XSIM

v0.7 2017/05/09

eXercise Sheets IMproved
the official successor of the exsheets package

Clemens Niederberger

https://github.com/cgnieder/xsim

contact@mychemistry.eu

Table of Contents

1. Licence, Requirements and README

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (LPPL), version 1.3 or later (http://www.latex-project.org/lppl.txt). The software has the status "maintained."

XSIM loads the packages expl3 [L3Pa], xparse [L3Pb], etoolbox [Leh15], array [MCo8] booktabs [Fea05] and translations [Nie15]. All of these packages are present on a modern and up to date TeX distribution such as TeX Live or MiKTeX so no further action should be needed. When you are using XSIM you should be using an up to date TeX distribution, anyway.

! Newer versions of XSIM may depend on newer versions of the support packages. Remember: it is always dangerous to update single packages. Always update your TeX distribution if you want an up to date version of a package. Be careful: if you're in the middle of an important project it might be better to wait with the update until you've finished the project. Every update might be breaking some things.

2. Motivation and Background

It has been quite a while since I first published exsheets [Nie17] in June 2012. Since then it has gained a user base and a little bit of popularity as the number of questions on tex.sx shows (99 at the time of writing) [var]. User questions, bug reports and feature requests improved it over the time. It still has a version number starting with a zero, though, which in my versioning system means I still consider it experimental.

This is due to several facts. It lacks a few features which I consider essential for a full version 1. For one thing it is not possible to have several kinds of exercises numbered independently. Using verbatim material such as listings inside exercises and solutions is not possible and the current workaround isn't that ideal either. One request which dates back quite a while now was to have different types of points to exercises…

All of those aren't easy to add due to the way exsheets is implemented right now. As a consequence I wanted to re-implement exsheets for a long time. This is what lead to XSIM. Internally the package works completely different.

! XSIM will be the official successor of exsheets which is now considered obsolete but will stay alive and will still receive bugfix releases. However, new features will not be added to exsheets any more.

3. How to Read the Manual

3.1. Nomenclature

Throughout this manual certain terms are used. This section explains their meaning in this manual.

collection A *collection* bundles a number of exercises of one type or all types of exercises within certain barriers in the document. Those exercise collections can be printed at any place in the document.

goal *Goals* are a certain type of properties with a numerical value the sum of which is available throughout the document.

parameter *Parameters* are options of exercise types which are the same for each exercise of a type and can be retrieved and used in exercise templates.

property *Properties* are options of exercises which are individual for each exercise and can be retrieved and used in exercise templates.

tag *Tags* are a certain type of properties with a csv list as value which can be used for selective usage of exercises.

template *Templates* are generic code frameworks which are used for typesetting XSIM's objects such as exercises, solutions, or grading tables.

3.2. Package Options

xsim has these package options:

`verbose`
Writes extensive information about what xsim is doing into the log file.

`final`
If used the exercise and solution environments will not rewrite the environment body files.

`clear-aux`
If used every time the total number of exercise changes xsim will write *less* information to the auxfile on the next run and only if the number of exercises stays stable between compilations the needed information will be written to the auxfile. *This needs more compilations until everything stabilizes but should reduce the probability of possibly faulty exercises after changes to the document.* The `final` option automatically disables this option. See also sections 5 on page 6 and B.2 on page 43.

Those options are used the usual way as package option

```
\usepackage[verbose]{xsim}
```

or as global option

```
\documentclass[verbose]{article}
```

or via the setup command:

`\xsimsetup{`⟨*options*⟩`}`
Set up xsim's package options and all other options described at other places in the manual.

3.3. Setting Options

Apart from the package options already described in section 3.2 xsim has further options. Those can be "toplevel" options or options belonging to a module.

`toplevel` = {⟨*value*⟩}
A toplevel option.

`module/sublevel` = {⟨*value*⟩} A sublevel option belonging to the module `module`

Both kinds of options are set with `\xsimsetup`:

```
\xsimsetup{
  toplevel = {value} ,
```

```
  module/sublevel = {value}
}
```

3.4. Command descriptions

Some commands do have a * symbol printed next to their names. This indicates that the command is expandable, *i. e.*, it is usable in an \edef or \write context and will expand according to its description. All other commands are engine protected, *i. e.*, in the sense of ε-TEX's \protected.

Some command name descriptions end with TF.

\SomeCommandTF⟨*arguments*⟩{⟨*true*⟩}{⟨*false*⟩}
A command with maybe some arguments and ending with the two arguments ⟨*true*⟩ and ⟨*false*⟩.

This means two things: the command is a conditional which tests something and depending on the outcome of the test leaves either the ⟨*true*⟩ argument (T) or the ⟨*false*⟩ argument (F) in the input stream. It also means two additional commands exist:

\SomeCommandT⟨*arguments*⟩{⟨*true*⟩}
The same as \SomeCommandTF but only with the ⟨*true*⟩ argument and no ⟨*false*⟩ argument.

\SomeCommandF⟨*arguments*⟩{⟨*false*⟩}
The same as \SomeCommandTF but only with the ⟨*false*⟩ argument and no ⟨*true*⟩ argument.

4. Exercises and Solutions

The two predefined environments for exercises and solutions are the following ones:

\begin{exercise}[⟨*properties*⟩]
Input and typeset an exercise. See section 7 on page 9 for details on exercise properties.

\begin{solution}[⟨*options*⟩]
Input and typeset the solution to the exercise of the previous exercise environment. See section 10 on page 22 for details on options of solutions.

```
\begin{exercise}
  A first example for an exercise.
\end{exercise}
\begin{solution}
  A first example for a solution.
\end{solution}
```

Exercise 1

A first example for an exercise.

As can be seen in the example a solution is not printed with the default setup. This can be changed using the following option.

`solution/print` = `true`|`false` Default: `false`
Set if solutions are printed or or not.

The option (belonging to the module `solution`) can either be set locally as option to the `solution` environment

```
\begin{solution}[print=true]
  A first example for a solution.
\end{solution}
```

or with the setup command for all following solutions:

```
\xsimsetup{
  solution/print = true
}
```

There is an completely analoguous option for the `exercise` environment:

`exercise/print` = `true`|`false` Default: `true`
Set if exercises are printed or or not.

More details on those two environments can be found in section 8 on page 16.

5. How the Exercise Environments Work

Both enviroments write the contents of their bodies verbatim to external files following a certain naming structure:

- ⟨*jobname*⟩`-`⟨*type*⟩`-`⟨*id*⟩`-exercise|solution-body.tex`

The name starts with the name of the job (which is the name of the document itself) followed by type and id of the corresponding exercise and then followed by the environment type. For example both enviroments from the first example have been written to files named

- `xsim_manual-exercise-1-exercise-body.tex` and
- `xsim_manual-exercise-1-solution-body.tex`, respectively.

Details on the ⟨*type*⟩ of an exercise will be given in section 6 on the following page. *The* ⟨*id*⟩ *of an exercise is a positive integer unique to each exercise environment regardless if the exercise is being printed or used at all.*

These external files are input when the respective exercise or solution is printed. An advantage of using external files is that *verbatim material is allowed* inside the environments. Each of those files contains some information about itself and where and why it was generated[1]:

```
% ------------------------------------------------------------------------
% file `xsim_manual-exercise-1-exercise-body.tex'
%   in folder `exercises/'
%
%     exercise of type `exercise' with id `1'
%
% generated by the `exercise' environment of the
%   `xsim' package v0.7 (2017/05/09)
% from source `xsim_manual' on 2017/05/09 on line 1
% ------------------------------------------------------------------------
  A first example for an exercise.
```

Arguably one downside of the approach using external files for each exercise and its solution is that your project folder will be cluttered with files. In order to deal with this somehow XSIM offers the following option:

`path` = {⟨*path name*⟩} (initially empty)
With this option a subfolder or path within the main project folder can be given. Exercises will be written to and included from this path. *The path must exist on your system before you can use it!* This document uses `path = {exercises}`.

XSIM writes a lot of stuff to the auxfile for re-using information on subsequent compilations. If you add exercises, change properties *etc.* it might happen that wrong information is staying in the auxfile and is wrongly used by XSIM. In such cases deleting the auxfile and doing a few fresh compilations may resolve your problems.
Sometimes the *existence of exercise or solution files from earlier compilations* may lead to wrong lists of exercises or solutions. In such cases it can be useful to delete all those files and doing a fresh compilation. It may be helpful to use a subfolder for those exernal files which will make deleting them a little bit easier. (Don't forget to both create the subfolder and set `path` accordingly then.)
Using the `clear-aux` option might help to reduce erroneous exercises.

6. New Exercise Types

It is easy to define new exercise environments together with a corresponding solution environment using the following command:

1. In this example the sourcecode line number is misleading as the example where the file was generated itself was an external file where the `exercise` environment indeed *was* on line 1.

`\DeclareExerciseType{`⟨*type*⟩`}{`⟨*parameters*⟩`}`
Declare a new exercise type analoguous to the `exercise` and `solution` environments.

Declaring a new exercise type will also define a new command:

`\numberof`⟨*exercise-env*⟩`s`
These commands hold the absolut number of used exercises of type ⟨*type*⟩. The meaning of ⟨*exercise-env*⟩ will become clear below when the exercise parameters are explained. It is always the same as the exercise environment name.

```
There are \numberofexercises~exercises and \numberofproblems~problem in this
manual.
```

There are 10 exercises and 1 problem in this manual.

XSIM's pre-defined environment pair has been defined as follows:

```
\DeclareExerciseType{exercise}{
  exercise-env      = exercise ,
  solution-env      = solution ,
  exercise-name     = \XSIMtranslate{exercise} ,
  solution-name     = \XSIMtranslate{solution} ,
  exercise-template = default ,
  solution-template = default
}
```

The above already is an example for almost all parameters that can (and often must) be set. Here is the complete list:

`exercise-env` = {⟨*exercise environment name*⟩}
The name for the environment used for the exercises of type ⟨*type*⟩. *This parameter is mandatory.* It can't be changed afterwards.

`solution-env` = {⟨*solution environment name*⟩}
The name for the environment used for the solutions of type ⟨*type*⟩. *This parameter is mandatory.* It can't be changed afterwards.

`exercise-name` = {⟨*exercise name*⟩}
The name of the exercises of type ⟨*type*⟩ – used for typesetting. *This parameter is mandatory.*

`solution-name` = {⟨*solution name*⟩}
The name of the solutions of type ⟨*type*⟩ – used for typesetting. *This parameter is mandatory.*

`exercise-template` = {⟨*exercise template*⟩}
The template used for typesetting the exercises of type ⟨*type*⟩. *This parameter is mandatory.* See section 12 on page 25 for details on templates.

