UDACITY

NANODEGREE PROGRAM SYLLABUS

Introduction to Programming

Overview

Learn the basics of programming through HTML, CSS, Python, and JavaScript. Get extensive practice with hands-on exercises and projects that demonstrate your grasp of coding fundamentals, and build confidence in your ability to think and problem-solve like a programmer.

Prerequisites

- You are self-driven and motivated to learn.
- 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 can communicate fluently and professionally in written and spoken English.

Educational Objectives

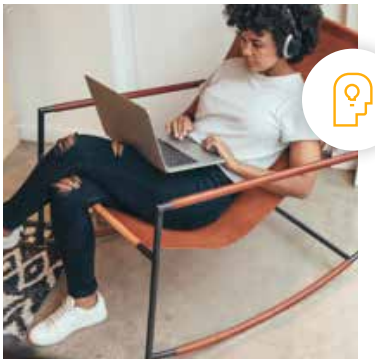
A graduate of this program will be able to:

- Create basic web pages using HyperText Markup Language (HTML).
- Modify web page style with Cascading Style Sheets (CSS).
- Write Python scripts that use core programming concepts, including variables, functions, loops, classes, objects, data types, conditionals, and debugging.
- Run Unix shell commands and Python code from a Command-Line Interface (CLI).
- Access and manipulate files on your computer using Python code.
- Use Python to get and process data from a web-based Application Programming Interface (API).
- Write basic JavaScript scripts that demonstrate core elements of the language, including data types, variables, loops, functions, arrays, and objects.
- Use JavaScript and the Document Object Model (DOM) to create and modify web page content.

Estimated Time: 4 months at 10hrs/week

Prerequisites:

- Access to a computer with a broadband connection, on which you'll install a professional code/text editor (e.g. Sublime Text or Atom).
- Able to solve and describe your solution to a math or programming problem.



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.

*The length of this program is an estimation of total hours the average student may take to complete all required coursework, including lecture and project time. If you spend about 10 hours per week working through the program, you should finish within the time provided. Actual hours may vary.

Course 1: Intro to Web Development

In this course you'll learn how to make basic web pages using HyperText Markup Language (HTML) and how to add style to your pages with Cascading Style Sheets (CSS). You'll begin by learning some basics about how the Web works, then build a very basic web page using only HTML, and finally explore how to add styles to your page with CSS. At the end of the course, you'll demonstrate your new skills by completing a project in which you create a web page that replicates a given design.

Course Project :
Animal Trading Cards

For this 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.

	LEARNING OUTCOMES	
LESSON ONE	**The Web and HTML**	• Describe the fundamentals of how the web works • Edit web pages using a text editor and test work in the browser • Create HTML files that use elements and tags to provide the structure of a web page • Write fully qualified URL pathways by identifying each part of file path structures
LESSON TWO	**Lab: Basic HTML Page**	• Demonstrate your understanding of HTML basics by creating a simple web page
LESSON THREE	**Styling with CSS**	• Use CSS to change basic style properties, like the font, color, and border of a given element • Use CSS type and class selectors to apply style to specific subsets of HTML elements • Separate the style of a web page from its structure and semantics • Apply style in multiple ways, including via a separate, linked stylesheet • Recognize tree structures in HTML and CSS code • Modify the layout and resizing behavior of a web page using containers and the flexible box model (flexbox) • Use Developer Tools to inspect the elements of a web page

Course 2: Intro to Programming with Python I

Learn basic programming with Python, one of the most versatile and widely used programming languages! You'll first learn core programming concepts and fundamental Python syntax by writing code to make a virtual "turtle" robot draw colorful shapes on the screen. You'll then learn how to write Python functions, run Python from a Command-Line Interface (CLI), manipulate strings and lists, and refactor your code to improve its structure and make it more modular.

Course Project :
Adventure Game

Create a simple, interactive, text-based adventure game in Python, using modules, loops, conditionals, and functions. This project will demonstrate your ability to write correct Python syntax, practice with fundamental programming logic, refactor code using functions, and ultimately write a complete Python script that results in a working, playable game.

