

The moodle package: generating MOODLE quizzes via L^AT_EX*

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

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

January 28, 2023

Abstract

This document describes the moodle package, made for writing MOODLE quizzes in L^AT_EX. In addition to typesetting the quizzes for proofreading or giving to students as handout, the package generates an XML file to be uploaded to a MOODLE server.

Contents

1	Introduction	3
1.1	Motivation	3
1.2	Workflow	4
2	Usage	5
2.1	Example Document	5
2.2	Package Options	6
2.3	Quiz and Question Environments	7
2.4	Question Types	9
2.4.1	True/False	9
2.4.2	Multiple Choice	9

*This document corresponds to moodle.sty v1.0, dated 2023/01/28.

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

[‡]current maintainer, author of the updates (v0.6 to v1.0).

2.4.3	Numerical	12
2.4.4	Short Answer	13
2.4.5	Essay	14
2.4.6	Matching	15
2.4.7	Cloze Questions and Subquestions	16
2.4.8	Description	18
2.5	Summary of the Key Options	18
2.6	Automated Generation of Questions	18
2.6.1	Script Generating T _E X Code	19
2.6.2	T _E X Code As a Template	20
3	Conversion to HTML	23
3.1	Level of Support	23
3.2	Graphics	27
3.2.1	Default <code>includegraphics</code>	27
3.2.2	TikZ Pictures	28
3.2.3	Package Option <code>tikz</code>	28
3.2.4	External Tools	28
3.2.5	Package Option <code>svg</code>	29
3.3	Verbatim Code	30
4	Internationalization	33
5	Package Development	35
5.1	Feature Requests, Bug Reports, and Contributions	35
5.2	Known Limitations	35
5.3	Compatibility	37
5.4	Unrelated Tip: Quality of MOODLE T _E X Images	37
	Version History	43
	Index	46

1 Introduction

1.1 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 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 \mathbb{R} instead of $\backslash\mathrm{mathbb}\{R\}$ or as complex as macros that generate entire paragraphs of text. The MOODLE editor, by contrast, requires you to type $\backslash\mathrm{mathbb}\{R\}$ 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 TikZ, 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 letters and roman font) while the \LaTeX package is referred to as moodle (lower case and sans serif font).

1.2 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 `pdf \LaTeX` (ASCII characters only), `X \LaTeX` , or `Lua \LaTeX` . This will also produce the file `(jobname)-moodle.xml`.
3. navigate to your course on MOODLE 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.

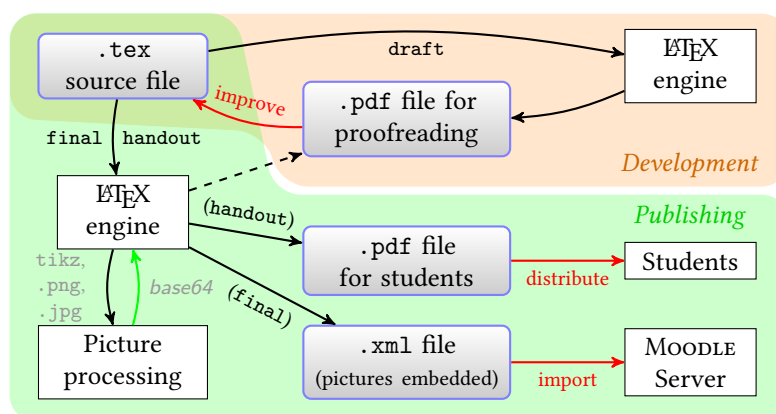


Figure 1: Block diagram describing a typical workflow using the moodle package.

2 Usage

The following pages presume the reader already has some familiarity with creating and editing MOODLE quizzes through the web interface. Users that are not familiar with MOODLE quizzes can learn more in the MOODLE documentation. For instance, https://docs.moodle.org/en/Question_types.

The xkeyval package is used to provide a key-value interface.

2.1 Example Document

Here is a simple example document:

```
\documentclass[12pt]{article}
\usepackage[section]{moodle}
\moodleregisternewcommands
\newcommand\monomial[1]{x^{#1}}
\newcommand\sillyanswer{What!?!}
\begin{document}
Quiz generated \LaTeX's \textsf{moodle} (\moodleversion, \moodledate).
Import the derived file \texttt{\jobname-moodle.xml} on Moodle.
\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`.

2.2 Package Options

- `draft` (*Opt*) [moodle] 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`.
- `final` (*Opt*) [moodle]
- `handout` (*Opt*) [moodle] 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 seed of its random generator is controlled by the macro `\RLsetrandomseed{<integer>}`. This option does not interfere with the generation of the XML file.
- `\RLsetrandomseed` [randomlist]
- `samepage` (*Opt*) [moodle] If the package option `samepage` is invoked, preferably used together with `handout` (`\usepackage[handout,samepage]{moodle}`), the package will try to keep every question on the same page. Very bad spacing can result from this. This option is experimental. Subquestions inside a `cloze` question are protected but the `cloze` question itself is not protected.
- `nostamp` (*Opt*) [moodle] 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 platform used and the package version. For instance:
- ```
<!-- This XML file is a question bank made for Moodle. -->
<!-- It was generated on 2023-01-28 22:13 by LuaLaTeX running -->
<!-- on Linux with the package moodle v1.0 (2023/01/28) -->
```
- The package option `nostamp` prevents this stamp from being written in the XML file.
- `section` (*Opt*) [moodle]  
`subsection` (*Opt*) [moodle]  
`section*` (*Opt*) [moodle]  
`subsection*` (*Opt*) [moodle]
- The package options `section` and `subsection` place each quiz as a new section or subsection, respectively. Starred variants correspond to unnumbered sections or subsections. To preserve compatibility with Version 0.5 of this package, the default is `subsection*`. Consequently, `\usepackage[subsection*]{moodle}` is equivalent to `\usepackage{moodle}`.
- `tikz` (*Opt*) [moodle] The package option `tikz` is described in [Section 3.2.3 on page 28](#).
- `svg` (*Opt*) [moodle] The package option `svg` is described in [Section 3.2.5 on page 29](#).
- LMS (*Opt*) [moodle]  
Default: `warn only`
- `=<X.Y>` lets you specify version numbers for the target MOODLE LMS instance (`X` and `Y` are major and minor version integers). When version numbers are provided, the use of recent quiz features is secured by a compatibility check and moodle raises relevant errors. The XML stamp (see `nostamp` above) also mentions MOODLE's target version. By default, moodle will just issue warnings when recent quiz features are used.

## 2.3 Quiz and Question Environments

A `.tex` document to generate MOODLE quizzes contains one or more quiz environments.

`quiz (env)` [moodle] [*⟨common options⟩*]{*⟨category name⟩*} defines a quiz, within which various question environments are nested. The mandatory argument, *⟨category name⟩*, names a category for MOODLE’s “question bank”: after import, the questions defined in this environment will be gathered in this category. Using the optional argument, options can be set at the quiz level. Although there are no quiz-specific options, any *⟨common options⟩* set with the quiz will be inherited by all questions contained within that environment.

`\moodleaset` [moodle] {*⟨options⟩*} is to be used to set options outside question environments; the option settings are local to  $\TeX$ -groups.

`\setcategory` [moodle] {*⟨category name⟩*} is to be used to change the current category inside a quiz environment and in between questions. Note that the quiz environment defines a category by its own.

`\setsubcategory` [moodle] {*⟨subcategory name⟩*} does the same with subcategories.

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 (Key)` [question] Default: 1 By default, each question is worth 1 point on the quiz. This setting may be changed with the `points` key or its synonym, `default grade`. For example, `points=2` makes that question worth two points.

`penalty (Key)` [question] Default: 0.1 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.

`fraction (Key)` [answer] 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 on the next page](#), from 0 (entirely wrong) to 100 (entirely correct).

In questions where several choices can be selected (see `multi` with option `multiple`), positive fractions must add up to exactly 100. It is also possible to set negative fractions

Table 1: Admissible positive values for the fraction key outside cloze environments:  $100 \cdot (p/q)$ .

| 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      |          |     |     |          |     |     |     |     |

(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 (Key) [answer]` 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. The default value, 0.01, may be changed. When no admissible fraction value is matched, the package raises an error.  
**Default: 0.01**

`feedback (Key)` 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.

`feedback (Key) [question]` If the `feedback` key is set for a question, then that feedback will appear to each student regardless of the student's answer.

`feedback (Key) [answer]` Answer-specific feedback (perhaps explaining a common mistake) may also be given by setting the `feedback` key *at the individual answer*.

`tags (Key)` The `tags` key sets a list of keywords 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 specified in between braces. Multiple tags can be set as a comma-separated list:



```
tags={tag1,tag2,{ leading whitespace},{including, comma}}
```

- `tags (Key)` [quiz] If the `tags` key is set at the quiz level, then that tag list will serve as a default for each question of the quiz.
- `tags (Key)` [question] Question-specific tags can be assigned by setting the `tags` key *at the question level*. The question-level `tags` key overrides eventual quiz-level tags.
- Users willing to specify a same tag for all questions of the quiz could also consider relying on MOODLE's category mechanism.

## 2.4 Question Types

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

### 2.4.1 True/False

- `truefalse (env)` [quiz] [`<question options>`]{`<question name>`} is an environment that defines a *True/False* question.

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.

### 2.4.2 Multiple Choice

- `multi (env)` [quiz] [`<question options>`]{`<question name>`} is an environment that defines a *Multiple Choice*

Table 2: Numbering modalities offered for `multi` questions.

| Sample        | a., b., ... | A., B., ... | 1., 2., ... | i., ii., ... | I., II., ... | •    |
|---------------|-------------|-------------|-------------|--------------|--------------|------|
| TeX syntax    | alph        | Alph        | arabic      | roman        | Roman        | none |
| MOODLE syntax | abc         | ABCD        | 123         | iii          | IIII         | none |

question.

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

```

\begin{multi}[\langle question options \rangle]{\langle question name \rangle}
 \langle question text \rangle
 \item[\langle answer options \rangle]* \langle correct answer \rangle
 \item[\langle answer options \rangle] \langle wrong answer \rangle
 :
 \item[\langle answer options \rangle] \langle wrong answer \rangle
\end{multi}

```

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

- `shuffle` (*bool*) [multi]  
Default: `true`    The Boolean key `shuffle` determines whether MOODLE will rearrange the possible answers in a random order. Setting `shuffle=false` will guarantee that the answers will be presented to the student in the order they were typed.
- `numbering` (*Key*) [multi]  
Default: `abc`    MOODLE offers different options for numbering the possible answers. You may set the numbering key to any of the values listed in Table 2. `numbering=none` produces an unnumbered list of answers. Note the *four* capital letters required by MOODLE's syntax to obtain upper-case Roman or alphabetic numerals.
- `fraction` (*Key*) [item]  
Default: `0`    The `fraction` key may be automatically set depending on the presence of an asterisk `*` right after the `\item`. By default, Starred items designate correct answers (`fraction=100`) while bare items designate incorrect answers (`fraction=0`). The key can be used to designate some wrong answers as being worth partial credit or sanction. For example, a question might read
- `fraction` (*Key*) [item\*]  
Default: `100`

---

```

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

```

---

- `single` (*bool*) [multi]  
Default: `true`    By default, the `multi` environment produces a multiple choice question where only one answer can be selected. This is called `single` mode.

In *single* mode, the teacher can set negative fractions for incorrect choices. This way, the expected grade of a student picking up choices randomly across a quiz is lowered. By default, incorrect answers that do not have a fraction specified correspond to `fraction=0`.

`sanction (Key) [multi]`  
 Default: 0

However, if the `sanction` key is set to a positive value, at quiz or question levels, such incorrect answers are sanctioned as if `fraction=-<sanction value>` was set for each of them.

`multiple (Key) [multi]`

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`. For this kind of question, the student gets a grade that corresponds to the total of the weights of the answers selected. Users may rely on two modalities to set the weights of the answers:

1. the *advanced mode* applies whenever a `fraction` key is set among the proposed answers. In this case, it is recommended that the user sets manually everything. Answers that are left with no fraction set are considered as neutral, selecting them will not change the grade. If `\item*` is used to designate some correct answers, moodle will distribute equally among those answers the points that are left for a total of 100% (only positive fractions count here). Note that doing so will easily yield situations where the weights are inadmissible (see Table 1 on page 8).
2. the *automatic mode* applies when correct answers are designated using `\item*` and no answer has a `fraction` key set. In this case, each correct answer is weighted to bring the same fraction of the maximum grade and each incorrect answer is weighted to cancel the benefits of the selection of one correct answer.

For example, the following two examples are equivalent:

---

```

\begin{multi}[multiple]{Automatic Mode}
 Which numbers are prime?
 \item 2
 \item* 5
 \item* 7
 \item 1
 \item 6
\end{multi}
\begin{multi}[multiple]{Advanced Mode}
 Which numbers are prime?
 \item[fraction=-50] 2
 \item[fraction=50] 5
 \item[fraction=50] 7
 \item[fraction=-50] 1
 \item[fraction=-50] 6
\end{multi}

```

---

Note that, in this example, negative fractions are set for wrong choices. This prevents students from obtaining a good grade with no merit if they select all answers.

Contrarily to the `single` mode where questions with negative fractions may lead to an overall negative grade, when multiple choices can be selected, the lowest grade MOODLE will take into account for the question is 0.

`allornothing` (*Key*) [*multi*] There also exists a “All-or-Nothing Multiple Choice” plugin for MOODLE that introduces a question type similar to a multiple choice with multiple correct answers, with the specificity that the points are given if and only if the student selects all correct answers. This kind of question is set up using the `allornothing` key. The recommended way for designating correct answers is with `\item*`. If instead the `fraction` is used, moodle will consider that non-negative fractions ( $> 0$ ) designate correct answers and negative fractions ( $\leq 0$ ) designate incorrect choices. The option `allornothing` supersedes the options `multiple` and `single`. To the best of our knowledge, MOODLE does not offer the all-or-nothing behavior for multiple choice questions embedded inside a `cloze` question.

### 2.4.3 Numerical

`numerical` (*env.*) [*quiz*] [`<question options>`]{`<question name>`} is an environment that defines a *Numerical* question which, in MOODLE, requires the student to input a real number in decimal form.

The typical format for this question type is:

```
\begin{numerical}[<question options>]{<question name>}
 <question text>
 \item[<answer 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={Forgot to antidifferentiate?}] <incorrect answer>
```

`tolerance` (*Key*) [*numerical*] The `tolerance` key can be used to specify the validity of answers within some margin.   
Default: 0 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 wildcard 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.

#### 2.4.4 Short Answer

`shortanswer` (*env.*) [quiz] [*question options*]{*question name*} is an environment that defines a *Short Answer* question. It 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[answer options] correct answer
 \item[answer 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` (*bool*) [shortanswer]    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 Boolean key `case sensitive` or its shorter synonym `usecase` .

`usecase` (*bool*) [shortanswer]

The **wildcard character** `*` can be used to grab answers that match a specific pattern. Following the order of answers, the first match will lead to the corresponding score and eventual feedback. As an example, take the following question

---

```
\begin{shortanswer}[usecase]{Newton's name}
 What was Newton's first name?
 \item Isaac
 \item[fraction=0,feedback={Simply Isaac!}] Isaa*
 \item[fraction=0,feedback={This one is Leibniz!}] *Gottfried*
 \item[fraction=0,feedback={First name, not title!}] Sir*
 \item[fraction=0,feedback={No\dots}] *
\end{shortanswer}
```

---

- the answer `Isaac` is the only one that gets rewarded,
- answers `Isaac Leibniz`, `Isaac Newton`, and `Isaak` yield the feedback *Simply Isaac!*,
- answers `Sir Gottfried Wilhelm`, `Sir Gottfried`, `Gottfried Wilhem`, and `Gottfried` yield the feedback *This one is Leibniz!*,
- answers `Sir Isaac`, and `Sir` yield the feedback *First name, not title!*,
- any answer that does not match the previous patterns yields the feedback *No...*

### 2.4.5 Essay

`essay (env.) [quiz]` [`<question options>`] [`<question name>`] is an environment that defines an *Essay* question.

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 <information 1 for grader>
 \item <information 2 for grader>
 \item <information n for grader>
\end{essay}
```

Instead of containing answers, the `\item` entries included in the essay question contain notes that will appear to whoever is grading the question manually. Contrarily to other question types, `\item` in essay questions do not take options.

`response required (bool) [essay]` 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 (Key)` [essay] 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:  
Default: `html`

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

`file` [response format] **file:** A file picker allowing the student to upload a file, such as a PDF or DOC file, containing the essay.

`html+file` [response format] **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` [response format] **text:** This editor allows only for entering plain text without any markup.

`monospaced` [response format] **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 (Key)` [essay] The key `response field lines` controls the height of the input box. For MOODLE, the admissible values are: 5, 10, 15, 20, 25, 30, 35, and 40. If the value set is not admissible, moodle will approximate the value:  
Default: `15`

- with either 5 or 40 if the value set was out of range, or
- with the next multiple of 5 otherwise.

`attachments allowed (Key)` [essay] The `attachments allowed` key controls *how many* attachments a student is allowed to upload. Admissible values are 0, 1, 2, 3, or unlimited.  
Default: `0`

`attachments required (Key)` [essay] You may also require the student to upload a certain number of attachments by setting `attachments required` to 0, 1, 2, or 3.  
Default: `0`

`template (Key)` [essay] 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.

#### 2.4.6 Matching

`matching (env)` [quiz] [`⟨question options⟩`]{`⟨question name⟩`} is an environment that defines a *Matching* question. It typically looks like this:

```
\begin{matching}[⟨question options⟩]{⟨question name⟩}
 ⟨question text⟩
 \item ⟨item 1⟩ \answer ⟨match 1⟩
 \item ⟨item 2⟩ \answer ⟨match 2⟩
\end{matching}
```

```

 :
 \item <item m> \answer <match m>
 \item \answer <no match 1>
 :
 \item \answer <no match n>
\end{matching}

```

`\answer` [matching] `<match>`s 1 through  $m$  are separated from their corresponding `<item>`s by the command `\answer`. Answers that match no item can be proposed at the end of the list, preceded by an empty item.

