

# The `verbments` package: Pretty printing source code in L<sup>A</sup>T<sub>E</sub>X\*

Dejan Živković  
Singidunum University, Serbia  
dzivkovic@singidunum.ac.rs

Version 1.2  
August 20, 2011

## Abstract

The `verbments` package provides an environment for syntax highlighting of source code in L<sup>A</sup>T<sub>E</sub>X documents. The highlighted source code output is formatted via powerful Pygments library of the Python language.

<b>Contents</b>		<b>5</b>	<b><code>fancyvrb</code> options</b>	<b>5</b>
<b>1 Introduction</b>	<b>1</b>	<b>6</b>	<b>Options</b>	<b>7</b>
<b>2 Installation</b>	<b>2</b>	<b>7</b>	<b>Todo list</b>	<b>8</b>
<b>3 Basic usage</b>	<b>2</b>	<b>8</b>	<b>Version history</b>	<b>8</b>
<b>4 Captions</b>	<b>4</b>			

## 1 Introduction

The `verbments` package overcomes some deficiencies of two other popular packages for pretty printing source code in L<sup>A</sup>T<sub>E</sub>X. Namely, the `listings` package is relatively old and lacks the utf-8 support. On the other hand, the `minted` package cannot split the highlighted source code over pages, nor it provides an option for individual highlighting styles.

The `verbments` package uses `minted` idea to delegate the task of actual syntax markup of the source code to an external software — the Pygments library. Marked code is then typeset using the `fancyvrb` and `framed` packages. (The

---

\*This work has been supported by the Serbian Ministry of Education and Science (project III44006).

package name *verbments* tries to convey this as being a compound word resembling two words *verbatim* and *pygments*.)

Pygments is written in Python and provides far superior syntax highlighting compared to conventional packages. For example, `listings` basically only highlights strings, comments, and keywords. Pygments, on the other hand, can be completely customized to markup any token of the source language. Furthermore, the number of supported languages is clearly in favor of Pygments with over 150 different programming and other languages supported. More information on Python and Pygments can be found at <http://python.org> and <http://pygments.org>.

## 2 Installation

Installation of the `verbments` package itself is simple—the `verbments.sty` file only needs to be on the path where L<sup>A</sup>T<sub>E</sub>X can find it. However, `verbments` package additionally requires that the Python language and its Pygments library are installed on the computer.

Python and Pygments are free software and can be downloaded from their web sites. Bear in mind that Windows support is sketchy at the moment, but instructions to properly configure the software may be found elsewhere on the Internet. (For example, good starting point is the documentation of the `minted` package.)

## 3 Basic usage

- `-shell-escape` First of all, don't forget to call the L<sup>A</sup>T<sub>E</sub>X compiler (`latex`, `pdflatex`, or `xelatex`) by passing it the `-shell-escape` option.
- `pyglist` Using `verbments` in a L<sup>A</sup>T<sub>E</sub>X document is straightforward—you simply enclose a source code in the `pyglist` environment:

```
begin{pyglist}[\langle options \rangle]
  \langle source code \rangle
end{pyglist}
```

The `pyglist` environment accepts a number of options in the `key=value` notation. For example, to highlight a Java source code, you may use the following `pyglist` environment:

```
\begin{pyglist}[language=java]
// Hello Java program
import java.util.*;
public class Hello {
    public static void main(String[] args) {
```

```

        Scanner kb = new Scanner(System.in);
        System.out.print("What is your name? ");
        String name = kb.nextLine();
        System.out.println("Hello " + name + "!");
    }
}
\end{pyglist}

```

This environment uses the default Pygments style and the Java example code is typeset as follows:

```

// Hello Java program
import java.util.*;
public class Hello {
    public static void main(String[] args) {
        Scanner kb = new Scanner(System.in);
        System.out.print("What is your name? ");
        String name = kb.nextLine();
        System.out.println("Hello " + name + "!");
    }
}

```

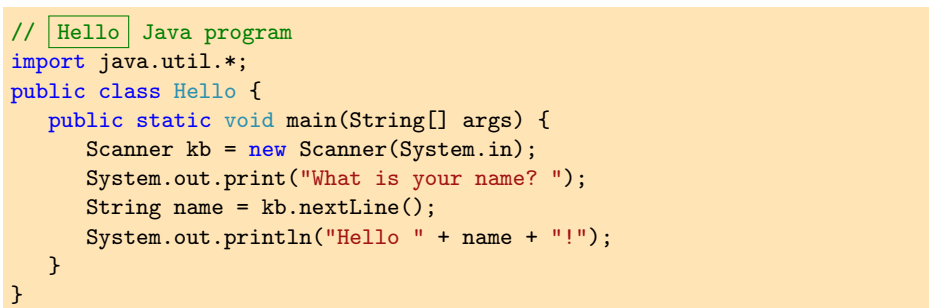
If the `texcl=true`, `style=vs` and `bgcolor=Moccasin` options are added to the option list of the previous `pyglist` environment for a similar Java code, i.e.,

```

\begin{pyglist}[language=java, texcl=true, style=vs, bgcolor=Moccasin]
// \fbox{Hello} Java program
.
.
.
\end{pyglist}

```

then the source code is typeset with the Visual Studio style and Moccasin background color:



```

// Hello Java program
import java.util.*;
public class Hello {
    public static void main(String[] args) {
        Scanner kb = new Scanner(System.in);
        System.out.print("What is your name? ");
        String name = kb.nextLine();
        System.out.println("Hello " + name + "!");
    }
}

```

`\plset` Since the option list of the `pyglist` environment may become lengthy, options

can be globally specified using the `\plset` command. For example:

```
\plset{language=java, texcl=true, style=vs, bgcolor=Moccasin}
```