		LEARNING OUTCOMES
LESSON ONE	**Turtles and Code**	• Use methods from Python's turtle module to draw simple geometric shapes • Define variables using assignment statements, and then use these variables in place of hard-coded constants • Work with fundamental data types, including integers, strings, and lists • Write compound statements and use indentation to indicate when code belongs to a given block • Import and use modules from the Python standard library. • Use code comments to add basic documentation and activate/de-activate code • Adjust the sequential order of code to alter the flow of execution. • Iterate over a data structure using a "for" loop • Use loop variables to generate a result that changes each time a loop runs and use nested loops in and correctly predict the number of times a nested loop will run • Recognize and fix common types of errors (including logical errors, syntax errors, and "usage" errors).

LESSON TWO	**Functions**	• Generate a sequence of numbers using Python's range function and iterate over the sequence using a loop • Perform basic calculations on constants and variables with Python arithmetic operators • Define and call Python functions to create more modular, reusable code • Use parameters in function definitions to make functions more flexible and to avoid hard-coded values • Distinguish between global and local variable scope • Use a Python conditional "if" statement to alter execution flow depending on a given condition • Use the Python modulo operator to create repeating patterns. • Use "return" statements when defining and calling Python functions. • Use Python's "random" module to generate random numbers and select random items from a list • Use Python comparison operators to compare values
LESSON THREE	**Shell Workshop**	• Distinguish between a GUI vs CLI and why the latter is an important tool for a developer • Output text to the command line with the "echo" command • Use BASH shell variables to store and retrieve values • Navigate directories using the "cd" command • List the contents of a directory using the "ls" command. • Modify commands by adding options (such as the "-a" option in "ls -a") • Organize files and directories using "mkdir" and "mv" commands. • Download files with the "curl" command • View files with the "cat" and "less" commands • Remove files and directories with the "rm" and "rmdir" commands
LESSON FOUR	**Python at Home**	• Use a Command-Line Interface (CLI) to run Python scripts. • Run simple and compound statements in Python's interactive mode via the CLI • Use Python's input function to get and use input from the user via the CLI • Use Python's print function to output text to the CLI. • Use Python's print function to debug code • Distinguish between how a Python function's return value behaves when the function is run from a code file vs interactive mode. • Recognize and fix type errors in Python • Use a Python traceback to trace through the relevant execution flow and identify the source of an error • Write a Python function and import it using interactive mode

LESSON FIVE	**Strings and Lists**	• Distinguish between variables and literals and use complex strings in your code that contain punctuation and newline characters • Use Python's length (len) function to get the length of a string and Iterate (loop) over the individual characters in strings • Use string indexes to access the character at a given location and handle index errors on strings • Use slicing to access a substring within a larger string and concatenation to join multiple strings • Use formatted string literals (f-strings) to concatenate values into strings and format them • Convert between string data types and integer data types and write functions to perform basic string manipulation tasks • Use common string methods to retrieve and manipulate string data and use Boolean values with strings and perform operations and methods on lists • Distinguish the key differences between mutable data structures and immutable data structures and perform augmented assignment to update the value of a variable • Use while loops to iterate over strings and lists and Identify and create infinite loops • Interrupt or break out of loops when needed and find and manipulate substrings and use the join method to concatenate strings from a list
LESSON SIX	**Style and Structure**	• Use a code linter to check whether code meets the conventions specified in the Python style guide (PEP 8) • Write multi-line strings in Python using a variety of techniques (triple quotes, escape characters, and implicit line joining) • Write a basic interactive Python program to meet specifications • Refactor a basic Python program to make the code easier to understand, more modular, and more flexible • Handle invalid user input in a basic interactive Python program • Use Python functions (instead of loops) to repeat a behavior repeatedly until a condition is met • Distinguish between global and local variable scope

Course 3: Intro to Programming with Python II

Advance your skills as a beginning programmer with Python—one of the most versatile and widely used programming languages! In this course, you will build on your understanding of fundamental Python and learn some more advanced skills, including how to work with files on your computer's disk, how to retrieve data using a web API, and how to use Object-Oriented Programming (OOP) to create your own classes, objects, and methods.

Course Project :
Rock Paper Scissors

In this project, you'll apply your Python and object-oriented programming skills to build a program that plays the game of Rock Paper Scissors. You'll build classes that represent the game and its players. You'll write computer players that follow various different strategies, as well as a human player class that lets a human play the game against the computer.

