Institute of Techgronomy Software Engineering Program

Full Time

Course Packet

Introduction

The Institute of Techgronomy offers a competency-based certification program in the Software Engineering field. The Institute of Techgronomy Software Engineering program will teach students the scope of skills necessary to become a software engineer. Software engineering is about building and 'engineering' software and the technical infrastructure that supports software - everything from large and complex data sets to advanced algorithms. It is, at the core, about solving problems and designing systems that function in the desired manner or that solves a problem.

This program focuses on software engineering principles, as well as strong fundamentals in data structures and algorithms. Learners will cover fundamental computer programming concepts including arrays, strings, algorithms, pointers, hash data structures, and software architecture, before moving on to focusing on front-end and back-end languages including JavaScript, using the terminal, C, Assembly, Shell, virtual machines, sockets, C++ and object-oriented programming, Elixir, network programming, Redis, and advanced algorithms and data structures. Our projects include a focus on software architecture, object-oriented design, and advanced back-end programming. Learners are also expected to complete 30-40 technical interview role plays to prepare for real job interviews, and undergo resume and cover letter reviews similar to peer code reviews. Overall, our Software Engineering program is designed to train learners to Silicon Valley standards in software engineering with an emphasis on structured problem solving, critical thinking, and extensive preparation for meeting employer demands for entry-level jobs.

Students Will Learn...

- Advanced algorithms
- Advanced data structures and databases
- C++/OOP
- Elixir
- Network Programming
- Sockets
- Shell Virtual Machines
- Javascript
- RESTful APIs, software architecture,
- Structured problem solving and debugging,
- Extensive use of industry-standard tools such as Git, IDEs, and terminal commands.

What to Expect

A 12-month remote training program Students will gain experience building and developing software. By the time students complete the program they will earn an industry-standard certification in Software Engineering from Institute of Techgronomy.

No tests, only projects.

Each focus of this program will involve completing projects in teams as well individually to ensure students are learning and applying their knowledge.

Build apps and sites with groups and on your own

Our projects include a focus on software architecture, object-oriented design, and advanced back-end programming. Work in groups and complete individual portfolio projects.

Showcase projects to recruiters

Students will showcase approximately 5 to 20 projects representing thousands of lines of code for employers and interviews.

40-hour-per-week time commitment
Students will need to devote 40 hours a week minimum in order to fully learn the content necessary to pass the course and become a data scientist.

Write ~100K lines of code across 20 projects
On average, students will write about 100,000 lines of code as they complete exercises, software projects, and coding challenges throughout the program. This high-quantity coding means students develop confidence in their code and applied software architecture design and implementation experience.

Interview training
As part of this program, students will complete technical interviews to prepare for job applications. Students will be guided on how to navigate challenging technical interviews including whiteboard coding.

Materials required for this course

- Must have a high school diploma or GED equivalent
- Be fluent in conversational English
- Must be familiar with computer systems
- Must have a working computer, internet access, and basic calculator
- Familiarity with code is recommended
- 15+ hours a week to devote to coursework and projects

Technical Requirements & Specs

- A computer running a compatible web browser (see below)
- Stable Internet connection (video interaction is not available offline)
- Speakers/headphones, microphone, and web-camera to hear audio from a computer (videos have subtitles as an alternative)
- Lessons can be completed on any desktop computer, smart phone, or tablet

Students will also need to ensure they have permission to download the Zoom app to the computer (e.g., if using a loaner or shared computer such as in a library) Supported Web Browsers Google Chrome and Firefox.

Performance will be suboptimal on Internet Explorer; Windows users should download Chrome or use Microsoft Edge (included in Windows 10).

About Competency-based Learning (or Mastery-based Learning)

Our programs and learning methods use competency-based learning, meaning that the goal for the learner is to become competent in a given skill, not to pass a test. They are expected to be able to perform the skill as they would in a job, to an acceptable level, with confidence that they are capable of doing the task.

Passing a test at 75% or even 95% or memorizing information is not reflective of what is required to be a successful developer, analyst, or engineer.

What's important is being able to solve problems, work collaboratively, thinking critically all the time as well as being able to identify and solve security issues quickly and effectively.

For example, in cybersecurity personnel, companies, and even countries and governments, do not want employees who are 85% competent at their job: who wants to stop a hacker from accessing only 85% of your information or database? No one - they should be completely stopped. If cybersecurity training never pushes learners to be fully competent, how will they suddenly become so on the job at a crucial time when they are expected to perform at full competency? The same goes for things such as autopilot: being competent at computer programming truly matters.

It's no surprise that employers prefer candidates who have gone through competency-based training.

The Limits of Passive Learning

The learning curve is notoriously deceptive for learners, and can be incredibly costly for employers. Learners tend to think that receiving a lot of information is equivalent to significant learning, when in fact, learning by doing and through repetition is indicative of significant learning.

Providing a series of lectures or even online videos provides information, but with a 10% retention rate of passive learning, learners aren't actually competent or efficient at performing a task related to the subject area. As a result, passive learning involves a steep learning curve on-the-job, which costs employers in salary, low efficiency, and possibly worse with costly on-the-job errors.

Here is a concrete example tied to coding. Companies use a version control system (VCS) for managing code, and the vast majority use a Git system. Using Git repositories, however, is not straightforward the first few times, especially when working in teams or groups. Since most professional engineers and developers work in teams, knowing how to use Git in a group setting is essential. Receiving an explanation of how to use Git is not sufficient. Learners need to decide how to structure and name their branches of code (sub-sections within the larger code repository), how to divide their work and unite their code, how to merge changes and additions to the code, and how branch permissions actually function. Making an error here can be extremely costly, up to all code and work is lost.

At Institute of Techgronomy, we want learners to make errors during our training programs, and not on the job. Learning to use Git is an example of endless situations where learning by doing results in a more competent employee than passive learning where much of the learning curve is yet to be done.

Project-Based Learning

Project-based learning is an approach to how a person learns that involves providing projects or problems that need to be solved, built, or created.

Learners use the problem as a launching point for researching new concepts, using trial and error, building solutions and learning as they build, evaluating the options available, making decisions about which solution to pursue, and more.

Active Learning
Project-based learning at its core focuses on the learner, not on an instructor or a professor or a teacher delivering knowledge. Project-based learning and peer code reviews are part of active learning: learners must actively participate, engage, and respond. This is different from passive learning such as lectures, presentations, MOOCs, or watching online videos.

Active learning requires learners to analyze, evaluate, and create. The majority of our curriculum is made up of projects which are essentially problems in the form of “build a solution for XYZ.” As learners research, analyze, build, test, and fix, their engagement is through the roof compared with passive learning and the retention rate is closer to 90% (compared to passive learning at 5-10%).

The Project Based Learning Spectrum
Many other tech learning providers claim to do project-based learning, however, just because their learners do one or two projects, it does not mean that they are doing project-based learning.

Above shows a spectrum of project-based learning: any form of knowledge transmission by a single source of truth (e.g. instructor or professor) takes away from the need to research, analyze, or evaluate, meaning learners don’t build critical thinking or problem-solving skills. If you aren’t solving problems but are asking for answers from an instructor, then you are hindered in developing problem-solving skills and learning how to be resourceful.

One of the most important things you need to learn to future-proof yourself is how to learn: unless you’re doing real-project-based learning, you will depend on someone else to provide answers for you.

Peer Learning

Peer learning is how and why learners learn from their peers, whether it’s in a formal learning context or not.

In today’s world, access to knowledge has changed and people are continually learning, growing both their knowledge base and their skills base. In a work environment where projects are king and collaboration is key, people have many more opportunities to learn from their peers naturally as part of their work.

Learning from peers could come in the form of asking for help to debug, sharing useful resources for a given project or exercise, peer code reviews that look at functionality and code quality, or working on a project together and sharing different approaches to the same problem. Peer learning has high engagement, is much more social and interactive, and more fun than passive learning in a lecture.

Why Use Peer Reviews

Students learn in being reviewed and in reviewing, and are operating at the top four levels of the skills pyramid. They must evaluate, analyze, think critically, and create ways to break the submitted solution and construct tests. The peer review process naturally fosters 21st-century skills. Project-based learning and peer code reviews are part of active learning: learners must actively participate, engage, and respond. This is different from passive learning such as lectures, presentations, MOOCs, or watching online videos.

The Peer Learning Spectrum

Many learning systems use a single source of truth or knowledge - a professor or an instructor - as the person who provides answers to particular problems, system designs, or questions. This discourages critical or analytical thinking and creativity.

In a peer learning system, having multiple sources of truth, in addition to significant knowledge access via the internet, learners need to think critically about what is important and why, possible solutions and why they would or would not be good choices, and different ways to solve the same problem.

Using a peer code review system exposes learners to new ways of solving the same problem of different architectures and code structures.

Peer Learning and the Learning Community

In giving and receiving peer reviews, students learn how to communicate about their work and how to give and receive feedback. This is important for developing soft skills as well as preparing for the workplace where peer reviews are generally part of tech jobs.

By participating as a reviewer and a reviewee, learners contribute to the overall learning community. Reviewers also receive feedback from reviewees, providing motivation for reviewers in a manner similar to what occurs in the workplace.

Similar to the process for reviewing academic articles, the peer review process is the evaluation of submitted work by your peers who are competent in their field of knowledge. The peer review process requires analysis, critical thinking, and creativity in order to evaluate submitted work.

The reviewee may have to explain what they have done and why, which helps them to develop communication skills. Reviewers need to understand quickly and identify possible areas where the submitted work does not meet project criteria, standards, or best practices.

TIPP - Technical Interview Preparation Program

Learners join our Technical Interview Preparation Program as they begin working on some of their final projects in the data science track. Learners do 40+ technical interviews to prepare for job applications.

Learners will play the role of both an interviewer and an interviewee as they complete 40 role play situations. Interviewers are given the topic or challenge question as well as the answer to the question in their documentation for each role play. Interviewees do not receive any advanced information until they arrive at the interview where they must complete the challenge. The point of interview practice is to develop specific skills in structured problem solving AND in communication. Practicing the art of communicating your thoughts is vital to successful interviews.

Learners are to treat the role play situations as real-life interviews, meaning you adhere to professional practices, time requirements, and general interview guidelines.
Following the interview, interviewers will complete a review of the role play exercise, providing helpful feedback for the interviewee.

This is one of the most unique elements of this program, as well as one of the most effective interview training programs in industry.

Course learning objectives

- Upon successful completion of this course, a student will meet the following outcomes:
- Investigate the complexity of professional coding practices and comply with corporate coding standards
- Demonstrate ability to recognize and reproduce software engineering best practices and receive feedback
- Identify basic coding challenges and develop the corresponding problem solving skills to resolve them

- Manage control flow in programming.
- Handle scalars (int, char, string).
- Apply memory address management.
- Demonstrate knowledge of basic Data structures (arrays).
- Apply the Agile methodology to manage a project by breaking it up into several phases.
- Establish and maintain a constant collaboration with stakeholders for continuous improvement.
- Select, differentiate, and analyze professional web development tools, techniques, and frameworks
- Demonstrate knowledge of industry best practices to be used to train others in their implementation
- Develop problem-solving skills appropriate for a Software Engineer and determine best solutions for evaluation and decision making applied to existing industry problems
- Use a professional code editor for building and debugging modern web and cloud applications.
- Use the industry standard version control system for tracking changes in computer files and coordinating work on those files among multiple users.
- Use a command line interface to interact with systems, control cloud based systems, and launch development and testing environments.
- Implement best practices for peer code review.
- Design and customize responsive mobile-first sites with Bootstrap, mixing responsive grid systems and extensive prebuilt components.
- Build user interface using efficient, and flexible JavaScript library.
- Compose complex UIs from small and isolated pieces of code called “components” using ReactJS.
- Use web framework Flask to write web applications.
- Apply web security best practices, and efficiently use the framework to avoid common mistakes.
- Create scalable web applications to be run in professional environments.
- Use open-source Python web framework (Django) for rapid and efficient development of secure websites.
- Apply Object Oriented Mapper technologies and SQL.
- Apply fundamentals of software architecture to organize system components.
- Apply Agile methodology to manage a project by breaking it up into several phases.
- Establish and maintain a constant collaboration with stakeholders for continuous improvement.
- Complete technical interview questions using the Question / Writing / Answering method

Course meeting schedule

Season	**Project Name**	**Description**
Preseason	Bootcamp Javascript	
Preseason	My bouncing box	
Preseason	My css is easy	
Preseason	My first backend	Build api that replies to different routes, URL naming structures
Season 1 Software Engineering	Bootcamp C	The coding environment, using the terminal functions, loop statements, types, variables, pointers and strings, arrays and pointers, memory allocation/structures, basic and more complex algorithms, a nested loop with if statements, advanced shell, pipe, multiple commands, 2D arrays and strings
Season 1 Software Engineering	My PrintF	Unlimited arguments, conversion between types and bases
Season 1 Software Engineering	My LS	Unix, architecture, folders and files, sort algorithms, makefile
Season 1 Software Engineering	My Tar	Create archive (zip), how folders and files are made, compliance with POSIX architecture
Season 1 Software Engineering	Readline	String manipulation, read, system call, using instructions that are hardcoded into the CPU
Season 1 Software Engineering	My Blockchain	Linked lists, graphs, parsing of command line, creating a graph that is the data structure behind a blockchain (linked list of linked list)

Season 1 Software Engineering	Core War	Creating virtual machine and compiler that transforms code from Assembly to binary, create parser, create compiler, create virtual machine that executes binary code
Season 2 Software Engineering	Redis Class	Recoding Redis in Ruby with hash data structures
Season 2 Software Engineering	CSS is easy I	Basic CSS with flexbox
Season 2 Software Engineering	ZSH	Parsing, command line, execution of command line, Unix processes (forks and piping)
Season 2 Software Engineering	LibASM	Redoing C library in Assembly
Season 2 Software Engineering	SQL Lite	Redcoding a database and implement an SQL parser (Uses hash data structure)
Season 2 Software Engineering	Malloc (Memory Allocation)	Linked lists, hash tables or trees to optimize memory given to a user, speed of execution of commands, software for hardware areas
Season 2 Software Engineering	Redis	C, coding Redis database Q value with features for hash, command line interface
Season 2 Software Engineering	My FTP	Server-client sockets, protocols, network programming, asynchronous, protocol FTP, file transfer, protocols, network programming, asynchronous, protocol FTP, file transfer, implementation of RFC
Season 2 Software Engineering	My Curl	Client Socket, protocols, network programming, HTTP, HTTP Header
Season 3 Software	Bootcamp C++	Syntax, begin OOP, classes,

Engineering		references, instances, methods
Season 3 Software Engineering	Abstract Virtual Machine	Recode a simple virtual machine in C++, heritage (a concept within OOP), change how you think about programming, docker and containers
Season 3 Software Engineering	My Chat	Client-server, connection between servers, implement RFC-MIRC, chat-room management
Season 3 Software Engineering	My Bsq	Rule-based vs probability-based algorithms
Season 3 Software Engineering	My Bc	Rule-based vs probability-based algorithms
Season 3 Software Engineering	My Mouse	Rule-based vs probability-based algorithms
Season 3 Software Engineering	My System Admin	Docker, install web server with database, server that delivers a website that has to connect with a Postgre Server
Season 3 Software Engineering	My Rabbit MQ	Elixir, syntax, functional programming
Season 3 Software Engineering	My Skype	Client-server and transfer of text, voice, and video via the network, creation of binary protocol, significant, use of audio library and codec library, building a large project
Season 4 Software Engineering	Open-source Project	
Season 4 Software Engineering	Final Project	Kernel

What is Live Coding Session

Live coding is an event we host where we solve a coding problem in front of learners. The key here is to share our thought process, and how we code. This live coding could be made by a technical of Institute of Techgronomy or by another student of the community. These sessions will happen once a week on Tuesday. Learners are informed about the project in advance so they can prepare any questions or any blockers they had while building the project.

Structure: The session is a remote live video call that can last for 1 or 2 hours.

Goal: The goal is to make a concept easy to understand for learners, especially if they are beginners, and share with them good practices.

What is Coding Collaboration Session

Coding collaboration sessions are designed to help students learn to work together. During each session, students of all levels are divided into small groups to work on a timed challenge. Each group will focus on a unique task and, at the conclusion of the session, will share their learnings with the other groups. At the end of the meeting, there is also a quick “Skills Check" quiz to present students with common terminology and questions that they may experience during an interview. This quiz does not count towards the Institute of Techgronomy curriculum as the question difficulty varies greatly and students are not expected to know all answers. These sessions are held once a week on Thursday.

Goals:

- Improve teamwork and critical thinking skills
- Build presentation skills
- Expose students to common interview questions and terminology
- Build community among students

Timeline

- Break into small groups (3-4 students) and assign projects: 10 minutes
- Groups work alone in breakout rooms: 20 minutes
- Program Manager reviews current work: Approx 15 mins (5 min per group)
- Groups return to main call and share work: 10 min
- Students take Skills Check and leave call once completed: 5 min

Course assignments by week (spreadsheet of what to complete each week)

Week	Assignments By Week	
1	Bootcamp JS - Quest 01	Bootcamp JS - Quest 04
	Bootcamp JS - Quest 02	
	Bootcamp JS - Quest 03	
2	Bootcamp JS - my moving box	my levenshtein
	Bootcamp JS - Quest 06	my spaceship
	Bootcamp JS - Quest 07	
3	My hamming DNA	
	My moving box realtime	
	My robot simulator	
4	My bouncing box	
	My css is easy	
	My first backend	
5	Project 1: Bootcamp C	
	Quest00	
	Quest01	
6	Quest02	
	Quest03	
	Quest04	
7	My Square	
	Quest05	
	Quest06	
8	Quest07	
	Quest08	
	My Cat	
9 & 10	My Christmas Tree	
	Project 2: My PrintF	

11 - 12	Project 3: My LS	
13 - 16	Project 4: My Tar	
17 - 18	Project 5: Readline	
19	Project 6: My Blockchain	
20	Season 2 Software Engineering	
	Bootcamp Ruby	
21	Bootcamp Ruby	
	My BSQ	
22 - 23	My BSQ	
24 - 25	My BC	
26 - 27	My Mouse	
28 - 30	My ZSH	

	My CSS is Easy	
	My SQLite	
31		
	My Curl	
32 - 33		
	My LibASM	
34-36		
	My Malloc	
37 - 41		
	Season 3 Software Engineering	Season 3 Rust
	Bootcamp C++	Bootcamp Rust Arc 1
42		
	My Abstract Virtual Machine	My String
	1 technical interview per week	My Mastermind Rust
43 - 44		
	My FTP	Bootcamp Rust Arc 2
	1 technical interview per week	My Ping Server
45 - 46		
	My Skype	My Redis Client
	1 technical interview per week	My Redis Server
47 - 50		
	Season 4 Software Engineering	
	Project 23: Open-source Project	
51		
	Project 23: Open-source Project	
	2 technical interviews	
52		
53	Project 24: Final Project	

	2 technical interviews	

Attendance, Probation, and Leave of Absence

Attendance Policy

Institute of Techgronomy's programs are designed to prepare students for a career in the technology industry and are modeled after what a typical work day may look like. For a student to be successful in this role, they must demonstrate responsibility and reliability. Employers define this as punctuality, regular attendance, and consistent progress. It is expected that students establish these good habits and attend each required Institute of Techgronomy meeting. Required meetings depend on the student's active program and will be communicated before their initial start date. Students are also expected to be on-time and communicate any conflicts in advance with their program manager. Repeated absences could result in disciplinary action, up to and including dismissal from the program. Students must maintain at least a 75% daily attendance record to be considered in good standing. Excused absences do not count towards this figure. Falling below this level may lead to probation and the need to establish a working plan with the student's program manager.

Probation and Dismissal Policy

Institute of Techgronomy Silicon Valley reserves the right to discipline or dismiss any student whose attendance, academic performance, or professional conduct does not meet the standards established by our rules and regulations. A student may be placed on probation for consistently failing to meet attendance or academic standards. Any student on probation must meet with their program manager to establish a learning improvement plan before continuing their curriculum. A student risks dismissal if they continue to not meet the guidelines established in their improvement plan. Any student who has been dismissed may appeal the action within 15 days of the initial dismissal. To file an appeal, a student must send an email to their program director or administrator explaining their reasons or grounds for appealing their dismissal. In the event of dismissal, all tuition through the current month will remain due. If a student has paid in advance, funds outside of the current billing cycle will be returned within 30 days of dismissal.

Leave of Absence Policy

If a student would like to take a "Leave of Absence", they must submit an email to their program manager with their reason for request, expected return date and initial date of request. This does not automatically ensure Institute of Techgronomy's approval and is not considered valid until approval has been granted. A leave of absence may not exceed 60 days and only one will be granted for any given student during a 12-month rolling enrollment period. Upon approval, no tuition is due until the student returns. The Student may resume the program at any time, but in the event the student resumes the program before the expected return date, tuition fees become payable. Upon return, the student must meet with their program manager to re-establish course completion goals.