UDACITY

NANODEGREE PROGRAM SYLLABUS

# Intro to Self-Driving Cars

# Overview

In this program, you'll sharpen your Python skills, apply C++, apply matrices and calculus in code, and touch on computer vision and machine learning. These concepts will be applied to solving self-driving car problems. At the end, you'll be ready for our **Self-Driving Car Engineer** Nanodegree program!

Prerequisites: Some experience with programming — writing short scripts in any programming language — and algebra is required. You should feel comfortable reading and modifying code that you are given, but it's all right if solving problems in code is still challenging.

**Estimated Time**:
4 Months at
10 hours / week

**Prerequisites**:
Programming &
Mathematics

**Flexible Learning**:
Self-paced, so
you can learn on
the schedule that
works best for you

**Need Help?**
udacity.com/advisor
Discuss this program
with an enrollment
advisor.

# 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.

**Course Project**
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.

**Course Project**
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.

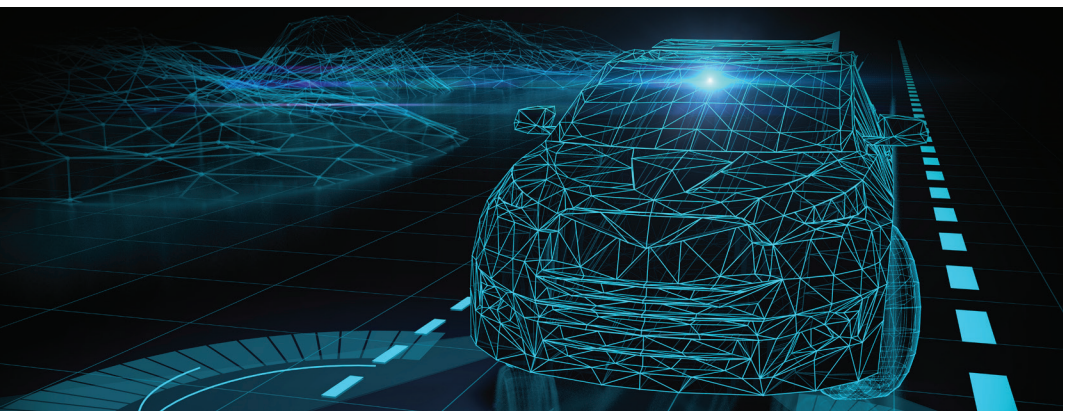
Object Oriented Programming (OOP) is an approach to programming that is especially useful when the things we are modeling in our code have obvious real-world counterparts (as is often the case when writing code for something as real as a car). In this course you'll learn the basics of OOP with a focus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

**Course Project**
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++.

**Course Project**
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.

**Course Project**
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.

**Course Project**
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.

**Course Project**
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.

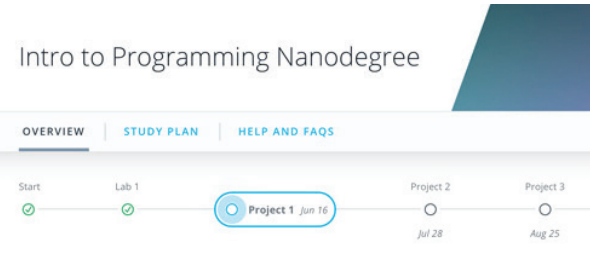
# Course 7: Machine Learning and Computer Vision

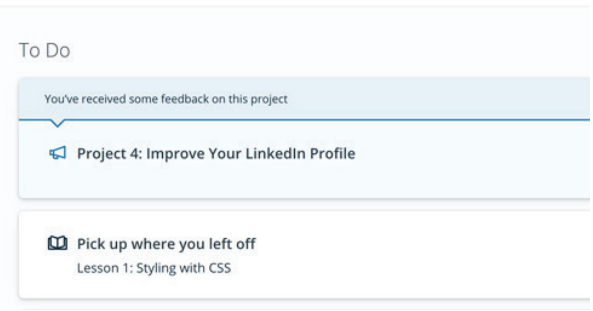
How do you teach a computer the difference between a photo of a car and a photo of a human? As humans we can make the distinction without trying, but teaching this intuition to a computer is much more work. In this course, you'll learn how a computer sees an image and how we can use machine learning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

