Introduction to Python

Isaac Ye, High Performance Technical Consultant
SHARCNET, York University
isaac@sharcnet.ca
Outlines

• What is python?

• Versions and Python in SHARCNET

• Simple practice

• Data structure

• Data handling practice

• Libraries
Python

- Open source general-purpose high-level programming language.
- Interpreted language
- Object Oriented, Procedural, Functional
- Easy to interface with C/ObjC/Java/Fortran
- Simplicity but many advantages such as automatic memory management
- many libraries(NumPy, SciPy, SymPy)
### Version 2.x / 3.x?

<table>
<thead>
<tr>
<th>2.x</th>
<th>3.x</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Released in late 2000</td>
<td>• Faster and neat</td>
</tr>
<tr>
<td>• Lots of web resources</td>
<td>• In near future all modules/libraries will be moved to 3.x versions</td>
</tr>
<tr>
<td>• Many third-party libraries</td>
<td></td>
</tr>
</tbody>
</table>

Note) You could use either version depending on your goal. However, it is okay for the most Python beginners to stay with 2.7.x version.
Python in SHARCNET

[isaac@orc-login2:~] module avail python
---------------------/opt/sharcnet/modules ----------------------------------
python/gcc/2.7.5   python/gcc/2.7.8   python/intel/2.7.5 python/intel/2.7.8 python/intel/3.4.2

[isaac@orc-login2:~] module load python/intel/2.7.8
[isaac@orc-login2:~] which python
/opt/sharcnet/python/2.7.8/intel/bin/python

[isaac@orc-login2:~] python
Python 2.7.8 (default, Sep 18 2014, 11:21:42) [GCC Intel(R) C++ gcc 4.4 mode] on linux2
Type "help", "copyright", "credits" or "license" for more information.
How to install python on my PC?

- Download the installation package at
  
  www.python.org/download

- Find the appropriate version based on your operating system in your PC (Mac OSX and LINUX have python in default)

- Or you may get ‘ANACONDA’ which has many pre-installed libraries at

  https://www.continuum.io/downloads
Setup a developing environment

• Various options available (iPython, IDEs) but here we use ‘terminal’ which is a default environment in SHARCNET.

• The source code is edited by ‘nano’ editor in the terminal.
Interactive interface to Python

Handy to run a simple code!

$ python
Python 2.7.8 (default, Sep 18 2014, 11:21:42)
[GCC Intel(R) C++ gcc 4.4 mode] on linux2
Type "help", "copyright", "credits" or "license" for more information.

>>> 2+3
5

>>> print('Hello world!')
Hello world!

>>> quit()
Running a Script

Easy to edit/run a long code!

```bash
$ cat test.py
2+3
print('Hello world!')
$ python test.py
Hello world!
```

You could put the interpreter info and make it executable

```bash
$ cat test.py
#!/opt/sharcnet/python/2.7.8/intel/bin/python
2+3
print('Hello world!')
$ chmod 777 test.py
$ ./test.py
Hello world!
```
Simple math practice

$ cat test.py
a = 55
print(a)
print('double of a is:', 2.*a)

b = 2.*a
print('b is:', b)

a = 100
print('double of a is:', 2.*a)
print('b is:', b)

$ python test.py
55
('double of a is:', 110.0)
('b is:', 110.0)
('double of a is:', 200.0)
('b is:', 110.0)

Assign 55 to variable a
Set ‘b=2a’
Reassign 100 to a
b value is not change!
Data structures: LISTS

• Defined by writing a list of comma separated values in square brackets which might have different types for each item but for most of case we keep them all same type.

```python
>>> odds = [1,3,5,7]
>>> print('fist and last:', odds[0], odds[3])
('fist and last:', 1, 7)
>>> print('fist and last:', odds[0], odds[-1], odds[-2])
('fist and last:', 1, 7, 5)
>>> odds.append('9')
>>> print(odds)
[1, 3, 5, 7, '9']
>>> odds.reverse()
>>> print('odds after reverse', odds)
('odds after reverse', ['9', 7, 5, 3, 1])

>>> names=['isaac', 'paul', 'bob']
>>> print(names)
['isaac', 'paul', 'bob']
>>> names[1] = 'pole'
>>> print(names)
['isaac', 'pole', 'bob']
>>> print(names[0:2])
['isaac', 'pole']
```
Data structures: STRINGS

