

# Python for Matlab Users

Shelley L. Knuth, Research Computing  
University of Colorado-Boulder

[https://github.com/ResearchComputing/Final\\_Tutorials/python\\_matlab\\_users.pdf](https://github.com/ResearchComputing/Final_Tutorials/python_matlab_users.pdf)

# Outline

- Introduction to Python
- Advantages and disadvantages
- Using Python
- Important Python packages
- Comparison of common programming commands and issues between Python and Matlab
- Purpose of Talk
  - Geared toward researchers who readily use Matlab
  - Some issue is pushing you away from Matlab
  - Little to no knowledge of Python

# Introduction to Python – What is it?

- Python is a programming language created in 1990 by Guido van Rossum
- Named for Monty Python
- Designed to be easy to use, learn, and understand
- Generalized programming language
  - No specific discipline use
- Open source – free!!!
- Cross-platform
- “Glue” language
  - You can call other programming language functions within Python

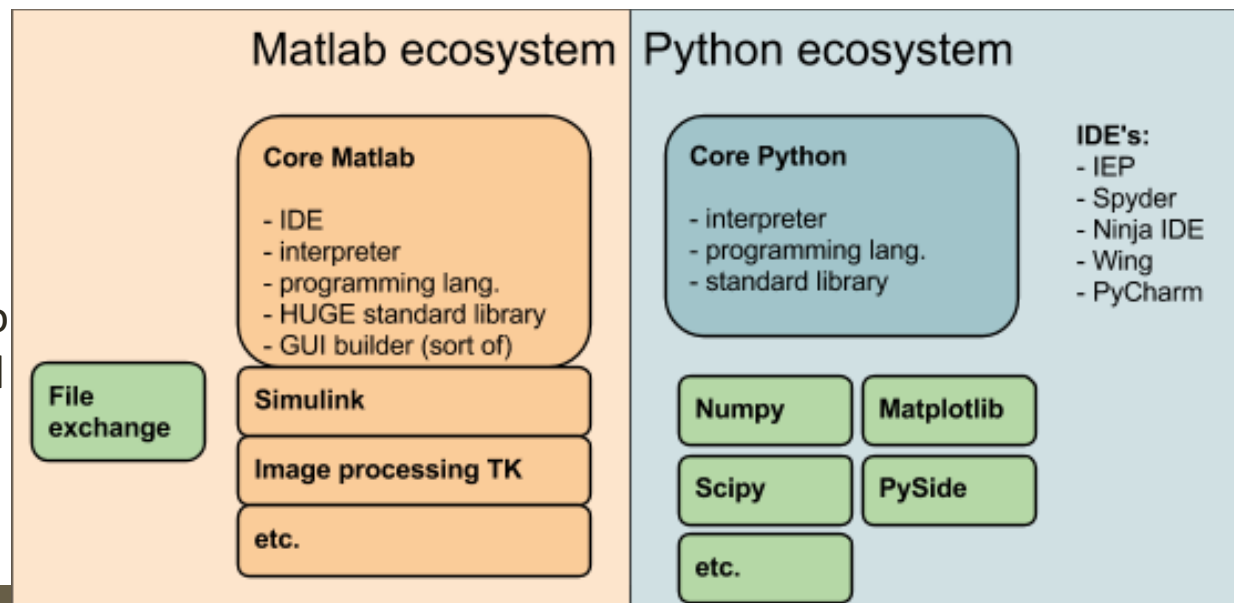
# How do I get Python?

- If you have a Mac or a Linux system, you might already have it
  - Type **python** or **which python** at the command line to see if you do
- If you don't, I recommend installing anaconda  
<http://continuum.io/downloads>
- Anaconda is a package manager that makes it easier to get everything you need for Python
- Once downloaded, double click, follow the prompts

# Introduction to Python

- Python itself is an official programming language
- The general python includes the programming language and interpreter
- Standard library
- Need additional packages to plot, to do scientific computing
- Also if want a user interface will need to pick one
- Matlab comes with these

[http://www.pyzo.org/python\\_vs\\_matlab.html](http://www.pyzo.org/python_vs_matlab.html)



# Advantages and Disadvantages

- Matlab – Advantages
  - Great IDE – Matlab Desktop
  - Can do a lot with plotting
  - Usually you can get access if you are at a university
  - Lots of online support
  - Dynamic language
- Matlab – Disadvantages
  - Expensive!!
  - Licensed, so it's closed source

<http://blogs.lt.vt.edu/safetyinnumbers/2014/04/23/technical-computing-wars-matlab-vs-python/>

# Advantages and Disadvantages

- Python – Advantages
  - Free!!!!
  - Open source
  - Easy to read
  - Powerful language
- Python – Disadvantages
  - Not as nicely packaged
  - Select a IDE
  - Science has been slow to catch up to using python in classes
  - Have to import libraries/packages

