Lecture 3
Python Advanced

Jaeyun Kang

Python Advanced - Built-In Function

enumerate

```
for i, name in enumerate(['body', 'foo', 'bar']):
        print(i, name)
```

```
0 body
1 foo
2 bar
```

Python Advanced - Built-In Function

lambda

```
sum = lambda a, b: a+b
print(sum(3,4)) // 7


def sum(a, b):
    return a+b
```

Python Advanced - Built-In Function

lambda

```
myList = [lambda a,b:a+b, lambda a,b:a*b]

print(myList[0](3,4)) // 7
print(myList[1](3,4)) // 12
```

Python Advanced - Built-In Function

list

```
print(list("python"))
//['p', 'y', 't', 'h', 'o', 'n']

print(list((1,2,3)))
//[1, 2, 3]
```

Python Advanced - Built-In Function

range

```
print(list(range(5))) // [0, 1, 2, 3, 4]

print(list(range(5, 10))) // [5, 6, 7, 8, 9]

print(list(range(1, 10, 2))) // [1, 3, 5, 7, 9]
```

Python Advanced - Built-In Function

zip

```
print(list(zip([1, 2, 3], [4, 5, 6])))
// [(1, 4), (2, 5), (3, 6)]

print(list(zip([1, 2, 3], [4, 5, 6], [7, 8, 9])))
// [(1, 4, 7), (2, 5, 8), (3, 6, 9)]

print(list(zip("abc", "def")))
// [('a', 'd'), ('b', 'e'), ('c', 'f')]
```

Python Advanced - Module

random

```
import random
print(random.random()) // 0.5384010330509867 4

print(random.randint(1, 10)) // 6

print(random.randint(1, 55)) // 43
```

Python Advanced - Module

```
# random_pop.py
import random

def random_pop(data):
        number = random.randint(0, len(data)-1)
        return data.pop(number)

if __name__ == "__main__":
        data = [1, 2, 3, 4, 5]
        while data: print(random_pop(data))
```

// ?

Python Advanced - Module

```
import random

data = [1, 2, 3, 4, 5]
random.shuffle(data)

print(data) // ?
```

Numpy

python library installation

1. Anaconda Prompt 관리자 권한으로 실행
2. pip install <패키지명>

Ex) pip install numpy (Already installed in anaconda)

Numpy

```
import numpy as np

a = np.array([1, 2, 3])  # Create a rank 1 array
print(type(a))           # Prints "<class 'numpy.ndarray'>"
print(a.shape)           # Prints "(3,)"
print(a[0], a[1], a[2])  # Prints "1 2 3"
a[0] = 5                 # Change an element of the array
print(a)                 # Prints "[5, 2, 3]"

b = np.array([[1,2,3],[4,5,6]])    # Create a rank 2 array
print(b.shape)                     # Prints "(2, 3)"
print(b[0, 0], b[0, 1], b[1, 0])   # Prints "1 2 4"
```

Numpy

```
import numpy as np

a = np.zeros((2,2))   # Create an array of all zeros
print(a)              # Prints "[[ 0.  0.]
                      #         [ 0.  0.]]"

b = np.ones((1,2))    # Create an array of all ones
print(b)              # Prints "[[ 1.  1.]]"
```

Numpy

```
import numpy as np


d = np.eye(2)          # Create a 2x2 identity matrix
print(d)               # Prints "[[ 1.  0.]
                       #          [ 0.  1.]]"



e = np.random.random((2,2))  # Create an array filled with random values
print(e)                     # Might print "[[ 0.91940167  0.08143941]
                             #              [ 0.68744134  0.87236687]]"
```

Numpy

```
import numpy as np

# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])


# Use slicing to pull out the subarray consisting of the first 2 rows
# and columns 1 and 2; b is the following array of shape (2, 2):
# [[2 3]
#  [6 7]]
b = a[:2, 1:3]
```

