



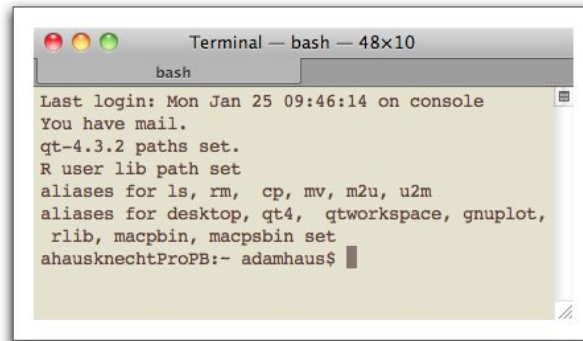




This Worksheet shows you the easiest way to start using Python!

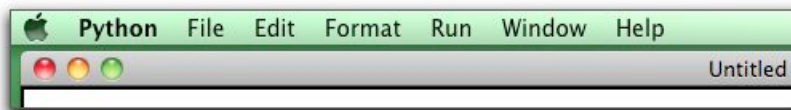
Start the **Terminal**  application by

-  Selecting the **Utilities** item from the **Go** menu located at the top of the screen (it may be in the Dock).
-  A the Utilities folder will be displayed in a window.
-  Double-clicking the **Terminal** application's **icon** located in the **Utilities** subfolder of the Applications folder.
-  A the **Terminal** application will start and its window will open



Start Python's basic integrated development environment (IDE) application called **idle** by

-  Typing **idle &** and the pressing the return key.
-  After a few minutes, idle will start up and open one or two windows and the menu bar



The menu's are used to control **idle**. Also, a message of the form “[k] process ID”

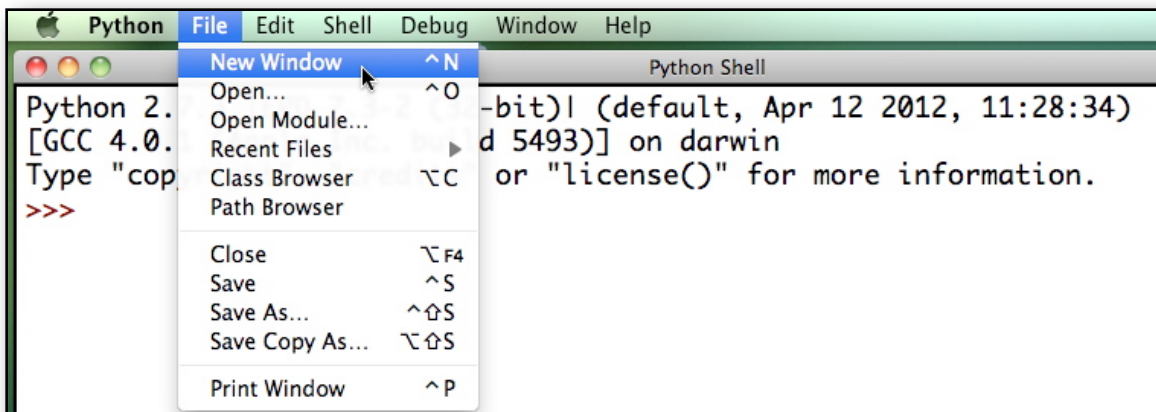
[1] 786

will be displayed in the Terminal's window. The integer **786** identifies the processing thread running **idle**.

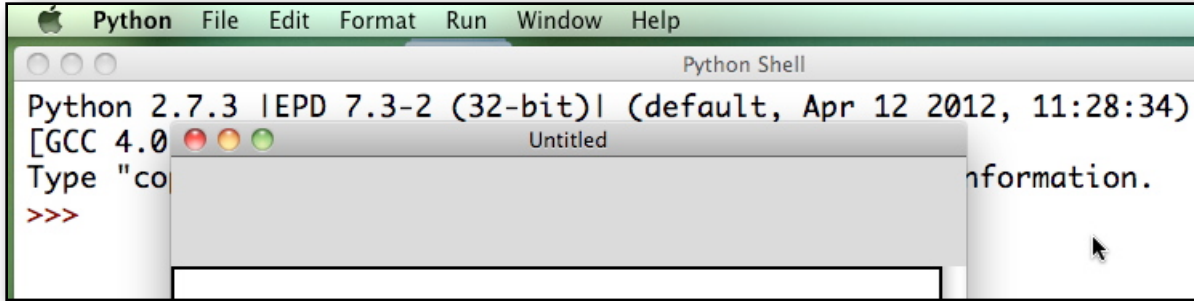
Note: The ampersand ‘&’ tells the operating system to run idle as a separate process.

