Preparation

- Download the project files from the Course Agenda (on the web site)
- Install them in a convenient directory.
- Open a terminal and “cd” to that directory.

Approaching Programming

Useful Practices

Objectives

Communicate a basic understanding of the programming process that will help you appreciate software limitations & troubleshoot problems with software used in seismological research.

Help you to start to develop basic programming skills to help you construct tools that you can use to process and analyze data.

Outline

- Learning to program & programming
- Activities
    - An Introduction to Programming Editors
    - An introduction to source-code management with git.

Programming

Programming skills empower you to explore ideas with a computer on your own.

Although we cannot teach you programming in a week, we can get you started.

To succeed, you have to continue to work at it, but a week of practice and discussion can move you far in the right direction.

Good Seismology Programming Practices

Remember, you are a geoscientist, not a programmer.

- Understand what you are programming.
  - Define your goals
    - Quick & Specific
    - General & Reusable
  - Outline
  - Create a task list
  - Create a flowchart
- Read code written by others
  - When do they use functions?
  - How do they structure their code?
- First, learn one language well.
- Learn a scripting language
  - bash
  - PERL
  - Python
  - javascript
  - Matlab
  - Mathematica
- Learn a compiled language
  - Fortran
  - C
  - C++
- Learn object-oriented programming
- Learn a good graphics library
- Use a code snippet utility
- Keep a programming journal

- Use a Package Manager
  - homebrew
  - macports
  - apt
- Use a good editor
  - Atom
  - Sublime Text
  - BBEdit
  - vim
- Comment your codes
  - Doxygen
  - ROBOdoc
- Write test codes as you develop
- Use Make or an IDE (Integrated Development Environment)
- Use git
- Know your compiler options
  - -g (debug)
  - -pg (profile)
  - -O (optimize)
- Use a debugger
  - gdb
  - idb
  - spyder (pydb)
- Profile your code
  - gprof
  - Instruments (OS X)
- Use maintained numerical libraries
  - LAPack
  - FFTW
  - ODEPack
  - GNU
  - Optimization
  - Numerical Recipes

Charles J. Ammon, Penn State, 2015

Activities

- First, we will explore a programming editor.
  - Goal: Show you enough to encourage you to find a good editor.
- Second, we will explore git, the source-control management system.
  - Goal: Show you enough to encourage you to learn enough to regularly use git.

The Atom Editor

Open the File

`Editors_and_Codes/docs/Programming_Editors.pdf`

Open the Atom Editor

Program Types

We can divide computer programs into two types

1) Interpreted

2) Compiled

Interpreted codes are usually lighter weight and slower than compiled, but they are often more transportable and can be powerful (especially when combined with compiled codes).

Interpreted Languages

Commonly used interpreted languages include

The UNIX Shell	Python
AWK	MatLab
PERL	Postscript
PHP	Javascript

These languages vary in their complexity and their power, but they all rely on an interpreter.

Shell Scripts as Programs

A shell script is a procedure or “recipe” to perform a sequence of UNIX commands.

```
#!/bin/csh
set PSFILE = 'tmap.ps'
set PROJ = '-JM6 -P -V'
set LIMITS = '-R90/102/0/14'
set TICKS = '-B2g.5/2g.5WeSn'
#
# Plot the seafloor
grdimage seafloor.grd $PROJ $LIMITS  -Y2.0 -Cus.cpt -K >! $PSFILE
#
#  plot the coastline
pscoast $PROJ $LIMITS $TICKS -Di -W0.5p -Na -O -K >> $PSFILE
#
awk '{print $7, $6, $9*$9*$9/2000}' neic.txt >! usgs.xy
#
psxy  usgs.xy $PROJ $LIMITS -Sc -G255/250/129 -W0.5p -O >> $PSFILE
```

Example - The AWK Interpreter

AWK Program File

```
BEGIN {
minNorth = 1e20;
maxNorth = -1e20
}
#  main processing script
{
if($2 == 6) {
if(minNorth > $7) minNorth = $7;
if(maxNorth < $7) maxNorth = $7;
}
#
END {
print "Min North: ", minNorth, "Max North: ", maxNorth
}
#
```

Interpret & Execute

```
> awk -f stripLocs.awk glData.txt

Min North:  213818.9735 Max North:  213819.1360
```

