Beamer Code Listing — Syntax highlighter

By Admin   August 17, 2021

Your guide to formatting and highlighting source codes in Beamer

Contents :

1. Fragile option is what you need to insert codes in Beamer frames

> If you try to insert code inside a beamer environment like you would do in any LaTeX document, you will get an error. Whether you type the code using the verbatim, minted or listings environment, since they are all different kinds of verbatim text, you will always get the same error!

To make this environments work, you just have to pass the `fragile` option to the frame where the code will go, and everything will work as expected. We already used this option in lesson: Your Guide to Beamer blocks.

```
% Fragile option
\begin{frame}[fragile]

% Frame content

\end{frame}
```

Copy

If you, as me, are interested in why this works, I am going to dive into it in this section “More details about fragile option“

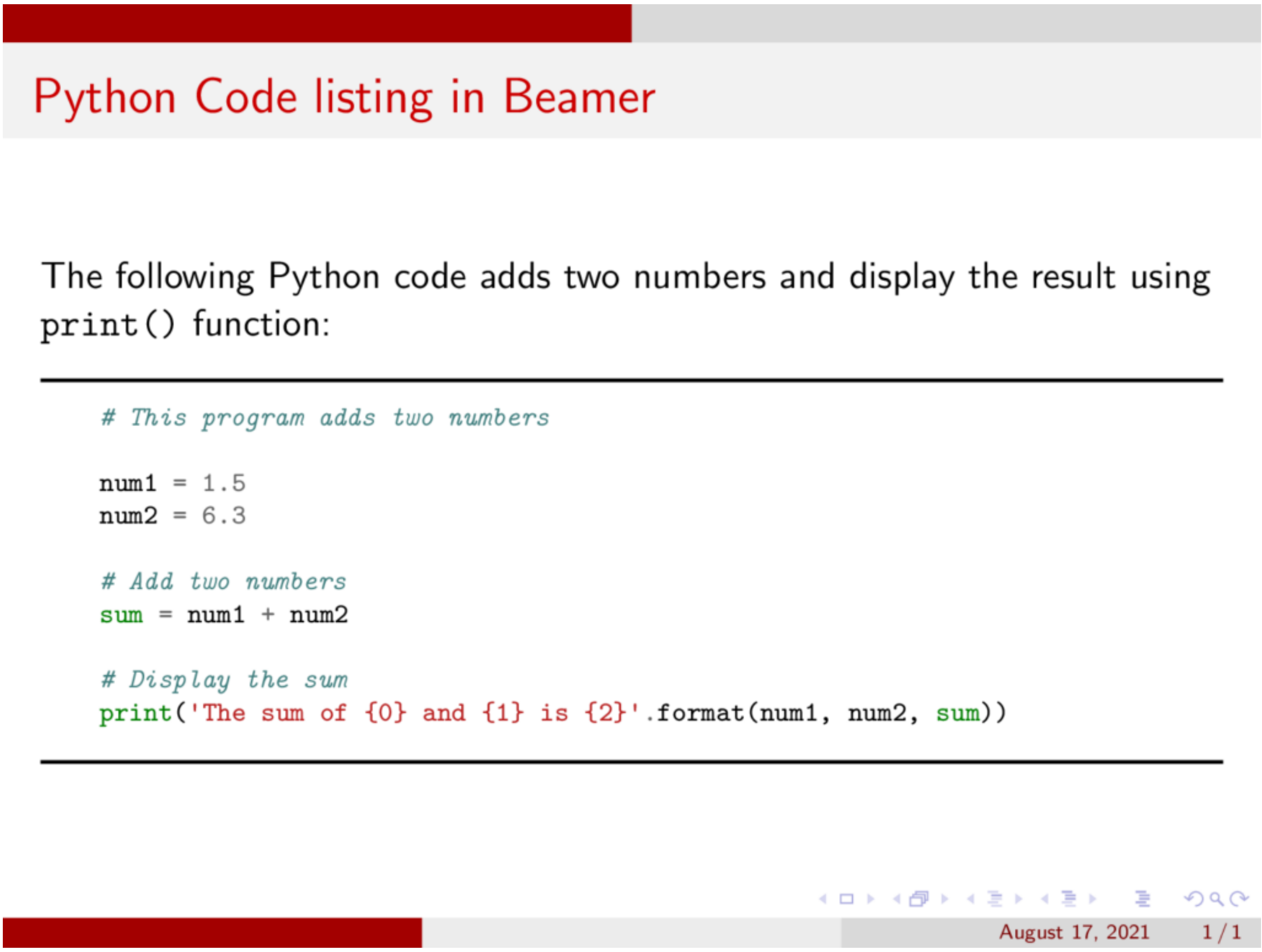
2. Code listing using minted package

To highlight the use of minted package for code syntax highlighting, we consider the following example:

Copy

```
% Code listing in Beamer
\documentclass{beamer}

% Theme choice
\usetheme{CambridgeUS}

% Code syntax highlighter
\usepackage{minted}

\begin{document}


\begin{frame}[fragile]
\frametitle{Python Code listing in Beamer}

The following Python code adds two numbers and display the result using \verb|print()| function:


\rule{\textwidth}{1pt}

\scriptsize
\begin{minted}{python}
    # This program adds two numbers

    num1 = 1.5
    num2 = 6.3

    # Add two numbers
    sum = num1 + num2

    # Display the sum
    print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
\end{minted}

\rule{\textwidth}{1pt}

\end{frame}

\end{document}
```

Compiling this code yields:



Python Code listing in Beamer

The following Python code adds two numbers and display the result using `print()` function:

```python
# This program adds two numbers

num1 = 1.5
num2 = 6.3

# Add two numbers
sum = num1 + num2

# Display the sum
print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))
```

August 17, 2021 1 / 1

Comments:

- For this illustrative example, we used CambridgeUS template. For more Beamer templates, check this lesson!
- We loaded minted package to highlight source codes in beamer.
- As we mentioned above, we added the option `[fragile]` to the frame environment in order to be able to insert, format and highlight source codes.

- The frame has the title “Python code listing in Beamer” which is added using the command `\frametitle{}`.
- Thanks to fragile option, we used the command `\verb|  |` to highlight the syntax `print()` without errors.
- We added `rule{\textwidth}{1pt}` before and after the code listing to draw a horizontal line with width equal to text width and height equal to 1pt.
- We used `minted` environment with the option `python` to specify the code language.

3. More details about fragile option

First of all, we have to understand what `fragile` means. This concept has to do with expansion and execution. TeX does these at the same time: first it reads a token, expands it so that only low-level tokens are left and then executes it. But this cycle isn’t always followed: for instance when moving text around this behavior changes.

In which situations will TEX move text?

For instance, when creating a table of contents, since all the chapters, sections and subsections are scattered around your document, but in the end they all end up in the table of contents. To do so, TeX, when reading and executing a `\section` command, will write the current section’s title and number to a `.toc` file. After all the commands have been parsed, TeX will fill the table of contents based on the data that was collected in the `.toc` file.

This can represent a problem because the meaning of code changes during TEX’s process, and only when the typesetting is done the actual meaning of some commands can be determined. So when writing data to a file, some expansions must not occur, because they are dependent on the current situation; examples of these are: commands with optional arguments, line breaks, footnotes and inline math.

To deal with this issues, TeX offers the `\protect` command, which can be used to protect fragile commands. In general, the commands that are not fragile (and thus need not protection) are called *robust*.

> But the real question was why did the frame environment need the fragile option? The beamer user manual tells us that using this option the contents of the frame are written to an external file and then read back.

But again one may ask why bother to do so. The truth is that beamer is a *very* complex package, to the point that it tunes the TeX internal character codes (which are, in turn, one of the most fundamental elements of TeX). When a frame contains fragile text, different internal mechanisms are used to typeset the frame to ensure that inside the frame the character codes can be reset.

The price of switching to another internal mechanism is that either you cannot use overlays (one of beamer's main features, and the reason why character codes are changed, since overlays are specified with the symbols `< >`, which have to change they character codes) or an external file needs to be written and read back (which is not always desirable, because of the extended compilation time, but is the last option available).

It is clear that the character codes are not easily reset when verbatim code appears inside the frame, since the verbatim environments change the character codes, so that you don't have to use {} as delimiters (and thus you can use them inside the command). For instance, I use the delimiters `\verb|  |` (but to write this, I had to use as delimiters + signs).

Next Lesson: 13 Beamer Font: Change its Size, Family and style

Search

About LaTeX-Beamer.com

LaTeX-Beamer.com is a personal website about creating stylish and modern presentations in LaTeX, through step-by-step lessons.