UDACITY

NANODEGREE PROGRAM SYLLABUS

Introduction to Programming

Overview

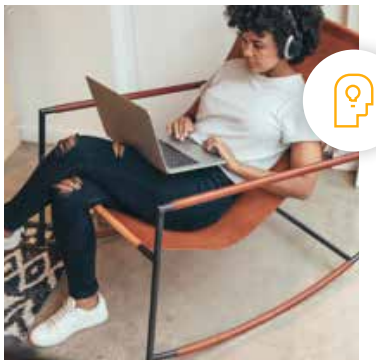
No prior experience with programming is required. You will need to be comfortable with basic computer skills, such as managing files, running programs, and using a web browser to navigate the Internet.

You will need to be self-driven and genuinely interested in the subject. No matter how well structured the program is, any attempt to learn programming will involve many hours of studying, practice, and experimentation. Success in this program requires meeting the deadlines set for your cohort and devoting at least 10 hours per week to your work. This requires some tenacity, and it is especially difficult to do if you don't find the subject interesting or aren't willing to play around and tinker with your code—so drive, curiosity, and an adventurous attitude are highly recommended!

You will need to be able to communicate fluently and professionally in written and spoken English.

Estimated Time: 4 Months at 10hrs/week

Prerequisites: No Coding Experience Needed



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: Intro to HTML

For this section, you will submit your very first programming file containing HTML code. HTML is the coding language for building websites. We recommend taking notes from this section and using your notes as the content for your HTML file. This project is a lab that is auto-graded in the classroom.

		LEARNING OUTCOMES
LESSON ONE	**Welcome to the Nanodegree**	• Understanding on how to set up for the program on your personal device. • Introduction to the "Programmer Mindset". • Successfully writing and rendering your first lines of HTML code with a text editor and browser
LESSON TWO	**Nanodegree Orientation**	• Understanding on how to submit projects. • Understanding on student support services offered for students. • Habits of Successful Students
LESSON THREE	**The World Wide Web**	• High level overview on how the web works. • Components of the web: browsers, HTTP requests, Servers, the Internet.
LESSON FOUR	**HTML Basics**	• HTML tags • Adding Images • HTML Syntax • Whitespace • Inline vs Block elements • HTML Document Structure

Course 2: Intro to CSS

In this section, you'll learn both HTML and CSS - both languages for developing websites. For the project, you'll use HTML and CSS to make Animal Trading Cards. You will apply your knowledge of HTML Document Structure to your html file and then create custom CSS styling based on your preferences. This project will demonstrate your understanding of linking CSS files in HTML files, implementing CSS classes to avoid repetition, as well create semantically organized HTML code.

Course Project :
Animal Trading Cards

In this project, you'll be creating a trading card for your favorite animal. You will use your knowledge of HTML to create the structure for your trading card. Then you will use CSS styling to design it.

	LEARNING OUTCOMES	
LESSON ONE	**Adding CSS to HTML**	• Understanding CSS basics • Divs, Spans, and Classes • Semantic Tags • Using DevTools in the Browser • Verifying HTML and CSS files • Debugging HTML and CSS code • Page Structure • Visual Styling • Designing with Boxes
LESSON TWO	**Animal Trading Cards**	• Apply what you've learned to create your first code-reviewed project.

Course 3: Intro to Python

In this section, you will learn the Python programming language. You will finish by building your own interactive game using Python that can be shared with your friends.

Course Project :
Adventure Game

Create an interactive game in Python using modules, loops, conditionals, and functions. Building this program will affirm your understanding of Python syntax, give you practice with fundamental programming logic, and demonstrate your ability to solve problems with code.

	LEARNING OUTCOMES	
LESSON ONE	**Turtles & Python**	• Write your first lines of Python code using turtles - a visual module that displays colorful rendition of programming.
LESSON TWO	**Functions - Part 1**	• Learn how to use functions to take an input and transform it into some output. • Explore syntax error messages and troubleshoot basic Python code.
LESSON THREE	**Functions - Part 2**	• Understand the difference between print and return statements • Learn how to manage the flow of a computer program using Boolean values, if statements, and While loops
LESSON FOUR	**Shell Workshop**	• Understand and practice working with the Command Line Interface (CLI)
LESSON FIVE	**Python at Home**	• Installing Python and learning Command Line Interface (CLI) basics • Learn how to store values in Variables and work with text as Strings • Selecting substrings with String Indexing

LESSON SIX	**Strings & Lists**	• Learn how to store values in Variables and work with text as Strings • Selecting substrings with String Indexing • Use String methods: slicing, concatenation, find, and replace • Use Lists to store more complex data • Use for loops to programmatically access each item within a List
LESSON SEVEN	**Version Control with Git**	• Learn about the benefits of version control and install version control.
LESSON EIGHT	**Working with Files**	• Understand how files are created and stored in computer memory. • Learn how to list files in a directory, work with filenames, and move and organize files. • Learn how to read text from a text file, process that text using string operations, and write text to another text file. • Work through some common bugs in text processing.
LESSON NINE	**Objects & Classes**	• Understand the difference between Objects and Classes in programming. • Define a new Class, understand the “self”, and defining special methods like initializers. • Creating instance variables. • Learn about inheritance, super Classes, and Class variables.

