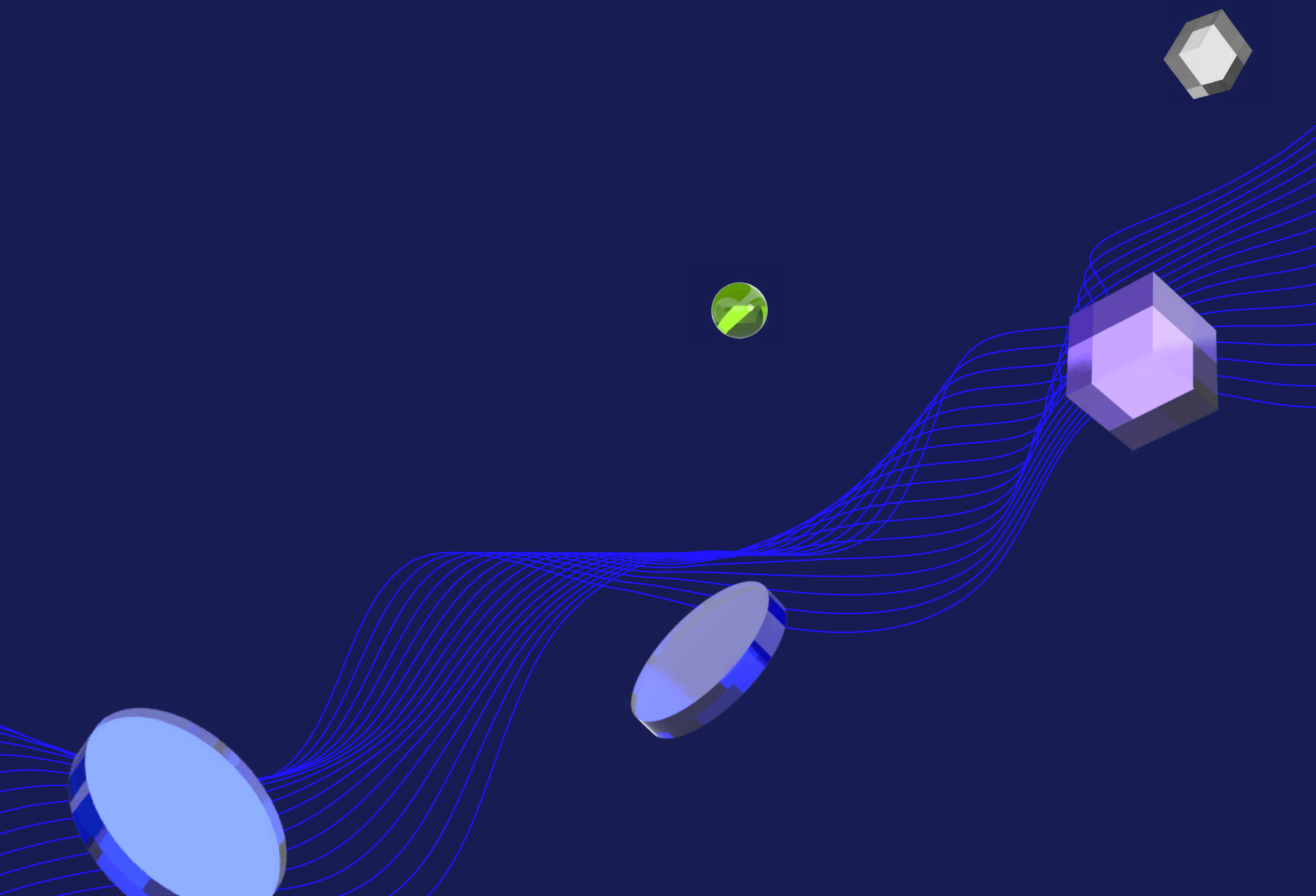
INDIVIDUAL LEARNERS

UDACITY

SCHOOL OF PROGRAMMING & DEVELOPMENT

Introduction to Programming

Nanodegree Program Syllabus



Overview

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

Learning 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 webbased Application Programming Interface (API).
- Write basic JavaScript scripts that demonstrate core elements of the language, including data types, variables, loops, functions, arrays, and objects.

Program information

Estimated Time

4 months at 10hrs/week*

Skill Level

Beginner

Prerequisites

Basic computer skills, such as managing files, running programs, and using a web browser to navigate the internet.

Required Hardware/Software

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.

*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 5-10 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 learners will make basic web pages using HyperText Markup Language (HTML) and add style to their pages with Cascading Style Sheets (CSS). They'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 the page with CSS. At the end of the course, learners will demonstrate their new skills by completing a project in which they create a web page that replicates a given design.