In order to execute Python code, the code must be saved in a file. To do this do,

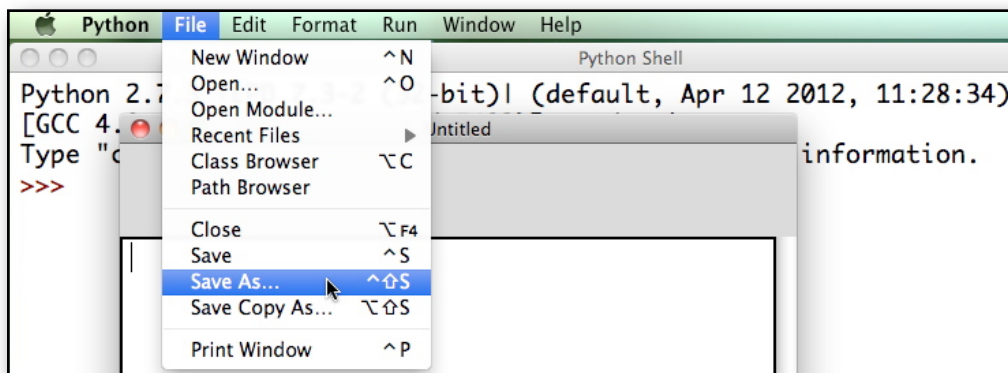
-  Select **New Window** from **idle's File** menu.



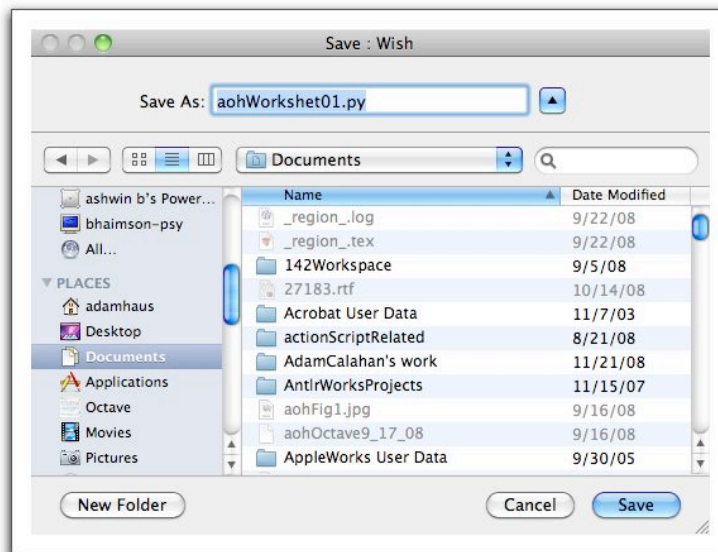
☞ A new **idle** program window will open with the title **Untitled**.



☞ Select the **Save As...** from **idle**'s the **File** menu.



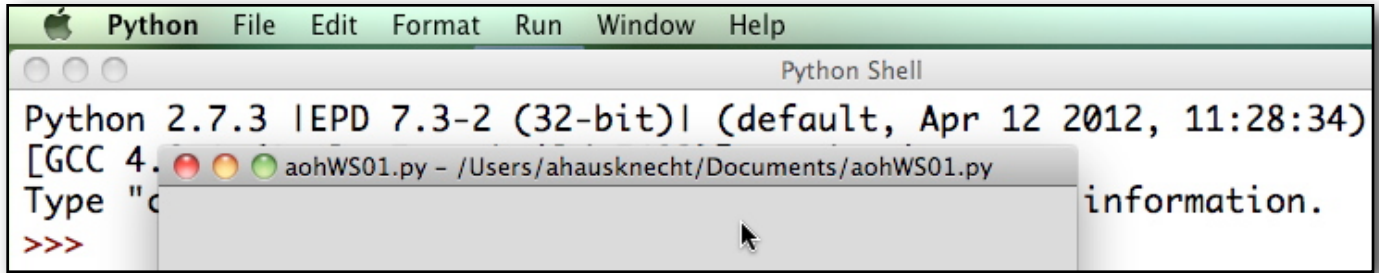
☞ A file dialog window will open.



