

Introduction to Python (Or E7 in a Day ...plus how to install)

Daniel Driver

E177-Advanced MATLAB

March 31, 2015

Overview

- 1 What is “Python”
- 2 Getting Started
- 3 “Matrix Lab”
- 4 How to Get it All
- 5 Intricacies

What is "Python"

What is "Python"

What is Python

- Interpreted Language
 - Code Read at Run Time
 - Dynamic Variables like MATLAB
 - `A='I am a string'`
 - `A=3`
- Open Source
- Two Major Versions
 - python 2.x -Older
 - python 3.x -Newer
 - Largely the Same for in Day to Day use
 - Will discuss differences later



Figure : Python Logo

<https://www.python.org/downloads/>

CPython-The “Python” You Think of

- CPython is Python implemented in C
- Generally when people say “Python” This is probably what they are referring to
- This is what we will be referring in this presentation.
- Python code interpreted at Run time
 - Turns into byte code
 - Byte code on Python Virtual Machine
- Not to be confused with Cython (which is a python library that compiles python into C code)

<http://www.thepython tree.in/python-vs-cython-vs-jython-vs-ironpython/>

Other Implementations of Python

- Python interface also implemented in
 - Java: Jython-bytecode runs on Java Virtual Machine(JVM)
 - only 2.5 so far
 - .NET: IronPython -Implemented in C# -byte code runs on Common Language Runtime (CLR)
 - Only 2.7 so far
 - PYPY- Just-In-Time Compiler produces Machine Code for most frequently execute code
 - Only 2.7 so far

<http://www.thepythontree.in/python-vs-cython-vs-jython-vs-ironpython/>

Getting Started

Getting Started

How to install Python

- Autodetect you computer
 - Correct install files
- With Linux or Mac
 - Can Use Package Manager
 - Apt/Yum or MacPorts

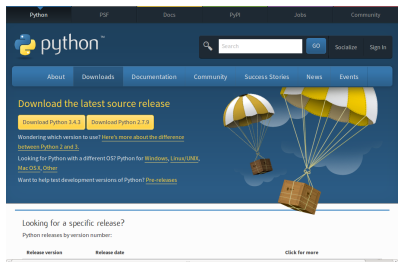
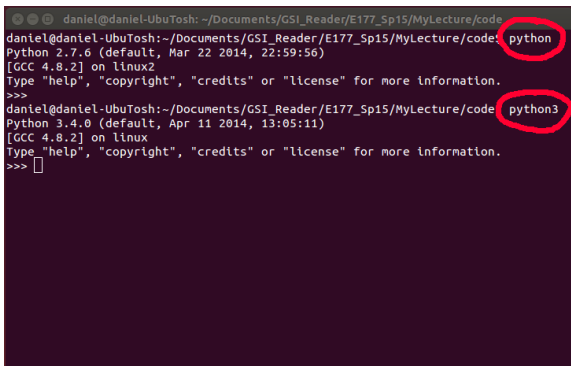


Figure : <https://www.python.org/downloads/>

- Well Maybe wait ... or skip ahead just go to <http://continuum.io/downloads> and get Anaconda ...**More on that later!**

