Welcome To AXSOS Academy Pre-Bootcamp!

Contents

Congratulations on making it this far! We are very excited to have you join our Bootcamp! Only a small percentage of applicants find themselves in your position, yet you've only just begun your journey. Prior to your first day, we need to get you up to speed with expectations, resources, and tools that you'll be heavily using during your time with us.

Getting Started

Your main goal from this bootcamp is to become a great software engineer during these 4 months.

How will we help you to achieve that?

1. **Improve Critical Thinking**: We will help you during Algorithm sessions which will develop your way of thinking (most questions that we will discuss in this session will be an important part of companies interview question)

2. **Reach Black Belt**: Spend all of your time and energy working on our curriculum with a goal to get black belts. Belt exams are real-world-related projects which you will be building for a certain amount of time for each track. We consider belt exams to be an important ingredient of the overall Bootcamp experience leading to your success after the program. The strength you gain by preparing and striving in these exams is a critical ingredient for achieving self-sufficiency as a developer.

Q: **What is the Belt Exam?** it is an exam after each stack (ex: Python) Belt exams are real-world-related projects which you will be building for a certain amount of time for each track. There are 3 levels of Belts:

➢ **Orange Belt**: you have 24 hours to finish the exam

➢ **Red Belt:** you have 5 hours to finish the exam

➢ **Black Belt**: you have 5 hours to finish the exam with additional requirements.

3. **Have a Career**: you will have regular career services during Bootcamp so you will be ready to apply for a job after this Bootcamp.

Our Methodology

1. Flipped learning

An educational approach in which the conventional notion of classroom-based learning is inverted: students are introduced to the learning material before class with classroom time then being used to deepen understanding through discussion with peers and problem-solving activities facilitated by teachers.

In short, in traditional learning students acquire knowledge in a classroom context and are then sent away to synthesize analyze and evaluate this after the class. In the flipped classroom students acquire knowledge before the class and use classroom time to practice and apply concepts and ideas through interaction with peers and teachers. After the class students reflect upon the feedback they have received in class and use this to further their learning. Proponents of the flipped classroom approach emphasize the "deep learning" or higher-level cognitive skills that it encourages.

Flipped Learning Enables:

- Student access to tools and technologies
- Student engagement in rigorous content
- Student immersion in diverse learning
- Student collaboration with peers
- Support for the learning process
- Student access to immediate expert feedback

Flipped Learning in the Classroom:

- Encourages student understanding
- Enables differentiation
- Ensures access to expert support

- Enables student engagement
- Creates a supportive learning environment
- Provides opportunities for collaboration

2. 20-Minute Rule

Although we want you to struggle, we want to control that process so you're not feeling discouraged or spending too much time on one problem to maintain the required pace. After struggling with a problem for 20 minutes, ask your cohort-mates for help. If the two (or more) of you still can't solve the problem in another 20 minutes, ask an instructor. Your instructor won't give you the answer but will help guide you through the steps to figuring out the solution. This process provides you with a taste of the real world but prevents you from getting so discouraged that you give up.

Remember these steps of the 20-Minute Rule case:

Bug encountered; code doesn’t work.

Step 1: Debug for 20 minutes. (Try to solve)

Step 2: Still stuck? Ask a cohort mate or two for help.

Step 3: Still stuck? Reach out to a TA or Instructor.

3. Strength Through Struggle

You are attending Coding Dojo to become a self-sufficient developer. We expect you to struggle through the material. A phrase you'll hear repeated often is "**strength through struggle**". When we ask you to struggle it is because we have seen first-hand that when a person perseveres in solving tough problems, they gain skills faster and retain them for longer. Each time you hit a wall remember that all it takes is time. We will never ask you to solve problems that are impossible. Come with the attitude that you will never give up. Soon, each challenge becomes fun and interesting rather than frustrating.

How to be Successful

You will spend more than 45 hours per week in this Bootcamp

1. Stay positive: See this video https://www.youtube.com/watch?v=CqgmozFr_GM

2. It will be a hard time but always focus on why you are here
https://www.youtube.com/watch?v=u4ZoJKF_VuA

3. Time Management https://www.youtube.com/watch?v=iDbdXTMnOmE

4. Honest and Hardworking

Stack Expectation

Over this time, you will go through an introductory course called Web Fundamentals, then take Python, Java and MERN (JavaScript) for 16 weeks. Web Fundamentals will cover front-end technologies over 4 weeks. Next comes the full stack portion. During the full stack portion of the program, look forward to learning different languages and frameworks: Python with Flask & Django, Java with Java Spring and JavaScript with Node.js and React. Over the course of the program, you're going to learn all the MVC, OOP, modularization, and AJAX techniques to give you the most well-rounded experience possible. Top it off by earning your Black Belt, our highest accolade. After each full-stack ends, you will have a week to work on a project to get your portfolio off.