- Defined by using of single (') or double(“) or triple(”’) quotes. Strings are immutable so that the change of the part in strings is not permitted.

```python
>>> Greeting = 'Hello'
>>> print(Greeting)
Hello
>>> print(Greeting[3])
l
>>> Greeting[3]='8'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
>>> len(Greeting)
5
>>> print(Greeting+' World!')
Hello World!
>>> Greeting1 = ''' There can be a space line
...     inbetween this
...     lines'''
>>> print Greeting1
There can be a space line inbetween this lines
```
Data structures: TUPLES

- Represented by a number of csv format. These are immutable and the output comes with nested parentheses.

```python
>>> (1,'Awesome')
(1, 'Awesome')
>>> pair=('+','plus')
>>> sign, name = pair
>>> print(sign, name)
('+', 'plus')
>>> a=1
>>> b=2
>>> a,b = b,a
>>> print(a,b)
(2, 1)
>>> print('a=',a,'b=',b)
('a=', 2, 'b=', 1)
>>> pair[0]
'+'
>>> pair[0] = '-'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```
Data structures: DICTIONARY

- Dictionaries are another container like lists, but instead of being indexed by a number like 0 or 1 it is indexed by a key which can be almost anything. The name comes from being able to use it to represent a dictionary.

```python
>>> extensions = {'Isaac':1000, 'Tom':2000, 'Mike':3000, 'Josh':4000}
>>> extensions['Isaac'] = 5000
>>> extensions
{'Isaac': 5000, 'Mike': 3000, 'Josh': 4000, 'Tom': 2000}
>>> extensions.keys()
['Isaac', 'Mike', 'Josh', 'Tom']
```
What we want to practice

- load large data (csv format) into memory
- calculate average, max, min along the column or row in the 2-D array data

```
[isaac@orc-login2:/work/isaac/swc/python/data] more data-01.csv
0,0,1,3,1,2,4,7,8,3,3,3,10,5,7,4,7,7,12,18,6,13,11,11,7,4,6,8,8,4,4,5,7,3,4,2,3,0,0
0,1,2,1,2,1,3,2,2,6,10,11,5,9,4,4,7,16,8,6,18,4,12,5,12,7,11,5,11,3,3,5,4,4,5,5,1,1,0,1
0,1,1,3,3,2,6,2,5,9,5,7,4,5,4,15,5,11,9,10,19,14,12,17,7,12,11,7,4,2,10,5,4,2,2,3,2,2,1,1
0,0,2,0,4,2,2,1,6,7,10,7,9,13,8,8,15,10,10,7,17,4,4,7,6,15,6,4,9,11,3,5,6,3,3,4,2,2,2,2,1
0,1,1,3,3,1,3,5,2,4,4,7,6,5,3,10,8,10,6,17,9,14,9,7,13,9,12,6,7,7,9,6,3,2,2,4,2,0,1,1
0,0,1,2,2,4,2,1,6,4,7,6,6,9,9,15,4,16,18,12,12,5,18,9,5,3,10,3,12,7,8,4,7,3,5,4,4,3,2,1
0,0,2,2,4,2,2,5,5,8,6,5,11,9,4,13,5,12,10,6,9,17,15,8,9,3,13,7,8,2,8,8,4,2,3,5,4,1,1,1
0,0,1,2,3,1,2,3,5,3,7,8,8,5,10,9,15,11,18,19,20,8,5,13,15,10,6,10,6,7,4,9,3,5,2,5,3,2,2,1
```

- plot the result
Loading csv format data

