Welcome to the ACA Summer Camp!

(Automation Coding Academy)

NOTE THAT THIS IS BACKEND ONLY COURSE

SOME URLS MIGHT BE SHARED IN CLASS FOR REFERENCES

TOOLS :

TEXT EDITOR : VSCODE - PREFFERED

PYTHON 3.6+++

POSTMAN

IKOTUN .C.

During the next two weeks, we will embark on an exciting journey into the world of programming. Programming has become an essential skill in today's technology-driven world. It empowers us to solve problems, automate tasks, and create innovative solutions that can shape the future.

In this course, we will explore the concept of programming and why it is so important in various fields. Even if you're not planning to become a professional programmer, learning programming can significantly benefit you in your personal and professional life. It enhances logical thinking, problem-solving abilities, and creativity, making you more adaptable and resourceful.

Our main focus will be on the **Python programming language**. Python is widely regarded as one of the best languages for beginners due to its simplicity and readability. We will cover all the essential building blocks of Python that are crucial to getting started with API development using Django.

Here are some of the topics we will cover:

Introduction to Programming: Understand what programming is and its significance in today's world. We'll explore real-life examples of how programming has transformed various industries.

Why Programming Matters: Discover the vast range of applications for programming, from creating websites and apps to automating tasks and analyzing data.

- Imagine you have a magic box, and you can tell this box what you want it to do. You can say things like "Draw a smiling sun" or "Count to ten" or "Tell me a funny joke."

- Now, the magic box doesn't understand our regular words like we do. Instead, it only understands a special language called "code." Code is like a secret language that only the magic box can understand.
- So, to make the magic box do what we want, we need to write the instructions in this special code. We write something like "Draw a big circle" or "Show the number 1 to 10" using the secret code.
- Once we have written all the instructions in the secret code, we give it to the magic box. The magic box reads the code and follows our instructions. It draws the sun, counts to ten, or tells us a joke, just like we asked!
- That's what programming is all about - giving instructions to the magic box (computer) using a special secret language (code) to make it do all sorts of fun and useful things!

Introduction to Python: Get acquainted with the Python programming language. We'll cover the basics of installing Python and setting up the development environment.

Python Essentials: Learn about variables, data types, and how to perform basic operations in Python. This foundation will be essential as we progress further.

Functions and Control Flow: Explore the concept of functions and how they help organize code. We'll also cover conditional statements (if, else, elif) and loops (for and while) to control the flow of our programs.

Collections in Python: Dive into Python collections, such as lists and tuples. We'll learn how to manipulate these data structures efficiently.

Input and Output: Understand how to interact with users through input and display output on the screen. We'll create interactive programs to make our code more engaging.

Introduction to Django: Explore the basics of Django, a powerful web development framework in Python. We'll see how it simplifies building web applications and websites.

By the end of this course, you will have a solid understanding of programming concepts and **Python fundamentals.** You'll be equipped to start your journey in web development using Django and have the skills to build exciting and interactive applications.

Get ready for an exciting and enriching learning experience in the world of programming! We can't wait to see what you create!

Why Coding??

Learning programming is valuable and important, even if individuals are not directly working in the field of software development or computer science. Here are some compelling reasons why programming is beneficial for individuals from various backgrounds and professions:

Problem-Solving Skills: Programming encourages the development of analytical and problem-solving abilities. It teaches you how to break down complex problems into smaller, manageable tasks, and then develop step-by-step solutions to tackle them. These problem-solving skills are transferable to many real-life situations, helping individuals make better decisions and solve everyday challenges more effectively.

Automation and Efficiency: Programming enables the automation of repetitive tasks. By writing code to perform these tasks, individuals can save time and effort in their daily routines. Automation allows them to focus on more critical and creative aspects of their work, leading to increased productivity and efficiency.

Data Analysis and Decision-Making: With programming, individuals can analyze and process large datasets to extract valuable insights and make data-driven decisions. This skill is beneficial in various fields, from marketing and finance to healthcare and research.

Creativity and Innovation: Programming is an expressive medium that allows individuals to create and innovate. It enables the development of interactive websites, mobile apps, games, and more. Embracing programming empowers individuals to turn their creative ideas into reality.

Enhanced Communication with Developers: For individuals working with software developers or technology teams, having a basic understanding of programming helps facilitate better communication. It allows them to convey their requirements and ideas more effectively and understand the development process.

Digital Literacy: In the digital age, programming is a fundamental aspect of digital literacy. Understanding how computers and software work empowers individuals to interact with technology more confidently and adapt to technological advancements.

Adaptability and Versatility: Programming languages are versatile tools that can be applied in various domains. Whether you work in finance, healthcare, education, or any other industry, programming skills can be tailored to suit your specific needs and tasks.

Career Advancement: In many industries, individuals with programming skills have a competitive advantage. They are often considered more valuable assets by employers, as technology continues to play an essential role in modern workplaces.

Entrepreneurship and Side Projects: Knowing how to code opens up opportunities for entrepreneurship and personal projects. Individuals can create their websites, apps, or digital products without relying on others, allowing them to pursue their entrepreneurial aspirations or passion projects.

Understanding Technology's Impact on Society: In a technology-driven world, programming knowledge helps individuals understand the impact of technology on society, privacy, security, and ethical considerations related to digital advancements.

In summary, programming is a valuable skill that goes beyond the boundaries of the software development industry. It equips individuals with problem-solving abilities, critical thinking, and creativity, making them more effective and adaptable in their chosen fields. Whether you're a business professional, scientist, artist, or educator, programming can enrich your skillset and open up new opportunities for personal and professional growth.

