Overview

A Question of Primes

Introduction to PythonTEX

Mathematics with Sympy

Plots with matplotlib

Web Services

Conclusions

Extra Examples

A Gentle Introduction to PythonTEX

Andrew Mertz   William Slough

Mathematics and Computer Science Department
Eastern Illinois University

October 23, 2013

Andrew Mertz, William Slough

Overview

A Question of Primes

Introduction to PythonTeX

Mathematics with Sympy

Plots with matplotlib

Web Services

Conclusions

Extra Examples

python™

TeX

Overview

A Question of Primes

Introduction to PythonTEX

Mathematics with Sympy

Plots with matplotlib

Web Services

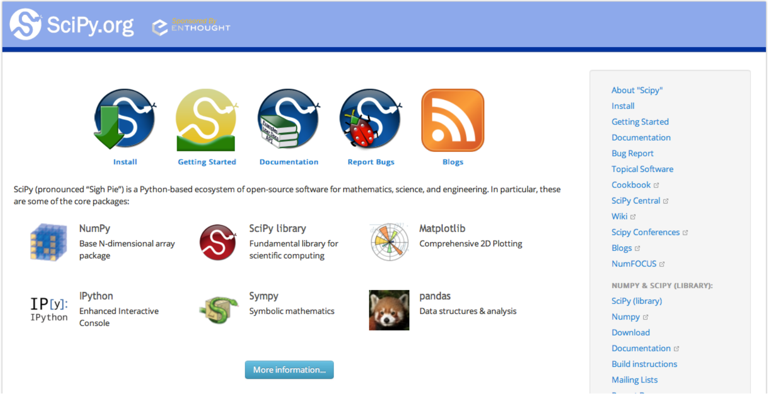
Conclusions

Extra Examples

Python Overview

- General purpose, high-level programming language
- Multi-paradigm: object-oriented, imperative, functional
- Comprehensive standard library
- Origins from late 1989
- Free and open-source

Python + Scientific Computing



A Recent Question on TeX Stack Exchange

> "I would like to write a LaTeX script that produces all the prime numbers between the numbers n and m, where $n < m$. How can I do this? I feel it should not be that hard, but I cannot seem to program it."
>
> — Kevin†

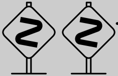
† tex.stackexchange.com/questions/134305/how-to-produce-a-list-of-prime-numbers-in-latex/134366#134366

From The TEXBook, Page 218 (1984)

The first 30 prime numbers are 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, and 113. You may not find this fact very startling; but you may be surprised to learn that the previous sentence was typeset by saying

```
The first thirty prime numbers are \primes{30}.
```

TEX did all the calculation by expanding the `primes` macro, so the author is pretty sure that the list of prime numbers given above is quite free of typographic errors.

Knuth's Code, -worthy

A Gentle Introduction to PythonTEX

Andrew Mertz, William Slough

Overview

A Question of Primes

Introduction to PythonTEX

Mathematics with Sympy

Plots with matplotlib

Web Services

Conclusions

Extra Examples

```
\newif\ifprime \newif\ifunknown % boolean variables
\newcount\n \newcount\p \newcount\d \newcount\a % integer variables
\def\primes#1{2,~3% assume that #1 is at least 3
\n=#1 \advance\n by-2 % n more to go
\p=5 % odd primes starting with p
\loop\ifnum\n>0 \printifprime\advance\p by2 \repeat}
\def\printp{, % we will invoke \printp if p is prime
\ifnum\n=1 and~\fi % and precedes the last value
\number\p \advance\n by -1 }
\def\printifprime{\testprimality \ifprime\printp\fi}
\def\testprimality{{\d=3 \global\primetrue
\loop\trialdivision \ifunknown\advance\d by2 \repeat}}
\def\trialdivision{\a=\p \divide\a by\d
\ifnum\a>\d \unknowntrue\else\unknownfalse\fi
\multiply\a by\d
\ifnum\a=\p \global\primefalse\unknownfalse\fi}
```

