Program Information

NavGurukul's flagship residential program works with learners from underserved communities to help them become programmers and gives them a guaranteed job once they finish the program. Currently, NavGurukul is training 1000+ students in seven residential centers across various states in India, 3 in partnerships with state governments. Six of these seven campuses have only girl students.The focus on gender is a conscious choice, given the dismal representation of the women in the IT sector and service sector alike. The learning on NavGurukul campuses is driven by an innovative self-paced pedagogy which is effective and scalable. The program is offered free of cost to ensure that it's accessible to low-income communities and enables them to get opportunities otherwise limited to only certain privileged social demographics.

Program Objectives

NavGurukul envisions a world where everyone has access to an affordable world class education that enables them to live up to their highest potential. The program objective is to enable underserved, under-resourced and underprivileged communities to get aspirational jobs, have a voice, and be equipped to support change in their social and economic surroundings. Software programming becomes a medium or a tool for the larger transformation in the lives of these students wherein they are able to garner skills, self-belief and opportunities irrespective of their social circumstances. The program aims at inclusive education that celebrates the successes of a collective, and enables sustainable financial independence for all students.

It is a 15-month fully-funded residential training program on Software Programming which provides a guaranteed job of Rs. 20,000/month as a starting salary after successful completion of the program. Student's stay, food, laptop and internet facilities are provided and they would just need to take care of their personal and medical expenses. Alongwith training on Software Programming, students will also be trained in English communication, personality development, leadership skills, financial and digital literacy, socio emotional learning and so on.

Pedagogy and Teaching Methodology

NavGurukul follows a unique pedagogical approach to solve this social challenge. It follows a self-paced, self-directed, learning-by-doing pedagogy with a strong emphasis on peer learning . What this essentially means is that students follow a detailed carefully curated self directed learning curriculum. This acts as a huge enabler as NavGurukul is able to accommodate students' pace based on their initial level of understanding, learning style and competencies, as well as their personal responsibilities that might arise.

Students learn programming not by just remembering or theoretically understanding the concepts, but by experimenting and problem solving in real time as well. This enables the students to gain application oriented understanding that lasts with them as they have done it themselves. This also enables students to build their confidence and self-esteem. NavGurukul also has a strong emphasis on peer-learning. A strong peer-mentoring system supports the mentee to align with the curriculum and get guidance wherever needed, while the student mentor acquires the confidence, social skills and better understanding of the concepts. NavGurukul closely tracks each student's progress and builds in interventions as and when needed.

The curriculum for learning programming should be **dynamic** and updated regularly to reflect the fast-changing needs of the IT sector and evolution of AI. Programming involves giving a computer a set of instructions to execute using a programming language. There are many different programming languages, each with its own syntax, features, and applications. Some of the most popular ones include Python, Java, JavaScript, C, and HTML. Programming also requires understanding the basic elements of a program, such as variables, data types, operators, functions, control flow, lists, and loops. These concepts are essential for any language and can help learners develop logical thinking and problem-solving skills. However, programming is not a static field. **New technologies, frameworks, libraries, and tools** are constantly emerging and evolving. To keep up with the latest developments and trends, programmers need to update their knowledge and skills regularly. Therefore, the **curriculum for learning programming should be flexible and adaptable** to incorporate new topics and resources as they become available. This way, learners can stay relevant and competitive in the IT sector and AI domain.

Student & Success centric Education

Throughout our six years of operations, we have consistently adhered to and established the following principles that are essential for designing a student-centric and success-oriented higher education:

1. We **continuously update and revise our curriculum** to align with the industry needs and expectations.

2. We enable our students to opt for **placements** as soon as they find suitable opportunities, and allow them to continue their learning at their own **pace**. Therefore, we have made the last two modules of our course optional for those students who secure a placement before completing them.

3. We **preserve** and uphold the distinctive features of Navgurukul's pedagogy that set us apart from the **conventional** colleges, as they are the key factors that contribute to the remarkable results and outcomes that we have achieved over the traditional programs or higher education.

We also communicate effectively to all our students and stakeholders, how:

4. The ultimate indicator of a successful course is the **placement rate** and quality of our students. **Grades** are misleading, as they do not necessarily reflect the life outcomes or career prospects of our students. Hence, our curriculum emphasizes the importance of focusing on the end goals and outcomes rather than the grades.