Terms

Frontend:

Frontend development is the process of creating the user interface and user experience of a

website or mobile application. It focuses on designing and implementing the visual elements and interactions that users see and interact with directly. Frontend development involves three essential technologies:

HTML (Hypertext Markup Language): HTML is the backbone of the web. It provides the structure and content of a web page, defining headings, paragraphs, images, links, forms, and more. HTML uses tags to mark up the elements and organizes them into a hierarchical structure.

CSS (Cascading Style Sheets): CSS is used to control the presentation and layout of HTML elements. It allows developers to specify styles like colors, fonts, spacing, and positioning. CSS separates the content from the presentation, enabling consistent and visually appealing designs across the website or app.

JavaScript: JavaScript is a powerful scripting language that enables dynamic interactions on web pages. It allows developers to add interactivity, animations, and event handling. With JavaScript, you can create responsive user interfaces and perform actions based on user input.

Tools Used in Frontend Development:

Code Editor: Developers use code editors like Visual Studio Code, Sublime Text, or Atom to write and edit HTML, CSS, and JavaScript code.

Version Control: Tools like Git and GitHub help track changes to the codebase, collaborate with other developers, and manage project versions.

Frontend Frameworks and Libraries: Popular frameworks like React, Angular, or Vue.js provide pre-built components and tools to streamline frontend development.

Backend in Terms of SSR and API:

Backend development involves building the server-side of web applications, handling business logic, and managing data. It enables communication between the frontend and the server, ensuring data processing, storage, and retrieval. Two primary approaches to backend development are:

Server-Side Rendering (SSR): SSR is a technique in which the server generates the complete HTML content and sends it to the client (browser). The client receives a fully rendered page, ready to display to the user. SSR is beneficial for search engine optimization (SEO) and initial page load performance.

API (Application Programming Interface): APIs are a set of rules and protocols that allow different software applications to communicate and interact with each other. In the context of backend development, APIs enable the frontend to request and exchange data with the server dynamically. APIs can return data in various formats, such as JSON or XML, which the frontend can use to update the user interface without a full page reload.

Database:

A database is a structured collection of data that stores information in a way that is easily retrievable and manageable. In web development, databases are essential for storing and organizing various types of data, such as user profiles, content, and application settings. Common types of databases used in web development include relational databases like MySQL and PostgreSQL, and NoSQL databases like MongoDB and Firebase.

In summary, frontend development focuses on creating the user interface using HTML, CSS, and JavaScript, while backend development handles the server-side logic and communication with the frontend. Server-Side Rendering (SSR) and APIs play key roles in exchanging data between the

frontend and the server. Databases are used to store and manage application data efficiently. Together, these components form a complete web application or mobile app.

Schedule :

Introduction to Programming

What is programming?

Why is programming important?

Exploring real-life examples of programming in action.

Introduction to the magic box analogy.

Basic concepts like giving instructions and understanding code.

High-Level and Low-Level Languages

Explanation of high-level and low-level languages in simple terms.

Understanding that computers only understand low-level languages, but we use high-level languages to talk to them.

Using the magic box analogy to illustrate the difference between high-level and low-level languages.

Introduction to Python

What is Python?

Why is Python a good language for beginners?

Installing Python and setting up the development environment.

Writing and running our first Python program: "Hello, World!"

Variables and Data Types in Python

Understanding variables and how they store information.

Introduction to different data types like numbers, strings, and booleans.

Basic operations with variables and data types.

Practice exercises to reinforce the concepts.

Functions and Control Flow

Explaining the concept of functions as reusable blocks of code.

Understanding if-else statements for decision-making.

Introduction to loops (for and while) for repetitive tasks.

Practice exercises to apply functions, if-else, and loops.

Intermediate Python and Introduction to Django

Modules and Operators

Exploring how modules provide additional functionality to Python.

Understanding various Python operators like arithmetic, comparison, and logical operators.

Applying modules and operators to solve problems.

Classes and Object-Oriented Programming (OOP)

Introduction to classes as blueprints for creating objects.

Understanding the concept of attributes and methods in classes.

Relating OOP concepts to everyday objects (e.g., cars, animals).

Hands-on exercises with classes.

Introduction to Django

What is Django, and why is it popular for web development?

Understanding the components of a Django project: models, views, templates, and URLs.

Setting up a simple Django project and running the development server.

Why APIS , why not SSR ?

Virtual Environment and Dependencies

Explaining the importance of virtual environments for isolating project dependencies.

Installing and managing Python packages using pip.

Creating a virtual environment for the Django project.

Building a Simple Django Web App

(Testing with Postman)

Combining Python and Django concepts to create a basic web application.

Creating models for data storage.

Writing views to handle user requests and display data.

Designing templates for the user interface.

Final project presentation and sharing accomplishments.

HIGH LEVEL / LOW LEVEL :

High-Level Language: Imagine you're writing instructions for the magic box using words and sentences in your own language (English or any other language you speak). High-level languages are like that; they are easy for us to read and understand because they use words and structures similar to human languages. Examples of high-level languages include Python, JavaScript, and Java.

Low-Level Language: In contrast, low-level languages use much simpler and more direct instructions that the magic box can understand directly. It's like speaking the magic box's secret language directly. Low-level languages are harder for us to read and understand because they are designed for the magic box, not for humans. Examples of low-level languages include Assembly language and machine code.