After import, MOODLE will recognize matches that are *exact* duplicates. If you intend multiple questions to have the same match, make sure that they are entered identically.

`shuffle` (*bool*) [matching] **Default: true** When students take a matching question, MOODLE always displays the proposed matches in random order. The matching question accepts the option `shuffle` to also randomly permute the items.

`drag and drop` (*bool*) [matching] **Default: false** The standard matching question offered by MOODLE corresponds to a dropdown box for choosing the match for each item. There also exists a “**drag and drop matching**” plugin for MOODLE that shows all items in a column (left), all proposed matches in a second column (right), and asks students to drag the correct match to each item with the mouse.

`dd` (*Key*) [matching] In this package, to enable drag-and-drop matching, use the key `drag and drop` or `dd` for short. Beware that, with the standard format (`drag and drop=false`), due to the limitations of dropdown boxes, MOODLE will not render  $\LaTeX$  or HTML code passed in the answers.

## 2.4.7 Cloze Questions and Subquestions

`cloze` (*env.*) [quiz] [`<question options>`]{`<question name>`} is an environment that defines a *Cloze* question.

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`, `numerical`, or `shortanswer` 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}
\end{cloze}

```




```

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, the question environments are *not* followed by a question name in braces.

- `multi (env)` [cloze] **multi:** [*subquestion options*] defines a Multiple Choice question inside a Cloze question,
- `numerical (env)` [cloze] **numerical:** [*subquestion options*] defines a Numerical question inside a Cloze question, and
- `shortanswer (env)` [cloze] **shortanswer:** [*subquestion options*] defines a Shortanswer question inside a Cloze question,
- `points (Key)` [cloze] **points** or `default grade` (Key) [cloze] Inside cloze environments, the `points` or `default grade` keys can be used to weight the worth of each subquestion. A specific constraint applies: the values should be positive integers.  
 Default: 1
- `fraction (Key)` [cloze] Inside cloze environments, the `fraction` key can be used to give partial credit or sanction for certain answers. The values specified must be integers and are independent of the admissible values listed in Table 1 on page 8.
- `single (bool)` [cloze] **single** Prior to 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 for questions where students must be able to select several answers. The modalities described in Section 2.4.2 on page 9 for setting the weight of answers apply. The only difference occurs in *advanced mode*: the sum of positive fractions may not be 100%. In this case, after importing the XML file, MOODLE will automatically scale the positive fractions for a total of 100% and leave intact the negative fractions.  
 Default: true
- `multiple (Key)` [cloze] **multiple**
- Within a cloze question, a multiple choice question may be displayed in three different modes:
- `inline (Key)` [cloze] **inline:** The inline dropdown box is the default choice when only one answer is to be selected (option `single`). The dropdown box is visually compact, but also prevents the use of mathematical or HTML formatting. This option is incompatible with `multiple` or `single=false` (dropdown boxes don't let you pick up several answers!).  

- `vertical (Key)` [cloze] **vertical:** displays the subquestion as a vertical column of radio buttons instead. This is the default when several items can be selected (options `multiple` or `single=false`).
- `horizontal (Key)` [cloze] **horizontal:** creates a horizontal row of radio buttons. This option works well when

the possible answers are short. The result is more compact than what `vertical` gives.

`shuffle` (*bool*) [cloze] 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` (*bool*) [cloze] Within a `cloze` question, the `shortanswer` question can be made case sensitive. This option, disabled by default, is selected with `case sensitive` or `usecase`.  
Default: `false`

`usecase` (*bool*) [cloze]

### 2.4.8 Description

`description` (*env*) [quiz] [*question options*]{*question name*} is an environment that defines a so-called *Description* question.

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 L<sup>A</sup>T<sub>E</sub>X's `description` environment. The scope of this redefinition is limited to the quiz environment.

The syntax for a Description question is as follows:

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

## 2.5 Summary of the Key Options

Table 3 on the next page, summarizes the key options available at the question and answer levels depending on the question type. For the essay questions, please refer to Section 2.4.5 on page 14.

## 2.6 Automated Generation of Questions

MOODLE's **Calculated Questions** types are not supported by this package.

However, much more flexibly, a scripting language can be used in combination with moodle to generate a large set of questions, based on a prototype.

In this case, the teacher building a quiz often wants one of the questions to be randomly picked up and presented to the student. To achieve this, we suggest to apply a specific tag to the questions sharing the same prototype. After import in MOODLE, when creating a quiz, this tag can be selected to narrow down a random selection of questions.

This way, the feature of calculated questions can be reproduced, while benefiting from the flexibility given by your favorite scripting language.

Table 3: Options offered at the question and answer levels for each question type.

| Question type  | Question |         |          |      |         |           |          |              |         |           |    | Answer   |          |           |
|----------------|----------|---------|----------|------|---------|-----------|----------|--------------|---------|-----------|----|----------|----------|-----------|
|                | points   | penalty | feedback | tags | shuffle | numbering | multiple | alldifferent | usecase | tolerance | dd | fraction | feedback | tolerance |
| Multichoice    | •        | •       | •        | •    | •       | •         | •        | •            |         |           |    | •        | •        |           |
| Numerical      | •        | •       | •        | •    |         |           |          |              |         | •         |    | •        | •        | •         |
| Short Answer   | •        | •       | •        | •    |         |           |          |              | •       |           |    | •        | •        |           |
| Matching       | •        | •       | •        | •    | •       |           |          |              |         |           | •  |          |          |           |
| True/False     | •        |         | •        | •    |         |           |          |              |         |           |    | •        |          |           |
| Description    |          |         | •        | •    |         |           |          |              |         |           |    |          |          |           |
| Cloze          |          | •       | •        | •    |         |           |          |              |         |           |    |          |          |           |
| Numerical      | •        |         |          |      |         |           |          |              |         | •         |    | •        | •        | •         |
| Short Answer   | •        |         |          |      |         |           |          |              | •       |           |    | •        | •        |           |
| Multi (inline) | •        |         |          |      | •       |           |          |              |         |           |    | •        | •        |           |
| Multi (horiz.) | •        |         |          |      | •       |           | •        |              |         |           |    | •        | •        |           |
| Multi (vert.)  | •        |         |          |      | •       |           | •        |              |         |           |    | •        | •        |           |

In this Section, two possible approaches are presented.

### 2.6.1 Script Generating TeX Code

Here are two examples inspired from the work of [A. Vohns](#). The first one relies on the native *Lua* capabilities of Lua<sup>La</sup>TeX.

```

\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=1 else points=2 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=1
 else:
 points=2
 cloze_print([x,y],points)
\end{python}
\end{quiz}

```

---

These two example codes yield the same XML content.

## 2.6.2 T<sub>E</sub>X Code As a Template

Instead of mixing T<sub>E</sub>X and scripting code into the same file, it is probably a better practice to write a `.tex` template file with predefined variables and manipulate the latter using an external script.

The moodle package facilitates the development of question templates. Specific control sequences (e.g.  $\TeX$  commands) can be used in place of answers or other parameters to the questions. In `draft` mode, this should be possible even where moodle expects numerical values (e.g. answers of numerical questions). After drafting and question development, the control sequences can then be substituted, by means of a scripting language, with instances of the variables pulled from a database (e.g. \*.csv file). This way, complex quizzes, with parameterized TikZ pictures for instance, can be produced.

This approach, briefly presented in the rest of this section, is demonstrated by C. Caprani in `genquiz`, with Python for scripting and Jinja as a templating engine.

The specific control sequence of the template is the  $\TeX$  command `\VAR`, taking the variable name to be substituted from the database as an argument. In order to highlight the fields for template development, in `draft` mode, this command is defined to typeset the variable name in bold red font. When generating the XML question bank, in `final` mode, the variable values are substituted for the specific control sequence. Variables can be used to define numerical answers, tolerances, and even parameterize TikZ pictures.

---

```

\documentclass{article}
\usepackage[draft]{moodle}
\usepackage{tikz}
\newcommand*{\VAR}[1]{\textcolor{red}{\textbf{#1}}}
\begin{document}
 \begin{quiz}{Disk or Square}
 \begin{cloze}[tags={disk area}]
 {Area of the Disk QID00\VAR{qidx}}
 \begin{tikzpicture}
 \draw[fill=black!20] circle[radius=1];
 \draw[<->] (0,0)--(1,0)node[above,midway]{\VAR{radius}};
 \end{tikzpicture}
 A disk has radius \VAR{radius}, what is its area?
 \begin{numerical}[points=1,penalty=0]
 \item[tolerance = \VAR{areatol}] \VAR{area}
 \item[fraction=0,feedback={Note the units.}] * % incorrect
 \end{numerical}
 Provide your answers to 3 significant digits.
 \end{cloze}
 \end{quiz}
\end{document}

```

---

A small python code that reads the template and renders the actual \*.tex file is shown below. In this example, the template file is called `quiz_template.tex` and the variables are hard-coded as a list of Python dictionaries. But nothing prevents you from generating them on-the-fly or picking them from a database. It is worth noting that `jinja2` will process all `\VAR{<name>}` occurrences, regardless of  $\TeX$  comments.