		LEARNING OUTCOMES
LESSON ONE	**Working with Files**	• Distinguish between files and other forms of data stored in memory (such as variables) • List files in a directory and extract file names • Move and organize files and read text from a text file • Process text using string operations • Write text output to a file • Identify and fix common bugs in text processing
LESSON TWO	**Web APIs**	• Use the BASH shell and the Python requests module to send requests to a web API • Use Python try/except blocks to handle exceptions • Recognize JSON as a standardized format for structuring data • Use Python dictionaries to structure data in key-value pairs • Use a Python loop to iterate over a list • Use a Python loop to iterate over a dictionary • Store and access nested elements of data structures within Python lists and dictionaries • Use a Python loop to iterate over data structures containing nested elements (e.g., lists within lists or dictionaries within dictionaries) • Use Python to retrieve data from another application via a web API

LESSON THREE	**Objects and Classes**	• Identify the key characteristics of classes and objects in Python. • Use the “isinstance” and “type” functions to identify the type of a given object in Python • Define a Python class and use it to instantiate an object • Use the self parameter to access the current instance (i.e., object) of a given Python class • Use class-level variables to store values that are shared across all instances of a Python class • Use instance-level variables in Python to store values that apply only to a specific instance (object) of a class • Use Python initializers to set initial values for an object when it is instantiated • Differentiate between is-a and has-a relationships with Python classes and subclasses • Use the super method in Python to create subclasses

Course 4: Intro to JavaScript

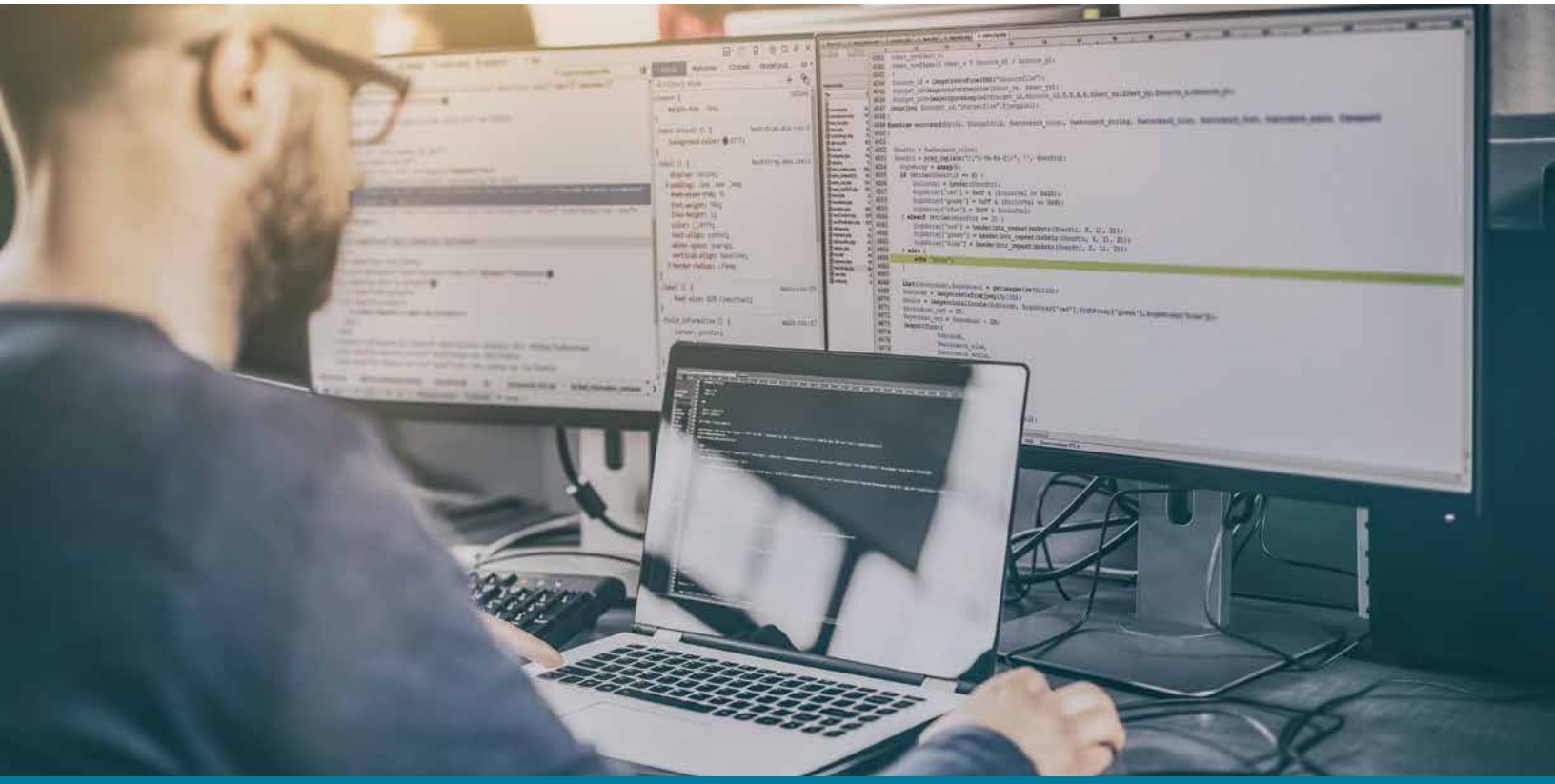
Learn the history of JavaScript and how it compares to Python programming. Understand how the DOM is formed, what nodes and elements are, and how to select items from the DOM. By the end, you'll write JavaScript code that allows the user to create a grid of squares representing their design, and apply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