Python Interpreter in Shell or Command Line

- -Start Terminal or Command Prompt(Windows)
- -To Start the Python Interpreter
 - python2: Run "python"
 - python3: Run "python3"
- exit by running "exit()" or pressing Ctrl-d



```
daniel@daniel-UbuTosh: ~/Documents/GSI_Reader/E177_Sp15/MyLecture/code
daniel@daniel-UbuTosh:~/Documents/GSI_Reader/E177_Sp15/MyLecture/code: python
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
daniel@daniel-UbuTosh:~/Documents/GSI_Reader/E177_Sp15/MyLecture/code: python3
Python 3.4.0 (default, Apr 11 2014, 13:05:11)
[GCC 4.8.2] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

The image shows a terminal window with a dark background. The prompt is 'daniel@daniel-UbuTosh: ~/Documents/GSI_Reader/E177_Sp15/MyLecture/code'. The first command entered is 'python', which is circled in red. It outputs 'Python 2.7.6 (default, Mar 22 2014, 22:59:56) [GCC 4.8.2] on linux2' and a help message. The second command entered is 'python3', also circled in red. It outputs 'Python 3.4.0 (default, Apr 11 2014, 13:05:11) [GCC 4.8.2] on linux' and a help message. The prompt is shown again at the end.

Whats Can I do Now?

- The Python interpreter Only Loads very limited functionality

Basic Data Types

Can `+`, `-`, `*`, `/`, `//`, `**` numbers
`**` is power, `//` is floor divide
python 2 `"/"` floor divide for integers

- int

- `>>> a=3`

- float

- `>>> a=3.0`

- `long(long integer)` [only in 2.x]

- `>>>a=long(3)`

- complex

- `>>> a=complex(3.0,1.0)`

- str

- `>>> a='Hello Class'`

Logic and If Statement

- Logic (`<`, `>` , and, not, or, `==`)

- `>>> 3>2`

- True

- `>>> 3==2`

- False

- `>>> 3==3`

- True

- `>>>not(3==3)`

- False

- If-Statement

- `>>> if 3>2:`

- ... `print('Hello Class')`
 - `'Hello Class'`

For loops and Data Structures

- The Python interpreter Only Loads very limited functionality

Pythonic For Loop

- range makes list of integers
 - `>>> range(5)`
`[0,1,2,3,4]`
- For Loop
 - `>>> for k in range(3):`
 `... print(k)`
0
1
2
 - `>>> a=['a',1,2.0,'b']`
 - `>>> for k in a:`
 `... print(k)`
a
1
2.0
b

Data Structures

- list (mutable, "Elements have same meaning")
 - `>>> a=[1.0,2,complex(1,2)]`
`>>> print(a[0])`
1.0
- tuple (immutable, "order has meaning")
 - `>>> a=(1,2.0,Long(3))`
- dict
 - `>>> a={'key1':1.0,'key2':'HelloClass'}`
 - `>>> print(a)`
{'key2': 'Hello Class', 'key1': 1.0}
 - `>>> print(a['key1'])`
'Hello Class'
 - `>>> print(a['key2'])`
1.0

Mutable Vs. Immutable (Function Example)

immutable

```
x = something # immutable type  
print(x)  
func(x)  
print(x) # prints the same thing
```

mutable

```
x = something # mutable type  
print(x)  
func(x)  
print(x) # might print something different
```

Mutable Vs. Immutable (Assignment Example)

immutable

```
x = something # immutable type
y = x
print(x)
# some statement that operates on y
print(x) # prints the same thing
```

mutable

```
x = something # mutable type
y = x
# some statement that operates on y
print(x)
# might print something different
```

Functions and Scripts

Function

- Lambda(Like MATLAB's Inline Func)

```
>>>a=lambda x:x**2
>>>a(3)
9
```

- Regular Function

```
>>> def TestFunction(x):
...     return x**2,3*x
>>>a,b=TestFunction(4)
>>>print(a)
16
>>>print(b)
12
```

"open" also available to open files. Returns object with read method and close method.
help(something) displays help for

Scripts

- Any command in *.py file
 - Just text file with Commands
 - Run as if typed by hand
- Run by execfile
 - >>> execfile("Test.py")
- Remember
 - Indentation Matter
 - Determines Block
 - Aka what is in an "If" or For loop or a function
 - 0 indexed
 - index with [] brackets
- Can Run in command line with
python Test.py

Thats it?

Thats about it...except for one more more thing...

import

import-Making Python Useful

Import causes Python to load a package!!

How to Use

- Load `PackageName.py`
 - `>>>import PackageName`
`outputs=PackageName.Func(inputs)`
- import Package with alias
 - `>>>import PackageName as PN`
`outputs=PN.Func(inputs)`
- import One Thing from Package
 - `>>>from PackageName import Func`
`outputs=Func(inputs)`
- Load Everything in Package
 - `>>>from PackageName import *`
`outputs=Func(inputs)`

Standard Library

Packages that come With Python but aren't Loaded automatically.

Full List at <https://docs.python.org/2/library/>

A Few Useful ones

- math-sqrt,cos,sin,exp,factorial
- cmath - math for complex math
- pickle - Like MATLAB Save
- profile - Profiles code
- bdb/pdb - debugging tools
- json and xml tools
- csv - read csv files
- os - operating system interfaces
- zipfile/tarfile- compression tools
- random- random number tools

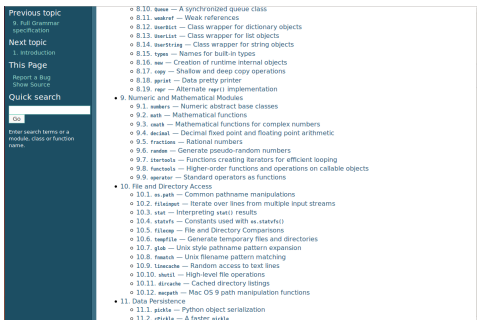


Figure : Website with Standard Library

"Matrix Lab"

"Matrix Lab"

External Packages for Scientific Computing

- Numpy - "Numerical Python" -arrays, Linear algebra on dense matrices
 - solve system of equations
 - invert matrix
 - Most Packages accept Numpy Arrays as inputs
 - http://wiki.scipy.org/NumPy_for_Matlab_Users
- Matplotlib - plotting library that was designed to be similar to matlab
<http://matplotlib.org/gallery.html>
- Scipy -fft, sparse arrays, sparse linear algebra, load .mat files
 - <https://docs.scipy.org/doc/scipy/reference/>
 - I use google a lot thought to find functions

Numpy Example-Solving System

Solve System of equations