☞ Navigate to the student's **Documents** folder. Enter a name a of the form
yourNameWS01.py

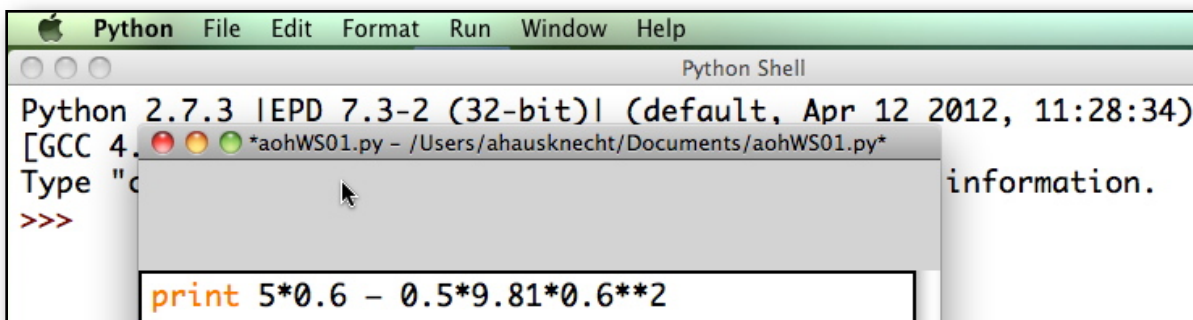
for the file and *press* the save button. Note the file's name must end with the suffix “.py”.

☞ The file dialog window will close and the **idle** program window's title will change to the file's path.

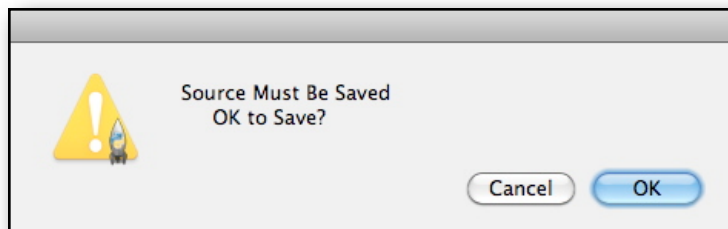
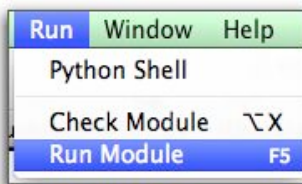


To execute a Python statement to the following:

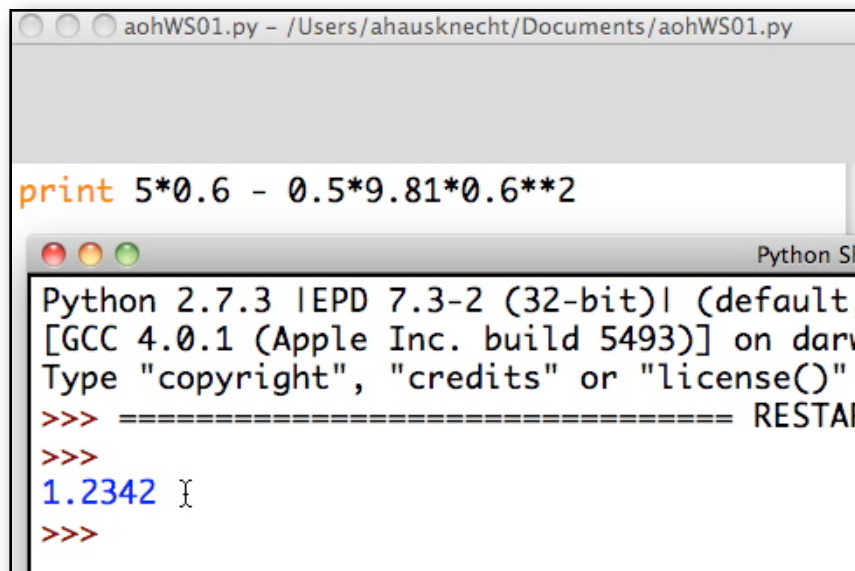
Enter the following Python statement into the **idle** program window: **print 5*6.0 - 0.5*9.81*0.6**2 .**



Then select the **Run Module** item from **idle**'s **Run** menu and click the **OK** button in the **Source Must Be Saved** dialog (you won't see this dialog if your program has already been saved).



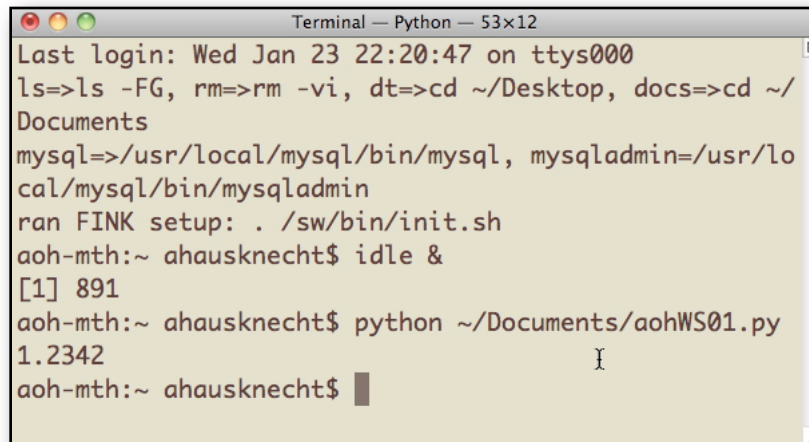
The **idle**'s Python Shell window will come to the front and **1.2342** will be displayed near the window's bottom.



To run your one-line Python program saved in the file **yourNameWS01.py** and located in the student's Documents folder, do the following

✋ Click in the Terminal window to make it active. Enter the statement
`python ~/Documents/yourNameWS01.py`

☞ Python will execute your program and 1.2342 will be displayed in the Terminal window.



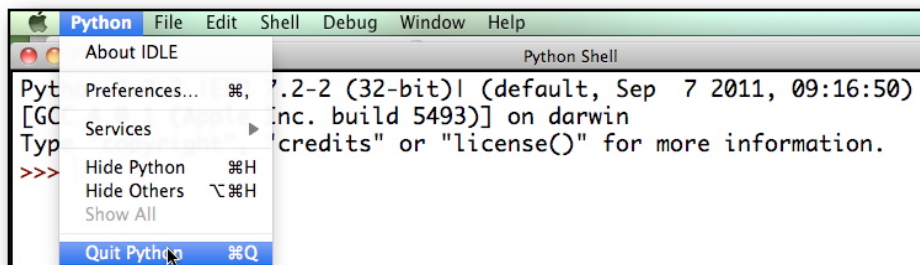
```
Terminal — Python — 53x12
Last login: Wed Jan 23 22:20:47 on ttys000
ls=>ls -FG, rm=>rm -vi, dt=>cd ~/Desktop, docs=>cd ~/Documents
mysql=>/usr/local/mysql/bin/mysql, mysqladmin=/usr/local/mysql/bin/mysqladmin
ran FINK setup: . /sw/bin/init.sh
aoh-mth:~ ahausknecht$ idle &
[1] 891
aoh-mth:~ ahausknecht$ python ~/Documents/aohWS01.py
1.2342
aoh-mth:~ ahausknecht$
```

A another way to write Python programs is to use **ipython** which is part of the Enthought's Python distribution.

To do this, first quit the **idle** application by

✋ Click on the **idle** window (to bring it to the front) and select **Quit** from the **Python** menu.

☞ **Idle** will terminate.



To start **ipython's notebook** by

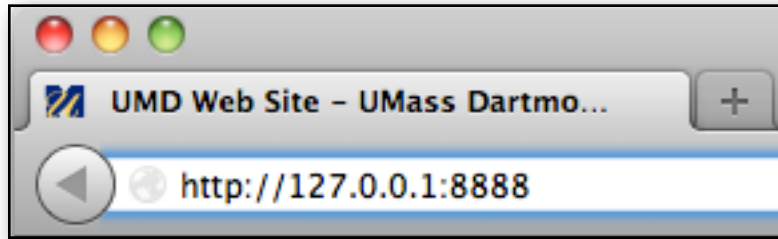
✋ Clicking on the **Terminal** window (to bring it to the front) and entering

```
ipython notebook --pylab=inline --no-browser
```

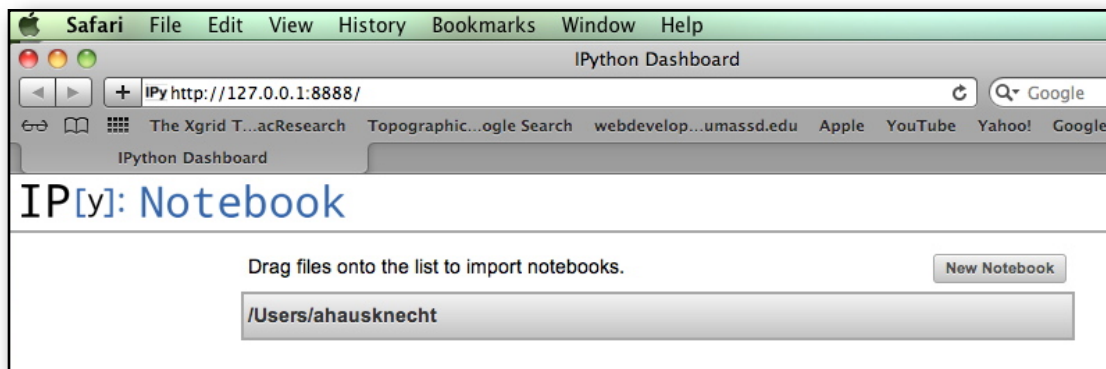
☞ The Terminal window will display messages similar to the last three lines shown below.

```
aoh-mth:~ ahausknecht$ ipython notebook --pylab=inline --no-browser
[NotebookApp] Using existing profile dir: u'/Users/ahausknecht/.ipython/profile_default'
[NotebookApp] The IPython Notebook is running at: http://127.0.0.1:8888
[NotebookApp] Use Control-C to stop this server and shut down all kernels.
```