---

```

import os
import jinja2

```

```

Tell jinja what to look for in the template
(from web.archive.org/web/20121024021221/http://e6h.de/post/11/)
latex_jinja_env = jinja2.Environment(
 variable_start_string=r"\VAR{",
 variable_end_string=r"}",
 comment_start_string=r"\#{",
 comment_end_string="}",
 loader=jinja2.FileSystemLoader(os.path.abspath(".")),
)

Load the template
template = latex_jinja_env.get_template("quiz_template.tex")

Prepare the variables or read from a database
database = [
 {"qid": 1, "radius": 2, "areatol": 0.1, "area": 12.57},
 {"qid": 2, "radius": 3, "areatol": 0.2, "area": 28.27},
 {"qid": 3, "radius": 4, "areatol": 0.4, "area": 50.26},
]

for row in database:
 # combine template and variables
 document = template.render(**row)

 # XML files are generated by moodle.sty in "final" mode
 document = document.replace(r"\usepackage[draft]{moodle}",
 r"\usepackage{moodle}")

 # write document
 with open(f"{row['qid']}.tex", "w", encoding="utf-8") as outfile:
 outfile.write(document)

```

---

Executing this Python code will result in three new `.tex` files in the current directory. These can then be compiled with a  $\text{\TeX}$  engine to produce the individual XML files. Of course, for a large number of quizzes it is preferable to automate this compilation, and to combine the large number of resulting XML files into a single XML file for upload to Moodle. See [genquiz](#) for more details.

Naturally, the concept of templating, demonstrated here with Python, may be orchestrated using other scripting languages.

Table 4: Text mode diacritic macros undergoing a tailored conversion to HTML.

| Definition              | Letter List <sup>1</sup>       | Description         | Samples                 |
|-------------------------|--------------------------------|---------------------|-------------------------|
| <code>\"⟨letter⟩</code> | a, e, i, o, u, y               | umlaut or diaeresis | ä Ä ë Ê Ë Ì Ĩ ö Ö ... Ÿ |
| <code>\'⟨letter⟩</code> | a, e, i, o, u                  | acute accent        | á Á é É í Í ó Ó ú Ú     |
| <code>\.⟨letter⟩</code> | c, e, g, i, z                  | overdot             | č Č ě Ě ě ğ Ğ ĩ Ĩ ž Ž   |
| <code>\=⟨letter⟩</code> | a, e, g, i, o, u, y            | macron              | ā Ā ē Ē ē ğ Ğ ĩ Ĩ ... Ÿ |
| <code>\^⟨letter⟩</code> | a, e, i, o, u                  | circumflex          | â Â ê Ê î Î ô Ô û Û     |
| <code>\`⟨letter⟩</code> | a, e, i, o, u                  | grave accent        | à À è È ì Ì ò Ò ù Ù     |
| <code>\~⟨letter⟩</code> | a, n, o                        | tilde               | ã Ã ñ Ñ õ Ö             |
| <code>\b⟨letter⟩</code> | b, d, k, l, n, t, z            | macron below        | ḃ Ḃ ḏ Ḑ ḑ Ḓ ḓ Ḕ Ḕ ... Ḑ |
| <code>\c⟨letter⟩</code> | c, s, t                        | cedilla             | ç Ç ş Ş ț Ț             |
| <code>\d⟨letter⟩</code> | a, b                           | underdot            | ạ Ạ ɸ ɸ                 |
| <code>\H⟨letter⟩</code> | o, u                           | double acute accent | ő Ó Ű Ű                 |
| <code>\k⟨letter⟩</code> | a, e, i, o, u                  | ogonek              | ą Ą ę ę ĩ Ĩ ı ı ą ą     |
| <code>\r⟨letter⟩</code> | a, u                           | overring            | â Â û Û                 |
| <code>\u⟨letter⟩</code> | a, e, g, i <sup>2</sup> , o, u | breve               | ă Ă ă ă ĩ Ĩ ı ı Ő Ő Ű Ű |
| <code>\v⟨letter⟩</code> | c, d, e, l, n, r, s, t, z      | caron or háček      | č Č đ Ď ě Ě ř ... Ž     |

<sup>1</sup> The lowercase letters listed also stand for their uppercase equivalent.

<sup>2</sup> pdfTeX v3.14159265 typesets `\u{i}` with an objectionable tittle. Use `\u{\i}`.

### 3 Conversion to HTML

#### 3.1 Level of Support

The package `moodle.sty` tries to automatically convert the  $\LaTeX$  code included in the questions into HTML for web display.

With this aim, a number of  $\TeX$  and  $\LaTeX$  macros, commands, and environments undergo a tailored treatment when moodle generates the XML file. A few tables describe the current level of support:

1. text mode diacritic macros (e.g. `\"u`) in Table 4,
2. text mode macros for ligatures (e.g. `\oe`) and other glyphs (e.g. `\aa`) in Table 5 on the following page,
3. horizontal spacing (e.g. `\quad`) and line breaking (e.g. `\`) macros in Table 6 on the next page,
4. text mode symbols (e.g. `\$`) and punctuation (e.g. `\textexclamdown`) macros in Table 7 on page 25, and finally
5. other  $\LaTeX$  commands (e.g. `\emph`) and environments (e.g. `center`) in Table 8 on page 26.

In addition, `<` and `>` will be converted to their HTML equivalents `&lt;` and `&gt;` in the XML file. This prevents portions of the code to be interpreted by MOODLE as HTML tags.

Table 5: Text mode ligature and glyph macros undergoing a tailored conversion to HTML.

| Lowercase        |        | Uppercase                    |                |
|------------------|--------|------------------------------|----------------|
| Definition       | Sample | Definition                   | Sample         |
| <code>\aa</code> | å      | <code>\AA</code>             | Å              |
| <code>\ae</code> | æ      | <code>\AE</code>             | Æ              |
| <code>\dh</code> | ð      | <code>\DH</code>             | Ð              |
| <code>\dj</code> | đ      | <code>\DJ</code>             | Đ              |
| <code>\i</code>  | ı      |                              |                |
| <code>\ij</code> | ij     | <code>\IJ</code>             | IJ             |
| <code>\j</code>  | ĵ      |                              |                |
| <code>\l</code>  | ł      | <code>\L</code>              | Ł              |
| <code>\ng</code> | ŋ      | <code>\NG</code>             | Ŋ              |
| <code>\o</code>  | ø      | <code>\O</code>              | Ø              |
| <code>\oe</code> | œ      | <code>\OE</code>             | Œ              |
| <code>\ss</code> | ß      | <code>\SS<sup>1</sup></code> | ß <sup>2</sup> |
| <code>\th</code> | þ      | <code>\TH</code>             | Þ              |

<sup>1</sup> Contrarily to most fonts, Libertine, used in this documentation and available for instance via the package `libertine`, defines the glyph ß.

<sup>2</sup>  $\LaTeX$  defines the `\SS` macro but  $\pdfTeX$  renders it as a doubled capital S.

Table 6: Text mode horizontal spacing and line breaking macros undergoing a tailored conversion to HTML.

| Definition                                                                                          | Description                     |
|-----------------------------------------------------------------------------------------------------|---------------------------------|
| <code>\</code> , or <code>\thinspace</code>                                                         | narrow non-breaking space       |
| <code>~</code> or <code>\_</code>                                                                   | non-breaking space              |
| <code>\&gt;<sup>1</sup></code> , <code>\: <sup>2</sup></code> or <code>\medspace<sup>2</sup></code> | mid space                       |
| <code>\; <sup>2</sup></code> or <code>\thickspace<sup>2</sup></code>                                | thick space                     |
| <code>\enspace</code>                                                                               | nut (1en wide space)            |
| <code>\quad</code>                                                                                  | mutton (1em wide space)         |
| <code>\qqquad</code>                                                                                | doubled mutton (2em wide space) |
| <code>\textvisiblespace</code>                                                                      | sample: <code>_</code>          |
| <code>\\</code> or <code>\newline</code>                                                            | start a new line                |
| <code>\par</code> or <code>\blank line</code>                                                       | start a new paragraph           |

<sup>1</sup> `\>` is defined in math-mode only.

<sup>2</sup> `\:`, `\medspace`, `\;`, and `\thickspace` require the package `amsmath`.