```
import numpy as np
#make matrix A
#[1,2]
#[0,1]
A=np.array([[1,2],[0,1]],dtype=float)

#make right hand side b
#[1]
#[2]
b=np.array([[1],[2]],dtype=float)

#solve the system of Equations
x=np.linalg.solve(A,b)
print('x=')
print(x)
```

Result

```
x=
[[ -3.]
 [ 2.]]
```

Numpy Example-Least Squares

Solve System of equations

```
import numpy as np
#Do linear Least Squares
xvals=np.linspace(1.1,10.2,4)
yvals=np.array([1.1,2.5,4.4,7.9])
#make array
A=np.ones((4,2),dtype=float)
#put x values in first column
A[:,0]=xvals

#solve for coefficients
coeffs,residual,rank,singularvals=\
np.linalg.lstsq(A,yvals) #use lstsq function
print('Coeffs from lstsq ')
print("y={0}x+{1}".format(*coeffs))

#Normal equations for least squares
NMat=A.T.dot(A) #Transpose A * A
bMat=A.T.dot(yvals) #Transpose A * yvals
coeffs=np.linalg.solve(NMat,bMat)
print('Coeffs from Normal Equations ')
print("y={0}x+{1}".format(*coeffs))
```

Result

```
Coeffs from lstsq
y=0.735164835165x+-0.178681318681
Coeffs from Normal Equations
y=0.735164835165x+-0.178681318681
```

- \ is the line continuation symbol in python
- # is the comment
- * in this case unpacks coefficients
 - As if I typed `format(coeffs[0],coeffs[1])`

Matplotlib Example-Uses least squares example from above

Solve System of equations