An Example - The Python Interpreter

Python Program File

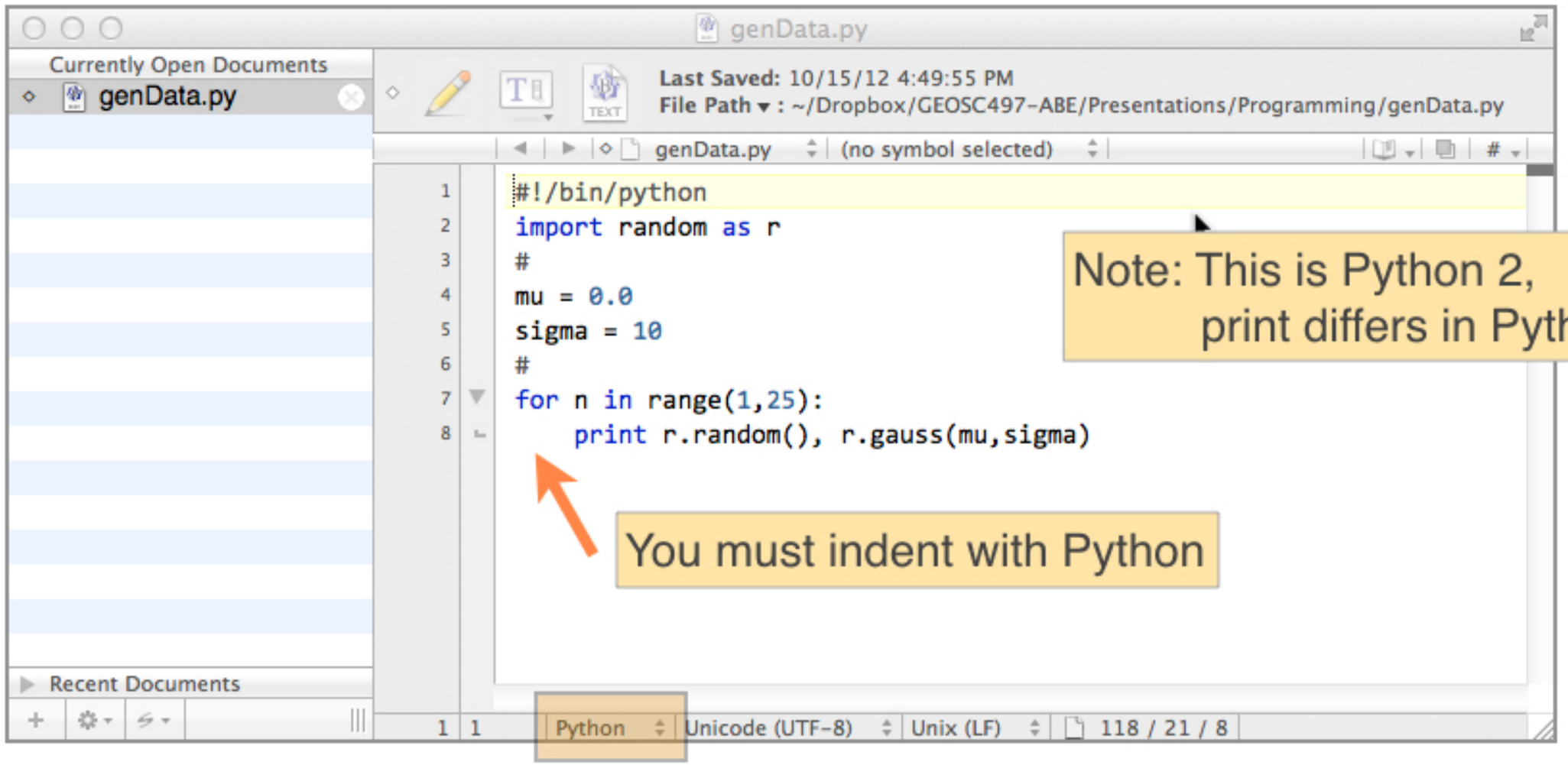
```
#!/bin/python
import random as r
#
mu = 0.0
sigma = 10
#
for n in range(1,25):
    print r.random(), r.gauss(mu,sigma)
```

Note: This is Python 2, print differs in Python 3

Interpret & Execute

```
> python genData.py > glData.txt
```

Do it. Create the following Python file and run it. Edit with BBEdit if you have it.



Interpret & Execute

```
> python genData.py > glData.txt
```

Terminal — tcsh — 89×27

```
[Ammons-MacBook-Pro:GEOSC497-ABE/Presentations/Programming] cammon% python genData.py
0.304222579149 1.7175453153
0.0284140753588 -13.1814360145
0.16487081825 5.45244042034
0.295709946723 -7.68701603155
0.410412963977 -0.957735646336
0.622056166466 24.4207460851
0.0143451005923 5.3739826658
0.303664231751 15.0623237651
0.0126316386501 -8.56806142462
0.443227505491 5.45321281135
0.173479033642 6.37559037086
0.65565278018 2.67860824386
0.434373046531 6.90835973232
0.127377689191 -9.14073778803
0.664825765818 9.90544598142
0.24683458231 5.55486744449
0.589830663469 -3.48697877644
0.120494302323 -9.28015702125
0.943648960144 1.05936307301
0.22687591573 -0.737321948026
0.965301131119 1.18509438918
0.583336740737 -4.05910269699
0.179572728021 4.6398088405
0.81631429638 9.55774160312
[Ammons-MacBook-Pro:GEOSC497-ABE/Presentations/Programming] cammon%
```

The results of running the python script: 25 pairs of random numbers.

Compiled Languages

Commonly used compiled languages include

Fortran
C
C++
Objective-C

These languages all rely on an compiler. You write the code and then compile it.

Scientific Programming

The number of programming languages continues to grow, as specialized tools continue to develop.

The dominant scientific languages are

Fortran
C & C++
Python
Java

Or a mix, often linked together with shell scripts.

Fortran Families

Fortran is a popular scientific programming language because it is fast & has complex variables native to the language. Fortran comes in a number of flavors

- Fortran IV, V, VI - really old stuff
- Fortran 77, common but old
- Fortran 90, see Fortran 95
- Fortran 95, 03, 08

C-Language Families

C is a popular computer system and scientific programming language because it is fast relatively portable. C comes in a number of flavors

ANSI C	The language of UNIX (Unix is written in C)
C++	C extended to include object oriented tools.
objective-C	Apple's older language (Mac, iOS)

Compiling Programs

To create faster executables, compilers translate programming languages into machine instructions that execute directly in the hardware.

