CSC/MAT-220: Discrete Structures

Lab4: PythonTeX

Thomas R. Cameron

10/13/2017

Introduction. PythonTeX is a TeX package that allows Python code to be run within a document, typesetting the results. This package is currently maintained by Geoffrey Poore and you can visit the github/wiki page by using the following link.

`https://github.com/gpoore/pythontex/wiki`

This lab will introduce you to some of the basic macros available in PythonTeX and provide you with some really neat examples. Please follow along in your own LaTeX file, including each example as you go. To compile this LaTeX file you will need to access your terminal, change directories to the one with your tex file, and run the following command.

pdflatex file.tex && pythontex file.tex && pdflatex file.tex

In addition, you will need the following preambles in your LaTeX file.

```
\usepackage{amsmath}
\usepackage{pythontex}
\usepackage{graphicx}
```

Py and Pyc Macros. The macro `\py{expression}` evaluates a Python expression and typeets its value. As an example, consider the following snippet of code.

```
1 The Binomial Coefficient ``5 choose 2'' is $\binom{5}{2}=\py{5*4/2}$
```

The Binomial Coefficient "5 choose 2" is $\binom{5}{2} = 10.0$

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

```
1 The Binomial Coefficient ``5 choose 2'' is $\binom{5}{2}=\pyc{import
    math; x=math.factorial(5); y=math.factorial(2)*math.factorial(5-2)
    ; print(x/y)}$
```

The Binomial Coefficient "5 choose 2" is $\binom{5}{2} = 10.0$

Note the subtle difference between the macros \py and \pyc. Because the latter only typesets what the Python expression prints, we can do some behind the scenes work and then print only what we want to be typesetted.

Pycode Macro. For more lines of Python code, it is ideal to use the \pycode environment. As with the macro \py, only what is printed gets typesetted. Consider the snippet of code below which generates a table with a loop.

```
\begin{pycode}
import math
print(r"\begin{tabular}{c | c | c | c}")
print(r"$n$ & $n!$ & Stirling's Formula & Relative Error \\ \hline")
for i in range(10,51,10):
    x=math.factorial(i)
    y=math.sqrt(2*math.pi*i)*(i/math.e)**(i)
    z=math.fabs((x-y)/y)
    print(r"%d & %e & %e & %e \\\hline" % (i,x,y,z))
print(r"\end{tabular}")
\end{pycode}
```

n	$n!$	Stirling's Formula	Relative Error
10	3.628800e+06	3.598696e+06	8.365359e-03
20	2.432902e+18	2.422787e+18	4.175011e-03
30	2.652529e+32	2.645171e+32	2.781536e-03
40	8.159153e+47	8.142173e+47	2.085461e-03
50	3.041409e+64	3.036345e+64	1.668034e-03

We can also use pycode to create a function that we make use of outside of the pycode environment!

```
\begin{pycode}
def fib(n):
    if n==0 or n==1:
        return 1
    else:
        return fib(n-1)+fib(n-2)
\end{pycode}
Table of the first 5 Fibonacci numbers:
\begin{tabular}{c | c | c | c | c | c}
$n$ & 0 & 1 & 2 & 3 & 4\\
\hline
$F_{n}$ & \py{fib(0)} & \py{fib(1)} & \py{fib(2)} & \py{fib(3)} & \py{
    fib(4)}\\
\hline
\end{tabular}
```

Table of the first 5 Fibonacci numbers:

n	0	1	2	3	4
F_n	1	1	2	3	5

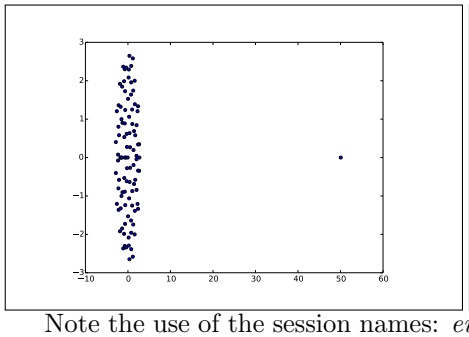
Sessions. Py, Pyc, and Pycode all have an optional session argument, this is why we are able to define a function inside of pycode and use it outside of pycode, as was done in the previous example. If no session argument is given, then the default session is used. Sessions may be used to speedup compile time, since sessions with different names may be executed in parallel.