Table 7: Text mode punctuation marks and symbol macros undergoing a tailored conversion to HTML. A baseline is represented in the samples of quotations marks, in order to draw attention to their vertical positioning.

| Package                                                      | Definition                                                         | Sample         |
|--------------------------------------------------------------|--------------------------------------------------------------------|----------------|
| L <sup>A</sup> T <sub>E</sub> X base                         | <code>\%</code>                                                    | %              |
|                                                              | <code>\#</code>                                                    | #              |
|                                                              | <code>\_</code>                                                    | _              |
|                                                              | <code>\textbackslash</code>                                        | \              |
|                                                              | <code>\\$</code>                                                   | \$             |
|                                                              | <code>\&amp;</code>                                                | &              |
|                                                              | <code>\\$</code>                                                   | §              |
|                                                              | <code>\{</code>                                                    | {              |
|                                                              | <code>\}</code>                                                    | }              |
|                                                              | <code>\texteuro</code>                                             | €              |
|                                                              | <code>\dots</code> or <code>\ldots</code>                          | ...            |
|                                                              | <code>\textexclamdown</code>                                       | !              |
|                                                              | <code>\textquestiondown</code>                                     | ?              |
|                                                              | <code>--</code>                                                    | —              |
|                                                              | <code>=</code>                                                     | =              |
|                                                              | <code>`</code> and <code>'</code> <sup>1</sup>                     | ‘ ’            |
|                                                              | <code>\textquoteleft</code> and <code>\textquoteright</code>       | ‘ ’            |
|                                                              | <code>`</code> and <code>'</code> <sup>1,2</sup>                   | “ ”            |
|                                                              | <code>\textquotedblleft</code> and <code>\textquotedblright</code> | “ ”            |
|                                                              | <code>\textquotesingle</code>                                      | ’              |
|                                                              | <code>"</code>                                                     | ” <sup>2</sup> |
|                                                              | <code>\textquotedbl</code>                                         | ”              |
| <code>\guilsinglleft</code> and <code>\guilsinglright</code> | ‹ ›                                                                |                |
| <code>\guillemotleft</code> and <code>\guillemotright</code> | « »                                                                |                |
| <code>\quotesinglbase</code>                                 | ↯                                                                  |                |
| <code>\quotedblbase</code>                                   | ↯                                                                  |                |
| eurosym                                                      | <code>\euro</code>                                                 | €              |
| babel                                                        | <code>\flq</code> and <code>\frq</code>                            | ‹ ›            |
|                                                              | <code>\flqq</code> and <code>\frqq</code>                          | « »            |
| babel, opt. french                                           | <code>\og</code> and <code>\fg</code> <sup>3</sup>                 | « »            |
| babel, opt. german                                           | <code>\glq</code> and <code>\grq</code>                            | ‘ ’            |
|                                                              | <code>\dq</code>                                                   | ” <sup>2</sup> |
|                                                              | <code>\glqq</code> and <code>\grqq</code>                          | “ ”            |

<sup>1</sup> When placed in math mode, the single straight quote (') is passed as-is to the XML, even when doubled (double prime symbol).

<sup>2</sup> In roman type, Lua<sup>A</sup>T<sub>E</sub>X and Xe<sup>A</sup>T<sub>E</sub>X typeset the double straight quote symbol (") and the command `\dq` as a double right quote ("). Instead, moodle follows pdf<sup>A</sup>T<sub>E</sub>X: whatever the T<sub>E</sub>X engine used, the double straight quote (") is passed to the XML.

<sup>3</sup> The way `\og` and `\fg` are typeset in the PDF depends on the current babel language. Regardless, moodle passes the symbols « and » to the XML.

Table 8:  $\LaTeX$  commands and environments undergoing a tailored conversion to HTML.

| Package              | Commands                                                                                                                                                                  | Environments                                |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------|
| $\LaTeX$ base        | <code>{}</code>                                                                                                                                                           | <code>\$. . . \$</code> (inline math)       |
|                      | <code>\relax</code>                                                                                                                                                       | <code>\( . . . \)</code> (inline math)      |
|                      | <code>\LaTeX</code>                                                                                                                                                       | <code>\$\$ . . . \$\$</code> (display math) |
|                      | <code>\TeX</code>                                                                                                                                                         | <code>\[ . . . \]</code> (display math)     |
|                      | <code>\emph{&lt;...&gt;}</code>                                                                                                                                           | <code>center</code>                         |
|                      | <code>\textbf{&lt;...&gt;}</code>                                                                                                                                         | <code>enumerate</code>                      |
|                      | <code>\textit{&lt;...&gt;}</code>                                                                                                                                         | <code>itemize</code>                        |
|                      | <code>\textsc{&lt;...&gt;}</code>                                                                                                                                         | <code>quote</code>                          |
|                      | <code>\textsuperscript{&lt;...&gt;}</code>                                                                                                                                | <code>quotation</code>                      |
|                      | <code>\textsubscript{&lt;...&gt;}</code>                                                                                                                                  |                                             |
|                      | <code>\texttt{&lt;...&gt;}</code>                                                                                                                                         |                                             |
|                      | <code>\underline{&lt;...&gt;}</code>                                                                                                                                      |                                             |
| hyperref             | <code>\href{&lt;...&gt;}{&lt;...&gt;}</code>                                                                                                                              |                                             |
| url or hyperref      | <code>\url{&lt;...&gt;}</code>                                                                                                                                            |                                             |
| babel, opt. french   | <code>\fup{&lt;...&gt;}</code><br><code>\up{&lt;...&gt;}</code>                                                                                                           |                                             |
| graphics or graphicx | <code>\includegraphics[&lt;...&gt;]{&lt;...&gt;}</code>                                                                                                                   |                                             |
| tikz                 | <code>\tikz[&lt;...&gt;]{&lt;...&gt;}</code>                                                                                                                              | <code>tikzpicture[&lt;...&gt;]</code>       |
| verbatim             | <code>\verbatiminput{&lt;...&gt;}</code>                                                                                                                                  |                                             |
| fancyverb or fvextra | <code>\VerbatimInput[&lt;...&gt;]{&lt;...&gt;}</code><br><code>\LVerbatimInput[&lt;...&gt;]{&lt;...&gt;}</code><br><code>\BVerbatimInput[&lt;...&gt;]{&lt;...&gt;}</code> |                                             |
| minted               | <code>\inputminted[&lt;...&gt;]{&lt;...&gt;}{&lt;...&gt;}</code>                                                                                                          |                                             |

A doubled dash will be converted to en-dash `&ndash;`; *outside math mode*. Empty groups `{}` will be passed to the XML only *in math mode*.

Be aware that, apart what is described previously, moodle *does not know how to convert any other T<sub>E</sub>X or L<sub>A</sub>T<sub>E</sub>X commands to HTML*. If other sequences are used, they may be passed verbatim to the XML file or may cause unpredicted results.

If you think of another L<sub>A</sub>T<sub>E</sub>X command that should be changed to an HTML equivalent, please have a look at [Section 5 on page 35](#).

`\htmlonly` [feedback] [*<Content for traditional output>*] [*<HTML content>*] is a command to be used inside question environments (text, answers, or feedback). It lets you pass directly code to the XML file while being ignored for the traditional output (PDF). The HTML content passed as an argument is subject to no particular processing and users should not expect to be able to pass dangerous characters like `\`, `%`, or `#`. An optional argument allows to pass contents to be processed for the traditional output. This argument is ignored for the XML output. For instance, one can write code like this in a question environment

```
\htmlonly[\fbox{PDF contents}]{
 <div style="border: 1px solid; display: inline-block;">
 HTML contents</div>
}
```

`\htmlregister` [moodle] [*<command>*] is a command that lets you specify a macro that must be expanded in the XML file. It works only when the macro is defined without optional argument.



`\moodleregisternewcommands` [moodle] When the list of user-defined macros is long, it becomes cumbersome to record them individually for expansion. Calling `\moodleregisternewcommands` watches for subsequent calls to `\newcommand`, `\renewcommand`, `\providecommand`, and their starred variants such that the corresponding commands are automatically expanded by moodle. Again, this works only if the macros are defined *without* optional argument.



## 3.2 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 `TikZ`. 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.

### 3.2.1 Default `includegraphics`

`\includegraphics` [graphicx] [*<options>*] [*<file>*] can be used to include graphic files in both PDF and the XML outputs. 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 T<sub>E</sub>X dimensions such as 4 in or 2.3 cm. In order to prepare the image for web viewing, this package converts those

`ppi` (*Key*) [`includegraphics`] dimensions to pixels using a default of 103 pixels per inch.<sup>1</sup> 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 `\moodleaset` command, rather than image-by-image.

`\graphicspath` [`graphicx`] `{\langle path1 \rangle}, {\langle path2 \rangle}, \dots` can be used to specify the locations of the pictures to be included.

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 pdf $\TeX$  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.

### 3.2.2 TikZ Pictures

TikZ is a user-friendly syntax layer for PGF, the macro  $\TeX$  package for creating graphics. More information on TikZ can be found at <https://ctan.org/pkg/pgf>.

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

### 3.2.3 Package Option `tikz`

`tikz` (*Opt*) [`moodle`] The moodle package admits a `tikz` option which has the following effects:

- the package `tikz` is loaded.
- `\includegraphics` [`tikz`]
  - `\includegraphics` embeds graphics in a TikZ 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` [`tikz`]
  - a macro `\embedaspict{\langle  $\TeX$  contents \rangle}` is provided for the conversion of inline  $\TeX$  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 TikZ-external library are disabled: compilation might get sensibly slower.

### 3.2.4 External Tools

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

<sup>1</sup>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.

1. *GhostScript* ([www.ghostscript.com](http://www.ghostscript.com)) is used to convert the PDF output from TikZ 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).

