GS-SB-406
Practical Introduction to Programming for Scientists

Steven Ludtke
sludtke@bcm.edu

Lecture 1:
Introduction

http://blake.bcm.edu/IP15

Course Details (Jan 2014)

Meets Monday & Friday, 9 - 10:30 AM, N315

Auditors welcome, but encouraged to register (if permitted)

Graded

- 50% homework/labs, 50% final project
- Grading will be lenient

Homework due before class every Friday by email to TA and me

- Amanda Koire <koire@bcm.edu> & cc: sludtke@bcm.edu

Laptops required on Mondays, good idea on Friday as well

Class lectures will be video-archived (unless I forget)

http://blake.bcm.edu/IP15

There is a special homework for this lecture, due before Friday!

Tentative Syllabus

Jan 5 - Introduction, strings, lists, data types
Jan 9 - Program flow
Jan 12 - More core language features & Lab 1
Jan 16 - Representation of numbers, Reading/writing files
Jan 19 - Holiday, no class
Jan 23 - Import, Exceptions, Genomic data processing, BioPython
Jan 26 - Numerical Processing/Plotting & Lab 2
Jan 30 - Object Oriented Programming introduction
Feb 2 - Programming Examples & Lab 3
Feb 6 - Web Server, HTML, XML, Databases
Feb 9 - GUI Programming & Lab 4
Feb 13 - Image Processing
Feb 16 - Holiday, no class
Feb 23 - “Making” & Lab 5
Feb 27 - Network Programming
Mar ? - presentation of class projects, finals week

Why should you learn how to program ?

Something you can’t find in existing software ?

Make repetitive tasks easier ?

You want to be a Maker ?

What to Expect

If you already know some programming

- Learn Python syntax and libraries

If you are starting from scratch

- Read & Modify existing scripts
- Automate tasks
- Write ‘small’ programs from scratch

8512 computer languages (vs 6909 human)

- Machine Language → Assembly Language
- Four of the first modern languages (50s):
  - FORTRAN (FORmula TRANslator)
  - LISP (LISt Processor)
  - ALGOL
  - COBOL (COmmon Business Oriented Language)
- BASIC (1963 - used in 70s-80s)
- C (1972)
- C++ (1983)
- Perl (1990)
- Python (1991)
- Ruby (1992)
- HTML (1994)
- Java (1995)

Why Python ?

Easy to learn !

Widely used

Many available libraries

Powerful

Scripting for 3rd party software

http://www.99-bottles-of-beer.net/

Python ?

PYTHON OOL- developed by Guido van Rossum, and named after Monty Python. (No one Expects the Inquisition) a simple high-level interpreted language. Combines ideas from ABC, C, Modula-3, and ICON. It bridges the gap between C and shell programming, making it suitable for rapid prototyping or as an extension of C. Rossum wanted to correct some of the ABC problems and keep the best features. At the time, he was working on the AMOEBA distributed OS group, and was looking for a scripting language with a syntax like ABC but with the access to the AMOEBA system calls, so he decided to create a language that was extensible; it is OO and supports packages, modules, classes, user-defined exceptions, a good C interface, dynamic loading of C modules and has no arbritrary restrictions.

www.python.org

Note: Python 3.x is available, but we will use Python 2.x since it is still more widely used

A Few Apps with Python Scripting

Blender	3-D modeler, animation, post production (free)
Gimp	Photoshop-like graphics editor (free)
Chimera	Structural biology visualization (free)
PyMol	Structural biology visualization (free)
OpenOffice	MS Office clone by Sun (free)
Maya	Professional 3-D Modeling and Animation
Poser	3-D modeling of humans
VTK	Visualization Toolkit (Scientific Visualization, free)
Abaqus	Finite element modeling (free)
EMAN2	Cryo-EM Image Processing (free)
Phenix	X-ray crystallography toolkit (free)
SciPy	Wide range of science/math tools in python (free)
BioPython	Bioinformatics toolkit for Python (free)

What Can CPUs Do ?

Store numbers (1 & 0)

Rearrange stored numbers

Math

Simple decisions based on numbers

Communicate

Python

Python is a "high level language"

Data storage

- ‘simple’ types - numbers, characters
- compound types - lists, strings, dictionaries, sets, ...