<http://blogs.lt.vt.edu/safetyinnumbers/2014/04/23/technical-computing-wars-matlab-vs-python/>

# Python Context

- Python is used by scientists, non-scientists, students, non-students...everyone
- Lots of development in recent years from community
- Python along with it's vast number of libraries are its appeal
- Most recent version: 3.4.2
- Python 3 is not compatible with Python 2
- A lot of code out there is written for Python 2.7



# Python and Matlab

Let's compare the two and see how easy it can be to transition to Python from Matlab

# Python vs. Matlab - General Syntax

	Matlab	Python
Element index	1	0
Comment	%	#
Print variable contents to screen	disp(x)	print(x)
Print string	'Hello Everyone!'	print "hello Everyone!"
Find help on a function	help func	Help(func)
Script file extension	.m	.py
Import library functions	Must be in MATLABPATH	from func import *
Matrix dimensions	size(x)	x.shape
Line continuation	...	\

# Number Types and Math

- In Matlab,  $21/3=7$  and  $23/3=7.667$
- In Python,  $21/3=7$  and  $23/3=7$
- In Python you must specify the type of number or it will simply output an integer
- To get the correct answer you should type:
  - $21.0/3.0$ , which  $=7.0$ , and  $23.0/3.$ , which  $=7.6666$

# Syntax in python

In Python, there are no brackets or semicolons. Instead, each command is interpreted as its own “block” by indentation:

```
1  var1=10
2  var2=20
3
4  if var2 > var1:
5      print(var2, "is greater than", var1)
6
7
```

# If/else statements

- If/else statements, unlike in Matlab, don't end
- The end is where the indentation, or block, ends

```
if (5 > 4):  
    print("The world is still sane")  
elif (5=5):  
    print("Even now it's still sane")  
else:  
    print("You have entered another dimension")
```

# Loops

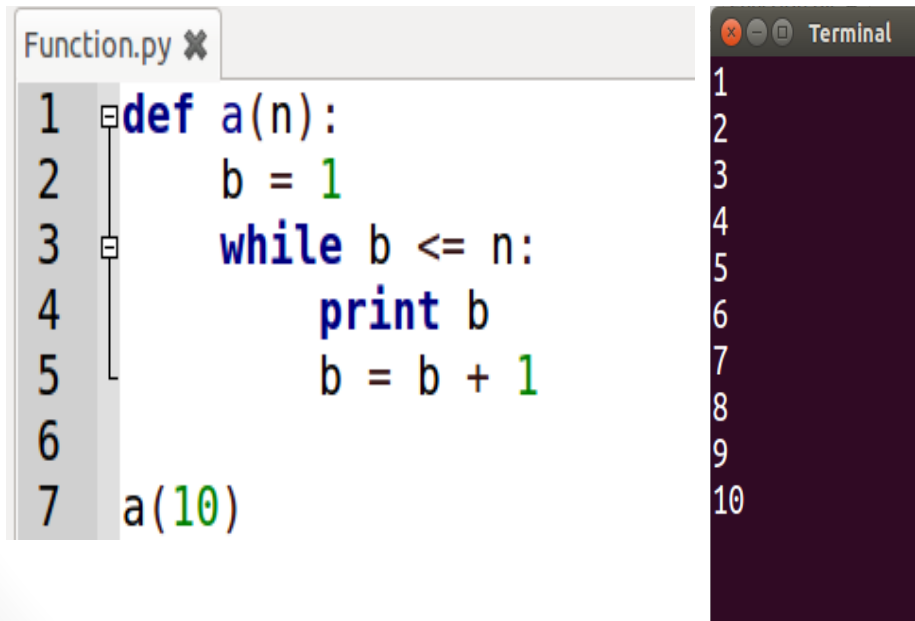
- The same is true for for and while loops:

```
2 factorial = 1
3 for j in range(10):
4     |
5     | factorial = factorial *(j+1)
6     | print(factorial)
```

```
1 var1=10
2 var2=20
3
4 while var1 < var2:
5     | print(var1+var2)
6     | var1= var1+1
7
```

# Functions

- Functions are defined using “def”
- Otherwise similar to Matlab



The image shows a code editor window titled 'Function.py' and a terminal window. The code editor contains the following Python code:

```
1 def a(n):  
2     b = 1  
3     while b <= n:  
4         print b  
5         b = b + 1  
6  
7 a(10)
```

The terminal window shows the output of the function, which is the numbers 1 through 10, each on a new line.

Suppose you wanted to list numbers from 1 to 10 but another time, you want to list them from 1 to 20.

Instead of writing the same code twice, write a function that you can call.