👉 Start up **Firefox** and enter the url <http://127.0.0.1:8888> shown in the figure above into **Firefox**.



👉 **iPython's Dashboard** will open in the computer's default browser.

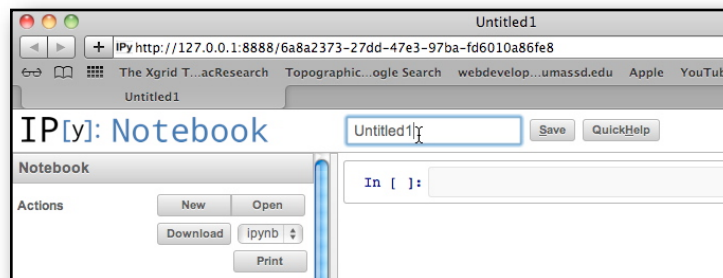


Note that the extra parameter `--pylab` causes **ipython** to automatically load important packages including *numpy* (large matrices, linear algebra) and *matplotlib* (plotting) needed for scientific computation. The inline parameter tells **ipython** to output plots to the notebook instead of displaying them in separate window.

Create a new notebook by doing the following.

👉 Click on the **New Notebook** button.

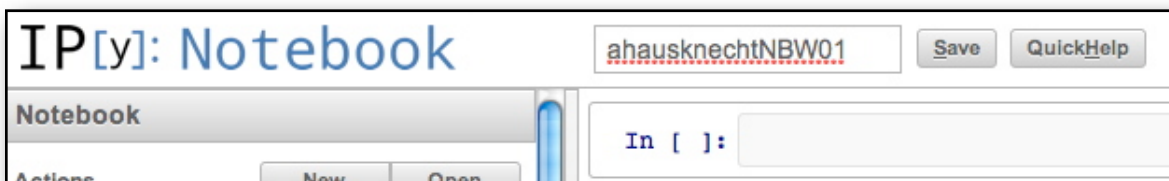
👉 A new untitled notebook will open in separate browser window.



👉 Click on the text field containing "Untitled1", enter "**yourNameNBW01**" and click the **Save** button.

👉 The notebook will be saved with the given name.

1.



To enter and execute a statement, do the following.



Click in the text field to the right of “In []:” and enter

$$5*0.6 - 0.5*9.81*0.6**2$$

Next press the **Shift and Return (Enter)** keys.

```
In [2]: 5*0.6-0.5*9.81*0.6**2
Out[2]: 1.2342
In [ ]:
```



The expression will be evaluated and the result will be displayed in an output cell (see the figure above).

To plot $y = \sin(x^2)$, do the following.



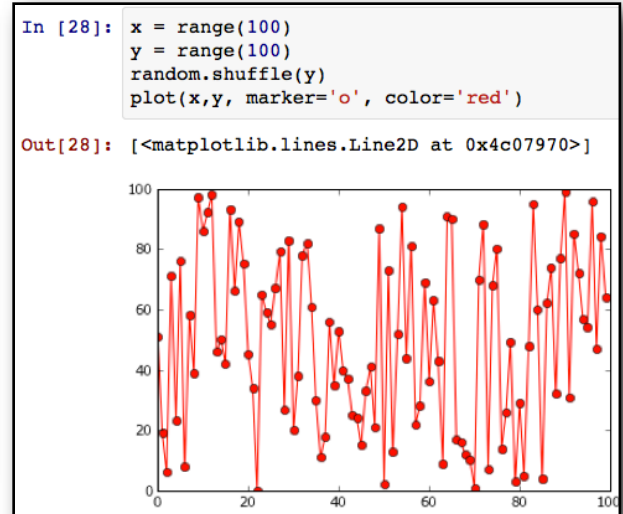
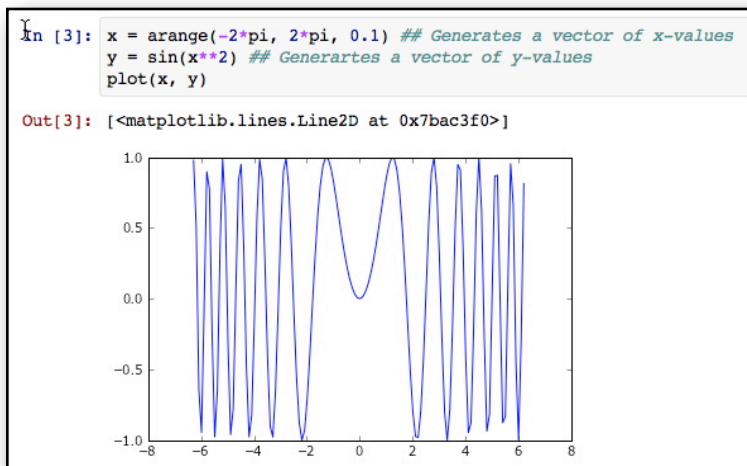
Click in the text field to the right of “In []:” and enter the lines (in the same cell)

```
x = arange(-2*pi, 2*pi, 0.1) ## Generates a vector of x-values
y = sin(x**2) ## Generates a vector of y-values
plot(x, y)
```

