Brief Tutorial on Python

Xiaobai Liu

Why Python

- Free
- Popular in Industry/Academia
- Development community
- Source codes available
- Easy to use
  - Variables without declaring
  - Define classes but are not enforced

like Java

Installation

- Mac OS

  $ sudo pip install python

- Ubuntu (or linux)

  $ sudo apt-get install python2.7

- Windows: (https://www.python.org/downloads/windows/)

Outline

- Part 1: Getting Started
- Part 2: Python Commands
- Part 3: Examples

1.1 as calculator

```
$ python
>>> 2*1024
>>> 2**1024
>>> for n in [1,2,3,4,5,6]
...                 print n**2

$CTRL-D
```

Stop interpreter

1.2 library

```
>>> from math import sqrt, exp
>>> exp(-1)
>>>sqrt(2)
```

1.3 Defining functions

```
>>> def f(x):
…                 return x*x

>>> f(2)

>>>f= lambda x: x*x
>>>f(2)
```

1.4 Files

Edit a file “Sfunc.py” with the following lines:

```
sef f(x):
    return x*x
```

Sef = Typo for def

```
>>> From Sfunc import f
>>> f(1.5)
```

function name

1.5 Scripts

Use an editor to create a file name SayHi containing the following lines

```
#! /usr/bin/ python
print " Hello World!"
print "- From your friendly Python program“
```

$chmod 755 SayHi

Not chmod 755 SayHi ~~.py~~

$./SayHi

Outline

- Part 1: Getting Started
- Part 2: Python Commands
- Part 3: Examples

2.1 Comments

In a .py file:

print " Hello world" # this is silly

""" This is an example of a long comment
that goes on
and on
and on."""

2.2 Numbers and data types

- Integers: 5, 23, -75
- Floats or floating point numbers: 5.0, - 23.09
- Long integer: 1234568901
- Check types:

```
>>>type(-75)
>>>a=3.4
>>>type(a)
```

- Complex numbers

```
>>> 2j-1
>>>complex(2,3)
```

2.3 Strings

Single or Double quotes

>>> “this is a string”

>>> ‘This is a string, too’

2.4 Lists and Tuples

Tuple

>>> (1,3,2)

>>> type((1 ,3 ,2))

<type ’tuple ’>

Lists

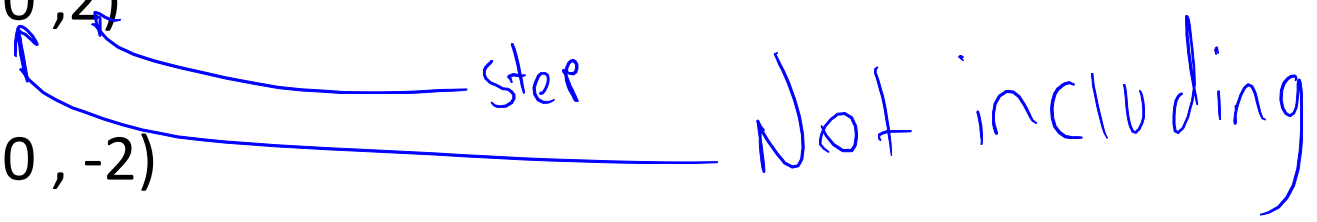
>>> [1 ,”helloo” ,4 ,(1 ,6)]

By nesting lists within lists in this way, we can build up complicated stuctures.

2.5 range

Create lists of integers

```
>>> range (17)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]
>> range (1 ,10)
[1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range ( -6 ,0)
[-6, -5, -4, -3, -2, -1]
>>> range (1 ,10 ,2)
[1, 3, 5, 7, 9]
>>> range (10 ,0 , -2)
[10 , 8, 6, 4, 2]
```



6.Boolean values

```
>>> True
True
>>> type( True)
<’bool ’>type
```

7. Expressions

Formed from variables, constants, function evaluations, and operators.

```
>>>2+2
>>> 2**100
>>> f((x-1)/(x+1))
```

8. Operators

Arithmetic operators: + for addition, - for subtraction, * for multi-plication, and / for division.

```
>>> 25/3
8
>>>25.0/3
8.333333333333339
>>>2**(1/2) ???
>>> 5 % 2
1
```

Other operators: <, >, <=, >=, ==, !=, <>.

operator // = wrap & floor as integer

9.Variables and Assignments

Variable name: any sequence of letters, numbers, and the underscore (_)

```
>>> x = 10
>>> x = x + 1
>>> print x
```

10.Decisions

```
x = 1
if x > 0:
        print " Friday is wonderful"
elif  x<-10:
        print " Monday sucks“
else:
        print no idea”
print " Have a good weekend“
```

