NumPy Python Module

Computing for Data Analytics (CPSC 4800)

Mourad Bouguerra

mbouguerra@langara.ca

Langara College

June, 2022

Lesson’s Outline

1. **Lesson’s Learning Objectives**

2. **NumPy Overview**
   - Creating NumPy Objects
   - Basic Operations
   - NumPy Functions
   - NumPy Application

Learning Objectives

Learning Objectives

✎ Upon completion of this lesson, you will learn:

- ❒ How to generate NumPy arrays of different dimensions?
- ❒ How to use NumPy functions?
- ❒ How to encode an images as a NumPy array?

Using NumPy

- ❒ A NumPy's object is the homogeneous multidimensional array (list).
  - ➦ all the elements of the same type
- ❒ NumPy dimensions are called axes
- ❒ A NumPy's object is created using the array function.
  - ➦ array(list or tuple, dtype)

NumPy Property	Description
ndim	Return the number of dimensions of an array
shape	Return the shape of an array
size	Return the number of elements of an array
dtype	Return the type of array elements

Using Numpy

Using NumPy Data Types

NumPy Data Types Examples

Data Type	Description
uint8	Unsigned 8-bit integer
int8	Signed 8-bit integer
float32	Signed 32-bit floating-point
float64	Signed 64-bit floating-point

Using Numpy

One Dimensional Array

❒ To create a one-dimensional array

```
import numpy as np
a_1 = np.array((1,2,3))
print(a_1)
# Output
# [1 2 3]
a_1 = np.array([x for x in range(10)],dtype=’uint8’)
print(a_1)
# Output
# [0 1 2 3 4 5 6 7 8 9]
```

Using Numpy

One Dimensional Array

❒ To create a one-dimensional array

```
a_1 = np.array([x for x in range(10)],dtype=’uint8’)
print(f’(Dimension,Shape,Size,Type)=({a_1.ndim},{a_1.shape,a_1.size,a_1.dtype})’)
# Output
# (Dimension,Shape,Size,Type)=(1,((10,), 10, dtype(’uint8’)))
```

Using Numpy

Two Dimensional Array

❒ To create a two-dimensional array

```
a_2 = np.array([[x for x in range(10)],[x for x in range(10)]],dtype=’float32’)
print(a_2)
# Output
# [[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
# [0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]]
```

Using Numpy

Two Dimensional Array

❐ To create a two-dimensional array

```
a_2 = np.array([[x for x in range(10)],[x for x in range(10)]],dtype=’float32’)
print(f’(Dimension,Shape,Size,Type)=({a_2.ndim},{a_2.shape,a_2.size,a_2.dtype})’)
# Output
# (Dimension,Shape,Size,Type)=(2,((2, 10), 20, dtype(’float32’)))
```

Using Numpy

Three Dimensional Array

❒ To create a three-dimensional array

```
a_3 = np.array(np.random.randint(low=0,high=255,size=(255,255,3)),dtype=’uint8’)
print(a_3)
# Output
# [[[164 147  51]
#   [ 44 191 110]
#   [ 89 244  98]
#   ...
```

Using Numpy

Three Dimensional Array

❒ To create a three-dimensional array

```
a_3 =
np.array(np.random.randint(low=0,high=255,size=(300,300,3)),dtype='uint8')


# (Dimension,Shape,Size,Type)=(3,((300, 300, 3),20,dtype('uint8')))
```

Using Numpy

N Dimensional Array

- To create a any-dimensional array
  - use reshape() function
  - Given $size = n$ and $shape = (a, b, c, d)$, then $n = a \times b \times c \times d$

```python
import numpy as np
a_4 = np.arange(1,101).reshape(2,2,5,5)
```

Using Numpy

N Dimensional Array

❐ To create a 4-dimensional array

```
a_4 = np.arange(1,101).reshape(2,2,5,5)
print(f’(Dimension,Shape,Size,Type)=({a_4.ndim},{a_4.shape,a_4.size,a_4.dtype})’)
# Output
# (Dimension,Shape,Size,Type)=(4,((2, 2, 5, 5), 100, dtype(’int32’)))
```

Class Activity

- Generate a **2 dimensional NumPy** array of 100 random elements between 200 and 300 inclusive.
- Check the dimension, shape, size and dtype of the generated **2 dimensional NumPy** array

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

Class Activity

- ✎ To generate random number between 0 and 1 use the following np.random.random()
- ✎
- ✎ What is the output of the following Python code?

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Basic Operations

Basic Operations

- ❒ Arithmetic operators on NumPy arrays apply elementwise
  - ➦ A new NumPy array is created to store the result

```
a = np.arange(start =0,stop=100,step=10)
b = np.arange(10)
a + b
# Output
# array([ 0, 11, 22, 33, 44, 55, 66, 77, 88, 99])
b * 2
# Output
# array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18])
a + b > 60
# Output
# array([False, False, False, False, False, False,  True,  True,  True,True])
```

Using Numpy

Using NumPy Mathematical Functions

- NumPy provides all the mathematical functions optimized for
  - multidimensional arrays
- All NumPy mathematical functions are applied elementwise

```
a = np.random.randn(4,5)
np.abs(a)
np.floor(a)
np.ceil(a)
```

Using Numpy

Using Numpy Functions

```
scores = np.random.randint(low=40,high=100,size=(35,2))
np.median(scores,axis=0)
# output
# array([76., 74.])
np.median(scores,axis=1)
# output
# array([54. , 74.5, 81. , 73.5, 81. , 67. , 76. , 70.5, 56.5, 68.5, 67. ,
#       83.5, 82.5, 74.5, 78. , 82.5, 80. , 90. , 55. , 93.5, 57.5, 60.5,
#       52.5, 87. , 75. , 73. , 60. , 71.5, 73.5, 78.5, 63.5, 79.5, 73. ,
#       54.5, 75.5])
```

Using NumPy Functions

❒ In addition to all mathematical functions, NumPy's the following satistical functions.