5. The inputs that each student brings to the course - such as their hard work, their values, their support to their peers, etc. - are critical to create a culture that fosters collaboration and enables everyone to achieve their goals in the shortest possible time. Hence, we cultivate these inputs from day one of our course.

Placements and Internship

We have been successful in securing placements for over 500 students in top companies, start-ups, tech and non-tech organisations etc. The students are placed either into full-time jobs, or in internships that convert into jobs or give them the right exposure and experience to launch their careers.Our efforts have resulted in providing opportunities for young professionals to kickstart their careers and contribute to the growth of various industries. Our average placement package is 2.5 LPA, and our highest package so far has been 8 LPA. Our aim is to enable our students to access careers in companies where they can learn and grow in a holistic work environment.

Our rigorous training programs and industry-relevant curriculum have enabled our students to stand out in the competitive job market. We take pride in all our alumni who have worked hard and fought through innumerable challenges to secure prestigious job offers.

Learning scheme

1 credit = 15 hours of teaching

1 credit = 30 hours of lab/self-practice

Break into 6 months each

After every 6 months, option for an exit

Map according to NSQF Levels

Module 0 - Gap Material - Basic Math			
Sr. No.	**Course Code**	**Course Title**	**Credit**
1	**M01**	**Arithmetic**	**4**
2	**M02**	**Algebra**	**6**

Module 1 - Number systems and Boolean Logic			
Sr. No.	**Course Code**	**Course Title**	**Credit**
1	**N01**	**Number Systems**	**4**
2	**N02**	**Binary Numbers**	**4**
3	**N03**	**Logic Gates**	**4**
4	**U01**	**Unix Operating System Commands**	**4**

Module 2 - Problem-solving using flowcharts			
Sr. No.	**Course Code**	**Course Title**	**Credit**
1	**F01**	**Introduction to Flowcharts**	**2**

2	F02	Operators and Conditionals	4
3	F03	Loops	4
4	F04	Nested Loops	4
5	F05	Lists	6

Module 3 - Problem-solving using C			
Sr. No.	**Course Code**	**Course Title**	**Credit**
1	C01	Introduction to Programming in C	6
2	C02	Introduction to Online Judge and problem solving using C	8
3	C03	C Pointers, Memory Management and Basic DSA	12

Module 4 - Learning Javascript & Web Development Fundamentals (Optional)			
Sr. No.	**Course Code**	**Course Title**	**Credit**
1	J01	Javascript / ES6	16
2	J02	Web Fundamentals I - Client/Server, Request/response, Browser/Web Server, IP address, DNS	6
3	J03	Web Fundamentals II - Development tools, Git, IDE, REST	6
4	J04	Web Scraping & API	6

Module 5 - Capstone Project in React (Optional)			
Sr. No.	**Course Code**	**Course Title**	**Credit**
1	P01	Capstone Project in React	10

Gap Material - Basic Math

Teaching Scheme	
Total Time	**Credit**
2-5 weeks	10

Objectives

1. Bridge the gap that exists between the students' understanding of mathematical concepts and what is needed to learn computer programming
2. Understand the basic concepts of Mathematics
3. Understanding of Algebraic equations
4. Significance in Mathematics in Programming
5. Given the focus on the students from the underprivileged backgrounds, the module is important to ensure a baseline understanding of Mathematics before learning programming

Outcomes

At the end of the course, a learner should be able to:

1. Solve basic mathematical expressions
2. Solve math word problems
3. Solve algebraic equations
4. Know the significance of the aforementioned concepts in Programming.

Syllabus

The module will cover:

1. Arithmetic
    a. BODMAS
    b. Modular Arithmetic
    c. Comparison Operators
2. Algebra
    a. Venn Diagrams
    b. Linear Equations

Module 1 - Number systems and Boolean Logic

Teaching Scheme	
Total Time	**Credit**
3-5 Weeks	**16**

Objectives

1. This module provides the students with an understanding of Number systems.
2. Understanding of binary number systems and being fluent in Binary Arithmetic.
3. Conversion from one number system to another.
4. Understanding of Logic gates and Boolean Logic

Outcomes

At the end of the course, a learner should be able to:

1. Understand the concept and applications of number systems, such as decimal, binary and octal
2. Convert numbers between different number systems and perform arithmetic operations on them.
3. Learn the basics of binary numbers and how they are used to represent data and information in computers.
4. Identify and use different types of logic gates, such as AND, OR, NOT, NAND, NOR, XOR, and XNOR.
5. Construct and analyze simple logic circuits using logic gates and truth tables.
6. Learn the fundamentals of Unix operating system commands and how to use them to perform various tasks on a terminal.
7. Navigate the file system, create and manipulate files and directories, and use pipes and redirections.
8. Use common Unix commands, such as ls, cd, pwd, cp, mv, rm, cat, grep, find, sort, wc, chmod, ps, kill, etc.

Syllabus

1. **Number Systems**
   - Ancient Number Systems
   - History of Numbers
   - History of Numerical Systems
   - Odometer Simulation
   - Parity Magic

2. **Binary Numbers**
   - Binary Counting
   - Binary Representation
   - Decimal to Binary Conversion
   - Binary to Decimal Conversion
   - Binary Addition

- Binary Subtraction

3. **Logic Gates**
   - What are Logic gates?
   - Importance of Logic gates
   - Truth Tables
   - NOT Gate
   - OR Gate
   - And Gate
   - XOR Gate
   - Boolean Operators

4. **Unix Operating System & Commands**
   - Introduction to Unix operating system and its features
   - Unix terminal and shell commands
   - File system structure and navigation
   - File and directory operations
   - Pipes and redirections
   - Common Unix commands and utilities

Module 2 - Problem-solving using flowcharts

Teaching Scheme	
Total Time	**Credit**
8-10 Weeks	**20**

Objectives

1. To develop an understanding of how to give instructions to the computer without dealing with the complexity of any programming language syntax.
2. To develop an understanding of how computers execute instructions
3. To develop an understanding of various programming constructs
4. To develop an ability to think and develop efficient solutions to a computing problem
5. To develop an ability to debug a program

Outcomes

At the end of the course,
1. Students will understand the process flow of how computers understand instructions.
2. Students will understand the concept of variables and learn how to name them.
3. Students will explore and analyze the instructions (start-stop, input-output, process) of flowcharts and create flowcharts for simple problems.
4. Students will learn about Arithmetic Operators (+ , - , * , /, // , %, **)
5. Students will learn to use Decision box

6. Students will learn to use Relational operators (<, <=, ==, >, >=, !=) and Logical Operators (AND, OR, NOT)
7. Students will learn to formulate flowcharts using these operators.
8. Students will revise the concept of dry run
9. Students will learn to create a loop
10. Students will understand the need for applying loops in situations where a set of repeating statements occur
11. Students will implement loops and conditionals to solve problems.
12. Students will be able to tell the need for lists
13. Students will be able to create a list and add elements
14. Students will be able to access an element at a particular position in a list
15. Find the length of a list
16. Change the value of any variable in the list

Syllabus

1. **Introduction to Flowcharts**
    a. Importance of Flowcharts
    b. Introduction to symbols used in a Flowchart.

2. **Operators and Conditionals**
    a. Arithmetic Operators
    b. Operation Box in Flowcharts
    c. Relational operators
    d. Decision Boxes in Flowcharts
    e. Formulating a Flowchart
    f. Dry Runs
3. **Loops**
    a. What are Loops?
    b. Use of Loops in Programming
    c. Loops in Flowchart
    d. Conditional Loop
    e. Infinite Loop
    f. Nested Loop
4. **Lists**
    a. List- Concept
    b. Need of Lists
    c. Indexing
    d. Terminologies of Lists
    e. Accessing Elements
    f. Modifying Values in a List
    g. Size of a List

Module 3 - Problem Solving Using C

Teaching Scheme	
Total Time	**Credit**
8-9 Weeks	**24**

Objectives

1. **Introduction to Programming in C**: The course is designed to provide complete knowledge of C language. Students will be able to develop logics which will help them to create programs and applications in C. Also by learning the basic programming constructs they can easily switch over to any other language in future.