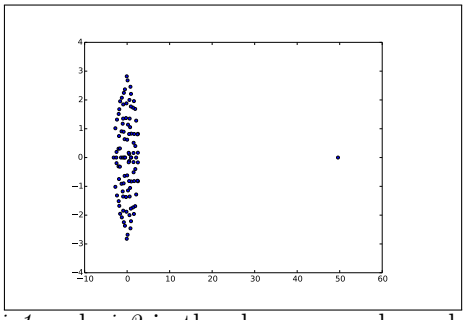
```
\begin{pycode}[eig1]
import matplotlib.pyplot as plt
import numpy as np

np.random.seed()
n=100
A=np.random.rand(n,n)
eig=np.linalg.eigvals(A)
x=eig.real
y=eig.imag
plt.scatter(x,y)
plt.savefig('eig1.pdf')
\end{pycode}
\IfFileExists{eig1.pdf}
{
    \includegraphics[scale=0.25]{
        eig1.pdf}
}
{
    \emph{eig1.pdf not found}
}
```

```
\begin{pycode}[eig2]
import matplotlib.pyplot as plt
import numpy as np

np.random.seed()
n=100
A=np.random.rand(n,n)
eig=np.linalg.eigvals(A)
x=eig.real
y=eig.imag
plt.scatter(x,y)
plt.savefig('eig2.pdf')
\end{pycode}
\IfFileExists{eig2.pdf}
{
    \includegraphics[scale=0.25]{
        eig2.pdf}
}
{
    \emph{eig2.pdf not found}
}
```





Note the use of the session names: *eig1* and *eig2* in the above example, and the use of the macro `\IfFileExists{}`. Multiple sessions are independent, they do not share variables or function definitions. Sometimes, this is exactly what we want, othertimes we would like to have global definitions that are available to all sessions. This is what the `\pythontexcustomcode` environment is for. For our last example, we will create a global function for finding all the prime numbers between two integers *start* and *end*. We will first use this function to list all primes between 1 and 100, and then we will use it to create a table listing the number of primes in a given interval. Do you notice anything interesting about this table?

```
\begin{pythontexcustomcode}{py}
def FindPrimes(start,end):
    x=[]
    if start==1:
        start=start+1
    for i in range(start,end+1):
        check=True
        for j in range(2,i):
            if i%j==0:
                check=False
                break
        if check==True:
            x=x+[i]
    return x
\end{pythontexcustomcode}
List of Primes between 1 and 100:
\begin{center}
\footnotesize{\py{FindPrimes(1,100)}}
\end{center}
Table of the number of Primes between given interval:
\begin{tabular}{c | c}
Interval & Number of Primes\\
\hline
$[1,1000]$ & \py[p1]{len(FindPrimes(1,1000))}\\
\hline
$[1001,2000]$ & \py[p2]{len(FindPrimes(1001,2000))}\\
\hline
$[2001,3000]$ & \py[p3]{len(FindPrimes(2001,3000))}\\
\hline
$[3001,4000]$ & \py[p4]{len(FindPrimes(3001,4000))}\\
\hline
$[4001,5000]$ & \py[p5]{len(FindPrimes(4001,5000))}\\
\hline
$[5001,6000]$ & \py[p6]{len(FindPrimes(5001,6000))}\\
\hline
\end{tabular}
```

List of Primes between 1 and 100:

[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]

Table of the number of Primes between given interval:

Interval	Number of Primes
$[1, 1000]$	168
$[1001, 2000]$	135
$[2001, 3000]$	127
$[3001, 4000]$	120
$[4001, 5000]$	119
$[5001, 6000]$	114