

PythonTeX Gallery

Geoffrey M. Poore

August 5, 2012

Abstract

PythonTeX allows you to run Python code from within L^AT_EX 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 pull in the results.

This can be simple. For example, `print('Python says hi!')` returns the following:

Python says hi!

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

Python says hi!

Python interaction can also be more complex. `print(str(2**2**2) + r'\endinput')` returns 16. In this case, the printed results include L^AT_EX code, which is correctly interpreted by L^AT_EX 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 prevents L^AT_EX from treating the newline at the end of the file as justification for a space character.

But we don't have to typeset the code. It can be hidden. And then we can access it later: **This is a message from Python.**

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.)

```
>>> â æ é ö ø = 123
>>> ß ç ñ ð Š = 456
>>> â æ é ö ø + ß ç ñ ð Š
579
```

*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^AT_EX documents. You should only compile PythonTeX documents from sources you trust.

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.

```
>>> x = 123
>>> y = 345
>>> z = x + y
>>> z
468
>>> def f(expr):
...     return(expr**4)
...
>>> f(x)
228886641
>>> print('Python says hi from the console!')
Python says hi from the console!
```

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.

```
var('x, y, z')
z = x + y
```

Now we can access what z is equal to:

$$z = x + y$$

Many things are possible, including some very nice calculus.

```
f = x**3 + cos(x)**5
g = Integral(f,x)
```

$$\int x^3 + \cos^5(x) dx = \frac{1}{4}x^4 + \frac{1}{5}\sin^5(x) - \frac{2}{3}\sin^3(x) + \sin(x)$$

It's easy to use arbitrary symbols in equations.

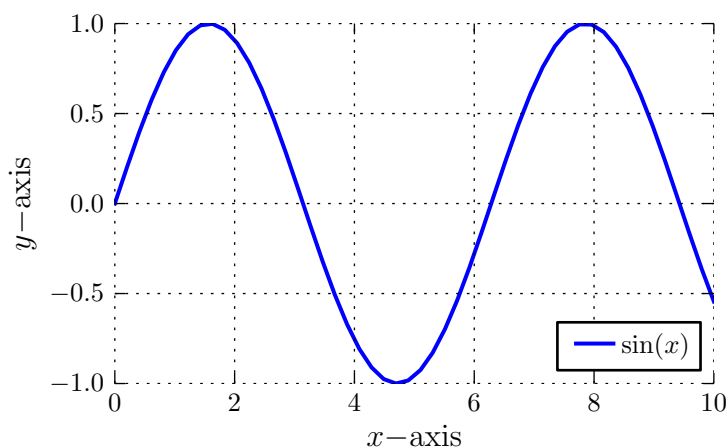
```
phi = Symbol('phi')
h = Integral(exp(-phi**2), (phi,0,oo))
```

$$\int_0^\infty e^{-\phi^2} d\phi = \frac{1}{2}\sqrt{\pi}$$

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!

```
rc('text', usetex=True)
rc('font', family='serif')
rc('font', size=10.0)
rc('legend', fontsize=10.0)
rc('font', weight='normal')
x = linspace(0,10)
figure(figsize=(4,2.5))
plot(x, sin(x), label='$\sin(x)$')
xlabel(r'$x\mathrm{-axis}$')
ylabel(r'$y\mathrm{-axis}$')
legend(loc='lower right')
# asynchronous version of plt.savefig()
aps.savefig('myplot.pdf', bbox_inches='tight')
```



5.1 Asynchronous saving of plots

The above example also shows the usage of the AsyncPyLabSave package which is shipped with PythonTeX, it works the same way as `plt.savefig()` except that it does not block the main program. Please see the documentation for a few multiprocessing caveats.

6 Basic pylab interaction

```
from scipy.integrate import quad
myintegral = quad(lambda x: e**-x**2, 0, inf)[0]
```

$$\int_0^{\infty} e^{-x^2} dx = 0.886226925453$$

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!

An Automated Derivative and Integral Table

```
1  from re import sub
2
3  var('x')
4
5  #Create a list of functions to include in the table
6  funcs = ['sin(x)', 'cos(x)', 'tan(x)',
7           'sin(x)**2', 'cos(x)**2', 'tan(x)**2',
8           'asin(x)', 'acos(x)', 'atan(x)',
9           'sinh(x)', 'cosh(x)', 'tanh(x)']
10
11  print(r'\begin{align*}')
12
13  for func in funcs:
14      #Put in some vertical space when switching to arc and hyperbolic funcs
15      if func=='asin(x)' or func=='sinh(x)':
16          print(r'\vspace{0.5in}\\')
17      myderiv = 'Derivative(' + func + ', x)'
18      myint = 'Integral(' + func + ', x)'
19      print(latex(eval(myderiv)) + '&= ' \
20            + latex(eval(myderiv+'.doit()')) + r'\quad & \quad')
21      print(latex(eval(myint)) + '&= ' \
22            + latex(eval(myint+'.doit()')) + r'\\')
23  print(r'\end{align*}')
```

