# Welcome

## General Information

### About the course

If you are curious to learn more about the field of software engineering, this course is for you. No prior knowledge or qualification in programming or software development is necessary. Software Engineers are in great demand and now is a great time to learn more about this exciting career path.

The field of software engineering is growing at an estimated rate of 22% according to the US Bureau of Labor and Statistics. In addition to aspiring software engineers, this course can also help others in related careers such as Product and Project Managers, IT Managers, Information Developers and others who communicate and interact with software engineers.

This course teaches you about the power of the Software Development Lifecycle (SDLC) and software development methodologies like Agile. Explore fundamental programming principles and foundations of design, architecture, and deployment. Investigate skills a software engineer needs and identify job opportunities with hands-on projects. You will also learn about programming basics and software development tools and stacks.

Also, throughout this course, you’ll hear from expert and novice software engineers to get a feel for what the job is like from their perspective.  After completing this introductory course you will be conversant in the tools, technologies, terminology, processes, and practices in software engineering.

### Recommended Background

* Basic computer and IT literacy.

## Course Learning Objectives and Syllabus

### Course Learning Objectives

After completing this course, you should be able to:

1. Describe software engineering, approaches to software development, and software development tools and technologies.

2. Evaluate different types of programming languages and synthesize basic programming constructs.

3. Outline approaches to application and deployment architectures.

4. Summarize the skills required in software engineering and describe the career options it provides.

### Syllabus

**Module 1:** SDLC- The Software Development Lifecycle

* What is Software Engineering?
* Insiders’ Viewpoint: What is software engineering?
* Introduction to the SDLC
* Activity: Phases in the SDLC
* Building Quality Software
* Requirements
* Software Development Methodologies
* Software Versions
* Software Testing
* Software Documentation
* Roles in Software Engineering Projects
* Insiders’ Viewpoint: Job Roles in Software Engineering Teams

**Module 2:** Introduction to Software Development

* Overview of the Web and Cloud Development
* Learning Front-End Development
* The Importance of Back-End Development
* Teamwork and Squads
* Insiders’ Viewpoint: Teamwork in Software Engineering
* Pair Programming
* Insiders’ Viewpoint: Pair Programming
* Activity: Categorizing Front-end and Back-end Technologies
* Introducing Application Development Tools
* More Application Development Tools
* Introduction to Software Stacks
* Hands-on Lab: Getting Started With an IDE
* Insiders’ Viewpoint: Tools and Technologies

**Module 3:** Basics of Programming

* Interpreted and Compiled Programming Languages
* Comparing Compiled and Interpreted Programming Languages
* Query and Assembly Programming Languages
* Understanding Code Organization Methods
* Insiders’ Viewpoint: Types of languages
* Activity: Programming Logic and Organizational Methods
* Branching and Looping Programming Logic
* Introduction to Programming Concepts Part 1, Part 2
* Hands-on Lab: Programming Fundamentals with Python

**Module 4:** Software Architecture, Design, and Patterns

* Introduction to Software Architecture
* Software Design and Modeling
* Object-Oriented Analysis and Design
* Insiders’ Viewpoint: Importance of Design and Software Architecture
* Activity: Create an Architectural Diagram
* Approaches to Application Architecture
* Architectural Patterns in Software
* Application Deployment Environments
* Production Deployment Components
* Insiders’ Viewpoint: Deployment Architecture

**Module 5:** Job Opportunities and Skillsets in Software Engineering

* What Does a Software Engineer Do?
* A Day in the Life of a Software Engineer
* Skills Required for Software Engineering
* Insiders’ Viewpoint: Advice to Future Software Engineers
* Insiders’ Viewpoint: Women in Software Engineering
* Job Outlook for Software Engineers
* Career Paths in Software Engineering
* Software Engineering Job Titles
* Insiders’ Viewpoint: Career Paths
* Activity: Matching Roles with Responsibilities
* Software Engineering Code of Ethics

**Module 6:** Final Quiz

* Final Quiz
* Final project: Software Engineering Job Search and Analysis

## Course Introduction

## Meet the Experts

## Grading Scheme

### GRADING SCHEME

This section contains information for those earning a certificate. Those auditing the course can skip this section and click next.

1. The course contains**5** Graded Quizzes, **1** Graded Final Exam, and an **optional**Final Peer Graded Project for extra credit.
2. The minimum passing mark for the **course** is 70%.
3. Permitted attempts are per **question**:
   * One attempt - For True/False questions
   * Two attempts - For any question other than True/False
4. There are no penalties for incorrect attempts.
5. Clicking the "**Submit**" button when it appears, means your submission is **FINAL**.  You will **NOT** be able to resubmit your answer to that question again.
6. Check your grades in the course at any time by clicking on the "Progress" tab.

# [Module 1 - SDLC - The Software Development Lifecyle](https://learning.edx.org/course/course-v1:IBM+CS0131EN+3T2022/block-v1:IBM+CS0131EN+3T2022+type@sequential+block@2b372004b4134935a728ffcbda29ea44)

## Module Introduction and Learning Objectives

### Module Introduction

This module provides you with an overview to the field of software engineering. In the first lesson of this module, you will be introduced to the field of software engineering, and learn about the software development lifecycle (SDLC), elements of building high-quality software, and writing requirements. In lesson two you will explore different approaches to building software and different career opportunities related to software development. You will also hear from experienced practitioners in the field and learn how different roles interact and work with each other as a team in order to develop enterprise-level software.

