GS-SB-406
Practical Introduction to Programming for Scientists

Steven Ludtke
sludtke@bcm.edu

Lecture 1:
Introduction

http://blake.bcm.edu/IP16

Course Details (Jan 2016)

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

Auditors welcome, but encouraged to register (if permitted)

Graded

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

Homework due before each class by email to TA (cc me)

- James (Micheal) Bell <James.Bell@bcm.edu>, cc: sludtke@bcm.edu

Good idea to bring your laptops to class! Some days required.

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

http://blake.bcm.edu/IP16

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

Syllabus

Jan 4 - Introduction, strings, lists, data types

Jan 8 - Program flow

Jan 11 - More core language features

Jan 15 - Representation of numbers, Reading/writing files

Jan 18 - Holiday, no class

Jan 22 - Import, Exceptions, Genomic data processing, BioPython

Jan 25 - Numerical Processing/Plotting

Jan 29 - Object Oriented Programming introduction

Feb 1 - Programming Examples

Feb 5 - Web Server, HTML, XML, Databases

Feb 8 - GUI Programming

Feb 12 - Image Processing

Feb 15 - Holiday, no class

Feb 19 - Network Programming

Feb 22 - Making + Lab

Feb 26 - TBA

? - presentation of class projects, finals week

TRS-80 Model I

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)

- volatile & nonvolatile storage

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, ...

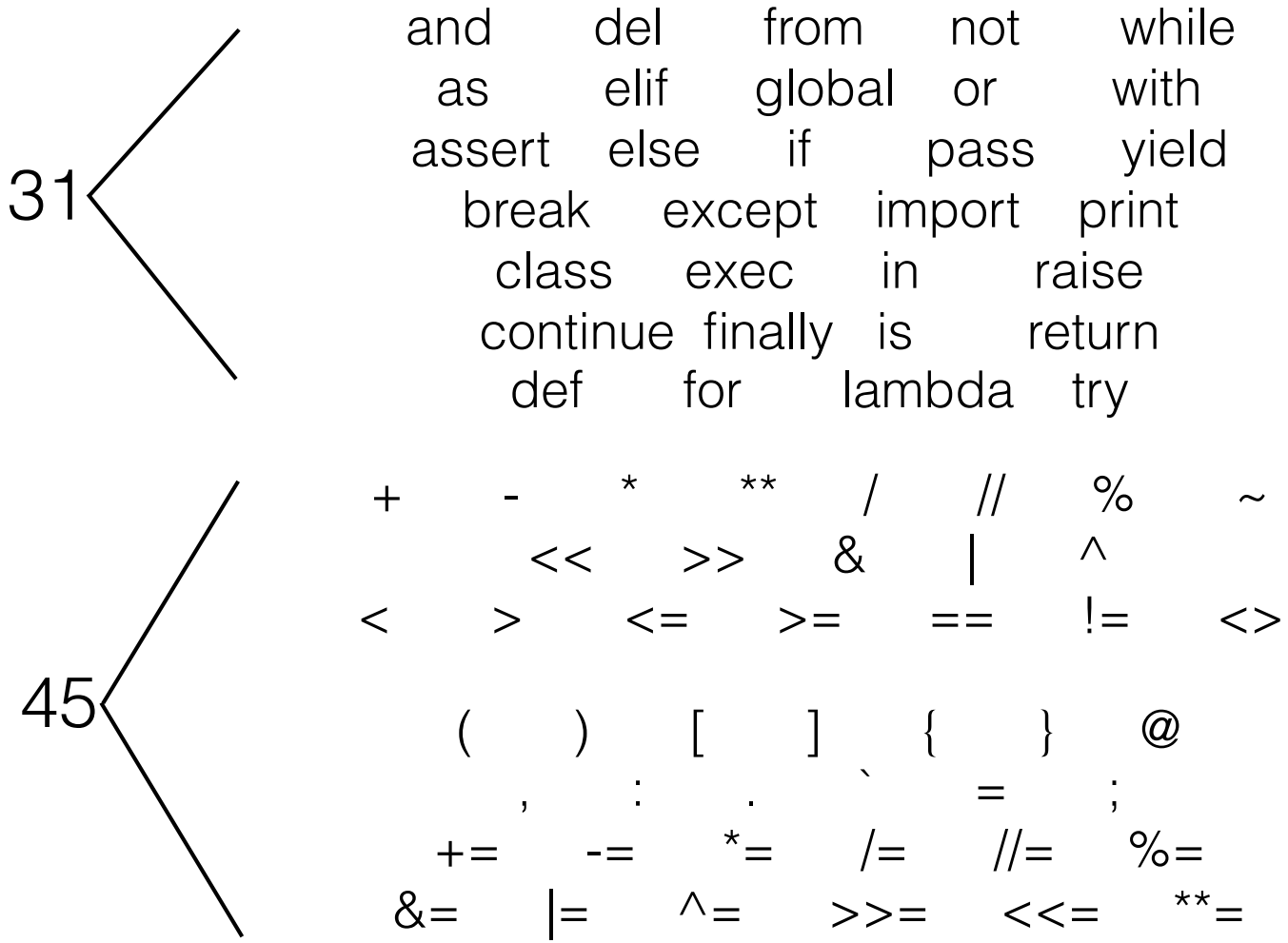
Operate on data

- 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



Digital Representation of Numbers

Bit	0-1	
Byte (char)	(8 bits)	0-255
Word (short)	(16 bits)	0-65,535
Longword (long)	(32 bits)	0-4,294,967,296
Long Longword	(64 bits)	$0\text{-}1.844 \times 10^{19}$
Float	(32 bits)	10^{38} (7 digits)
Double	(64 bits)	10^{308} (15 digits)