Usually done together.

The GNU Compilers

If you are going to use the open-source C, C++, objective-C, or Fortran, you are likely going to use the gnu compiler suite.

gcc - C/C++/Objective-C compiler

gfortran - Fortran compiler

Compiled vs Interpreted

abeTestCalc.f

Last Saved: 10/16/12 5:10:01 AM
File Path : ~/Dropbox/GEOSC497-ABE/Presentations/Programming/ABE-Programming01-Files/abeTestCalc.f

```
      program abeTestCalc
c
c     compile with: gfortran -o abeTestCalc abeTestCalc.f
c     or
c     compile with gfortran -o abeTestCalc -O3 abeTestCalc.f
c
      double precision x, pi
      integer i

      pi = acos(-1.0D0)
      x = 0.0
      do 1 i=1,100000000
         x = x + dsin(0.5*pi)
1     continue

      write(*,*) x, pi

      stop
      end
```

a fortran code
-compiled-

abeTestCalc.py

Last Saved: 10/16/12 4:57:26 AM
File Path : ~/Dropbox/GEOSC497-AB...ing01-Files/abeTestCalc.py

```
#!/bin/python
import math
#
pi = math.acos(-1.0)
#
n = 1
x = 0
while n <= 100000000:
    x = x + math.sin(0.5*pi)
    n = n + 1
print x, pi
```

a python code
-interpreted-

Both codes sum
100,000,000 sin(x) values.

Compiled vs Interpreted

abeTestCalc.c

Currently Open Documents
- abeTestCalc.c
- abeTestCalc.f
- abeTestCalc.py

Last Saved: 10/16/12 8:27:29 AM
File Path : ~/Dropbox/GEOSC497-ABE/Presentations/Programming/ABE-Programming01-Files/abeTestCalc.c

abeTestCalc.c | (no symbol selected)

```c
#include <stdio.h>
#include <math.h>

/*
 compile with: gcc -o abeTestCalc abeTestCalc.c
*/
main()
{
    int n;
    double x, pi;

    pi = acos(-1.0);

    n = 0;
    while(n < 100000000)
    {
        x = x + sin(0.5*pi);
        n += 1;
    }
    printf("x = %f, pi = %.10f\n",x,pi);
}
```

a C code compiled

Sum 100,000,000 sin(x) values.

Recent Documents

22 | 1 | ANSI C | Unicode (UTF-8) | Unix (LF) | 301 / 39 / 22

Run Time Comparison

Comparisons are hard to construct, but the basic idea is illustrated with the simple example.

Time Used

```
Terminal — tcsh — 60×16
unix >time python abeTestCalc.py
100000000.0 3.14159265359
49.200u 0.061s 0:49.43 99.6%    0+0k 0+0io 0pf+0w
unix >gcc abeTestCalc.c -o abeTestCalc
unix >time abeTestCalc
x = 100000000.000000, pi = 3.1415926536
1.672u 0.000s 0:01.67 100.0%    0+0k 0+0io 0pf+0w
unix >gfortran -o abeTestCalc abeTestCalc.f
unix >time abeTestCalc
   100000000.00000000         3.141592653589793
2.095u 0.001s 0:02.09 100.0%    0+0k 0+0io 0pf+0w
unix >
```

Mathematica:
Pi = 3.14159265358979

Programming Constructs

The fundamental programming constructs are elements of any major programming language, interpreted or compiled. They include

Loops

foreach / for / do / while

Conditionals

if-then-else-end / case-switch

Generic Loops

```
while(boolean statement)
{
    do something if statement is true ...
}
```

```
for(i=0; i<nData; i++)
{
    do something...
}
```

Example Loops

A C Language While Loop

```
x = 0;
while(x <= 10)
{
  printf("%f %f\n", x, x*x);
  x = x+1;
}
```

A Python While Loop

```
x = 0
while(x <= 10):
   print format(x,"0.2f"),format(x*x,"0.2f")
   x = x + 1
```

Note: This is Python 2,
print differs in Python 3

Example Loops

A Fortran Language While Loop

```
While loops exist in recent fortran version
but are not sure to exist in F77.
```

A Matlab While Loop

```
x = 10;
while(x <= 0)
   sprintf('%5.0f %5.0f',x,x*x)
x = x + 1;
end
```

Do it. Start up the python interpreter and enter the following commands.

```
Terminal — tcsh — 76×22
[Ammons-MacBook-Pro:~] cammon% python
Python 2.7.2 (default, Jun 20 2012, 16:23:33)
[GCC 4.2.1 Compatible Apple Clang 4.0 (tags/Apple/clang-418.0.60)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> x = 0
>>> while(x <= 10):
...     print format(x,"0.2f"),format(x*x,"0.2f")
...     x = x + 1
...
0.00 0.00
1.00 1.00
2.00 4.00
3.00 9.00
4.00 16.00
5.00 25.00
6.00 36.00
7.00 49.00
8.00 64.00
9.00 81.00
10.00 100.00
>>> ^D
[Ammons-MacBook-Pro:~] cammon%
```

You must indent with Python

Note: This is Python 2,
print differs in Python 3

Things I typed.

