

Self-contained Literate Programming

Volker Grabsch

January 2, 2009

Contents

1	Introduction	2
2	Hello World in Python	2
3	Building this document	3
3.1	LaTeX source of this document	4
3.2	Extracted Python source of this document	6

1 Introduction

This document demonstrates the technique of self-contained literate programming. It is a Python program, its LaTeX documentation and build instruction all at once. It does not need any external tools besides the obvious (Python and LaTeX). Note that the listing below which shows the program's output is really generated automatically, by running the program.

2 Hello World in Python

This is a simple main function:

```
def main():  
    print "Hello , World!"
```

And this is the usual way to call the main function:

```
if __name__ == "__main__":  
    main()
```

Output:

```
Hello , World!
```

3 Building this document

The following command builds the whole project:

```
sh hello-world.tex
```

That works because the LaTeX source is also a valid shell script. It performs the following steps:

1. Show each command before executing, and cancel in case of an unexpected error:

```
set -xe
```

2. Extract all files of this document:

```
sed '/^\n/,/hello.py$/ s,^,#,' >hello.py <hello-world.tex
```

3. Run the Python script and save its output:

```
python hello.py >hello.out
```

4. Compile the LaTeX source to a PDF document:

```
pdflatex hello-world.tex  
pdflatex hello-world.tex
```

5. Remove all temporary files:

```
rm -f hello-world.{aux,log,out,toc}  
rm -f hello.{py,out}
```

3.1 LaTeX source of this document

```
\documentclass[a4paper,12pt]{article} % 2>/dev/null || <<'% build '

% The following command builds the whole project:
%
% sh hello-world.tex

\usepackage[left=2cm,right=2cm,top=2cm,bottom=2cm]{geometry}
\parindent 0cm
\usepackage[usenames,dvipsnames]{color}
\usepackage{listings}
\lstset{showstringspaces=false}
\lstset{frame=shadowbox}
\lstset{rulesepcolor=\color{Gray}}
\usepackage{hyperref}

\begin{document}

\title{Self-contained Literate Programming}
\author{Volker Grabsch}
\maketitle

\tableofcontents

\newpage
\section{Introduction}
This document demonstrates the technique of
self-contained literate programming.
It is a Python program,
its LaTeX documentation
and build instruction all at once.
It does not need any external tools
besides the obvious (Python and LaTeX).
Note that the listing below
which shows the program's output
is really generated automatically,
by running the program.

\section{Hello World in Python}

This is a simple main function:
\begin{lstlisting}[language=Python]% hello.py
def main():
    print "Hello, World!"
\end{lstlisting}
And this is the usual way to call the main function:
\begin{lstlisting}[language=Python]% hello.py
if __name__ == "__main__":
    main()
\end{lstlisting}
Output:
\lstinputlisting{hello.out}

\newpage
\section{Building this document}

The following command builds the whole project:
\begin{lstlisting}[language=sh]%
sh hello-world.tex
\end{lstlisting}
That works because the LaTeX source
is also a valid shell script.
```

```

It performs the following steps:
\begin{enumerate}

\item
Show each command before executing,
and cancel in case of an unexpected error:
\begin{lstlisting}[language=sh]%
% build

set -xe

\end{lstlisting} % 2>/dev/null || <<'% build '

\item
Extract all files of this document:
\begin{lstlisting}[language=,basicstyle=\scriptsize]%
% build

sed '/^\s*\./hello.py$/ s,^,#, ' >hello.py <hello-world.tex

\end{lstlisting} % 2>/dev/null || <<'% build '

\item
Run the Python script and save its output:
\begin{lstlisting}[language=sh]%
% build

python hello.py >hello.out

\end{lstlisting} % 2>/dev/null || <<'% build '

\item
Compile the LaTeX source to a PDF document:
\begin{lstlisting}[language=sh]%
% build

pdflatex hello-world.tex
pdflatex hello-world.tex

\end{lstlisting} % 2>/dev/null || <<'% build '

\item
Remove all temporary files:
\begin{lstlisting}[language=sh]%
% build

rm -f hello-world.{aux,log,out,toc}
rm -f hello.{py,out}

\end{lstlisting} % 2>/dev/null || <<'% build '

\end{enumerate}

\newpage
\subsection{LaTeX source of this document}
\lstinputlisting [language={LaTeX}TeX], basicstyle=\scriptsize]{hello-world.tex}

\newpage
\subsection{Extracted Python source of this document}
\lstinputlisting [language=Python, basicstyle=\scriptsize]{hello.py}

\end{document}

```

3.2 Extracted Python source of this document

```
##\documentclass[a4paper,12pt]{article} % 2>/dev/null || <<'% build'
#
#
##% The following command builds the whole project:
##%
##% sh hello-world.tex
#
#
\usepackage[left=2cm,right=2cm,top=2cm,bottom=2cm]{geometry}
# \parindent 0cm
\usepackage[usenames,dvipsnames]{color}
\usepackage{listings}
# \lstset{showstringspaces=false}
# \lstset{frame=shadowbox}
# \lstset{rulesepcolor=\color{Gray}}
\usepackage{hyperref}
#
#
\begin{document}
#
#
\title{Self-contained Literate Programming}
\author{Volker Grabsch}
\maketitle
#
\tableofcontents
#
#
\newpage
\section{Introduction}
#This document demonstrates the technique of
#self-contained literate programming.
#It is a Python program,
#its LaTeX documentation
#and build instruction all at once.
#It does not need any external tools
#besides the obvious (Python and LaTeX).
#Note that the listing below
#which shows the program's output
#is really generated automatically,
#by running the program.
#
#
\section{Hello World in Python}
#
#This is a simple main function:
\begin{lstlisting}[language=Python]% hello.py

def main():
    print "Hello, World!"

\end{lstlisting}
#And this is the usual way to call the main function:
\begin{lstlisting}[language=Python]% hello.py

if __name__ == "__main__":
    main()

\end{lstlisting}
#Output:
\lstinputlisting{hello.out}
#
#
\newpage
\section{Building this document}
#
#The following command builds the whole project:
\begin{lstlisting}[language=sh]%
#
#sh hello-world.tex
#
#\end{lstlisting}
#That works because the LaTeX source
#is also a valid shell script.
```

```

#It performs the following steps:
#\begin{enumerate}
#
#\item
#Show each command before executing,
#and cancel in case of an unexpected error:
#\begin{lstlisting}[language=sh]%
##% build
#
#set -xe
#
#\end{lstlisting} % 2>/dev/null || <<'% build'
#
#\item
#Extract all files of this document:
#\begin{lstlisting}[language=,basicstyle=\scriptsize]%
##% build
#
#sed '/^\//,/hello.py$/ s,^,#,' >hello.py <hello-world.tex
#
#\end{lstlisting} % 2>/dev/null || <<'% build'
#
#\item
#Run the Python script and save its output:
#\begin{lstlisting}[language=sh]%
##% build
#
#python hello.py >hello.out
#
#\end{lstlisting} % 2>/dev/null || <<'% build'
#
#\item
#Compile the LaTeX source to a PDF document:
#\begin{lstlisting}[language=sh]%
##% build
#
#pdflatex hello-world.tex
#pdflatex hello-world.tex
#
#\end{lstlisting} % 2>/dev/null || <<'% build'
#
#\item
#Remove all temporary files:
#\begin{lstlisting}[language=sh]%
##% build
#
#rm -f hello-world.{aux,log,out,toc}
#rm -f hello.{py,out}
#
#\end{lstlisting} % 2>/dev/null || <<'% build'
#
#\end{enumerate}
#
#\newpage
#\subsection{LaTeX source of this document}
#\lstinputlisting[language={[LaTeX]TeX},basicstyle=\scriptsize]{hello-world.tex}
#
#\newpage
#\subsection{Extracted Python source of this document}
#\lstinputlisting[language=Python,basicstyle=\scriptsize]{hello.py}
#
#\end{document}

```