Introduction to Python for Research Workflows

David A. Lifka, Ph.D.
Cornell Center for Advanced Computing

January 20, 2012

Research Computing Ecosystem

- Desktop Tools
  - Editors
  - Spreadsheets
  - Mathematics & statistical packages
- Modeling & Simulation
  - Parallel programming
    - Multi-process
    - Multi-core
  - Batch scheduling
  - Cloud computing
- Distributed Resources and Collaboration
  - Accessing remote data sources
  - Using remote instrumentation
  - Moving data & programs
- Data Intensive Science

Data Intensive Computing Applications

Modern Research is Producing Massive Amounts of Data

- Microscopes
- Telescopes
- Gene Sequencers
- Mass Spectrometers
- Satellite & Radar Images
- Distributed Weather Sensors
- High Performance Computing (especially HPC Clusters)

Research Communities Rely on Distributed Data Sources

- Collaboration
- Virtual Laboratory's
- Laboratory Information Management Systems (LIMS)

New Management and Usage Issues

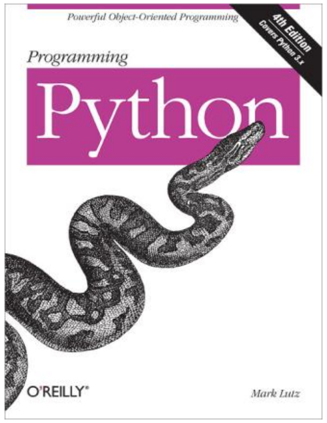
- Security
- Reliability/Availability
- Manageability
- Data Locality – You can't ftp a petabyte to your laptop….

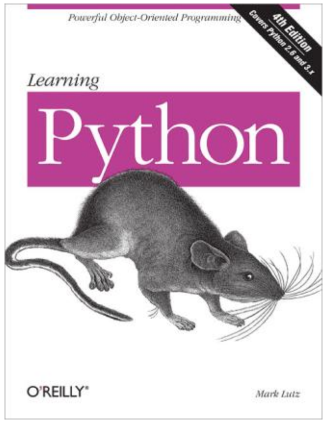
Why Python?

- Fast & easy to learn
- Popular – many researchers use it
- Wealth of open source libraries and examples
- Convenient for rapid prototyping of complex computer tasks
- Great for “gluing together” other programs and tasks into a custom workflow
- Time-saver for repetitive tasks
- Portable (runs on most computing platforms)

Some Recommendations

- Enthought Python
  - http://www.enthought.com/
- O'Reilly
  - http://oreilly.com/python/index.html
- Lifka's course web site
  - http://www.cac.cornell.edu/~lifka/STSCI4060/STSCI4060.htm




Powerful Object-Oriented Programming
4th Edition
Covers Python 2.6 and 3.x
Learning
Python
O'REILLY
Mark Lutz

Hello World!