```
import matplotlib.pyplot as mplot
#plot
mplot.plot(xvals, yvals, 'x')
mplot.plot(xvals, coeffs[0]*xvals+coeffs[1])
mplot.xlabel('y')
mplot.ylabel('x')
mplot.legend(['measured', 'Fit'], loc=0)
mplot.show()
```

Best way to learn Matplotlib -

<http://matplotlib.org/gallery.html> and look at more examples

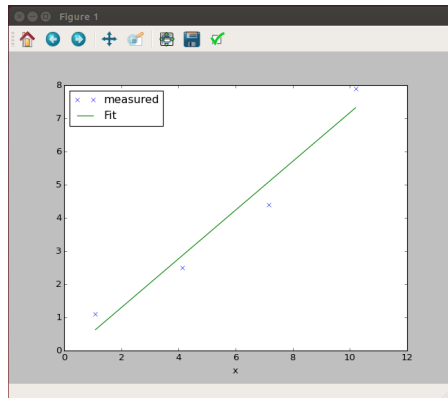


Figure : Result

How to Get it All

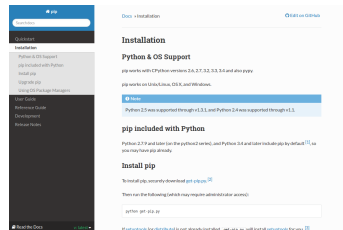
How to Get it All

Packages with PyPi and Pip

- Go to <https://pip.pypa.io/en/latest/installing.html> for pip install instructions

Pip=Pip installs package

- Package manager for PyPi
- Search Package Repository
 - "pip search < *PackageName* >"
- Install Package from PyPi
 - "pip install < *PackageName* >"
- Upgrade Package
 - "pip install < *PackageName* > --upgrade"



https://pip.pypa.io/en/latest/reference/pip_install.html

Anaconda

- One Stop Shop
- Installs Python
- 195+ of the Most Used Packages
 - Numpy
 - Scipy
 - Matplotlib
 - pip
- Also Ipython, Spyder and IDLE
- <http://continuum.io/downloads>

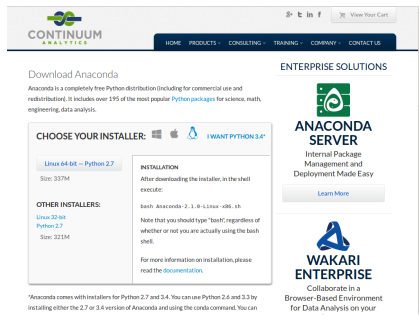


Figure : Anaconda Download

Python Shell with IDLE

- Provides Window for Python interpreter
- Can be Useful in Windows
- Python 2 and 3 install their own versions

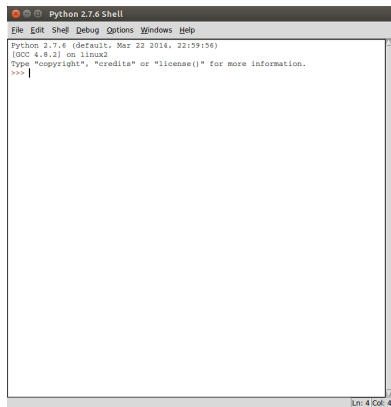
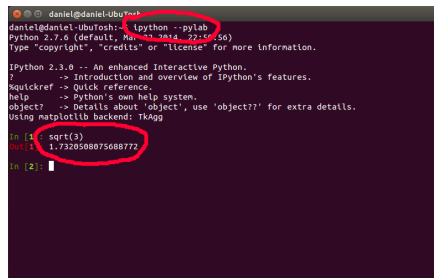


Figure : IDLE Python Shell

Ipython-Interactive Python

- More User Friendly Interpreter
- Auto-Complete
- Command History Across Sessions
- Starting with pylab flag
"ipython - -pylab" loads more Packages automatically to make more "MATLAB Like" including
 - math
 - numpy
 - matplotlib.pyplot

A terminal window with a dark purple background. The prompt is 'daniel@daniel-UbuTosh:~'. The command 'ipython - -pylab' is entered and executed. The output shows 'Python 2.7.6 (default, Mar 22 2014, 22:04:56)' and 'Type "copyright", "credits" or "license" for more information.' followed by 'IPython 2.3.0 -- An enhanced Interactive Python.' and a list of help options. The command 'sqrt(3)' is entered at the prompt 'In [1]:', and the output 'Out[1]: 1.7320508075688772' is displayed. The prompt 'In [2]:' is shown at the bottom.

```
daniel@daniel-UbuTosh:~$ ipython - -pylab
Python 2.7.6 (default, Mar 22 2014, 22:04:56)
Type "copyright", "credits" or "license" for more information.

IPython 2.3.0 -- An enhanced Interactive Python.
?                -> Introduction and overview of IPython's features.
%quickref        -> Quick reference.
help             -> Python's own help system.
object?         -> Details about 'object', use 'object??' for extra details.
Using matplotlib backend: TkAgg

In [1]: sqrt(3)
Out[1]: 1.7320508075688772

In [2]:
```

Figure : Ipython with pylab flag

Spyder - A really nice Python IDE

- Editor, Variable Inspector, Console, Integrated Debugging Tools, Integrated Profiler

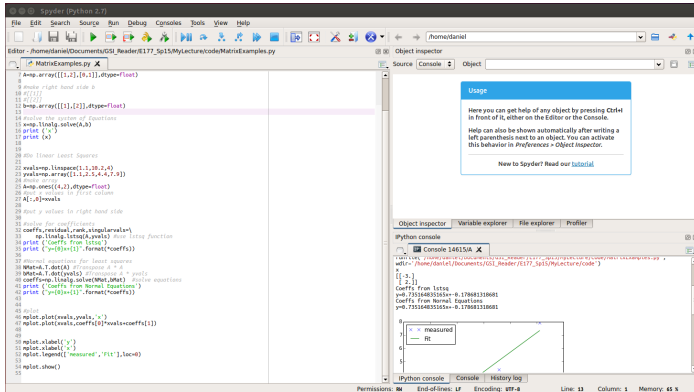


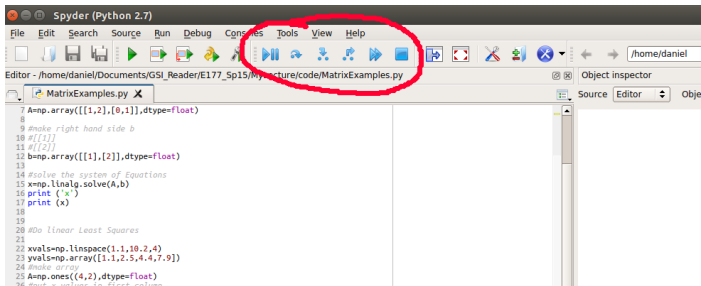
Figure : Spyder IDE

Spyder - Debugging

- F12 to set breakpoint

