquestiondown`		Unicode and LATEX	LATEX only
¡		`\textexclamdown`		Unicode and LATEX	LATEX only

[1] the package `babel` loaded with option `french` defines `\og` and `\fg` for the same symbols. These are also supported by moodle.

As the ultimate purpose of this package is the generation of XML files, future versions of moodle will attempt to maintain backwards compatability with earlier versions regarding the XML output, apart from bug fixes. Backwards compatibility of the PDF output is not yet guaranteed, however, in case the author or users discover better ways for the PDF to display the underlying XML data to be proofread.

In other words, compiling your current `.tex` file with a future version of moodle should produce the same XML file it does now (apart from bug fixes), but it might produce a more informative, and hence different, PDF output.

10 Unrelated Tip: Quality of Moodle TEX Images

This has nothing to do with moodle, but is a Frequently Asked Question in is own right. On some servers, at least, Moodle's default "TEX Filter" for displaying mathematical notation is of abysmally poor quality, rending mathematics as low-resolution PNG's. One solution that has worked for me is to go to "Course Administration → Filters," turn "TEX Notation" *off*, but turn "MathJax" *on*. This forces TEX code to be rendered by MathJax instead of Moodle, producing much higher-quality results.

11 Version History

0.8 Matthieu Guerquin-Kern (2021-01-04)

Added

- Support for inclusion of GIF pictures.
- Added package option `svg` to avoid the rasterization of vector graphics.
- New macro `\setsubcategory` to define subcategories, reflected in PDF and XML.
- Package option `handout` for sharing PDF with students.
- Extensions can be omitted when including pictures.
- Description question type.
- LuaLATEX is now supported (and recommended for UTF8-coded sources).
- Examples of ways to reproduce the behavior of calculated questions.
- Command to trigger the automatic recording of new commands.
- Mechanism to match `fraction` key to values accepted by Moodle.
- A `fractiontol` key to control the tolerance in this mechanism.
- Support for inverted punctuation marks ¿ and `¡`.
- Support for `\_` and `\textbackslash`.
- Support for the wildcard character as an answer in numerical questions.

Changed

- Template of Essay questions is now shown in PDF.
- The macro `\setcategory` is reflected by a new section in PDF.
- In matching questions, warnings are raised if the number of items is insufficient.

- Improved display of matching questions in PDF.
- The package iftex is now required.
- An error is thrown when `fraction` is set to an invalid value.
- In numerical questions, the tolerance can be set in exponent form.
- Nicer PDF rendering of numbers in numerical questions if siunitx is loaded.
- Included PNG and JPEG files are now directly converted to base64.

Fixed

- TeX's inline math (`$...$`) can now contain escaped dollar signs (`\$`).
- Closing braces escaped in cloze subquestions outside math environment.
- The scope of the `tolerance` key is now respected.

0.7 Matthieu Guerquin-Kern (2020-09-06)

Added

- Support for inclusion of verbatim code.
- Package option `tikz`.
- Support for `\"Y` and `\"y`.
- New commands converted to XML.
- Adding a stamp comment in XML, package option offered to disable this behavior.
- Support for the `\tikz` command.
- A different directory can be specified for picture inclusion.
- Warn user of the `babel` package set for french that autospacing must be deactivated.
- Square bracket math delimiters are recognized and converted properly.
- Support of breve and caron diacritical marks.

Changed

- In multi with multiple answers allowed, choosing all options no longer results in a good grade. An automatic penalty mechanism is introduced. Can be overridden by manually setting fractions.

Removed

- Irrelevant `penalty` tag in cloze subquestions.

Fixed

- Non-integer fractions can now be specified in cloze subquestions.
- Signifiantly squeeze PNG images size by skipping ancillary data.
- Enumerate or itemize environment can now be nested in question items.
- Several pictures can be included in a question without being mixed in the XML file.
- management and rendering of fraction in questions.

- Correctly handling a LaTeX starting the last item of a question.
- Closing braces escaped in cloze subquestions. This allows LaTeX equations or images to be included.
- Image inclusion with MacOS.

0.6b Matthieu Guerquin-Kern (2019-11-27)

Added

- New package options to set section or subsection at the quiz level.
- True/False question type is now supported.
- Moodle tags can now be specified for questions (and rendered in PDF as well).
- In cloze questions, the `multiresponse` subquestion type is now supported.

Removed

- External dependency on `OpenSSL`.
- Irrelevant tags were written in XML for matching questions.

Fixed

- Ti*k*Z externalization now works when using XeLaTeX.
- It is now possible to set points manually among several correct answers in multichoice questions.
- General feedbacks can now contain backslashes.
- Several quizzes can now be defined in a single source file, each specifying a category for Moodle's question bank.
- Correct encoding information in now written in XML depending on the LaTeX compiler used.

0.6a Matthieu Guerquin-Kern (2019-06-21)

Added

- XeLaTeX is now recommended when using UTF8-encoded sources (support of accents).
- Feedbacks are now displayed in the PDF file produced.
- The `optipng` utility is used (and required) to reduce the size of images embedded in the XML file.
- Question options and settings are now displayed in the PDF file
- Supporting more LaTeX macros for symbols and accents (mostly diacritical marks and ligatures).
- Introduce shuffle options in cloze-multi subquestions.
- Package option `final`.

Changed

- In draft mode, Ti*k*Z externalization in no longer triggered.

Fixed

- In the different question types, the feedback fields are now converted for HTML allowing LaTeX equation and images.
- Documentation improvements (limitations and previously undocumented features).

0.5 Anders O.F. Hendrickson (2016-01-05) — Initial version