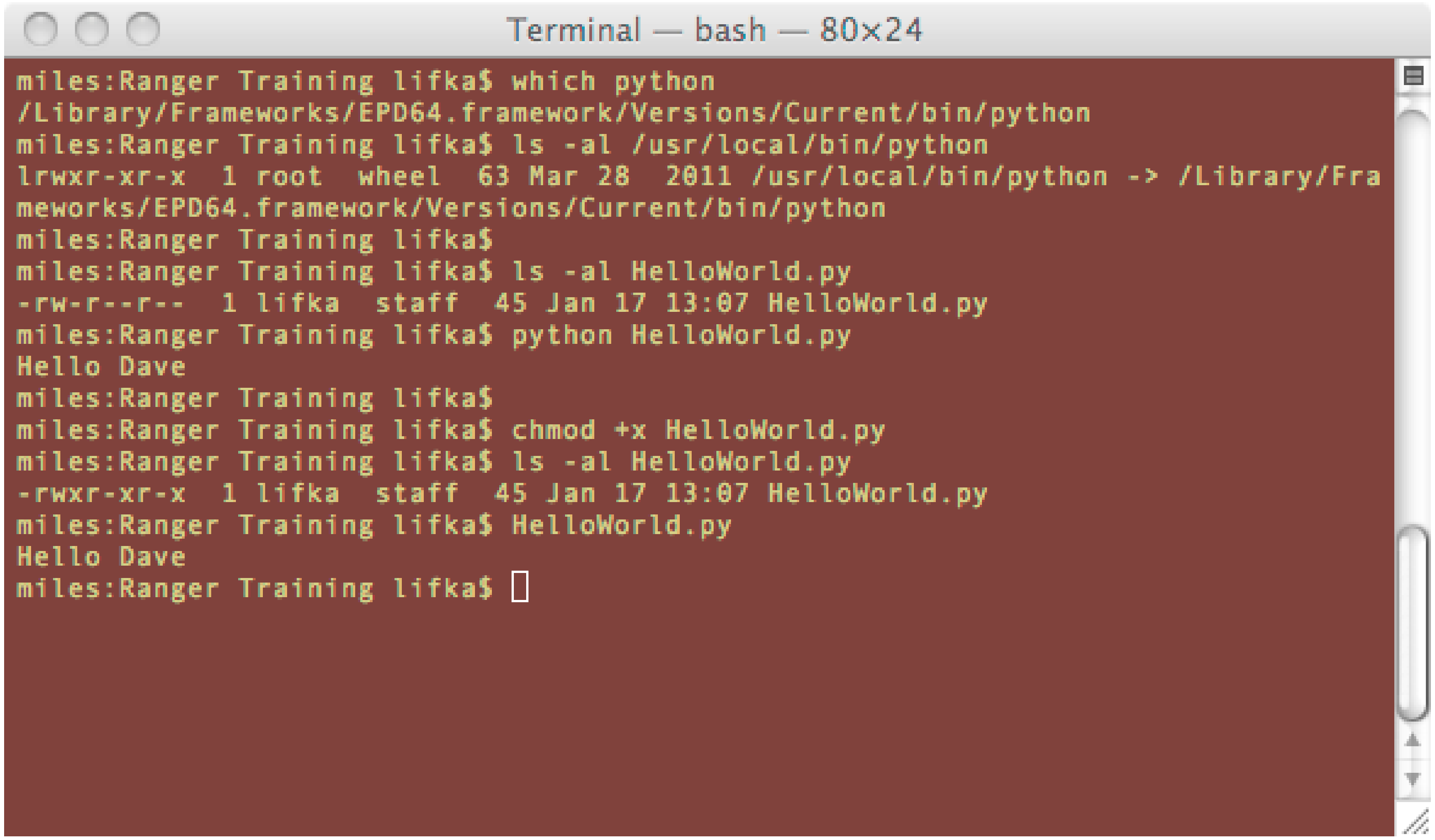
- First line of a script is the path to the shell executable
  - Should be set to the path of the executable on the system the script will be run on.

- This is not needed on Microsoft Windows-based systems, instead, file extensions are "associated" with the correct executable.
  - .py -> Python

- Scripts can be run like standard executables if:
  - On UNIX systems you set appropriate "execute" permissions:
    - `chmod u+x helloworld.pl`
  - On Window-based systems the extensions are properly "associated"

- Scripts can also be run by first invoking the scripting executable and providing a path the script you want it to run:
  - `python ~lifka/scripts/helloworld.py`

Lab 1: Hello World & 2 methods of running python



Terminal — bash — 80×24

```
miles:Ranger Training lifka$ which python
/Library/Frameworks/EPD64.framework/Versions/Current/bin/python
miles:Ranger Training lifka$ ls -al /usr/local/bin/python
lrwxr-xr-x  1 root  wheel  63 Mar 28  2011 /usr/local/bin/python -> /Library/Fra
meworks/EPD64.framework/Versions/Current/bin/python
miles:Ranger Training lifka$
miles:Ranger Training lifka$ ls -al HelloWorld.py
-rw-r--r--  1 lifka  staff  45 Jan 17 13:07 HelloWorld.py
miles:Ranger Training lifka$ python HelloWorld.py
Hello Dave
miles:Ranger Training lifka$
miles:Ranger Training lifka$ chmod +x HelloWorld.py
miles:Ranger Training lifka$ ls -al HelloWorld.py
-rwxr-xr-x  1 lifka  staff  45 Jan 17 13:07 HelloWorld.py
miles:Ranger Training lifka$ HelloWorld.py
Hello Dave
miles:Ranger Training lifka$ 
```

