An Introduction to Python

Harrison B. Prosper
Florida State University

What is Python?

- Python is a scripting language, created by Guido van Rossum, that is:
  - interpreted
  - object-oriented
  - dynamically typed
  - simple
  - powerful and
  - free!
- Python website:
  - http://www.python.org

Getting Started

- The Python interpreter has been ported to all the popular platforms.

- Running Python interactively:
  - python
  - >>> print "Thou lump of foul deformity"
  - Thou lump of foul deformity
  - >>> x = 2; y = 5
  - >>> z = x*y
  - >>> z
  - 10

Writing Python Programs

- Points to note
  - Python code is structured using indentation
  - Indentation is important, so use a program such as emacs or xemacs to write code, or, better still, the Python development program idle.

- Guidelines
  - Write a few lines, then test, then repeat!
  - The first line of a script should contain
  - #!/usr/bin/env python
  - This tells the operating system which program to use to interpret the instructions that follow.

The "Hello World" Program

```
#!/usr/bin/env python
#-----------------------------------------------------
# File:  helloworld.py
# Description: A really boring program
# Created: 10-Mar-2005, Harrison B. Prosper
#-----------------------------------------------------
print "Hello World"
```

Make the file helloworld.py executable:

```
chmod +x helloworld.py
```

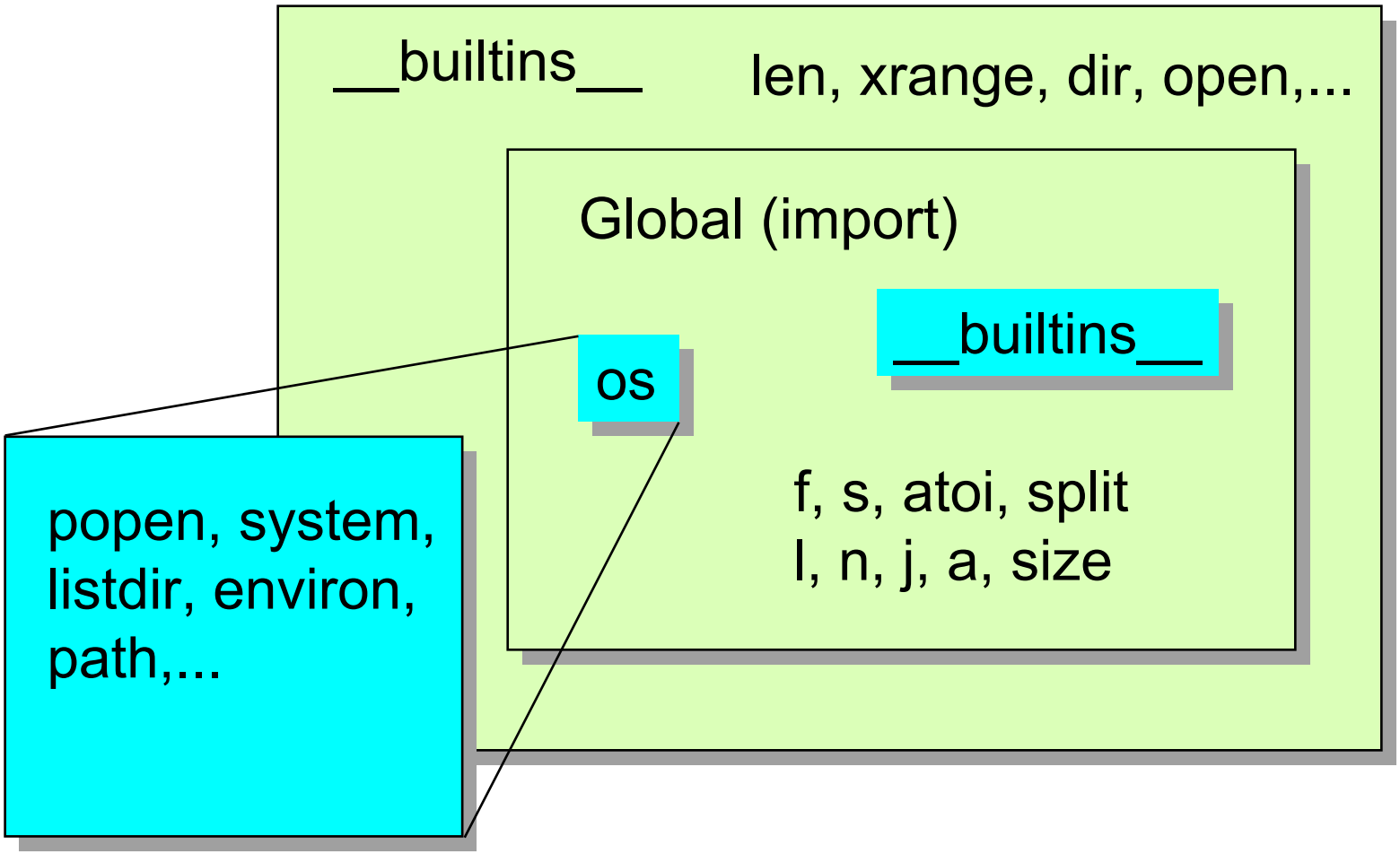
Run it:

```
./helloworld.py
```

Example 1

```
from string import atoi, split # Add atoi, split to global namespace.
import os                      # Add os to global namespace.
f    = os.popen('cd; ls -l')   # Pipe command output to file.
s    = f.read()                # Read into a single string.
l    = split(s,'\n')           # Split into a list of strings.
n    = len(l)                  # Get length of list.
for j in xrange(n):            # Iterate over a range object.
        a = split(l[j])        # Split into a list of strings.
        if (len(a) > 4):       # If length of list > 4 convert
                size = atoi(a[4])   # string value into an integer.
                print size, l[j]
        else:                       NO BRACES BECAUSE
                print l[j]          PYTHON USES INDENTATION!
```

Namespaces


__builtins__
len, xrange, dir, open,...
Global (import)
__builtins__
os
f, s, atoi, split
l, n, j, a, size
popen, system,
listdir, environ,
path,...

Looking at Namespaces

```
l = dir()
for j in xrange(len(l)):
        print l[j]
```

Create a list of the names in the global namespace and print them

```
l = dir(__builtins__)
```