Numpy

```
import numpy as np

a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# [[2 3]
#  [6 7]]
b = a[:2, 1:3]

# A slice of an array is a view into the same data, so modifying it
# will modify the original array.
print(a[0, 1])   # Prints "2"
b[0, 0] = 77     # b[0, 0] is the same piece of data as a[0, 1]
print(a[0, 1])   # Prints "77"
```

Numpy

```
import numpy as np

# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])


row_r1 = a[1, :]    # Rank 1 view of the second row of a
row_r2 = a[1:2, :]  # Rank 2 view of the second row of a
print(row_r1, row_r1.shape)  # Prints "[5 6 7 8] (4,)"
print(row_r2, row_r2.shape)  # Prints "[[5 6 7 8]] (1, 4)"
```

Numpy

```
import numpy as np

# Create the following rank 2 array with shape (3, 4)
# [[ 1  2  3  4]
#  [ 5  6  7  8]
#  [ 9 10 11 12]]
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])

# We can make the same distinction when accessing columns of an array:
col_r1 = a[:, 1]
col_r2 = a[:, 1:2]
print(col_r1, col_r1.shape)  # Prints "[ 2  6 10] (3,)"
print(col_r2, col_r2.shape)  # Prints "[[ 2]
                             #         [ 6]
                             #         [10]] (3, 1)"
```

Numpy

```
import numpy as np

x = np.array([1, 2])   # Let numpy choose the datatype
print(x.dtype)         # Prints "int64"

x = np.array([1.0, 2.0])   # Let numpy choose the datatype
print(x.dtype)             # Prints "float64"

x = np.array([1, 2], dtype=np.int64)   # Force a particular datatype
print(x.dtype)                         # Prints "int64"
```

Numpy

```
import numpy as np

x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

# [[ 6.0  8.0]
#  [10.0 12.0]]
print(x + y)
print(np.add(x, y))


# [[-4.0 -4.0]
#  [-4.0 -4.0]]
print(x - y)
print(np.subtract(x, y))
```

Numpy

```
import numpy as np

x = np.array([[1,2],[3,4]], dtype=np.float64)
y = np.array([[5,6],[7,8]], dtype=np.float64)

# [[ 5.0 12.0]
#  [21.0 32.0]]
print(x * y)
print(np.multiply(x, y))


# [[ 0.2        0.33333333]
#  [ 0.42857143  0.5       ]]
print(x / y)
print(np.divide(x, y))
```

Numpy

```
import numpy as np

x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])

v = np.array([9,10])
w = np.array([11, 12])

# Inner product of vectors; both produce 219
print(v.dot(w))
print(np.dot(v, w))
```

Numpy

```
import numpy as np

x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])

v = np.array([9,10])
w = np.array([11, 12])

# Matrix / vector product: both produce the rank 1 array [29 67]
print(x.dot(v))
print(np.dot(x, v))
```

Numpy

```
import numpy as np

x = np.array([[1,2],[3,4]])
y = np.array([[5,6],[7,8]])

v = np.array([9,10])
w = np.array([11, 12])

# Matrix / matrix product; both produce the rank 2 array
# [[19 22]
#  [43 50]]
print(x.dot(y))
print(np.dot(x, y))
```

Numpy

```
import numpy as np

x = np.array([[1,2],[3,4]])

print(np.sum(x))  # Compute sum of all elements; prints "10"
print(np.sum(x, axis=0))
# Compute sum of each column; prints "[4 6]"
print(np.sum(x, axis=1))
# Compute sum of each row; prints "[3 7]"
```

Numpy

```
import numpy as np

x = np.array([[1,2], [3,4]])
print(x)    # Prints "[[1 2]
            #         [3 4]]"
print(x.T)  # Prints "[[1 3]
            #         [2 4]]"


# Note that taking the transpose of a rank 1 array does nothing:
v = np.array([1,2,3])
print(v)    # Prints "[1 2 3]"
print(v.T)  # Prints "[1 2 3]"
```

Numpy

```
import numpy as np

# We will add the vector v to each row of the matrix x,
# storing the result in the matrix y
x = np.array([[1,2,3], [4,5,6], [7,8,9], [10, 11, 12]])
v = np.array([1, 0, 1])
y = x + v  # Add v to each row of x using broadcasting
print(y)  # Prints "[[ 2  2  4]
          #         [ 5  5  7]
          #         [ 8  8 10]
          #         [11 11 13]]"
```

Numpy

1. If the arrays do not have the same rank, prepend the shape of the lower rank array with 1s until both shapes have the same length.