Python Numbers

integers

effectively unlimited

floating point

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

complex

5.0+3.0j

Characters - ASCII

0	<NUL>	32	<SPC>	64	@	96	`	128	Ä	160	†	192	¿	224	‡
1	<SOH>	33	!	65	A	97	a	129	Å	161	°	193	¡	225	·
2	<STX>	34	"	66	B	98	b	130	Ç	162	¢	194	¬	226	‚
3	<ETX>	35	#	67	C	99	c	131	É	163	£	195	√	227	„
4	<EOT>	36	$	68	D	100	d	132	Ñ	164	§	196	ƒ	228	‰
5	<ENQ>	37	%	69	E	101	e	133	Ö	165	•	197	≈	229	Â
6	<ACK>	38	&	70	F	102	f	134	Ü	166	¶	198	Δ	230	Ê
7	<BEL>	39	'	71	G	103	g	135	á	167	ß	199	«	231	Á
8	<BS>	40	(	72	H	104	h	136	à	168	®	200	»	232	Ë
9	<TAB>	41	)	73	I	105	i	137	â	169	©	201	…	233	È
10	<LF>	42	*	74	J	106	j	138	ä	170	™	202		234	Í
11	<VT>	43	+	75	K	107	k	139	ã	171	´	203	À	235	Î
12	<FF>	44	,	76	L	108	l	140	å	172	¨	204	Ã	236	Ï
13	<CR>	45	-	77	M	109	m	141	ç	173	≠	205	Õ	237	Ì
14	<SO>	46	.	78	N	110	n	142	é	174	Æ	206	Œ	238	Ó
15	<SI>	47	/	79	O	111	o	143	è	175	Ø	207	œ	239	Ô
16	<DLE>	48	0	80	P	112	p	144	ê	176	∞	208	–	240	
17	<DC1>	49	1	81	Q	113	q	145	ë	177	±	209	—	241	Ò
18	<DC2>	50	2	82	R	114	r	146	í	178	≤	210	“	242	Ú
19	<DC3>	51	3	83	S	115	s	147	ì	179	≥	211	”	243	Û
20	<DC4>	52	4	84	T	116	t	148	î	180	¥	212	‘	244	Ù
21	<NAK>	53	5	85	U	117	u	149	ï	181	µ	213	’	245	ı
22	<SYN	54	6	86	V	118	v	150	ñ	182	∂	214	÷	246	ˆ
23	<ETB>	55	7	87	W	119	w	151	ó	183	∑	215	◊	247	˜
24	<CAN>	56	8	88	X	120	x	152	ò	184	∏	216	ÿ	248	¯
25	<EM>	57	9	89	Y	121	y	153	ô	185	π	217	Ÿ	249	˘
26	<SUB>	58	:	90	Z	122	z	154	ö	186	∫	218	/	250	˙
27	<ESC>	59	;	91	[	123	{	155	õ	187	ª	219	€	251	°
28	<FS>	60	<	92	\	124	|	156	ú	188	º	220	‹	252	¸
29	<GS>	61	=	93	]	125	}	157	ù	189	Ω	221	›	253	˝
30	<RS>	62	>	94	^	126	~	158	û	190	æ	222	fi	254	˛
31	<US>	63	?	95	_	127	<DEL>	159	ü	191	ø	223	fl	255	ˇ

Characters - Unicode

All strings in Python 3 are Unicode!

Over 120,000 different characters

Multiple representations

UTF-8, an 8-bit variable-width encoding which maximizes compatibility with ASCII;

UTF-16, a 16-bit, variable-width encoding;

UTF-32, a 32-bit, fixed-width encoding.

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 - makes an 'iterator' covering a range

enumerate

eval

raw_input

len

max,min

reversed, sorted

type, isinstance

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/IP16
Everyone should fill out this form, even if you are informally auditing the class!!!
If you are taking the class for a grade, doing this will get you a '4'

Install Anaconda - Python 3.5 (https://www.continuum.io/downloads)

Run Spyder (just use the installed 'launcher' for now)

Make sure you can run the following 1 line program in Spyder:

```
print([(i,i**2) for i in range(10)])
```

When you run it, it should produce:

```
[(0, 0), (1, 1), (2, 4), (3, 9), (4, 16), (5, 25), (6, 36), (7, 49), (8, 64), (9, 81)]
```

Familiarize yourself with the organization of the documentation at
www.python.org (Python 3.5)