Example Loops

A Fortran Language Do Loop

```
Do 10 i = 1,10
  write(stdout,'(f10.4,1x,f10.4)') x, x*x)
10    continue
```

A Matlab For Loop

```
for i=1:10
   sprintf('%5.0f %5.0f',x,x*x)
end
```

Example Loops

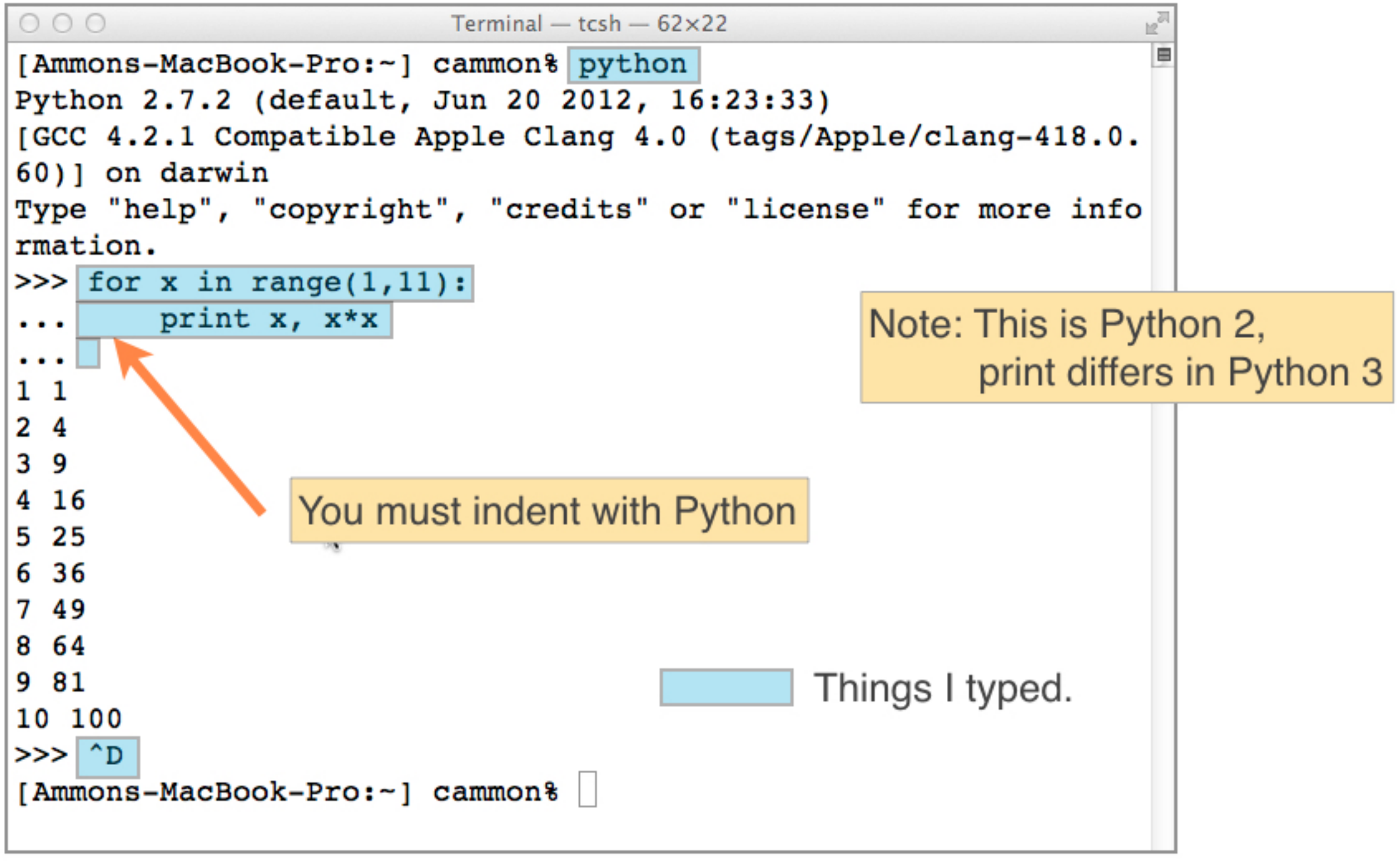
A C Language For Loop

```
for(x=1;x<=10;x++)
{
  printf("%f %f\n", x, x*x);
}
```

A Python for Loop

```
for x in range(1,11):
   print "format(x,"0.0f "),format(x*x,"0.0f\n")
```

Do it. Start up the python interpreter and enter the following commands.



Generic Conditionals

```
if(boolean statement)
{
    do something if statement is true ...
}
else
{
    do something else if statement is false ...
}
```

Example Conditionals

A C Language Conditional

```
if(x < 0 && y > 12.3)
{
  printf("%.4f %.4f\n", x, y)
}
```

A Python Conditional

Note: This is Python 2,
print differs in Python 3

```
if(x < 0 & y > 12.3):
   print "format(x,"0.4f "),format(y,"0.4f\n")
```

Example Conditionals

A Fortran Language Conditional

```
if(x .lt. 0 .and. y .gt. 12.3) then
  write(stdout,'(f10.4,1x,f10.4)') x, y)
end if
```

A Matlab Conditional

```
if(x < 0 && y > 12.3)
    sprintf('%0.4f %0.4f',x,y)
end
```

Libraries & Frameworks

The heart of scientific computation often involves the use of a community-wide library of specialize computation routines

LAPack / ODEPack / Numerical Recipes

You should not write low-level routines, you should use these established libraries.