Create a list of the names in the namespace of the module __builtins__

```
l = dir(os)
```

Do likewise for the operating system module os

Example 2 (map, lambda)

```
l = dir(__builtins__)          # List names in module __builtins__
o= map(lambda x: x+`\n',l) # For every element x of list l apply
                               # the one-line function f(x) = x+`\n'
                               # and return the result in the list o.
f = open('builtins.txt','w')   # Open a file object f for output.
f.writelines(o)                # Write out list of newline-terminated
                               # strings.
f.close()                      # Close file object.

from string import strip       # Add strip to global namespace.
g = open('builtins.txt','r')   # Open file for input.
t  = g.readlines()             # Read entire file into list t.
t  = map(strip,t)              # Strip off whitespace (here newline).
```

Example 2.1 – Fix Me!

```
from string import *          # Make all string functions available.

# Read all lines into the list records
records = open('data/mu_tqb_test.dat', 'r').readlines()

# Place columns names into list "names" (from first record)
names  = split(records[0])

# Transform from character to float (skipping first record)
# Problem: Fix the bug!!! Hint: Use another map
data = map( lambda y: atof(y),
            map( lambda x: split(x), records[1:]) )
```

Example 3 (filter)

```
from string import strip, find          # Import strip and find
l = open('builtins.txt').readlines()    # Read entire file into list l.
l = map(strip,l)                        # Strip off whitespace.

o = filter(lambda x: find(x,`Error') > -1, l) # For every x in list l
                                        # apply the one-line function
                                        # f(x) = find(x,'Error') > -1
                                        # If f(x) is True
                                        # copy x to output list o.
```

Example 3.1 (filter)

```
# Filter on last column
signal = filter( lambda x: x[-1] > 0, data )

backg = filter( lambda x: x[-1] < 1, data )
```

Sequence Types

- **Immutable types** (can't be changed)
  - Strings aString = 'Go boil your head'
  - Tuples aTuple = ('Bozo',42,3.14,aString)
- **Mutable types** (can be changed)
  - Lists aList = [aString, aTuple,1997]
  - Dictionaries aMap = {'string':aString, 2:aList}
- **General operators** on sequence s
  - len(s), min(s), max(s), s[i], s[i:j] (= s[i]..s[j-1])
  - x [not] in s True if item x [not] in s, else False
  - s + t concatenate sequences s and t
  - n*s n copies, concatenated

