Note on generating Latex code from Lua and Python

Nasser M. Abbasi

July 3, 2015 page compiled on July 3, 2015 at 4:22pm

Small example is given that generate Latex code on the fly. One uses Lua in which the Lua code is embedded inside the main Latex file itself using `\begin{luacode}....\end{luacode}` and the other example uses Python to generate the complete Latex document to a file, which is then compiled using either `pdflatex` or `lualatex`.

When using the Lua method, the file has to be compiled using `lualatex`.

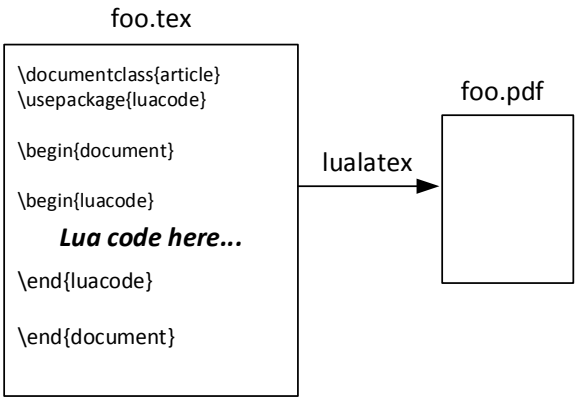
Python is one of the few dynamics languages that supports multi-line raw strings using the `r""" .... """` construct. This feature allows any arbitrary content to be used as is inside this construct. It also allows one to concatenate Python variables with the string to build the final Latex output. Without this feature, using Python to generate Latex code would not be practical to do.

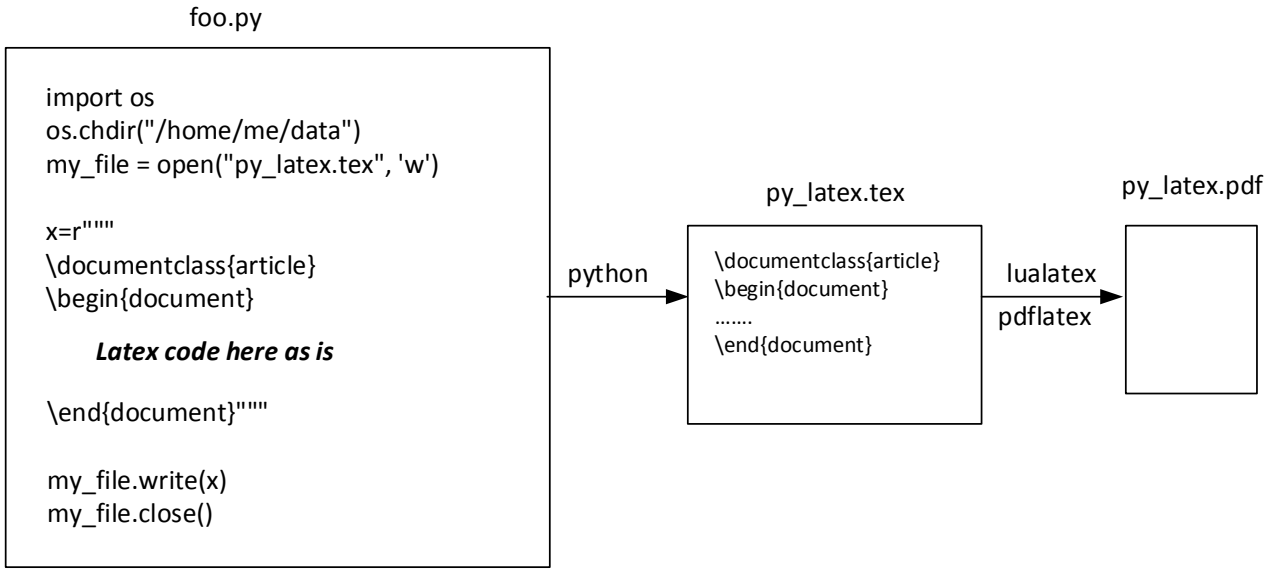
The example used (slightly modified) is an arbitrary one which builds small multiplication table taken from `why-does-luaexec-work-but-the-luacode-environment-doesnt` answer by Herbert. There was no specific reason to pick this example, I just wanted to try to compare using Lua vs. Python to do this, and I was browsing and just saw this example there and it seemed like a good one to try to compare Lua with Python on. More examples will be added.