They are written by experts in the methods, you are a geoscientist.

Libraries & Frameworks

Indeed, many computers come with these standard libraries in the system and especially optimized for the CPU type, etc.

The way that you use them is to call the functions in the library and then “link” your codes to the library when you compile your code.

OS X LAPack

Terminal — tcsh — 74×17

```
[Ammons-MacBook-Pro:/usr/lib] cammon% cd /usr/lib
[Ammons-MacBook-Pro:/usr/lib] cammon% ls *lapack*
libclapack.dylib        libf77lapack.dylib      liblapack.dylib
[Ammons-MacBook-Pro:/usr/lib] cammon% nm liblapack.dylib | grep -i SGESVD
0000000000397b31 T _SGESVD
0000000000397b31 T _SGESVD_
0000000000397b31 T _sgesvd
0000000000397b31 T _sgesvd_
[Ammons-MacBook-Pro:/usr/lib] cammon% gcc -o myCode myCode.c -lclapack
```

Frameworks

Most modern languages come with thousands of pre-written tools to help you accomplish tasks. Some of them are amazing.

So instead of spending your time writing code, you spend your time reading library documentation. It's worth it.

Example: Python's urllib

parseEQList.py

Currently Open Documents
- genData.py
- parseEQList.py

Recent Documents

Last Saved: 10/15/12 8:14:15 PM
File Path : ~/Dropbox/GEOSC497~ABE/Presentations/Programming/parseEQList.py

parseEQList.py | (no symbol selected)

```
1  #!/bin/python
2  import urllib
3  # Get the file from the US Geological Survey.
4  eqData = urllib.urlopen("http://earthquake.usgs.gov/earthquakes/feed/csv/4.5/week")
5  # Store the contents in 'theEQList'.
6  theEQList = eqData.read()
7  # process the event list
8  print theEQList
9  #
10 eqData.close()
11 #
```

Note: This is old, use urllib2

10 | 7 | Python | Unicode (UTF-8) | Unix (LF) | 280 / 35 / 11

Legacy Code

Many of the codes you will use are probably very old. If they work and run fast, just read the code so you know what it does and use them.

If they take a long time to run, check the age of the libraries that they are using. Old libraries can be slow.

Your advisor's version of LAPack may be optimized for defunct hardware...

Programming Tools

A number of tools have their own scientific interpretive languages. For computation and graphics, some common choices are

Matlab / Octave

Mathematica / Maple

For text formatting

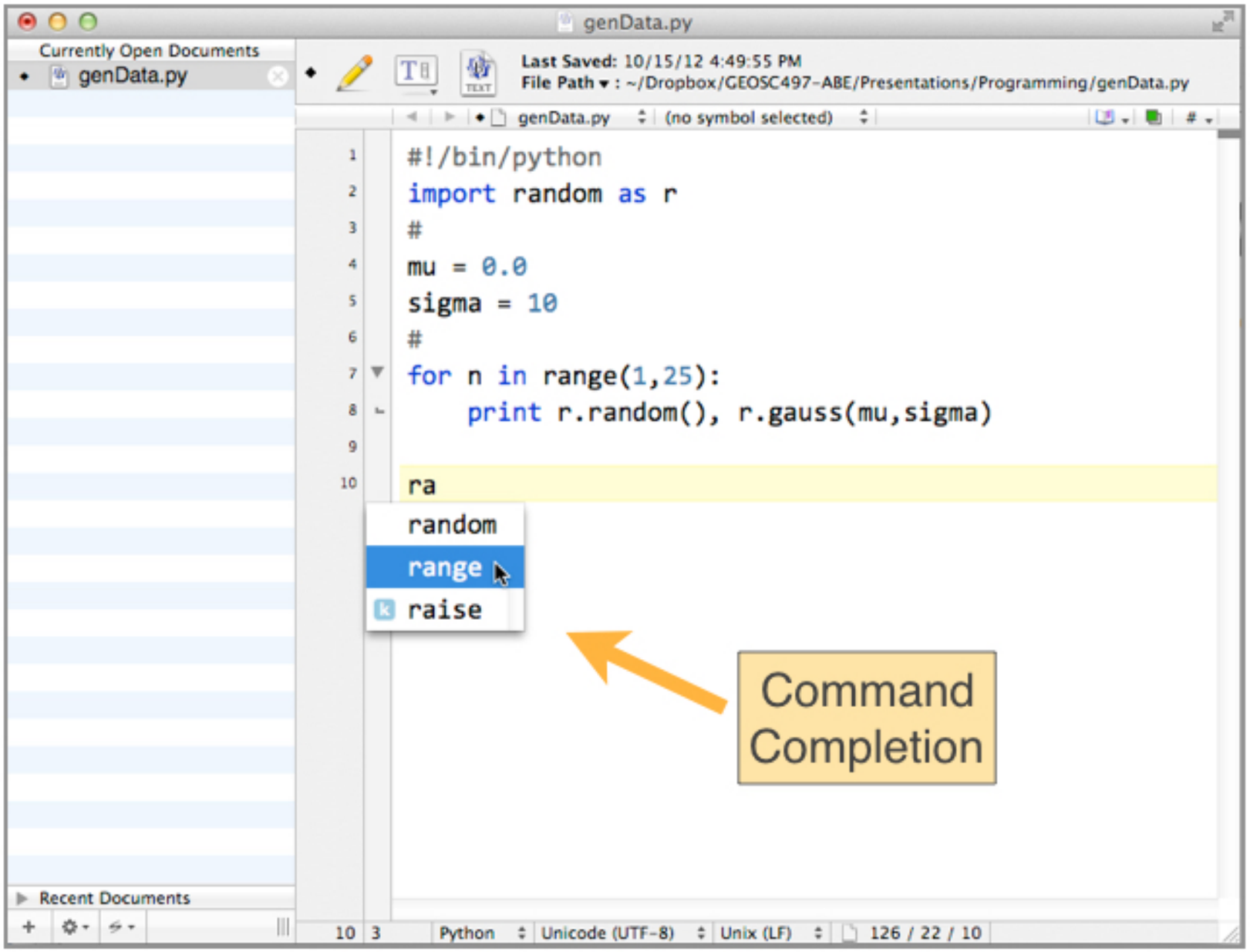
LaTeX.