Course Project :
Pixel Art Maker

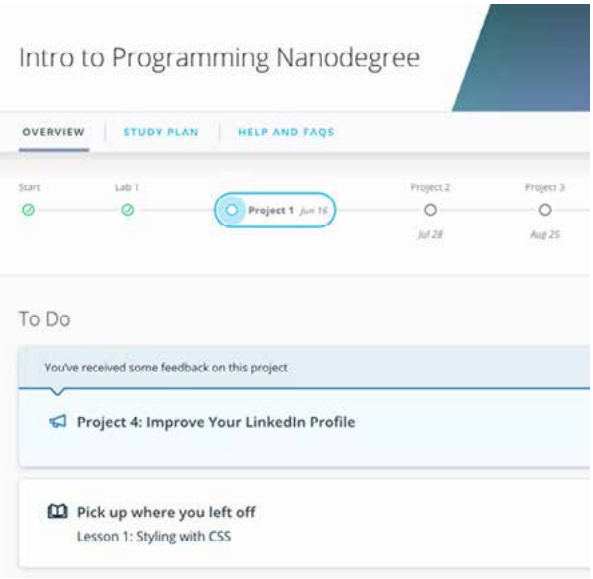
For this project, you'll build a single-page web app that allows users to draw pixel art on a customizable canvas!

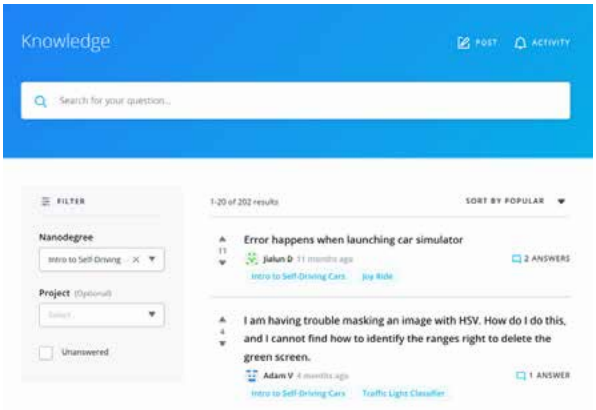
	LEARNING OUTCOMES	
LESSON ONE	**What is JavaScript?**	• Understand the history of JavaScript and start writing your code immediately using the JavaScript console
LESSON TWO	**Data Types & Variables**	• Learn to represent real-world data using JavaScript variables and distinguish between the different 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