Overview

A Question of Primes

Introduction to PythonTEX

Mathematics with Sympy

Plots with matplotlib

Web Services

Conclusions

Extra Examples

David Carlisle's Response

```
\makeatletter
\def\primes#1#2{{%
  \def\comma{\def\comma{, }}%
  \count@\@ne\@tempcntb#2\relax\@curtab#1\relax
  \@primes}}
\def\@primes{\loop\advance\count@\@ne
\expandafter\ifx\csname p-\the\count@\endcsname\relax
\ifnum\@tempcntb<\count@\else
  \ifnum\count@<\@curtab\else\comma\the\count@\fi\fi\else\repeat
\@tempcnta\count@\loop\advance\@tempcnta\count@
\expandafter\let\csname p-\the\@tempcnta\endcsname\@ne
\ifnum\@tempcnta<\@tempcntb\repeat
\ifnum\@tempcntb>\count@\expandafter\@primes\fi}
\makeatother
```

Karl Koeller's Response

A solution using the \pgfmathisprime macro provided by Alain Matthes' tkz-euclide package:

```
\usepackage{tkz-euclide}

\newif\ifcomma

\newcommand{\primes}[2]{%
  \commafalse%
  \foreach\numb in {#1,...,#2}{%
     \pgfmathisprime{\numb}%
     \ifnum\pgfmathresult=1
       \ifcomma, \numb\else\numb\global\commatrue\fi%
     \fi%
  }%
}
```

Can PythonTEX Make This Simpler?

Yes. . .

Overview

A Question of Primes

Introduction to PythonTEX

Mathematics with Sympy

Plots with matplotlib

Web Services

Conclusions

Extra Examples

Evaluating Expressions With `\py`

The macro `\py{expression}` evaluates a Python expression and typesets its **value**.

```
Did you know that $2^{65} = \py{2**65}$?
```

Did you know that $2^{65} = 36893488147419103232$?

Overview

A Question of Primes

Introduction to PythonTEX

Mathematics with Sympy

Plots with matplotlib

Web Services

Conclusions

Extra Examples

Evaluating Expressions With `\pyc`

The macro `\pyc{expression}` evaluates a Python expression and typesets anything that it **prints**.

```
Did you know that $2^{65} = \pyc{print(2**65)}$?
```

Did you know that $2^{65} = 36893488147419103232$?

While "printing" adds little in this case, it is important for more complex examples.

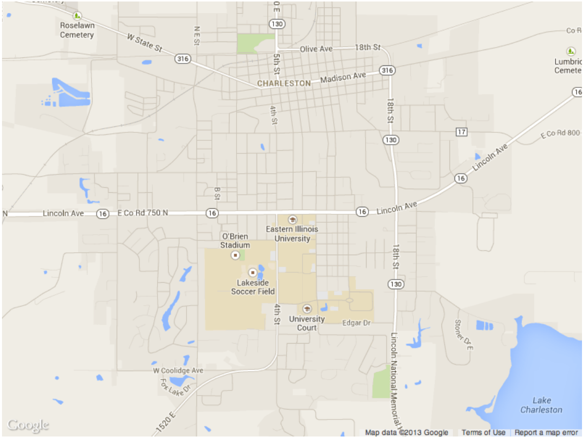
A More Complex Example Using \pyc

\pyc{showGoogleMap("Tokyo", 11)}

A Gentle Introduction to PythonTEX
Andrew Mertz, William Slough
Overview
A Question of Primes
Introduction to PythonTEX
Mathematics with Sympy
Plots with matplotlib
Web Services
Conclusions
Extra Examples

Andrew Mertz, William Slough

Charleston, Illinois USA




```
\pyc{showGoogleMap("600 Lincoln,Charleston,IL", 14)}
```

Generating Tables With `pycode`

```
\begin{pycode}
print(r"\begin{tabular}{c|c}")
print(r"$m$ & $2^m$ \\ \hline")
print(r"%d & %d \\" % (1, 2**1))
print(r"%d & %d \\" % (2, 2**2))
print(r"%d & %d \\" % (3, 2**3))
print(r"%d & %d \\" % (4, 2**4))
print(r"\end{tabular}")
\end{pycode}
```

```
\begin{tabular}{c|c}
$m$ & $2^m$ \\ \hline
1 & 2 \\
2 & 4 \\
3 & 8 \\
4 & 16 \\
\end{tabular}
```