### Learning Objectives

* Define software engineering and explain the responsibilities of a software engineer.
* Discuss key advantages of applying the software development lifecycle (SDLC) when developing enterprise software.
* Describe each phase of the SDLC
* Analyze common software engineering processes required for building-high quality software.
* Explain the requirement gathering process.
* Differentiate between a User Requirement Specification (URS), a Software Requirement Specification (SRS), and a System Requirement Specification (SysRS) document.
* Compare and contrast waterfall, V-shape model, and agile methods of software development.
* Distinguish between functional, non-functional, and regression testing.
* Identify and describe different types of documentation.
* Compare and contrast the common roles and responsibilities on a software engineering project.

## What is Software Engineering

## Insiders' Viewpoints: What is software engineering

## Introduction to the SDLC

## Phases of the SDLC

### Activity: Phases in the SDLC

## Building Quality Software

## Requirements

## Practice Quiz: Overview of Software Engineering

## Software Development Methodologies

## Software Versions

## Software Testing

## Software Documentation

## Roles in Software Engineering Projects

## Insiders' Viewpoints: Job Roles in Software Engineering Teams

## Practice Quiz: The Software Building Process and Associated Roles

## Summary & Highlights

Congratulations! You have completed this module. At this point, you know:

* Software engineering is the application of scientific principles to the design and creation of software.
* Responsibilities of a software engineer include designing, building, and maintaining software systems.
* Using the SDLC can improve efficiency and reduce risks by:
  + - letting team members know what they should be working on and when
    - facilitating communication between the customer, other stakeholders, and the development team
    - letting stakeholders know where they fit into that process and
    - letting cross-domain teams know when they have completed their tasks so development can move to the next phase.
* Common software engineering processes are requirements gathering, design, coding, testing, releasing, and documenting.
* The requirement gathering process entails identifying stakeholders, establishing goals and objectives, eliciting requirements from the stakeholders, documenting the requirements, analyzing, prioritizing, and confirming the requirements.
* An SRS is a document that captures the functionalities that the software should perform and also establishes benchmarks or service levels for its performance.
* A URS is a subset of the SRS that details user specification requirements.
* The SysRS contains the same information as an SRS, but can also additionally include system capabilities, interfaces, and user characteristics, policy requirements, regulation requirements, personnel requirements, performance requirements, security requirements, and system acceptance criteria.
* Waterfall, V-shape model, and agile are all different methodologies for implementing the software development life cycle.
* Functional testing is concerned with inputs and corresponding outputs of the system under test, non-functional testing tests for attributes such as performance, security, scalability, and availability. Whereas regression testing confirms that a recent change to the application, such as a bug fix, does not adversely affect already existing functionality.
* Types of documentation include requirements, design, technical, quality assurance, and user.
* There are many different roles involved in a software engineering project. Some of them include project manager or scrum master, stakeholder, system or software architect, UX designer, software developer, tester or QA engineer, site reliability or Ops engineer, product manager or owner, and technical writer or information developer.

# [Module 2: Introduction to Software Development](https://learning.edx.org/course/course-v1:IBM+CS0131EN+3T2022/block-v1:IBM+CS0131EN+3T2022+type@sequential+block@8456ad9a547c43a0904885ee249dca1b)

## Module Introduction and Learning Objectives

### Module Introduction

This module introduces you to the concepts of application development. In lesson 1 you’ll learn about the differences in front-end and back-end development in web and cloud applications and the technologies used in both. You’ll also learn about how working in teams and pairs can enhance the development process and make it more efficient. In lesson 2 you will explore many of the application development tools that a software engineer uses to write, test, and release code and be introduced to software stacks that support the execution of an application. Finally, in the hands-on lab, you’ll learn how to use an integrated development environment (IDE) to develop and run code.

### Learning Objectives

Describe how websites and cloud applications work.

Identify the technologies you can use to create reactive and responsive websites.

Identify the back-end technologies you can use to create functionality in your websites.

Describe how teams and squads work in a software engineering context.

Identify the benefits of different pair programming styles.

Classify common web technologies into front-end and back-end technologies.

Identify commonly used types of development tools, including version control software, libraries, and frameworks.

Describe how some key developer tools work, including CI/CD tools, build tools, packages, and package managers.

Describe software stacks and identify the advantages of three popular stacks

Identify parts of an IDE and use them to create a simple software application.

## Overview of Web and Cloud Development

## Learning Front-End Development

## The Importance of Back-End Development

## Teamwork and Squads

## Insiders' Viewpoints: Teamwork in Software Engineering

## Pair Programming

## Insiders' Viewpoints: Pair Programming

## Activity: Categorizing Front- and Back-End Technologies

## Practice Quiz: Introduction to Development

## Introducing Application Development Tools

## More Application Development Tools

## Introduction to Software Stacks

## Hands-on Lab: Getting Started With an IDE

## Insiders' Viewpoints: Tools and Technologies

## Practice Quiz: Tools in Software Development

## Summary & Highlights

### Congratulations! You have completed this module. At this point, you know:

* How websites are built and displayed, and how they communicate with the back-end servers.
* How different front-end technologies work together to create reactive and responsive websites.
* How back-end development covers a wide range of technologies including business logic, security, and database access.
* Effective teamwork can result in better quality code with fewer bugs, better-skilled team members, and less stress for everyone.
* Pair programming is a great way to share knowledge and skills between developers, resulting in better solutions and improved efficiency.
* You can use developer tools to track who makes what changes to your code withversion control software, access libraries of reusable code, and use frameworks to build and deploy applications in a standard way.
* CI/CD tools, build tools, packages, and package managers help you build and distribute your applications.
* A software stack is a combination of technologies for creating applications and solutions.

## Graded Quiz: Introduction to Software Development