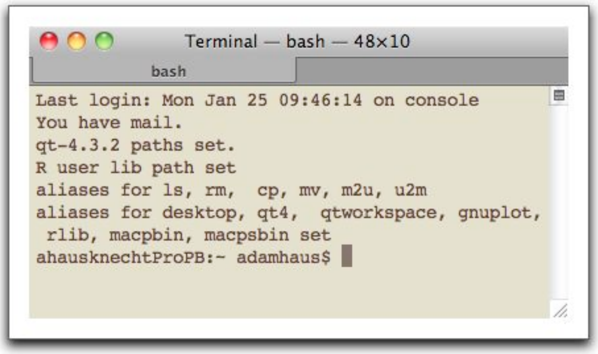
This Worksheet shows you several ways to start using Enthought's distribution of Python!

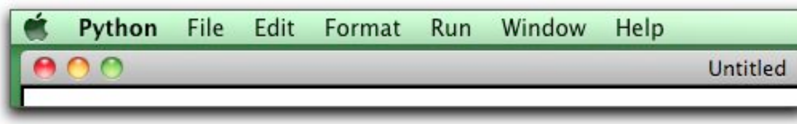
Start the **Terminal**  application by

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



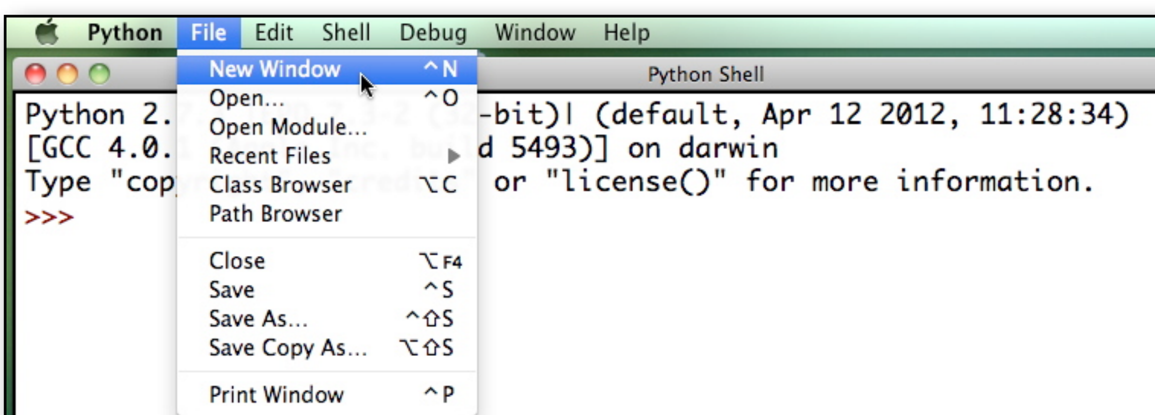
The menu's are used to control **idle**. Also, a message of the form "[k] process ID"

[1] 786

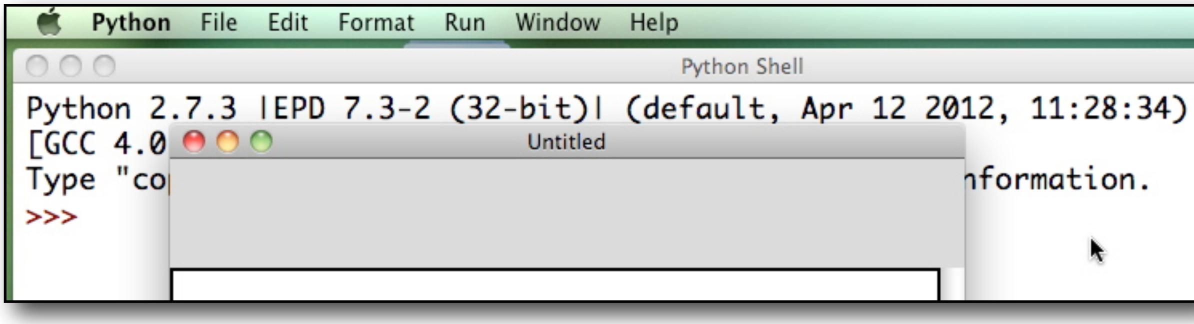
will be displayed in the Terminal's window. The integer **786** identifies the processing thread running **idle**. **Note**: The ampersand '**&**' tells the operating system to run idle as a separate process.

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

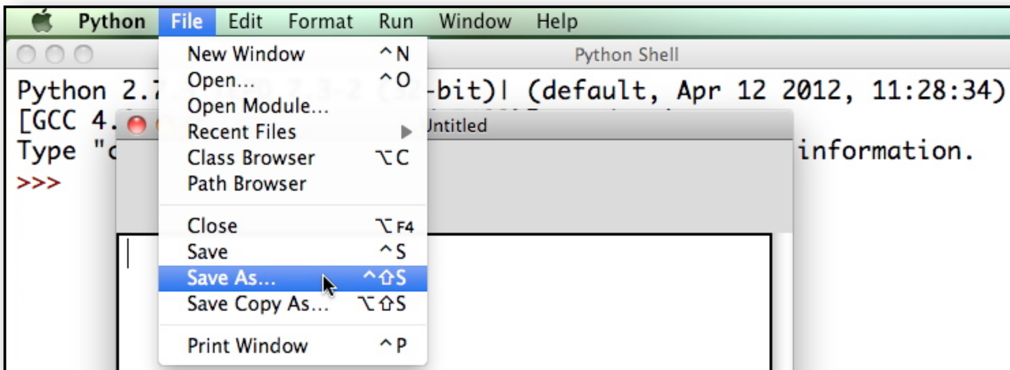
- Select **New Window** from **idle**'s **File** menu.