`solution-template` = {⟨*solution template*⟩}
The template used for typesetting the exercises of type ⟨*type*⟩. *This parameter is mandatory.* See section 12 on page 25 for details on templates.

`counter` = {⟨*counter name*⟩}
The counter used for the exercises of type ⟨*type*⟩. If not explicitly set the counter with the same name as `exercise-env` is used. Otherwise the specified counter is used. This enables to have different types of exercises sharing a common counter. This parameter can't be changed afterwards.

`number` = {⟨*integer*⟩}
An internal parameter that is used to keep track of the number of exercises of a type. This parameter cannot be set or changed by the user.

It is possible to change some of the parameters after an exercise type has been defined. Those include `exercise-name`, `solution-name`, `exercise-template`, and `solution-template`:

`\SetExerciseParameter`{⟨*type*⟩}{⟨*parameter*⟩}{⟨*value*⟩}
Usable to set a single parameter to a new value.

`\SetExerciseParameters`{⟨*type*⟩}{⟨*parameters*⟩}
Set several parameters at once. ⟨*parameters*⟩ is a csv list of key/value pairs.

If you try to set an already set but fixed parameter like `exercise-env` a warning will be written to the log file. For all parameters that can be changed also options exist wich can be set via `\xsimsetup`. They are explained in section 8.2 on page 17.

! All exercises of a type use the parameters (*e. g.*, `exercise-template`) that's *currently active.* If you want exercises with a different look you should use different exercises types.

7. Exercise Properties

7.1. Predefined Properties

Exercise like the `exercise` environment and possibly others defined with `\DeclareExerciseType` have a number of predefined properties:

`id` = {⟨*integer*⟩}
Holds the internal id of an exercise. *Cannot be set by the user.*

`ID` = {⟨*text*⟩}
Holds the user id of an exercise if defined. Otherwise it is equal to `id`.

`counter` = {⟨*text*⟩}
Holds the counter value represenation of an exercise (*i. e.*, what you usually know as `\the`⟨*counter*⟩). *Cannot be set by the user.*

`counter-value` = {⟨*integer*⟩}
Holds the counter value of an exercise (*i. e.*, what you usually know as `\arabic{`⟨*counter*⟩`}`).
Cannot be set by the user.

`subtitle` = {⟨*text*⟩}
Holds the subtitle of an exercise.

`points` = {⟨*number*⟩}
Holds the reachable points of an exercise.

`bonus-points` = {⟨*number*⟩}
Holds the reachable bonus-points of an exercise.

`print` = `true`|`false`
Holds the print boolean of an exercise.

`print!` = `true`|`false`
Holds a special print boolean of an exercise, see page 16.

`use` = `true`|`false`
Holds the usage boolean of an exercise.

`use!` = `true`|`false`
Holds a special usage boolean of an exercise, see page 16.

`used` = `true`|`false`
True if an exercise has been used at least once. For an existing exercise this is only false for exercises that have been collected (*cf*. section 9 on page 19).

`tags` = {⟨*csv list of tags*⟩}
Holds the list of tags the exercise should be associated with.

`topics` = {⟨*csv list of topics*⟩}
Holds the list of topics the exercise should be associated with.

`page` = {⟨*text*⟩}
Holds the page counter value represenation of an exercise (*i. e.*, what you usually know as `\thepage`).

`page-value` = {⟨*integer*⟩}
Holds the page counter value of an exercise (*i. e.*, what you usually know as `\arabic{page}`).

`section` = {⟨*text*⟩}
Holds the section counter value represenation of an exercise (*i. e.*, what you usually know as `\thesection`).

`section-value` = {⟨*integer*⟩}
Holds the section counter value of an exercise (*i. e.*, what you usually know as `\arabic{section}`).

`chapter` = {⟨*text*⟩}
Holds the chapter counter value represenation of an exercise (*i. e.*, what you usually know as `\thechapter`). *Only if a command* `\chapter` and *a counter* `chapter` *exist.*

`chapter-value` = {⟨*integer*⟩}
Holds the chapter counter value of an exercise (*i. e.*, what you usually know as `\arabic{chapter}`). *Only if a command* `\chapter` and *a counter* `chapter` *exist.*

Some of these properties are fixed and cannot be set by the user. Those include `id`, `counter`, and `counter-value`. The others can be set using the optional argument of the exercise environment.

```
\begin{exercise}[subtitle={This is a subtitle},points=4,bonus-points=1]
  An exercise where some properties have been set.
\end{exercise}
```

Exercise 2 *This is a subtitle*

An exercise where some properties have been set.

_____/ 4 (+1) p.

7.2. Declaring Own Properties

XSIM offers the possibility to declare additional exercise properties:

`\DeclareExerciseProperty!*-`{⟨*property*⟩}
Declares the property ⟨*property*⟩.

If used with the optional `!` star a unique property is defined which means that each exercise must have a property value distinct from all other exercises (all means all – *independent from the exercise type*).

If used with the optional `*` a boolean property is defined which means that it only should get the values `true` or `false` and if used without value it gets the value `true` instead of an empty value. If any other value is used the property is set to `false`. A boolean option obviously cannot be unique. The optional `*` takes precedence over the optional `!`, *i. e.*, if both are present the property is boolean *but not* unique.

If used with the optional `-` a property is defined which won't get updated through subsequent compilation runs but is only set when the exercise is used.

`\DeclareExercisePropertyAlias`{⟨*property 1*⟩}{⟨*property 2*⟩}
Declares ⟨*property 1*⟩ to be an alias of ⟨*property 2*⟩. This means that each time ⟨*property 2*⟩ is set ⟨*property 1*⟩ will be set to the same value *unless* it has been set already. As an example: property `ID` is an alias of property `id`.

This is better demonstrated with an example:

```
\begin{exercise}
  \lipsum[4] % from package `lipsum'
  \verb+\GetExerciseProperty{id}+: \GetExerciseProperty{id} \par
  \verb+\GetExerciseAliasProperty{ID}+: \GetExerciseAliasProperty{ID} \par
  \verb+\GetExerciseProperty{ID}+: \GetExerciseProperty{ID}
\end{exercise}
\begin{exercise}[ID=foo-bar]
  \lipsum[4]
  \verb+\GetExerciseProperty{id}+: \GetExerciseProperty{id} \par
  \verb+\GetExerciseAliasProperty{ID}+: \GetExerciseAliasProperty{ID} \par
  \verb+\GetExerciseProperty{ID}+: \GetExerciseProperty{ID}
\end{exercise}
```

Exercise 3

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
`\GetExerciseProperty{id}`: 3
`\GetExerciseAliasProperty{ID}`: 3
`\GetExerciseProperty{ID}`: 3

Exercise 4

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.
`\GetExerciseProperty{id}`: 4
`\GetExerciseAliasProperty{ID}`: 4
`\GetExerciseProperty{ID}`: foo-bar

The power of properties will get more clear when reading section 12 on page 25 about templates.

7.3. A Special Kind of Property: Exercise Goals

Exercise goals are a generic concept in XSIM for exercise properties like `points` or `bonus-points`. Those are properties which can (only) get a decimal number as value the sum of which is calculated and available (after a compilation) throughout the document.

`\DeclareExerciseGoal{`⟨*goal*⟩`}`
Declare a new exercise goal named ⟨*goal*⟩ and also a property called ⟨*goal*⟩.

`\TotalExerciseTypeGoal{`⟨*type*⟩`}{`⟨*goal*⟩`}{`⟨*singular*⟩`}{`⟨*plural*⟩`}`
Get the sum of goal ⟨*goal*⟩ for all exercises of type ⟨*type*⟩. ⟨*singular*⟩ and ⟨*plural*⟩ are placed after the sum in the input stream depending on whether the sum equals 1 or not.

`\TotalExerciseTypeGoals{`⟨*type*⟩`}{`⟨*list of goals*⟩`}{`⟨*singular*⟩`}{`⟨*plural*⟩`}`
Get the sum of goal all goals in ⟨*list of goals*⟩ for all exercises of type ⟨*type*⟩. The goal names in ⟨*list of goals*⟩ must be separated with +. ⟨*singular*⟩ and ⟨*plural*⟩ are placed after the sum in the input stream depending on whether the sum equals 1 or not.

`\TotalExerciseGoal{`⟨*goal*⟩`}{`⟨*singular*⟩`}{`⟨*plural*⟩`}`
Get the sum of goal ⟨*goal*⟩ for all exercises. ⟨*singular*⟩ and ⟨*plural*⟩ are placed after the sum in the input stream depending on whether the sum equals 1 or not.

`\TotalExerciseGoals{`⟨*list of goals*⟩`}{`⟨*singular*⟩`}{`⟨*plural*⟩`}`
Get the sum of goal all goals in ⟨*list of goals*⟩ for all exercises. The goal names in ⟨*list of goals*⟩ must be separated with +. ⟨*singular*⟩ and ⟨*plural*⟩ are placed after the sum in the input stream depending on whether the sum equals 1 or not.

`\AddtoExerciseTypeGoal{`⟨*type*⟩`}{`⟨*goal*⟩`}{`⟨*value*⟩`}`
Adds ⟨*value*⟩ to the goal ⟨*goal*⟩ of exercise type ⟨*type*⟩.

`\AddtoExerciseTypeGoalPrint{`⟨*type*⟩`}{`⟨*goal*⟩`}{`⟨*value*⟩`}{`⟨*singular*⟩`}{`⟨*plural*⟩`}`
Adds ⟨*value*⟩ to the goal ⟨*goal*⟩ of exercise type ⟨*type*⟩. The value and – depending on wether the value equals 1 or not – ⟨*singular*⟩ or ⟨*plural*⟩ are left in the input stream.

`\AddtoExerciseGoal{`⟨*goal*⟩`}{`⟨*value*⟩`}`
Adds ⟨*value*⟩ to the goal ⟨*goal*⟩ of the current exercise type. (To be used within exercises.)

`\AddtoExerciseTypeGoalPrint{`⟨*goal*⟩`}{`⟨*value*⟩`}{`⟨*singular*⟩`}{`⟨*plural*⟩`}`
Adds ⟨*value*⟩ to the goal ⟨*goal*⟩ of the current exercise type. The value and – depending on wether the value equals 1 or not – ⟨*singular*⟩ or ⟨*plural*⟩ are left in the input stream. (To be used within exercises.)

`\ExerciseGoalValuePrint{`⟨*value*⟩`}{`⟨*singular*⟩`}{`⟨*plural*⟩`}`
Print ⟨*value*⟩ and – depending on wether the value equals 1 or not – ⟨*singular*⟩ or ⟨*plural*⟩.