and press the **Shift and Return (Enter)** keys.



The plot will be generated and added to the notebook (see the figure on the left below).



To plot data, do the following: Click in the text field to the right of “In []:” after the previous plot and enter the lines (in the same cell)

```
x = range(100)      # Generate a list of 100 integers from 0 through 99
y = range(100)      # Generate a list of 100 integers from 0 through 99
random.shuffle(y)   # Scramble y
plot(x, y, marker='o', color='red') # Plot the data
```

and press the **Shift and Return (Enter)** keys.



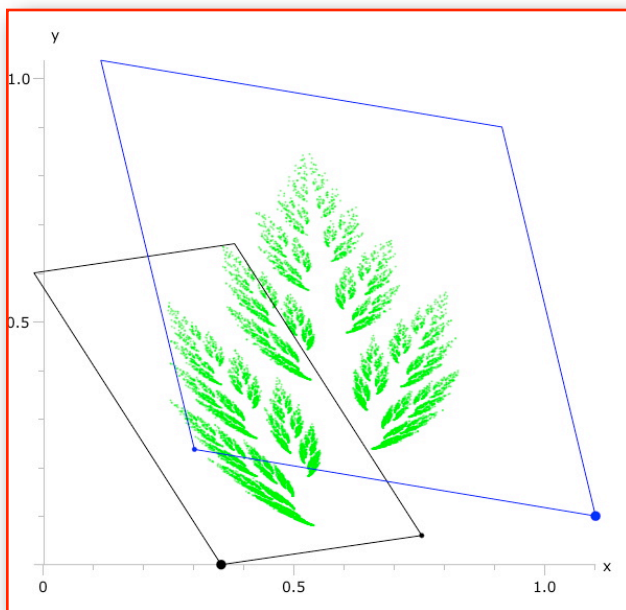
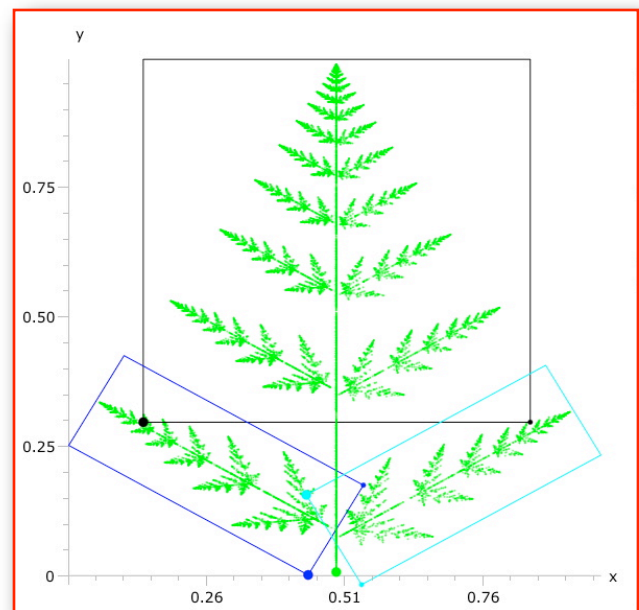
The plot will be generated and added to the notebook (see the figure on the right above).