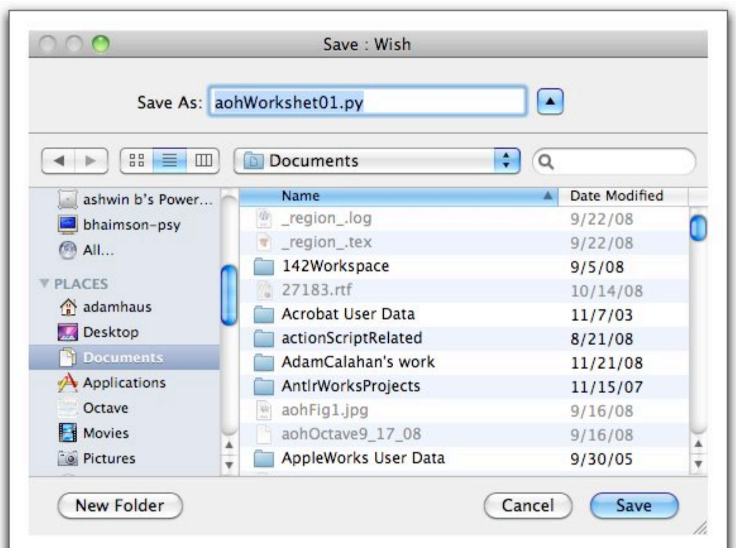
A new **idle** *program* window will open with the title **Untitled**.



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



A file dialog window will open.

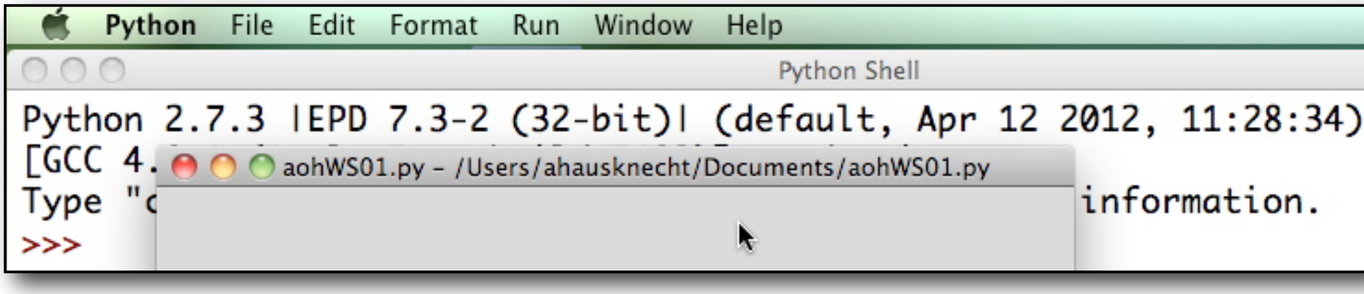


Navigate to the student's **Documents** folder. Enter a name a of the form

yourNameWS01.py

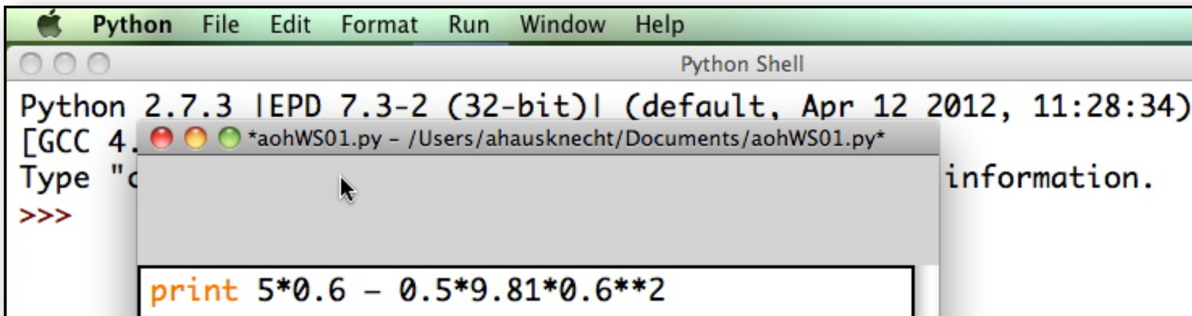
for the file and *press* the save button. Note the file's name must end with the suffix ".py".

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

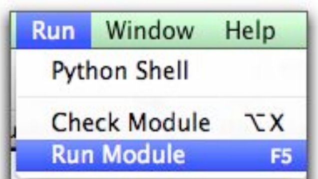


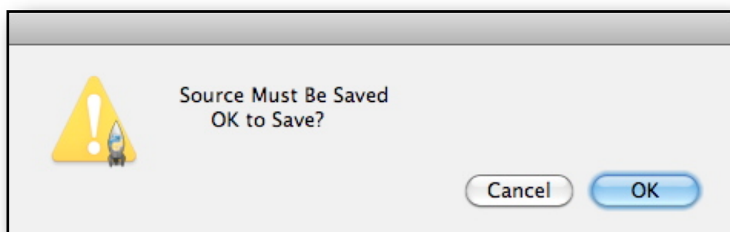
To execute a Python statement to the following:

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

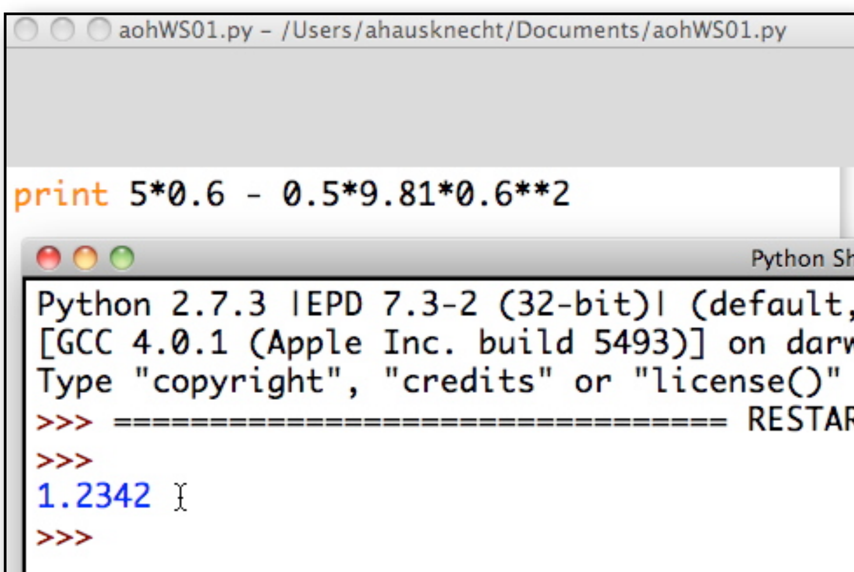


Then select the **Run Module** item from **idle**'s **Run** menu and click the **OK** button in the **Source Must Be Saved** dialog (you won't see this dialog if your program has already been saved).





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

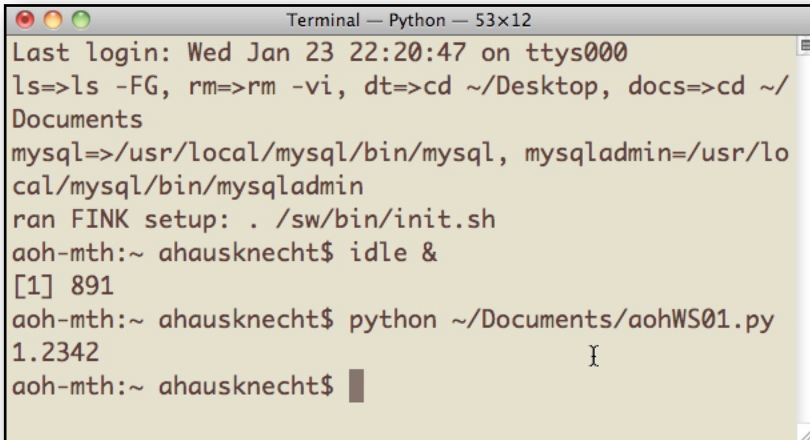


To run your one-line Python program saved in the file **yourNameWS01.py** and located in the student's Documents folder, do the following

✋ 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.



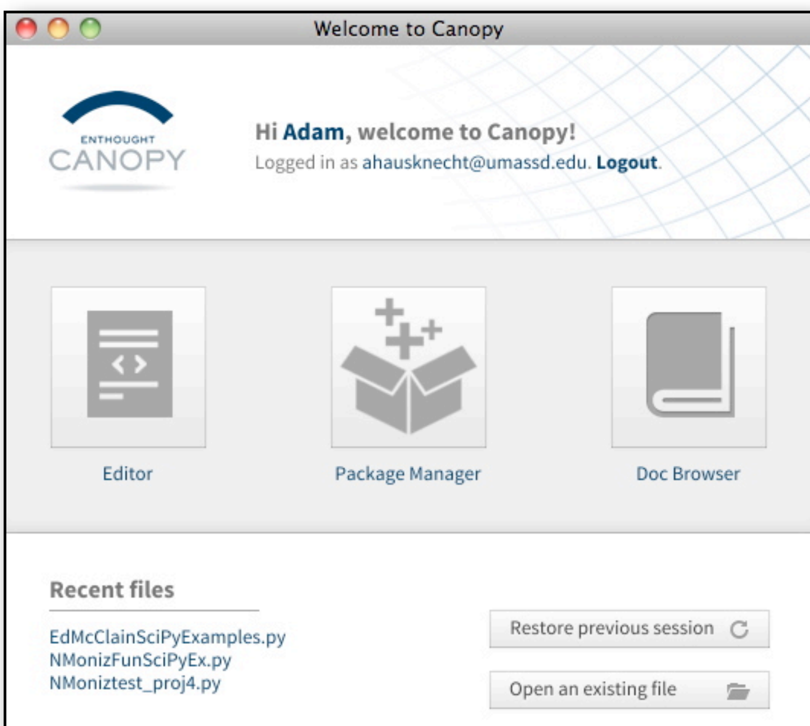
Another way to write Python programs is to use Enthought's new **Canopy** IDE. Start Canopy by

✋ Selecting the **Applications** item from the **Go** menu located at the top of the screen.

☞ The **Applications** folder will be displayed in a window.

