FULL STACK PYTHON

IIT CURRICULUM

7995599720 www.clearit.com md.rahamath6685@gmail.com

Introduction to Full stack Python Overview

What is Full Stack Python?
Why Full stack Python?
Where is it used?
Career Opportunities
Learning Journey

Module 01: Core Python

Python Introduction & setup environment

what are the software's required to learn python

Python version installation
Visual studio code installation

An identifier (variable)

What an identifier(variable)
Rules for an identifier(variable)

Module 02: Data types in Python

Numeric Types:

int: Integer values
float: Floating-point values, or decimal numbers
complex: Complex numbers

Sequence Types:

str: String, a sequence of characters
list: Ordered, mutable collection of items
tuple: Ordered, immutable collection of items
range: Immutable sequence of numbers

Mapping Type:

- dict: Unordered collection of key-value pairs

Set Types:

set: Unordered, mutable collection of unique items

frozenset: Immutable version of a set

Boolean Type:

bool: Represents True or False

Binary Types:

bytes: Immutable sequence of bytes

byte array: Mutable sequence of bytes

memor yview: Provides memory access to the binary data

None Type:

NoneType: Represents the absence of a value

Module 03: Type Casting in Python

Introduction to Type Casting

Definition of type casting

Importance of type casting in Python

Implicit vs Explicit type casting

Implicit Type Casting (Automatic Conversion)

When Python automatically converts one data type to another

Benefits and limitations of implicit type casting

Explicit Type Casting (Manual Conversion)

Methods for explicit type casting

Common built-in functions used for type casting:

int() – Convert to integer

float() – Convert to float

str() – Convert to string

- list() – Convert to list
- tuple() – Convert to tuple

set() – Convert to set
dict() – Convert to dictionary
bool() – Convert to boolean

Type Casting Between Numeric Types

Conversion between integers, floats, and complex numbers
Handling precision loss in conversions

Type Casting Between Sequences

Casting between lists, tuples, and sets
Converting strings to lists or tuples

Casting with Strings

Converting strings to numbers and vice versa
Handling errors during invalid conversions

Complex Type Casting

Working with complex numbers
Converting real numbers to complex numbers

Type Casting in Conditional Statements

Casting for logical conditions
Type conversions in if and while conditions

Common Errors in Type Casting

ValueError: When conversions fail
Handling exceptions during type conversion

Module 04: Types of Operators in Python

Introduction to Operators

Definition of operators

- Role of operators in expressions
- Types of operators in Python

Arithmetic Operators

Used for mathematical operations

Operators:

+ : Addition

- : Subtraction

* : Multiplication

/ : Division (returns float)

// : Floor Division (returns integer)

% : Modulus (remainder of division)

** : Exponentiation

Comparison (Relational) Operators

Used to compare two values

Operators:

== : Equal to

!= : Not equal to

> : Greater than

< : Less than

>= : Greater than or equal to

<= : Less than or equal to

Assignment Operators

Used to assign values to variables

Operators:

= : Simple assignment

+= : Add and assign

-= : Subtract and assign

*= : Multiply and assign

/= : Divide and assign

//= : Floor divide and assign

%= : Modulus and assign

- **= : Exponentiation and assign

Logical Operators

Operators:

and : Returns True if both statements are true

or : Returns True if one of the statements is true

not : Reverses the result, returns False if the result is true

Bitwise Operators

Operators:

& : AND

| : OR

^ : XOR

~ : NOT

<< : Left Shift

>> : Right Shift

Membership Operators

Operators:

in : Returns True if a value is found in the sequence

not in : Returns True if a value is not found in the sequence

Identity Operators

Operators:

is : Returns True if both variables are the same object

is not : Returns True if both variables are not the same object

Ternary (Conditional) Operator

Allows conditional expressions in a compact form

Syntax: value_if_true if condition else value_if_false

Operator Precedence

Understanding operator precedence

Importance of parentheses to override precedence

Precedence order from highest to lowest:

**

*, /, %, //

+, -

Comparisons: ==, !=, >, <, >=, <=

Logical: not, and, or

Module 05: Control Flow Statements in Python

Introduction to Control Flow

Definition of control flow

Importance of control flow in programming

Overview of control flow statements in Python

Conditional Statements (Selection Statements)

if Statement

Syntax and basic usage

if-else Statement

Adding an alternative block of code when the condition is false

if-elif-else Statement

Multiple conditions

Nested if Statements

Using an if statement inside another if

Looping Statements (Iteration Statements)