What We'll Cover Today

- Data types & associated operators
- Interactive input
- Lists & dictionaries
- Logic & looping structures
- Reading & writing files
- Running & interacting with applications outside a Python script
- Using FTP from Python

Numeric Data Types

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/example1.py

```
a = 2
b = 3
c = a + b
print "a + b = ", c
print a, "+", b, "=",c
print "Binary:", bin(a), "+", bin(b), "=", bin(c)
print "Octal:", oct(a), "+", oct(b), "=", oct(c)
print "Hexadecimal:", hex(a), "+", hex(b), "=", hex(c)
# using octal formatting
print "%03o + %03o = %03o" % (a, b, c)
# using hexadecimal formatting
print "%03x + %03x = %03x" % (a, b, c)
print "Complex numbers:"
a = 3
b = 4j
c = a + b
print "real =", c.real, "imaginary =", c.imag, "cartesian style =", a, "+", b
a = 3.14159
b = 2
c = a * b
print "a * b = ", c
```

Numeric Operators

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/example2.py

```
print "Add:", a, "+", b, "=",a + b
print "Subtract:", a, "-", b, "=",a - b
print "Divide:", a, "/", b, "=",a / b
print "Multiply:", a, "*", b, "=",a * b
print "Exponent:", a, "**", b, "=",a ** b
print "Modulus:", a, "%", b, "=",a % b
print a, "+= 10 =",
a += 10
print a
print a, "-= 10 =",
a -= 10
print a, "*= 10 =",
a *= 10
print a
print a, "/= 10 =",
a /= 10
print a
print a, "%= 10 =",
a %= 10
print a
```

Numeric Comparisons

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/example3.py

```
a = 3
b = 3
c = 2
if (a == b): print a, "==", b
if (a != c): print a, "!=", c
if (a > c): print  a, ">", c
if (c < a): print  c, "<", a
if (a >= c): print a, ">=", c
if (a >= b): print a, ">=", b
if (c <= a): print c, "<=", a
if (a <= b): print a, "<=", b
```

Interactive Input

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/example4.py

```
import sys

print “hit enter to continue”
wait = sys.stdin.readline()

print “enter an integer:”,
a = int(sys.stdin.readline())
print “Integer: “, a

print “enter a float:”,
b = float(sys.stdin.readline())
print “Float: “, b
```

Strings

```
firstname = "David”
lastname = "Lifka"
print firstname + " " + lastname
```

String Operators

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/example5.py

```
# plus "+"
my_name = firstname + " " + lastname
print my_name

# length "len()"
print "len(my_name) = ",len(my_name)

# Sub-strings
print "Remove 1 character from the front of string: my_name[1:len(my_name)] = ", my_name[1:len(my_name)]
print "Remove all but the final character of string: my_name[len(my_name)-1:len(my_name] = ",
my_name[len(my_name)-1:len(my_name)]
my_name = firstname + " " + lastname
print "Remove the final 5 characters from", my_name, ": my_name[0:len(my_name)-5] = ",
my_name[0:len(my_name)-5]
print "Remove all but the first three characters from", my_name, ": my_name[0:3] = ", my_name[0:3]

# index()
my_name = firstname + " " + lastname
print "my_name.index('L') = ", my_name.index('L')

# rindex()
print "my_name.rindex('a') = ", my_name.rindex('a')
print "my_name[my_name.index('L'):my_name.rindex('a')+1] =
",my_name[my_name.index('L'):my_name.rindex('a')+1]
```

String Comparisons

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/example6.py

```
apples = "apples”
oranges = "oranges”
bananas = "bananas”
# == (equals)
if (apples == "apples"):
           print ”apples == " + apples
# != (not equals)
if (apples != oranges):
           print apples + " != " + oranges
#  > (greater than)
if (bananas > apples):
           print bananas + " > " + apples
#  < (less than)
if (apples < oranges):
           print apples + " < " + oranges
# >= (greater than or equals)
if (oranges >= apples):
           print oranges + " >= " + apples
# <= (less than or equals)
if (bananas <= oranges):
           print bananas + " <= " + oranges + "\n”
```

