Creating and Automating Exams with LaTeX & Friends

Uwe Ziegenhagen

Abstract

Although LaTeX is used a lot in academia and education only few teachers use it to prepare exams for their students. In this article we show how the `exam` package can be used to create various exercise types and how exercises can even be created randomly using Python.

1 Introducing the `exam` class

The `exam` package [1] is maintained by Philip Hirschhorn, the current version 2.6 is from November 2017. It supports various question types that are described in the well-written manual accompanying the package.

A very basic example for a `exam`-based exam sheet can be found in Listing 1. It uses `exam` as documentclass. Inside the document a `questions` environment is used, with item-like `\question` commands that take the number of achievable points for this exercise as optional parameter.

```
\documentclass[12pt]{exam}

\begin{document}\Large

\begin{questions}
\question[10] Who was Albert Einstein?
\question[10] Compute \(e = m \cdot c^2 \)!
\end{questions}

\end{document}
```

Listing 1: A basic example

For exams in languages other than English the exam-specific terms, see Listing 2 for an example translating the terms into German.

```
\pointpoints{Punkt}{Punkte}
\bonuspointpoints{Bonuspunkt}{Bonuspunkte}
\renewcommand{\solutiontitle}{\noindent\textbf{Lö
    sung:}\enspace}
\chqword{Frage}
\chpgword{Seite}
\chpword{Punkte}
\chbpword{Bonus Punkte}
\chsword{Erreicht}
\chtword{Gesamt}
\hpword{Punkte:}
\hsword{Ergebnis:}
\hqword{Aufgabe:}
\htword{Summe:}
```

Listing 2: Localization, here for German

The package also provides function for the layout of header and footer, separately for the first page and all subsequent pages. The corresponding commands, see Listing 3 and the result in Figures 2 and 3, each have three parameters, for the left, the center, and the right part of the corresponding header/footer.

1. (10 points) Who was Albert Einstein?
2. (10 points) Compute $e = m \cdot c^2$!

Figure 1: Output of Listing 1

```
\pagestyle{headandfoot}
\firstpageheadrule
\runningheadrule
\firstpageheader{<left>}{<center>}{John Doe \\
    Statistics 101 - 2019}
\runningheader{<l>}{<c>}{Statistics 101 - 2019}
\firstpagefooter{\today}{ACME University}{\thepage
    \,/\,\numpages}
\runningfooter{\today}{ACME University}{\thepage
    \,/\,\numpages}

\begin{document}\Large
\begin{questions}
\question[10] Who was Albert Einstein?
\question[10] Compute \(e = m \cdot c^2 \)!
\end{questions}
\end{document}
```

Listing 3: Setting header & footer

<left> John Doe Statistics 101 - 2019

1. (10 points) Who was Albert Einstein?
2. (10 points) Compute $e = m \cdot c^2$!

Figure 2: Resulting output (top) for Listing 3

August 19, 2019 ACME University 1 / 1

Figure 3: Resulting output (bottom) for Listing 3

Questions can be further divided, the `exam` packages provides the following environments:

- parts
- subparts
- subsubparts

Inside these environments individual subquestions are added with `\part`, `\subpart` or `\subsubpart`, see Listing 4 for an example using "parts" and "subparts".

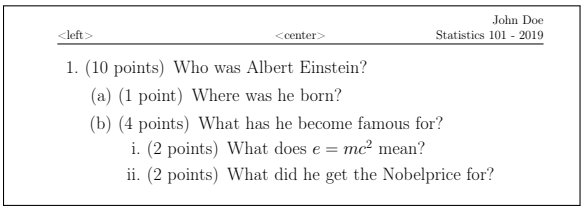

Figure 4: Resulting output for Listing 4

```
\question[10] Who was Albert Einstein?

\begin{parts}
  \part[1] Where was he born?
  \part[4] What has he become famous for?
  \begin{subparts}
     \subpart[2] What does \(e=mc^2\) mean?
     \subpart[2] What did he get the Nobelprice for?
   \end{subparts}
\end{parts}

\end{questions}
\end{document}
```

Listing 4: Subdivisions \part and \subpart

Besides the text-based questions we have seen so far, exam class offers several environments for multiple choice and fill-in questions:

- **choices** for vertical choices using letters
- **checkboxes** for vertical checkboxes
- **oneparcheckboxes** for horizontally aligned checkboxes

With \fillin[solutiontext] a horizontal line is printed, where the students are supposed to put their answer. As seen in the listings, the correct answer is defined by the \CorrectChoice command. To typeset a version of the exam that has the correct answers and solutions highlighted, "answers" needs to be added to the list of class options.

```
\question Who was not a Beatle?

\begin{choices}
\choice John
\choice Paul
\choice George
\CorrectChoice Benedict
\end{choices}
```

Listing 5: Example for choices

```
\question Who was not a Beatle?

\begin{checkboxes}
\choice John
\choice Paul
\choice George
\CorrectChoice Benedict
\end{checkboxes}
```

Listing 6: Example for checkboxes

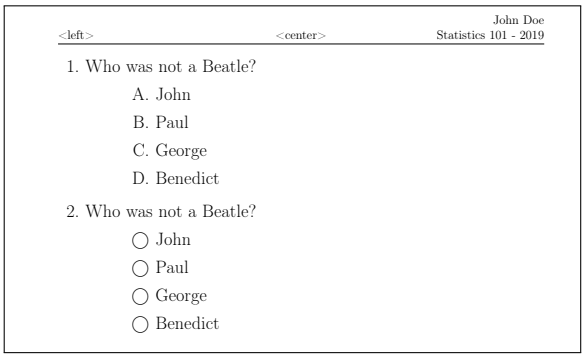

Figure 5: Output for Listing 5

```
\question Who was not Beatle?

\begin{oneparcheckboxes}
\choice John
\choice Paul
\choice George
\choice Ringo
\CorrectChoice Benedict
\end{oneparcheckboxes}

\question \fillin[James Bond][7em] has the \enquote{
     license to kill}.
```

Listing 7: oneparcheckboxes and fillin

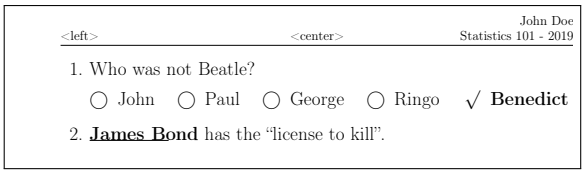

Figure 6: Oneparcheckboxes and fillin from Listing 7, with "answers" option set

To create space for answers the package supports not only the related TEX-commands, but it also provides several environments itself. Listing 8 shows a few commands to create empty vertical space, Listing 9 shows examples for the "enriched" solution space commands that provide lines, dotted lines or a grid.

```
% simple vertical space
\vspace*{<length>}

% vertical space to the end of the page
\vspace*{\stretch{1}}
\newpage

% empty framed box
\makeemptybox{<length>}

% empty framed box to the end of the page
\makeemptybox{\stretch{1}}
\newpage
```

Listing 8: TEX-Commands for simple answer space

```
\fillwithlines{<length>} % for lines
% Remark: \linefillheight for the inter-line spacing

\fillwithdottedlines{<length>} % for dotted lines
% Remark: distance in \dottedlinefillheight

\fillwithgrid{<length>} %
% \setlength{\gridsize}{5mm}
% \setlength{\gridlinewidth}{0.1pt}

\answerline[answer] % for short answers
```

Listing 9: Commands for enriched answer space

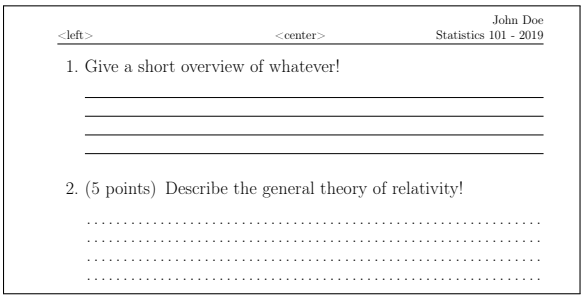

Figure 7: Resulting output for `\fillwithlines` and `\fillwithdottedlines` from Listing 9