Reasons of Learning Python:

1. Very popular in the Scientific Computation community.
2. Combined with the packages IPython, NumPy, Matplotlib, Scipy and others, Python can be used as an open-source replacement for MATLAB.
3. Can be used to glue together several programs written in different languages (FORTRAN, C/C++, Java).
4. Python is the programming language for many significant packages including the open-source computer algebra system Sage (<http://www.sagemath.org/>), the open-source 3D animation studio Blender (<http://www.blender.org/>), the 3D game engine Panda 3D (<http://www.panda3d.org/>), the open-source windowing and multimedia-OpenGL library Pyglet (<http://www.pyglet.org/>), and 3D Graphics and Simulation for everyone package Visual 3D (<http://vpython.org>).

Visual Python Examples:

1. **Fractals:** VisualPython can be used to create fractals such as the ones shown below:

**Leaf****Fern**

Recall that fractals are generated by iteratively applying a family of affine transformations $\{A_i\}_{i=1}^n$ that map the unit square $S = [0, 1] \times [0, 1]$ to a parallelogram or line segment. Starting with a random initial point p_0 in S , an A_i is selected with probability determined by the determinant of its linear part and a new point $p_1 = A_i(p_0)$ is calculated and plotted. This process is repeated over and over again. The set of all the generated points form the fractal images shown above. For example, the Leaf fractal shown on the left, was generated by the two affine transforms

$$A_1(x, y) = (0.4x - 0.3733y + 0.3533, 0.06x + 0.6y),$$

$$A_2(x, y) = (-0.8x - 0.1867y + 1.1, 0.1371x + 0.8y + 0.1).$$

The black and blue parallelograms shown in the Leaf figure are the images of the boundary of the unit square under A_1 and A_2 , respectively. Similarly, the Fern fractal, shown on the right above, was generated

using four affine transformation including one whose linear part has determinant zero which maps S to the vertical line segment from $(0.4987, 0.007)$ to $(0.4987, 0.307)$.

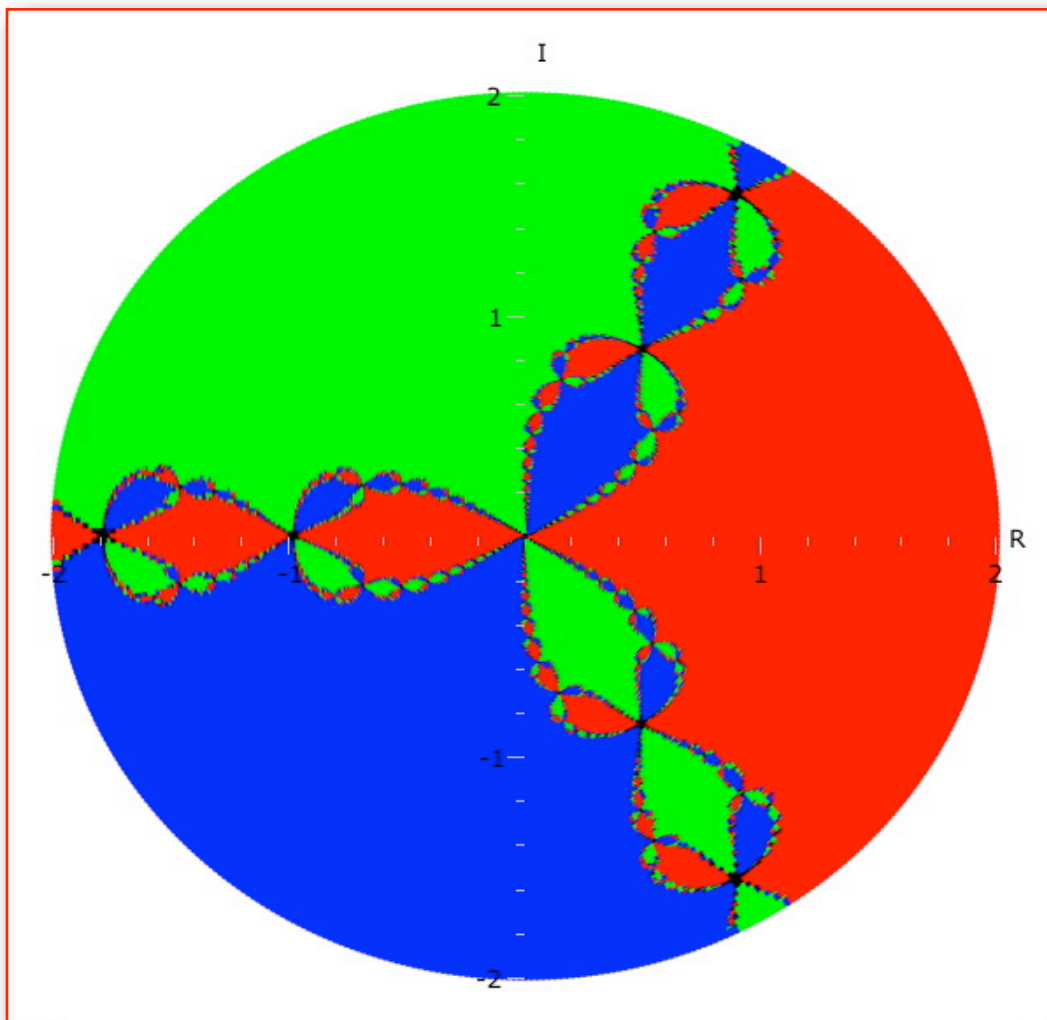
2. **Newton's Basin:** Recall that Newton's Method is used to find the roots of a polynomial including its complex roots. The method uses an initial guess z_0 in the complex plane and then generates a sequence of approximations $z_0, z_1, z_2, \dots, z_k, \dots$ that converge one of the roots of the polynomial. The question arises as to which root of the polynomial will be approximated a sequence generated from a given initial guess. The figure below (a fractal), gives the surprising answer for the case of the polynomial $p(z) = z^3 - 2$ whose roots are