In this example it's listing all numbers from 1 to 10. To list all numbers from 1 to 20, you would just call: `a(20)`

# Writing a script in Python

- Open a text window
- Type commands
- Save file to <filename>.py
- Let's try it!!
- In a text window, type the following:

```
#Our test program  
print "I like test programs"  
print "They are fun"  
print "This is great!"
```

Then save file to test.py



# Running a Script in Python

- First, you need to select your IDE or whether you'll run off the command line
- Options:
  - Type **python** at the command line
  - Type **ipython** at the command line
  - Type **IDLE** at the command line
  - Type **spyder** at the command line
    - This looks a bit like the Matlab Desktop
- I use ipython typically

# To Run the script...

- Depending on what you're using for your IDE it might be different for running the script
- For python or ipython type **import test** and your program should run

# Important Python Packages

- Python is nothing without its libraries
- Many of them created and modified by the community
- Here are some additional python packages you will need to get to do any kind of scientific computing
- These packages allow you to do nearly everything Matlab:
  - Numpy – Matlab core
  - SciPy – Matlab Toolboxes
  - Matplotlib – graphing
  - Ipython – like the desktop environment

# NumPy and SciPy

- Matlab is extremely useful in manipulating matrices
- Python itself cannot do that very well; very bare bones
- However, the libraries numPy and sciPy were written to make scientific computing easy
  - Provide common mathematical and numerical routines as part of functions within the libraries
  - Makes Python function similar to Matlab
- NumPy: provides basic routines for manipulating large arrays and matrices
- SciPy: extends NumPy's functionality with Fourier transformation, regression, etc
- Should install both

# Using NumPy and SciPy

- How do I get it?
  - If you've installed Python using Anaconda it comes with it
- How do I use it?
  - When you start up Python, you are using basic Python and whatever libraries you have imported
  - To import these libraries, at the top of your script, or on the command line, type:

```
import numpy  
import scipy
```

(continued on next slide!!)

# Using NumPy and SciPy

- If you are using a large number of calls, however, it's better to import the library under some shorter name so that you can access numPy and SciPy objects
- Instead, type the following:

```
import numpy as np  
import scipy as sp
```

Then you would use it in ways such as:

```
np.array([1., 2., 3., 4.])
```

# Important Features of Numpy

- Array function

```
a=np.array([1,2,3],float)
b=np.array([5,2,6],float)
```

Output:

```
In [16]: a+b
```

```
Out[16]: array([ 6.,  4.,  9.])
```

```
In [17]: a-b
```

```
Out[17]: array([-4.,  0., -3.])
```

```
In [18]: a*b
```

```
Out[18]: array([ 5.,  4., 18.])
```

<http://www.engr.ucsb.edu/~shell/che210d/numpy.pdf>

# Important Features of Numpy

- Other important functions:

**Sum, prod** – add, multiply all items in an array

**mean, std** – average/std. dev of all items in an array

**min, max** – minimum and maximum values in array

**floor, ceil** – lower and upper integer

**pi** – 3.1415926...

**e** – 2.71828...

**sort** – sort array



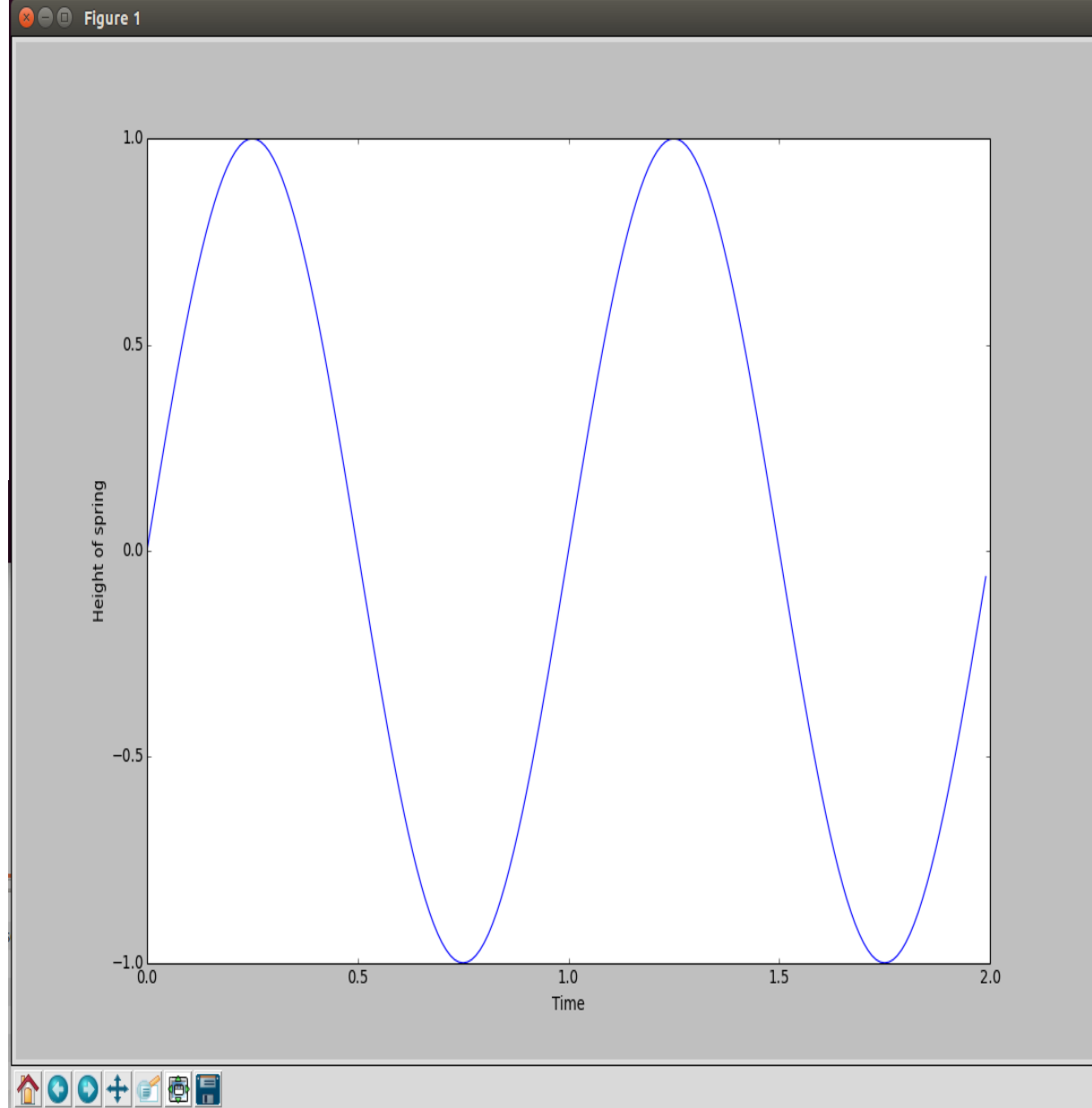
# Matplotlib.pyplot

- Matplotlib is a library of functions that makes python look like you were plotting points in MATLAB.
- The following example will be using matplotlib

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def f(t):
5     return np.sin(2*np.pi*t)
6
7 t = np.arange(0.0, 2.0, 0.01)
8 plt.ylabel("Height of spring")
9 plt.xlabel("Time")
10 plt.plot(t, f(t))
11 plt.show()

```



Stepping through line by line:

1. Library for math functions
2. Library for graphing
4. Create function  $f(t)$  which creates a  
sin wave
7. Make the x values go from 0 to 2,  
stepping by .01
8. Label the y-axis
9. Label the x-axis
10. Plot the graph of  $t$  and  $f(t)$
11. Show graph on screen

# Exercise

Your turn!

To get familiar with Python, do the following:

1. Create a variable of random numbers with 10 rows and 2 columns and call the variable “test”
2. Label the x-axis “Awesome X-axis”
3. Label the y-axis “Awesome Y-axis”
4. Plot a line graph of the data

# Solutions

1. Create a variable of random numbers with 10 rows and 2 columns and call the variable “test”

```
import numpy as np  
test=np.random.rand(10,2)
```

2. Label the x-axis “Awesome X-axis”

```
import matplotlib.pyplot as plt  
plt.xlabel("Awesome X-axis")
```

3. Label the y-axis “Awesome Y-axis”

```
plt.ylabel("Awesome Y-axis")
```

4. Plot a line graph of the data

```
plt.plot(test[0,:],test[1,:])  
plt.show()
```

# Thanks for Attending!

- Useful documentation: [docs.python.org](https://docs.python.org)
- Slides available at [https://github.com/ResearchComputing/Final\\_Tutorials/python\\_matlab\\_users.pdf](https://github.com/ResearchComputing/Final_Tutorials/python_matlab_users.pdf)
- Email: [rc-help@colorado.edu](mailto:rc-help@colorado.edu)
- [Shelley.knuth@colorado.edu](mailto:Shelley.knuth@colorado.edu)
- Twitter: @shelley\_knuth

# References

- <https://wiki.python.org/moin/BeginnersGuide>
- <http://www.stat.washington.edu/~hoytak/blog/whypython.html>
- <http://www.sthurlow.com/python/>
- <http://www.engr.ucsb.edu/~shell/che210d/numpy.pdf>
- [www.matplotlib.org](http://www.matplotlib.org)
- [www.python.org](http://www.python.org)