`\printgoal{`⟨*value*⟩`}`
Print ⟨*value*⟩ according to option `goal-print`. Defined in terms of `\ExerciseGoalValuePrint`.

`\printpoints{`⟨*type*⟩`}`
Print the sum of points for all exercises of type ⟨*type*⟩ followed by an appropriate translation of the words “point” or “points”, respectively.[2] Defined in terms of `\TotalExerciseTypeGoal`.

2. See section 13 on page 38 for details on the definition and usage of language dependent words.

`\printtotalpoints`
Print the sum of points for all exercises followed by an appropriate translation of the words "point" or "points", respectively. Defined in terms of `\TotalExerciseGoal`.

`\addpoints*{`⟨*value*⟩`}`
Adds ⟨*value*⟩ to the points of the current exercise type. (To be used within exercises.) Prints the value followed by an appropriate translation of the words "point" or "points", respectively. The starred version prints nothing. Defined in terms of `\AddtoExerciseGoal` and `\AddtoExerciseGoalPrint`.

`\points{`⟨*value*⟩`}`
Print ⟨*value*⟩followed by an appropriate translation of the words "point" or "points", respectively. Defined in terms of `\ExerciseGoalValuePrint`.

`\printbonus{`⟨*type*⟩`}`
Print the sum of bonus points for all exercises of type ⟨*type*⟩ followed by an appropriate translation of the words "point" or "points", respectively. Defined in terms of `\TotalExerciseTypeGoal`.

`\printtotalbonus`
Print the sum of bonus points for all exercises followed by an appropriate translation of the words "point" or "points", respectively. Defined in terms of `\TotalExerciseGoal`.

`\addbonus*{`⟨*value*⟩`}`
Adds ⟨*value*⟩ to the bonus points of the current exercise type. (To be used within exercises.) Prints the value followed by an appropriate translation of the words "point" or "points", respectively. The starred version prints nothing. Defined in terms of `\AddtoExerciseGoal` and `\AddtoExerciseGoalPrint`.

The two existing goals are defined with

```
\DeclareExerciseGoal{points}
\DeclareExerciseGoal{bonus-points}
```

When goal values are printed the decimal number is fed to a function which can be changed using the following option:

`goal-print` = {⟨*code*⟩} Default: `#1`
How to format goal values. Use `#1` to refer to the actual number.

At last some examples for a custom command: let's say you want a command which prints the complete sum for all exercises of all exercise types of both `points` *and* `bonus-points` added up:

```
\NewDocumentCommand\printsumofpointsandbonus{}{%
  \TotalExerciseGoals{points+bonus-points}
```

```
    {\,\XSIMtranslate{point}}
    {\,\XSIMtranslate{points}}%
}
```

Here is how you could mimick the command `\totalpoints` from exsheets:

```
\NewDocumentCommand\pointsandbonus{}{%
  \TotalExerciseGoal{points}{}{}%
  \IfExerciseGoalsSumF{bonus-points}{=0}
    {\,(+\,\TotalExerciseGoal{bonus-points}{}{})}%
  \,\XSIMtranslate{points}%
}
```

7.4. A Special Kind of Property: Exercise Tags

Exercise tags are a generic concept in XSIM for exercise properties like `tags` or `topics`. Those are properties which can (only) get a csv list of strings as value. Those strings can be used to selectively use exercises. See section 8 on the next page for details on *usage* of exercises and the difference to *printing* an exercise and how to use exercise tags for selection.

`\DeclareExerciseTagging{`⟨*tag*⟩`}`

This defines an exercise tagging group named ⟨*tag*⟩. It also defines a property named ⟨*tag*⟩. In addition two options are defined: an option named ⟨*tag*⟩ which can be used for selection and an boolean option ⟨*tag*⟩`/ignore-untagged`.

The two existing tagging groups have been defined and preset with the following code:

```
\DeclareExerciseTagging{tags}
\DeclareExerciseTagging{topics}
\xsimsetup{tags/ingore-untagged=false}
```

This means that these options are available:

`tags` = {⟨*csv list of tags*⟩}

Choose the set of tags whose associated exercises should be printed.

`topics` = {⟨*csv list of topics*⟩}

Choose the set of tags whose associated exercises should be printed.

`tags/ignore-tagging` = `true`|`false` Default: `false`

If set to true exercises with no tags will be printed even if tags have been chosen with the option `tags`.

`topics/ignore-tagging` = `true`|`false` Default: `true`

If set to true exercises with no topics will be printed even if tags have been chosen with the option `topics`.

It may happen that you choose certain tags for printing and want one or two exercises to be printed or used even if they don't match the tagging criteria. For this reason two additional properties exist which can be set to an exercise:

`print!` = `true`|`false`

If set to true the exercise will be printed (and thus used) regardless of other conditions.

`use!` = `true`|`false`

If set to true the exercise will be used regardless of other conditions.

8. Using and Printing an Exercise

8.1. What the Environments do

When an exercise is started with `\begin{exercise}` (or other environments defined through `\DeclareExerciseType`) then different things happen depending on different settings:

- If the *insert mode* is active nothing happens, see section 9 on page 19 for details on this.
- Else the id integer is incremented.
- If the exercise is *used* the corresponding counter is stepped and the exercise is added to the "use list". The properties `counter` and `use` are updated accordingly.
- If an exercise is *printed* then it is also *used*. An exercise that isn't used cannot be printed. Being printed means two things: being added to the "print list" and being typeset at the position where the exercise is placed in the source file. If an exercise is *not printed but used* it means that the counter will be stepped. This can be useful for creating an exercise sheet only containing the solutions for some exercises.
- If an exercise is printed certain hooks and template code is inserted around the environment body.

```
\begin{exercise}[print=false]
  This exercise will not be printed but the exercise counter will be
  incremented nonetheless. Its solution will be printed in the list of
  solutions.
\end{exercise}
\begin{solution}
  The solution of the exercise that has not been printed.
\end{solution}
```

The schematic structure of an exercise is shown in figure 1 on the next page.

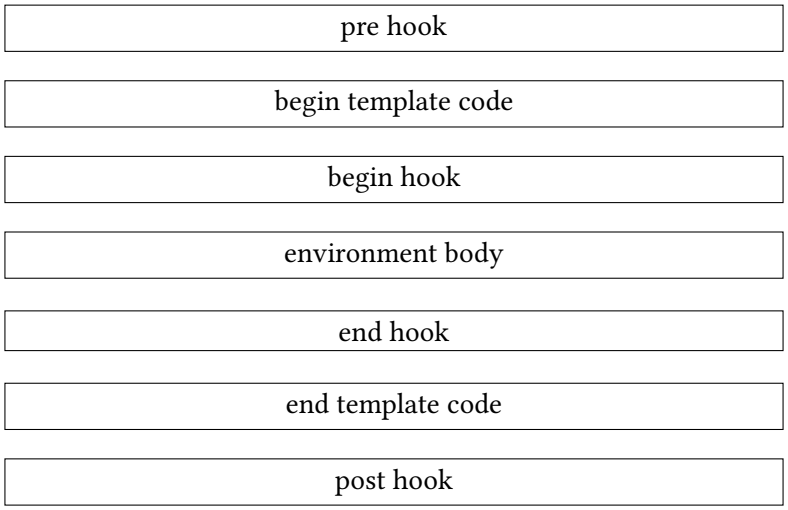

FIGURE 1: Schematic structure of an exercise or solution.

8.2. Environment Options & Hooks

For each exercise type there are the following options for both environments, the environments' names are the module names for the options (here using the "exercise" type):

`exercise/print` = `true`|`false` Default: `true`
Determines if exercises of type "exercise" are printed.

`exercise/use` = `true`|`false` Default: `true`
Determines if exercises of type "exercise" are used.

`exercise/within` = {⟨*counter*⟩} (initially empty)
Adds the exercise counter to the reset list of the counter ⟨*counter*⟩. *Beware that if the counter is a shared counter this will affect* all objects *using this counter!*

`exercise/the-counter` = {⟨*code*⟩}
An interface for redefining the counter representation command `\the`⟨*counter*⟩.

`exercise/template` = {⟨*template*⟩}
An interface for `\SetExerciseParameter{exercise}{exercise-template}{`⟨*template*⟩`}`.

`solution/template` = {⟨*template*⟩}
An interface for `\SetExerciseParameter{exercise}{solution-template}{`⟨*template*⟩`}`.

`exercise/name` = {⟨*name*⟩}
An interface for `\SetExerciseParameter{exercise}{exercise-name}{`⟨*name*⟩`}`.

`solution/name` = {⟨*name*⟩}
An interface for `\SetExerciseParameter{exercise}{solution-name}{`⟨*name*⟩`}`.

`exercise/pre-hook` = {⟨*code*⟩} (initially empty)
The code for the *pre exercise hook* for exercises of the type “exercise”.

`exercise/begin-hook` = {⟨*code*⟩} (initially empty)
The code for the *begin exercise hook* for exercises of the type “exercise”.

`exercise/end-hook` = {⟨*code*⟩} (initially empty)
The code for the *end exercise hook* for exercises of the type “exercise”.

`exercise/post-hook` = {⟨*code*⟩} (initially empty)
The code for the *post exercise hook* for exercises of the type “exercise”.

`solution/print` = `true`|`false` Default: `false`
Determines if solutions of type “exercise” are printed.

`solution/pre-hook` = {⟨*code*⟩} (initially empty)
The code for the *pre solution hook* for solutions of the type “exercise”.

`solution/begin-hook` = {⟨*code*⟩} (initially empty)
The code for the *begin solution hook* for solutions of the type “exercise”.

`solution/end-hook` = {⟨*code*⟩} (initially empty)
The code for the *end solution hook* for solutions of the type “exercise”.

`solution/post-hook` = {⟨*code*⟩} (initially empty)
The code for the *post solution hook* for solutions of the type “exercise”.

8.3. (Re-) Inserting a Certain Exercise

If you know type and `id` of an exercise you can (re-)insert every existing exercise, *i. e.*, every exercise whose external file exists.

`\printexercise`{⟨*type*⟩}{⟨*id*⟩}
Inserts the exercise of type ⟨*type*⟩ with the `id` ⟨*id*⟩.

```
1 \printexercise{exercise}{5}
```

Exercise 5

This exercise will not be printed but the exercise counter will be incremented nonetheless. Its solution will be printed in the list of solutions.