With the options globally set in this way, now to get the previous highlighted example code it is enough to specify:

```
\begin{pyglist}
// \fbox{Hello} Java program
.
.
.
\end{pyglist}
```

## 4 Captions

Source code listings can contain captions. For example,

```
\begin{pyglist}[language=java,caption={First Java program}]
// Hello Java program
.
.
.
\end{pyglist}
```

produces the following result:

Listing 1: First Java program

```
// Hello Java program
.
.
.
```

Caption label used as a prefix to the caption text of a listing is `Listing` by default. This can be changed using the `listingname` option. For example,

```
\begin{pyglist}[language=java,caption={First Java program},%
                listingname=\textbf{Program}]
// Hello Java program
.
.
.
\end{pyglist}
```

produces the following result:

### Program 2: First Java program

```
// Hello Java program
.
.
.
```

It is also possible to specify the background color of the caption text and label of a listing, as well as their individual font (and possibly other) characteristics. For example,

```
\begin{pyglist}[language=java,caption={First Java program},%
    listingnamefont=\sffamily\bfseries\color{yellow},%
    captionfont=\sffamily\color{white},captionbgcolor=gray]
// Hello Java program
.
.
.
\end{pyglist}
```

produces the following result:

**Listing 3:** First Java program

```
// Hello Java program
.
.
.
```

`\listofpyglistings` If source code listings in a document are decorated with captions, their list can be produced with the `\listofpyglistings` command. This is akin to the list of figures, list of tables and other table-of-contents counterparts.

`\listofpyglistingsname` Heading of the list of listings is `Listings` by default. This can be changed using the `\listofpyglistingsname` command. For example:

```
\listofpyglistingsname{List of Programs}
```

## 5 fancyvrb options

The `pyglist` environment is actually typeset using the `Verbatim` environment of the `fancyvrb` package. That's why all `fancyvrb` options are also in effect in the `pyglist` environment. For a more detailed `fancyvrb` options description, please refer to the documentation of the `fancyvrb` package. For example,

```

\fvset{frame=single}
\begin{pyglist}[language=c,style=tango,numbers=left,numbersep=5pt]
/* Hello World program */

#include<stdio.h>

main()
{
    printf("Hello World");
}
\end{pyglist}
\fvset{frame=none}

```

produces the following result:

```

1  /* Hello World program */
2
3  #include<stdio.h>
4
5  main()
6  {
7      printf("Hello World");
8  }

```

The fancyvrb options set with the `\fvset` command are global. They can be also set locally using the `fvset` option of the `pyglist` environment. For example,

```

\begin{pyglist}[language=java,caption={First Java program},%
    listingnamefont=\sffamily\bfseries\color{white},%
    captionfont=\sffamily\color{white},captionbgcolor=gray,%
    fvset={frame=bottomline,framerule=4pt,rulecolor=\color{gray}}]
// Hello Java program
.
.
.
\end{pyglist}

```

produces the following result:

**Listing 4: First Java program**

```

// Hello Java program
.
.
.

```

## 6 Options

The following is a full list of available options for the `pyglist` environment.

- `abovecaptionskip` (dimension) White space length above caption of the listing (default: `\bigskipamount`).
- `belowcaptionskip` (dimension) White space length below caption of the listing (default: `0pt`).
- `bgcolor` (color name) Background color of the listing (default: `white`).
- `caption` (string) Caption text of the listing (default: `none`).
- `captionbgcolor` (color name) Background color of the caption text and label of the listing (default: `none`).
- `captionfont` (font spec) Font (and possibly other) specifications prepended to the caption text of the listing (default: `{}`).
- `encoding` (string) Encoding of intermediate input and output files with syntax markup used by Pygments (default: `latin1`). For more information, please refer to the Pygments documentation.
- `fontsize` (font size) Font size to use for the listing (default: `auto`—the same as the current font).
- `fvset` (options) The `fancyvrb` package options to use locally for the listing (default: `none`). For more information, please refer to the `fancyvrb` documentation.
- `gobble` (integer) Removes the first  $n$  characters from each input line of the source code (default: `0`).
- `language` (string) Language of the source code whose syntax is to be highlighted by Pygments (default: `text`). For more information, please refer to the Pygments documentation.
- `label` (string) Label that makes the listing referable via `\ref{label}` (default: `none`).
- `listingname` (string) Caption label to use as a prefix to the caption text of the listing (default: `Listing`).
- `listingnamefont` (font spec) Font (and possibly other) specifications prepended to the caption label of the listing (default: `{}`).
- `mathescape` (boolean) Enables  $\LaTeX$  math mode inside comments (default: `false`).
- `numbers` (none|left|right) Numbering of the listing lines (default: `none`).
- `numbersep` (dimension) Gap between numbers and start of the listing lines (default: `12pt`).

`showspaces` (boolean) Enables visible spaces. The space character is printed as the symbol  $\sqcup$  (default: *none*).

`style` (string) Style for Pygments to use for the source code highlighting (default: `default`). For more information, please refer to the Pygments documentation.

`texcl` (boolean) Enables L<sup>A</sup>T<sub>E</sub>X code inside comments (default: `false`).

## 7 Todo list

- Implement package distribution file. (Problems: (1) the % sign in the Pygments style file is gobbled up by `\DocStrip` in examples; (2) spurious blanks appear in the highlighted example code.)
- Implement escape sequences. (This is probably more suitable task for the Pygments lexers and formatters?)

## 8 Version history

### v1.2 2011/08/20

Added `captionbgcolor`, `captionfont`, `listingnamefont`, and `fvset` options.

Did minor code cleaning.

Revised documentation to include the new options.

### v1.1 2011/07/07

Added the `\listofpyglistingsname` user command.

Changed handling of the list of listings.

Revised documentation and examples.

### v1.0 2011/06/07

Initial public release.