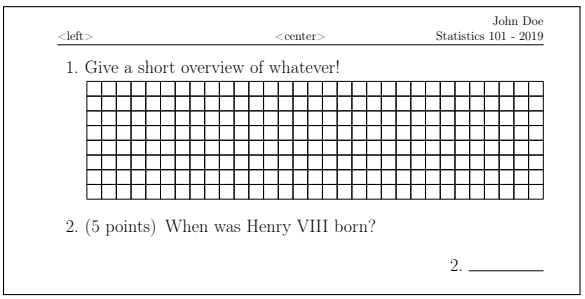

Figure 8: Resulting output for `\fillwithgrid` and `\answerline` from Listing 9

To insert solutions into the exam, one can use the `solution` – see Listing 10 and Figure 9 – or one of the following environments:

- solutionorbox
- solutionorlines
- solutionordottedlines
- solutionorgrid

For the `solutionorgrid` environment an example is provided in Listing 11 that – depending on whether the class option "answers" is set ot not – either presents a plot of a quadratic function or just a grid where the students are to draw the function themselves.

```
\begin{questions}
\question[1] How much does lead (Pb) weigh?

\begin{solution}
Pb weighs \SI{11,342}{\gram\per \centi\meter^3}
\end{solution}

\end{questions}
\end{document}
```

Listing 10: Using the solution environment

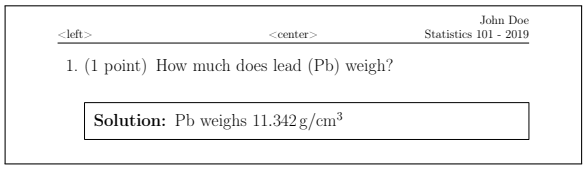

Figure 9: Resulting output for Listing 10

```
\question[5] Draw the function $3x^2+4x+5$!

\begin{solutionorgrid}[8cm]
\begin{tikzpicture}[baseline]
\begin{axis}[
axis y line=center,axis x line=middle,grid=both,
xmax=5,xmin=-5,ymin=0,ymax=10,
xlabel=$x$,ylabel=$y$,xtick={-5,...,5},
ytick={0,...,11},anchor=center]
\addplot[smooth,blue,thick,samples=100]{3*x^2+4*x+5}
    ;
\end{axis}
\end{tikzpicture}
\end{solutionorgrid}
```

Listing 11: Example for `solutionorgrid`

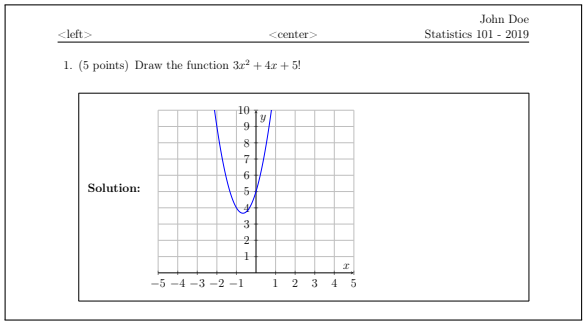

Figure 10: Resulting document for Listing 11 with class option "answers"

As mentioned earlier the different question environment take the number of points as optional parameters. To assist with the creation of the grading table `exam` features commands for vertical or horizontal grading tables that are either based on the page or exercise number. Listing 1 shows the corresponding commands, Figure 1 a resulting document using the `\gradetable[h][questions]` command.

```
\gradetable[v][questions] vertically per question
\gradetable[h][questions] horizontally per questions
\gradetable[v][pages] vertically per page
\gradetable[h][pages] horizontally per page
```

1. (2 points) What's the specific weight of air?
2. (2 points) What's the specific weight of air?

Question:	1	2	Total
Points:	2	2	4
Score:			

Figure 11: A grading table, two LaTeX runs are required

2 Automating `exam`

In this section we want to show, how exam questions can be created individually for each student to prevent cheating as each student has her or his individual exam. We also use QR codes that are printed behind each exercise, thus generating a teacher-friendly version by eleminating the need calculate all the individual results herself as a modern smartphone we be sufficient to see the result immediately.