9. Collecting Exercises

9.1. Background

XSIM knows the concept of "exercise collections". A collection of exercises can be useful when you want to print a certain group of exercises several times. Each collection must have a unique name with which you can refer to the corresponding collection. A collection is realized by declaring the collection and by surrounding the exercises belonging to the collection with a certain pair of commands (this is explained in the next section).

Let's say you have several files of math exercises where one only contains geometry exercises and another only calculus exercises and so on. Surrounding the `\input` of each file with said pair of commands for a certain collection all exercises of the corresponding file now are a collection which then can be printed at once whereever you want the collection of exercises to be printed. By choosing certain tags (see section 7.4 on page 15) inside each collection you could even cherry-pick exercises from the external file.

9.2. Usage

A collection must be declared in the preamble. Using a pair of commands explained below exercises between those commands are added to the corresponding collection but not printed. After a collection is completed the collection can be printed as often as needed.

`\DeclareExerciseCollection{`⟨*collection name*⟩`}`
Define a new collection ⟨*collection name*⟩ in the document preamble.

`\collectexercisestype{`⟨*collection name*⟩`}{`⟨*exercise type*⟩`}`
Opens the collection ⟨*collection name*⟩ which now collects all exercises of type ⟨*exercise type*⟩ until the collection is closed with `\collectexercisesstop`. Collections of other types are not collected.[3]

`\collectexercises{`⟨*collection name*⟩`}`
Opens the collection ⟨*collection name*⟩ which now collects all exercises until the collection is closed with `\collectexercisesstop`.[4]

`\collectexercisesstop{`⟨*collection name*⟩`}`
Closes the collection ⟨*collection name*⟩.[5]

`\printcollection[`⟨*options*⟩`]{`⟨*collection name*⟩`}`
Prints the collection ⟨*collection name*⟩, *i. e.*, all exercises collected earlier. This command cannot be used before the corresponding collection has been closed correctly.

Valid options are the following:

`headings` = `true`|`false` Default: `false`
If true a heading for each exercise type is inserted.

3. This command starts a group with `\begingroup`!
4. This command starts a group with `\begingroup`!
5. This command ends a group with `\endgroup`!

`headings-template` = {⟨*template*⟩} Default: `collection`

The heading template used when `headings` = `{true}`.

Those options can also be set via `\xsimsetup` using the module `print-collection`.

! Please be aware that exercises are not used or printed while they are collected. Also their counters are *not stepped* during the process. This only happens when they are printed the first time, *cf*. the `used` property. At that time also the properties `page`, `section` and `chapter` are set.

The usage should be clear:

```
\collectexercises{foo}
\begin{exercise}
  This exercise is added to the collection `foo'.
\end{exercise}
\collectexercisesstop{foo}
```

Once the collection is closed it can be printed:

```
\printcollection{foo}
```

Exercise 6

This exercise is added to the collection ‘foo’.

Actually a collection can be printed *before* it is opened, too. This needs *at least* two compilations, though.

You can open several collections at the same time:

```
\collectexercises{foo}
  ...
\collectexercisestype{bar}{exercises}
  ...
\collectexercisesstop{bar}
  ...
\collectexercisesstop{foo}
```

Exercises will be added to each open collection.

There is one generic collection called “`all exercises`”. As the name already suggests it will hold all exercises. So if you say

```
\printcollection{all exercises}
```

all exercises will be printed.

If you use `\label`s inside of exercises and you print exercises more than once in your document (by reusing a collection for example) you will get

```
LaTeX Warning: There were multiply-defined labels.
```

Equally if you have environments like `\begin{equation}` which step a counter inside an exercise or solution the counter will be stepped each time the exercise is used.

At last now an example using external files, collections and tags:

```
% preamble:
% \DeclareExerciseCollection{foo-easy}
% \DeclareExerciseCollection{foo-medium}
% \DeclareExerciseTagging{difficulty}

% document:
\collectexercises{foo-easy}
\xsimsetup{difficulty=easy}
\input{foo.tex}
\collectexercisesstop{foo-easy}
% collection `foo-easy' now contains all exercises of file `foo.tex' tagged
% with `difficulty=easy'

\collectexercises{foo-medium}
\xsimsetup{difficulty=medium}
\input{foo.tex}
\collectexercisesstop{foo-medium}
% collection `foo-medium' now contains all exercises of file `foo.tex'
% tagged with `difficulty=medium'
```

The recommended usage is similar to the last example. It is safer printing a collection only once and only after it has been collected. You probably also will make sure that the exercises in a collection are unique, *i. e.*, that an exercises is not part of several collections – at least not if both collections are printed in the same document.

10. Printing Solutions

There are different commands for printing the solutions to exercises:

`\printsolutionstype*[`⟨*options*⟩`]{`⟨*exercise type*⟩`}`
Prints the solutions of all used exercises of type ⟨*exercise type*⟩. The starred version only prints the solutions of all printed exercises of type ⟨*exercise type*⟩.

`\printsolutions*[`⟨*options*⟩`]`
Prints the solutions of all used exercises of all types ordered by type. The starred version only prints the solutions of all printed exercises of all types.

`\printallsolutions*[`⟨*options*⟩`]`
Prints the solutions of all used exercises of all types ordered by appearance in the document. The starred version only prints the solutions of all printed exercises of all types.

`\printsolution[`⟨*options*⟩`]{`⟨*type*⟩`}{`⟨*id*⟩`}`
Prints the solution of the exercise of type ⟨*type*⟩ with the `id` ⟨*id*⟩.

```
1 \printsolutionstype{exercise}
```

Solutions to the Exercises

Solution 1

A first example for a solution.

Solution 5

The solution of the exercise that has not been printed.

Solution 9

Try to fill in these blanks. All of them are created by using the `\blank` command.

The options can be diveded into two groups. The ones in the first group modify the layout.

`headings` = `true`|`false` Default: `true`
If true a heading for each exercise type is inserted.

`headings-template` = `{`⟨*template*⟩`}` Default: `default`
The heading template used when `headings` = `{true}`.

The ones in the second group set conditions selecting which solutions are printed. If you combine those conditions a solution is printed if it meets either of the conditions.

`section` = `true`|`false`|⟨*integer*⟩ Default: `false`
If you set `section` = `{true}` only solutions of exercises of the current section are printed. If you set `section` = `{4}` only solutions of exercises in a section with number 4 are printed.

`chapter` = `true`|`false`|⟨*integer*⟩ Default: `false`
If you set `chapter` = `{true}` only solutions of exercises of the current chapter are printed. If you set `chapter` = `{4}` only solutions of exercises in a chapter with number 4 are printed.

`collection` = `false`|⟨*collection name*⟩ Default: `false`
If used only solutions of exercises belonging to collection ⟨*collection name*⟩ are printed.

The conditions can be combined. The following call will only print solutions from exercises in section 3 of chapter 2:

```
1 \printsolutions[chapter=2,section=3]
```

! The selection per section or per chapter relies on the *counter numbers* of the sections or chapters, respectively. This means if section numbers are reset (*e. g.* by `\chapter` or `\appendix`) and you have exercises from *different* sections with *the same section number* the solutions of *all those exercises* will be printed. This means you only should use the `section` selection when section are the top document level headings (apart from parts) and you have no exercises in the appendix. Similar considerations are valid for the `chapter` selection.

All options can also be set via `\xsimsetup` using the module `print-solutions`.

```
1 \printsolutions[section=4,headings-template=per-section]
```

Solutions to the Exercises of Section 4

Solution 1

A first example for a solution.

```
1 \printsolution{exercise}{5}
```

Solution 5

The solution of the exercise that has not been printed.

11. Grading Tables

When you create exercises it may not only be desirable to be able to add points and bonus-points to a question (see section 7.3 on page 12 about exercise goals) but also to be able to output a grading table. XSIM has built-in means for this.

`\gradingtable[`⟨*options*⟩`]`
Print a grading table.

Valid options for this command are

`template` = {⟨*template*⟩} Default: `default`
Choose the template used for the grading table.

`type` = {⟨*exercise type*⟩} (initially empty)
Choose the exercise type for which the table is printed.

Both option defaults can be changed with `\xsimsetup` setting the options using `grading-table`:

```
\xsimsetup{
  grading-table/template = default*
}
```

An example:

```
\gradingtable[type=exercise]
```

Exercise	Points	reached
1	0	
2	4	
3	0	
4	0	
5	0	
6	0	
7.	2.5	
8.	2.5	
9	0	
10	0	
total	9	

Or using the "`default*`" template:

```
1 \gradingtable[template=default*,type=exercise]
```

Exercise	1	2	3	4	5	6	7.	8.	9	10	total
Points	0	4	0	0	0	0	2.5	2.5	0	0	9
reached											

Available templates and how to define new ones are explained in sections 12.3.3 on page 30 and 12.4 on page 31. XSIM per default provides two templates "`default`" and "`default*`", the first one has a vertical layout, the second a horizontal layout. Both templates can be used per type like in the examples above or for all types at once by leaving the specification `type` away:

```
1 \gradingtable
```

	Points	reached
Exercise 1	0	
Exercise 2	4	
Exercise 3	0	
Exercise 4	0	
Exercise 5	0	
Exercise 6	0	
Exercise 7.	2.5	
Exercise 8.	2.5	
Exercise 9	0	
Exercise 10	0	
Problem 1	5	
total	14	

12. Styling the Exercises – Templates

12.1. Background

Whenever XSIM outputs something to be typeset it uses so-called templates for the task. XSIM knows of three different kinds of templates:

- environment templates (see section 12.3.1 on page 30),
- heading templates (see section 12.3.2 on page 30) and
- grading table templates (see section 12.3.3 on page 30)

The most important one for the styling of the exercises are the environment templates. Those templates give you complete control over the look and arrangement of an exercise. To be able to do this XSIM provides a large number of commands which can be used only inside template definitions.[6] Those commands are explained in the next section. Their usage will hopefully become clear in the examples in section 12.4 on page 31. Having full control over the layout comes at a price: you need to be able to program yourself in order to achieve certain layouts.[7]

12.2. Commands for Usage in Template Definitions

12.2.1. Goals

`\IfExerciseGoalTF{⟨goal⟩}{⟨relation and value⟩}{⟨true⟩}{⟨false⟩}`
Checks the sum of goal ⟨*goal*⟩ against ⟨*relation and value*⟩.

`\IfExerciseGoalSingularTF{⟨goal⟩}{⟨true⟩}{⟨false⟩}`
Checks if the value of the goal ⟨*goal*⟩ of the current exercise equals 1. This is the same as `\IfExerciseGoalTF{⟨goal⟩}{=1}{⟨true⟩}{⟨false⟩}`.