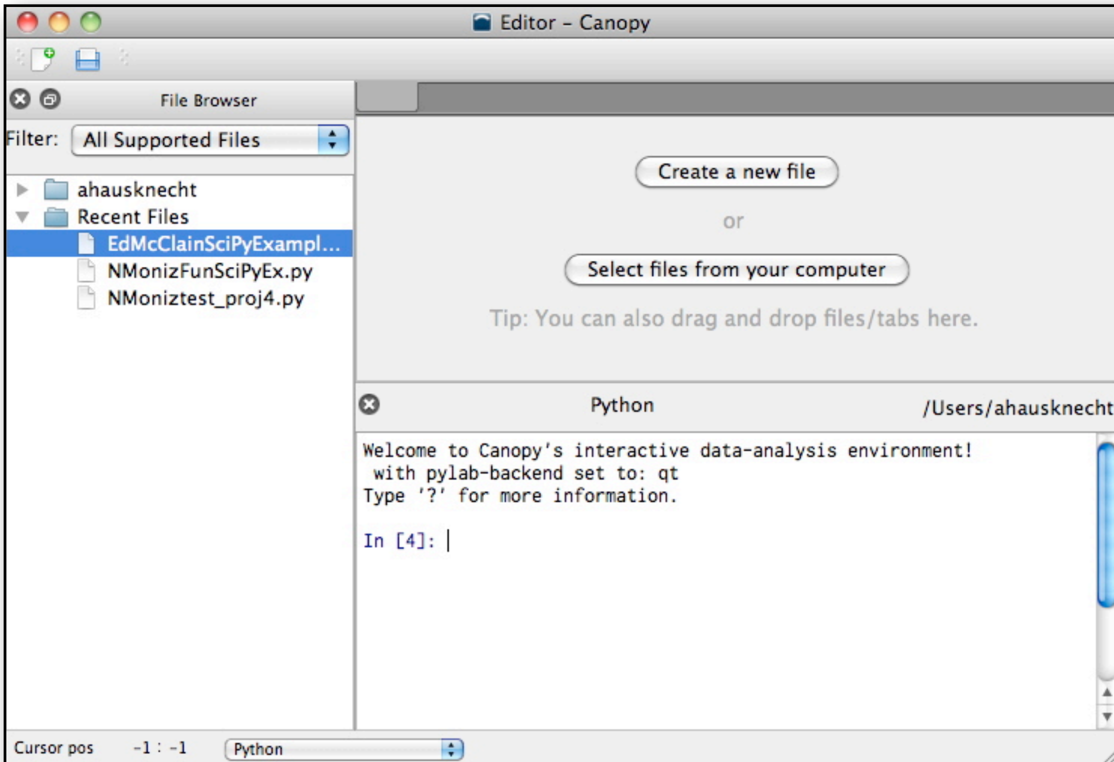
✋ Double-click the **Canopy** application's **icon** located in the Applications folder.

☞ The **Canopy** application will start and its window will open



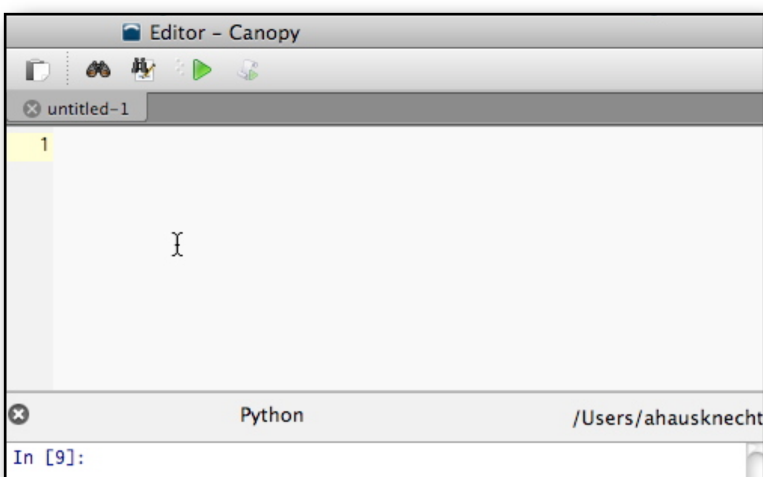
Click the **Editor** icon located on the right in Canopy's Welcome window.

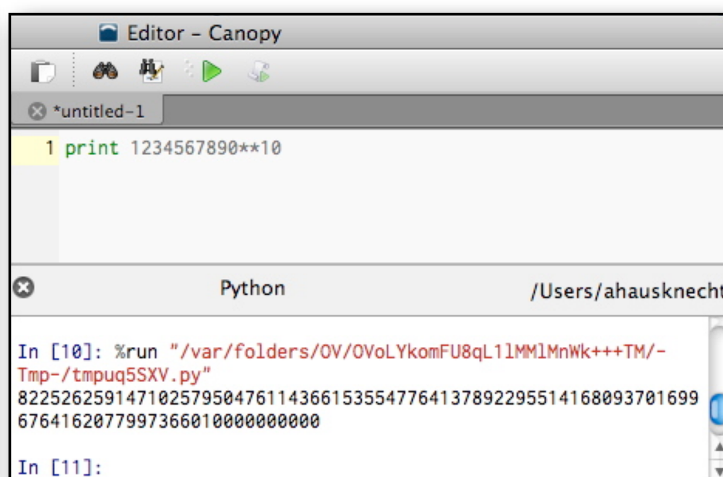
The **Canopy's Editor** window will open with three panes: the **File Browser** (left side), the **File Pane** (right top), and an interactive **Python Shell** (right bottom)



Click the **Create a new file** button in the editor.

The a new empty **Untitled-1** file will be displayed in the **File Pane**.



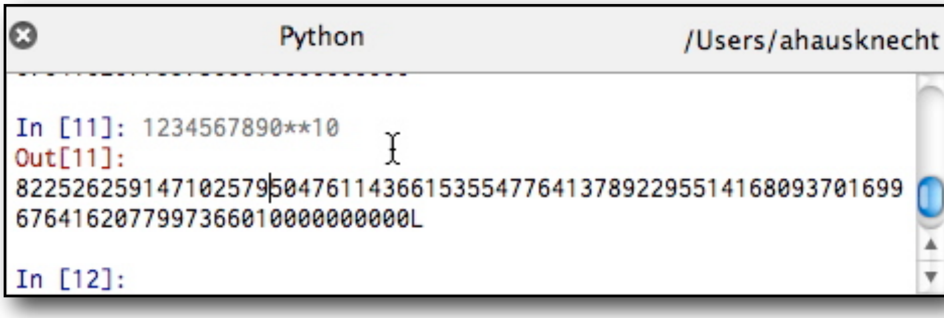


Enter **print 123456789**10** into **Untitled-1** and press the green arrow icon to run execute the file.