```python
>>> import numpy
>>> numpy.loadtxt(fname='data-01.csv', delimiter=',')
array([[ 0.,  0.,  1., ...,  3.,  0.,  0.],
       [ 0.,  1.,  2., ...,  1.,  0.,  1.],
       [ 0.,  1.,  1., ...,  2.,  1.,  1.],
       ...
       [ 0.,  1.,  1., ...,  1.,  1.,  1.],
       [ 0.,  0.,  0., ...,  0.,  2.,  0.],
       [ 0.,  0.,  1., ...,  1.,  1.,  0.]])
```

NumPy stands for Numerical Python. The most powerful feature of NumPy is n-dimensional array. This library also contains basic linear algebra functions, Fourier transforms, advanced random number capabilities and tools for integration with other low level languages like Fortran, C and C++

```python
>>> help(numpy.loadtxt)
loadtxt(fname, dtype=<type 'float'>, comments='#', delimiter=None, converters=None, skiprows=0, usecols=None, unpack=False, ndmin=0)
    Load data from a text file.
```
Checking loaded data

```python
>>> import numpy
>>> data = numpy.loadtxt(fname='data-01.csv', delimiter=',')

>>> data.shape
(60, 40)

>>> data
array([[ 0.,  0.,  1., ...,  3.,  0.,  0.],
       [ 0.,  1.,  2., ...,  1.,  0.,  1.],
       [ 0.,  1.,  1., ...,  2.,  1.,  1.],
       ...
       [ 0.,  1.,  1., ...,  1.,  1.,  1.],
       [ 0.,  0.,  0., ...,  0.,  2.,  0.],
       [ 0.,  0.,  1., ...,  1.,  1.,  0.]]

>>> numpy.max(data)
20.0

>>> numpy.min(data)
0.0

>>> numpy.mean(data)
6.1487499999999997

>>> type(data)
<type 'numpy.ndarray'>

>>> data.dtype
dtype('float64')

>>> print('first value in data:', data[0,0])
('first value in data:', 0.0)

>>> print('middle value in data:', data[30,20])
('middle value in data:', 13.0)
```
Slicing data array

```python
>>> data[0:4, 0:10]
array([[ 0.,  0.,  1.,  3.,  1.,  2.,  4.,  7.,  8.,  3.],
       [ 0.,  1.,  2.,  1.,  2.,  1.,  3.,  2.,  2.,  6.],
       [ 0.,  1.,  1.,  3.,  3.,  2.,  6.,  2.,  5.,  9.],
       [ 0.,  0.,  2.,  0.,  4.,  2.,  2.,  1.,  6.,  7.]]

>>> small = data[:3, 36:]
>>> small
array([[ 2.,  3.,  0.,  0.],
        [ 1.,  1.,  0.,  1.],
        [ 2.,  2.,  1.,  1.]])

>>> small2 = 2*small
>>> small2
array([[ 4.,  6.,  0.,  0.],
        [ 2.,  2.,  0.,  2.],
        [ 4.,  4.,  2.,  2.]])

>>> small2.trace()
8.0
>>> small2.transpose()
array([[ 4.,  2.,  4.],
        [ 6.,  2.,  4.],
        [ 0.,  0.,  2.],
        [ 0.,  2.,  2.]])
```
Handling data in array using

```python
>>> patient0 = data[0, :]
>>> print('max inflammation for patient 0:', patient0.max())
('max inflammation for patient 0:', 18.0)
>>> patient3 = data[3, :]
>>> print('max inflammation for patient 0:', patient3.max())
('max inflammation for patient 0:', 17.0)
>>> numpy.mean(data, axis=0)
array([ 0.        ,   0.45      ,   1.11666667,   …
     ,   1.13333333,   0.56666667])
>>> dailymean=numpy.mean(data, axis=0)
>>> dailymean.shape
(40,)
>>> numpy.mean(data,axis=1)
array([ 5.45 ,  5.425,  6.1  ,  5.9  ,
     …  5.9  ])
>>> numpy.mean(data,axis=1).shape
(60,)
```
Plotting image

```python
>>> import matplotlib.pyplot
>>> image = matplotlib.pyplot.imshow(data)
>>> matplotlib.pyplot.show()
>>> avg_inflammation = numpy.mean(data, axis=0)
>>> avg_plot = matplotlib.pyplot.plot(avg_inflammation)
>>> matplotlib.pyplot.show()
```
‘FOR’-loop

• For iterative work, ‘FOR’-loop is also available in Python like most languages.