`\IfExerciseTypeGoalsSumTF{⟨type⟩}{⟨list of goals⟩}{⟨relation and value⟩}{⟨true⟩}{⟨false⟩}`
Ckecks the sum of all goals in ⟨*list of goals*⟩ for the exercises of type ⟨*type*⟩ against ⟨*relation and value*⟩.

`\IfExerciseGoalsSumTF{⟨type⟩}{⟨list of goals⟩}{⟨relation and value⟩}{⟨true⟩}{⟨false⟩}`
Ckecks the sum of all goals in ⟨*list of goals*⟩ for all exercises of all types against ⟨*relation and value*⟩.

`\TotalExerciseTypeGoal{⟨goal⟩}{⟨type⟩}{⟨singular⟩}{⟨plural⟩}`
Print the sum of goal ⟨*goal*⟩ for the exercises of type ⟨*type*⟩ and append ⟨*singular*⟩ or ⟨*plural*⟩ depending on wether the sum equals 1 or not.

`\TotalExerciseGoal{⟨goal⟩}{⟨singular⟩}{⟨plural⟩}`
Print the sum of goal ⟨*goal*⟩ for all exercises of all types and append ⟨*singular*⟩ or ⟨*plural*⟩ depending on wether the sum equals 1 or not.

12.2.2. Properties

* `\IfExercisePropertyExistTF{⟨property⟩}{⟨true⟩}{⟨false⟩}`
Tests wether an exercise property with the name ⟨*property*⟩ is defined.

`\IfExercisePropertySetTF{⟨property⟩}{⟨true⟩}{⟨false⟩}`
Tests wether the exercise property ⟨*property*⟩ has been set for the current exercise.

* `\GetExerciseProperty{⟨property⟩}`
Retrieves the value of the property ⟨*property*⟩ for the current exercise.

6. The last sentence is wrong: those commands can be used anywhere but most of them only give useful results inside of templates.
7. I plan to incorporate the most common layouts – and maybe some fancy ones, too – in the examples section 12.4 on page 31 but at the time of writing this is still up in the air.

`\GetExercisePropertyTF{`⟨*property*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`
Tests wether the exercise property ⟨*property*⟩ has been set for the current exercise. Inside the ⟨*true*⟩ branch you can refer to the retrieved value either with `#1` or with `\PropertyValue`

`\SetExerciseProperty{`⟨*type*⟩`}{`⟨*id*⟩`}{`⟨*property*⟩`}{`⟨*value*⟩`}`
Set the property ⟨*property*⟩ of exercise of type ⟨*type*⟩ and id ⟨*id*⟩ to ⟨*value*⟩.

* `\IfExerciseBooleanPropertyTF{`⟨*property*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`
Checks wether the boolean property ⟨*property*⟩ has value `true` or ⟨*false*⟩ and leaves the corresponding argument in the input stream. Gives an error if ⟨*property*⟩ is not a boolean property.

* `\GetExerciseAliasProperty{`⟨*property*⟩`}`
Retrieves the value of the property of which ⟨*property*⟩ is an alias of for the current exercise.

`\SaveExerciseProperty{`⟨*property*⟩`}`⟨*macro*⟩
Saves the value of the property ⟨*property*⟩ for the current exercise in macro ⟨*macro*⟩.

`\GlobalSaveExerciseProperty`
Globally saves the value of the property ⟨*property*⟩ for the current exercise in macro ⟨*macro*⟩.

`\ExercisePropertyIfSetTF{`⟨*type*⟩`}{`⟨*id*⟩`}{`⟨*property*⟩`}{`⟨*true*⟩`}{`⟨*false*⟩`}`
Test if the property ⟨*property*⟩ has been set for the exercise of type ⟨*type*⟩ with id ⟨*id*⟩.

* `\ExercisePropertyGet{`⟨*type*⟩`}{`⟨*id*⟩`}{`⟨*property*⟩`}`
Retrieves the value of the property ⟨*property*⟩ for the exercise of type ⟨*type*⟩ with id ⟨*id*⟩.

* `\ExercisePropertyGetAlias{`⟨*type*⟩`}{`⟨*id*⟩`}{`⟨*property*⟩`}`
Retrieves the value of the property of which ⟨*property*⟩ is an alias of for the exercise of type ⟨*type*⟩ with id ⟨*id*⟩.

`\ExercisePropertySave{`⟨*type*⟩`}{`⟨*id*⟩`}{`⟨*property*⟩`}`⟨*macro*⟩
Saves the value of the property ⟨*property*⟩ for the exercise of type ⟨*type*⟩ with id ⟨*id*⟩ in macro ⟨*macro*⟩.

`\ExercisePropertyGlobalSave{`⟨*type*⟩`}{`⟨*id*⟩`}{`⟨*property*⟩`}`⟨*macro*⟩
Globally saves the value of the property ⟨*property*⟩ for the exercise of type ⟨*type*⟩ with id ⟨*id*⟩ in macro ⟨*macro*⟩.

12.2.3. Parameters

* `\GetExerciseParameter{`⟨*parameter*⟩`}`
Retrieves the value of the parameter ⟨*paramater*⟩ for the current exercise.

* `\GetExerciseName`
Retrieves the value of the parameter `exercise-name` for the current exercise or of the parameter `solution-name` for the current solution.

* `\ExerciseParameterGet{`⟨*type*⟩`}{`⟨*id*⟩`}{`⟨*parameter*⟩`}`
Retrieves the value of the parameter ⟨*parameter*⟩ for the exercise of type ⟨*type*⟩ with id ⟨*id*⟩.

12.2.4. Tags

`\ForEachExerciseTag{⟨type⟩}{⟨code⟩}`
Loops over all tags of tag type ⟨*type*⟩ for the current exercise applying ⟨*code*⟩ each time. Inside ⟨*code*⟩ you can refer to the corresponding tag with `#1`.

`\ListExerciseTags{⟨type⟩}{⟨between⟩}`
Lists all tags of tag type ⟨*type*⟩ for the current exercise using ⟨*between*⟩ as a separator.

`\UseExerciseTags{⟨type⟩}{⟨between two⟩}{⟨between⟩}{⟨between last two⟩}`
Lists all tags of tag type ⟨*type*⟩ for the current exercise using ⟨*between*⟩ as a separator and ⟨*between last two*⟩ as separator between the last two tags of the list. If the list only consists of two tags ⟨*between two*⟩ is used as separator.

12.2.5. Further Commands for Usage in Template Definitions

* `\ExerciseType`
Can be used to refer to the current exercise type.

* `\ExerciseID`
Can be used to refer to the current exercise id.

* `\ExerciseCollection`
Can be used in certain templates to refer to the collection that is currently inserted.

* `\numberofusedexercises`
Holds the total number of used exercises. Useful in table template definitions.

* `\ExerciseTableType{⟨code⟩}`
In table template definitions this macro either expands to the given exercise type or – if no type has been given – to ⟨*code*⟩.

* `\IfInsideSolutionTF{⟨true⟩}{⟨false⟩}`
Tests if the template is used inside a solution environment or not.

`\ForEachPrintedExerciseByType{⟨code⟩}`
Loops over each *printed* exercise ordered by the exercise types and within each type by id. Inside ⟨*code*⟩ you can refer to several properties of the corresponding exercise:

- `#1`: the type of the exercise
- `#2`: the id of the exercise
- `#3`: the counter of the exercise
- `#4`: the subtitle of the exercise
- `#5`: the points of the exercise
- `#6`: the bonus points of the exercise

`\ForEachUsedExerciseByType{`⟨*code*⟩`}`
Loops over each *used* exercise ordered by the exercise types and within each type by id. Inside ⟨*code*⟩ you can refer to several properties of the corresponding exercise:

- `#1`: the type of the exercise
- `#2`: the id of the exercise
- `#3`: the counter of the exercise
- `#4`: the subtitle of the exercise
- `#5`: the points of the exercise
- `#6`: the bonus points of the exercise

`\ForEachPrintedExerciseByID`
Loops over each *printed* exercise order by the exercise id. Inside ⟨*code*⟩ you can refer to several properties of the corresponding exercise:

- `#1`: the type of the exercise
- `#2`: the id of the exercise
- `#3`: the counter of the exercise
- `#4`: the subtitle of the exercise
- `#5`: the points of the exercise
- `#6`: the bonus points of the exercise

`\ForEachUsedExerciseByID`
Loops over each *used* exercise order by the exercise id. Inside ⟨*code*⟩ you can refer to several properties of the corresponding exercise:

- `#1`: the type of the exercise
- `#2`: the id of the exercise
- `#3`: the counter of the exercise
- `#4`: the subtitle of the exercise
- `#5`: the points of the exercise
- `#6`: the bonus points of the exercise

* `\XSIMtranslate{`⟨*keyword*⟩`}`
Delivers the translation of ⟨*keyword*⟩ according to the current document language (in the meaning of a babel [Bra16] or polyglossia [Cha15] language). Existing keywords and keyword translations (and how to add new ones) are explained in section 13 on page 38.

`\XSIMexpandcode{`⟨*code*⟩`}`
Expands ⟨*code*⟩ like `\edef` does and leaves the result in the input stream.

* `\XSIMifchapterTF{`⟨*true*⟩`}{`⟨*false*⟩`}`
Returns ⟨*true*⟩ if both a macro `\chapter` and a counter `chapter` are defined and ⟨*false*⟩ otherwise.

\XSIMmixedcase{⟨*code*⟩}
Converts the full expansion[8] of ⟨*code*⟩ to mixed case:
\XSIMmixedcase{this is some text} This is some text
This command expands ⟨*code*⟩ *before converting it.*

\XSIMputright⟨*macro*⟩{⟨*code*⟩}
Extends the macro definition of ⟨*macro*⟩ with ⟨*code*⟩ putting it to the right. This is more or less a local version of the LaTeX kernel macro \g@addto@macro.

* \XSIMifeqTF{⟨*code 1*⟩}{⟨*code 2*⟩}{⟨*true*⟩}{⟨*false*⟩}
Checks if the full expansion[8] of ⟨*code 1*⟩ and ⟨*code 2*⟩ is the same tokenlist.

* \XSIMifblankTF{⟨*code*⟩}{⟨*true*⟩}{⟨*false*⟩}
Checks if the full expansion[8] of ⟨*code*⟩ is blank (*i. e.*, if it is empty or only consists of spaces).

12.3. Declaring Templates

12.3.1. Environment Templates

\DeclareExerciseEnvironmentTemplate{⟨*name*⟩}{⟨*begin code*⟩}{⟨*end code*⟩}
Declare the environment template ⟨*name*⟩.

