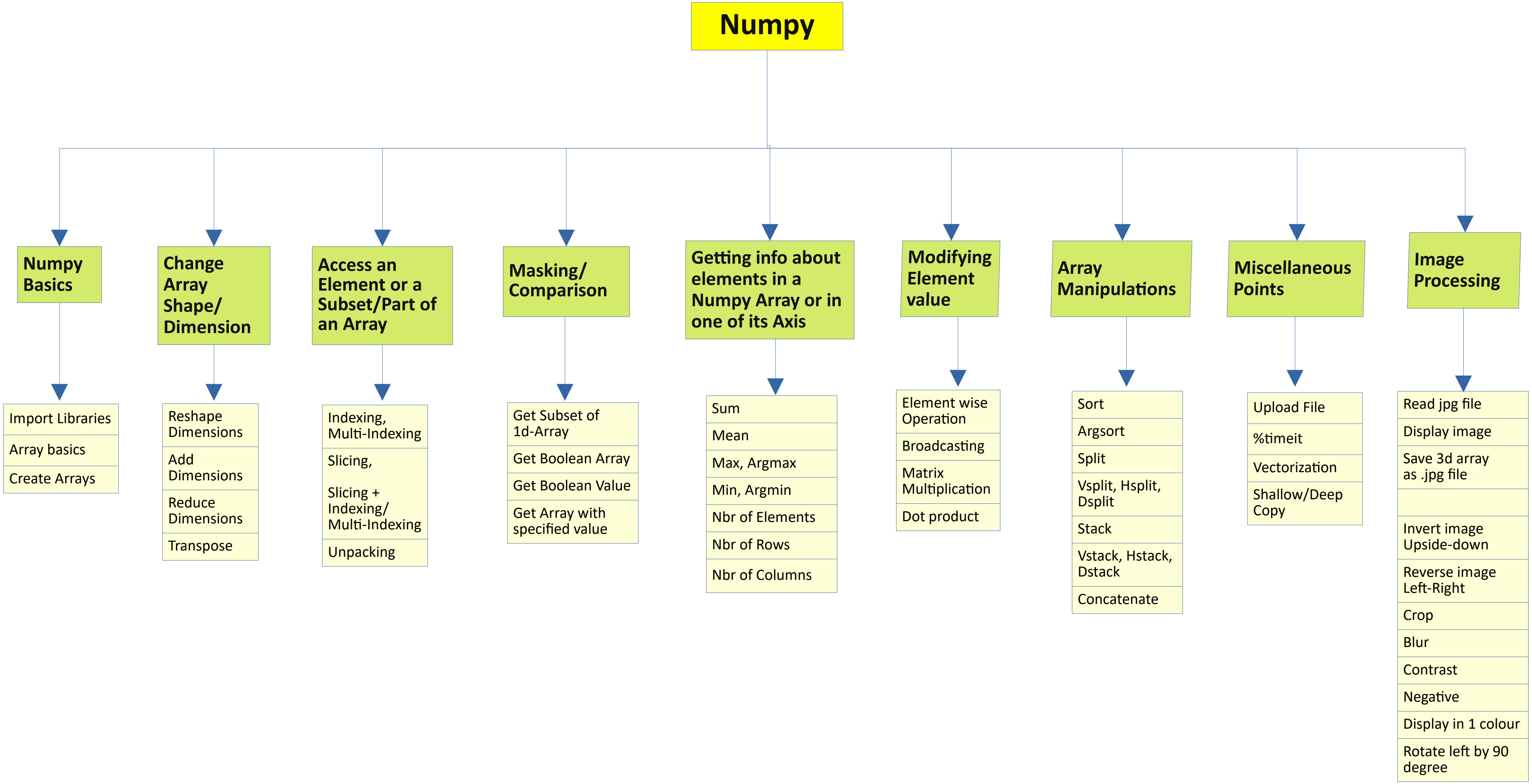

Numpy

Numpy Basics

- Import Libraries
- Array basics
- Create Arrays

Change Array Shape/ Dimension

- Reshape Dimensions
- Add Dimensions
- Reduce Dimensions
- Transpose

Access an Element or a Subset/Part of an Array

- Indexing, Multi-Indexing
- Slicing,
- Slicing + Indexing/ Multi-Indexing
- Unpacking

Masking/ Comparison

- Get Subset of 1d-Array
- Get Boolean Array
- Get Boolean Value
- Get Array with specified value