We will first work on the LaTeX-part before we automate the whole creation process. First we first define a simple math question, see Listing 12.

```
\begin{questions}
\question[5] Calculate!

\begin{parts}
  \part[1] \(12345 + 67890 = \) \fillin[80235]
\end{parts}
```

Listing 12: A simple math exercise

We then use the `\qrcode` command from the `qrcode` package. This command takes just one parameter, the text that needs to be encoded, in our case this will be numeric result of the calculation. For the vertical and horizontal alignment we use the `\hfill` and `\vspace` commands, see Listing 13. This results in the document shown in Figure 12.

```
\begin{questions}
\question[5] Calculate!

\begin{parts}
  \part[1] \(12345 + 67890 = \) \fillin[80235] \hfill
       \qrcode{80235}\vspace{2em}
   \part[1] \(12345 + 67890 = \) \fillin[80235] \
        hfill\qrcode{80235}\vspace{2em}
```

Listing 13: Adding and aligning QR codes

Next we develop the required Python code, see Listing 14. We create a function that computes two random integers and their sum as well as the LaTeX-string to typeset the exercise with the QR code.

To automate our LaTeX document with the Python code we use the `pythontex` package by Geoffrey M Poore [2], that we presented in another talk at TUG 2019, and create the LaTeX document shown partially in Listing 15.

```
\pyc{from random import randrange}

\begin{pycode}
def gen_exercise():
    a = randrange(1000, 10000, 1)
    b = randrange(1000, 10000, 1)
    c = a + b
    a = str(a)
    b = str(b)
    c = str(c)
    return '\\(' + a + ' + ' + b + ' = \\) \\fillin['
          + c + '] \\hfill\\qrcode{' + c + '}\\vspace
         *{3em}'
\end{pycode}
```

Listing 14: The Python code

The code in the `\pyc` command parameter is only executed, it does not generate any printed text. The same holds for the `pycode` environment as well. Inside our `parts` environment we then use the `\py` to call the our function. It creates and return the expression this command requires. With the sequence `pdflatex`, `pythontex`, `pdflatex` we can then compile the final document similar to the one shown before in Figure 12.

```
\pyc{from random import randrange}

\begin{pycode}
def gen_exercise():
    a = randrange(1000, 10000, 1)
    b = randrange(1000, 10000, 1)
    c = a + b
    a = str(a)
    b = str(b)
    c = str(c)
    return '\\(' + a + ' + ' + b + ' = \\) \\fillin['
          + c + '] \\hfill\\qrcode{' + c + '}\\vspace
         *{3em}'
\end{pycode}

\begin{questions}
\question[5] Calculate!

\begin{parts}
\part[1] \py{gen_exercise()}
\part[1] \py{gen_exercise()}
\part[1] \py{gen_exercise()}
\part[1] \py{gen_exercise()}
\part[1] \py{gen_exercise()}
\end{parts}
```

Listing 15: Excerpt of the document including the `pythontex` code

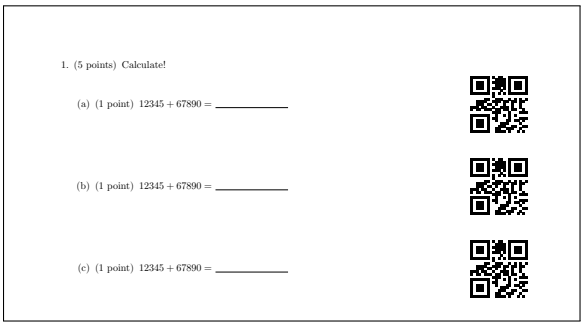

Figure 12: Resulting output (excerpt)

3 Summary

In this article we have presented the most important features of the `exam` class and shown how easy exams can be typeset with LaTeX. We have also shown how individual exercises can be created to allow more variability in the numerical values used in the exam.

Accompanying to this article is the more extensive presentation held at TUG 2019 for which the interested reader can find the slides at `www.uweziegenhagen.de`.

References

[1] P. Hirschhorn. *Using the exam document class*, 2017. `https://ctan.math.illinois.edu/macros/latex/contrib/exam/examdoc.pdf`

[2] G. M. Poore. Pythontex: reproducible documents with latex,python, and more. *Comput. Sci. Disc.* 8(1), 2015.

◇ Uwe Ziegenhagen
Escher Str. 221
50739 Cologne
Germany
`ziegenhagen@gmail.com`
`https://www.uweziegenhagen.de`