`\ghostscriptcommand` [moodle]

If your system requires a different command line to invoke *Ghostscript*, you may change it by invoking:

```
\ghostscriptcommand{\executable filename}
```

2. When external graphics files such as PDF are included, the *ImageMagick* software ([www.imagemagick.org](http://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` [moodle]

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` [moodle]

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  $(pdf|xe|lua)latex$ . Including ps graphics or using TikZ 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.
- $\TeX$  must be able to call system commands; that is, `\write18` must be enabled. For Mik $\TeX$ , this means adding `--enable-write18` to the command line of  $(pdf|xe|lua)latex$ ; for  $\TeX$  Live, 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-zealous security policy. For example, [see this](#).
- Users of the `circuitikz` package (<https://www.ctan.org/pkg/circuitikz>) must enclose their circuits' TikZ code in the `tikzpicture` environment instead of the historical `circuitikz` environment. That is required, as of TikZ 2.1, by the external library.

 experimental

### 3.2.5 Package Option `svg`

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

`svg (Opt)` [moodle] The moodle package admits an experimental `svg` option which has the following effects:

- `\includegraphics` [svg]
- `\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 being 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 un-optimized `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.

The call of external `svg` manipulation utilities can be modified using the macros:

`\PDFtoSVGcommand` [svg] `\PDFtoSVGcommand:` for conversion from `PDF` to `SVG`,

`\SVGtoPDFcommand` [svg] `\SVGtoPDFcommand:` for conversion from `SVG` to `PDF`, and

`\optiSVGcommand` [svg] `\optiSVGcommand:` for the optimization of `svg` files.

### 3.3 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`, `verbatim` and other standard utilities is not supported.

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

- `\verbatiminput` [verbatim]
1. `\verbatiminput{\filename}` from the `verbatim` package inserts verbatim code in both the `PDF` and the `XML` for `MOODLE`, without fancy additions.
- `\VerbatimInput` [fvextra]
2. The macro `\VerbatimInput{\options}\filename}` from `fancyvrb` or `fvextra` does more, with several options and settings offered (see below). The variants `\BVerbatimInput` and `\LVerbatimInput` are also supported, with no difference on the `XML` output. The variants with a star are unsupported and result in errors when used.
- `\BVerbatimInput` [fvextra]
- `\LVerbatimInput` [fvextra]

Table 9: Options and corresponding values considered for XML generation of verbatim material with `\VerbatimInput` and `\inputminted`.

Option keys	Possible values
<code>gobble</code>	$\langle integer \rangle$
<code>autogobble</code> <sup>1</sup>	true or false
<code>tabsize</code>	$\langle integer \rangle$
<code>numbers</code>	none, left, right, or both <sup>2</sup>
<code>firstnumber</code>	auto, last, or $\langle integer \rangle$
<code>firstline</code>	$\langle integer \rangle$
<code>lastline</code>	$\langle integer \rangle$
<code>numberblanklines</code>	true or false
<code>highlightlines</code> <sup>2</sup>	$\{ \langle comma-separated list of integers or ranges \rangle \}$
<code>style</code> <sup>1</sup>	$\langle string \rangle$

<sup>1</sup> `autogobble`, `numbers=both`, and `style` are from `minted`.

<sup>2</sup> line highlighting is offered only with `fvextra` or `minted` loaded.

`\inputminted` [`minted`]

3. On top of that `\inputminted[\langle options \rangle]{\langle lang \rangle}{\langle filename \rangle}` from the `minted` package offers syntax highlighting tailored to the specified language.

The `moodle` package handles these three commands to pass the code in the output XML. With `\inputminted`, an external Python tool, `pygmentize`, performs syntax analysis and its HTML formatter is used to populate the XML. With the other commands, the contents of the file is passed almost as-is to the XML: in order to survive MOODLE import and HTML rendering, characters `<`, `>`, `&`, `'`, and `"` are converted to HTML equivalents.

With `\VerbatimInput` and `\inputminted`, the options that are taken care of for XML generation are listed in Table 9. Using `\fvset{\langle key=value,... \rangle}`, options can be set globally. Equivalently, with `minted`, `\setminted[\langle lang \rangle]{\langle key=value,... \rangle}` 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).

Here is an example:

---

```

\documentclass[12pt,a4paper]{article}
\usepackage[section]{moodle}
\usepackage{minted}
\begin{document}
\begin{quiz}[tags={minted}]{LaTeX Quiz}
 \begin{filecontents*}{excerpt.tex}
% !TeX encoding = UTF-8
% !TeX spellcheck = en_US
% !TEX TS-program = lualatex
\documentclass{article}
\usepackage[nostamp]{moodle}

```

```

\ifPDFTeX % FOR LATEX and PDFLATEX
 \usepackage[utf8]{inputenc} % necessary
 \usepackage[T1]{fontenc} % necessary
\else % assuming XELATEX or LUALATEX
 \usepackage{fontspec}
\fi
\end{filecontents*}
\begin{numerical}[tolerance=0]{Loading a Class}
 Consider the following \LaTeX code excerpt.\\
 \inputminted[numbers=left]{latex}{excerpt.tex}
 On which line is the class loaded?
 \item[feedback={
 yes! \inputminted[highlightlines={4}]{latex}{excerpt.tex} }] 4
 \item[feedback={Line 3 is just a comment.},fraction=0] 3
 \item[feedback={Line 5 loads the package \texttt{moodle}},fraction=0] 5
\end{numerical}
\begin{multi}[single]{IDE}
 Consider the following \LaTeX code excerpt.\\
 \inputminted[numbers=left]{latex}{excerpt.tex}
 Which \TeX engine will be used by the IDE for compilation.
 \item[feedback={Have a closer look at line 3}] \texttt{tex}
 \item[feedback={Have a closer look at line 3}] \texttt{latex}
 \item[feedback={Have a closer look at line 3}] \texttt{pdflatex}
 \item[feedback={Have a closer look at line 3}] \texttt{xelatex}
 \item* \texttt{lualatex}
\end{multi}
\end{quiz}
\end{document}

```

---

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 utilities 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 overridden using `\fvset` after the preamble.



## 4 Internationalization

This section is intended for authors of MOODLE quizzes writing in one or several languages other than English. It seems reasonable to assume that those users have heard of the packages `babel` or `polyglossia` (X<sub>Y</sub>TeX or LuaTeX only), that both aim at enforcing language-related rules to L<sup>A</sup>T<sub>E</sub>X documents.

The contents of the XML file that is generated by moodle depends entirely on the user input and MOODLE's XML syntax: there is little room for moodle to internationalize something here. Instead, the focus of this section is the PDF typesetting that is determined by moodle, where some internalization efforts can be of help.

Both `babel` and `polyglossia` provide ways of specifying a language for the document or a part of it. If one of these packages is loaded in the preamble, moodle will automatically load the package translator and rely on it to provide translations of its keys (see Table 10) in different languages, with some knowledge on aliases. The package translations does a very similar job and can be loaded in the preamble to serve, if desired, as a replacement for translator.

Currently, full-support is provided for English, French, German, Italian, and Spanish. Very limited support is provided for Catalan, Croatian, Czech, Danish, Dutch, Estonian, Finnish, Hungarian, Icelandic, Lithuanian, Norsk, Polish, Portuguese, Romanian, Swedish, and Turkish.

Users may define their own translations in the preamble. For instance

```
\AfterEndPreamble{
 \DeclareTranslation{\<babel language name>}{True}{Foo}%
 \DeclareTranslation{\<babel language name>}{False}{Bar}%
}
```

Note that, when the package translations is loaded, `\AfterEndPreamble` must not be used.

Contributions to improve or broaden the linguistic support are very welcome.

Table 10: moodle's language keys for internationalization of the PDF typesetting.

Key	Context
<code>True</code>	indicates option "True" in a True/False question.
<code>False</code>	indicates option "False" in a True/False question.
<code>cloze</code>	tag indicating a "cloze" question.
<code>description</code>	tag indicating a "description" question.
<code>essay</code>	tag indicating an "essay" question.
<code>matching</code>	tag indicating a "matching" question.
<code>multi</code>	tag indicating a "multichoice" question.
<code>numerical</code>	tag indicating a "numerical" question.
<code>shortanswer</code>	tag indicating a "shortanswer" question.

*Continued on next page*

Table 10 – *Continued from previous page*

Key	Context
truefalse	tag indicating a “true/false” question.
Shuffle	tag indicating that options offered will be shuffled in a “multichoice” or “matching” question.
Single	tag indicating that only one option can be selected in a “multichoice” question.
Multiple	tag indicating that several options can be selected in a “multichoice” question.
marked out of	tag indicating the weight of the question (maximum number of points), followed by a number.
penalty	tag indicating the penalty factor applied for each wrong attempt in adaptive mode.
tags	indicates the beginning of a tag list characterizing the question.
All-or-nothing	tag indicating that all correct options must be selected to get credited a good answer.
Case-Sensitive	tag indicating that the case of characters matters for a “Shortanswer” question.
Case-Insensitive	tag indicating that the case of characters does not matter for a “Shortanswer” question.
Drag and drop	tag indicating that a “matching” question relies on the Drag-and-drop plugin.
Information for graders	indicates the beginning of a paragraph where instructions for the graders of an “essay” question are given.
Response template	indicates the beginning of a paragraph where the answer template of an “essay” question is represented.
editor	tag indicating that, for answering an essay question, an editor with HTML support will be proposed.
editorfilepicker	tag indicating that, for answering an essay question, an editor with HTML support and a file picker will be proposed.
plain	tag indicating that, for answering an essay question, an editor with no markup support will be proposed.
monospaced	tag indicating that, for answering an essay question, an editor with fixed-width font and no markup support will be proposed.
noinline	tag indicating that, for answering an essay question, a file picker will be proposed.
Total of marks	at the end of the quiz, indicates the sum of the weights of all questions, followed by a colon and a number.