Getting info about elements in a Numpy Array or in one of its Axis

- Sum
- Mean
- Max, Argmax
- Min, Argmin
- Nbr of Elements
- Nbr of Rows
- Nbr of Columns

Modifying Element value

- Element wise Operation
- Broadcasting
- Matrix Multiplication
- Dot product

Array Manipulations

- Sort
- Argsort
- Split
- Vsplit, Hsplit, Dsplit
- Stack
- Vstack, Hstack, Dstack
- Concatenate

Miscellaneous Points

- Upload File
- %timeit
- Vectorization
- Shallow/Deep Copy

Image Processing

- Read jpg file
- Display image
- Save 3d array as .jpg file
- Invert image Upside-down
- Reverse image Left-Right
- Crop
- Blur
- Contrast
- Negative
- Display in 1 colour
- Rotate left by 90 degree

Numpy Basics

Libraries, Array basics

import numpy as np
import matplotlib.pyplot as plt
 	
import timeit
import copy as cp

b.dtype
b.ndim or np.ndim(b)
b.shape or np.shape(b)

Create Array

b = np.array(a) *#a is a List*
b = np.array(a, dtype = "float")
b = np.array(range(1,5))

b = np.arange (5)
b = np.arange (1, 5, 0.5)

b = np.ones(12)
b = np.ones((5,5),dtype='int')

Change Array Shape/Dimension

Reshape Dimensions

b = a.reshape(2, 4,-7)	#Auto reshape is done when negative number is given

Add Dimensions

b = a[np.newaxis, np.newaxis]	#Adds 2 dimensions on Left
b = a[: , np.newaxis]	#Adds dimension on Right
b = a [: , : , np.newaxis, :]	#Adds dimension in Middle
b = a[: , None]	*'None' can be used like newaxis to add dimension on Left, Right or Middle.*

Reduce Dimensions

b = np.squeeze(a, axis=N)
b = np.squeeze(a)

b = a.flatten()

Transpose

b = a.T

Access an Element or a Subset/Part of an Array

Indexing, Multi-Indexing

b = a[2]	Access element in 1d Array
b = a[[2, 6, 4]]	Multi Indexing Using a List
b = a[x]	x is a 1d array, e.g. [2 6 4]
a [i] [j] ; a [i, j]	Access element in 2d Array
a [i] [j] [k] ; a [i, j, k]	Access element in 3d/nd Array
b = a[[0,1], [2,3], [1,1]]	Multi Indexing in 3d/nd-Array Gets 2 elements - a[0][2][1] & a[1][3][1]

Slicing, Slicing + Indexing/ Multi-Indexing

b[-5:-1:1]	Slicing 1d Array (same as List)
a[0:3, 0:1, 0:2]	Slicing in 3d/nd Array
b = a[0:2, 1]	Slicing + Accessing on a 2d Array => Gives a 1d array
b = a[2:3, 1, 1]	Slicing + Accessing on a 3d Array => Gives a 2d array
b = a[[2, 1], 0:2] #a.shape = (3, 3)	Slicing **+** Multi-Indexing some elements => Gives a Subset of Array
b = a[[2, 1, 0], 0:2] #a.shape = (3, 3)	Slicing **+** Multi-Indexing all elements => Reorders the rows

Unpacking

a, b, c, d = M (M.shape = 4 x n)	Unpack 2d array. Each row goes to a variable
a, b = M (M.shape = 2 x n1 x n2)	Unpack 3d array. a, b become 2d arrays of shape => n1, n2.
x, y = a.shape	Unpacking to put Array rows/cols into variables

Masking/Comparison on an Array to get a Subset, Boolean Array/Value

Masking, Comparison

For only 1d-Arrays	
b = a[a < 6]	Create Subset of 1d-Array. Use same array for masking
b = a[c < 6]	Create Subset of 1d-Array. Use different array of same shape for masking
b = a[c]	Create Subset of 1d Array using Boolean list of same shape for masking.
For nd-Arrays	
b = a < 6	Create nd-Boolean Array. Use array & a number for masking.
b = a < b	Create nd-Boolean Array. Use 2 arrays for masking
b = np.any(a < b)	ANY => Get a Boolean value as result
b = np.all(a < b)	ALL => Get a Boolean value as result
b = np.where(a < 0, 'neg', 'pos')	WHERE => Get same shape array with specified values

