

Scientific Computing: Lecture 1

- Introduction to course, syllabus, software
- Getting started
 - Enthought Canopy, TextWrangler editor, python environment, ipython, unix shell
- Data structures in Python
 - Integers, floats, strings, lists, tuples, dictionaries, functions

CLASS NOTES

- ✘ Take a look at course website ASAP. Download class codes.
- ✘ Read Chap. 1 by Tuesday
- ✘ See me if you have concerns or questions.

Course information

- Instructor: Dr. Gladden, VCRSP & Assoc. Prof. of Physics
- Office: Lyceum 313
- Email: jgladden@olemiss.edu
- Website: www.phy.olemiss.edu/~jgladden/sci_comp/
(for example code, lecture slides, assignments)
- Syllabus and course description.
- Office hours: Please schedule with Ms. Sarah Krueger
(skrueger@olemiss.edu)
- Textbook:
A Primer on Scientific Programming with Python
5th edition, Hans Petter Langtangen
This is a good resource which we will refer to regularly, but we will not be working through it chapter by chapter.

Course Goals

- The goal of this course is to provide you with both a general understanding of fundamental concepts in scientific computing and to teach you the skills to implement them to solve problems in your research.
- Scientific computing is a HUGE field with many specialized niches. We will focus on the fundamental concepts which form the basis for these specializations.
- We will be using the programming (scripting) language Python which has gained popularity in the scientific community (and many other areas!)
 - Scripted rather than compiled
 - Cross platform, even for GUIs (windows, menus, mouse, ...)
 - Easy to learn (as you will see)
 - Similar syntax as Matlab, but more flexible
 - Large and mature code base (libraries)
- This, however, is NOT intended to be a course in Python!

Course tools

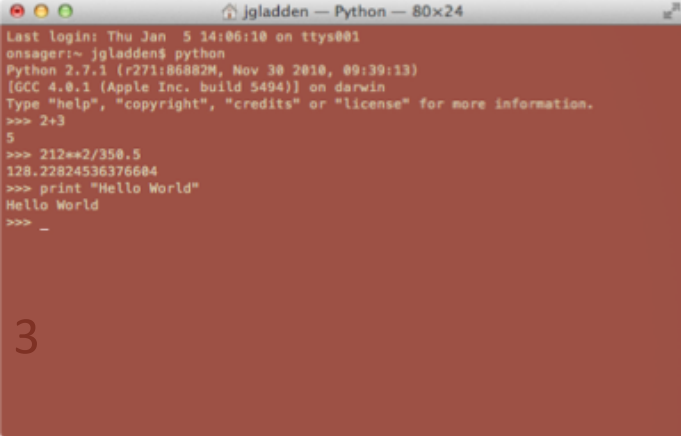
- All software we will use in class is freely available. Links to downloads are on the Resources page of the Course website.
- Python:
 - Recent major version change (2.7 to 3.4). For compatibility we will use 2.7.
 - Already installed on MacOSX and Linux. Can be downloaded for Windows.
 - Another option is a "kitchen sink" python distribution with most scientific libraries already included. A popular one is Enthought Canopy (free for basic subscription)
 - Programs are just text files. Want a code editor that has "syntax highlighting" for python. Textwrangler is good on Mac, Notepad++ is common for Windows. Many options -> Google! Canopy has a decent built in editor.
- Saving your class work. Several options here:
 - You are encouraged to bring your own laptop!
 - Some classroom computers may be available.
 - A USB thumb drive (>2Gb) is probably easiest.
 - You can save files to a scratch directory on the desktop and upload them to a network drive (Box, Google Drive, DropBox, ftp server, ...)
 - Do NOT expect all your work to be safe on these computers!