2. The two arrays are said to be *compatible* in a dimension if they have the same size in the dimension, or if one of the arrays has size 1 in that dimension.

3. The arrays can be broadcast together if they are compatible in all dimensions.

4. In any dimension where one array had size 1 and the other array had size greater than 1, the first array behaves as if it were copied along that dimension

Numpy

```
import numpy as np

# Compute outer product of vectors
v = np.array([1,2,3])  # v has shape (3,)
w = np.array([4,5])    # w has shape (2,)

# To compute an outer product, we first reshape v to be a column
# vector of shape (3, 1); we can then broadcast it against w to yield
# an output of shape (3, 2), which is the outer product of v and w:
# [[ 4  5]
#  [ 8 10]
#  [12 15]]
print(np.reshape(v, (3, 1)) * w)
```

Numpy

```
import numpy as np

v = np.array([1,2,3])  # v has shape (3,)

# Add a vector to each row of a matrix
x = np.array([[1,2,3], [4,5,6]])

# x has shape (2, 3) and v has shape (3,) so they broadcast to (2, 3),
# giving the following matrix:
# [[2 4 6]
#  [5 7 9]]
print(x + v)
```

Numpy

```
import numpy as np

w = np.array([4.5])    # w has shape (2,)

x = np.array([[1.2.3], [4.5.6]])


# [[ 5  6  7]
#  [ 9 10 11]]
print((x.T + w).T)

# Another solution is to reshape w to be a column vector of shape (2, 1);
# we can then broadcast it directly against x to produce the same output.
print(x + np.reshape(w, (2, 1)))
```

Numpy

```
import numpy as np


x = np.array([[1,2,3], [4,5,6]])


# Multiply a matrix by a constant:
# x has shape (2, 3). Numpy treats scalars as arrays of shape ();
# these can be broadcast together to shape (2, 3), producing the
# following array:
# [[ 2  4  6]
#  [ 8 10 12]]
print(x * 2)
```

Scipy

```
from scipy.misc import imread, imsave, imresize

# Read an JPEG image into a numpy array
img = imread('assets/cat.jpg')
print(img.dtype, img.shape)  # Prints "uint8 (400, 248, 3)"

# numpy broadcasting
img_tinted = img * [1, 0.95, 0.9]

# Resize the tinted image to be 300 by 300 pixels.
img_tinted = imresize(img_tinted, (300, 300))

# Write the tinted image back to disk
imsave('assets/cat_tinted.jpg', img_tinted)
```

Scipy

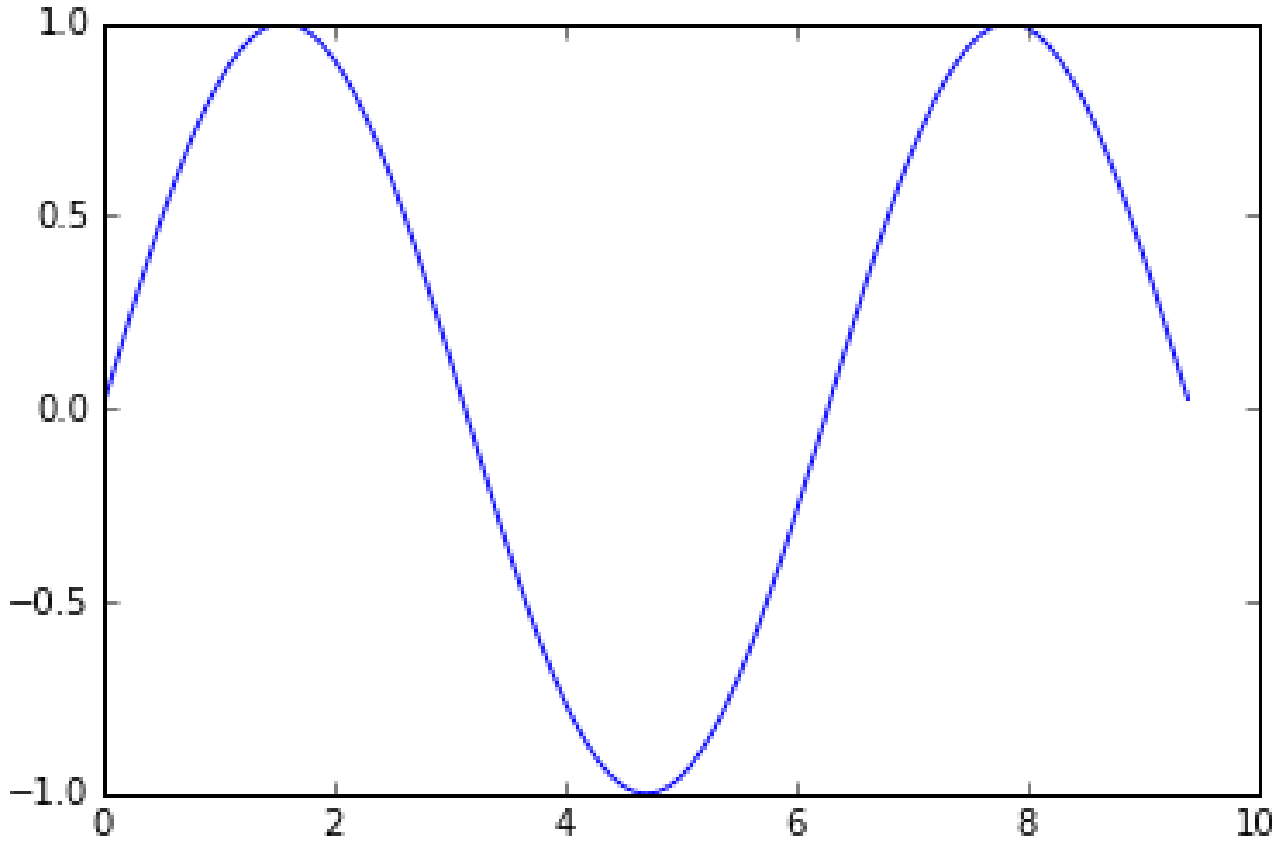
Matplotlib

```
import numpy as np
import matplotlib.pyplot as plt


# Compute the x and y coordinates for points on a sine curve
x = np.arange(0, 3 * np.pi, 0.1)
y = np.sin(x)


# Plot the points using matplotlib
plt.plot(x, y)
plt.show()  # You must call plt.show() to make graphics appear
```

Matplotlib



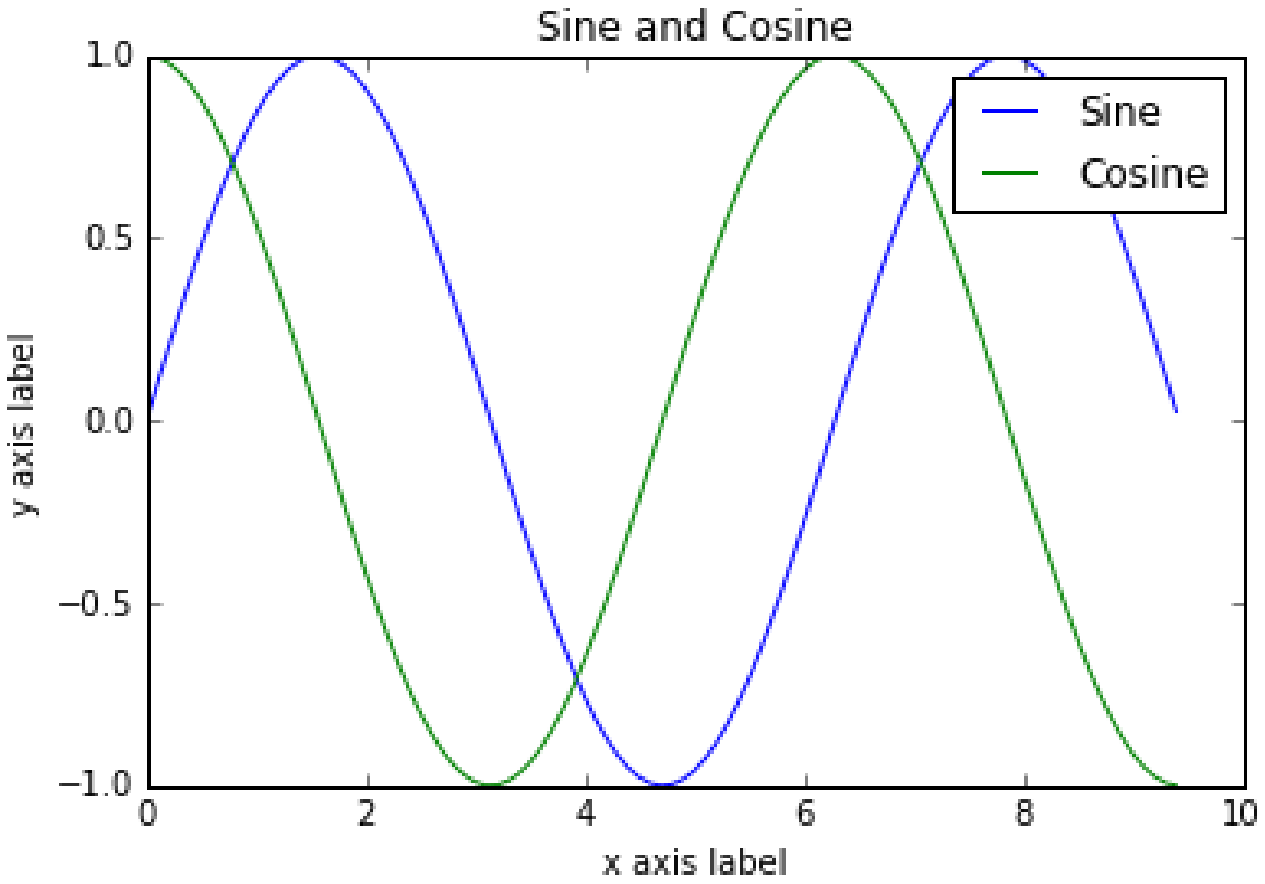
Matplotlib

```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)

# Plot the points using matplotlib
plt.plot(x, y_sin)
plt.plot(x, y_cos)
plt.xlabel('x axis label')
plt.ylabel('y axis label')
plt.title('Sine and Cosine')
plt.legend(['Sine', 'Cosine'])
plt.show()
```

Matplotlib


Sine and Cosine
Sine
Cosine
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10
y axis label
x axis label

Matplotlib

```
import numpy as np
import matplotlib.pyplot as plt


# Compute the x and y coordinates for points on sine and cosine curves
x = np.arange(0, 3 * np.pi, 0.1)
y_sin = np.sin(x)
y_cos = np.cos(x)


# Set up a subplot grid that has height 2 and width 1, and set the first such subplot as active.
plt.subplot(2, 1, 1)

# Make the first plot
plt.plot(x, y_sin)
plt.title('Sine')
```