A Programmer's Editor

At a minimum, you have to use a good programming aware editor (not pico).

Try

http://atom.io



Integrated-Development Environments (IDE's)

raylDisp.xcodeproj — main.c

Build raylDisp: **Succeeded** | 8/20/12 at 5:29 PM

Run Stop raylDisp 〉My Mac 64-bit Scheme Breakpoints Editor View Organizer

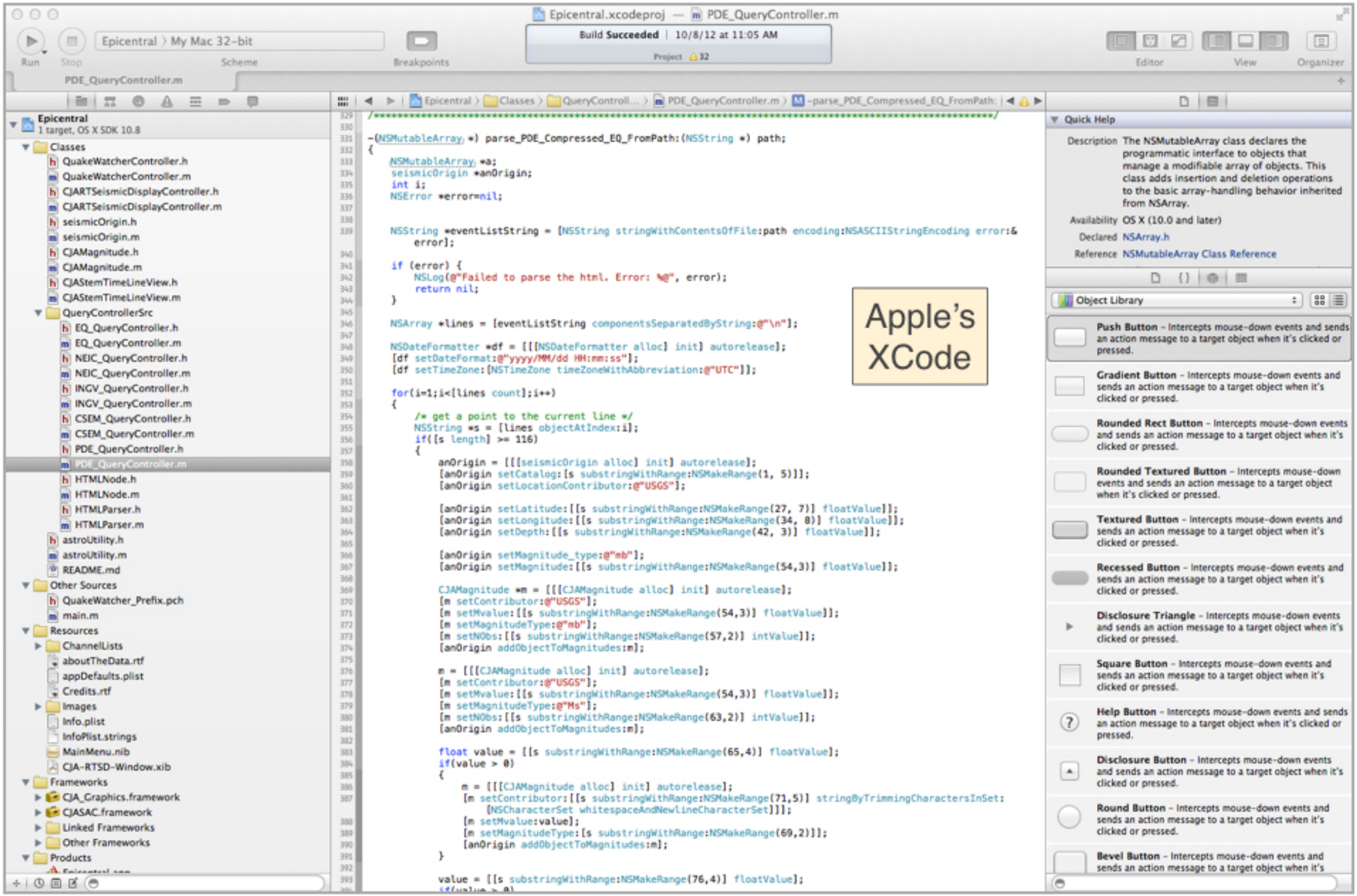
raylDisp 〉raylDisp 〉main.c 〉IntegrateLoveEigenFtns()