Topics for the semester

- Introduction to Python (4 weeks)
 - Data structures, flow control, conditionals, input/output, functions
 - Graphical representation of data
 - Plots with “matplotlib”, “publication quality”, multiple data sets and visualization.
 - Object oriented programming
- Linear and non-linear regression (1 week)
- Numeric differentiation and Symbolic Mathematics (2 weeks)
 - Numeric precision and discretization error issues & Sympy
- Numeric integration and systems of ODEs (2 weeks)
- Partial Differential Equations (1 week) – for graduate students only
- Roots of polynomials and other functions (1 week)
- Matrix algebra and manipulation (1 week)
- Parallel computing (1 week)
- Graphical User Interfaces (GUIs) [or other!] (1 week)

Getting started with python

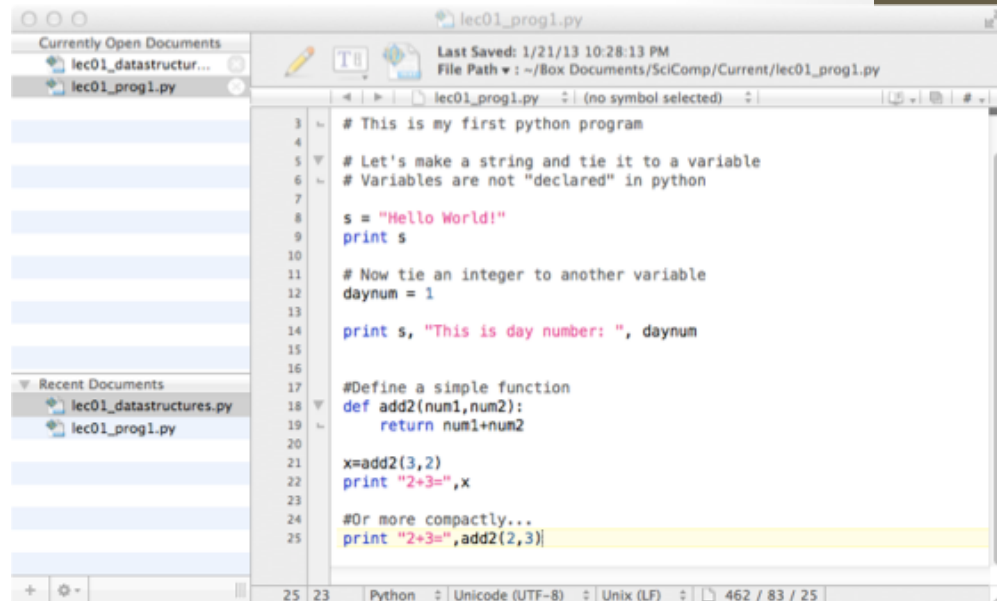
- Open a terminal window (Applications -> Utilities -> Terminal)
- Type `python`
- Type `2+3` and press enter
- Type `212**2/350.5`
- Type `print "Hello World"`
 - `print("Hello World")` in Python 3
- Press Control-D (or `exit()`) to exit.
- Now repeat the above commands in ipython.
- These are called the “interpreters” which execute commands line by line. Useful for debugging or quick and dirty calculations you don’t really need to save.
- Real work is done by typing these commands into a text file (with “.py” extension and then having the interpreter execute the commands line by line (or block by block)

A screenshot of a terminal window titled "jgladden - Python - 80x24". The terminal shows the following text:

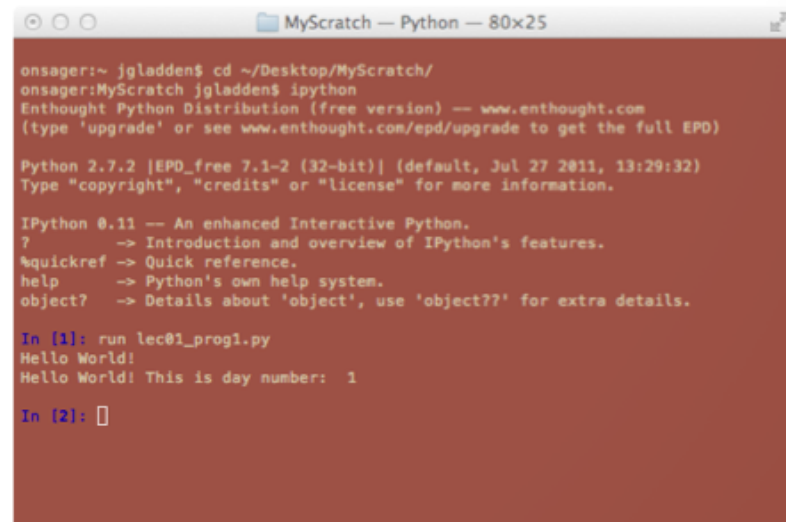
```
Last login: Thu Jan 5 14:06:10 on ttys001
jgladden:~ jgladden$ python
Python 2.7.1 (r271:8682M, Nov 30 2010, 09:39:13)
[GCC 4.0.1 (Apple Inc. build 5494)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 2+3
5
>>> 212**2/350.5
128.22824536376604
>>> print "Hello World"
Hello World
>>> _
```

Writing and running a program

- Create a directory “MyScratch” on the Desktop.
- Open up TextWrangler, type the following commands in a new window.
- Save the file as “lec01_prog1.py” in your new directory.
- In your terminal window, type `cd ~/Desktop/MyScratch/` then `ipython`.
- Now execute the program with `run lec01_prog1.py`
- Note the syntax highlighting and other features of Textwrangler and Canopy.
- Try same program in regular python interpreter with `python lec01_prog1.py`.



```
3 # This is my first python program
4
5 # Let's make a string and tie it to a variable
6 # Variables are not "declared" in python
7
8 s = "Hello World!"
9 print s
10
11 # Now tie an integer to another variable
12 daynum = 1
13
14 print s, "This is day number: ", daynum
15
16
17 #Define a simple function
18 def add2(num1,num2):
19     return num1+num2
20
21 x=add2(3,2)
22 print "2+3=",x
23
24 #Or more compactly...
25 print "2+3=",add2(2,3]
```



```
onsager:~ jgladden$ cd ~/Desktop/MyScratch/
onsager:MyScratch jgladden$ ipython
Enthought Python Distribution (free version) -- www.enthought.com
(type 'upgrade' or see www.enthought.com/epd/upgrade to get the full EPD)

Python 2.7.2 [EPD_free 7.1-2 (32-bit)] (default, Jul 27 2011, 13:29:32)
Type "copyright", "credits" or "license" for more information.

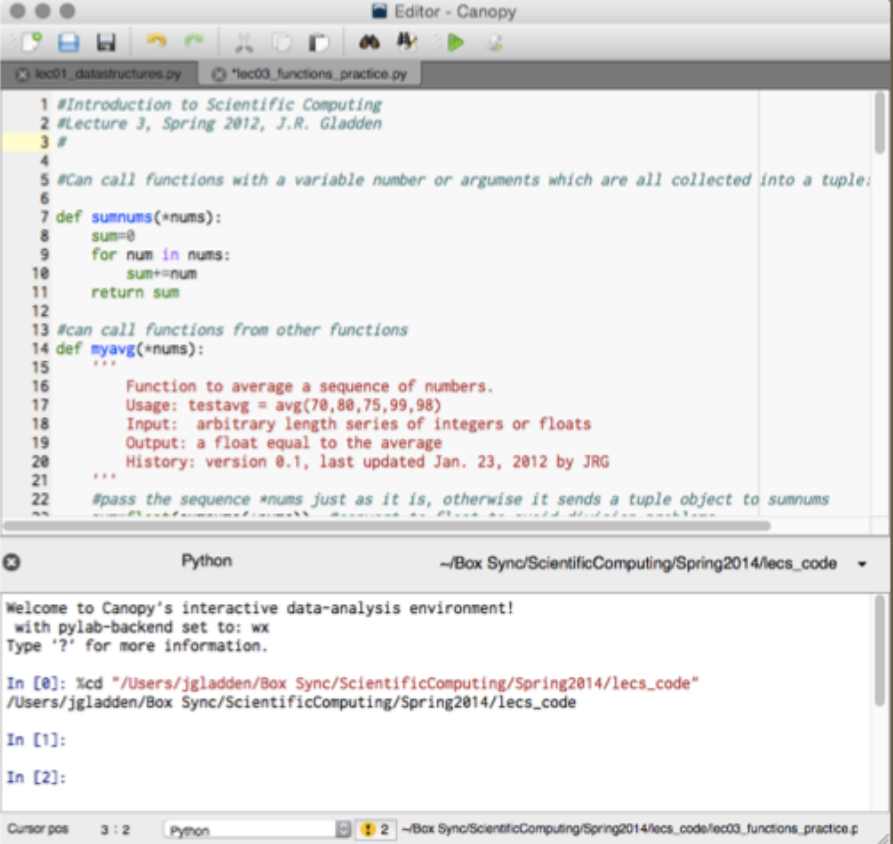
IPython 0.11 -- An enhanced Interactive Python.
?      -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help   -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.

In [1]: run lec01_prog1.py
Hello World!
Hello World! This is day number: 1

In [2]: ]
```

Enthought Canopy

- Canopy is a python integrated development environment for scientists.
- Commercial, but freely available to academics. Need to create an account.
- Convenient editor and interpreter and module management (installation and updates)
- Not the best editor and can crash on more complex programs.
- <https://www.enthought.com/product/canopy/>



The screenshot displays the Enthought Canopy IDE interface. The top window, titled 'Editor - Canopy', shows a Python script with the following content:

```
1 #Introduction to Scientific Computing
2 #Lecture 3, Spring 2012, J.R. Gladden
3 #
4
5 #Can call functions with a variable number of arguments which are all collected into a tuple:
6
7 def sumnums(*nums):
8     sum=0
9     for num in nums:
10        sum+=num
11    return sum
12
13 #can call functions from other functions
14 def myavg(*nums):
15     ...
16     Function to average a sequence of numbers.
17     Usage: testavg = avg(70,80,75,99,98)
18     Input: arbitrary length series of integers or floats
19     Output: a float equal to the average
20     History: version 0.1, last updated Jan. 23, 2012 by JRG
21     ...
22 #pass the sequence *nums just as it is, otherwise it sends a tuple object to sumnums
23
```

The bottom window, titled 'Python', shows the interactive console output:

```
Welcome to Canopy's interactive data-analysis environment!
with pylab-backend set to: wx
Type '?' for more information.

In [0]: %cd "/Users/jgladden/Box Sync/ScientificComputing/Spring2014/lecs_code"
/Users/jgladden/Box Sync/ScientificComputing/Spring2014/lecs_code

In [1]:
In [2]:
```


Python Notebooks

- Interesting option for interactive work, especially for display to an audience.
- Similar to Mathematica Notebooks – mix of code and descriptive text.
- Run through a browser!
- "Sessions" with formatted text, graphics, media, and live code can be easily shared.
- Sessions can be run on remote machines through the browser
 - Code is actually executed in the remote machine which could be a large parallel machine.

Data structures in python

- All programming languages have various types of data structures.
- Common to all are:
 - Strings: `s="Hello World"`, or `s='Hello World'`
 - Integers (no decimal): `n=10`
 - Floating point numbers or floats (decimal): `mass=10.0`
- Specific to python are:
 - Lists: `grades = [95, "Bobby", 97.5, "Sue", 82, "Sarah"]`
 - Tuples: `temperatures = (22.3, 23.1, 24.3)`
 - Dictionaries:
`rgbcolors={ 'red' : (1, 0, 0), 'green' : (0, 1, 0), 'blue' : (0, 0, 1) }`
- These are all “built in” data types in all python distributions
- Code examples ...

Modules (Libraries)

- Much of python's power comes from external modules or libraries. There are literally thousands of them and counting.
- Some come with every python distribution (the standard library), some must be downloaded and installed.
- They are loaded with the `import` statement in a variety of ways.
 - `import math` or `import math as m`,
 - `from math import *` or `from math import sin`
 - Note: Trig functions are RADIAN based (not degrees).
- Each library includes a bunch of functions that will be available after importing.
- Code examples

Functions

- Functions are a way of performing a specific task that will need to be done repeatedly.
- Starts with `def` keyword.
- NOTE: indentation IS important in python! Each block of code must have the same indentation.
- Rule of thumb
 - After each “:”, indentation must increase.
- Functions must be defined before they are called.
- Generally, define all functions in top of program OR in a separate file (called in with an `import` statement) if there are many functions.

```
def functname(arg1, arg2, ...):  
    # python statements  
    # to process arguments  
    return result
```

```
#Example: add 2 numbers  
def add2(num1, num2):  
    sum=num1+num2  
    return sum
```

```
myresult = add2(3,4)  
#myresult now has value of 7
```