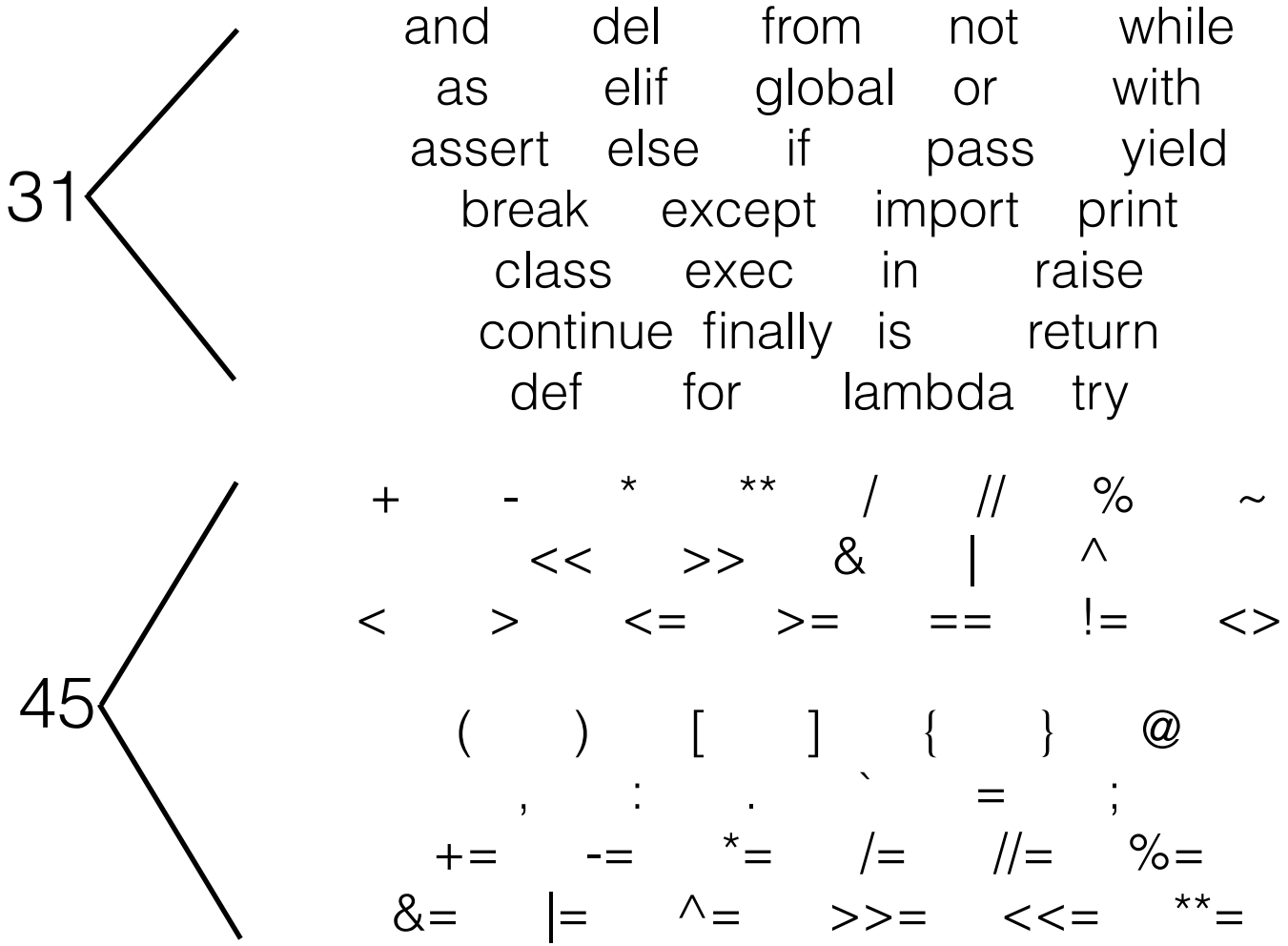
Operate on data

- statements - a=b*10, print b*5+3, if a>5 : a/=2, ...
- functions - sin(a), len(x), ...
- methods (functions on an object) - “abc”.count(“b”)

Interact with the outside world

- User interactions - raw_input()
- Disk and other device access - file i/o
- Networking to other computers

Python Reserved Words



Numbers

integers

- 32-bit (-2,147,483,647 - 2,147,483,648)
- long - effectively unlimited

floating point

- 64-bit (15 significant figs, $<10^{308}$)

complex

- 5.0+3.0j

Strings

'string'

"also a string"

"""This too

but this one can span lines"""

"A"+" test"

"A test"

Lists

```
[item1,item2,item3,...]   # items can be anything
a=[0,1,2,3,4,5,6]         # A list of 7 numbers
a[n]                      # nth element in list
a[n:m]                    # sublist elements n to m-1
a[-n]                     # nth item from the end
a[3] -> 3
a[1:4] -> [1,2,3]
a[-2] -> 5
a[2:-2] -> [2,3,4]
a[2]="x" -> [0,1,"x",3,4,5,6]
tuples:  a=(0,1,2,3,4,5,6)     # tuples are immutable
a[3] -> 3
a[3]=5 -> ERROR!
```

List Methods

append, extend

del, remove

count

index

reverse, sort

Methods of Strings

upper, lower, title, capitalize

count, find, rfind, index

replace

split

regular expressions later...

Sets

Sets have no order and are unique, but can be iterated over

set([1,2,3,4,5])

add, remove, discard, clear

issubset, issuperset

union, intersection, difference

Dictionaries

keys must be immutable, values are arbitrary

{ k1:v1, k2:v2, k3:v3, ... }

Example:

a={ 1:2,2:3,”a”:”b”,2.0:3.2,(1,2):”really?” }

a[1] -> 2

a[(1,2)] -> “really?”

a[2] -> 3.2

Dictionary Methods

has_key

keys

values

items

Some Built-in functions

int, float, str, list, tuple, set, dict - Converts between types

range, xrange - makes a list or iterator covering a range

enumerate

eval

input & raw_input

len

max,min

reversed, sorted

type, isinstance

Installing Python

See my book chapter for additional tips

Mac OSX - Included (strongly suggest MacOS 10.9/10.10)

Linux - Included, but make sure you have 2.7.x

Windows

Download from www.python.org

Run installer

OR you may consider Anaconda from:

http://continuum.io/downloads

Installing ipython

http://ipython.scipy.org

Linux - use your package manager

Mac: if you use fink or macports, use that, otherwise:

sudo easy_install ipython

sudo easy_install readline

Windows:

- Anaconda may be the simplest solution on Windows
- Alternatively, install SetupTools

https://pypi.python.org/pypi/setuptools#windows

then use easy_install

Resources

www.python.org

http://docs.python.org/tutorial/

pypi.python.org

www.scipy.org

Homework 1

(Auditors too!)

There is a survey in the homework section at http://blake.bcm.edu/IP15
Everyone must fill out this form, even if you are informally auditing the class !!!

Install Python 2.7.x and (optionally) ipython

> You should be able to compute 1+1 and get 2 using python on your laptop before next class

Familiarize yourself with the documentation at www.python.org (Python 2.7)