raylDisp
1 target, OS X SDK 10.8
- raylDisp
  - main.c
  - raylDisp.1
- Products
  - raylDisp

```
         Compute the integrals of the eigenfunctions.
 */
/*****************************************************************************************************/

void IntegrateRaylEigenFtns(DOUBLE kvalue, DOUBLE *r1,DOUBLE *r2,DOUBLE *r3,DOUBLE *r4, DOUBLE *integrals)
{
    DOUBLE z;
    DOUBLE drdz[4],r[4];
    int nSteps, i;
    DOUBLE vp,vs,rho,lambda,mu;
    DOUBLE potentialEnergy, kineticEnergy;

    /* zStart and zFinish are globals */
    nSteps = (zStart-zFinish)/dz+1;

    integrals[0] = 0.0;
    integrals[1] = 0.0;
    integrals[2] = 0.0;
    integrals[3] = 0.0;

    /* store the value of the wavenumber in the global variable k */
    k = kvalue;
    /* this is a very crude integration approach */
    z = zStart;
    for(i=0;i<nSteps;i++)
    {
        getModelParametersForDepth(z, &vp, &vs, &rho, &lambda, &mu);

        /* extract the eigenfunctions and compute their derivatives at this depth */
        r[0] = r1[i];
        r[1] = r2[i];
        r[2] = r3[i];
        r[3] = r4[i];

        raylDerivatives(z,r,drdz);

        /* compute the sums */
        integrals[0] += rho*(r[0]*r[0] + r[1]*r[1]);

        integrals[1] += (lambda+2.0*mu)*r[0]*r[0] + mu*r[1]*r[1];

        integrals[2] += lambda*r[0]*drdz[1] - mu*r[1]*drdz[0];

        integrals[3] += (lambda+2.0*mu)*drdz[1]*drdz[1] + mu*drdz[0]*drdz[0];

        z = z - dz;
    }

    integrals[0] *= dz*0.5;
    integrals[1] *= dz*0.5;

    integrals[2] *= dz;
    integrals[3] *= dz*0.5;
```

Apple's XCode

Integrated-Development Environments (IDE's)



Apple's XCode

Object-Oriented Programming

Program Types

We can divide programs into two types

1) Procedural

2) Object Oriented

Fortran and C are procedural - action proceeds line by line.

C++/Python/Java programs can do the same, but they also allow you to create objects & then pass messages back and forth to them.

Why Object-Oriented?

Object-oriented programming evolved from a desire to create source code that was flexible and easy to maintain.

As programs grow in length the modification or maintenance of the code becomes a substantial problem.

Today, many languages, libraries & frameworks are object-oriented, so this approach is quite common.

Classes & Objects

The heart of object-oriented programming are classes, which are software tools that encapsulate the idea of objects and include instance variables & methods (functions).

We write the code for classes, and we use that code to create instances of those classes that we call objects.

We create objects in our codes and send messages to those objects to do things.

MATLAB Handle Graphics

If you have used handle graphics in Matlab, the idea is similar. You get a handle to the axis object and then you send the axis object messages

```
h = gca();
set(h,'FontName','Helvetica');
set(h,'FontSize', 18);
%
set(h,'XMinorTick','on');
set(h,'YMinorTick','on');
```

An Example Python Class

geoLocation.py

Currently Open Documents

geoLocation.py

Last Saved: 10/21/12 7:56:30 PM
File Path ▾ : ~/Dropbox/GEOSC497-ABE/Presentations/Programming/ABE-Programming02-Files/geoLocation.py

```
#
# This class has three instance variables
#   lat -> latitude in decimal degrees (+N)
#   lon -> longitude in decimal degrees (+E)
#   depth -> depth in km
class GeoLocation:
    #
    def __init__(self):
        "Initializing Location"
        # These are default values
        self.lat = 38.6362
        self.lon = -90.2362
        self.depth = 0
    #
    def set_lat_lon_depth(self,lat,lon,depth=0):
        self.lat = lat
        self.lon = lon
        self.depth = depth
    #
    def description(self):
        #
        return "Location:    %.3fN %.3fE %.2f km " % \
        (self.lat, self.lon, self.depth)
    #
    def radius(self):
        #
        return 6371-self.depth
    # this is executed when print object is
    def __str__(self):
        return self.description()
```

Declare a class

Initialization method

Changing instance variables method

Display variable values method

Perform a calculation method

Special Python method for print statement

Recent Documents

geoLocation.py

35 8 Python Unicode (UTF-8) Unix (LF) 948 / 113 / 44

An Example Python Class

Terminal — Python — 80×24

```
> python
Python 2.7.2 (default, Jun 20 2012, 16:23:33)
[GCC 4.2.1 Compatible Apple Clang 4.0 (tags/Apple/clang-418.0.60)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from geoLocation import *
>>> pt = GeoLocation()
>>> print pt
Location:    0.000N 0.000E 0.00 km
>>> pt.set_lat_lon_depth(40.8,-78.0,0)
>>> print pt
Location:    40.800N -78.000E 0.00 km
>>> print pt.radius()
6371
>>>
```

Using the python interpreter with the file geoLocation.py.