Environment templates are used by the exercise and solution environments. Those are the templates set with the parameters exercise-template and solution-template.
The predefined template is called "default", see section 12.4.1 on the following page.

12.3.2. Heading Templates

\DeclareExerciseHeadingTemplate{⟨*name*⟩}{⟨*code*⟩}
Declare the heading template ⟨*name*⟩.

Heading templates are used by \printsolutions, \printsolutionstype and \printcollection. Those are the templates set with the option headings-template of the modules print-solutions and print-collection.
The predefined templates are "default", "collection", "per-section" and "per-chapter" see section 12.4.5 on page 34.

12.3.3. Grading Table Templates

\DeclareExerciseTableTemplate{⟨*name*⟩}{⟨*code*⟩}
Declare the grading table template ⟨*name*⟩.

Table templates are used by \gradingtable. Those are the templates set with the option template of module grading-table
The predefined templates are "default" and "default", see sections 12.4.6 on page 35 and 12.4.7 on page 36.

8. This is a \romannumeral expansion [Flo].

12.4. Examples

12.4.1. The `default` Exercise Template

Below the definition of the `default` exercise template provided by XSIM is shown:

```
\DeclareExerciseEnvironmentTemplate{default}{%
  \subsection*
    {%
      \XSIMmixedcase{\GetExerciseName}\nobreakspace
      \GetExerciseProperty{counter}%
      \IfInsideSolutionF
        {%
          \GetExercisePropertyT{subtitle}
            { {\normalfont\itshape\PropertyValue}}%
        }%
    }
  \GetExercisePropertyT{points}
    {%
      \marginpar
        {%
          \IfInsideSolutionF{\rule{1.2cm}{1pt}\slash}%
          \PropertyValue
          \GetExercisePropertyT{bonus-points}{~(+\PropertyValue)}%
          ~\XSIMtranslate {point-abbr}%
        }%
    }%
}
{}
```

12.4.2. A New Exercise Type Using tcolorbox

Let's say we want exercises to be put in a `tcolorbox`. We want a bold title and. if given, an italic subtitle. Exercises should also have the points after the subtitle in parentheses if given. Let's also say we want those to be an additional exercise type in addition to the ones XSIM already provides. This is shown with the following code which is also how the problems in this manual have been defined:

```
\DeclareExerciseEnvironmentTemplate{tcolorbox}
  {%
    \tcolorbox[
      colback = red!5!white ,
      colframe = red!75!black ,
      colbacktitle = yellow!50!red ,
      coltitle = red!25!black ,
      breakable ,
```

```
      drop shadow ,
      beforeafter skip = .5\baselineskip ,
      title =
        \textbf{\GetExerciseName~\GetExerciseProperty{counter}}%
        \GetExercisePropertyT{subtitle}{ \textit{\PropertyValue}}%
        \IfInsideSolutionF{%
          \GetExercisePropertyT{points}{ % notice the space
            (%
              \PropertyValue
              \IfExerciseGoalSingularTF{points}
                {\XSIMtranslate{point}}
                {\XSIMtranslate{points}}%
            )%
          }%
        }%
    ]%
  }
  {\endtcolorbox}

\DeclareExerciseType{problem}{
  exercise-env = problem ,
  solution-env = answer ,
  exercise-name = Problem ,
  solution-name = Answer ,
  exercise-template = tcolorbox ,
  solution-template = tcolorbox
}
```

See it in action:

```
\begin{problem}[subtitle=My subtitle,points=5]
  This is a problem using a subtitle and points.
\end{problem}
\begin{answer}
  This is the answer to problem~\GetExerciseProperty{counter}.
\end{answer}
```

Problem 1 *My subtitle* (5points)

This is a problem using a subtitle and points.

12.4.3. Mimicking exsheets' `runin` Template

The following example shows how you could mimick exsheets' `runin` template. The outcome isn't exactly the same since exsheets doesn't use `\marginpar` but the result should look very similar. A safer definition would use a real sectioning command for the title.

```
\usepackage{needspace}
\DeclareExerciseEnvironmentTemplate{runin}
  {%
    \par\vspace{\baselineskip}
    \Needspace*{2\baselineskip}
    \noindent
    \textbf{\XSIMmixedcase{\GetExerciseName}~\GetExerciseProperty{counter}}%
    \GetExercisePropertyT{subtitle}{ \textit{#1}} %
    \GetExercisePropertyT{points}{%
      \marginpar{%
        \PropertyValue
        \GetExercisePropertyT{bonus-points}{+\PropertyValue}%
        \,\IfExerciseGoalSingularTF{points}
            {\XSIMtranslate{point}}
            {\XSIMtranslate{points}}%
      }%
    }%
  }
  {}
```

See it in action:

```
\xsimsetup{exercise/template=runin}
\renewcommand*\theexercise{\arabic{exercise}.}
\begin{exercise}[subtitle=exsheets' runin,points=2.5]
  \lipsum[4]
\end{exercise}
```

Exercise 7. *exsheets' runin* Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula. 2.5 points

12.4.4. Mimicking exsheets' `margin` Template

The following example shows how you could mimick exsheets' `margin` template.

```
\usepackage{needspace}
\DeclareExerciseEnvironmentTemplate{margin}
  {%
    \par\vspace{\baselineskip}
    \Needspace*{2\baselineskip}
    \noindent
    \llap{%
      \smash{%
        \tabular[t]{@{}r@{}}
          \textbf{\XSIMmixedcase{\GetExerciseName}~\GetExerciseProperty{
  counter}}
        \IfExercisePropertySetT{points}{%
          \tabularnewline
          (%
            \GetExerciseProperty{points}%
            \GetExercisePropertyT{bonus-points}{+#1}%
            \,\XSIMtranslate{point-abbr}%
          )%
        }%
      \endtabular
    } % notice the space
  }%
}
{}
```

See it in action:

```
\xsimsetup{exercise/template=margin}
\renewcommand*\theexercise{\arabic{exercise}.}
\begin{exercise}[subtitle=exsheets' margin,points=2.5]
  \lipsum[4]
\end{exercise}
```

Exercise 8. (2.5 p.) Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

12.4.5. The Headings Templates

XSIM defines four heading templates which only differ by which text they output:

```
\DeclareExerciseHeadingTemplate{default}
  {\section*{\XSIMtranslate{default-heading}}}
\DeclareExerciseHeadingTemplate{collection}
  {\section*{\XSIMtranslate{collection-heading}}}
\DeclareExerciseHeadingTemplate{per-section}
  {\section*{\XSIMtranslate{per-section-heading}}}
\DeclareExerciseHeadingTemplate{per-chapter}
  {\section*{\XSIMtranslate{per-chapter-heading}}}
```

Section 13 on page 38 shows how the translations are defined.

12.4.6. The `default` Table Template

This template is the one used for grading tables per default. It has a vertical layout.

```
\DeclareExerciseTableTemplate{default}{%
  \XSIMputright\ExerciseTableCode{%
    \toprule
    \XSIMifblankTF{\ExerciseType}
      {}
      {\XSIMmixedcase{\GetExerciseParameter{exercise-name}}}
    &
    \XSIMmixedcase{\XSIMtranslate{points}} &
    \XSIMtranslate{reached} \\
    \midrule
  }%
  \ForEachUsedExerciseByType{%
    \XSIMifeqTF{#1}{\ExerciseTableType{#1}}
      {%
        \XSIMifblankTF{\ExerciseType}
          {%
            \XSIMputright\ExerciseTableCode{%
              \XSIMmixedcase{\ExerciseParameterGet{#1}{exercise-name} }%
            }%
          }
          {}%
        \XSIMputright\ExerciseTableCode
          {#3 & \XSIMifblankTF{#5}{\printgoal{0}}{\printgoal{#5}} & \\ }%
      }
      {}%
  }
  \XSIMputright\ExerciseTableCode{%
    \midrule
    \XSIMtranslate{total} &
    \XSIMifblankTF{\ExerciseType}
      {\TotalExerciseGoal{points}{}{}}
```

```
      {\TotalExerciseTypeGoal{\ExerciseType}{points}{}{}} &
    \\ \bottomrule
  }%
  \XSIMexpandcode{%
    \noexpand\begin{tabular}{\XSIMifblankTF{\ExerciseType}{l}{c}cc}
      \noexpand\ExerciseTableCode
    \noexpand\end{tabular}%
  }%
}
```

The part

```
\XSIMifblankTF{\ExerciseType}{ ... }{ ... }
```

repeatedly checks if an exercise type has been given for the table. This makes it possible to design the table differently if it is for one exercise type only (the `true` case) or for all exercise types (the `false` case). `\ExerciseTableType{`⟨*code*⟩`}` either expands to the given exercise type or to ⟨*code*⟩.

12.4.7. The `default*` Table Template

The second of the predefined grading table templates. It has a horizontal layout.

! If you have a lot of exercises the width of a table with this layout may exceed the text width of the document!

```
\DeclareExerciseTableTemplate{default*}{%
  \XSIMputright\ExerciseTableCode{%
    \toprule
    \XSIMifblankTF{\ExerciseType}
      {}
      {\XSIMmixedcase{\GetExerciseParameter{exercise-name}}}
      &%
  }%
  \ForEachUsedExerciseByType{%
    \XSIMifeqTF {#1} { \ExerciseTableType {#1} }
      {
        \XSIMifblankTF{\ExerciseType}
          {%
            \XSIMputright\ExerciseTableCode{%
              \XSIMmixedcase{\ExerciseParameterGet{#1}{exercise-name} }%
            }%
          }
```

```
          {}%
        \XSIMputright\ExerciseTableCode{#3 &}
      }
      {}%
  }%
  \XSIMputright\ExerciseTableCode{%
    \XSIMtranslate{total} \\
    \midrule
    \XSIMmixedcase{\XSIMtranslate{points}} &
  }%
  \ForEachUsedExerciseByType{%
    \XSIMifeqTF{#1}{\ExerciseTableType{#1}}
      {%
        \XSIMputright\ExerciseTableCode{%
          \XSIMifblankTF{#5}{\printgoal{0}}{\printgoal{#5}} &}%
      }
      {}%
  }%
  \XSIMputright\ExerciseTableCode{%
    \XSIMifblankTF{\ExerciseType}
      {\TotalExerciseGoal{points}{}{}}
      {\TotalExerciseTypeGoal{\ExerciseType}{points}{}{}}%
    \\ \midrule
    \XSIMtranslate{reached} &%
  }%
  \ForEachUsedExerciseByType{%
    \XSIMifeqTF{#1}{\ExerciseTableType{#1}}
      {\XSIMputright\ExerciseTableCode{&}}
      {}%
  }%
  \XSIMputright\ExerciseTableCode{ \\ \bottomrule }%
  \def\numberofcolumns{%
    \XSIMifblankTF{\ExerciseType}
      {\numberofusedexercises}
      {\csname numberof \ExerciseType s\endcsname}%
  }%
  \XSIMifeqF{\numberofcolumns}{0}
    {%
      \begin{tabular}{l*{\numberofcolumns}{c}c}
        \ExerciseTableCode
      \end {tabular}%
    }%
}
```

The part

```
\XSIMifblankTF{\ExerciseType}{ ... }{ ... }
```

repeatedly checks if an exercise type has been given for the table. This makes it possible to design the table differently if it is for one exercise type only (the `true` case) or for all exercise types (the `false` case). `\ExerciseTableType{`⟨*code*⟩`}` either expands to the given exercise type or to ⟨*code*⟩.