Tuples (immutable)

```
                          0     1       2       3     4     5
>>> IamAtuple = ('The','time',3.141,'has',42,'come')
>>> IamAtuple[0]; IamAtuple[5]
'The'
'come'

>>> IamAtupleSlice = IamAtuple[3:6]
>>> IamAtupleSlice
('has',42,'come')

>>> IamAtupleCopy = IamAtuple[:]
>>> len(IamAtupleCopy)
6
```

Lists (mutable)

```
>>> IamAlist = ['The','time',3.141,'has',42,'come']
>>> IamAlist[0]
'The'

>>> IamAlist[2] = (1,2,3); IamAlist
['The','time',(1,2,3),'has',42,'come']

>>> IamAlist.sort()
[42,'The','come','has','time',(1,2,3)]

>>> IamAlist.append(1997); IamAlist
[42,'The','come','has','time',(1,2,3),1997]
```

Dictionaries (mutable)

```
>>> IamAmap = {}                          # An empty dictionary
>>> IamAmapAlso = {1:'one',(2,'boo'):'two'}

>>> IamAmap['language'] = ('Python',2007)      # Bind keys to
>>> IamAmap['comment'] = 'Superb!'             # values.
>>> IamAmap['cost'] = 0
>>> IamAmap['map'] = IamAmapAlso

>>> IamAmap.keys()                        # Get list of keys
['cost','language','comment','map']

>>> IamAmap.values()                      # Get list of values
[0,('Python',2007),'Superb!',{1:'one',(2,'boo'):'two'}]
```

Functions (def, None, return)

```
def goofy(a, b='yahoo', **k):          # a is a required argument.
                                       # b is a defaulted argument.
                                       # **k is a set of keyword args
    r1 = r2 = None                     # None is a Python object.
    if ( k.has_key('title') ):         # Check if key 'title' exists.
        r2 = k['title']                # Get value for given key.
    return (42, r2)                    # Return as a tuple.
```

```
>>> a, b = goofy(1,'silly', title='2001',author='A.C. Clarke')
>>> print a, b
42 2001
```

Classes

```
class FourVector:                                   # Name of class

    def __init__(self, E=0,px=0,py=0,pz=0):         # Constructor
        self.ok = True                              # Creation ok
        if E >= 0:
            self.E, self.py, self.pz, self.pz = E,px,py,pz
        else:
            self.ok = False                         # Creation failed

    def __del__(self):                              # Destructor
        pass                                        # Do nothing

>>> p = FourVector(12,-14,13,80)
```

Classes, __str__ method

```
        def __str__(self):                  # Used by print and str(*)
                s =      'E %d\n'  % self.E  # Uses the C printf
                s = s + 'pz %d\n' % self.px  # format strings
                s = s + 'py %d\n' % self.py
                s = s = 'pz %d'    % self.pz
                return s                    # Return string
# Note: We could also have written the above
#         as s  = '%s %d\n' % ('E',self.E),  etc..
>>> print p
E 80
px 12
py  -14
pz 13
```

Formatting Strings

```
names = {}                                  # Define empty map
names['product'] = 'Python'                 # Set (key, value) pairs
names['year'] = 2007
names['version'] = 2.5

s = ''                                      # Empty string
s = s + 'Product: %(product)s, ' \           # Format string
        'Version: %(version)2.2f, ' \
        'Year: %(year)d' % names

>>> print s, ; print '….'                   # Note use of comma
Product: Python, Version: 2.50, Year: 2007 ….
                                            # to suppress newline
```

Classes, __add__, __sub__

```
def __add__(self, v):          # Implement +
        u = FourVector()
        u.E  = self.E  + v.E
        u.px = self.px + v.px
        u.py = self.py + v.py
        u.pz = self.pz + v.pz
        return u

def __sub__(self, v):          # Implement -
        :       :
```

Classes, __mul__

```
def __mul__(self, v):                    # Implement *
        u =  self.px*v.px+self.py*v.py+self.pz*v.pz
        return self.E*v.E - u
```

```
>>> p = FourVector(10,1,2,3)
>>> q = FourVector(20,0,0,-1)
>>> pq = p * q
```