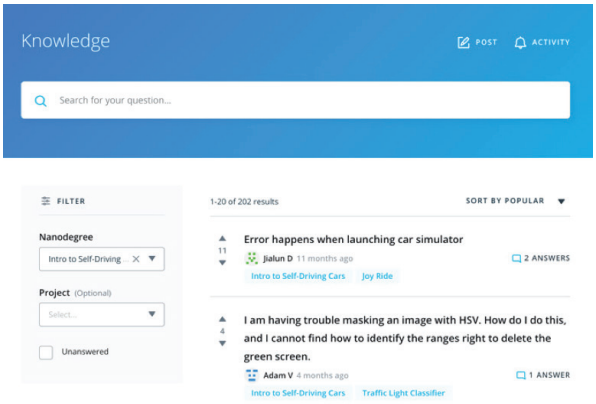
**Course Project**
Image Classifier from Scratch

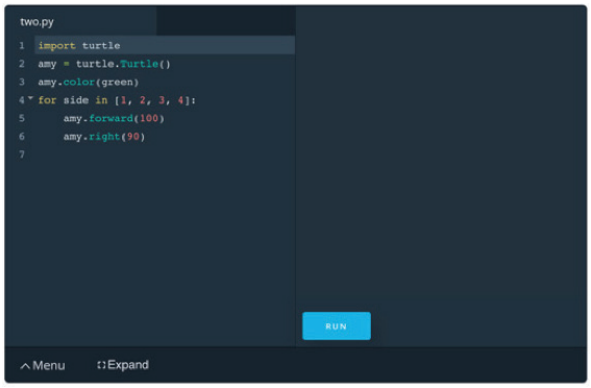
In this project you'll build an image classifier from scratch. When you're done you'll have an algorithm that can reliably classify an image as "pedestrian" or "car.

# Our Classroom Experience




### REAL-WORLD PROJECTS

Build your skills through industry-relevant projects. Get personalized feedback from our network of 900+ project reviewers. Our simple interface makes it easy to submit your projects as often as you need and receive unlimited feedback on your work.

### KNOWLEDGE

Find answers to your questions with Knowledge, our proprietary wiki. Search questions asked by other students, connect with technical mentors, and discover in real-time how to solve the challenges that you encounter.

### STUDENT HUB

Leverage the power of community through a simple, yet powerful chat interface built within the classroom. Use Student Hub to connect with your fellow students in your Executive Program.

### WORKSPACES

See your code in action. Check the output and quality of your code by running them on workspaces that are a part of our classroom.

### QUIZZES

Check your understanding of concepts learned in the program by answering simple and auto-graded quizzes. Easily go back to the lessons to brush up on concepts anytime you get an answer wrong.

### CUSTOM STUDY PLANS

Preschedule your study times and save them to your personal calendar to create a custom study plan. Program regular reminders to keep track of your progress toward your goals and completion of your program.

### PROGRESS TRACKER

Stay on track to complete your Nanodegree program with useful milestone reminders.

# Learn with the Best

## Sebastian Thrun

UDACITY PRESIDENT

Scientist, educator, inventor, and entrepreneur, Sebastian led the self-driving car project at Google X and founded Udacity, whose mission is to democratize education by providing lifelong, on-demand learning to millions of students around the world.

## Andy Brown

CURRICULUM LEAD

Andy has a bachelor's degree in physics from MIT, and taught himself to program after college (mostly with Udacity courses). He has been helping Udacity make incredible educational experiences since the early days of the company.

## Cezanne Camacho

COURSE DEVELOPER

Cezanne is an expert in computer vision with an M.S. in Electrical Engineering from Stanford University. Inspired by anyone with the drive and imagination to learn something new, she aims to create more inclusive and effective STEM education.

## Andrew Paster

INSTRUCTOR

Andrew has an engineering degree from Yale, and has used his data science skills to build a jewelry business from the ground up. He has additionally created courses for Udacity's Self-Driving Car Engineer Nanodegree program.

# Learn with the Best

## Anthony Navarro

PRODUCT LEAD

Anthony is a US Army combat veteran with an M.S. in Computer Engineering from Colorado State University. Prior to being a Product Lead at Udacity, he was a Senior Software Engineer at Lockheed Martin in their Autonomous Systems R&D division.

## Elecia White

ENGINEER, AUTHOR, HOST

Elecia is an embedded software engineer at Logical Elegance, Inc, the author of O'Reilly's Making Embedded Systems, and host of the Embedded. fm podcast. She enjoys sharing her enthusiasm for engineering and devices.

## Tarin Ziyaee

DIRECTOR OF AI

As the Director of Artificial Intelligence at Voyage Auto, Tarin works to deliver low-cost, self-driving taxis. He brings a total of 14 years experience in perception and deep neural networks working with companies such as Apple.

# All Our Nanodegree Programs Include:

## EXPERIENCED PROJECT REVIEWERS

REVIEWER SERVICES

- Personalized feedback & line by line code reviews
- 1600+ Reviewers with a 4.85/5 average rating
- 3 hour average project review turnaround time
- Unlimited submissions and feedback loops
- Practical tips and industry best practices
- Additional suggested resources to improve

## TECHNICAL MENTOR SUPPORT

MENTORSHIP SERVICES

- Questions answered quickly by our team of technical mentors
- 1000+ Mentors with a 4.7/5 average rating
- Support for all your technical questions

## PERSONAL CAREER SERVICES

CAREER COACHING

- Personal assistance in your job search
- Monthly 1-on-1 calls
- Personalized feedback and career guidance
- Interview preparation
- Resume services
- Github portfolio review
- LinkedIn profile optimization

# Frequently Asked Questions

## PROGRAM OVERVIEW

### WHY SHOULD I ENROLL?

Self-driving cars are the future of smart transportation, and this introductory program is the perfect way to start your journey to a self-driving car career! This program enables anyone with minimal programming experience to learn the essentials of programming a self-driving car, from machine learning to object-oriented programming to probabilistic robotics. You will learn how to solve problems in both Python and C++ as you discover what makes self-driving cars possible.

### WHAT JOBS WILL THIS PROGRAM PREPARE ME FOR?

This program does not focus on job preparation. While many of the skills you learn are foundational skills for a career in self-driving cars, the goal of the program is for you to master the core skills necessary so you can progress to the advanced curriculum. When you graduate, you can enroll in the Self-Driving Car Engineer Nanodegree program, which will fully prepare you for a career in the field. If you are new to the field of self-driving cars, you can consider this a two-stage path to career readiness.

### HOW DO I KNOW IF THIS PROGRAM IS RIGHT FOR ME?

If you're interested in self-driving cars, and already have a basic understanding of programming and algebra, then this program is right for you! If you're considering a career in this amazing field, but don't yet possess all the skills and experience necessary to start applying for jobs today, this program will be your first step toward a self-driving car career. When you graduate, you will be ready for the Self-Driving Car Engineer Nanodegree program, where you'll learn everything you need to know to launch your career.

### WHAT IS THE DIFFERENCE BETWEEN THE INTRO TO SELF-DRIVING CARS NANODEGREE PROGRAM AND THE SELF-DRIVING CAR ENGINEER NANODEGREE PROGRAM?

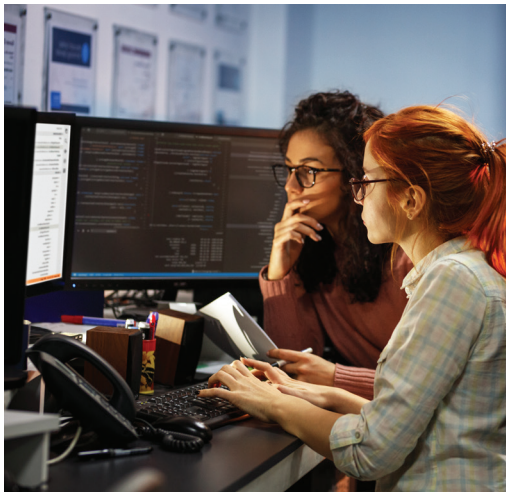
The **Intro to Self-Driving Cars Nanodegree** program is an intermediate program open to anyone with an interest in autonomous systems and some programming and/or quant background. The **Self-Driving Car Engineer Nanodegree** program is an advanced program focusing on in-depth knowledge of autonomous systems. The program is designed for those with moderate to high programming, technical, and/or quant skills.