2. **Introduction to Online Judge and Problem Solving using C**: The course aims to provide exposure to problem-solving through programming using online judge platforms. Students will learn how to read problem statements, design algorithms, write and test code, submit solutions and analyze feedback. Students will also learn how to use various tools and techniques to optimize their code for performance and memory efficiency.

3. **C Pointers, Memory Management and Basic DSA**: The course introduces the concepts of pointers, dynamic memory allocation, memory management and basic data structures and algorithms in C. Students will learn how to use pointers to manipulate arrays, strings, linked lists, etc. Students will also learn how to implement common algorithms using pointers.

Outcomes

1. **Introduction to Programming in C**: Students will be able to write, compile, debug and run basic C programs using variables, operators, expressions, statements and control structures. Students will also be able to understand the syntax and semantics of C language and use them to solve simple problems.

2. **Introduction to Online Judge and Problem Solving using C**: Students will be able to use online judge platforms to practice and improve their programming and problem-solving skills in C. Students will also be able to apply various tools and techniques to optimize their code for performance and memory efficiency.

3. **C Pointers, Memory Management and Basic DSA**: Students will be able to use pointers to manipulate data in memory and implement basic data structures and algorithms in C. Students will also be able to understand the concepts of dynamic memory allocation, memory management and memory leaks and how to avoid them.

Syllabus

- **Introduction to Programming in C**:
    - Fundamentals of computing and programming
    - Data types, constants, variables and operators in C

    - Input/output statements and formatted output
    - Selection statements and logical operators
    - Repetition statements and loops
    - Arrays and strings
    - Functions and parameter passing
    - Scope and lifetime of variables

- **Introduction to Online Judge and Problem Solving using C**:
    - Online judge platforms and problem statements
    - Algorithm design and pseudo-code
    - Code testing and debugging
    - Code submission and feedback
    - Time and space complexity analysis
    - Performance optimization techniques
    - Memory optimization techniques
    - Common errors and pitfalls

- **C Pointers, Memory Management and Basic DSA**:
    - Pointers and address arithmetic
    - Dynamic memory allocation and deallocation
    - Memory management and memory leaks
    - Pointers and arrays
    - Pointers and strings
    - Pointers and functions
    - Linked lists using pointers

Module 4 - Learning Javascript & Web Development Fundamentals (Optional)

Teaching Scheme	
Total Time	**Credit**
10-12 Weeks	**30**

Objectives

- **Javascript / ES6**: The course is designed to provide a comprehensive introduction to JavaScript, the most widely used scripting language for web development. Students will learn the syntax, features, and best practices of modern JavaScript (ES6), as well as how to use it to create dynamic and interactive web pages. Students will also get a glimpse of how to use JavaScript libraries and frameworks such as jQuery or React to enhance their web development skills.

- **Web Fundamentals I - Client/Server, Request/response, Browser/Web Server, IP address, DNS**: The course aims to provide a solid foundation of web development by explaining the basic concepts and components of the web. Students will learn how the web works, how web browsers and web servers communicate using HTTP requests and responses, how IP addresses and DNS work, and how to use web development tools such as Chrome DevTools and Postman.

- **Web Fundamentals II - Development tools, Git, IDE, REST**: The course introduces the essential tools and techniques for web development, such as version control, code editors, debugging tools, and RESTful APIs. Students will learn how to use Git and GitHub for managing their code, how to choose and use an IDE (Integrated Development Environment) such as Visual Studio Code or Atom, how to debug and test their code using tools such as Chrome DevTools and Postman, and how to consume and create RESTful APIs using JSON and AJAX.
- **Web Scraping using JS & using APIs**: The course teaches how to extract data from websites and web APIs using web scraping techniques in JavaScript. Students will learn how to use Node.js libraries such as Axios to use APIs in their project.

Outcomes

- **Javascript / ES6**: Students will be able to write, run, and debug JavaScript code using modern features and syntax. Students will also be able to use JavaScript to manipulate the Document Object Model (DOM) and add interactivity and functionality to web pages. Students will also be able to use JavaScript libraries and frameworks such as jQuery or React to create basic web applications.
- **Web Fundamentals I - Client/Server, Request/response, Browser/Web Server, IP address, DNS**: Students will be able to understand the basic architecture and components of the web. Students will also be able to use web development tools such as Chrome DevTools and Postman to inspect and analyze HTTP requests and responses. Students will also be able to understand how IP addresses and DNS work and how they affect web browsing.
- **Web Fundamentals II - Development tools, Git, IDE, REST**: Students will be able to use essential tools and techniques for web development, such as version control, code editors, debugging tools, and RESTful APIs. Students will also be able to use Git and GitHub for managing their code, how to choose and use an IDE such as Visual Studio Code or Atom, how to debug and test their code using tools such as Chrome DevTools and Postman, and how to consume and create RESTful APIs using JSON and AJAX.
- **Web Scraping using JS & using APIs**: Students will be able to use web scraping techniques in JavaScript to extract data from websites and web APIs. Students will also be able to use Node.js libraries such as Axios to use APIs in this project.

Syllabus

- **Javascript / ES6**:
  - Introduction to JavaScript and ES6
  - Variables, data types, operators, and expressions
  - Control structures, loops, and functions
  - Arrays, objects, and JSON
  - Document Object Model (DOM) and events
  - AJAX
  - Basics of React/jQuery

- **Web Fundamentals I - Client/Server, Request/response, Browser/Web Server, IP address, DNS**:
  - Introduction to web development and web architecture
  - HTTP protocol and request/response cycle
  - Web browsers and web servers
  - Chrome DevTools and Postman
  - IP addresses and DNS
  - Domain names and hosting
  - URL structure and query parameters
  - Cookies and sessions

- **Web Fundamentals II - Development tools, Git, IDE, REST**:
  - Introduction to web development tools and techniques
  - Version control and Git
  - GitHub and GitHub Pages
  - Code editors and IDEs
  - Debugging and testing tools
  - RESTful APIs and CRUD operations
  - JSON and AJAX
  - Fetch API and Axios

- **Web Scraping using JS & using APIs**:
  - Introduction to web scraping and web APIs
  - Node.js and npm
  - Axios
  - User Agent and anti-scraping measures

Module 5 - Capstone Project using React (Optional)

Teaching Scheme	
Total Time	**Credit**
4-6 Weeks	**10**

Objectives

- The course is designed to provide students with hands-on experience of working on a large-scale web application using ReactJS, a popular front-end library for building user interfaces.
- Students will learn how to read, understand, and modify an existing codebase that uses ReactJS and other web development technologies, such as HTML, CSS, JavaScript, and APIs.
- Students will also learn how to use various tools and techniques for developing, testing, debugging, and deploying a ReactJS application, such as code editors, Chrome DevTools, Git, GitHub Pages, and React Developer Tools.
- Students will apply their knowledge and skills to create a capstone project that showcases their ability to use ReactJS to build a dynamic and interactive web application that meets the given requirements and specifications.
- Students will also demonstrate their understanding of the concepts and best practices of ReactJS, such as components, props, state, hooks, routing, and custom hooks.

Outcomes

- Students will be able to work on a large-scale web application using ReactJS and other web development technologies, such as HTML, CSS, JavaScript, and APIs.
- Students will be able to read, understand, and modify an existing codebase that uses ReactJS and follow the code style and conventions.
- Students will be able to use various tools and techniques for developing, testing, debugging, and deploying a ReactJS application, such as code editors, Chrome DevTools, Git, GitHub Pages, and React Developer Tools.
- Students will be able to create a capstone project that showcases their ability to use ReactJS to build a dynamic and interactive web application that meets the given requirements and specifications.
- Students will be able to demonstrate their understanding of the concepts and best practices of ReactJS, such as components, props, state, hooks, routing, and custom hooks.

Syllabus

- Introduction to the course and the capstone project
- Choosing an existing open source project to develop your capstone project on
- Setting up the development environment and tools
- Exploring and understanding the existing codebase and its structure, style, and conventions
- Identifying the requirements and specifications for your capstone project
- Planning and designing your capstone project using ReactJS and other web development technologies

- Implementing your capstone project using ReactJS and other web development technologies
- Testing and debugging your capstone project using various tools and techniques
- Deploying your capstone project using GitHub Pages or other hosting services
- Presenting and demonstrating your capstone project and its features

Reading Material

We provide students with high-quality content and curriculum for learning which is either created or curated by us. We regularly update and revise our curriculum to align with the current and future demands of the sector and the advancements of AI.