Classes, Inheritance

```
class Particle(FourVector):
        def __init__(self, label='e-',E=0,px=0,py=0,pz=0):

                # Call FourVector constructor to initialize E, etc.
                FourVector.__init__(self, E,px,py,pz)

                if self.ok:                     # Check for success
                        self.label = label # Create another attribute

        def __del__(self):
                print 'Goodbye cruel world!'
```

Dynamically Created Attributes - 1

```
from math import *                    # Import all math functions

class Particle(FourVector):
        :       :
        def __getattr__(self, attribute): # Method to get values
                                          # of dynamic attributes
                if attribute == 'mass':
                        x = sqrt(self.E**2
                                -(self.px**2+self.py**2+self.pz**2))
                        return x
                else:                     # Ok; have a tantrum!
                        raise AttributeError, attribute
```

Dynamically Created Attributes - 2

```
class Particle(FourVector):
        :       :
        def __setattr__(self, attribute, value): # Method to set
                                        # values of dynamic attributes
                if attribute == 'version':

#NB:   Do not write self.version = value because this
#      would create a statically defined attribute! Instead set it
#      using the object's dictionary attribute:

                        self.__dict__[attribute] = value
                else:
                        raise AttributeError, attribute
```

Odds and Ends - 1

Changing the Python search path on-the-fly

```
import sys
sys.path.append('~harry/python/tutorial')
sys.path.append('~harry/python/examples')
```

Spawning commands to the operating system

```
import os
os.system('cd; ls -la')
```

Debugging Python scripts

```
from pdb import run
run('p = FourVector(90,1,1,2)')
```

Odds and Ends - 2

Evaluating strings and updating current namespace

```
p = eval('FourVector(90,1,1,2)')
```

Executing statements and updating current namespace)

```
exec('p = FourVector(90,1,1,2)')
                    # See also, execfile(filename)
```

Handling errors

```
>>> s = open('FourVector.txt','r').read()
>>> try:                # Try following statements and
...     p = eval(s)     #  if an error (i.e., an exception) occurs
... except:              # execute these statements.
...     exec(s)          # Note: Can also catch  specified
>>> print p             # exceptions with the except keyword.
```

PyROOT - 1

```
from time import sleep
from ROOT import *
                    :       :
canvas = TCanvas('wtmass', 'W Transverse Mass',
                             500, 0, 500, 500)
hist = TH1F('hwtmass', 'M_{WT} (GeV)', 50, -2.5, 2.5)
                    :       :
INDEX = 26
for row, record in enumerate(data):
        wtmass = record[INDEX]
        hist.Fill(wtmass)
        if row % 100 == 0:
                canvas.cd(); hist.Draw(); canvas.Update()
sleep(10)       # Wait for 10 seconds
```

PyROOT - 2

```
hist.Scale(1.0/hist.Integral())
hist.SetLineColor(2)
hist.SetLineWidth(2)

hw.SetLineColor(4)
hw.SetLineWidth(2)

cw.cd()
hist.Draw()
hw.Draw("SAME")
cw.Update()
cw.SaveAs(".gif")
cw.SaveAs(".pdf")
gApplication.Run()   # Block program
```

KDE Class - 1

```
from math import *
class KDE:
        def __init__(self, data):
                self.data = data
                # Compute bandwidth h
                n  = len(data)
                m = sum(data) / n    # Compute average
                #  Compute variance v; x + (y-m)**2 → x
                v  = reduce(lambda x, y: x+(y-m)**2,data,0.0)/n
                d  = 1.0
                h  = sqrt(v) * ( 4.0/( (d+2)*n) )**(1.0/(d+4))
                self.h  = h
                self.w = 0.5 / h**2
                self.a = 1.0 / (h * sqrt(2 * pi))
```

KDE Class - 2

```
from math import *
class KDE:
        def __init__(self, data):
                :
        def setbandwidth(self, h):
                self.h = h

        def density(self, x):
                h = self.h; w = self.w
                p = reduce(lambda z, y: z + exp(-w*(x-y)**2),
                                self.data, 0.0)
                return self.a * p / len(self.data)
```