$$\begin{aligned}\frac{\partial}{\partial x} \sin(x) &= \cos(x) \\ \frac{\partial}{\partial x} \cos(x) &= -\sin(x) \\ \frac{\partial}{\partial x} \tan(x) &= \tan^2(x) + 1 \\ \frac{\partial}{\partial x} \sin^2(x) &= 2 \sin(x) \cos(x) \\ \frac{\partial}{\partial x} \cos^2(x) &= -2 \sin(x) \cos(x) \\ \frac{\partial}{\partial x} \tan^2(x) &= (2 \tan^2(x) + 2) \tan(x)\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial x} \operatorname{asin}(x) &= \frac{1}{\sqrt{-x^2 + 1}} \\ \frac{\partial}{\partial x} \operatorname{acos}(x) &= -\frac{1}{\sqrt{-x^2 + 1}} \\ \frac{\partial}{\partial x} \operatorname{atan}(x) &= \frac{1}{x^2 + 1}\end{aligned}$$

$$\begin{aligned}\frac{\partial}{\partial x} \sinh(x) &= \cosh(x) \\ \frac{\partial}{\partial x} \cosh(x) &= \sinh(x) \\ \frac{\partial}{\partial x} \tanh(x) &= -\tanh^2(x) + 1\end{aligned}$$

$$\int \sin(x) dx = -\cos(x)$$

$$\int \cos(x) dx = \sin(x)$$

$$\int \tan(x) dx = \frac{1}{2} \log(\tan^2(x) + 1)$$

$$\int \sin^2(x) dx = \frac{1}{2}x - \frac{1}{2} \sin(x) \cos(x)$$

$$\int \cos^2(x) dx = \frac{1}{2}x + \frac{1}{2} \sin(x) \cos(x)$$

$$\int \tan^2(x) dx = -x + \tan(x)$$

$$\int \operatorname{asin}(x) dx = x \operatorname{asin}(x) + \sqrt{-x^2 + 1}$$

$$\int \operatorname{acos}(x) dx = x \operatorname{acos}(x) - \sqrt{-x^2 + 1}$$

$$\int \operatorname{atan}(x) dx = x \operatorname{atan}(x) - \frac{1}{2} \log(x^2 + 1)$$

$$\int \sinh(x) dx = \cosh(x)$$

$$\int \cosh(x) dx = \sinh(x)$$

$$\int \tanh(x) dx = -x - \log(\tanh(x) - 1)$$

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

```
1 (x, y, z) = symbols('x,y,z')
2 f = Symbol('f(x,y,z)')
3
4 # Define limits of integration
5 x_llim = 0
6 x_ulim = 2
7 y_llim = 0
```

```

8  y_ulim = 3
9  z_llim = 0
10 z_ulim = 4
11
12 print(r'\begin{align*}')
13
14 # Notice how I define f as a symbol, then later as an actual function
15 left = Integral(f, (x,x_llim,x_ulim), (y,y_llim,y_ulim), (z,z_llim,z_ulim))
16 f = x*y+y*sin(z) + cos(x+y)
17 right = Integral(f, (x,x_llim,x_ulim), (y,y_llim,y_ulim), (z,z_llim,z_ulim))
18 print(latex(left) + '&=' + latex(right)+r'\\')
19
20 # For each step, I move limits from an outer integral to an inner, evaluated
21 # integral until the outer integral is no longer needed
22 right = Integral(Integral(f,(z,z_llim,z_ulim)).doit(), (x,x_llim,x_ulim), \
23                  (y,y_llim,y_ulim))
24 print('&=' + latex(right)+r'\\')
25
26 right = Integral(Integral(f,(z,z_llim,z_ulim),(y,y_llim,y_ulim)).doit(), \
27                  (x,x_llim,x_ulim))
28 print('&=' + latex(right)+r'\\')
29
30 right = Integral(f,(z,z_llim,z_ulim),(y,y_llim,y_ulim),(x,x_llim,x_ulim)).doit()
31 print('&=' + latex(right)+r'\\')
32
33 print('&=' + latex(N(right))+r'\\')
34
35 print(r'\end{align*}')

```

$$\begin{aligned}
\int_0^4 \int_0^3 \int_0^2 f(x,y,z) \, dx \, dy \, dz &= \int_0^4 \int_0^3 \int_0^2 xy + y \sin(z) + \cos(x+y) \, dx \, dy \, dz \\
&= \int_0^3 \int_0^2 4xy - y \cos(4) + y + 4 \cos(x+y) \, dx \, dy \\
&= \int_0^2 18x - 4 \sin(x) + 4 \sin(x+3) - \frac{9}{2} \cos(4) + \frac{9}{2} \, dx \\
&= 4 \cos(3) + 4 \cos(2) - 4 \cos(5) - 9 \cos(4) + 41 \\
&= 40.1235865133293
\end{aligned}$$

9 Including stderr

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

```
1 x = 123
2 y = 345
3 z = x + y +
```

This code causes a syntax error:

```
File "py_errorsession_9.py", line 3
    z = x + y +
               ^
```

SyntaxError: invalid syntax

The file name that appears in the message can be customized using the package option `stderrfilename`.