Course 4: Intro to JavaScript

Learn the history of JavaScript and how it compares to Python programming. Understand how the DOM is formed, what Nodes and Elements are, and how to select items from the DOM. By the end, you'll write JavaScript code that allows the user to create a grid of squares representing their design, and apply colors to those squares to create a digital masterpiece

Course Project :
Pixel Art Maker

For this project, you'll build a single-page web app that allows users to draw pixel art on a customizable canvas!

	LEARNING OUTCOMES	
LESSON ONE	**What is JavaScript?**	• Understand the history of JavaScript and start writing your code immediately using the JavaScript console.
LESSON TWO	**Data Types & Variables**	• Learn to represent real-world data using JavaScript variables, and distinguish between the different data types in the language.
LESSON THREE	**Conditionals**	• Learn how to add logic to your JavaScript programs using conditional statements.
LESSON FOUR	**Loops**	• Harness the power of JavaScript loops to reduce code duplication and automate repetitive tasks.
LESSON FIVE	**Functions**	• Dive into the world of JavaScript functions. Learn to harness their power to streamline and organize your programs.
LESSON SIX	**Arrays**	• Learn how to use Arrays to store complex data in your JavaScript programs.
LESSON SEVEN	**Objects**	• Meet the next JavaScript data structure: the Object. Learn to use it to store complex data alongside Arrays.

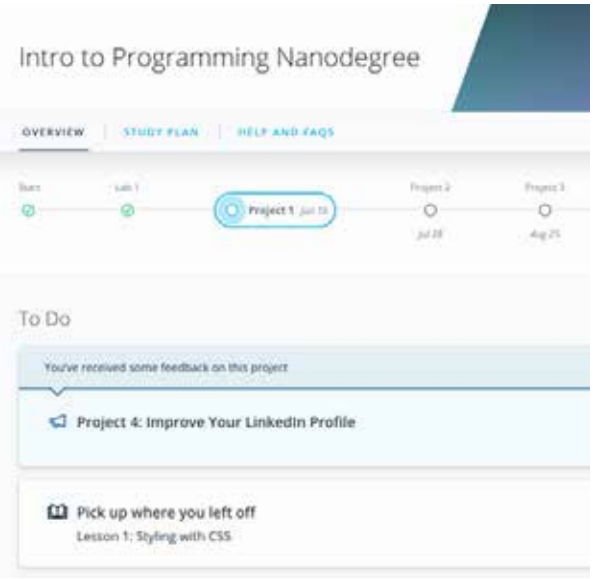
LESSON EIGHT	**ES6 Syntax**	• Learn how to update your code to comply with this major JavaScript update.
LESSON NINE	**The Document Object Model**	• Understand how the DOM is formed, what Nodes and Elements are, and how to select items from the DOM.
LESSON TEN	**Creating Content with Javascript**	• Use JavaScript and DOM methods to create new page content, update existing content, and delete content.
LESSON ELEVEN	**Working with Browser Events**	• Learn what an event is, how to listen for an event and respond to it, what data is included with an event, and the phases of an event.

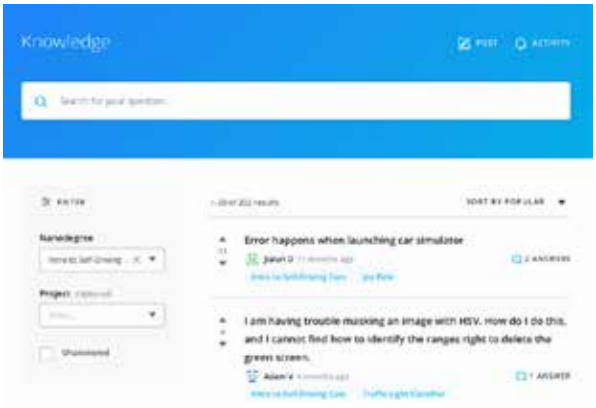
Extracurricular

In this section, there is no project submission. Instead, you will explore a quick overview of the vast world of programming. After this section, you'll have a better understanding of different options you have as a programmer.

