**PROJECT1\_DL\_CV - SELF DRIVING CAR**

**COURSE OVERVIEW**

1. Use Computer Vision techniques via OpenCV to identify lane lines for a self-driving car.
2. Learn to train a Perceptron-based Neural Network to classify between binary classes.
3. Learn to train Convolutional Neural Networks to identify between various traffic signs.
4. Train Deep Neural Networks to fit complex datasets.
5. Master Keras, a power Neural Network library written in Python.
6. Build and train a fully functional self driving car to drive on its own

**INSTALLING ANACONDA**

1. <https://problemsolvingwithpython.com/01-Orientation/01.05-Installing-Anaconda-on-Linux/>
2. The above link can be used to install anaconda properly on linux
3. After installing anaconda if conda command does not work on the terminal which is the anaconda prompt(windows equivalent in linux) we need to set the path as follows:

* source ~/.bashrc
* export PATH=/home/ark/anaconda3/bin:$PATH
* conda list
* Setting Up Anaconda Environments:-
* conda search "^python$"

1. NOTE:to run anaconda either anaconda-navigator or in anaconda3/bin use./anaconda-navigator
2. Installing open-cv:

* Source activate base =>all the packages will be installed here
* conda install -c menpo opencv

**TEXT EDITOR:** VISUAL STUDIO CODE

PART 1 : PYTHON BASICS

**PART 2: NUMPY INTRO**

Numpy is a highly used package in python specially in the field of data-science and has many capabilities like powerful N-dimensional arrays,strong broadcasting tools,tools for C,C++ integration and various linear transform and fourier transform and random number capabilities.

1. arange in numpy (gives an array) => range in python (gives a list)

**PART3 : OPEN CV FOR LANE DETECTION**

Gradient: Measure of change in brightness over adjacent pixels

Edge:rapid change in brightness

Step1->convert image to greyscale

A colored picture has pixels which are a combination of 3 intensities-RGB;while a black and white(grey scale) image has only 1 channel ,each pixels with only 1 intensity value ranging from 0 to 256->thus making it easy to compute!

Step 2-> smoothening the image by reducing noise using gaussian blur