NumPy Function	Description
median	Compute the median along the specified dimension
mean	Compute the arithmetic mean along the specified dimension
std	Compute the standard deviation along the specified dimension
var	Compute the variance along the specified dimension
quantile	Compute the q^{th} quantile along the specified dimension

Using Numpy

Using Numpy Functions

```
scores = np.random.randint(low=40,high=100,size=(35,2))
np.median(scores,axis=0)
# output
# array([76., 74.])
np.median(scores,axis=1)
# output
# array([54. , 74.5, 81. , 73.5, 81. , 67. , 76. , 70.5, 56.5, 68.5, 67. ,
#        83.5, 82.5, 74.5, 78. , 82.5, 80. , 90. , 55. , 93.5, 57.5, 60.5,
#        52.5, 87. , 75. , 73. , 60. , 71.5, 73.5, 78.5, 63.5, 79.5, 73. ,
#        54.5, 75.5])
```

Class Activity

- ✎ Generate a **2 dimensional NumPy** array of 100 random elements between 30 and 95 inclusive.
- ✎ Compute the following statistics along the two dimensions
  - ➦ Mean
  - ➦ Median
  - ➦ Standard Deviation
  - ➦ Variance
  - ➦ Minimum
  - ➦ Maximum

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

RGB Model

RGB

- ✎ *In a RGB model*
  - ✏ *the combination of* **red**, **green**, *and* **blue**
  - ✏ *produces colors in the visible spectrum*
- ✎ *a pixel is implemented by:*
  1. *true color, or 24-bit (3 bytes)*
     - ✔ *Each color will be stored in 8-bit (1-byte)*
     - ✔ $256 \times 256 \times 256 = 16,777,216$ *possible colors*
  2. *32-bit (4 bytes)*
     - ✔ *The fourth byte stores the Alpha value*
- ✎ *The most common model in computer graphics*

RGB Model

RGB

✎ *In a RGB model*

- ✏ *the combination of* **red**, **green**, *and* **blue**
- ✏ *produces colors in the visible spectrum*

✎ *a pixel is implemented by:*

① *true color, or 24-bit (3 bytes)*

- ❒ **rgb(0.5,0.75,0.32)**

② *32-bit (4 bytes)*

- ❒ **rgba(0.5,0.75,0.32,0.4)**

RGB Basic Colors

✎ Each color is assigned a value between 0 (0.0) and 255 (1.0)

(red,green,blue)	Hexadecimal	Color Name	Display
(0, 0, 0)	#000000	black	⬛
(0, 0, 255)	#0000*FF*	blue	🟦
(0, 255, 0)	#00*FF*00	green	🟩
(0, 255, 255)	#00*FFFF*	cyan	🟦
(255, 0, 0)	#*FF*0000	red	🟥
(255, 0, 255)	#*FF*00*FF*	magenta	🟪
(255, 255, 0)	#*FFFF*00	yellow	🟨
(255, 255, 255)	#*FFFFFF*	white	⬜

RGB Basic Colors

✎ Each color is assigned a value between 0 (0.0) and 255 (1.0)

(red,green,blue)	Hexadecimal	Color Name	Display
(0.0, 0.0, 0.0)	#000000	black	⬛
(0.0, 0.0, 1.0)	#0000FF	blue	🟦
(0.0, 1.0, 0.0)	#00FF00	green	🟩
(0.0, 1.0, 1.0)	#00FFFF	cyan	🟦
(1.0, 0.0, 0.0)	#FF0000	red	🟥
(1.0, 0.0, 1.0)	#FF00FF	magenta	🟪
(1.0, 1.0, 0.0)	#FFFF00	yellow	🟨
(1.0, 1.0, 1.0)	#FFFFFF	white	⬜

RGB Gray Colors

- Gray level by setting $red = green = blue$

(red,green,blue)	Hexadecimal	Color Name	Display
$(51, 51, 51)$	$\#333333$	Dark Gray	⬛
$(127, 127, 127)$	$\#7F7F7F$	Gray	▩
$(222, 222, 222)$	$\#DEDEDE$	Gray	⬜

NumPy Application

NumPy Image Encoding

- Many machine learning in Python require the data to be encoded as
  - ➦ a NumPy array
- An image data can encode as
  - ➦ a NumPy array

```python
import PIL as pil
from PIL import Image
image = Image.open('data/myself.jpg')
image
```

NumPy Application

NumPy Image Encoding

❒ To display image properties

```python
print(image.format)
print(image.size)
print(image.mode)
```

NumPy Application

NumPy Image Encoding

- To convert the image to a grayscale image

```
image_grayscale = image.convert('L')
image_grayscale
```

NumPy Application

NumPy Image Encoding

❒ To save the image

```
image_grayscale.save('data/myself-grayscale.jpg')
```

NumPy Application

NumPy Image Encoding

❒ To convert the image to a NumPy array

```
image_np = np.asarray(image)
print(f’Dimension = {image_np.ndim}’)
print(f’Dimension = {image_np.shape}’)
print(f’Size = {image_np.size}’)
print(f’Dimension = {image_np.dtype}’)
```

Class Activity

- ✎ Using a color image
  - ➦ Load the image and display the image
  - ➦ Display the image properties
  - ➦ Convert the image into a grayscale
  - ➦ Save the grayscale image
  - ➦ Convert both images to NumPy arrays and display their properties

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn