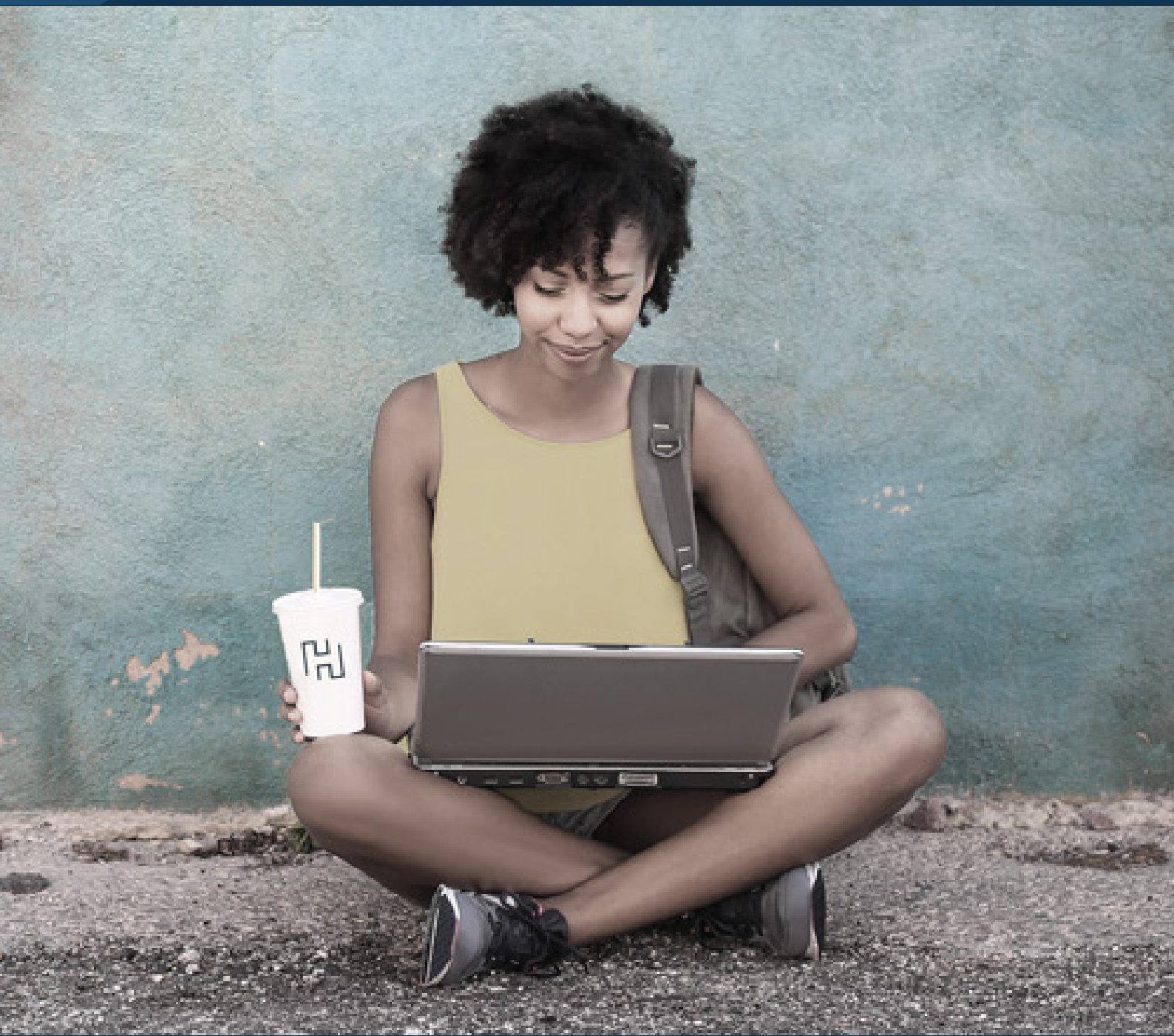
Hyperiondev

Certified

Software Engineer Bootcamp



python Software Foundation

facebook

Google

Cambridge Social Ventures

UNIVERSITY OF CAMBRIDGE

Overview

If the idea of analysing a situation and seeing how it can be improved using software excites you, then software engineering may be the career for you. Software engineering involves more than just coding. This discipline uses principles applicable to a breadth of large-scale software systems. Ultimately, you'll be able to construct software solutions to solve specific business problems.

The engineering process involves working with stakeholders to understand the requirements and limitations of a software system. The software engineer analyses these requirements, and then designs, implements, deploys and maintains the software system.

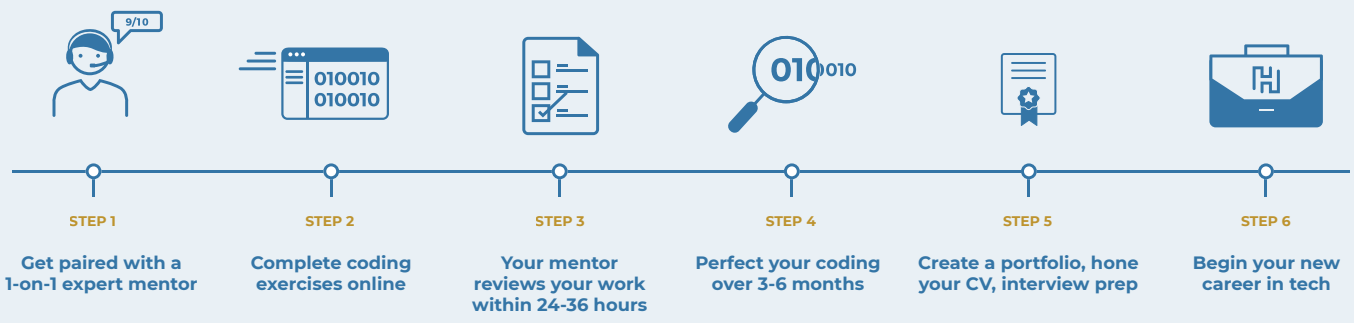
No prior knowledge of coding is required when taking this bootcamp, as we help you progress from beginner to advanced, becoming job-ready in only six months. Right from the start of the bootcamp, you're taught how to think like a programmer by developing systematic algorithms to solve various problems.

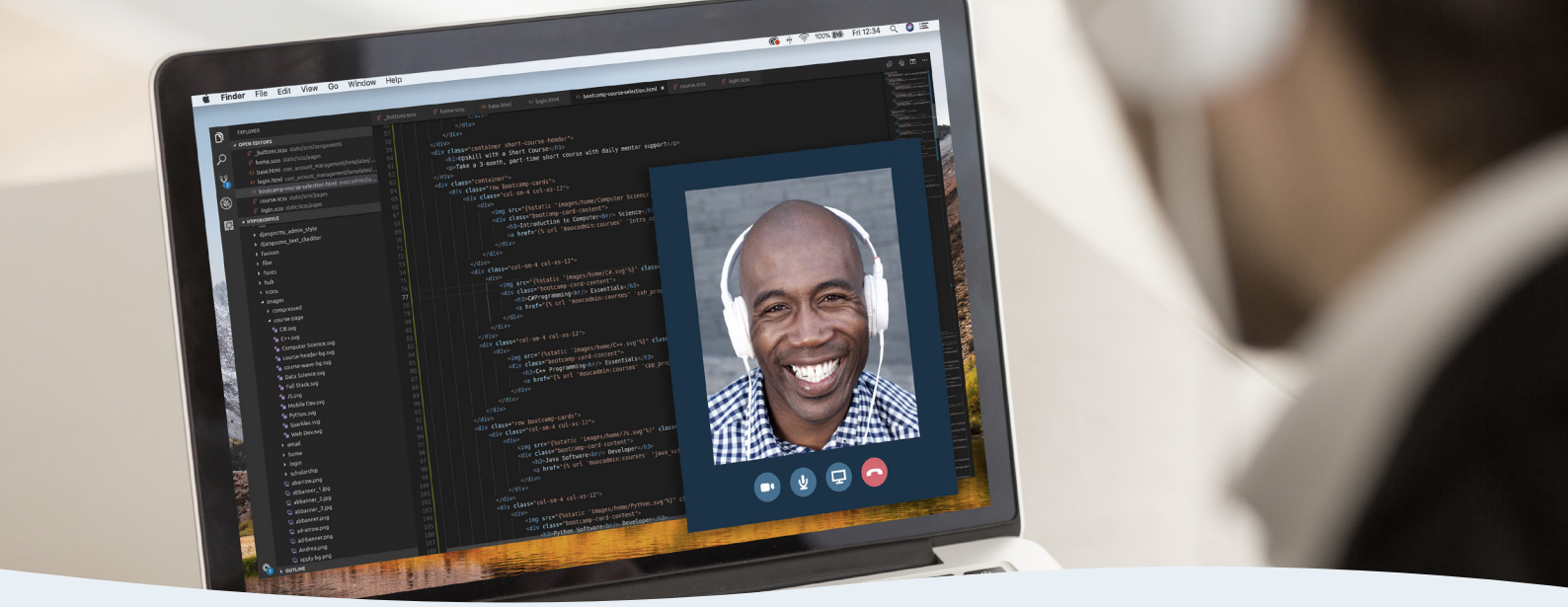
Going Beyond Software Development

You'll learn how to write code that can interact with databases, and that uses established design patterns and algorithms to create useful software that solves real-world problems. Advanced-level outcomes also include being able to test, debug, deploy and maintain software systems, as well as guaranteeing their quality.

Throughout the bootcamp, you will be guided to develop the skills required to think beyond mere software development and deployment. You'll also learn to manage a software development project using Agile development, while communicating with technical and non-technical stakeholders. Here is where you learn how software forms part of a system's architecture, and how to apply best practice principles during the software development lifecycle.

The Process



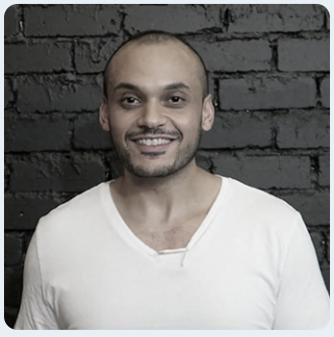


Outcomes of the Bootcamp:

- Design solutions to problems, and express them using pseudo-code and algorithms.
- Write useful code using Python and Java, two of the best programming languages to learn as they're used throughout the industry to create various web and mobile applications.
- Understand and apply computer science fundamentals, including data structures such as heaps, stacks and lists. Other fundamentals covered include algorithms for sorting and hashing, and using Big O Notation to analyse the performance of an algorithm.
- Use established algorithms to implement machine learning.
- Use agile development for software development projects.
- Design, plan, build, test, debug, refactor, deploy and maintain a software system.
- Use established design patterns and Git to ensure version control.
- Become job-ready with our career support team that guides and prepares you for the tech career you're aiming for.

Mentors powered by CoGrammar

Our mentors are expertly trained by CoGrammar, the only company that sources, trains and integrates code reviewers into the lives and bootcamp curriculum of our students. The on-demand code review method helps our students become fluent in the language of their choice.



Our 1-on-1 code review centric approach works

Code review enables you to learn to code the right way through mastery of deeper aspects of software development that are a prerequisite for a career in coding We help you master the deeper aspects of industry-level development and set the foundation for a lucrative career in coding.

Here's why learning through code review is smarter

Don't make the same mistakes as computers

- Automated code checking is like spell check for computer programs. But you can't write a world-class essay with just good spelling - you need the right tone, facts, grammar, and style. Only human review of your code can help you learn aspects of coding that are analogous to tone and style that will make you truly fluent as a developer - automated graders just can't help you learn this!

Get unstuck with on-demand technical help

- Most online programming courses have extremely high dropout rates because students get frustrated by generic automated error messages and give up. Through daily and rapid review, your mentor will help you debug your programs and move forward so never drop out.

Be exposed to the industry standards from day one

- Developers in the real world have their work assessed by a senior developer through the technique of code review. We're the only bootcamp in the world that exposes our students to this technique from day 1 so you get an unfair advantage in the job market.

We layer a proven 1-on-1 personalised mentorship approach over our code review.

Industry experts tailored to your goals

- You'll be paired with your own dedicated mentor who will tailor your learning experience to your desired career outcomes with 1:1 calls, career coaching, and live chat and email support.

Join a community of career-changers

- Learn as part of a cohort of students all working towards ultimate career fulfilment. Join group tutorials, community chats and meetups, and peer coaching.

Free of fear of failure

- 1-on-1 mentoring builds trust with your mentor and lets you progress at your own pace. Establish a safe space to discuss any roadblocks without fear of failure.

Why Choose Software Engineering as a Lucrative Career?

Software engineering is a creative career that allows you to work with code and people, as well as hardware and other computer systems. This career places you at the heart of the digital economy, with endless scope for growth. Software engineers hold valuable skills that enable them to earn good salaries. In fact, according to Glassdoor, the national average salary for a software engineer in the United States is $109,087. In the UK, the national average software engineer salary is £37,469 and according to PayScale. The South African market pays software engineers on average an estimated **R560,137** per year, according to Indeed.

If you're looking for a career that is both rewarding and lucrative, software engineering delivers on both. However, those who develop software engineering skills can also choose to pursue other career paths, some of which we'll delve into next.

How We Get You Hired

We're with you every step of your journey, and our support doesn't end when you graduate. Our career services are developed to help you stand out from the crowd, and grab the attention of top employers.

Technical CV and Portfolio

Receive technical assistance in getting your CV industry-ready according to accepted best-practice format.

Bootcamp Certificate

Walk away with a newly minted certificate as evidence of your skills and expertise in software engineering.

Interview Preparation

Know what to expect when getting ready for that big interview with expert interview preparation from professionals who have been where you are.

Join our hiring network

We work with select hiring partners to place our students in new jobs within six months of graduating. There are also internship placements available with select partners.

Career Paths

The Business Analyst

An analyst assumes responsibility for eliciting requirements from the client, and modelling them for those who will actually build the system.

Responsibilities include:

- Meeting the client and gathering system requirements.
- Clarifying specific requirements, as there may be ambiguities.
- Modelling the system requirements for engineers.
- Documenting progress made and reporting back to the client.
- Presenting the system to the client.
- Drafting acceptance tests to ensure the system is signed off by the client.

Average Salaries

A business analyst in the United States can earn an average salary of $70,170 per year.

The Software Architect

Large systems require lots of high-level planning to ensure code is written in a structured manner. This allows for changes to be implemented as required. Like traditional architects, software architects draw up the blueprint of the code that'll implement the functions of the system.

Responsibilities include:

- Creating a high-level design of the system.
- Determining the operating platform that the application runs on, and other middleware requirements of the system.
- Modelling the various software modules that the system is broken down into.Exploring and implementing emerging front-end web technologies.
- Coordinating with various teams to ensure the modules work when integrated into one whole.
- Communicating challenges, changes and progress to the business analyst.

Average Salaries

A software architect in the United States can earn an average salary of $128,715 per year.

The Developer

A developer, or often referred to as a programmer, is tasked with writing the code that implements the system. Without developers, software systems wouldn't exist.

Responsibilities include:

- Translating the architecture to a model in the solution domain for implementation.
- Thoroughly documenting the code written.
- Writing tests for the code produced, and running them on the code.
- Fixing any bugs that might crop up.

Average Salaries

A developer in the United States can earn an average salary of $79,985 per year.

Structure of the Bootcamp

This bootcamp helps you progress from learning the basics of programming to becoming a certified software engineer with a rewarding and satisfying job. Proceed from novice to advanced level, and land the successful career you deserve:

Before you start

Bootcamp Prep

Learn about the software development sector and how HyperionDev supports you in achieving your development goals. Start programming with Python to attain a clearer idea of whether a career in the software development industry is really for you.

Beginner level

Introduction to Programming

Get to grips with the fundamentals of programming and the Python programming language. You also learn the basic concepts and master fundamental skills needed to code in Python.

Intermediate level

Introduction to Software Engineering

Learn Java, arguably the most popular programming language. Understand how industry professionals develop software by exploring the best practices they use. You're also challenged to work on your programming skills, enabling you to deliver the most effective solutions for clients.

Advanced level

Data Science, Algorithms and Advanced Software Engineering

Take on the more advanced software engineering concepts, and explore aspects such as deployment and maintenance best practice, quality assurance, Big O Notation, machine learning and algorithms, among others.

Post graduation

Interview and Getting Hired

Post graduation, receive career support and guidance, including interview preparation, CV review, direct referral to our hiring partners, and potential internship placement at select hiring partners. Most of our students get hired within six months of graduating.

Breakdown of Syllabus

The bootcamp is structured to allow you to start coding as soon as possible. Tasks are designed to:

1. Teach you the theory needed to develop your skills.
2. Give you the platform to practice implementing your new knowledge by completing one or more practical activities.

Remember, with HyperionDev, you're never alone. You can contact your mentor for 1:1 support whenever you need help with a task. The code you submit for each task is reviewed by your mentor who is an industry expert, to help improve efficiency and quality of code, within 36 hours of submission.

Introduction to Programming

Tasks: 25
Capstone Projects: 4

	Tasks	Description
1	**Thinking like a Programmer - Pseudo Code I**	Learn how pseudo code can help you clarify your thoughts and properly plan your programs before you start to write any code.
2	**Thinking like a Programmer - Pseudo Code II**	Delve further into algorithm design and representation.
3	**Your first computer program**	Get acquainted with Python, the powerful, easy to learn and extremely popular, high-level programming language.
4	**Variables - Storing data in programs**	Learn how to store and interact with the data in our programs using variables
5	**The String Data Type**	Learn how to store and manipulate text using the String data type.
6	**Numerical Data Types**	Explore the different types of numbers used in the Python programming language.
7	**Beginner Control Structures - if Statements**	Learn how to use the if statement to make decisions in your program.
8	**True or False? The Boolean Data Type**	Gain a better understanding of conditional statements by learning about the Boolean data type.
9	**Beginner Control Structures - else Statements**	Learn how to control the order in which statements are executed using the else statement.
10	**Beginner Control Structures - elif Statements**	Learn how to check for multiple conditions using elif statements.
11	**Logical Programming - Operators**	Learn how to tell the compiler how to perform specific mathematical, relational or logical operations using operators.

12	**Capstone Project - Variables and Control Structures**	Put your knowledge of variables and control structures to the test by creating an investment calculator.
13	**Beginner Control Structures - While Loop**	Learn how to execute a block of code repeatedly until a given condition returns false using while loops.
14	**Beginner Control Structures - For Loop**	Learn how to use the for loop to repeat a section of code a specified number of times.
15	**Capstone Project - Nested Loops**	Learn how to nest various loops within one another.
16	**Defensive Programming - Error Handling**	Discover the different types of errors in that might occur in your programs and how to handle them.
17	**String Handling**	Learn how to manipulate text using Python's built-in functions.
18	**Working with External Data Sources - Input**	Create smarter programs by learning how to read data from text files.
19	**Working with External Data Sources - Output**	Learn how to write data to text files.
20	**Capstone Project - Files**	Put everything you've learnt about files to the test in this comprehensive task.
21	**Beginner Data Structures - The List**	Discover the most frequently used and versatile collection data type used in Python - the list
22	**Beginner Data Structures - Lists and Dictionaries**	Learn how to manipulate lists and become acquainted with dictionaries.
23	**Beginner Programming with Functions - Using built-in functions**	Learn how to use Python's built-in functions to provide better modularity for your programs and encourage code reuse.
24	**Beginner Programming with Functions - Defining your own functions**	Create your own Python functions to carry out specific tasks.
25	**Capstone Project - Lists, Functions and String Handling**	Use all the knowledge you have gained throughout this course to create a useful program.

Introduction to Software Engineering

Tasks: 20
Capstone Projects: 1

	Tasks	Description
1	Thinking Like a Software Engineer I - Introduction to Software Engineering	Get acquainted with software engineering and discover the attributes of a good software engineer.
2	Introduction to Java Programming I - Java Basics	Discover the fundamental concepts of Java, such as variables, data types and control structures.
3	Introduction to Java Programming II - Data Structures (Arrays)	Learn how one-dimensional and multidimensional arrays are declared, created, initialized and processed.
4	Introduction to Java Programming III - Methods	Learn how to create and efficiently use Java methods.
5	Thinking Like a Software Engineer II - Diving into a Large Code base and Writing Maintainable Code	Discover how to tackle large codebases and write maintainable code.
6	Object Oriented Programming	Learn the fundamentals of Object-Oriented Programming.
7	Introduction to Version Control and Git	Explore the Git version control system and the GitHub collaboration platform.
8	Git Basics	Dive into using Git and discover how to set up a repository, use common Git commands, commit a modified file, view your project's history and branch.
9	The Software Process	Delve into the concepts of the software development process and the software process models.
10	Agile Development	Learn about agile development and one of the most popular agile methodologies - Extreme Programming.
11	Advanced OOP	Learn about inheritance and how it can be utilised to make your code neater and more logical.
12	Java Text I/O	Learn how to read from and write to external storage mediums.
13	Recursion	Explore the concepts of recursive programming and how to “think recursively”.
14	Object Oriented Design and Design Patterns	Examine design patterns in the realm of software development.
15	Defensive Programming	Learn how to guard against errors you don’t expect.
16	Debugging	Discover how to detect and remove existing and potential errors in your programs using Eclipse.

17	**Software Testing**	Familiarise yourself with the three stages of testing: development testing, release testing and user testing.
18	**Refactoring**	Explore the concept of refactoring and the variety of tools provided by Eclipse which allow you to refactor your code quickly and easily.
19	**Software Documentation**	Learn about the various forms of software documentation and how they can improve the quality of your software.
20	**Java Collections Framework**	Discover the Java Collections Framework, a collection of interfaces and classes that helps to store and process data efficiently.

Data Science, Algorithms and Advanced Software Engineering

Tasks: 22
Capstone Projects: 2

	Tasks	Description
1	**System Requirements and Design**	Explore best practice guidelines for defining your product and UI/UX design guidelines and tools.
2	**System Architecture**	Discover the various components that make up and interact with a software system.
3	**Introduction to Databases**	Compare relational, graph and NoSQL databases.
4	**Design and build relational database**	Design and build a relational database by applying normalisation principles.
5	**Introduction to Java Database Programming**	Learn how to communicate with your database using SQL and MySQL.
6	**Java Database Programming: The JDBC**	Explore the JDBC, the Java API for accessing relational databases.
7	**Capstone Project I**	Design a system that interacts with a database.
8	**Introduction to Computer Science Fundamentals and Big O Notation**	Learn what computer science is and discover how Big O notation is used to describe the performance of algorithms.
9	**Algorithms: Sorting and Hashing**	Learn how to implement popular computer science algorithms for sorting and hashing.
10	**Algorithms: Collections Framework**	Explore the Collections API algorithms.
11	**Unit Testing in Eclipse Using JUnit**	Learn how to use the popular JUnit testing framework to write and run tests.

12	**Quality Assurance**	Discover how to ensure that your developed software is both dependable and secure.
13	**Deployment and Maintenance Best Practice**	Discover the best practice guidelines for ensuring effective deployment and maintenance of software systems.
14	**Machine Learning: Introduction to Machine Learning**	Get acquainted with supervised and unsupervised machine learning.
15	**Machine Learning: Introduction to Scikit-Learn**	Learn how to setup and use the extremely useful scikit-learn library.
16	**Machine Learning Algorithms: Regression Analysis**	Delve further into machine learning by making use of the machine learning algorithms included in scikit-learn.
17	**Machine Learning Algorithms: Clustering**	Learn about finding structure in data by discussing the clustering machine learning algorithm.
18	**Capstone Project II**	Put your knowledge of machine learning to the test in this comprehensive task.
19	**Interview Preparation 1: Concurrency**	Discover what concurrency is and the various approaches to concurrency control.
20	**Interview Preparation 2: Interfaces**	Discover what interfaces are, the advantages of using interfaces and how interfaces are implemented.
21	**Career Guidance 1**	Prepare your CV to embark on a career as a software engineer.
22	**Career Guidance 2**	Improve your CV

Hyperiondev

Become A

Certified Software Engineer

Start Your Free Bootcamp Prep

Browse Courses

Follow us on social media