A temporary Python file will be created and executed. Its output is displayed in the Python shell.

Enter the expression **123456789**10** in the Python shell and press the **Return** key.

The expression is evaluated and the result is displayed in the Python shell.

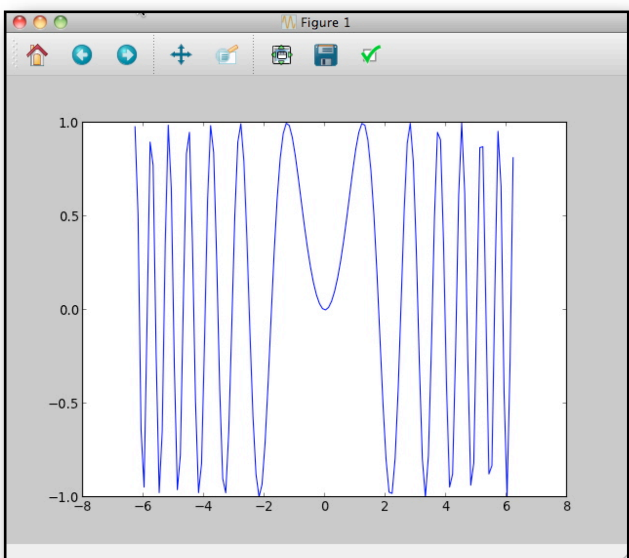


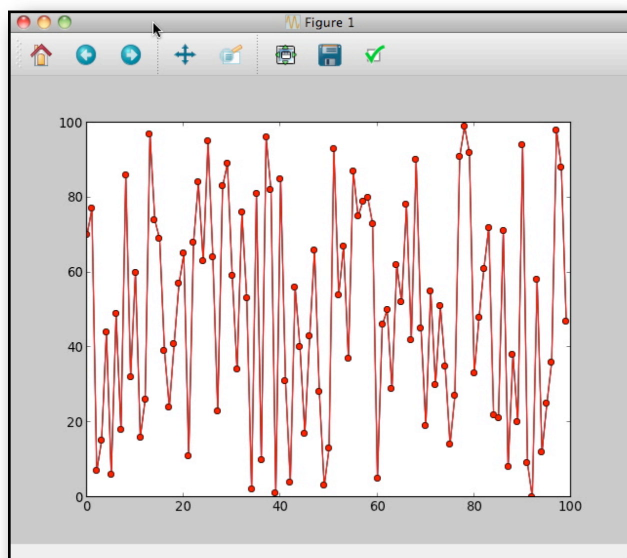
To plot $y = \sin(x^2)$ over $[-2\pi, 2\pi]$, enter the following lines in the Python shell after the prompt **In [n]:**

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

and *press* the **Return** key after entering each line.

A window will open containing the plot of $y = \sin(x^2)$ (see the figure on the left below).





To plot data, enter the following lines in the Python shell after the prompt **In [n]:**

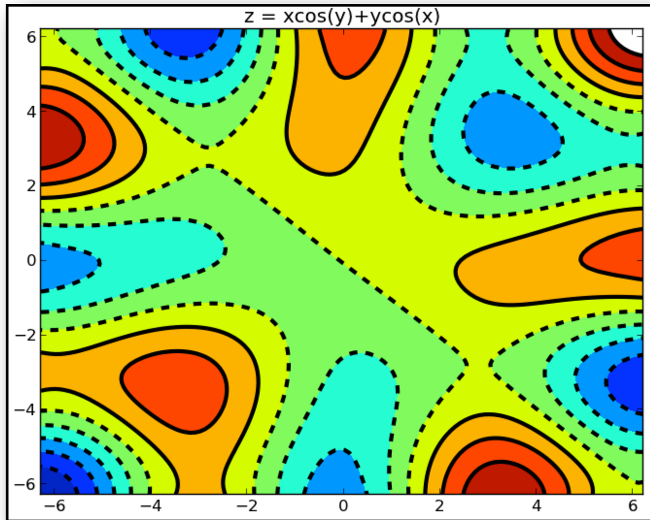
```
x = range(100) # Generate a vector of x-values from 0 to 99
y = range(100) # Generate a vector of y-values from 0 to 99
random.shuffle(y) # Randomize the order of the y-values
plot(x,y, marker = 'o', color = 'red')
```

and *press* the **Return** key after entering each line.

A window will open containing the plot of the data (see the figure on the right above).

To plot contours of $z = f(x,y)$, enter the following lines in the Python shell after the prompt **In [n]:**

```
In [1]: clf
In [2]: xx = arange(-2*pi, 2*pi, 0.1)
In [3]: yy = xx
In [4]: x, y = meshgrid(xx,yy)
In [5]: z = x*cos(y) + y*cos(x)
In [6]: zMin, zMax = z.min(), z.max()
In [7]: c = arange(zMin, zMax, (zMax-zMin)/10)
In [8]: hold(True)
In [9]: contourf(x, y, z, levels = c)
In [10]: contour(x, y, z, linewidths = 3,
               levels = c, colors = 'black')
In [11]: title('z = xcos(y)+ycos(x)')
In [12]: xlabel('x');  ylabel('y')
In [13]: show()
```



Basic Programming Examples:

Python uses *indentation* and colons ':' rather than parentheses to group statements. The indentation level in a block of statements must match! Examples of the **control** statements **if-elif-else**, **for-loop**, and **while-loop** are shown in the table. Notice that the two statements inside the **while-loop** are at the same indentation level.

if-elif-else	for-loop	while-loop

`x = random.randint(-10,10)`
`if x > 0:`
`  print 'x is positive'`
`elif x == 0:`
`  print 'x is zero'`
`else:`
`  print 'x is negative'`
	`for i in range(10, 20, 2):`
	`   print i, i**2, i**3`
	
	or
	
	`for i in [10, 12, 14,16,18]:`
	`   print i, i**2, i**3`
		`s, k , n = 0, 1, 100`
		`while k <= n:  # Start`
		`     s += k     # Add k to s`
		`     k += 1     # Add 1 to k`
		`s2 = n*(n+1)/2 # The fast way`
		`print 'The sum of integers ',`
		`print 'from 1 to 100 is',`
		`print ' %d = %d'% (s, s2)`

Try each of the examples above by entering them in Canopy's Python shell.
Here is an example of a Python function that finds square roots via **Newton's Method**:

```
def squareRootN(a):
    """Uses Newton's Method to find the square root of 'a'"""
    if a < 0: # Use a recursive call and convert to a complex number
        return complex(0, squareRootN(-a))
    elif a == 0:
        return 0
    else: # Now use Newton's method
        a = float(a)             # Make sure that 'a' is a float (not an int)
        x0, x1 = 0.0, a/2.0      # Initialize
        while x1 != x0:          # Repeat until x0 and x1 agree
            x0 = x1              # Save the previous guess
            x1 = (x0 + a/x0)/2.0 # Find a better guess
        return x0

ans = squareRootN(-3) # Test squareRootN
print ans, ans**2
```

Notice that this function finds roots of negative numbers as well as real numbers! The last two lines test the function. To try this example, do the following:

- Delete the text in the **Untitled-1** and enter the code shown above.
- Save the file by clicking the save icon (third from the left edge of the control bar) with the name **squareRootN.py** in the Documents folder.
- Press the run icon ▶ to execute the file.

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

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

✋ Click on the **Canopy's Welcome** window (to bring it to the front) and select **Quit** from the **Canopy** menu.

☞ **Canopy** will terminate.

Start **ipython's notebook** by

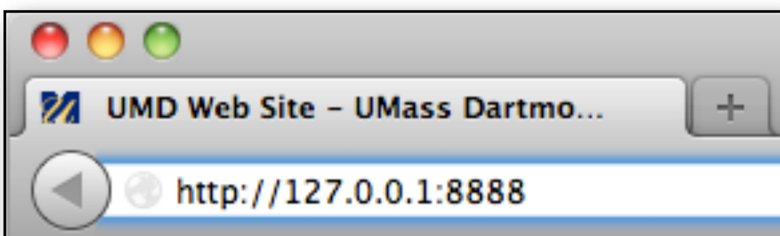
✋ Click on the **Terminal** window (to bring it to the front) and entering

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

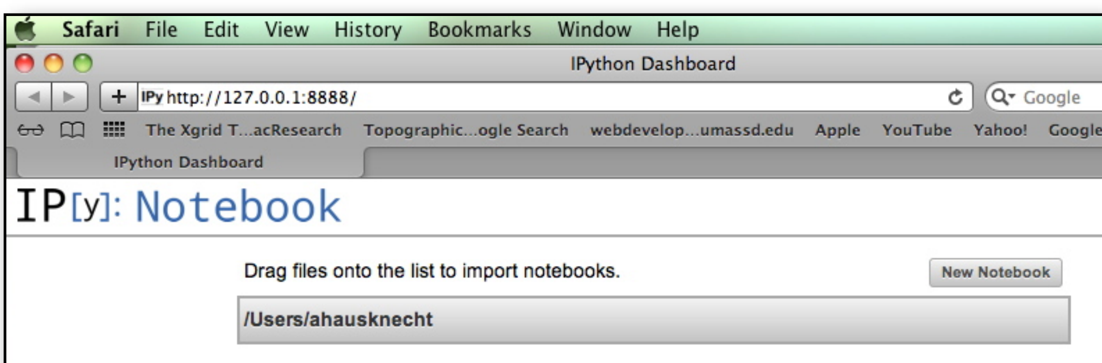
☞ The Terminal window will display messages similar to the last three lines shown below.

```
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.

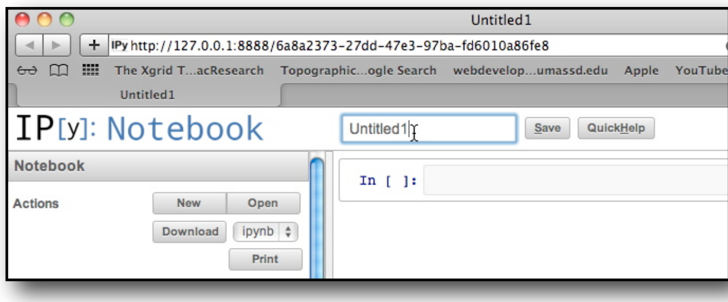


Note that the extra parameter **--pylab** causes **ipython** to automatically load important packages including *numpy* (large matrices, linear algebra) and *matplotlib* (plotting) needed for scientific computation. The inline parameter tells **ipython** to output plots to the notebook instead of displaying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

Create a new notebook by doing the following.

Click on the **New Notebook** button.

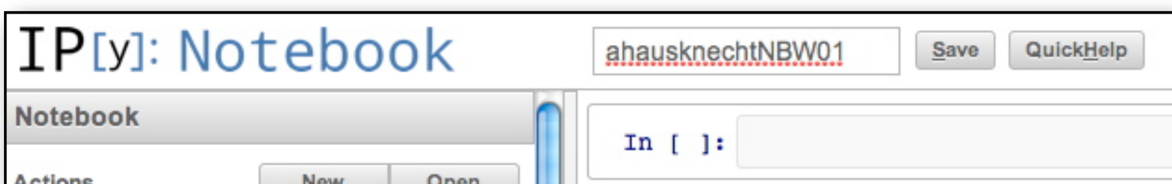
A new untitled notebook will open in separate browser window.

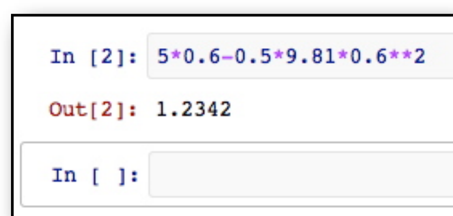


Click on the text field containing "Untitled1", enter **"yourNameNBW01"** and click the **Save** button.

A 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.

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 page 6).

Try any of the other examples above in the **ipython's** notebook.

Additional Examples: Enter any of the following in **ipython's** notebook.

EXAMPLE 1: A While-loop to convert degrees Fahrenheit to degrees Centigrade.

1. NOTE: Use '%g' for number formatting. The '%g' uses as little space as possible to print the variable's value.
2. NOTE: Use '+=' operator to increment the variable 'C'.

ENTER THE FOLLOWING LINES IN AN EMPTY IPYTHON CELL!
Press the SHIFT-REURN keys together to execute the code!

```
print '-----------------------------'# Table Heading.
C, dC = -20, 5                      # Starting value for C and increment of C in loop.
while C <= 40:                      # loop heading with condition.
   F = (9.0/5.0)*C + 32             # Conversion to degrees Fahrenheit.
   print "%4g C %4g F"%(C, F)       # Print formatted results.
   C += dC                          # Increment C and return to top of loop.
print '-----------------------------'# Table ending.
```

EXAMPLE 2: Basic operations on lists.
A basic list can be thought of as a vector or 1 by n matrix.
ENTER EACH LINE BELOW IN **DIFFERENT EMPTY** IPYTHON CELL.
Press the SHIFT-REURN keys together to execute each cell's code!

```
C = [-10, -5, 0, 5, 10, 15, 20, 25, 30] # Create a list.
C.append(35)          # Add a new element.
print C               # PRESS SHIFT RETURN to view the list.
C = C + [40, 45]      # Extend the list.
print C               # PRESS SHIFT RETURN to view the list.
C.insert(0, -15)      # Insert at front of list.
print C               # PRESS SHIFT RETURN to view the list.
del C[2]              # Delete C's third element.
print C               # PRESS SHIFT RETURN to view the list.
print len(C)          # PRESS SHIFT RETURN: Length of the list C.
print C.index(10)     # PRESS SHIFT RETURN: Location of 10 in C.
print 10 in C         # PRESS SHIFT RETURN#: Is 10 an element of C?
print C[-1]           # PRESS SHIFT RETURN: View the last element of C.
print C[-2]           # PRESS SHIFT RETURN: Next to last element of C.
```

EXAMPLE 3: Degree conversion via for-loop
ENTER THE LINES BELOW IN AN EMPTY IPYTHON CELL.

```
Cdegrees = [-20, -15, -10, -5, 0, 5, 10, 15, 20, 25, 30, 35, 40]
for C in Cdegrees:
    F = (9.0/5)*C + 32
    print C, F
```

EXAMPLE 4: Use the shorthand method for generating lists: `list = [ expression(x) for x in list ].`
ENTER THE LINES BELOW IN AN EMPTY IPYTHON CELL.

```
Cdegrees = [-5 + i*0.5 for i in range(21)]
Fdegrees = [ (9.0/5)*C + 32 for C in Cdegrees]
# Print the results as a right justified tables using the 'xd' and 'x.yE' format
# specifiers  and the 'zip' lists function.
print "    C        F\n--------------"
for C, F in zip(Cdegrees, Fdegrees): # zip produces pairs of C's and F's
    print "%5d %5.2E" %(C,F)
```

EXAMPLE 5: A Recursive Function

ENTER THE FOLLOWING IN A NEW IPYTHON CELL!

```
def fact(n):
    if n < 2:  # The basis step
        return 1
    else: # The recursive step
        return n*fact(n-1)

print "%d! = %d"%(50, fact(50))
```

EXAMPLE 6: Approximating the sine function

Plot Taylor polynomial approximations to Sin(x)

sin(x) = x - x^3/3! + x^5/5! - x^7/7! + ...

ENTER THE FOLLOWING IN A NEW IPYTHON CELL!

```
def taylorSin(x, n):
    s, xSqr, term = x, x**2, x
    for i in range(1,n):
        twoI = 2*i
        term *= -xSqr/float((twoI+1)*twoI)
        s += term
    return s

## Generate the x-values
xValues = arange(-2*pi, 2*pi, .01)
n = len(xValues)
yValues = sin(xValues)
## Plot y = sin(x)
plot(xValues, sin(xValues), 'r', linewidth = 3)
axis([-2*pi, 2*pi, -1.5, 1.5])
## Now overlay plots of the Taylor polynomial approximations
for p in range(5):
    for i, x in zip(range(n), xValues):
        yValues[i] = taylorSin(x, p)
    plot(xValues, yValues)

title("Taylor Polynomial Approximations to y  = sin(x)")
xlabel('x'); ylabel('y');
```

EXAMPLE 7: Using SciPy's quad function to plot a function defined by an integral.

Plot the function $f(x) = \int_1^x \frac{\sin(t)}{t} dt$.

ENTER THE FOLLOWING IN A NEW IPYTHON CELL!