```python
>>> word = 'lead'

>>> for char in word:
...     print(char)
...
lead

>>> length = 0
>>> for num in 'lead':
...     length=length+1
...

>>> length
4

>>> len(word)
4

>>> range(0,10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> for num in range(0,10):
...     print("{0} squared is {1}".format(num, num*num))
...
0 squared is 0
1 squared is 1
2 squared is 4
3 squared is 9
4 squared is 16
5 squared is 25
6 squared is 36
7 squared is 49
8 squared is 64
9 squared is 81
```
The `glob` library contains a function, also called `glob`, that finds files and directories whose names match a pattern. We provide those patterns as strings: the character `*` matches zero or more characters, while `?` matches any one character. We can use this to get the names of all the CSV files in the current directory:

```
$ ls data-0*
data-01.csv  data-02.csv  data-03.csv  data-04.csv

>>> import glob
>>> print(glob.glob('data*.csv'))
['data-01.csv', 'data-02.csv', 'data-03.csv', 'data-04.csv']

>>> import numpy
>>> import matplotlib.pyplot
>>> filenames = sorted(glob.glob('data*.csv'))
>>> filenames = filenames[0:4]
>>> filenames
['data-01.csv', 'data-02.csv', 'data-03.csv', 'data-04.csv']
```
Handling multiple files (cont’d)

```python
>>> for f in filenames:
...     print(f)
...     data = numpy.loadtxt(fname =f, delimiter=','
...     numpy.mean(data, axis=0)
...     numpy.mean(data, axis=1)
...

data-01.csv
array([ 0. , 0.45 , 2.48333333, 1.5 , 1.13333333, 0.56666667])
array([ 5.45 , 5.425, 6.1 , 6.25 , 6.4 , 7.05 , 5.9 ])

data-02.csv
array([ 0. , 0.5 , 0.93333333, 1.58333333, 2.46666667, 1.5 , 1.13333333, 0.53333333])
array([ 6.35 , 5.7 , 5.9 , 6.025, 6.55 , 7.2 , 6.925])

data-03.csv
array([ 0. , 0. , 0.58333333, 1.26666667, 1.4 , 0.46666667, 0. , 0. ])
array([ 4. , 4.225, 3.9 , 4.325, 3.575, 4.075, 0. , 0. ])

data-04.csv
array([ 0. , 0.46666667, 2.41666667, 1.5 , 0.91666667, 0.43333333])
array([ 5.725, 6.125, 5.925, 5.65 , 5.025, 6.275, 6.05 ])```
Making choices

• We can ask Python to take different actions, depending on a condition with ‘if’ statement:

```python
>>> num=37
>>> if num> 100:
...     print('greater')
... else:
...     print('not greater')
... not greater

$ cat test3.py
num = -3
if num > 0:
    print(num, "is positive")
elif num == 0:
    print(num, "is zero")
else:
    print(num, "is negative")
print("done")
$ python test3.py
(-3, 'is negative')
done
```
Checking the loaded data

```python
import numpy

data=numpy.loadtxt(fname='data-01.csv', delimiter=',',

if numpy.max(data, axis=0)[0] == 0 and numpy.max(data, axis=0)[20] == 20:
    print('Suspicious looking maxima!')
elif numpy.sum(numpy.min(data, axis=0)) == 0:
    print('Minima add up to zero!')
else:
    print('Seems ok!')

$ python 1.py
Suspicious looking maxima
```
Function

- `def` is the keyword to define a function. `add` in the above example is the name. All functions require a parameter list surrounded by an open bracket "(" and close bracket ")" even if there are no parameters. `return` is also a keyword which is required return a value. If it isn't provided `None` is returned. Function bodies are block like if statements and for loops.

```python
>>> def power(a,b):
...     return a**b
... >>> power(2,3)
8
```
Libraries