	LEARNING OUTCOMES	
LESSON ONE	**Front-End Programming**	Learn about front-end web developers who create intuitive and responsive websites.
LESSON TWO	**Back-End Programming**	Learn about back-end web programmers who write server-side code to build web apps that serve millions of people worldwide.
LESSON THREE	**Mobile Programming**	Learn about mobile programming and the differences between iOS and Android programming.
LESSON FOUR	**Data Analysis Programming**	Learn about data analysts who analyze data to direct growth and make informed decisions.

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 fellow students in your program as you support and learn from each other.

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

Karl Krueger

INSTRUCTOR

Karl is a Course Developer at Udacity. Before joining Udacity, Karl was a Site Reliability Engineer (SRE) at Google for eight years, building automation and monitoring to keep the world's busiest web services online..

Stephen Welch

INSTRUCTOR

Kelly is the Product Lead for the Web Development Nanodegree programs at Udacity.

Julia Van Cleve

INSTRUCTOR

Julia is a Content Developer at Udacity and was previously a middle school math teacher in San Jose, CA. She also dabbled in freelance web development, designing websites for small businesses in the Bay Area.

James Parkes

INSTRUCTOR

James received his degree in Computer Science and Mathematics, then went on to become a Udacity instructor in several programs. His personal mission is clear: to open the doors of opportunity for others by empowering them with excellent educational experiences.

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 SUPPORT

- Resume support
- Github portfolio review
- LinkedIn profile optimization

Frequently Asked Questions

PROGRAM OVERVIEW

WHY SHOULD I ENROLL?

Knowing how to code can give you an edge in a growing variety of fields. Whether you're interested in becoming an artificial intelligence engineer or a web developer—or simply want to use programming to enhance your current career—you'll need a strong foundation, and in this program, you'll build a strong foundation in fundamental programming concepts. You won't need any prior experience with coding to enroll, and we've extensively tested the lessons with beginning students to make sure they're understandable, engaging, and effective.

WHAT JOBS WILL THIS PROGRAM PREPARE ME FOR?

While this is an introductory course that is not designed to prepare you for a specific job, after completing this program, you will be familiar with the fundamental skills used in web development, including HTML, CSS, JavaScript, and Python.

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

If you want to learn to code but have little or no experience, this program offers the perfect starting point.

ENROLLMENT AND ADMISSION

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

No. This Nanodegree program accepts all applicants regardless of experience and specific background.

WHAT ARE THE PREREQUISITES FOR ENROLLMENT?

No prior experience with programming is required.
You will need to be comfortable with basic computer skills, such as managing files, running programs, and using a web browser to navigate the Internet.

You will need to be self-driven and genuinely interested in the subject. No matter how well structured the program is, any attempt to learn programming will involve many hours of studying, practice, and experimentation. Success in this program requires meeting the deadlines set for your termand devoting at least 10 hours per week to your work.

You will need to be able to communicate fluently and professionally in written and spoken English.

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

The only technical skills required for this program are basic computer skills.

FAQs Continued

TUITION AND TERM OF PROGRAM

HOW IS THIS NANODEGREE PROGRAM STRUCTURED?

The Intro to Programming Nanodegree program is comprised of content and curriculum to support three (3) 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 and platform. 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.

I HAVE GRADUATED FROM THE INTRO TO PROGRAMMING NANODEGREE PROGRAM BUT I WANT TO KEEP LEARNING. WHERE SHOULD I GO FROM HERE?

The Intro to Programming Nanodegree program gives you a solid foundation from which to start a wide variety of more advanced and more specialized programs.
Here are some recommendations for what you might want to try next.

- **Front-End Web Developer Nanodegree program**
- **Full Stack Web Developer Nanodegree program**
- **Data Foundations Nanodegree program**
- **Data Analyst Nanodegree program**
- **AI Programming with Python Nanodegree program**
- **Android Basics Nanodegree program**
- **iOS Developer Nanodegree Program**

Please note that for some of these programs, you may need additional prerequisites that are not covered in the Intro to Programming Nanodegree program. You can find detailed info on these prerequisites on the pages linked above.

SOFTWARE AND HARDWARE

WHAT SOFTWARE AND VERSIONS WILL I NEED IN THIS PROGRAM?

Software/version requirements:

- Python 3

FAQs Continued

- A code/text editor, such as vim, Sublime Text, Atom, or VSCode
- A web browser
- A command line interface, such as Terminal (on Mac) or Git Bash (on Windows)

Software/version requirements:

- A modern personal computer running macOS, Windows, or Linux, with a high-speed Internet connection.

WHAT SOFTWARE AND VERSIONS WILL I NEED IN THIS PROGRAM?
The Intro to Programming Nanodegree program teaches Python 3.