Course Project

Animal Trading Cards

For this project, learners will use HTML and CSS to make Animal Trading Cards. They will apply their knowledge of HTML Document Structure to their HTML file and then create custom CSS styling based on their preferences. This project will demonstrate understanding of linking CSS files in HTML files, implementing CSS classes to avoid repetition, as well create semantically organized HTML code.

Lesson 1

The Web & 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 2

Lab: Basic Html Page

- Demonstrate your understanding of HTML basics by creating a simple web page.

Lesson 3

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! First learn core programming concepts and fundamental Python syntax by writing code to make a virtual “turtle” robot draw colorful shapes on the screen. Then learn how to write Python functions, run Python from a Command-Line Interface (CLI), manipulate strings and lists, and refactor 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 the 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.

Lesson 1

Turtles & 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 2

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 3

Shell Workshop

- Distinguish between a GUI and 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 4

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.

- 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 5

Strings & Lists

Lesson 6

Style & 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

Beginners can advance their programming skills with Python—one of the most versatile and widely used programming languages. In this course, learners will build on their understanding of fundamental Python and learn some more advanced skills including how to work with files on a computer's disk, retrieve data using a web API, and use object-oriented programming (OOP) to create classes, objects, and methods.

Course Project

Rock Paper Scissors

In this project, learners will apply their Python and object-oriented programming skills to build a program that plays the game of Rock Paper Scissors. They'll build classes that represent the game and its players. They'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.

Lesson 1

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.

- 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 2

Web APIs

- 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.

Lesson 3

Objects & Classes

Course 4

Intro to Javascript

Learn the basics of JavaScript. Learners will create and use the primary data types in JavaScript and use conditionals and loops to control the flow of their JavaScript code. They'll write their first JavaScript functions and learn how to use arrays and objects to store collections of data. By the end of the course, learners will use their new JavaScript knowledge to build the core functionality of an eCommerce shopping cart.

Course Project

Build a Shopping Cart

In this project, learners will use their JavaScript development skills to build the core functionality of an eCommerce shopping cart for Kirana's fruit market. They will create a collection of products and then write the JavaScript code to add, remove, and change the quantity of a product in a shopping cart. They will also write the JavaScript code for the checkout functions: making a purchase and returning change.

Lesson 1

What is Javascript?

- Explain the history of JavaScript.
- Run JavaScript code in the browser console.
- Use console.log statements in JavaScript to log out values.

Lesson 2

Data Types & Variables

- Declare variables in JavaScript to represent data.
- Define and use number, string, and Boolean data types.
- Use operators to manipulate numbers.
- Use string properties and methods to manipulate strings.
- Use strong and loose equality in comparisons.

Lesson 3

Conditionals

- Add conditional logic to control program flow.
- Identify truthy and falsy values.
- Deploy ternary operators for more concise conditional logic.
- Use switch statements to chain multiple else if conditions.

Lesson 4

Loops

- Use for and while loops to reduce code duplication and automate repetitive tasks.
- Nest loops for more complex automation.
- Use assignment operators to write more concise code.

Lesson 5

Functions

- Create and run JavaScript functions to streamline and organize code.
- Use return statements to return items when a function runs.
- Explain how scope works in JavaScript.
- Use let and const in block scope.
- Store functions in variables using function expressions.
- Use inline function expressions.

Lesson 6

Arrays

- Create and use JavaScript arrays to store a collection of data.
- Use indexes to reference an array element.
- Deploy array properties and methods including length, push, pop, and splice to manipulate arrays.
- Loop through arrays using forEach and map.

Lesson 7

Objects

- Create and use JavaScript objects to store a collection of data as key:value pairs.
- Use dot notation and bracket notation to access object elements.
- Add and access methods in JavaScript objects.

Meet your instructors.

Karl Krueger

Course Developer at Udacity

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.

Kelly Howard

Product Lead at Udacity

Kelly is the product lead for the Web Development Nanodegree programs at Udacity.

Julia Van Cleve

Content Developer at Udacity

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.

Abe Feinberg

Educational Psychologist

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.

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.

