PythonTEX Gallery

Geoffrey M. Poore

July 17, 2012

Abstract

PythonTEX allows you to run Python code from within LaTeX 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 LaTeX code, which is correctly interpreted by LaTeX 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 LaTeX 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 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
```

*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 LaTeX documents. You should only compile PythonTEX documents from sources you trust.

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

3 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.

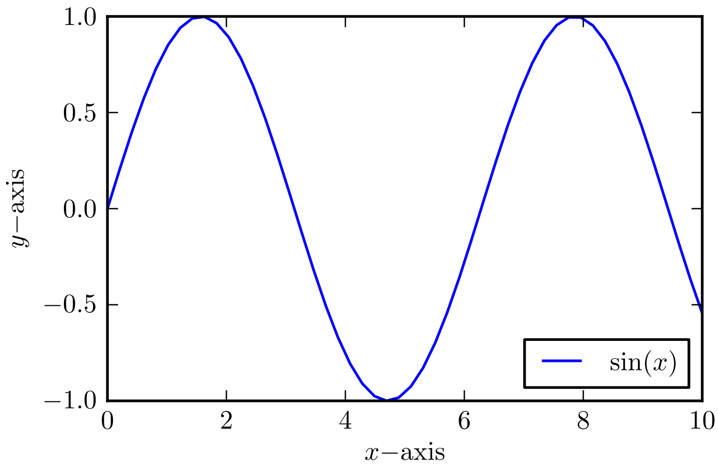
```
phi = Symbol(r'\phi')
h = Integral(exp(-phi**2), (phi,0,oo))
```

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

4 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')
savefig('myplot.pdf', bbox_inches='tight')
```



5 Basic pylab interaction

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

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

6 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

```
from re import sub

var('x')
```

```
#Create a list of functions to include in the table
funcs = ['sin(x)', 'cos(x)', 'tan(x)',
         'sin(x)**2', 'cos(x)**2', 'tan(x)**2',
         'asin(x)', 'acos(x)', 'atan(x)',
         'sinh(x)', 'cosh(x)', 'tanh(x)']

print(r'\begin{align*}')

for func in funcs:
    #Put in some vertical space when switching to arc and hyperbolic funcs
    if func=='asin(x)' or func=='sinh(x)':
        print(r'\vspace{0.5in}\\')
    myderiv = 'Derivative(' + func + ', x)'
    myint = 'Integral(' + func + ', x)'
    print(latex(eval(myderiv)) + '&=' \
            + latex(eval(myderiv+'.doit()')) + r'\quad & \quad')
    print(latex(eval(myint)) + '&=' \
            + latex(eval(myint+'.doit()'))+ r'\\')
print(r'\end{align*}')
```

$$\frac{\partial}{\partial x} \sin(x) = \cos(x) \qquad \int \sin(x)\, dx = -\cos(x)$$

$$\frac{\partial}{\partial x} \cos(x) = -\sin(x) \qquad \int \cos(x)\, dx = \sin(x)$$

$$\frac{\partial}{\partial x} \tan(x) = \tan^2(x) + 1 \qquad \int \tan(x)\, dx = \frac{1}{2}\log\left(\tan^2(x) + 1\right)$$

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

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

$$\frac{\partial}{\partial x} \tan^2(x) = \left(2\tan^2(x) + 2\right)\tan(x) \qquad \int \tan^2(x)\, dx = -x + \tan(x)$$

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

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

$$\frac{\partial}{\partial x} \operatorname{atan}(x) = \frac{1}{x^2 + 1} \qquad \int \operatorname{atan}(x)\, dx = x\operatorname{atan}(x) - \frac{1}{2}\log\left(x^2 + 1\right)$$

$$\frac{\partial}{\partial x} \sinh(x) = \cosh(x) \qquad \int \sinh(x)\, dx = \cosh(x)$$

$$\frac{\partial}{\partial x} \cosh(x) = \sinh(x) \qquad \int \cosh(x)\, dx = \sinh(x)$$

$$\frac{\partial}{\partial x} \tanh(x) = -\tanh^2(x) + 1 \qquad \int \tanh(x)\, dx = -x - \log(\tanh(x) - 1)$$

7 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

```
(x, y, z) = symbols('x,y,z')
f = Symbol('f(x,y,z)')

# Define limits of integration
x_llim = 0
```

```python
x_ulim = 2
y_llim = 0
y_ulim = 3
z_llim = 0
z_ulim = 4

print(r'\begin{align*}')

# Notice how I define f as a symbol, then later as an actual function
left = Integral(f, (x,x_llim,x_ulim), (y,y_llim,y_ulim), (z,z_llim,z_ulim))
f = x*y+y*sin(z) + cos(x+y)
right = Integral(f, (x,x_llim,x_ulim), (y,y_llim,y_ulim), (z,z_llim,z_ulim))
print(latex(left) + '&=' + latex(right)+r'\\')

# For each step, I move limits from an outer integral to an inner, evaluated
# integral until the outer integral is no longer needed
right = Integral(Integral(f,(z,z_llim,z_ulim)).doit(), (x,x_llim,x_ulim), \
        (y,y_llim,y_ulim))
print('&=' + latex(right)+r'\\')

right = Integral(Integral(f,(z,z_llim,z_ulim),(y,y_llim,y_ulim)).doit(), \
        (x,x_llim,x_ulim))
print('&=' + latex(right)+r'\\')

right = Integral(f,(z,z_llim,z_ulim),(y,y_llim,y_ulim),(x,x_llim,x_ulim)).doit()
print('&=' + latex(right)+r'\\')

print('&=' + latex(N(right))+r'\\')

print(r'\end{align*}')
```

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

8 Including stderr

PythonTEX allows code to be typset next to the stderr it produces. This requires the package option `stderr`

```
x = 123
y = 345
z = x + y +
```

This code causes a syntax error:

```
  File "py_errorsession_8.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`.