```
from scipy.integrate import quad
xValues = arange(0, 10, .01)
yValues = zeros(len(xValues)) # Create a vector of zeros of the same length as xValues
errors = zeros(len(xValues))
i=0
for x in xValues: # Use quad to approximate f(x) with the error estimate
    yValues[i], errors[i] = quad(lambda t: sin(t)/t, 1, x) # Pass an anonymous function
    i += 1
print "Max Error: ", max(errors)
plot(xValues, yValues, '+-')
title("y(x) = integral(sin(t)/t, 1, x)")
xlabel('x'); ylabel('y');
```

EXAMPLE 8: Solution of $y''+py'+qy'=0$ for various initial conditions.

Step 1: Convert the 2nd-order linear ODE to a first-order 2D system

Let $v=y'$, then $dv/dt=y''$; hence,

$$dy/dt = 0\cdot y + 1\cdot v$$

$$dv/dt = -q\cdot v - p\cdot v$$

Thus, in terms of matrices and vectors

$$\mathbf{X}' = \begin{bmatrix} dy/dt \\ dv/dt \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -q & -p \end{bmatrix}\begin{bmatrix} y \\ v \end{bmatrix}$$

ENTER THE FOLLOWING IN A NEW IPYTHON CELL!

```
from scipy.integrate import odeint
# UNCOMMENT/COMMENT OUT EACH OF lines A., B., C., or D.
# A. Undamped
p = 0; q = 1;
# B. Over-damped
## p = -1.0;  q = -2.0;
# C. Critically-damped
## p = 2.0; q = 1.0;
# D. Under-damped
## p = 2.0; q = 2.0;
##
def func(X, t):
    return [0*X[0]+1*X[1], -q*X[0]+-p*X[1]]

t = arange(0, pi, .01)
initialConds = []
for y0 in arange(-1, 1.25, 0.25):
    for dydt0 in arange(-1, 1.25, 0.25):
        initialConds.append([y0, dydt0])

figure(1); hold(True)
title("Solutions of y'' + py' + qy = 0")
xlabel('t'); ylabel('y');
figure(2); hold(True)
title("Phase Plane Plots of \nSolutions of y'' + py' + qy = 0")
xlabel('y'); ylabel('dy/dt');
for X0 in initialConds:
    X = odeint(func, X0, t)
    figure(1); plot(t, X[:,0])
    figure(2); plot(X[:,0], X[:,1])
```

EXAMPLE 9: Near Resonance: Solution of $y'' + \omega^2 y = a\cos(bt)$

In terms of matrices and vectors $\mathbf{X}' = \begin{bmatrix} dy/dt \\ dv/dt \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega^2 & 0 \end{bmatrix} \begin{bmatrix} y \\ v \end{bmatrix} + \begin{bmatrix} 0 \\ a\cos(b) \end{bmatrix}$

ENTER THE FOLLOWING IN A NEW IPYTHON CELL!

```
from scipy.integrate import odeint
w = 2.0; a = 1.0;
dw = 0.1 ## Should be small;
b = w + dw
def func(X, t):
    return [0*X[0]+1*X[1], -(w**2)*X[0] + 0*X[1] + a*cos(b*t)]

t = arange(0, 120, .01)
figure(1); hold(True)
tStr = "Near Resonance Solution of\n"
title(tStr + "y'' + w^2y = acos(bt), b = w+%g"%(dw))
xlabel('t'); ylabel('y');
figure(2); hold(True)
tStr = "Phase Plane Plot of a Near Resonance Solution of\n"
title(tStr + "y'' + w^2y = acos(bt), b = w+%g"%(dw))
xlabel('y'); ylabel('dy/dt');
X = odeint(func, [1, 0], t)
figure(1); plot(t, X[:,0])
figure(2); plot(X[:,0], X[:,1])
```

EXAMPLE 10: Modeling the Interior Temperature of a Barn using the model:

$$dT/dt = k\left(C(t) - T\right) = f(T)$$

where $T(t)$ = temperature of the inside of a barn with no internal heating or cooling with $T(0) = 60$

k = temperature coefficient = 0.25

$C(t)$ = temperature outside the barn $= 70 - 10\cos\left(\frac{\pi}{12}t\right), 0 \le t \le 24$.

ENTER THE FOLLOWING IN A NEW IPYTHON CELL!

```
from scipy.integrate import odeint
k = 0.25
def C(t):
    return 70 - 10*cos(pi/12*t)
## Create a function that returns f(T) as a 1 x 1 matrix.
def func(T, t):
    return [k*(C(t) - T[0])] ## Note that T is a vector of length 1

## Create a vector of t-values
t = arange(0, 24, .01)
T = odeint(func, 60, t)
plot(t , T, linewidth = 2)
plot(t , C(t), 'r', linewidth = 2)
title("Solution of dT/dt = k(C(t)- T), \n where C(t)= 70 - 10cos(pi/12 t)")
xlabel('t'); ylabel('T');
## Draw a horizontal line at at Ts
axhline(y=70, color = 'k', linewidth = 3)
## Set the axes limits
axis([0, 24, 50, 90])
```

EXAMPLE 11: A Simple Game

Notes: 1. "raw_input(...)" is used for keyboard input.

2. "if-elif-else" statement is used to test a user's guess.

3. In general, what is minimum number of many guesses needed to win?

Start **idle** (or **canopy)** and create a new file window (or pane) and save it as "YourNameGame.py".

ENTER THE CODE BELOW IN THE NEW WINDOW.

```
import math
from random import randint
# Generate a target integer to guess in the range 0 to 1023.
target = randint(0, 1024)
while True: # loop forever.
    # Request text input from the user with a prompt.
    guess = raw_input("Enter an integer guess (0 to 1023, or 's' to STOP) => ")
    # Check to see if the user entered a string of digits.
    if not guess.isdigit():
        break  # user wants to stop, so jump out of the loop.
    # Convert the digit string to an integer.
    intGuess = int(guess)
    # Test the guess using Python's "if-elif-else" statement
    # and print a message to the user.
    if intGuess == target:
        print "You won!"
        # Generate a new target to guess.
        target = randint(0, 1024)
    elif intGuess > target:
        print "Your guess is too big!"
    else:
        print "Your guess is too small!"
#
print "The game is over!"
```