- Loading ‘math’ library which has various functions. It must be used by referencing it like ‘math.cos()’

- Importing entire namespace in math library so it can be used without reference like ‘cos()’

```python
>>> import math
>>> math.cos(60)
-0.9524129804151563
>>> pi
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'pi' is not defined
>>> from math import *
>>> pi
3.141592653589793
```
Library: Math

- This module is always available. It provides access to the mathematical functions defined by the C standard.

```python
>>> cos(30)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'cos' is not defined
>>> import math
>>> math.cos(30)
0.15425144988758405
>>> math.pi
3.141592653589793
```

- A long library name can be shorten as an alias

```python
>>> import math as m
>>> dir(m)
```
Library: NumPy

- Offers Matlab-ish capabilities within Python
- Fast array operations
- 2D arrays, multi-D arrays, linear algebra

```python
>>> import numpy as np
>>> data = np.loadtxt(fname = 'data-01.csv', delimiter=','
>>> np.min(data, axis=0)
array([ 0.,  0., ... 0.,  0.])

>>> from numpy import poly1d
>>> p = poly1d([3,4,5])
>>> print p
2
3 x + 4 x + 5
>>> print p*p
4 3 2
9 x + 24 x + 46 x + 40 x + 25
>>> print p.integ(k=6)
3 2
1 x + 2 x + 5 x + 6
>>> print p.deriv()
6 x + 4
>>> p([4,5])
array([ 69, 100])
```
Matplotlib

- Plotting vast variety of graphs, starting from histograms to line plots to heat plots. You can also use Latex commands to add math to your plot.

```python
$ cat 3.py
import numpy
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt

mu, sigma = 100, 15
x = mu + sigma*numpy.random.randn(10000)
n, bins, patches = plt.hist(x,50, normed=1, facecolor='blue', alpha=0.75)
y = mlab.normpdf(bins, mu, sigma)
l = plt.plot(bins, y, 'r--', linewidth=1)

plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title(r'$\mathrm{Histogram\ of\ IQ:\ }\mu=100, \ \sigma=15$')
plt.axis([40, 160, 0, 0.03])
plt.grid(True)
plt.show()
```
SciPy

- SciPy contains additional routines needed in scientific work: for example, routines for computing integrals numerically, solving differential equations, optimization, and sparse matrices (note: Scipy sub-packages need to be imported separately)

```python
$ cat 2.py
from scipy.interpolate import interp1d
import numpy
import matplotlib.pyplot

x = numpy.arange(0,10)
y = numpy.array([3., -4., -2., -1., 3., 6., 10., 8., 12., 20.])
f = interp1d(x,y,kind='cubic')

xint = numpy.arange(0, 9.01, 0.01)
yint = f(xint)

img=matplotlib.pyplot.plot(x,y,'o', c='b')
img=matplotlib.pyplot.plot(xint,yint,'-r')
matplotlib.pyplot.show(img)
```
• SymPy is a Python library for symbolic mathematics. Symbolic computation deals with the computation of mathematical objects symbolically. This means that the mathematical objects are represented exactly. (note: It is an extra feature so that you may need to install it)

```python
>>> from sympy import symbols
>>> x, y = symbols('x y')
>>> eq = x+2*y
>>> eq
x + 2*y
>>> eq +1
x + 2*y + 1
>>> eq -x
2*y
>>> from sympy import expand, factor
>>> eq1 = expand(x*eq)
>>> eq1
x**2 + 2*x*y
>>> factor(eq1)
x*(x + 2*y)
>>> eq1.subs([(x,3),(y,-1)])
3
>>> diff(sin(x),x)
cos(x)
>>> limit(1/x, x, oo)
0
>>> integrate(6*x**5, x)
x**6
>>> solve(x**4-1, x)
[-1, 1, -I, I]
>>> solve([x+5*y-2, -3*x+6*y-15], [x,y])
{x: -3, y: 1}
```
Thank you!