Course Introduction

Computational Physics

Course Introduction
Outline

- Course Description
- Course Information
  - Instructors
  - Web Page
  - Schedule
- Getting Started with Computers
- Getting Started with Python
Course Description

- Introduction to computational methods for simulating physical systems and solving problems.
- Sophomore Level
- No previous experience with computational methods
- "Lab Course"
Course Description

The Learning Curve

• We need to learn many new things...
  • Dealing with Lab Computers
    • Unix operating system
    • X Windows System
    • Text Editors for X Windows
    • Report Writing (e.g. with Open Office)
  • Using Computer Language: Python
    • Python's user interface
    • Arrays
    • Objects
    • Interactive, Scripts, Functions
    • Plotting
Course Description
The Learning Curve

- Once we learn these things we will learn to:
  - Simulate Physical Systems
  - Solve Ordinary Differential Equations
  - Solve Partial Differential Equations

- Examples of Physics ...
  - Motion of projectiles with air resistance
  - Motion of pendulum and Chaos
  - Orbits in the Solar System
  - Diffusion
Course Information

Instructors

- Pete Schloerb (Instructor)
  - 847 LGRT-B - 545 4303
  - Email: schloerb @ astro.umass.edu
- Teaching Assistant: Yuping Tang
  - 619F LGRT-B
  - Email: yupingt @ astro.umass.edu
- Computer Lab Supervisor: Tariq Ahmad
  - Report Problems to: rt @ cs.astro.umass.edu
Textbook

- No official textbook – we teach what you need to know.
- Google is a vast resource for python and for numerical techniques
- If you want a book, try:
  - Computational Physics: Problem Solving with Python, 3rd Edition
  - Rubin H. Landau, Manuel J. Páez, Cristian C. Bordeianu
Course Information

Web Pages

www.astro.umass.edu/~schloerb/ph281

This is subject to change.
What do we do?

- Exercises (every class)
  - Short problems; practice
- Projects (every few weeks)
  - Calculation of a real Physics Problem
  - Explore behavior
  - Submit Lab Report
- Final Problem
A Word to the Wise

- Attend Class
  - There is not always a lecture, but we often review matters that have been troubling
  - Instructors are here to help
- Don't fall behind
  - Exercises and Projects take longer than you think.
- Learn Python!
  - Spend some time with tutorials and manual.
Course Information

Tentative Schedule

- Python!!
- Interpolation; Numerical Integration
- Random Numbers and Simulation
- Model Fitting
- Differential Equations
  - Ordinary
  - Partial
Getting Started with Computers
General Lab Guidelines

- See document on desktop for rules.
- Lab open 24/7; use combination lock.
- All computers are the same.
- Ubuntu Lucid (12.04); Gnu/Linux
- Report problems to Professor or to rt@cs.astro.umass.edu in off hours
Lab Do's

- Use the lab!
- Respect equipment.
- You are responsible for your own backups.
- Move computers in their slide holders gently.
- Log off when you are done.
- Last one out, please turn off lights and lock door.
Lab Don'ts

- Don't share combination of door lock.
- Don't unplug any cable or power cord.
- No public internet connections in the lab. Don't connect your computer to lab network.
- No food or drink in Lab.
Login/Loginout

- Login Procedure
  - Login with your assigned username
  - First Time:
    - Enter temporary password
    - Change to new password
  - Thereafter ... remember your password!!

- Logout Procedure
  - Click mouse on menubar at top of screen
  - Select “Log Out”
  - Select “Log Out” when dialog box appears.
Starting Firefox Web Browser
Starting Terminal Shell
```matlab
% display results in a table that looks nice
title('Falling Ball');
for i=1:n
    fprintf('%5.1f %10.2f\n',t(i),x(i))
end
```
Getting Started with Python
Why Python?

