LC 101

A Code Odyssey

Course Overview

LC 101 is a 20-week class designed to teach the skills necessary to begin a career in technology. Students should expect to spend at least 20 hours a week on coursework, including time spent in class. The class meets twice a week for 3 hour sessions. Outside of class you are expected to spend 10-15 hours doing readings, watching videos, working on practice exercises, and completing coding assignments. Assignments will generally be due once every week.

Evaluation

The course is Pass/Fail. You will receive a status of Pass or Fail for each Unit:

Pass: To Pass, students must complete all of the required assignments by the end of the Unit. Evaluation for passing each unit will be based on completion of weekly graded assignments, and a final project or problem set.

Fail: Students who do not complete the required assignments by the end of a Unit will receive a Fail for that Unit and will be not able to continue to the next.

For each Unit, students who Pass will receive a virtual Achievement badge. Students who Pass all Units will receive a LC 101 certificate of completion.

Course Objectives

This course is broken down into 3 Units:

Unit 1: Programming Fundamentals — the core of coding

Unit 2: Universal Web — the basics of web applications

Unit 3: Skill Tracks — tracks oriented toward job-ready skills

The goal of this class is to teach you the skills you need in order to launch a career in technology. The course has been built to provide you with a solid foundation in the most common areas of computer programming, and then allow you to dive deeper into one of those areas based on your interests.

Upon successful completion of the course, students will be able to:

- Use the foundational structures of code to solve problems in the Python programming language, as well as at least one skill track language (Java, Javascript, or C#).
- Build simple web applications that include a database to manage user data.
- Effectively use common developer tools and implement best practices to write professional-quality code.
- Create programming projects from scratch using in-demand skills and technologies.

Unit 1: Programming Fundamentals (6 Weeks)

Unit 1 covers core, universal programming concepts with a focus on problem-solving with the Python programming language. This will provide students with the conceptual building blocks that every programmer needs.

Topics and Skills

- Statements and variables
- Data types and typecasting
- Mathematical and comparison operators
- Data structures: lists and dictionaries
- Code style best practices
- Flow control: conditionals and loops
- Creating and using functions
- Scope
- Introduction to objects and classes

Unit 2: Universal Web (7 Weeks)

Unit 2 covers the foundational elements of a modern web application, with the notable exception of Javascript. Students will learn basic web languages and professional web developer tools, preparing them to eventually work on either front-end or back-end applications.

Topics and Skills

- HTML: tags, forms, and semantics
- CSS: selectors and style rules
- Information storage: using databases / SQL
- How web apps work
- Python as a back-end language
- Model-View-Controller design for web applications
- Command-line usage
- Version control using Github

Unit 3: Skill Tracks (7 Weeks)

Upon completion of Unit 2, each class will complete one skill track. These tracks are oriented toward job-ready skills, and will be chosen by LaunchCode based on market demand in each location. Here are descriptions of each possible skill track:

Object-Oriented Programming and Java Web Applications

This track teaches essential skills for back-end Java developers, that is, programmers who write Java code that runs on a web server. Students will learn in-demand Java technologies including Hibernate and Spring MVC to build Java web applications.

Topics and Skills

Core Java programming

- Procedural Java
- Data types
- Java collections
- Proper use of exceptions
- Static modifiers

Object-oriented skills and principles

- Defining classes and creating objects
- Class Inheritance
- Polymorphism and interfaces
- Access Modifiers

Multi-featured web applications in Spring MVC (using Spring Boot)

- Create and configure controllers
- Create and configure persistent model classes via Hibernate
- Create views using Thymeleaf templates

Object-Oriented Programming in C# and ASP.NET MVC Applications

This track teaches essentials for C# developers wanting to create modern web applications. Students will learn important object-oriented concepts C#, along with essentials for building web applications with the ASP.NET MVC platform

Core C# programming

- Procedural C#
- Data types
- C# collections
- Proper use of exceptions
- Static modifiers

Object-oriented skills and principles

- Defining classes and creating objects
- Class Inheritance
- Polymorphism and interfaces

- Access Modifiers

Multi-featured web applications in ASP.NET MVC

- Create and configure controllers
- Create and configure persistent model classes via ADO.NET
- Create views using Razor templates

Javascript and Front-End Web Programming

Front-end programming refers to the code running on a user's computer, with a focus on design and user interface (UI). This track will teach the the most powerful language for front-end development, Javascript, to allow students to create multi-featured front-end applications from scratch.

Topics and Skills

Core JavaScript programming

- Javascript syntax
- Higher order functions
- Javascript objects

Modern web tools

- The jQuery library
- AJAX for real-time web page updates
- Using APIs to integrate with third-party platforms like Twitter and Youtube
- Connecting to a data store

Improving user experience

- Using the Bootstrap framework for quick and easy styling
- Adapting websites for all devices with responsive design

Working in a web browser

- Browser compatibility issues and testing
- Debugging in the browser

Course Structure

In this course, learning will take place in two settings: at home online, and in person during class.

- **Outside of Class**: Before each class, students are expected to complete Prep Work, which consists of a combination of readings, videos, practice exercises. You will not be able to fully participate in class without having completed the Prep Work. For homework, in addition to the Prep Work for the next day of class, there are also Assignments to complete.

- **During Class**: During class time, we will go over the material you learned during Prep Work and work on Studio problems in small groups.

In Class

- **Review**: In class, we will review the material from the chapter you read, with live examples and question-and-answer sessions.
- **Mixin**: We will occasionally have special activities, called “Mixins”, which are meant to enrich the class and contribute to success in the class and after the class is over.
- **Studio**: In most class sessions, students work together in small groups on an in-class assignment called a “Studio”.
- **Review / Workshop**: Some classes will have review sessions that go over some concepts in more depth, or workshops that cover concepts that go beyond the scope of the course materials. Some of these will be targeted more towards those less comfortable with the material, and others towards those more comfortable.

Grading

Students will submit their assignments to an online grading platform. For many assignments, automatic grading scripts will run tests on student code and provide an automatic grade for correctness. Other assignments will require students to “demo” their projects to their Teaching Fellow, and the Teaching Fellow will manually input their grade.

All assignments are graded as complete or incomplete, with students receiving either a “0” or “1”.

Assignments will have due dates, but students are permitted up to two late submitted assignments. If a student needs to submit more than two late assignments, they may continue in the course, but must speak to the course instructor

Attendance

We expect students to attend every class session, and we will keep attendance. If students do have to miss a class, they must notify their Teaching Fellow. Continued absences will be grounds for disallowing a student to continue to subsequent Units.

Additional Course Details

Getting Help

Modern programming is done collaboratively. Throughout the course, there will be an emphasis on working in groups to tackle problems together. Students will be divided into Advisory Groups, each led by a Teaching Fellow (TF). TFs will be available to assist individual

students during class, and to help the group through common mistakes. Outside of class, students can use the class discussion board, where questions can be answered by peers and TFs.

Academic Honesty

Academic honesty is an important focus of this course. Working with others to improve your skills is both acceptable and encouraged, but there is a difference between asking for help and submitting someone else's work. Below are some examples to differentiate between the two:

- If you encounter a bug in your program, a classmate may look at your code to help you identify the source of your problem, but you may not look at the classmate's code to find their solution to that problem.
- You are encouraged to search through sites such as Stack Overflow when trying to debug an error in your code, but you may not search online for complete solutions to your specific assignment.
- Do not post complete programs or functions on the class discussion board. Post only snippets or isolated sections of code. If you need to post a complete program or function in order to get help, do so in a private message to the TFs or instructors.

If you have a question about how to work with another student on graded homework, ask the instructor or course staff.