While Loop

- Executes a block of code as long as the condition is true
- Syntax and usage

for Loop

Loops over a sequence (list, tuple, dictionary, string, or range)

Syntax and usage

Nested Loops

Placing one loop inside another

Loop Control Statements

break Statement: Exits the loop prematurely

continue Statement: Skips the current iteration and moves to the next

pass Statement: Does nothing, used as a placeholder for future code

Iterating Over Data Structures

for loop with different data types:

Strings

Lists

Tuples

Dictionaries

Sets

List Comprehensions

Using for loops to generate lists in a concise way

The else Clause in Loops

Using the else clause in for and while loops

Explanation and examples

Switch / Jumping/ Match Statements

Module 06: Functions in Python

Introduction to Functions

- Introduction to Modules, Packages, Libraries
- Definition of a function
- Advantages of using functions

Code reusability
Modularity
Maintainability
Built-in functions vs user-defined functions

Function Syntax and Basics

Function syntax
def keyword
Function name
Parameters
Return statement

Function Parameters and Arguments

Positional Arguments:

Passing arguments in the correct order

Keyword Arguments:

Using parameter names to pass arguments

Default Parameters:

Setting default values for parameters

Variable-Length Arguments:

*args: Passing a variable number of non-keyword arguments
**kwargs: Passing a variable number of keyword argument

Return Statement

Returning values from a function
Returning multiple values as tuples

Scope of Variables

- Local Variables: Variables defined inside a function
- Global Variables: Variables defined outside all functions
- Modifying global variables inside functions using the global keyword

Lambda Functions

Definition of lambda (anonymous) functions
Syntax of lambda functions
When to use lambda functions

Higher-Order Functions

Definition: Functions that take other functions as arguments or return functions

Map, Filter, and Reduce Functions

map(): Applies a function to all elements in an iterable
filter(): Filters elements based on a function
reduce(): Applies a function cumulatively to the elements of an iterable (from the functools module)

Recursion

Definition of recursive functions
Understanding base cases and recursive cases
Common use cases for recursion
Tail recursion and its limitations in Python

Function Decorators

Definition of decorators
Syntax and usage of decorators
Common built-in decorators: @staticmethod, @classmethod, @property

Function Annotations

Syntax of function annotations
Adding metadata to function arguments and return values

Nested Functions

Functions inside other functions

- Accessing outer function variables inside inner functions (closure)

Generators and yield Statement

Difference between functions and generators

Use of yield in place of return to create generators

Module 07: Object-Oriented Programming (OOP)

Review of Basic OOP Concepts

Class and Object

Attributes (Instance and Class Variables)

Methods (Instance and Class Methods)

The self-keyword

Constructor (__init__method)

Inheritance in Python

Definition of inheritance

Types of inheritance:

Single inheritance

Multiple and Multilevel inheritance

Hierarchical inheritance

Hybrid inheritance

Overriding methods

super() function and method resolution order (MRO)

Polymorphism

Definition of polymorphism in OOP

Polymorphism with functions and objects

Method overriding vs method overloading in Python

Duck typing

Encapsulation and Abstraction

Definition of encapsulation

Private and protected members in Python

- Getters and setters
- Definition of abstraction
- Abstract classes and methods using the abc module

Class and Static Methods

Difference between class methods and static methods
Use cases of class methods and static methods
@classmethod and @staticmethod decorators
Class vs instance level data

Properties and Property Decorators

Definition of properties in Python
Creating getter, setter, and deleter methods using @property
Read-only properties

Operator Overloading

Definition and importance of operator overloading
Overloading arithmetic operators
Overloading comparison operators
Overloading assignment operator

Special (Magic/Dunder) Methods

Overview of magic methods in Python
Common dunder methods

Abstract Base Classes (ABC)

Role of abstract base classes in OOP
Creating abstract classes using the abc module
Implementing interfaces in Python using abstract methods

Module 08: File Handling in Python

Introduction to File Handling

Definition of file handling
Importance of working with files in programming

- Types of files: text files vs binary files

File Operations

Opening files in different modes:

Read (r)

Write (w)

Append (a)

Read and write (r+, w+, a+)

Binary modes (rb, wb, ab, etc.)

