INTRO TO SELF-DRIVING CARS NANODEGREE SYLLABUS

Course 1: Bayesian Thinking

In this course you will learn a mathematical framework known as Bayesian Inference. This is the same framework that underlies a self-driving car's understanding of itself and the world around it. It's what allows a car to use unreliable sensor data to achieve surprisingly accurate estimates of its own location in the world (known as localization). It also underlies the tracking algorithms that self-driving cars use to predict what other traffic on the road will do in the future. By the end of this course you will not only understand Bayesian Inference, you will be able to see the world the way a self-driving car does.

Project 0: Joy Ride
Jump into writing code that controls a simulated vehicle. Send throttle and steering commands to the car to try and get it to navigate around a test track.

Project 1: 2D Histogram Filter in Python
In this first project, you will write the `sense` and `move` functions for a 2-dimensional histogram filter in Python.

Course 2: Working with Matrices

This course will focus on two tools which are vital to self-driving car engineers: **object oriented programming** and **linear algebra**.

Object Oriented Programming (OOP) is an approach to programming that is especially useful when the things we are modeling in our code have obvious real-world counterparts (as is often the case when writing code for something as real as a car). In this course you'll learn the basics of OOP with a focus on how to use **classes** which others have created.

Linear Algebra is the mathematics of matrices. For our purposes in this course we will treat it as a tool. The main goal will be to gain a basic proficiency with this powerful tool by making the notation of linear algebra approachable and understandable. With this tool in hand you'll be able to implement any of the numerous algorithms you'll encounter as you continue your self driving car career (including the ubiquitous Kalman Filter).

The goal of this course is very pragmatic: by the end you will know how to **use** the tools— deeper theoretical understanding is something you will acquire gradually as you continue your journey through this Nanodegree program, and the rest of your career.

Project 2: Implement a Matrix Class
In this project you'll practice using your object oriented programming and matrix math skills by filling out the methods in a partially-completed `Matrix` class.

Course 3: C++ Basics

C++ is the language of self-driving cars, and code written in this language can run incredibly fast. This course will be the first step in a long and rewarding journey towards C++ expertise. The goal for this course is translation: given a program written in Python, you will be able to translate it into C++.

Project 3: Translate Python to C++
In this project you'll apply your knowledge of C++ syntax by translating the Histogram Filter code from the first course into C++.

Course 4: Performance Programming in C++

In C++ basics you focused on the bare minimum required to write code that runs correctly. In this course you will start to explore how to write **good** code that runs correctly. We'll focus primarily on the low level language features of C++ which can make C++ fast, but we'll also discuss other best practices as well.

Project 4: Performant C++
A self-driving car can't afford to waste any cycles or memory unnecessarily. In this project you'll take some functioning (but inefficient) C++ code and optimize it.

Course 5: Navigating Complex Data Structures

What data structure should I use to model **this** relationship? What algorithm will accomplish **that** goal? These are the questions that self-driving car engineers think about on a daily basis. Algorithmic thinking is a skill you'll continue to refine throughout your programming career. In this course you'll focus on some of the data structures and algorithms that show up most frequently in self-driving cars.

Project 5: Planning an Optimal Path
You turn on your self-driving car, buckle up, and enter a destination. Navigating from A → B is not an easy problem. In this project you'll use your knowledge of data structures (in particular, **graph** data

structures) and search algorithms to write an algorithm which uses a map and traffic information to find the quickest route between two points.

Course 6: Visualizing Calculus and Controls

It's sometimes convenient to represent the world as a **discrete** grid of cells. But that isn't quite right. And when it comes time to actually issue control commands about steering, throttle, and braking we have to stop pretending the world is discrete because, in reality, the world is **continuous.**

In this course you'll learn the basics of calculus, the mathematics of continuity. To visualize the continuous trajectories of the real-world you'll also learn to use some of Python's most popular visualization libraries.

Project 6: Trajectory Visualizer
As a self-driving car engineer, a lot of the code you write involves simulation, visualization, testing, and debugging. In this project you'll write a visualization tool that will let you visualize the continuous trajectories that come from various search and control algorithms.

Course 7: Machine Learning and Computer Vision

How do you teach a computer the difference between a photo of a car and a photo of a human? As humans we can make the distinction without trying, but teaching this intuition to a computer is much more work. In this course you'll learn how a computer sees an image and how we can use machine learning to teach a computer to identify images programmatically.

Project 7: Image Classifier from Scratch
In this project you'll build an image classifier from scratch. When you're done you'll have an algorithm that can reliably classify an image as "pedestrian" or "car."