- Widely Used
  - Python is Number 1 S/W tool in Astronomy
  - Lots of programs and libraries (called “packages” written for Python)
- Interactive
  - Use it like a graphing calculator
  - Speeds development of programs
- Graphics integrated with language
- Object Oriented
- Portable (if you are careful)
- Free
How do I Learn Python?

- See python.org for information.
  - NB: There are two major versions of Python: version 2 and version 3. We use version 2 in class.
  - Tutorial at: docs.python.org/2/tutorial/index.html
- Many other tutorials on line. Google “python tutorial” for example.
- Get Python and fool around:
  - www.enthought.com/products/products/canopy/
  - Free for students!!
Python Objects

- Objects can:
  - Have characteristics - Attributes
  - Do stuff - Methods
- Think of a cat....
  - Cats have characteristics: breed, fur color, name, etc.
  - Cats do stuff: hunt, eat, sleep, play, purr ...
- The goings on in a real cat are very complicated ... but we don't have to know all that to use a cat.
- S/W objects can be used without deep knowledge of how they work!
Object Inheritance

- Objects can be organized in a hierarchy where more and more specific cases inherit attributes and methods from higher levels:

  - Example:
    Mammal
    Carnivore
    Feline
    Tabby Cat

- Inheritance in S/W allows us to reuse useful parts of objects that might already exist.
Python Variables and Data Structures

- Integer (called int)
- Real (called float – for floating point)
- Complex (called complex)
- Logical (called bool – for boolean)
- String – an array of characters
- List – an array of variables (of any kind)
- Tuples – a kind of list that is immutable
- Dictionaries – a list of “key” and “value” pairs.

*The “type” function identifies the variable.*
Dealing with Python Lists

- Make a list with 5 items:
  - `s = [0, 1, 2, 3, 4]`
  - `g = 'Hello World!'` (a string is a kind of list)
- Get the ith item of the list (index starts at 0):
  - `s[i]`
- Get items i through j of the list:
  - `s[i:j+1]`
- Concatenate 2 lists s and t:
  - `s+t`
- Get length of list
  - `len(s)`
Try this out!

- Start terminal window
- Start “interactive python” in terminal window
  - Type: ipython
- Enter some python commands....
- Exit ipython
  - Type: exit or quit

Note: we could also use the “python” interpreter directly rather than “ipython”, but the latter provides more useful capabilities as we proceed in this class.
Some ipython features

- OS Commands:
  - `mkdir new_dir`: make a new directory “new_dir”
  - `ls`: show contents of current directory
  - `pwd`: show current directory
  - `cd new_dir`: change to directory “new_dir”

- Help:
  - `help(object)`: provides help on “object”
  - `magic`: lists “magic” commands
    - `%quickref` is a magic command which provides a short list of magic commands
  - `object??`: tells about “object”
Calculate Something

* Basic python doesn't know much ...
* We must “import” useful functions from the “math” module:

```python
> from math import sin  # imports the “sin” function
   Call the sin function: sin(theta)

> from math import *  # imports all math functions
   Call the sin function: sin(theta)

> import math  # imports the math module
   Call the sin function: math.sin(theta)

> import math as m  # imports math, calls it “m”
   Call the sin function: m.sin(theta)
```
Write a function

```
In [10]: import math

In [11]: def sind(theta):
   ....:     t = theta/180.*math.pi
   ....:     return(math.sin(t))
   ....:

In [12]: sind(90)
Out[12]: 1.0
```
Write a script

Use Text Editor to create a file with python commands.

In this example: myscript.py

To run the script enter:

```
# file: myscript.py
A = 'Hello World!'
print A
```

```
In [13]: run myscript.py
Out[13]: Hello World!
```
Try some stuff out!

- Define some numbers and operate on them (+,-,*,/)
- Build a long string by concatenating short strings
- Select substrings in your long string using the : operator.
- Write a “degree input” function to compute cosine.
- Write a file with all trig functions ready to accept degree arguments (e.g. sind, cosd, tand, atand, atan2d)