Reproducible Documents with **PythonTeX**

Geoffrey M. Poore
Department of Physics, Union University

SciPy 2013 — June 27

Reproducible documents

- Reproducible workflow
    - Results and figures created in reproducible manner
    - Makefiles, ...

Reproducible documents

- Reproducible workflow
  - Results and figures created in reproducible manner
  - Makefiles, ...
- Embedded code
  - Code embedded in document
  - Roots in literate programming
  - Sweave, knitr, IPython, SageTeX, SympyTeX, Pweave, ...

Reproducible documents

- Reproducible workflow
  - Results and figures created in reproducible manner
  - Makefiles, ...
- Embedded code
  - Code embedded in document
  - Roots in literate programming
  - Sweave, knitr, IPython, SageTeX, SympyTeX, Pweave, ...
  - **PythonTeX**: emphasize LaTeX integration, usability, performance

Using PythonTeX

- Install from CTAN or `github.com/gpoore/pythontex`
- `\usepackage{pythontex}`
- 3-step compile when Python code needs to run (otherwise, just run LaTeX; all output is cached)
  - LaTeX $\rightarrow$ saves Python code in auxiliary file
  - PythonTeX $\rightarrow$ executes Python code
  - LaTeX $\rightarrow$ brings Python output into document

Using PythonTeX

LaTeX code

```
\begin{pycode}
from __future__ import division
num = 3571/1597
print(num)
\end{pycode}

\pyc{print(853/17)}

\py{num}

$ \sympy{pi*cos(2)} = \sympy{N(pi*cos(2))} $
```

Output

2.2360676268
50.1764705882
2.2360676268
$\pi \cos{\left(2 \right)} = -1.30736384451114$

Example analysis: 100-step random walk

Random walk data: `walk_data.txt`

```
-1
0
-1
0
1
2
...
```

Objectives

- Find the average distance and max distance from the origin during the walk
- Plot the walk

Load data, calculate average and max

LaTeX code

```
\begin{pycode}
f = open('walk_data.txt')
dist_data = [abs(int(n)) for n in f.readlines()]
f.close()

max_dist = max(dist_data)
ave_dist = sum(dist_data)/len(dist_data)
\end{pycode}

Average distance:  \py{ave_dist}.  Max:  \py{max_dist}.
```

Output

Average distance: 5.46. Max: 12.

Plotting

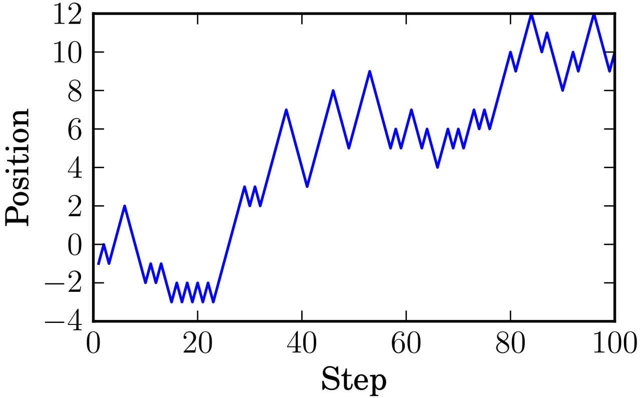
LaTeX code

```
\begin{pycode}[plot-session]
from pylab import *
rc('text', usetex=True)
rc('font', family='serif')

f = open('walk_data.txt')
data = [int(num) for num in f.readlines()]
f.close()

figure(figsize=(3.5,2))
plot(range(1,101), data)
xlabel('Step')
ylabel('Position')
savefig('walk.pdf', bbox_inches='tight')
\end{pycode}

\includegraphics{walk.pdf}
```

- Runs in its own session `plot-session`
- Sessions automatically run in parallel (`multiprocessing`)

Plotting

Output



What if … errors or warnings?

LaTeX code

```
\begin{pycode}
f = open('walk_data.txt')
dist_data = [abs(int(n)) for n in f.readlines()]
f.close()

max_dist = mx(dist_data)
ave_dist = sum(dist_data)/len(dist_data)
\end{pycode}

Average distance:  \py{ave_dist}.  Max:  \py{max_dist}.
```

Output

Average distance: **??**. Max: **??**.

- LaTeX run still completes, with warnings and placeholders for missing content

What if … errors or warnings?

PythonTeX run summary

```
This is PythonTeX v0.12

----  Messages for py:default:default  ----
  Traceback (most recent call last):
* PythonTeX stderr - error on line 157:
    File "<outputdir>\py_default_default.py", line 51, in <module>
      max_dist = mx(data)
  NameError: name 'mx' is not defined

--------------------------------------------------
PythonTeX:  scipy_talk_2013 - 1 error(s), 0 warning(s)
```

- Worst-case scenario: PythonTeX can only trace an error or warning back to a single command or environment, rather than to a single line (delimiters in `stderr`)

What if ... errors or warnings?

Package option `rerun`: threshold for re-executing code

- `never`
- `modified`
- `errors`
- `warnings`
- `always`

Remember: All output is cached

What if … new data?

- PythonTeX provides a utilities class
- An instance called `pytex` is always created

What if ... new data?

- PythonTeX provides a utilities class
- An instance called `pytex` is always created
- The `add_dependencies()` method tells PythonTeX to track dependencies
  - Changes determined by modification time or hash

What if … new data?

- PythonTeX provides a utilities class
- An instance called `pytex` is always created
- The `add_dependencies()` method tells PythonTeX to track dependencies
    - Changes determined by modification time or hash

Fixed code

```
f = open('walk_data.txt')
pytex.add_dependencies('walk_data.txt')
```

What if … new data?

- PythonTeX provides a utilities class
- An instance called `pytex` is always created
- The `add_dependencies()` method tells PythonTeX to track dependencies
    - Changes determined by modification time or hash

Fixed code

```
f = open('walk_data.txt')
pytex.add_dependencies('walk_data.txt')
```

- Could define a custom `open()` that automatically tracks any files opened for reading

Converting to other formats

`depythontex`

- LaTeX document with PythonTeX $\rightarrow$ plain LaTeX document
  - Substitute all Python-generated content
  - Convert typeset code into `verbatim`, `fancyvrb`, `listings`, or `minted` format
- Output suitable for conversion to HTML via Pandoc, journal submission, etc.

Supporting other languages

- Adding **full** support requires two templates and a new class instance, probably less than 100 lines of code total
  - Most of code implements equivalent of PythonTeX utilities

Supporting other languages

- Adding **full** support requires two templates and a new class instance, probably less than 100 lines of code total
  - Most of code implements equivalent of PythonTeX utilities
- Just added: Ruby

LaTeX code

```
\begin{rubycode}
puts "Ruby says hello."
\end{rubycode}
```

Output

```
Ruby says hello.
```

- Coming soon: Julia

Conclusion

Future directions

- Additional features for debugging and fine-tuning
- Better support for macro programming and LaTeX packages using PythonTeX

  `\newcommand{\power}[2]{\py{#1**#2}}`

  `\power{2}{4}` $\Rightarrow$ 16
- Interface for markdown, reST, others?

`github.com/gpoore/pythontex`