Overview

A Question of Primes

Introduction to PythonTEX

Mathematics with Sympy

Plots with matplotlib

Web Services

Conclusions

Extra Examples

Generating Tables With `pycode`

```
\begin{pycode}
print(r"\begin{tabular}{c|c}")
print(r"$m$ & $2^m$ \\ \hline")
print(r"%d & %d \\" % (1, 2**1))
print(r"%d & %d \\" % (2, 2**2))
print(r"%d & %d \\" % (3, 2**3))
print(r"%d & %d \\" % (4, 2**4))
print(r"\end{tabular}")
\end{pycode}
```

m	2^m
1	2
2	4
3	8
4	16

Generating Tables With a Loop

```
\begin{pycode}
lo, hi = 1, 6
print(r"\begin{tabular}{c|c}")
print(r"$m$ & $2^m$ \\ \hline")
for m in range(lo, hi + 1):
  print(r"%d & %d \\" % (m, 2**m))
print(r"\end{tabular}")
\end{pycode}
```

m	2^m
1	2
2	4
3	8
4	16
5	32
6	64

Defining a Function

```
\begin{pycode}
def fib(n):   # nth Fibonacci value
  a, b = 0, 1
  for i in range(n):
    a, b = b, a + b
  return a
\end{pycode}
```

```
Did you know that $F_{10} = \py{fib(10)}$?
```

Did you know that $F_{10} = 55$?

Sessions

`py`, `pyc`, and `pycode` all have an optional `session` argument.

This argument determines the name of the Python session in which the code is executed.

Sessions with different names may be executed in parallel providing a speedup.

If a session is not specified, then the default session is used.

Introducing pythontexcustomcode

```
\begin{pythontexcustomcode}{py}
def makeTable(lo, hi):
  print(r"\begin{tabular}{c|c}")
  print(r"$m$ & $2^m$ \\ \hline")
  for m in range(lo, hi + 1):
    print(r"%d & %d \\" % (m, 2**m))
  print(r"\end{tabular}")
\end{pythontexcustomcode}
```

The `pythontexcustomcode` environment evaluates the code block at the start of each "session" – which makes it a great place to define well-tested functions.

```
\begin{pythontexcustomcode}{py}
python code block
\end{pythontexcustomcode}
```

Overview

A Question of Primes

Introduction to PythonTEX

Mathematics with Sympy

Plots with matplotlib

Web Services

Conclusions

Extra Examples

Generating Tables in Multiple Sessions

```
\pyc[table1]{makeTable(1, 4)}
```

m	2^m
1	2
2	4
3	8
4	16

```
\pyc[table2]{makeTable(4, 10)}
```

m	2^m
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

Overview
A Question of Primes
Introduction to PythonTEX
Mathematics with Sympy
Plots with matplotlib
Web Services
Conclusions
Extra Examples

Generating Tables From a Function

```
\begin{pythontexcustomcode}{py}
def makeTableFromFunction(lo, hi, funct, label):
  print(r"\begin{tabular}{c|c}")
  print(r"$m$ & %s \\ \hline" % label)
  for m in range(lo, hi + 1):
    print(r"%d & %d \\" % (m, funct(m)))
  print(r"\end{tabular}")
\end{pythontexcustomcode}

\pyc{makeTableFromFunction(7, 11, fib, "$F_{m}$")}
```

m	F_m
7	13
8	21
9	34
10	55
11	89

Generating Tables From a Library Function

Python excels in the quantity and quality of its modules.

Modules make additional functions available. To use them, the corresponding module needs to be imported.

```
\begin{pythontexcustomcode}{py}
import math
\end{pythontexcustomcode}
```

```
\pyc{makeTableFromFunction(30, 33, math.factorial, "$m!$")}
```

m	$m!$
30	265252859812191058636308480000000
31	8222838654177922817725562880000000
32	263130836933693530167218012160000000
33	8683317618811886495518194401280000000

Generating Tables From a Library Function

With the import statement the module name is needed each time a member of the module is used.

To avoid this, a `from import` statement can be used.

This can shadow other functions and should be used with care.

```
\begin{pythontexcustomcode}{py}
from math import factorial
\end{pythontexcustomcode}
```

```
\pyc{makeTableFromFunction(30, 33, factorial, "$m!$")}
```

m	$m!$
30	265252859812191058636308480000000
31	8222838654177922817725562880000000
32	263130836933693530167218012160000000
33	8683317618811886495518194401280000000

Remember Kevin?

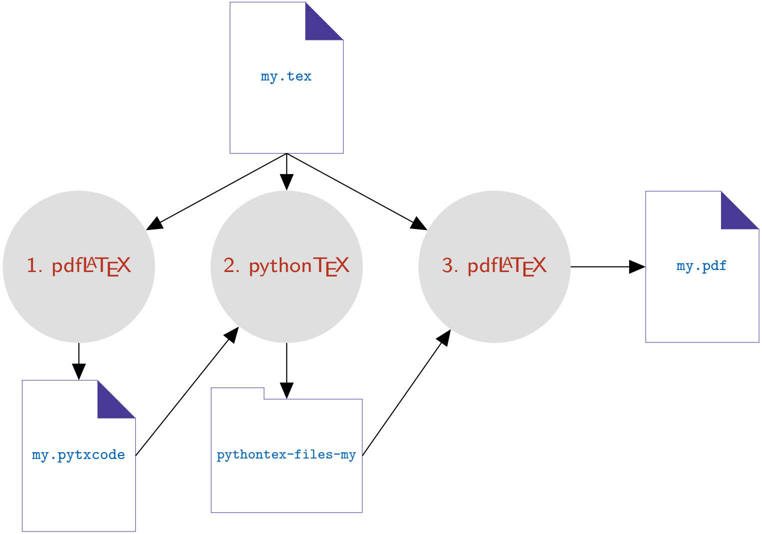
```
\begin{pythontexcustomcode}{py}
from sympy import prime

def generatePrimes(n):    # Assume n >= 3
  for i in range(1, n):
    print("%d, "  % prime(i))
  print("and %d%%" % prime(n))
\end{pythontexcustomcode}
```

```
The first 30 primes are \pyc{generatePrimes(30)}.
```

The first 30 primes are 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, and 113.

Processing PythonTEX Files



Symbolic Mathematics With Sympy

```
>>> from sympy import *

>>> var("x, y")     # Define symbolic variables
(x, y)

>>> z = (x + y)**3 # Define an expression

>>> z               # Display z
(x + y)**3

>>> expand(z)       # Display the expansion of z
x**3 + 3*x**2*y + 3*x*y**2 + y**3

>>> latex(expand(z))
'x^{3} + 3 x^{2} y + 3 x y^{2} + y^{3}'
```

Expanding Binomials

```
\begin{pycode}
from sympy import *
var("x, y")

binomials = []
for m in range(3, 6):
  binomials.append((x + y)**m)

print(r"\begin{align*}")
for expr in binomials:
  print(r"%s &= %s\\" % (latex(expr), latex(expand(expr))))
print(r"\end{align*}")
\end{pycode}
```

$$\begin{aligned}
(x+y)^3 &= x^3 + 3x^2y + 3xy^2 + y^3\\
(x+y)^4 &= x^4 + 4x^3y + 6x^2y^2 + 4xy^3 + y^4\\
(x+y)^5 &= x^5 + 5x^4y + 10x^3y^2 + 10x^2y^3 + 5xy^4 + y^5
\end{aligned}$$

Overview

A Question of Primes

Introduction to PythonTEX

Mathematics with Sympy

Plots with matplotlib

Web Services

Conclusions

Extra Examples

A Little Bit of Calculus

```
\begin{pycode}
functions = [sin(x), cos(x), tan(x)]
print(r"\begin{align*}")
for f in functions:
  d = Derivative(f, x)
  print(latex(d) + "&=" + latex(d.doit()) + r"\\")
print(r"\end{align*}")
\end{pycode}
```

$$\frac{d}{dx} \sin\left(x\right) = \cos\left(x\right)$$

$$\frac{d}{dx} \cos\left(x\right) = - \sin\left(x\right)$$

$$\frac{d}{dx} \tan\left(x\right) = \tan^{2}\left(x\right) + 1$$

Andrew Mertz, William Slough

Overview

A Question of Primes

Introduction to PythonTEX

Mathematics with Sympy

Plots with matplotlib

Web Services

Conclusions

Extra Examples

A Little Bit More

```
\begin{pycode}
functions = [sin(x), cos(x), tan(x)]
print(r"\begin{align*}")
for f in functions:
  i = Integral(f, x)
  print(latex(i) + "&=" + latex(i.doit()) + r"\\")
print(r"\end{align*}")
\end{pycode}
```

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

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

$$\int \tan(x)\, dx = -\frac{1}{2} \log\left(\sin^{2}(x) - 1\right)$$

Andrew Mertz, William Slough

Overview

A Question of Primes

Introduction to PythonTEX

Mathematics with Sympy

Plots with matplotlib

Web Services

Conclusions

Extra Examples

Stirling's Triangle

Stirling's Triangle for Subsets

n	$\left\{ n \atop 0 \right\}$	$\left\{ n \atop 1 \right\}$	$\left\{ n \atop 2 \right\}$	$\left\{ n \atop 3 \right\}$	$\left\{ n \atop 4 \right\}$	$\left\{ n \atop 5 \right\}$	$\left\{ n \atop 6 \right\}$	$\left\{ n \atop 7 \right\}$	$\left\{ n \atop 8 \right\}$
0	1								
1	0	1							
2	0	1	1						
3	0	1	3	1					
4	0	1	7	6	1				
5	0	1	15	25	10	1			
6	0	1	31	90	65	15	1		
7	0	1	63	301	350	140	21	1	
8	0	1	127	966	1701	1050	266	28	1

Stirling's Triangle (code excerpt)

```
from sympy.functions.combinatorial.numbers import *

for n in range(numberOfRightHandColumns):
  print("%d" % n)
  for k in range(n + 1):
    print("& %d" % stirling(n, k))
  print(r"\\")
```

Overview

A Question of Primes

Introduction to PythonTEX

Mathematics with Sympy

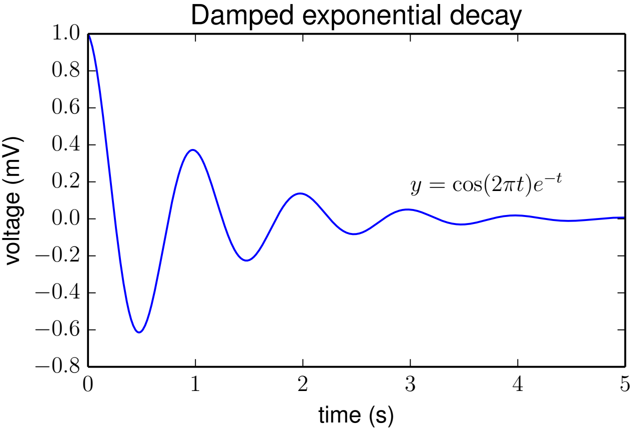
Plots with matplotlib

Web Services

Conclusions

Extra Examples

Plotting With Matplotlib



Inspired by a plot from matplotlib.org/1.3.1/gallery.html

Plot Details, Part 1

```
\begin{pycode}
from pylab import *

# Define f(t), the desired function to plot
def f(t):
  return cos(2 * pi * t) * exp(-t)

# Generate the points (t_i, y_i) to plot
t = linspace(0, 5, 500)
y = f(t)

# Begin with an empty plot, 5 x 3 inches
clf()
figure(figsize=(5, 3))

# Use TeX fonts
rc("text", usetex=True)
```

Andrew Mertz, William Slough

Overview
A Question of Primes
Introduction to PythonTEX
Mathematics with Sympy
Plots with matplotlib
Web Services
Conclusions
Extra Examples

Plot Details, Part 2

```
# Generate the plot with annotations
plot(t, y)
title("Damped exponential decay")
text(3, 0.15, r"$y = \cos(2 \pi t) e^{-t}$")
xlabel("time (s)")
ylabel("voltage (mV)")

# Save the plot as a PDF file
savefig("myplot.pdf", bbox_inches="tight")

# Include the plot in the current LaTeX document
print(r"\begin{center}")
print(r"\includegraphics[width=0.85\textwidth]{myplot.pdf}")
print(r"\end{center}")
\end{pycode}
```

Overview

A Question of Primes

Introduction to PythonTEX

Mathematics with Sympy

Plots with matplotlib

Web Services

Conclusions

Extra Examples

Simple Access of a Web Service

Many powerful and freely available web services can be accessed though the libraries of Python.

Python has excellent JSON, XML and networking libraries.

The first web service we will use is Google's Geocoding API.

Geocoding is the process of converting an address into geographic coordinates such as latitude and longitude.

Reverse geocoding is the process of converting geographic coordinates into a human-readable address.