Lab 2: Data Types & Operators

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/lab2.py

- Write Python script that does the following:
  - Prompts for an number “a” and number “b” and then prints our “c” for each of the following cases (a+b), (a-b), (a*b), (a/d) & (a%b)
  - Try the same where the numbers are integers and floats

- Now do the following with strings:
  - Prompt for a string “a” and a string “b”
  - Print out the length of each string
  - Concatenate the two strings into on
  - Use index to separate the two strings again

Lists - 1

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/example7.py

```
# 1 dimensional List of numbers
a = [0,1,2,3.14159,4,5,6,7,8,9]
print a
for i in range(len(a)):
            print “a[",i,"] =",a[i]
# 1 dimensional List of strings
animals = ["cats","dogs", "birds", "fish"]
for a in range(len(animals)):
            print “animals[",a,"] =",animals[a]
# 2 dimensional List of numbers
A2D = [[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]]
for i in range(4):
            print A2D[i]
# Clear contents of the List
A2D = []
c=0
for i in range(4):
            A2D.append([])
            for j in range(4):
                        c += 1
                        A2D[i].append©
                        print A2D[i][j],"\t”,
            print
```

Lists - 2

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/example7.py

```
for i in range(4):
        for j in range(4):
                print A2D[i][j],"\t",
        print
# 2 dimensional List of strings
A2D = [["c","a","t","s"],["d","o","g","s"],["f","i","s","h"],["b","i","r","ds"]]
for i in range(4):
        for j in range(4):
                print A2D[i][j],
        print
# using len
for i in range(len(A2D)):
        for j in range(len(A2D[i])):
                print A2D[i][j],
        print
animals = ["cats","dogs", "birds", "fish"]
more_animals = ["cows","horses","sheep"]
# extend
animals.extend(more_animals)
print animals
# insert (indexes start at 0)
animals.insert(3,"mice")
print animals
```

Lists - 3

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/example7.py

```
# remove
animals.remove("mice")
print animals
# index
animals.insert(3,"mice")
animals.insert(5,"mice")
print animals
print "\"mice\" first found at: ", animals.index("mice")
# count
print "\"mice\" found: ", animals.count("mice"), "times"
# sort in place
animals.sort()
print animals
# reverse - reverse sort in place
animals.reverse()
print animals
# delete
del animals[3]
print animals
```

Lists - 4

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/example7.py

```
# Lists as stacks (Last In First Out)
# Build a stack using append
stack = []
for i in range (5):
        stack.append(i)
        print "pushing " + str(stack[i])
# now pop the elements off the stack
for i in range (len(stack)):
        print "popping " + str(stack.pop())
# Lists as queues (First In First Out)
queue = []
queue = deque(queue)
for i in range (5):
        queue.append(i)
        print "queuing " + str(queue[i])
# now elements from the queue using popleft()
for i in range (len(queue)):
        print "dequeuing " + str(queue.popleft())
```

Dictionaries - 1

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/example8.py

```
# Dictionaries are essentially (Associative Arrays)
Names = {'dal16':'David Lifka','rda1':'Resa Alvord','shm7':'Susan Mehringer','plr5':'Paul Redfern'}
# keys
for key in Names.keys():
        print key+"\t"+Names[key]
# del - deletes a value from a Dictionary
del Names['dal16']
for key in Names.keys():
        print key+"\t"+Names[key]
# values
for value in Names.values():
        print value
# has_keys - test whether a dictionary key is present
if Names.has_key('plr5'):
        print "plr5 exists"
if Names.has_key('dal16'):
        print "dal16 exists"
else:
        print "dal16 does not exist"
# iteritems to retrieve the key and the value at the same time
for k,v in Names.iteritems():
        print k,v
```