13. Exercise Translations

`\DeclareExerciseTranslation{`⟨*keyword*⟩`}{`⟨*language*⟩`}{`⟨*translation*⟩`}`
Declare the translation of ⟨*keyword*⟩ for language ⟨*language*⟩.

`\DeclareExerciseTranslations{`⟨*keyword*⟩`}{`⟨*translations*⟩`}`
Declare the translations of ⟨*keyword*⟩ for several languages at once. See an example of the usage below.

* `\XSIMtranslate{`⟨*keyword*⟩`}`
Delivers the translation of ⟨*keyword*⟩ according to the current document language (in the meaning of a babel [Bra16] or polyglossia [Cha15] language).

`\ForEachExerciseTranslation{`⟨*code*⟩`}`
Loops over all translations of all keywords known to XSIM. Inside ⟨*code*⟩ you can refer to the keyword with `#1`, to the language with `#2`, and to the translation with `#3`.

As an example how to use `\DeclareExerciseTranslations` here is how the translations for `exercise` have been defined:

```
\DeclareExerciseTranslations{exercise}{
  Fallback = exercise ,
  English  = exercise ,
  French   = exercice ,
  German   = \"Ubung
}
```

Table 1 shows all existing keywords with all predefined translations.

Table 1: Translation keywords predefined by XSIM.

keyword	language	translation
exercise	Fallback	exercise
exercise	English	exercise
exercise	French	exercice

continues

keyword	language	translation
exercise	German	\"Ubung
question	Fallback	question
question	English	question
question	French	question
question	German	Aufgabe
solution	Fallback	solution
solution	English	solution
solution	French	solution
solution	German	L\"osung
point-abbr	Fallback	p.
point-abbr	English	p.
point-abbr	French	p.
point-abbr	German	P.
point	Fallback	point
point	English	point
point	French	point
point	German	Punkt
points	Fallback	points
points	English	points
points	French	points
points	German	Punkte
reached	Fallback	reached
reached	English	reached
reached	French	atteint
reached	German	erreicht
total	Fallback	total
total	English	total
total	French	totalement
total	German	insgesamt
default-heading	Fallback	\XSIMmixedcase {\GetExerciseParameter {solution-name}s} to the \XSIMmixedcase {\GetExerciseParameter {exercise-name}s}
default-heading	English	\XSIMmixedcase {\GetExerciseParameter {solution-name}s} to the \XSIMmixedcase {\GetExerciseParameter {exercise-name}s}
default-heading	German	\XSIMmixedcase {\GetExerciseParameter {solution-name}en} zu den \XSIMmixedcase {\GetExerciseParameter {exercise-name}en}
collection-heading	Fallback	\XSIMmixedcase {\GetExerciseParameter {exercise-name}s}

continues

keyword	language	translation
collection-heading	English	\XSIMmixedcase {\GetExerciseParameter {exercise-name}s}
collection-heading	German	\XSIMmixedcase {\GetExerciseParameter {exercise-name}en}
per-section-heading	Fallback	\XSIMmixedcase {\GetExerciseParameter {solution-name}s} to the \XSIMmixedcase {\GetExerciseParameter {exercise-name}s} of Section\nobreakspace \ExerciseSection
per-section-heading	English	\XSIMmixedcase {\GetExerciseParameter {solution-name}s} to the \XSIMmixedcase {\GetExerciseParameter {exercise-name}s} of Section\nobreakspace \ExerciseSection
per-section-heading	German	\XSIMmixedcase {\GetExerciseParameter {solution-name}en} zu den \XSIMmixedcase {\GetExerciseParameter {exercise-name}en} in Abschnitt\nobreakspace \ExerciseSection
per-chapter-heading	Fallback	\XSIMmixedcase {\GetExerciseParameter {solution-name}s} to the \XSIMmixedcase {\GetExerciseParameter {exercise-name}s} of Chapter\nobreakspace \ExerciseChapter
per-chapter-heading	English	\XSIMmixedcase {\GetExerciseParameter {solution-name}s} to the \XSIMmixedcase {\GetExerciseParameter {exercise-name}s} of Chapter\nobreakspace \ExerciseChapter
per-chapter-heading	German	\XSIMmixedcase {\GetExerciseParameter {solution-name}en} zu den \XSIMmixedcase {\GetExerciseParameter {exercise-name}en} in Kapitel\nobreakspace \ExerciseChapter

14. Cloze Tests and Blank Lines

Similar to exsheets XSIM provides a command \blank:

\blank*[⟨*options*⟩]{⟨*text to be filled in*⟩}
Creates a blank in normal text or in an exercise but fills the text of its argument if inside a solution. If used at the *begin of a paragraph* \blank will do two things: it will set the linespread according to an option explained below and will insert \par after the lines. If you don't want that use the starred version.

Those are the options for customization:

blank/blank-style = {⟨*code*⟩} Default: \underline{#1}
Instructions for typesetting the blank cloze. Refer to the filled in space with #1.

`blank/filled-style` = {⟨*code*⟩} Default: `\underline{#1}`
Instructions for typesetting the filled cloze. Refer to the filled in text with `#1`

`style` = {⟨*code*⟩}
Shortcut for setting both `blank-style` and `filled-style` at once.

`blank/scale` = {⟨*decimal number*⟩} Default: `1`
Scales the blank to ⟨*decimal number*⟩ times its natural width.

`blank/width` = {⟨*dim*⟩} (initially empty)
Sets the blank to a width of ⟨*dim*⟩. This takes precendence over `scale`.

`blank/linespread` = {⟨*decimal number*⟩} Default: `1`
Set the linespread for the blank lines. This only has an effect if `\blank` is used at the begin of a paragraph.

`blank/line-increment` = {⟨*dim*⟩} Default: `1pt`
The blank line is built in multiples of this value. If the value is too large you may end up with uneven lines. If the value is too small you may end up with a non-ending compilation. Experiment with values to find the suiting one for your use case.

`blank/line-minimum-length` = {⟨*dim*⟩} Default: `2em`
The minimal length a line must have before it is built step by step.

```
This is a \blank{blank} outside in normal text.
\begin{exercise}
  Try to fill in \blank[width=4cm]{these} blanks.  All of them
  \blank{are created} by using the \cs{blank} \blank{command}.
\end{exercise}
\xsimsetup{blank/filled-style=\textcolor{red}{#1}}
\begin{solution}[print]
  Try to fill in \blank[width=4cm]{these} blanks.  All of them
  \blank{are created} by using the \cs{blank} \blank{command}.
\end{solution}
```

This is a _____ outside in normal text.

Exercise 9

Try to fill in _____ blanks. All of them _____
by using the `\blank` _____.

Solution 9

Try to fill in these blanks. All of them are created by using the `\blank` command.

A number of empty lines are easily created by setting the `width` option:

```
Write up the pros and cons of \xsim\ over \pkg{exsheets}:

\blank[width=4.8\linewidth,linespread=1.5]{}
```

Write up the pros and cons of XSIM over exsheets:

A. Future Plans

XSIM is complete in so far as it is perfectly usable to create exams or exercise and solution sections in books with the most freedom in layout already. But still there are features which would be useful additions. Below I list all ideas that I currently plan to add to XSIM:

- random selection of exercises from a collection.
- a document class `xsim-exam` for creating exams; this class should itself feature the possibility of creating different versions of an exam, maybe already provide multiple choice questions and so on; one could also think about automatic creation of running headers and footers, *i. e.*, means for changing the layout of the exam; following the spirit of XSIM this should probably be done using templates as well.

I am very open to suggestions regarding features, both in general and specifically regarding the document class.

B. FAQ & How to...

This section serves as a kind of gallery showing solutions to common problems. I expect this section to grow over the years. Some examples especially regarding other layouts are also shown in example files added to this package.

B.1. ...Know if XSIM Needs Another Compilation?

If XSIM wants you to recompile your document it writes the following to the logfile:

```
**********************************************
* xsim warning: "rerun"
```

```
*
* Exercise properties may have changed. Rerun to get them synchronized.
**********************************************
```

So just check the logfile regularly (which you should be doing anyway) and keep your eyes open.

B.2. …Resolve Getting Repeatedly Wrong Exercise Properties or Wrong Exercise Lists?

XSIM writes a lot of stuff to the auxfile for re-using information on subsequent compilations. If you add exercises, change properties *etc.* it might happen that wrong information is staying in the auxfile and is wrongly used by XSIM. In such cases deleting the auxfile and doing a few fresh compilations may resolve your problems.

Sometimes the *existence of exercise or solution files from earlier compilations* may lead to wrong lists of exercises or solutions. In such cases it can be useful to delete all those files and doing a fresh compilation. It may be helpful to use a subfolder for those exernal files which will make deleting them a little bit easier. (Don't forget to both create the subfolder and set `path` accordingly then.)

Using the `clear-aux` option might help to reduce erroneous exercises.

B.3. …Resolve Strange Errors After Updating?

XSIM writes a lot of stuff to the auxfile. An update may well change how this is done so deleting the auxfile and doing a few fresh compilations may resolve your problems.

B.4. `! TeX capacity exceeded, sorry [text input levels=15].` Why?

You probably tried to use an exercise or solution in a macro of some sort. This generally will fail.[9] A minimal example to reproduce the problem:

```
\documentclass{article}
\usepackage{xsim}
\begin{document}
\def\x{%
\begin{exercise}
\end{exercise}%
}\x
\end{document}
```

But there should never be the need to hide the environments inside of a macro, anyway.

9. The reasons are not entirely clear to me.

B.5. …Put a Star (or Another Symbol) in Headings of Exercises That Are Special?