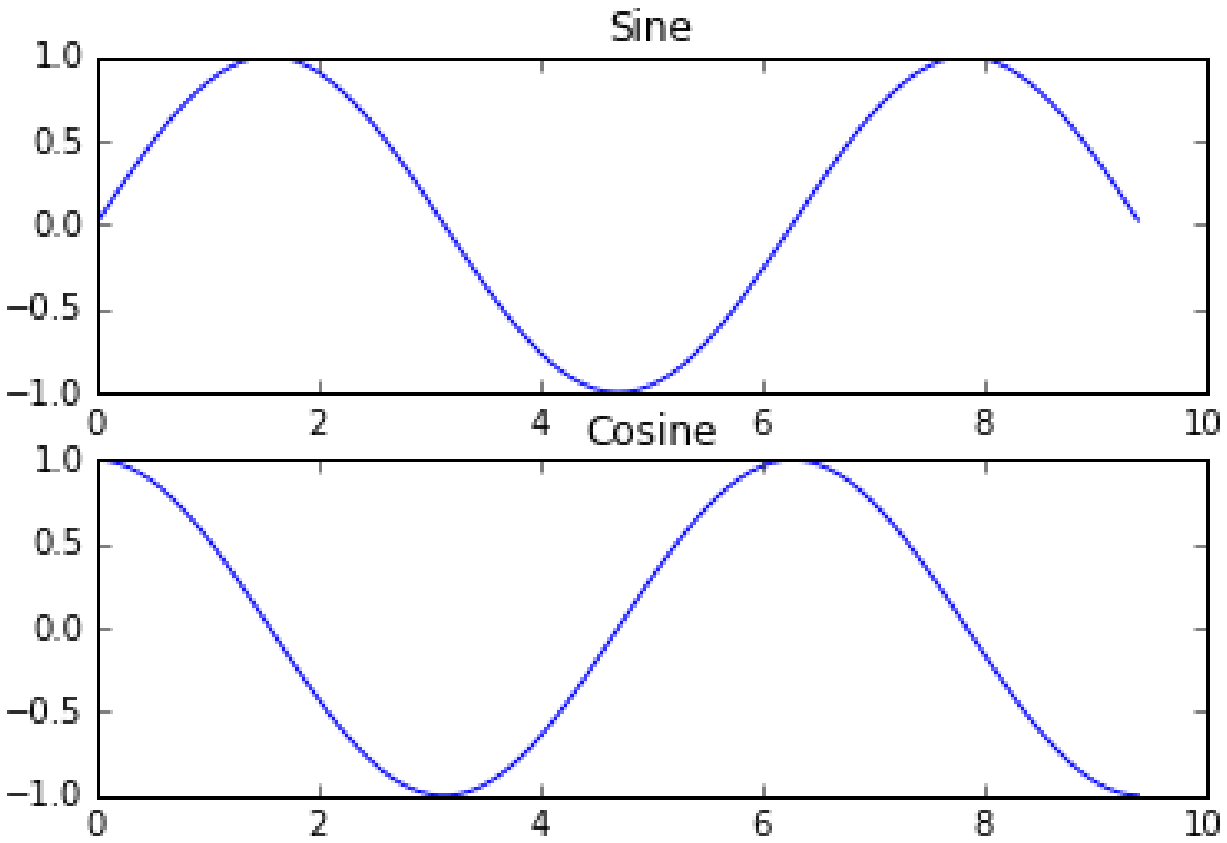
Matplotlib

Continued..

```
# Set the second subplot as active, and make the second plot.
plt.subplot(2, 1, 2)
plt.plot(x, y_cos)
plt.title('Cosine')

# Show the figure.
plt.show()
```

Matplotlib



Matplotlib

```
import numpy as np
from scipy.misc import imread, imresize
import matplotlib.pyplot as plt

img = imread('assets/cat.jpg')
img_tinted = img * [1, 0.95, 0.9]

plt.subplot(1, 2, 1)
plt.imshow(img)

# Show the tinted image
plt.subplot(1, 2, 2)

plt.imshow(np.uint8(img_tinted))
plt.show()
```

Matplotlib


0
50
100
150
200
250
300
350
400
0
50
100
150
200
0
50
100
150
200
250
300
350
400
0
50
100
150
200

Additional Python Image Library

Pillow

http://pillow.readthedocs.io/en/4.2.x/reference/Image.html

Try some examples! (Image open, rotate, display..)

OpenCV

http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_tutorials.html

Try some advanced examples! (RGB to BGR conversion, face detection..)

Question