$$r_{red} = \sqrt[3]{2}, \quad r_{green} = \eta \sqrt[3]{2}, \quad r_{blue} = \eta^2 \sqrt[3]{2}$$

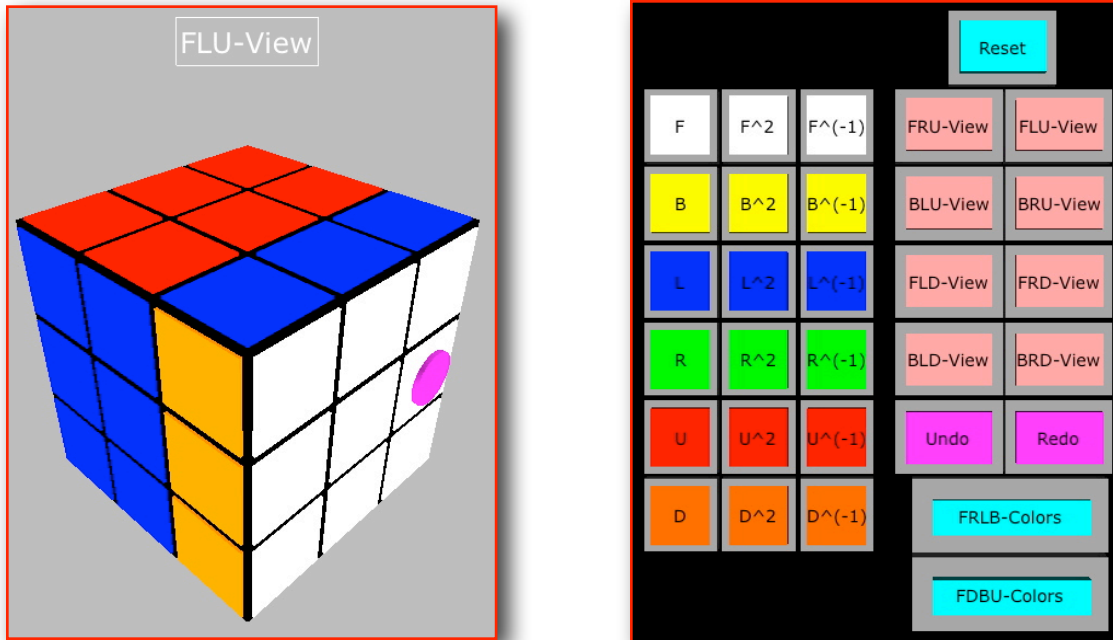
where

$$\eta = \cos(2\pi/3) + i\sin(2\pi/3) = (-1 + \sqrt{3}i)/2$$

is a primitive cube root of unity. Points in the red regions are initial guesses of sequences that converge to $\sqrt[3]{2}$. Similarly, points in the green or blue regions are initial guesses of sequences that converge to r_{green} or r_{blue} respectfully.



3. **Rubik's Cube Simulation.** Written for my *Modern Algebra* course (MTH 441 Fall 2011).



4. **Maze Ball Bounce:** This simulation was inspired by a student's incomplete project for my *Introduction to Scientific Computation* (MTH 280 Spring 2010). The idea is to experimentally determine which initial velocities of a ball released in the center of the maze will allow the ball to eventually exit the maze. The simulation assumes that a ball bounces off a wall without loss of energy such that the angle of incidence and angle of reflection are equal.