```
44  
45 #plot  
46 mplot.plot(xvals,yvals,'x')  
47 mplot.plot(xvals,coeffs[0]*xvals+coeffs[1])  
48
```

- Blue buttons at top to step through



Spyder - Profiler

- F10
- Or start from Run menu

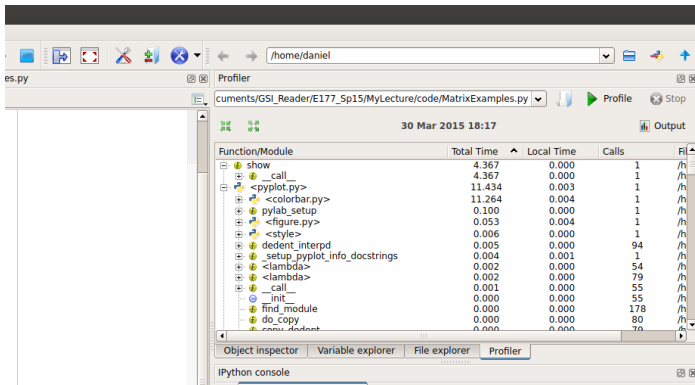


Figure : Spyder IDE-Profiler

Spyder - Python Consoles

- Integrated Ipython and Python Consoles
- Can Click Errors in Console and takes you too Line in Code

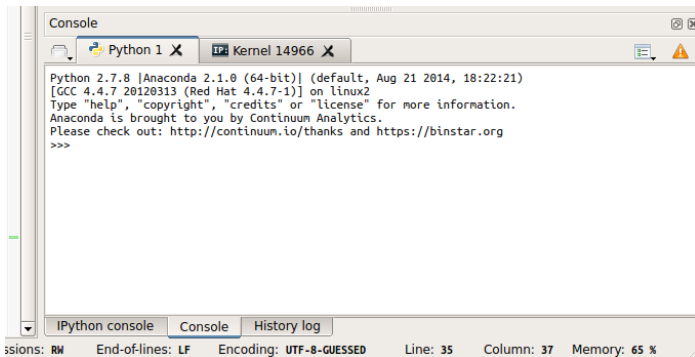


Figure : Spyder IDE-Consoles

Intricacies

Intricacies

Python 2.x

- 2.x is older more widely adopted interface
- Latest Version is 2.7.9
 - $3/2=1$ "Floor Division"
 - `print 'Hello'`

<http://www.toptal.com/python/why-are-there-so-many-pythons>

Python 3.x

- 3.4.3 latest of new interface
 - Controversial - Ignored Backward compatibility to "do it right"
 - Unicode support
 - Differences in integer division $3/2=1.5$ but floor division given by $3//2=1$
 - `print('Hello')`
 - Some cool and useful features
 - More consistent in behavior and syntax
 - no fully supported by all packages and implementations (Yet)
 - Most major packages are available

<http://www.toptal.com/python/why-are-there-so-many-pythons>

Other Implementations of Python (Revisited)

- Python interface also implemented in
 - Java: Jython-bytecode runs on Java Virtual Machine(JVM)
 - only 2.5 so far
 - .NET: IronPython -Implemented in C# -byte code runs on Common Language Runtime (CLR)
 - Only 2.7 so far
 - PYPY- Just-In-Time Compiler produces Machine Code for most frequently execute code
 - Only 2.7 so far
- **Package availability can be limited**
 - **Out of Date**
 - **Or just don't exist**

<http://www.thepythonic.com/python-vs-cython-vs-jython-vs-ironpython/>

Python Vs MATLAB

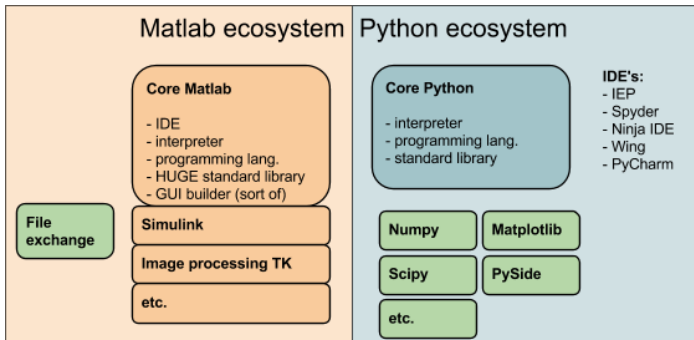


Figure : http://www.pyzo.org/python_vs_matlab.html

MATLAB Pros

- MATLAB is the product of a single Company
 - Results is a potentially easier user experience
- MATLAB aka “Matrix Lab” was Setup for Scientific computing from the start
 - Python was Setup to be more general programming environment.
 - Generality in Python can make the “ecosystem” a little more complicated
- MATLAB is Mostly Backward compatible
 - Python can sort of “Change underneath you”
 - Less worrisome with major projects (better managed)
- “\” is pretty darn tootin awesome
 - Automatically looks for best solver
 - Some of the best algorithms in the world go into it
 - Python has linear algebra but takes more work
- Simulink is also a pretty good library

MATLAB Cons/Python Pros

- Expensive
 - Python is Free
- MATLAB is proprietary- No way to see how functions work
 - Python and most packages are open source
 - With Python people can look at and extend algorithms
- MATLAB is expressive for math but can be very slow
 - Pure Python is equally slow or slower
 - Extensions like Numpy make it fast
 - Python is more like a programming language
 - Forces you to think about what the code is doing
 - This leads to potentially better coding practices
- MATLAB Started as Fortran Wrappers
 - Sort of hodge podge together over time
 - Python was designed for beauty

More Python Pros

- Can put multiple Functions in a File
 - Function definitions can be put in scripts
 - Easier to make packages
- Easy FileIO and really powerful string parsing and modifying functions
- Python OOP is said to be more flexible (You can judge next time)
- Use of [] for indexing and () for function calls
- Indexes start at 0 which can be much more natural
 - Easier when interfacing with C
- Its good to be part of the cult
 - Lots of people writing code for you to use (Free!)
 - Lots of support (Yay stackoverflow)

Why is it so Trendy?

- Because it is free I know anyone can run my code
 - Labmates
 - Collaborators
 - Genius 7 year old in Grandma's basement
- Python tends to have newest innovations in programming
 - For example early adopter of dictionaries
 - Packages and Namespace before MATLAB
 - Anyone with a good idea can add it
 - Not possible in MATLAB because closed source
 - Cython packages adds key words to speed up python
- Because it is so open lots of packages for interfacing available
 - Can easily "talk" between environments

Why is it so Trendy?

- Most major scientific programs now have a python interface
 - FENICS (Finite element package)
 - Trilinos (Parallel Computing Library)-PyTrilinos
 - Petsc (Parallel/sparse Matrix Library)-Petsc4y
 - MPI (Message Passing Interface for Parallel computing)-mpi4py
 - OpenCV -Computer Vision
- Lots of Non-scientific Packages too
 - BeautifulSoup-reading and parsing webpages
 - Django - for building webpages
 - MySQLdb - Access and Query MySQL databases
 - SQLAlchemy - Another SQL database access library
 - Py2Neo - Access and Query Neo4j databases
 - Sympy - Symbolic Algebra like Mathematica in Python
 - Blender - 3D graphics and animation software
 - PyGame - 2D Game Library