## 5 Package Development

### 5.1 Feature Requests, Bug Reports, and Contributions

This package is developed as a collaborative project, currently hosted on the Gitlab server instance <https://framagit.org/mattgk/moodle>. The project's activity can be monitored there: reported issues, last modifications, ...

Contributions, either bug reports or fixes, are welcome. Users willing to help can either sign-in with an existing GITHUB, GITLAB.COM, or BITBUCKET account or register a new account.

Of course, getting in touch with the package maintainer by [email](#) works as well.

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.

When experiencing a problem and before reporting it, please check whether or not something similar has already been filed as an issue [here](#). If the problem appears to be new, please report it by following these steps:

1. Prepare a *minimal* working example, i.e. a `.tex` file shrunk down to the strict minimum (loaded packages, code, ...) while still showing the faulty behavior upon compilation.
2. Gather and send the `*.tex`, `*.log` and `*-moodle.xml` files together with an explanation about
  - your local working environment (T<sub>E</sub>X engine and distribution, platform, external tools used, ...),
  - the version of the MOODLE instance you are using (for instance see the file `https://<server-domain>/lib/upgrade.txt` )
  - the faulty behavior, and
  - what you expected instead.

### 5.2 Known Limitations

When using with pdfT<sub>E</sub>X for compilation, moodle supports question contents with ASCII characters only. The use of non-ASCII characters may work in some cases but will most probably yield compilation errors or undesired XML contents.

Instead, moodle will work flawlessly when X<sub>Y</sub>T<sub>E</sub>X and LuaT<sub>E</sub>X are used to compile UTF-8 encoded documents.

Table [11 on the next page](#) lists some features supported, limitations, and bugs.

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

Table 11: Content enrichment (pictures, equations) support after XML import in MOODLE v3.5.7, depending on the question type.

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

<sup>1</sup> MOODLE prompts the student for an answer and then compares it to the solutions provided. This is text-only.

<sup>2</sup> MOODLE uses a dropdown list to let one choose among the possible answers. This forbids either picture inclusion and  $\LaTeX$  rendering.

<sup>3</sup> Not supported by MOODLE (in this context, answer-specific feedback represents lots of possible combinations).

<sup>4</sup> 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.

<sup>5</sup> 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.

<sup>6</sup> Picture and  $\LaTeX$  rendering could be done, but only after submission and only if the keyval "response format" is set to "html".

<sup>7</sup> MOODLE only reveals the feedback when hovering the checkmark or X mark with the mouse.

- MOODLE’s feature of designating feedback for correct, partially correct, and incorrect answers.
- Hints
- Units handling in numerical questions

### 5.3 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.

The package option `LMS` lets you specify the targeted MOODLE version and helps ensuring compatibility.

As the ultimate purpose of this package is the generation of XML files, future versions of moodle will attempt to maintain backwards compatibility 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.

### 5.4 Unrelated Tip: Quality of MOODLE T<sub>E</sub>X Images

This has nothing to do with moodle, but is a Frequently Asked Question in its own right. On some servers, at least, MOODLE’s default “T<sub>E</sub>X Filter” for displaying mathematical notation is of abysmally poor quality, rendering mathematics as low-resolution PNG’s. One solution that has worked for me is to go to “Course Administration → Filters,” turn “T<sub>E</sub>X Notation” *off*, but turn “MathJax” *on*. This forces T<sub>E</sub>X code to be rendered by MathJax instead of MOODLE, producing much higher-quality results.

## Version History

### 1.0 Matthieu Guerquin-Kern (2023-01-28)

#### Added

- For templating purposes, in draft mode, general TeX code can now be used in items and `tolerance` key of numerical questions (courtesy of Colin Caprani).
- Support for the `quote` and `quotation` environments in the HTML conversion process (courtesy of Gerald Teschl).

- Experimental package option `pluginfile` mimics the way MOODLE embeds pictures in the XML.
- Answer box for Essay questions with `handout`.
- Key-value package option `LMS` to target a specific version of MOODLE.
- Support for more text mode diacritic and horizontal spacing commands.
- Macros `\moodleversion` and `\moodledate` are defined.
- Experimental package option `samepage` to avoid splitting questions across multiple pages.
- `sanction` key to set default mark for incorrect answers.
- Started the internationalization of the package based on translator (optionally translations). English, French, German, Italian, and Spanish are fully supported (with help from Jürgen Vollmer for German and Romano Giannetti for Italian and Spanish). Contributions are welcome.
- Warn user of the `babel` package set for Turkish that using the shorthands will not play well with moodle.
- Support for `\i` in conversion to HTML.
- `svg` option support for Windows (courtesy of Wolf Müller).

### Changed

- In Cloze `multi` subquestions, the default mode is `inline` when only one answer is to be selected and `vertical` when several answers can be selected.
- Single straight quotes (apostrophe) in math mode are passed as-is to the XML (with help from Keno Wehr).
- Empty groups `{ }` in math-mode are passed as-is to the XML (with help from Keno Wehr).
- Inside `cloze` subquestions, `points` are forced to the nearest positive integer.
- XML stamp exposes compilation time.
- Labels of proposed answers now following MOODLE's convention (1., 2., ...) in the PDF typesetting of `multi` questions.
- Automatic sanction mechanism for incorrect choices in `multi` questions with multiple answers allowed. Now applies also inside `cloze` questions.
- In `cloze` subquestions, non-integer fractions are rounded.
- Rewording of some indications in the PDF, related to internationalization.
- The answers of `matching` questions are now converted to HTML (accents and `\htmlonly` in mind).

### Removed

- Support for non-ASCII characters abandoned when compiling with `pdflatex`.

### Fixed

- In math-mode, `\underline` is no longer translated to HTML.
- Preexisting pictures no longer get deleted when including graphics without specifying the extension (courtesy of Colin Caprani).

- In answers of matching questions, some  $\TeX$  macros will not break compilation. A problem with curly braces persists.
- The cloze subquestions now inherit `points` set at the higher levels.
- Option `single=false` now works in Cloze multi questions.
- Undesired trailing spaces are removed from offered answers in matching question.
- Compatibility with siunitx version 3.
- Compatibility with minted starting from 2021/12/24 v2.6.
- Different issues with graphics handling on Windows (*magick*, *del*, *move*,...).
- Symbols `<` and `>` are translated to HTML equivalents, also in text-mode.
- Warning german-writing and (Xe|Lua) $\TeX$  users about troubles caused by text-mode umlauts.
- In cloze subquestions items and outside math environments, the equal symbol (=) no longer interferes with the Cloze syntax.
- A pagebreak occurring inside cloze shortanswers would reset page numbers.
- The redefinition of the `description` environment is limited to the scope of the quiz.
- PDF typesetting of matching answers (line breaks, repeated matches).
- Symbols `\%` found inside math mode are escaped for HTML.
- The `fraction` keys specified inside cloze questions are forced to integer values as required by MOODLE's XML format.
- True/False no longer broken when hyperref is loaded.
- Tolerances in numerical answers now correctly displayed in PDF.
- Warning for users running a too old version of graphics.
- In the code included with `verbatim`, `fancyvrb`, or `fvextra`, characters that matter for HTML are escaped.

#### 0.9 Matthieu Guerquin-Kern (2021-02-07)

##### Added

- Support for **all-or-nothing multiple choice** questions.
- Support for the ogonek diacritical mark via `\k{\langle...}\rangle`.
- Warn user of the babel package set for a German-related language that using the character " will not play well with moodle.
- Support for babel commands related to German quotes.
- Support for en-dash (–) outside of math mode.
- Support for `\%` in conversion to HTML.
- Command `\htmlonly []{ }` to pass HTML contents directly to the XML file.

##### Changed

- An error is issued when a graphics conversion step fails.

- Irrelevant points are no longer written at the `cloze` question level in `PDF` and `XML`.
- The total number of marks is shown in the `PDF` at the end of each quiz.
- The `tags` key can now be used to specify a comma-separated list of tags.

#### **Fixed**

- Answer text of `shortanswer` questions is converted to `HTML`.
- Paragraph breaks in `multi` and `essay` items no longer break compilation.
- Question text in `Essays` was not shown in `PDF` file.
- Commands `\textsc`, `\underline`, `\url`, and `\href` yielded `HTML` code with inadequate double quotation marks.
- Broken base64-encoding pipeline for images under Windows (thanks to Andreas Vohns).
- Repeated single right quotation marks no longer merged in math mode (thanks to Alberto Albano).