## ENROLLMENT AND ADMISSION

### DO I NEED TO APPLY? WHAT ARE THE ADMISSION CRITERIA?

There is no application. This Nanodegree program accepts everyone, regardless of experience and specific background.

# FAQs Continued



### WHAT ARE THE PREREQUISITES FOR ENROLLMENT?

Students should have some experience with programming—writing short scripts in a programming language—and be comfortable with algebra. You should also feel comfortable reading and modifying code that you are given, with the understanding that solving problems in code may still be challenging.

### IF I DO NOT MEET THE REQUIREMENTS TO ENROLL, WHAT SHOULD I DO?

We have a number of Nanodegree programs and free courses that can help you prepare, including: **Intro to Computer Science**, **Programming Foundations with Python**, **Linear Algebra Refresher**, **Intro to Programming Nanodegree** Program.

## TUITION AND TERM OF PROGRAM

### HOW IS THIS NANODEGREE PROGRAM STRUCTURED?

The Intro to Self-Driving Cars Nanodegree program is comprised of content and curriculum to support eight (8) projects. We estimate that students can complete the program in four (4) months, working 10 hours per week.

Each project will be reviewed by the Udacity reviewer network. Feedback will be provided and if you do not pass the project, you will be asked to resubmit the project until it passes.

### HOW LONG IS THIS NANODEGREE PROGRAM?

Access to this Nanodegree program runs for the length of time specified in the payment card above. If you do not graduate within that time period, you will continue learning with month to month payments. See the **Terms of Use** and **FAQs** for other policies regarding the terms of access to our Nanodegree programs.

### CAN I SWITCH MY START DATE? CAN I GET A REFUND?

Please see the Udacity Nanodegree program **FAQs** for policies on enrollment in our programs.

### I HAVE GRADUATED FROM THE INTRO TO SELF-DRIVING CARS NANODEGREE PROGRAM BUT I WANT TO KEEP LEARNING. WHERE SHOULD I GO FROM HERE?

Students who graduate from the Intro to **Self-Driving Cars nanodegree** program should consider the **Self-Driving Car Engineer**, **Robotics Software Engineer**, or **Flying Car nanodegree** programs.

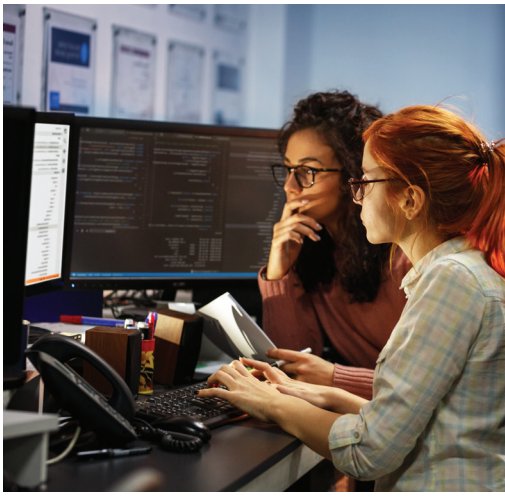
## SOFTWARE AND HARDWARE

### WHAT SOFTWARE AND VERSIONS WILL I NEED IN THIS PROGRAM?

For this Nanodegree program, you will need to have the minimum equipment requirements outlined here: **https://www.udacity.com/tech-requirements**.

# FAQs Continued

### WHICH VERSION OF TENSORFLOW, KERAS, ROS, AND C++ ARE TAUGHT IN THIS PROGRAM?



The following versions are taught in this program (subject to update):

- TensorFlow Version 1.3
- Keras version 2
- ROS Kinetic
- Python Version 3
- C++ Version 11