Note: This is Python 2, print differs in Python 3

Foundation Classes

Most operating systems and languages come with large class frameworks that ease the task of programming.

I recommend that in any system, you become familiar with

1. String classes
2. Container classes (lists, arrays, dictionaries)
3. File classes

The Python String Class

```
In [4]: import string

In [5]: s = "this is my first test string; it's pretty simple."

In [6]: s.capitalize()
Out[6]: "This is my first test string; it's pretty simple."

In [7]: s.lower()
Out[7]: "this is my first test string; it's pretty simple."

In [8]: s.upper()
Out[8]: "THIS IS MY FIRST TEST STRING; IT'S PRETTY SIMPLE."

In [9]: s.lower()
Out[9]: "this is my first test string; it's pretty simple."

In [10]: s.title()
Out[10]: "This Is My First Test String; It'S Pretty Simple."

In [12]: s.lower()
Out[12]: "this is my first test string; it's pretty simple."
```

Create a string.

Send it messages.

The Python String Class

```
In [18]: s
Out[18]: "this is my first test string; it's pretty simple."

In [14]: s.split()
Out[14]: ['this', 'is', 'my', 'first', 'test', 'string;', "it's", 'pretty',
'simple.']

In [15]: s.split(";")
Out[15]: ['this is my first test string', " it's pretty simple."]

In [16]: s.find("pretty")
Out[16]: 35

In [17]: s[35:41]
Out[17]: 'pretty'
```

Python Lists

In scientific fortran and C, arrays store numbers (integers, floats, complex numbers, etc.).

In object-oriented programs, arrays hold objects (actually, they usually store pointers to objects).

In python, arrays are called lists (an array in python is a list containing objects of the same class).

Python Lists

```
In [28]: a = ["red", "green", "blue", "red"]
In [38]: print a[0]
red

In [29]: a.count("red")
Out[29]: 2

In [30]: a.sort()
In [31]: print a
['blue', 'green', 'red', 'red']

In [34]: a.reverse()
In [35]: print a
['red', 'red', 'green', 'blue']

In [36]: a.insert(1,"yellow")
In [37]: print a
['red', 'yellow', 'red', 'green', 'blue']
```

Note: This is Python 2,
print differs in Python 3

Python Lists

Note that the strings in the array are objects and we can send them messages.

```
In [28]: a = ["red", "green", "blue", "red"]
In [42]: a[0].upper()
Out[42]: 'RED'

In [44]: b = [x.upper() for x in a]
In [45]: print b
['RED', 'YELLOW', 'RED', 'GREEN', 'BLUE']

In [46]: b = [x.title() for x in a]; print b
['Red', 'Yellow', 'Red', 'Green', 'Blue']
```

Note: This is Python 2, print differs in Python 3

Python Dictionaries

Dictionaries are container classes that allow you to associate a set of “keys” with a set of “values”. This makes a convenient way to access the values (and very readable code).

The first line defines a dictionary with three key-value pairs.

```
In [49]: colors = {"red":[255,0,0],"green":[0,255,0],"blue":[0,0,255]}
In [53]: print colors
{'blue': [0, 0, 255], 'green': [0, 255, 0], 'red': [255, 0, 0]}

In [54]: rgb = colors["green"]
In [55]: print rgb
[0, 255, 0]
```

Now you “index” a value with the key.

Note: This is Python 2,
print differs in Python 3

Python Files

The python function “open” returns a file object that allows you to query and read a file, or write a new one.

```
In [61]: f = open("colorValues.txt",'r')

In [62]: lines = f.readlines()
In [63]: f.close()

In [64]: print lines[0]
maroon #800000 128,0,0
In [65]: print lines[1]
dark red #8B0000  139,0,0

In [67]: a = lines[0].split()
In [68]: print a[0], a[2]
maroon 128,0,0

In [69]: a[2].replace(',','/')
Out[69]: '128/0/0'
```

Note: This is Python 2, print differs in Python 3

Python File Handling

Python provides powerful tools for exploring files and directories. You can compare files (filecmp module) match filenames (fnmatch module), etc.

The “os” module provides some convenient access to the file system (and other items like environment variables, etc).

File Matching Example

findFrameworks.py

Currently Open Documents

findFrameworks....

Last Saved: 10/22/12 8:53:16 PM

File Path : ~/Dropbox/GEOSC497-ABE/Presentations/Programming/ABE-Programming02-Files/findFrameworks.py

findFrameworks.py | (no symbol selected)

```
1  import os,fnmatch
2
3  def findFiles(topdirectory,pattern):
4      for path, files, dirs in os.walk(topdirectory):
5          for name in files:
6              if fnmatch.fnmatch(name,pattern):
7                  yield os.path.join(path,name)
8
9  #
10 for pyfile in findFiles("/System/Library/Frameworks/","*.framework"):
11     print pyfile
```

Recent Documents

11 17 Python Unicode (UTF-8) Unix (LF) 283 / 36 / 11

Note: This is Python 2,
print differs in Python 3

This is a powerful way to process many files at once.

Google for thousands of python examples, and visit

http://python.org

for documentation and more details.

Activities & Homework