Our Platforms

1. **Discord**: you will have a discord account for communication with your colleagues, instructor, and TA (Like messenger): This is the support channel Here.
2. **Coding Dojo**: you will have a coding dojo account that will have the whole material and tasks. This will be sent to you by Coding Dojo themselves.
3. **Zoom**: we will send you a zoom link for online lectures.

Introduction to programming

What is a program?

A program is a list of instructions you write for the computer to perform. These instructions can be written in one file or many files, but they are in a form that the computer can understand.

What is programming?

Programming is the act of writing down instructions for the computer. As a programmer, you will be acting as a translator of sorts: you know what you want to do, but now you need to tell the computer how to do it in a way that it understands. The computer can do nearly everything you ask, as long as you tell it in a certain way. There are many ways to tell a computer how to do something, which is what a programming language is.

What is a programming language?

A programming language is one way that you can tell the computer what you want it to do. For example, you can use Python to write a program. You will write Python code in one or more files, and then run the program. Running, or executing, a program is simply telling the computer to perform the tasks that you have written down. Another programming language is JavaScript. In terms of creating and running the program, it is very similar to Python. However, it has a different syntax and set of rules when writing programs. Even though they may have a different syntax, they still are ways to communicate with a computer.

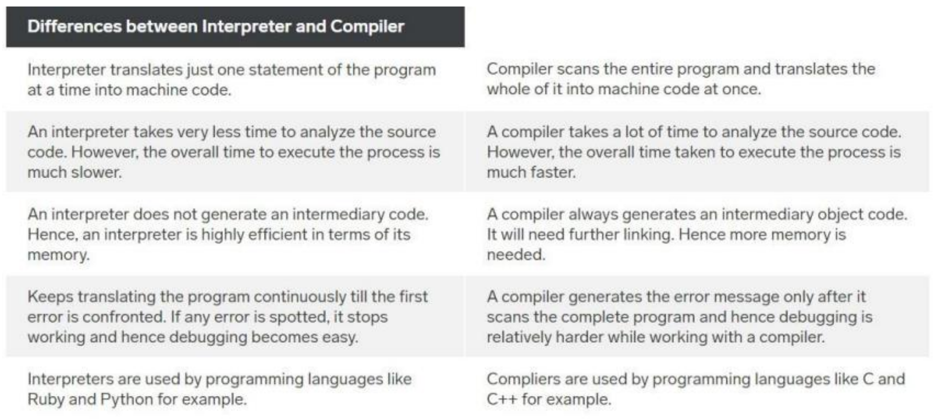
****Let Us Go with this video for some new concepts Introduction to Coding.**

Learn how to learn:

Let's begin with Learning How to Learn as it is a Powerful mental tool to help you master tough subjects, through this link: Learning How to Learn. This course will help you to learn about how the brain uses two very different learning modes and how it encapsulates (“chunks”) information. It will also cover illusions of learning, memory techniques, dealing with procrastination, and best practices shown by research to be most effective in helping you master tough subjects.

Compilers VS interpreters

Compilers and interpreters are programs that help convert the high-level language like Python and JavaScript (Source Code) into machine codes (ones and zeros) to be understood by the computers. Computer programs are usually written in high level languages. A high-level language is one that can be understood by humans. To make it clear, they contain words and phrases from the languages in common use – English or other languages for example. However, computers cannot understand high level languages as we humans do. They can only understand the programs that are developed in binary systems known as a machine code. To start with, a computer program is usually written in high level language described as a source code. These source codes must be converted into machine language and here comes the role of compilers and interpreters.



Differences between Interpreter and Compiler	
Interpreter translates just one statement of the program at a time into machine code.	Compiler scans the entire program and translates the whole of it into machine code at once.
An interpreter takes very less time to analyze the source code. However, the overall time to execute the process is much slower.	A compiler takes a lot of time to analyze the source code. However, the overall time taken to execute the process is much faster.
An interpreter does not generate an intermediary code. Hence, an interpreter is highly efficient in terms of its memory.	A compiler always generates an intermediary object code. It will need further linking. Hence more memory is needed.
Keeps translating the program continuously till the first error is confronted. If any error is spotted, it stops working and hence debugging becomes easy.	A compiler generates the error message only after it scans the complete program and hence debugging is relatively harder while working with a compiler.
Interpreters are used by programming languages like Ruby and Python for example.	Compliers are used by programming languages like C and C++ for example.