The code below shows one possible modification of an exercise template which allows to easily create bonus exercises:

```
% preamble:
\usepackage{amsymb}
% declare boolean property:
\DeclareExerciseProperty*{bonus}
\DeclareExerciseEnvironmentTemplate{bonus}
  {%
    \subsection*
      {%
        % test for boolean property and insert star symbol if true:
        \IfExerciseBooleanPropertyT{bonus}{\llap{$\bigstar$ }Bonus }%
        \XSIMmixedcase{\GetExerciseName}\nobreakspace
        \GetExerciseProperty{counter}%
        \IfInsideSolutionF
          {%
            \IfExercisePropertySetT{subtitle}
              { {\normalfont\itshape\GetExerciseProperty{subtitle}}}%
          }%
      }
    \GetExercisePropertyT{points}
      {%
        \marginpar
          {%
            \IfInsideSolutionF{\rule{1.2cm}{1pt}\slash}%
            \PropertyValue
            \GetExercisePropertyT{bonus-points}
              {\nobreakspace(+\PropertyValue)}%
            \nobreakspace\XSIMtranslate{point-abbr}%
          }%
      }%
  }
  {}
```

The usage is now as follows:

```
\xsimsetup{exercise/template = bonus}
% set the boolean property to true
\begin{exercise}[bonus]
  A bonus question.
\end{exercise}
```

★ Bonus Exercise 10

A bonus question.

B.6. …Create and Use xsim Style Files?

xsim offers you the possibility to create own *style files*. Let's say you want to have a style called `math-exam`. Then you need to save all necessary definitions in a file called:

`xsim.math-exam.code.tex`

The first command in the file should be `\xsimstyle{math-exam}`. This file can now be loaded into your document using `\loadxsimstyle{math-exam}`:

```
\documentclass[DIV=18,parskip=half]{scrartcl}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}

\usepackage[clear-aux]{xsim}
\loadxsimstyle{math-exam}

\title{Math Exam \#3}
\date{2017-03-28}
```

In this style file stuff like template and property definitions should happen. This is more or less a convenient way to

- keep the preamble "clean" and
- define re-usable styles without the need of copying the document preamble to another document.

A style file is like a package or class file, *i. e.*, `@` has category code 11 (letter).

The formal description of the commands:

`\xsimstyle*{`⟨*style name*⟩`}`
The first command in a xsim style file called `xsim.`⟨*style name*⟩`.code.tex` which defines the xsim style ⟨*style name*⟩. The starred version activates expl3 syntax.[10]

`\loadxsimstyle{`⟨*csv list of style names*⟩`}`
Load one or more styles into the document.

! At the moment this mechanism offers no advantages over creating a custom package or simply `\input`ing a file. Future versions might provide additional features.

10. Those uses who want this will know what it means. If you don't know what it means you will not need it.

B.7. …Print All Solutions Grouped by Section?

Here is an idea how to get a list of all solutions grouped by the section the corresponding exercises are appearing in.

```
% preamble:
% \usepackage{etoolbox}
% \newcounter{sections}

% document:
\setcounter{sections}{1}
\whileboolexpr
  { test {\ifnumless{\value{sections}}{\value{section}+1}} }
  {
    \printsolutions[section=\value{sections},headings-template=per-section]
    \stepcounter{sections}
  }
```

For this manual we then get the following list.[11]

Solutions to the Exercises of Section 4

Solution 1

A first example for a solution.

Solutions to the Exercises of Section 8

Solution 5

The solution of the exercise that has not been printed.

Answers to the Problems of Section 12

Answer 1 *My subtitle*

This is the answer to problem 1.

11. Taking care of the fact that we're in the appendix now which means we can use `\value{section}`. Therefore this manual does `\edef\lastsection{\arabic{section}}` right before `\appendix`

Solutions to the Exercises of Section 14

Solution 9

Try to fill in these blanks. All of them are created by using the `\blank` command.

C. The xsimverb package

XSIM comes bundled with another package called xsimverb. This package loads a very small subset of XSIM which allows to create environments which write their contents verbatim to external files. It provides the following commands (which of course are also available in XSIM, too):

`\XSIMfilewritestart*{`⟨*file name*⟩`}`
Start writing to the file named ⟨*file name*⟩. This should be the *last* command in the *begin* definition of an environment. If is is used in an environment with arguments where the *last* argument is optional you should check if the optional argument is given and use the starred version if the test is negative. This is demonstrated in an example below using xparse's `\NewDocumentEnvironment`. *If you want an environment with only an optional argument you should use xparse's commands to define it. Due to the way how `\newenvironment` scans for optional arguments you'll otherwise may end up with leading spaces gobbled from the first line in your environment.*

`\XSIMfilewritestop`
Stop writing to the file. This should be the *first* command in the *end* definition of an environment.

`\XSIMsetfilebegin{`⟨*code*⟩`}`
This command can be used to write something to the external file *before* the environment contents. Must be set before `\XSIMfilewritestart` in the *begin* definition.

`\XSIMsetfileend{`⟨*code*⟩`}`
This command can be used to write something to the external file *after* the environment contents. Must be set before `\XSIMfilewritestart` in the *begin* definition.

`\XSIMgobblechars{`⟨*integer*⟩`}`
Determines how many characters are cut off of the beginning of each line of the environment body before it is written to the file. The default value is 0.

An example of how to use those commands:

```
\documentclass{article}
\usepackage{xsimverb,listings}

\makeatletter
\NewDocumentEnvironment{example}{o}
  {%
```

```
    \XSIMsetfilebegin{\@percentchar\space file `\jobname.tmp'}%
    \XSIMsetfileend{\@percentchar\space bye bye}%
    \IfNoValueTF{#1}
      {\XSIMfilewritestart*{\jobname.tmp}}
      {\XSIMfilewritestart{\jobname.tmp}}%
  }
  {%
    \XSIMfilewritestop
    \lstinputlisting[language={[LaTeX]TeX}]{\jobname.tmp}%
    \input{\jobname.tmp}
  }
\makeatother

\begin{document}

\begin{example}
bla bla \LaTeX
\end{example}

\end{document}
```

The `tmp` file produced by the above example will contain the following three lines (if the file itself was called `test.tex`):

```
% file `test.tmp'
bla bla \LaTeX
% bye bye
```

D. Example Documents Coming With This Package

The repository of this package [12] includes a number of example documents demonstrating how different aspects of this package work or how different kinds of problems can be solved or how different kinds of layouts can be achieved. They include:

- `boxed-headings.tex` – the headings of exercises put in a tcolorbox;
- `code-and-output.tex` – an example for the xsimverb package, see section C on the preceding page;
- `collections.tex` – demonstrating collections, see section 9 on page 19;
- `crossref.tex` – hyperlinks from exercises to solutions and back;
- `description-list.tex` – exercises as items in a `description` list;

12. GitHub: `https://github.com/cgnieder/xsim/`, CTAN: `http://www.ctan.org/pkg/xsim/`

- `different-point-types.tex` – defining custom point types and using them, see section 7.3 on page 12;
- `difficulties.tex` – defining custom tags and using them, see section 7.4 on page 15;
- `floating.tex` – exercises as floats;
- `listings.tex` – using code listings inside of exercises and solutions;
- `multiplechoice.tex` – defining a multiple choice exercise type with custom properties;
- `various.tex` – the original test file of the package author, demonstrating various aspects of the package.

E. All Exercise Examples

! You will notice that some exercises from section 12.4 on page 31 look differently in this section. That is because all exercises of a type use the template that's *currently active*. If you want exercises with a different look you should use different exercises types.

The following list is created with this code:

```
1 \xsimsetup{exercise/template = bonus}
2 \printcollection[headings]{all exercises}
```

Exercises

Exercise 1

A first example for an exercise.

Exercise 2 *This is a subtitle*

An exercise where some properties have been set.

_____/
4 (+1) p.

Exercise 3

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

`\GetExerciseProperty{id}`: 3

`\GetExerciseAliasProperty{ID}`: 3
`\GetExerciseProperty{ID}`: 3

Exercise 4

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

`\GetExerciseProperty{id}`: 4
`\GetExerciseAliasProperty{ID}`: 4
`\GetExerciseProperty{ID}`: foo-bar

Exercise 5

This exercise will not be printed but the exercise counter will be incremented nonetheless. Its solution will be printed in the list of solutions.

Exercise 6

This exercise is added to the collection ‘foo’.

Exercise 7. *exsheets’ runin*

_____/2.5 p.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Exercise 8. *exsheets’ margin*

_____/2.5 p.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Exercise 9

Try to fill in _____ blanks. All of them _____ by using the `\blank` _____.

★ Bonus Exercise 10

A bonus question.

Problems

Problem 1 *My subtitle* (5points)

This is a problem using a subtitle and points.

F. All Solution Examples

Solutions to the Exercises

Solution 1

A first example for a solution.

Solution 5

The solution of the exercise that has not been printed.

Solution 9

Try to fill in these blanks. All of them are created by using the `\blank` command.

Answers to the Problems

Answer 1 *My subtitle*

This is the answer to problem 1.

G. References

[Bra16] Johannes Braams, current maintainer: Javier Bezos.
babel. version 3.9q, Feb. 24, 2016 (or newer).
URL: http://mirror.ctan.org/macros/latex/required/babel/.

[Cha15] François Charette, current maintainer: Arthur Reutenauer.
polyglossia. version 1.42.0, Aug. 6, 2015 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/polyglossia/.

[Fea05] Simon Fear. booktabs. version 1.61803, Apr. 14, 2005 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/booktabs/.

[Flo] Bruno Le Floch. *Cunning (La)TeX tricks*.
URL: http://tex.stackexchange.com/a/19769/ (visited on 03/02/2017).

[L3Pa] The LaTeX3 Project Team. l3kernel. version SVN 6377, Jan. 19, 2016 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/l3kernel/.

[L3Pb] The LaTeX3 Project Team.
l3packages. version SVN 6377, Jan. 19, 2016 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/l3packages/.

[Leh15] Philipp Lehman, current maintainer: Joseph Wright.
etoolbox. version 2.2a, Aug. 2, 2015 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/etoolbox/.

[MC08] Frank Mittelbach and David Carlisle.
array. version 2.4c, Sept. 9, 2008 (or newer).
URL: http://mirror.ctan.org/macros/latex/required/tools/.

[Nie15] Clemens Niederberger. translations. version 1.2e, Nov. 7, 2015 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/translations/.

[Nie17] Clemens Niederberger. exsheets. version 0.21i, Feb. 8, 2017 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/exsheets/.

[var] various. *Questions tagged 'exsheets'*.
URL: http://tex.stackexchange.com/questions/tagged/exsheets (visited on 03/06/2017).

H. Index