PROGRAM OVERVIEW
This four-part course prepares the non-computer science engineer, or the natural science educated individual who has no computing background, with an introduction to programming with Python, C++ programming language, discrete mathematics and data structures. This course is a prerequisite for Bootcamp II.

PART 1: INTRODUCTION TO PROGRAMMING WITH PYTHON (JUNE 21 – JULY 4, 2020)
Part 1: Introduction to Programming with Python is an accelerated introduction to computer programming through the Python programming language and it is intended to prepare those with little experience for further education in computer science. This course covers foundational computing principles using the Python programming language for demonstration. Topics covered include procedural programming and control flow, functions and common data structures and an introduction to file input/output and object-oriented programming. This course uses Python version 3.

Objectives
At the end of Part 1, the learner should be able to:

- Understand the basic concepts of computer programming.
- Understand the building blocks of procedural programs.
- Understand the use of common data structures.
- Write useful procedural programs in the Python programming language.
- Test and debug programs.
- Understand the basics of file I/O and Object-Oriented Programming in the context of the Python Programming Language.

PART 2: THE C++ PROGRAMMING LANGUAGE (JULY 5 – JULY 18, 2020)
Part 2: The C++ Programming Language teaches the learner to apply foundations of computing and procedural programming to the C++ programming language. The particulars of the C++ programming language syntax and semantics for procedural programming will be covered. This part will also expand the study of programming with low-level programming concepts like static data types, arrays, pointers and basic memory management. Assignments and quizzes will emphasize applying problem solving skills to solving computing problems.

Objectives
At the end of Part 2, the learner should be able to:

- Understand and use the basic procedural programming features of the C++ programming language.
- Understand and use of low-level C++ programming features like static data types, arrays, and pointers.
- Understand and use the basics of dynamic memory management.
- Test and debug programs.
- Write useful procedural programs in the C++ programming language.

Continued on back...
PART 3: DISCRETE MATHEMATICS
(JULY 19 – AUGUST 1, 2020)

**Part 3: Discrete Mathematics** is a first introduction to discrete mathematics with emphasis on its use in computer science. Topics covered include: formal logic, proof techniques, mathematical induction, set theory, combinatorics, probability, relations, functions and graph theory.

**Objectives**

At the end of Part 3, the learner should be able to:

- Understand the important basic concepts of discrete mathematics and their application to computer science and computational problem solving.
- Understand formal logic and the axiomatic method.
- Understand and apply mathematical induction.
- To be able to map theoretical ideas into programs.

PART 4: DATA STRUCTURES
(AUGUST 2 – AUGUST 15, 2020)

**Part 4: Data Structures** extends the study of structured programming by presenting the concepts and implementations of various abstract data types (ADTs) including lists, stacks, queues, trees, hash tables and graphs. The tradeoffs associated with implementing these ADTs with alternative data structures will be discussed. An introduction to algorithm complexity will be covered.

**Objectives**

At the end of Part 4, the learner should be able to:

- Understand and use the most common data structures used in computer programming.
- Understand the tradeoffs between different implementations of a given abstract data type.
- Select the appropriate data structure for a given application.
- Understand and use algorithm complexity to compare different algorithms.

**Course Prerequisites**

This course does not have prerequisites.

**Course Materials**

Learners will need a personal computer capable of running the current standard distribution of Python version 3 and a C++ development environment.

**Grading Policy**

Each part contains five quizzes, two assignments and one final exam. To successfully pass and move forward to the next part, the learner must pass all with a minimum score of 70 percent.

- There is one quiz after every two lectures.
- Multiple quiz attempts are allowed.
- There is one graded assignment after every 5 lectures. Assignments will have a deadline and can only be submitted once.
- The final exam is comprehensive and covers material from all the lectures contained in each part. A limited number of attempts are allowed for the final exam.