Day 2

Andrew NG do the new courses

→ numpy
→ need for numpy
→ images
→ Load camera feed
→ opencv

Numpy

→ usage of arrays
→ python uses lists

→ Native level is the OS level
So c & c++ work at native level.
(read/watch python webinar 2)

★ using jupyterlab

```
a = 6
a = 8
```

Here we did not change the value of a, we just assigned it to something else.

```
import numpy as np

Lists = [3, 6, 7, 1]
n = np.array(Lists)
type(n)
# numpy.ndarray

you can also put a set instead of list.
np.array({2, 5, 7})
# array({2, 5, 7}, dtype = object)

d = {'1': 'apple', '2': 'mango'}
np.array(d)
# array({'1': 'apple', '2': 'mango'},
        dtyp = object)

twod = np.array([[1, 2], [3, 4]])
twod
array([[1, 2],
       [3, 4]])
But if there were diff number of elements
then it would just form lists.
eg  t = np.array([[1,2], [3,4,5]])
t
array([list([1, 2]), list([3,4,5])],
      dtype = object)
```

dtype → type of array

* y = np.array([1, 2, "77"])

y

O/P: array(['1', '2', '77'], dtype = <U21)

↑ unicode

note → elements change to string not integers, because strings are more dominating.

Check → quickstart tutorial: numpy

- n.shape → gives shape of array
- n.size → total no. of items
- list(range(9, 16, 2)) → [9, 11, 13, 15]

starting inclusive — ending exclusive — jumping points

- arange → range implementation of np

np.arange(7, 12, 2)

O/P: array([7, 9, 11])

If you do not remember the format or use of a function, use:

np.arange? // The ? gives the documentation for np.arange

arange → gives a range of any datatype.

```
np.arange(40).reshape(8,5)
```

40 elements — 8 rows — 5 columns

The no of row x col = total no. of elements you have in the array.

```
arr1 = np.arange(40) # array([0,1....,39])
arr2 = arr1.reshape(8,5)
arr1[0] = 1000
arr2
# array([[1000, 1, 2, 3, 4],
         [5 ...          ]
                  :    ])
```

Value of arr[0] will change.

* Python itself is call by value but when you go into the code working at nature level, it is using call by reference.

arr1 vs arr2
False
Why? values → same
dim → different

.IMAGE
from PIL import Image
import numpy as np
pil = Image.open("thor.jpg")
All the pixels in the pic are rgb pixels (red green blue)

arr = np.array(pil) # depict pic as array
arr.shape
(246, 220, 3)
row / column / depth [for png, this will be 4]

import matplotlib.pyplot as plt
# plt.imshow(a) It'll plot a in a graph

red = [:, :, 0]
plt.imshow(red)
It'll show only the red part
So it basically only show elements having that colour.

★ you can also specify if you only want to view a part of the page.

w = arr.copy() → creates a copy of the image

w[:, :, [1,2]].shape
gives shape of image

w[:, :, [1,2]] = 0
Now, the image will switch to red, & the part which was red, will be visible

use?
helps to know what images consist of.

g = arr.mean(axis=2)
plt.imshow(g, cmap = "gray")

It'll show the image in a gray scale
If we do not use cmap, it'll use multicolour grayscale.

axis = 0 → across row
= 1 → accross column
= 2 → across the depth

pip install opencv-python

```
from PIL import Image
import numpy as np
import cv2 → for image processing
import matplotlib.pyplot as plt
```

```
frame = cv2.imread("thor.jpg")
```
↑ this reads bgr instead of rgb.

To convert

```
rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

```
cv2.CascadeClassifier()
```
↖ helps us detect faces

one feature after another like a sequence

```
# explore opencv.
# check out haar cascade algorithms
```
↓

```
① classifier = cv2.CascadeClassifier("_____.xml")
② faces = classifier.detectMultiScale(frame)
```

o/p : it'll print 4 values : x, y, w, h
(w = width, h = height)

```
x, y, w, h = faces[0]
```

gives a part of the image if we do plt.imshow(rgb)

rgb [y : y+h , x : x+w]

faces holds a list of faces & each faces holds 4 values.

out = cv2.rectangle(rgb, (x, y), (x+w, y+h), (0, 0, 255), 4)

③ thickness

plt.imshow(out)

This will focus on the part specified by cv2.rectangle

Video capturing

```
import cv2
cap = cv2.VideoCapture(0)        # which cam I am using
add ① & ②
While True :

    ret , frame = cap.read()     # add ②
    if ret :
    cv2.imshow("My window", frame)

    key = cv2.waitkey(1)         # If this is high, it'll lag a lot.
```

unicode value

```
if key == ord("q"):
    break
```

→ this basically means that if you click on it, it'll break out of the code.

```
cap.release()
cv2.destroyAllWindows()
```

* for face in faces:

```
    x, y, w, h = face
    rgb = [y:y+h, x:x+w]
    frame = (3)
    cv2.imshow('Video', frame).
```

Now RUN.

It'll detect faces

check this out

* how to put image over image

* rules for broadcasting

→ Shape should be same, start checking from right to left (←)

→ if it has 1 in it, then it will work.