Closing a file after operations

Reading and Writing to Files

Reading from a file:

Reading the entire file at once

Reading line by line

Using read(), readline(), readlines() methods

Writing to a file:

Writing strings to a file

Writing multiple line

Using writelines() for writing iterables

File Pointers and File Positioning

Understanding the file pointer

Using seek() to change the file pointer position

Using tell() to get the current file pointer position

Working with Binary Files

Reading and writing binary data

Differences between text and binary file handling

Handling binary data with struct module (overview)

Context Manager for File Handling

The with statement for automatic file management

- Benefits of using context managers for file handling

File Methods

Overview of file object methods like:

- flush()
- truncate()
- fileno()
- isatty()

Directory and File Operations

Working with directories and files using the os module

Creating, renaming, and deleting files and directories

Checking file/directory existence with os.path module

Getting file properties (size, modification time, etc.)

File Handling with shutil Module

Copying files and directories

Moving files

Deleting files and directories

Error Handling in File Operations

Handling file I/O errors

Common file-related exceptions (FileNotFoundError, IOError, etc.)

Module 09: Exception Handling in Python

Introduction to Exception Handling

What are exceptions?

Importance of exception handling in programming

Difference between errors and exceptions

The try, except, finally Block

Structure of exception handling in Python:

- try block for executing code that may raise an exception
- except block for catching exceptions
- finally block for cleanup actions

Catching Multiple Exceptions

Catching different types of exceptions in a single try-except block

Handling multiple exceptions with multiple except clauses

Using else with try-except Block

Using the else block to execute code only if no exceptions occur

Raising Exceptions

Manually raising exceptions using the raise keyword

Customizing the exception message

Custom Exception Classes

Creating user-defined exceptions

Defining custom exception classes for specific use cases

Exception Hierarchy in Python

Understanding the built-in exception hierarchy

Common built-in exceptions: ValueError, TypeError, KeyError, IndexError, etc.

Handling Nested Exceptions

Nested try-except blocks

Propagating exceptions from inner to outer blocks

Logging Exceptions

Logging exceptions using the logging module

Customizing log messages for exception handling

Multi-Threading in Python

process based

Thread based

Regular Expressions

- Synchronization

Module 10: Django

Introduction to Django
What is Django
Features of Django
How to create a project
How to create application
Working with complete file structure in Django after creating Django project & application
How to create more than one application
How to create a urls.py file at application to improve performance
Working with MVT design pattern
Working with templates folder for frontend development
Working with Static folder for frontend design development
Implementing JavaScript in Django
Implementing bootstrap in Django
Working with model class in Django
Working with Django forms

Working with Django model relationship

One To One Relationship
Many To One Relationship
Many To Many Relationship

Django Exceptions

Working with predefine exception
Working with custom exception
Django ORM
Django Cookies & Sessions implementations
Django Custom Routing
Django Image uploading
Django file uploading

Authentication, Authorization and Website Integration

Module 11: SQL Data Base

Introduction to SQL

History and evolution of SQL
SQL vs NoSQL
Types of databases (RDBMS, column-based, key-value, etc.)
Database concepts: Tables, Rows, Columns, Relationships

SQL Data Types

Numeric types (INT, FLOAT, DECIMAL)
Character types (CHAR, VARCHAR, TEXT)
Date and time types (DATE, TIME, TIMESTAMP)
Boolean types
BLOB (Binary Large Object)

Database Design

Normalization (1NF, 2NF, 3NF, BCNF)
Denormalization
Primary keys, foreign keys, and unique keys
Indexing
Constraints (NOT NULL, DEFAULT, UNIQUE, CHECK)

Basic SQL Queries

SELECT statement
WHERE clause and logical operators (AND, OR, NOT)
ORDER BY clause
LIMIT and OFFSET clauses
DISTINCT keyword

SQL Functions

Aggregate functions (COUNT, SUM, AVG, MIN, MAX)
Scalar functions (UPPER, LOWER, LENGTH, ROUND)

- Date functions (NOW, CURDATE, DATE_ADD, DATE_SUB)

Joins in SQL

INNER JOIN
LEFT JOIN (or LEFT OUTER JOIN)
RIGHT JOIN (or RIGHT OUTER JOIN)
FULL OUTER JOIN
CROSS JOIN
Self joins

Subqueries and Nested Queries

Single-row subqueries
Multi-row subqueries
Correlated subqueries
EXISTS and NOT EXISTS clauses

Set Operations

UNION and UNION ALL
INTERSECT
EXCEPT (or MINUS)

Data Manipulation Language (DML)

INSERT statement
UPDATE statement
DELETE statement
TRUNCATE statement

Data Definition Language (DDL)

CREATE TABLE
ALTER TABLE (add, modify, drop columns)
DROP TABLE
CREATE VIEW, DROP VIEW

Constraints in SQL

- PRIMARY KEY constraint
- FOREIGN KEY constraint

UNIQUE constraint
CHECK constraint
DEFAULT constraint

Transactions in SQL

ACID properties (Atomicity, Consistency, Isolation, Durability)
COMMIT and ROLLBACK
SAVEPOINT
Transaction isolation levels (READ UNCOMMITTED, READ COMMITTED, REPEATABLE READ, SERIALIZABLE)

Indexes in SQL

Purpose of indexes
Types of indexes (single-column, multi-column)
Unique and non-unique indexes
Full-text index
Index performance considerations

SQL Views

Creating views
Updating views
Dropping views
Advantages and limitations of views

Stored Procedures and Functions

Creating stored procedures
IN, OUT, and INOUT parameters
Creating user-defined functions
Differences between stored procedures and functions

Module 12: HTML

Introduction to HTML

What is HTML?
HTML and the World Wide Web
Role of HTML in Web Development
HTML Editors and Development Environment Setup
Basic HTML Document Structure (DOCTYPE, <html>, <head>, <body>)

HTML Document Structure

HTML Elements and Tags
Block-level vs Inline Elements
HTML Attributes (Global and Element-specific Attributes)
Void Elements (e.g., <img>,
, <input>)

Text Formatting and Semantics

Paragraphs, Headings, and Divisions
Semantic HTML: <header>, <footer>, <article>, <section>
Text-level elements: , , , etc.
Lists: Ordered (<ol>) and Unordered (<ul>) Lists
Quotes: Blockquote and Inline Quotes

Links and Navigation

Creating Hyperlinks with <a>
Linking to External and Internal Resources
Email Links and Telephone Links
Image Links
Navigation Bars and Menus (with <nav>)

Images and Multimedia

Inserting Images with <img>
Alt Attribute and Image Descriptions

- Responsive Images (<picture>, srcset)
- Embedding Audio (<audio>)
- Embedding Video (<video>)

Using <iframe> for External Content (e.g., YouTube)

Tables

Creating Tables: <table>, <tr>, <td>, <th>
Table Headers, Footers, and Captions
Colspan and Rowspan Attributes
Table Accessibility Considerations

Forms and Input Handling

Form Structure: <form>, action, method
Common Input Types: Text, Password, Email, Number, Date, etc.
Checkboxes, Radio Buttons, and Select Dropdowns
Textarea and Submit Buttons
Form Validation (Required Fields, Pattern Matching)
Labeling Forms and Improving Accessibility

HTML5 Semantic Elements

The Role of Semantic HTML in Modern Development
New Structural Elements in HTML5 (<header>, <footer>, <main>, <aside>)
Using <section> and <article> for Content Segmentation
Benefits for SEO and Accessibility

Embedded Content

Embedding External Resources with <iframe>
Inline SVG Graphics
Embedding External Stylesheets and JavaScript Files
The <embed> and <object> Elements for External Applications (PDF, Flash)

Module 13: CSS

Introduction to CSS

What is CSS?

- History and Evolution of CSS
- Advantages of CSS in web development
- Types of CSS: Inline, Internal, External

Basic CSS Syntax and Structure
CSS Selectors: Element, ID, Class, Universal, Grouping

CSS Box Model

Understanding the Box Model
Margins, Borders, Padding, and Content
Box-sizing property

CSS Selectors in Depth

Attribute Selectors
Pseudo-Classes and Pseudo-Elements
Combinators: Descendant, Child, Adjacent, General Sibling

CSS Layout Techniques

Positioning: Static, Relative, Absolute, Fixed, Sticky
Display Property: Block, Inline, Inline-Block, None
Float and Clear
CSS Flexbox and CSS Grid: Introduction and Key Properties

Typography in CSS

Font Properties: Font-Family, Font-Size, Font-Weight, Font-Style
Text Properties: Text-Align, Text-Transform, Text-Decoration, Line-Height
Using Web Fonts

Styling Links and Lists

Styling Hyperlinks: Link States
Styling Ordered, Unordered, and Definition Lists

Colors, Backgrounds, and Borders

Color Models: RGB, RGBA, HEX, HSL, HSLA
Background Properties: Background-Color, Background-Image, Background-Position, Background-Repeat, Background-Attachment

- Border Properties: Border-Width, Border-Style, Border-Color, Border-Radius
- Gradients: Linear, Radial

CSS Units and Values

Absolute Units: px, pt, cm, mm
Relative Units: em, rem, vw, vh, %, fr
Calculations using the calc() function