Richard Kalehoff

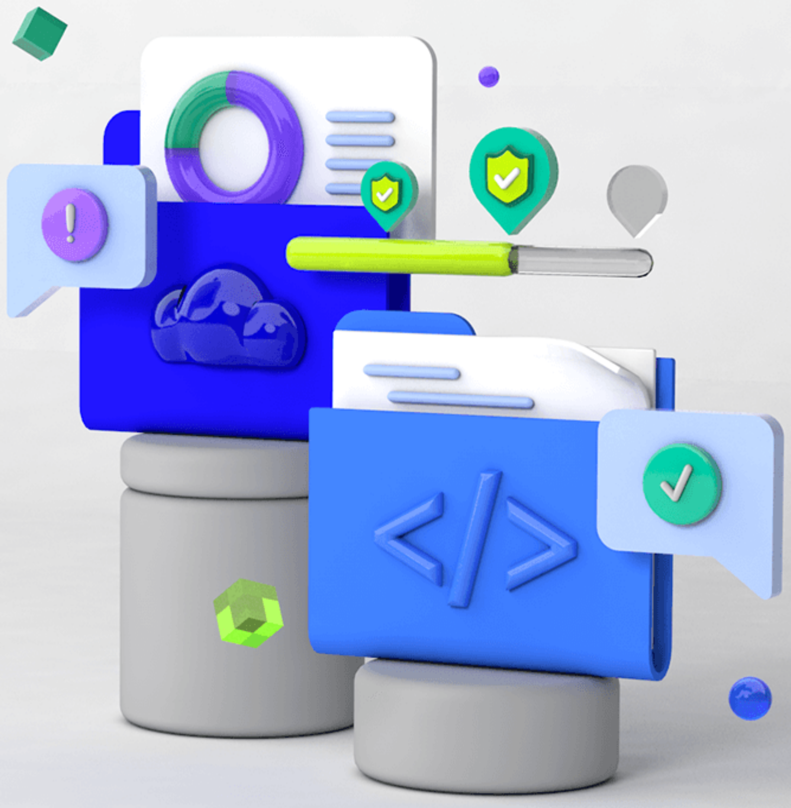
Course Developer at Udacity

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.

Rachel Manning

Full Stack Freelance Developer

Rachel is a full stack freelance developer and educator where she spent 3 years as a bootcamp curriculum developer while developing full stack freelance projects. An advocate for continued learning, she is passionate about mentoring women and underserved community in technology.



Udacity’s learning experience

Hands-on Projects

Open-ended, experiential projects are designed to reflect actual workplace challenges. They aren’t just multiple choice questions or step-by-step guides, but instead require critical thinking.

Quizzes

Auto-graded quizzes strengthen comprehension. Learners can return to lessons at any time during the course to refresh concepts.

Knowledge

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

Custom Study Plans

Create a personalized study plan that fits your individual needs. Utilize this plan to keep track of movement toward your overall goal.

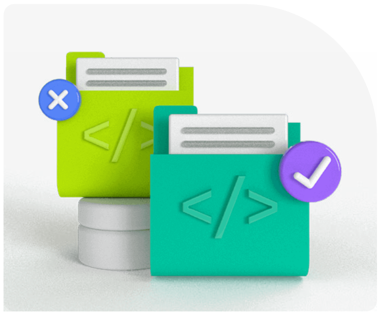
Workspaces

See your code in action. Check the output and quality of your code by running it on interactive workspaces that are integrated into the platform.

Progress Tracker

Take advantage of milestone reminders to stay on schedule and complete your program.

Our proven approach for building job-ready digital skills.



Experienced Project Reviewers

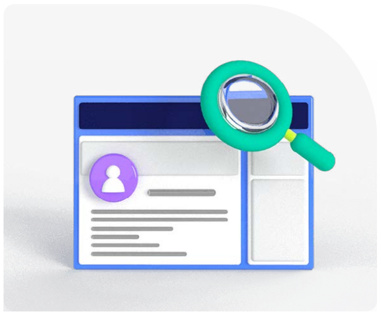
Verify skills mastery.

- Personalized project feedback and critique includes line-by-line code review from skilled practitioners with an average turnaround time of 1.1 hours.
- Project review cycle creates a feedback loop with multiple opportunities for improvement—until the concept is mastered.
- Project reviewers leverage industry best practices and provide pro tips.

Technical Mentor Support

24/7 support unblocks learning.

- Learning accelerates as skilled mentors identify areas of achievement and potential for growth.
- Unlimited access to mentors means help arrives when it's needed most.
- 2 hr or less average question response time assures that skills development stays on track.



Personal Career Services

Empower job-readiness.

- Access to a Github portfolio review that can give you an edge by highlighting your strengths, and demonstrating your value to employers.*
- Get help optimizing your LinkedIn and establishing your personal brand so your profile ranks higher in searches by recruiters and hiring managers.

Mentor Network

Highly vetted for effectiveness.

- Mentors must complete a 5-step hiring process to join Udacity's selective network.
- After passing an objective and situational assessment, mentors must demonstrate communication and behavioral fit for a mentorship role.
- Mentors work across more than 30 different industries and often complete a Nanodegree program themselves.

*Applies to select Nanodegree programs only.

UDACITY

Learn more at

www.udacity.com/online-learning-for-individuals →

01.06.22 | V1.0