Needs to be between if & else

Result?
A. Friday is wonderful
Have a good weekend
B. Friday is wonderful

Line indents

11. Loops

```
for i in [2, 4, 6, 0]:
        print i
else:
        print “blastoff”
```

2
4
6
0
blastoff

```
n = 0
while n < 10:
        print n
        n = n + 3
```

0
3
6
9

loop statements: break, continue

12. Access lists

```
>>> a = [2, " Jack", 45, "23 Wentworth Ave"]
>>> a[0]
2

>>>a[0] = 2002
>>> len(a)
4

>>>a[2:3]
45
```

x[:5] is equivalent to x[0:5] and x[2:] is equivalent to x[2:len(x)].

```
>>> list [18:: -1]
[17 , 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

>>> [2 ,3 ,6 ,10] + [4 ,0 ,0 ,5 ,0]
[2, 3, 6, 10, 4, 0, 0, 5, 0]

>>> x = [3, 6, 8, 9]
>>> x. append (999)
>>> x
[3, 6, 8, 9, 999]
```

```
>>> x = ['a', 'c', '3', 'd', '7']
>>> x. insert (0 ,100)
>>> x
[100 , 'a', 'c', '3', 'd', '7']

>>> x. remove ('a')
>>> x
[100 , 'c', 'junk ', '3', 'd', '7']

>>>x. pop (0)
100
>>> x
['c', 'junk ', '3', 'd', '7']

>>> x. pop()
'7'
>>> x
['c', 'junk ', '3', 'd']
```

13. Strings

- A string is a sequence of characters;
- Similar as Lists: starting at 0, slicing, len, concatenation
- Immutable: not allowed to change individual chars;
- Methods: .capitalize(), .find(), .index()

```
>>> a = 'gobbletygook is refreshing'
>>> a. capitalize()
' Gobbletygook is refreshing'
```

Outline

- Part 1: Getting Started
- Part 2: Python Commands
- Part 3: Examples

Example-1: Add two Matrices

```
#addMatrix.py
# Program to add two matrices using nested loop

X = [[12,7,3],
    [4 ,5,6],
    [7 ,8,9]]

Y = [[5,8,1],
    [6,7,3],
    [4,5,9]]

result = [[0,0,0],
          [0,0,0],
          [0,0,0]]

# iterate through rows
for i in range(len(X)):
   # iterate through columns
   for j in range(len(X[0])):
       result[i][j] = X[i][j] + Y[i][j]

for r in result:
   print(r)
```

Example-2: Transpose a Matrix $\simeq M_{3x2} \rightarrow M_{2x3}$

```
#transpose.py
# Program to transpose a matrix using nested loop

X = [[12,7],
    [4 ,5],
    [3 ,8]]

result = [[0,0,0],
          [0,0,0]]

# iterate through rows
for i in range(len(X)):
    # iterate through columns
    for j in range(len(X[0])):
        result[j][i] = X[i][j]

for r in result:
    print(r)
```

i=0 j=0 j=1
i=1 j=0 j=1

//swap

Example-3: Matrix Multiplication

```
#multiplication.py
# Program to multiply two matrices using nested loops
# 3x3 matrix
X = [[12,7,3],
    [4 ,5,6],
    [7 ,8,9]]
# 3x4 matrix
Y = [[5,8,1,2],
    [6,7,3,0],
    [4,5,9,1]]
# result is 3x4
result = [[0,0,0,0],
          [0,0,0,0],
          [0,0,0,0]]
# iterate through rows of X
for i in range(len(X)):
   # iterate through columns of Y
   for j in range(len(Y[0])):
       # iterate through rows of Y
       for k in range(len(Y)):

for r in result:
   print(r)
```

Example-4: Sort a String

```
 #sortString.py
# Program to sort alphabetically the words form a string
provided by the user

# change this value for a different result
my_str = "Hello this Is an Example With cased letters"

# uncomment to take input from the user
#my_str = input("Enter a string: ")

# breakdown the string into a list of words
words = my_str.split()

# sort the list
words.sort()

# display the sorted words

print("The sorted words are:")
for word in words:
   print(word)
```

Example-5: Remove Punctuations from a String

```
 # punctuation.py
# define punctuation
punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''

my_str = "Hello!!!, he said ---and went."

# To take input from the user
# my_str = input("Enter a string: ")

# remove punctuation from the string
no_punct = ""
for char in my_str:
   if char not in punctuations:

# display the unpunctuated string
print(no_punct)
```

Not in

Example-6: Image Resolution

```
 #imageSize.py
def jpeg_res(filename):
   """This function prints the resolution of the jpeg image
file passed into it"""
   # open image for reading in binary mode
   with open(filename,'rb') as img_file:
      # height of image (in 2 bytes) is at 164th position
      img_file.seek(163)

      # read the 2 bytes
      a = img_file.read(2)

      # calculate height
      height = (a[0] << 8) + a[1]

      # next 2 bytes is width
      a = img_file.read(2)

      # calculate width
      width = (a[0] << 8) + a[1]

   print("The resolution of the image is",width,"x",height)

jpeg_res("sd.jpg")
```

Example-7: Read Files

```
#

import glob

python_files = glob.glob('*.py')
for file_name in sorted(python_files):
    print('    ------' + file_name)

    with open(file_name) as f:
        for line in f:
            print('    ' + line.rstrip())

    print
```

Example-8: Scatters and Lines

```
 #linePlot.py
# visualize data points and lines
import numpy as np
import matplotlib.pyplot as plt
import plotly.plotly as py

line = plt.figure()

np.random.seed(5)
x = np.arange(1, 101)
y = 20 + 3 * x + np.random.normal(0, 60, 100)
plt.plot(x, y, "o")



# draw vertical line from (70,100) to (70, 250)
plt.plot([70, 70], [100, 250], 'k-', lw=2)

# draw diagonal line from (70, 90) to (90, 200)
plt.plot([70, 90], [90, 200], 'k-')

plot_url = py.plot_mpl(line, filename='mpl-docs/add-line')
```