Getting info about elements in a Numpy Array or in one of its Axis

Sum, Mean, Min, Max, Arg, Length	

Sum	b = a.sum() or b = np.sum(a)
	b = np.sum (a, axis = 1)

Mean	b = a.mean() or b = np.mean(a)
	b = np.mean (a, axis = 1)

Min	b = a.min() or b =np.min(a)
	b = np.min (a, axis = 1)

Max	b = a.max() or b = np.max(a)
	b = np.min (a, axis = 0)

argmin	b = np.argmax(a) ;
	b = np.argmax(a, axis =0)

argmax	b = np.argmin(a) ;
	b = np.argmin(a, axis =1)

len(a)	Number of Elements; when a is a 1d Array
len(a)	Number of Rows; when a is a 2d Array
len (a [0])	Number of Columns; when a is a 2d Array

Modifying Element value

Element wise Operations	
a + b a - b a * b a**b a/b OR a + 2 a – 2 a * 2 a**2 a/2	between 2 arrays of same shape Or between an array and a number
np.add(a,b) or np.add(a,3) Other numpy functions are -> subtract, multiply, divide, power	between 2 arrays of same shape Or between an array and a number
Slicing + Assigning a Value	a [: , : , 0 : 2] = 5 OR a [: , : , 0 : -1] = 5 *(negative index)*
Slicing + Multi Indexing + Assigning a Value	a[: , : , [0, 1, 4]] = 5
To repeat an array rows M times and columns N times. Like **Manual Broadcasting**	b = np.tile(a, (M, N))
Matrix Multiplication	a @ b OR np.matmul(a, b)
dot product	**np.dot(a, b)**

Miscellaneous Points

Upload file, Vectorization, %timeit	
Upload a Text file to Jupyter Notebook	a = np.loadtxt(‘FileName.txt’, dtype = ‘int’)
List Comprehension (Create new List from a List)	L = [i**2 for i in a]
%timeit	%timeit [i**2 for i in a] *(a is a List)* %timeit a**2 *(a is an Array)*
Vectorization of a function	b = np.vectorize(function_name)

Shallow Copy, Deep Copy	
Shallow Copy	a.view( ), reshape, assignment, slicing
Deep Copy	a.copy(), Any mathematical operation (Element wise operation), Masking/Comparison
How to check if Deep or Shallow Copy happened	np.shares_memory(a, b) => (True/False)

Array Manipulations

Sort, Split, Stack, Concatenate	
sort	a = np.sort(a, axis = 2)
argsort	b = np.argsort(a, axis = 1)
Split (number of sub-arrays specified)	b1, b2 = np.split(a, 2, axis=1) *# a.shape = (3, 4)*
Split (Slicing specified)	b1, b2, b3 = np.split(a, [2, 3], axis = 1) *# a.shape = (3, 4)*
vsplit (always axis = 1)	b1, b2 = np.vsplit(a, 2)
hsplit (always axis = 0)	b1, b2 = np.hsplit(a, 2)
dsplit (always axis = 2)	b1, b2 = np.dsplit(a, 2)
stack	b = np.stack((a,a), axis=0)
vstack always axis = 1	b = np.vstack((a,a))
hstack always axis = 0	b = np.hstack((a,a))
dstack always axis = 2	b = np.dstack((a,a))
concatenate	b = np.concatenate((a,a), axis = 0)

Image Processing

Basics	
Read a jpg file	a = np.array(plt.imread('abc.jpg'))
Display an image	plt.imshow(a)
Save 3d Array as jpg/jpeg file	plt.imsave('abc.jpeg', a) or abc.jpg

Image Manipulations	
Invert image upside-down (row invert)	plt.imshow (a[: : -1, : , :])
Reverse left-right of an image (column invert)	plt.imshow (a[: , : :-1 , :])
Crop Image	plt.imshow (a[100:460 , 100:460 , :])
Blur Image	plt.imshow (a[::10 , ::10, :])
Contrast Image	plt.imshow (np.where(a >150, 255, 0))
Negative of Image	plt.imshow (255 - a)
Display image with single colour	a[: , :, [0, 2]] = 0 ; plt.imshow(a)

Rotate image 90 degree to left	a = np.transpose(a, (1, 0, 2)) ;
	plt.imshow(b) *(1st and 2nd dimensions get swapped)*

Usual Transpose of 3d Array	b = a.T Then b => a(2, 1, 0) *(1st and 3rd dimensions get swapped)*