#### **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.
- `Lua $\TeX$`  is now supported (and recommended for `UTF-8` 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.
- Significantly 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  $\LaTeX$  command 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**

- TikZ externalization now works when using  $X_{\LaTeX}$ .
- It is now possible to set points manually among several correct answers in multichoice questions.
- General feedback 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 is now written in XML depending on the  $\LaTeX$  compiler used.

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

**Added**

- $X_{\LaTeX}$  is now recommended when using UTF-8 encoded sources (support of accents).
- Feedbacks are now displayed in the PDF file produced.
- The *OptiPNG* utility is used 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, TikZ externalization is no longer triggered.

**Fixed**

- In the different question types, the feedback fields are now converted for HTML allowing L<sup>A</sup>T<sub>E</sub>X equations and images.
- Documentation improvements (limitations and previously undocumented features).

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

## Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

<b>A</b>	
allornothing (key) [multi] . . . . .	<i>12</i>
[answer]:	
feedback (key) . . . . .	<i>8</i>
fraction (key) . . . . .	<i>7</i>
fractiontol (key) . . . . .	<i>8</i>
\answer [matching] . . . . .	<i>16</i>
attachments_allowed (key) [essay] . .	<i>15</i>
attachments_required (key) [essay] .	<i>15</i>
<b>B</b>	
Booleans:	
case_sensitive [cloze] . . . . .	<i>18</i>
case_sensitive [shortanswer] . .	<i>13</i>
drag_and_drop [matching] . . . . .	<i>16</i>
response_required [essay] . . . . .	<i>14</i>
shuffle [cloze] . . . . .	<i>18</i>
shuffle [matching] . . . . .	<i>16</i>
shuffle [multi] . . . . .	<i>10</i>
single [cloze] . . . . .	<i>17</i>
single [multi] . . . . .	<i>10</i>
usecase [cloze] . . . . .	<i>18</i>
usecase [shortanswer] . . . . .	<i>13</i>
\BVerbatimInput [fvextra] . . . . .	<i>30</i>
<b>C</b>	
case_sensitive (boolean) [cloze] . . . .	<i>18</i>
case_sensitive (boolean)	
[shortanswer] . . . . .	<i>13</i>
[cloze]:	
case_sensitive (boolean) . . . . .	<i>18</i>
default_grade (key) . . . . .	<i>17</i>
fraction (key) . . . . .	<i>17</i>
horizontal (key) . . . . .	<i>17</i>
inline (key) . . . . .	<i>17</i>
multiple (key) . . . . .	<i>17</i>
points (key) . . . . .	<i>17</i>
shuffle (boolean) . . . . .	<i>18</i>
single (boolean) . . . . .	<i>17</i>
usecase (boolean) . . . . .	<i>18</i>
vertical (key) . . . . .	<i>17</i>
cloze (env.) . . . . .	<i>16</i>
<b>D</b>	
dd (key) [matching] . . . . .	<i>16</i>
default_grade (key) [cloze] . . . . .	<i>17</i>
default_grade (key) [question] . . . . .	<i>7</i>
description (env.) . . . . .	<i>18</i>
draft (option) [moodle] . . . . .	<i>6</i>
drag_and_drop (boolean) [matching] . .	<i>16</i>
<b>E</b>	
\embedaspict [tikz] . . . . .	<i>28</i>
environments:	
cloze . . . . .	<i>16</i>
description . . . . .	<i>18</i>
essay . . . . .	<i>14</i>
matching . . . . .	<i>15</i>
multi . . . . .	<i>9, 17</i>
numerical . . . . .	<i>12, 17</i>
quiz . . . . .	<i>7</i>
shortanswer . . . . .	<i>13, 17</i>
truefalse . . . . .	<i>9</i>
[essay]:	
attachments_allowed (key) . . . . .	<i>15</i>
attachments_required (key) . . . . .	<i>15</i>
response_field_lines (key) . . . . .	<i>15</i>
response_format (key) . . . . .	<i>15</i>
response_required (boolean) . . . . .	<i>14</i>
template (key) . . . . .	<i>15</i>
essay (env.) . . . . .	<i>14</i>
<b>F</b>	
[feedback]:	
\htmlonly . . . . .	<i>27</i>

feedback (key) . . . . .	8	feedback . . . . .	8
feedback (key) [answer] . . . . .	8	feedback [answer] . . . . .	8
feedback (key) [question] . . . . .	8	feedback [question] . . . . .	8
file (other) [response_format] . . . . .	15	fraction [answer] . . . . .	7
final (option) [moodle] . . . . .	6	fraction [cloze] . . . . .	17
fraction (key) [answer] . . . . .	7	fraction [item*] . . . . .	10
fraction (key) [cloze] . . . . .	17	fraction [item] . . . . .	10
fraction (key) [item*] . . . . .	10	fractiontol [answer] . . . . .	8
fraction (key) [item] . . . . .	10	height [includegraphics] . . . . .	27
fractiontol (key) [answer] . . . . .	8	horizontal [cloze] . . . . .	17
[fvextra]:		inline [cloze] . . . . .	17
\BVerbatimInput . . . . .	30	multiple [cloze] . . . . .	17
\LVerbatimInput . . . . .	30	multiple [multi] . . . . .	11
\VerbatimInput . . . . .	30	numbering [multi] . . . . .	10
		penalty [question] . . . . .	7
<b>G</b>		points [cloze] . . . . .	17
\ghostscriptcommand [moodle] . . . . .	29	points [question] . . . . .	7
\graphicspath [graphicx] . . . . .	28	ppi [includegraphics] . . . . .	28
[graphicx]:		response_field_lines [essay] . . . . .	15
\graphicspath . . . . .	28	response_format [essay] . . . . .	15
\includegraphics . . . . .	27	sanction [multi] . . . . .	11
		tags . . . . .	8
<b>H</b>		tags [question] . . . . .	9
handout (option) [moodle] . . . . .	6	tags [quiz] . . . . .	9
height (key) [includegraphics] . . . . .	27	template [essay] . . . . .	15
horizontal (key) [cloze] . . . . .	17	tolerance [numerical] . . . . .	12
html (other) [response_format] . . . . .	15	vertical [cloze] . . . . .	17
htmlfile+ (other) [response_format] . . . . .	15	width [includegraphics] . . . . .	27
\htmlonly [feedback] . . . . .	27		
\htmlregister [moodle] . . . . .	27	<b>L</b>	
		LMS (option) [moodle] . . . . .	6
<b>I</b>		\LVerbatimInput [fvextra] . . . . .	30
\imagemagickcommand [moodle] . . . . .	29		
[includegraphics]:		<b>M</b>	
height (key) . . . . .	27	[matching]:	
ppi (key) . . . . .	28	\answer . . . . .	16
width (key) . . . . .	27	dd (key) . . . . .	16
\includegraphics [graphicx] . . . . .	27	drag_and_drop (boolean) . . . . .	16
\includegraphics [svg] . . . . .	30	shuffle (boolean) . . . . .	16
\includegraphics [tikz] . . . . .	28	matching (env.) . . . . .	15
inline (key) [cloze] . . . . .	17	[minted]:	
\inputminted [minted] . . . . .	31	\inputminted . . . . .	31
[item]:		monospaced (other) [response_format] . . . . .	15
fraction (key) . . . . .	10	[moodle]:	
[item*]:		draft (option) . . . . .	6
fraction (key) . . . . .	10	final (option) . . . . .	6
		\ghostscriptcommand . . . . .	29
<b>K</b>		handout (option) . . . . .	6
Keys:		\htmlregister . . . . .	27
allornothing [multi] . . . . .	12	\imagemagickcommand . . . . .	29
attachments_allowed [essay] . . . . .	15	LMS (option) . . . . .	6
attachments_required [essay] . . . . .	15	\moodleregisternewcommands . . . . .	27
dd [matching] . . . . .	16	\moodleaset . . . . .	7
default_grade [cloze] . . . . .	17	nostamp (option) . . . . .	6
default_grade [question] . . . . .	7		



<code>\SVGtoPDFcommand</code> .....	30	<code>truefalse (env.)</code> .....	9
<code>svg (option) [moodle]</code> .....	6, 30		
<code>\SVGtoPDFcommand [svg]</code> .....	30		
		<b>U</b>	
		<code>usecase (boolean) [cloze]</code> .....	18
		<code>usecase (boolean) [shortanswer]</code> .....	13
		<b>T</b>	
<code>tags (key)</code> .....	8		
<code>tags (key) [question]</code> .....	9		
<code>tags (key) [quiz]</code> .....	9		
<code>template (key) [essay]</code> .....	15		
<code>text (other) [response_format]</code> .....	15		
<code>[tikz]:</code>			
<code>\embedaspict</code> .....	28		
<code>\includegraphics</code> .....	28		
<code>tikz (option) [moodle]</code> .....	6, 28		
<code>tolerance (key) [numerical]</code> .....	12		
		<b>V</b>	
		<code>[verbatim]:</code>	
		<code>\verbatiminput</code> .....	30
		<code>\VerbatimInput [fvextra]</code> .....	30
		<code>\verbatiminput [verbatim]</code> .....	30
		<code>vertical (key) [cloze]</code> .....	17
		<b>W</b>	
		<code>width (key) [includegraphics]</code> .....	27