Dictionaries - 2

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/example8.py

```
netids = {'dal16':'David Lifka','rda1':'Resa Alvord','shm7':'Susan Mehringer','plr5':'Paul Redfern'}
for id in netids.keys():
        print id,"\t",netids.get(id)
# Use sort() to sort dictionary by keys
sorted_keys = netids.keys()
sorted_keys.sort()
for id in sorted_keys:
        print id,"\t",netids[id]
# 23. Use sort with a lambda function to sort dictionary by values
students = netids.items()
students.sort(lambda (k1,v1),(k2,v2):cmp(v1,v2))
for id,name in students:
        print id + "\t" + name
```

Logic & Looping Structures

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/example9.py

```
# if, elif, else
a = 5
b = 10
if (a < b ):
        print a,"<",b
if (b < a):
        print b,"<",a
else:
        print b,"is not <",a
if (b < a):
        print b,"<",a
elif (b == a):
        print b,"==",a
else:
        print b,"must be >",a
```

```
# for loops
for i in range(10):
        print i
for i in range(0,100,10):
        print i
for i in range(10,0,-1):
        print i
# while loop i = 0
while(i < 10):
        i += 1
        print i

# infinite loops
#for i in itertools.count():
#       print "here"
#while (true):
#       print "here"
```

Lab 4: Matrix Multiply

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/lab3.py

```
from numpy import *
# A = N x R
# B = R x M
# C = N x M
N = 2
R = 3
M = 4
a = zeros((N,R))
a[0] = [1, 2, 4]
a[1] = [2, 6, 0]
b = zeros((R,M))
b[0] = [4, 1, 4, 3]
b[1] = [0, -1, 3, 1]
b[2] = [2, 7, 5, 2]
c = zeros((N,M))
# compute c
# print a, b & c
# now try using numpy…
d = array([[1, 2, 4],[2, 6, 0]])
D = matrix(d)
e = array([[4, 1, 4, 3],[0, -1, 3, 1],[2, 7, 5, 2]])
E = matrix(e)
# print D * E
```

Reading & Writing Files

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/example10.py

```
import random
import os

# 1. open a file for writing (note >> open a file for append)
out = open('columns.txt', 'w')
for i in range(10):
        a = random.random()
        b = random.random()
        c = random.random()
        out.write(str(a)+"\t"+str(b)+"\t"+str(c)+"\n")
out.close()

# open a file for reading
input = open('columns.txt', 'r')
for i in input:
        i = i[:-1]
        (a, b, c) = i.split("\t")
        print "a = ",a,"\tb =",b,"\tc =",c
input.close()
os.unlink("columns.txt”)
```

Running & Interacting with Applications

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/example11.py

```
import sys,os

command = "ls -al"
child = os.popen(command)
for i in child.readlines():
        i = i[:-1]
        sys.stdout.write(str(i)+"\n")
command = "/Applications/TextEdit.app/Contents/MacOS/TextEdit"
child = os.popen(command)
```

Using FTP from Python

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/example12.py

```
from ftplib import FTP
import getpass
# Ftp example
print ”Using ftp”
server = "arecibo.tc.cornell.edu"
ftp = FTP(server)
print "Username: ",
user = sys.stdin.readline()
user = user[:-1]
pswd = getpass.getpass()
ftp.login(user, pswd)
ftp.cwd("/legacypulsars/Data/pulsars/")
ftp.retrbinary('RETR J2235+1506.52396.043', open('J2235+1506.52396.043', 'wb').write)
ftp.quit()
```

Web Download Example

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/example13.py

```
# Web download example
import urllib # - For Web Download Example
print "Downloading data via http"
url = urllib.urlopen("http://www.cac.cornell.edu/~lifka/Downloads/Ranger/data.csv")
dump = url.readlines()
for r in range(len(dump)):
       print str(r+1)+">"+str(dump[r]),
        line = dump[r]
        line = line[:-1]
        (a, b, c) = line.split("\t")
        print a, b, c, ";",
        print int(a) + int(b) + int(c)
```

Lab 5: Reading & Parsing Data from the Web

http://www.cac.cornell.edu/~lifka/Downloads/Ranger/lab4.py

```
# Some hints
line = line[:-1] # what does this do?
(a, b, c) = line.split("\t”) # what types are a, b & c?

1) For each row read print a, b & c followed by a “:” followed by the sum of a, b & c
```

Thank You!

- Questions?