When the generated Latex code is compiled, it will generate this output:

$1 \times 1 = 1$	$1 \times 2 = 2$	$1 \times 3 = 3$	$1 \times 4 = 4$	$1 \times 5 = 5$	$1 \times 6 = 6$
$2 \times 1 = 2$	$2 \times 2 = 4$	$2 \times 3 = 6$	$2 \times 4 = 8$	$2 \times 5 = 10$	$2 \times 6 = 12$
$3 \times 1 = 3$	$3 \times 2 = 6$	$3 \times 3 = 9$	$3 \times 4 = 12$	$3 \times 5 = 15$	$3 \times 6 = 18$
$4 \times 1 = 4$	$4 \times 2 = 8$	$4 \times 3 = 12$	$4 \times 4 = 16$	$4 \times 5 = 20$	$4 \times 6 = 24$
$5 \times 1 = 5$	$5 \times 2 = 10$	$5 \times 3 = 15$	$5 \times 4 = 20$	$5 \times 5 = 25$	$5 \times 6 = 30$
$6 \times 1 = 6$	$6 \times 2 = 12$	$6 \times 3 = 18$	$6 \times 4 = 24$	$6 \times 5 = 30$	$6 \times 6 = 36$

The two methods are best contrasted and compared using the following diagram. The python script is first compiled using Python, then the generated Latex file is compiled using either lualatex or pdflatex.





Here is the Lua and the Python code, side by side, for comparison

Lua	Python

\documentclass{article}
\usepackage{luacode}
 	
\begin{document}
 	
\begin{luacode}
tex.print("\\begin{tabular}{|l|l|l|l|l|l|l|l|l|}\\hline")
num=6
for i=1,num do
  for j=1,num do
  ixj='$'..i..'\\times'..j..'='..i*j..'$';
  tex.print(ixj)
  if(j<num) then
  tex.sprint('&')
  else
  tex.sprint('\\\\')
  end
  end
end
tex.print("\\hline\\end{tabular}")
\end{luacode}
 	
\end{document}
	import os
	os.chdir("/home/me/data")
	my_file = open("py_latex.tex", 'w')
	
	num = 6
	
	x=r"""
	\documentclass{article}
	\begin{document}
	
	\begin{tabular}{|l|l|l|l|l|l|l|l|l|}\hline
	"""
	for i in range(1,num+1):
	  for j in range(1,num+1):
	  x=x+'$'+str(i)+' \\times '+str(j)+' = '+str(i*j)+'$'
	  if j<num :
	  x=x+'&'
	  else:
	  x=x+'\\\\ \n'
	
	x=x+r"""\hline\end{tabular}
	\end{document}"""
	
	my_file.write(x)
	my_file.close()

The Latex file generated by the Python code above is the following, which is then compiled using `pdflatex`

```
\documentclass{article}
\begin{document}

\begin{tabular}{|l|l|l|l|l|l|l|l|l|}\hline
$1 \times 1 = 1$&$1 \times 2 = 2$&$1 \times 3 = 3$&$1 \times 4 = 4$&$1 \times 5 = 5$&$1 \times 6 = 6$\\
$2 \times 1 = 2$&$2 \times 2 = 4$&$2 \times 3 = 6$&$2 \times 4 = 8$&$2 \times 5 = 10$&$2 \times 6 = 12$\\
$3 \times 1 = 3$&$3 \times 2 = 6$&$3 \times 3 = 9$&$3 \times 4 = 12$&$3 \times 5 = 15$&$3 \times 6 = 18$\\
$4 \times 1 = 4$&$4 \times 2 = 8$&$4 \times 3 = 12$&$4 \times 4 = 16$&$4 \times 5 = 20$&$4 \times 6 = 24$\\
$5 \times 1 = 5$&$5 \times 2 = 10$&$5 \times 3 = 15$&$5 \times 4 = 20$&$5 \times 5 = 25$&$5 \times 6 = 30$\\
$6 \times 1 = 6$&$6 \times 2 = 12$&$6 \times 3 = 18$&$6 \times 4 = 24$&$6 \times 5 = 30$&$6 \times 6 = 36$\\
\hline\end{tabular}
\end{document}
```

It is clear than both using Lua or Python to generate the table is much easier than using direct Tex coding (programming directly in Tex is very hard for most of us and can only be done by the real Latex programmers with many long years of Latex programming).

Using a higher level language such as Lua or Python is much simpler as the languages are easier to program in.

Currently I use Python to generate Latex files. For example, the following page was generated from one Python script.

There are advantages to both methods. And it is left to the reader to decide which method might work for them. This note is meant to only illustrate the methods and document them.