Using Google's Geocoding Service

Andrew Mertz, William Slough

Overview
A Question of Primes
Introduction to PythonTEX
Mathematics with Sympy
Plots with matplotlib
Web Services
Conclusions
Extra Examples

```
from urllib2 import urlopen
from urllib import urlencode
import json

def findLatlong(address):
  # Build the data needed to call the Goggle API
  query = {"address": address, "sensor": "false"}
  data = urlencode(query)
  url = "http://maps.googleapis.com/maps/api/geocode/json?"
  url += data

  # Fetch and parse
  result = json.load(urlopen(url))
  latlong = result["results"][0]["geometry"]["location"]
  return (latlong["lat"], latlong["lng"])

The latitude and longitude of Tokyo is \py{findLatlong("Tokyo")}
```

The latitude and longitude of Tokyo is (35.6894875, 139.6917064)

Executing a Subprocess

Python can run other programs and use their output.

Here we use webkit2png to render a web page as an image that is included in the document.

```
import subprocess

def showWebpage(url, filename):
  subprocess.call(["webkit2png", "-o", filename,
    "-F", "javascript",
    "-w", "5",
    url])
  print(r"\begin{center}")
  print(r"\includegraphics{%s}" % filename)
  print(r"\end{center}")
```

How the Maps Were Made (pseudocode)

```
def showGoogleMap(address, zoomlevel):
  # Find the latitude and longitude of the address

  # Build a web page with the JavaScript needed to load
  # a Google Map at the given location and zoom level

  # Save the web page to a temporary file

  # Use showWebpage to display the map
```

Overview

A Question of Primes

Introduction to PythonTEX

Mathematics with Sympy

Plots with matplotlib

Web Services

Conclusions

Extra Examples

Issues

- PythonTEX adds significant processing time

  Appropriate use of sessions can reduce this time, but there is still a large overhead.

- Debugging Python code within TEX is difficult

  Test complex Python code outside of TEX first

- TEX macros that have arguments generated by Python fail on first processing step

  Add such TEX macros from within Python

- Use parentheses for print statements: `print(x)`.
- Be clear of the differences between `\py` and `\pyc`.
- When using Beamer use the frame option `fragile=singleslide` if able.
- Be skeptical of SymPy results.
- If all else fails delete the `pythontex-files` folder.

Questions?

References

Python: python.org

SciPy: scipy.org

PythonTEX(Geoffrey Poore): www.ctan.org/pkg/pythontex

Anaconda(Python distribution): store.continuum.io/cshop/anaconda

webkit2png: github.com/adamn/python-webkit2png

How to Shorten a Long URL


```
from urllib2 import Request
def shortenURL(longURL):
  # Build the data needed to call the Goggle API
  url = "https://www.googleapis.com/urlshortener/v1/url"
  query = {"longUrl": longURL, "key": googleAPIKey}
  data = json.dumps(query)
  request = Request(url, data,
    {"Content-Type": "application/json"})

  # Fetch and parse
  result = json.load(urlopen(request))
  shortURL = result["id"]
  print(r"\url{%s}%%" % shortURL)
```

```
Here is a short url \pyc{shortenURL(
"http://mirror.jmu.edu/pub/CTAN/macros/latex/contrib/pythontex/pythontex.pdf")}
```

Here is a short url http://goo.gl/sfT8S5.

Overview
A Question of Primes
Introduction to PythonTEX
Mathematics with Sympy
Plots with matplotlib
Web Services
Conclusions
Extra Examples

Mail Merge

Address: `to field`
Hello `name field`, I just wanted to say hello.

```
to,name
js@example.com,John Smith
mw@example.com,Mike White
tb@example.com,Tom Blue
```

Mail Merge

```
\begin{pythontexcustomcode}{py}
import csv

def mailMerge(filename, texcommand):
  csvFile = open(filename, "r")
  csvReader = csv.DictReader(csvFile)
  for row in csvReader:
    setCommand = r"\def\mail%s{%s}"

    for keyValuePair in row.items():
      print(setCommand % keyValuePair)

    print(r"%s\vfill" % texcommand)
\end{pythontexcustomcode}

\newcommand{\mailBody}{
Address: \mailto\\
Hello \mailname, I just wanted to say hello.
}
```

Mail Merge

```
\pyc{mailMerge("../data.csv", r"\mailBody")}
```

Address: js@example.com
Hello John Smith, I just wanted to say hello.

Address: mw@example.com
Hello Mike White, I just wanted to say hello.

Address: tb@example.com
Hello Tom Blue, I just wanted to say hello.