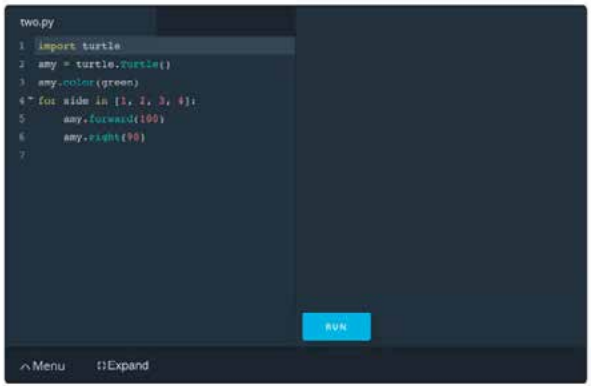
LESSON EIGHT	**The Document Object Model**	• Understand how the DOM is formed, what nodes and elements are, and how to select items from the DOM
LESSON NINE	**Creating Content with Javascript**	• Use JavaScript and DOM methods to create new page content, update existing content and delete content
LESSON TEN	**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



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.

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

Create a custom study plan to suit your personal needs and use this plan to keep track of your progress toward your goal.

PROGRESS TRACKER

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

Learn with the Best

Karl Krueger

INSTRUCTOR

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.

Kelly Howard

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.

Abe Feinberg

INSTRUCTOR

Abe is a science teacher and educational psychologist who loves learning and finding out how things work. He has a particular interest in using AI to optimize education and his ultimate goal is to replace himself with a robot that can teach better than he can.

Richard Kalehoff

INSTRUCTOR

Richard is a Course Developer with a passion for teaching. He has a degree in computer science. He first worked for a nonprofit doing everything from front end web development, to backend programming, to database and server management.

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

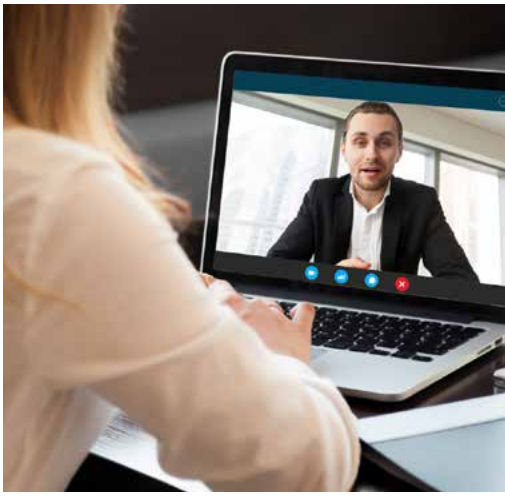
PERSONAL CAREER SERVICES

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?

General Requirements:

- You are self-driven and motivated to learn.
- 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 can communicate fluently and professionally in written and spoken English.

Program-Specific Requirements:

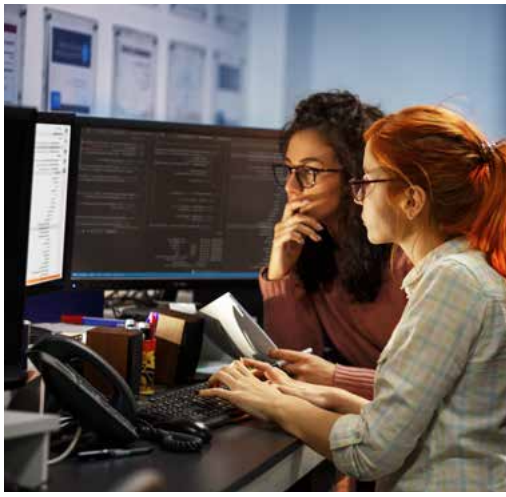
- You have access to a computer with a broadband connection, on which you'll install a professional code/text editor (e.g. Sublime Text or Atom).
- You can independently solve and describe your solution to a math or programming problem.

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 4 projects. We estimate that students can complete the program in 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.

FAQs Continued

SOFTWARE AND HARDWARE

WHAT SOFTWARE AND VERSIONS WILL I NEED IN THIS PROGRAM?

There are no software and version requirements to complete this Nanodegree program. All coursework and projects can be completed via Student Workspaces in the Udacity online classroom. Udacity's basic tech requirements can be found at https://www.udacity.com/tech/requirements.

WHICH VERSION OF PYTHON IS TAUGHT IN THIS PROGRAM?

The Intro to Programming Nanodegree program teaches Python 3.