

# PythonTeX Gallery

Geoffrey M. Poore

November 15, 2014

## Abstract

PythonTeX allows you to run Python code from within L<sup>A</sup>T<sub>E</sub>X documents and automatically include the output. This document serves as an example of what is possible with PythonTeX.\*

## 1 General Python interaction

We can typeset code that is passed to Python, and bring back the results.

This can be simple. For example, `??` returns the following:

```
?? PythonTeX ??
```

Or we could access the printed content verbatim (it might contain special characters):

```
?? PythonTeX ??
```

Python interaction can also be more complex. `??` returns `?? PythonTeX ??`. In this case, the printed results include L<sup>A</sup>T<sub>E</sub>X code, which is correctly interpreted by L<sup>A</sup>T<sub>E</sub>X to ensure that there is not an extra space after the 16. Printed output is saved to a file and brought back in via `\input`, and the `\endinput` command stops input immediately, before L<sup>A</sup>T<sub>E</sub>X gets to the end of the line and inserts a space character there, after the 16.

Printing works, but as the last example demonstrates, you have to be careful about spacing if you have text immediately after the printed content. In that case, it's usually best to assemble text within a PythonTeX environment and store the text in a variable. Then you can bring in the text later, using the `??` command. The `??` command brings in a string representation of its argument. First we create the text. `?? PythonTeX ??` Then we bring it in: `??`. The `??` command can even bring in verbatim content, as long as you are using the pdfTeX or XeTeX engines. (Unfortunately, verbatim does not work with LuaTeX, due to a bug.<sup>†</sup>)

We don't have to typeset the code we're executing. It can be hidden. And then we can access it later: `??`.

---

\*Since PythonTeX runs Python code (and potentially other code) on your computer, documents using PythonTeX have a greater potential for security risks than do standard L<sup>A</sup>T<sub>E</sub>X documents. You should only compile PythonTeX documents from sources you trust.

<sup>†</sup><http://tracker.luatex.org/view.php?id=733>

## 2 Pygments highlighting

PythonTeX supports syntax highlighting via Pygments. Any language supported by Pygments can be highlighted. Unicode is supported. Consider this snippet copied and pasted from a Python 3 interactive session. (Using random strings of Unicode for variable names is probably not a good idea, but PythonTeX will happily highlight it for you.)

```
?? PythonTeX ??
```

There is also a Pygments command for inline use: `??`.

## 3 Python console environment

PythonTeX includes an environment that emulates a Python interactive session. Commands are entered within the environment, each line is treated as input to an interactive session, and the result is typeset.

```
?? PythonTeX ??
```

It is possible to refer to the values of console variables later on in inline contexts, using the `??` command. For example, the value of  $z$  was `??`.

## 4 Basic SymPy interaction

PythonTeX allows us to perform algebraic manipulations with SymPy and then properly typeset the results.

We create three variables, and define  $z$  in terms of the other two.

```
?? PythonTeX ??
```

Now we can access what  $z$  is equal to:

$$z = ??$$

Many things are possible, including some very nice calculus.

```
?? PythonTeX ??
```

$$?? = ??$$

It's easy to use arbitrary symbols in equations.

```
?? PythonTeX ??
```

$$?? = ??$$

## 5 Plots with matplotlib

We can create plots with matplotlib, perfectly matching the plot fonts with the document fonts. No more searching for the code that created a figure!

It is possible to pass page dimensions and similar contextual information from the L<sup>A</sup>T<sub>E</sub>X side to the Python side. If you want your figures to be, for example, a particular fraction of the page width, you can pass the value of `??` to the Python side, and use it in creating your figures. See `??` in the main documentation for details.

You may want to use matplotlib's PGF backend when creating plots.  
?? PythonTeX ??

## 6 Basic pylab interaction

?? PythonTeX ??

$$\int_0^\infty e^{-x^2} dx = ??$$

## 7 An automated derivative and integral table

PythonTeX allows some amazing document automation, such as this derivative and integral table. Try typing that by hand, fast!

?? PythonTeX ??  
?? PythonTeX ??

## 8 Step-by-step solutions

Using SymPy, it is possible to typeset step-by-step solutions. In this particular case, we also use the `mdframed` package to place a colored background behind our code.

### Step-by-Step Integral Evaluation

?? PythonTeX ??

?? PythonTeX ??

## 9 Including stderr

PythonTeX allows code to be typeset next to the stderr it produces. This requires the package option `makestderr`.

?? PythonTeX ??

This code causes a syntax error:

?? PythonTeX ??

The package option `stderrfilename` allows the file name that appears in the error message to be customized.