CSS Transitions and Animations

CSS Transitions: Transition Properties, Timing Functions
CSS Animations: Keyframes, Animation Properties

Responsive Design with CSS

Media Queries: Breakpoints and Usage
Viewport Meta Tag
Responsive Units: %, vw, vh, rem, em
Mobile-First Approach
Flexbox and Grid for Responsive Layouts

CSS Variables (Custom Properties)

Declaring and Using CSS Variables
Scope and Inheritance of Variables

Browser Compatibility and Vendor Prefixes

Handling Cross-browser Compatibility
Vendor Prefixes for Different Browsers: -webkit-, -moz-, -ms-, -o-
Tools for Compatibility Testing

Advanced CSS Features

CSS Grid Advanced Techniques: Grid Areas, Template Layouts
Advanced Flexbox Layout Patterns
CSS Shapes and Masks
CSS Clip-Path Property
CSS Filters: Blur, Grayscale, Drop Shadows, etc.

- Advanced Selectors (Nth-child, Nth-of-type)

CSS for Web Accessibility

- Ensuring Text Readability and Color Contrast
- Focus and Active States for Keyboard Navigation
- CSS Guidelines for Accessible Web Design

CSS Grid vs. Flexbox

- When to Use Grid vs. Flexbox
- Differences and Use Cases

Module 14: JavaScript

Introduction to JavaScript

- History and Overview
  - Brief history of JavaScript
  - ECMAScript and standardization
- Setting Up the Development Environment
  - Browsers and DevTools
  - Node.js setup (optional)
- Basic Syntax
  - Comments, variables, keywords
  - Data types and type coercion
  - Expressions and operators

JavaScript Fundamentals

- Variables and Scope
  - var, let, and const
  - Hoisting
  - Global, local, block scope
- Data Types
  - Primitive types: string, number, boolean, null, undefined, symbol, bigint
  - Complex types: object, array, function
- Type Conversion
  - Implicit and explicit conversion
  - typeof operator

Control Structures

- Conditionals
  - if, else if, else
  - Ternary operator
  - switch statement
- Loops
  - for, while, do...while
  - Iterating over objects and arrays (for...in, for...of)
  - break and continue

Functions

- Defining Functions
  - Function declarations and expressions
  - Arrow functions
  - Immediately Invoked Function Expressions (IIFE)
- Parameters and Arguments
  - Default parameters
  - Rest parameters and spread syntax
- Scope and Closures
  - Lexical scoping
  - Closures and practical use cases
- Callback Functions
  - Synchronous vs asynchronous callbacks

Object-Oriented Programming (OOP) in JavaScript

- Objects
  - Creating objects (object literals, new Object())
  - Accessing and modifying object properties
- Prototypes
  - Prototype chain
  - Prototypal inheritance
- Classes and Inheritance
  - Defining classes (class keyword)
  - Constructors

- Class inheritance (extends, super)
- Static methods and properties

Arrays and Advanced Array Methods

- Array Basics
  - Creating arrays, accessing elements
  - Array length, adding/removing elements
- Iterating Over Arrays
  - forEach(), map(), filter(), reduce(), some(), every()
- Array Mutability
  - Array methods that modify vs return new arrays
- Multi-dimensional Arrays
  - Working with nested arrays

Error Handling and Debugging

- Types of Errors
  - Syntax errors, runtime errors, logical errors
- Error Handling
  - try...catch block
  - finally statement
  - Throwing custom errors
- Debugging Tools
  - Using browser DevTools
  - Debugging with console methods (log, warn, error, time)

Asynchronous JavaScript

- Callbacks
  - Defining and using callbacks
- Promises
  - Creating and consuming promises
  - then(), catch(), and finally()
  - Promise chaining

- Async/Await
  - Writing asynchronous code with async and await
  - Error handling in async functions
- Event Loop
  - How JavaScript handles asynchronous operations
  - Microtasks and macrotasks

Document Object Model (DOM) Manipulation

- Understanding the DOM
  - DOM tree and nodes
- Selecting Elements
  - getElementById(), querySelector(), etc.
- Manipulating Elements
  - Changing content (innerHTML, textContent)
  - Changing attributes, classes, styles
- Event Handling
  - Adding event listeners (click, keydown, etc.)
  - Event delegation
  - Preventing default behavior

Browser APIs

- Timers
  - setTimeout(), setInterval()
- Local Storage and Session Storage
  - Storing and retrieving data
- Fetch API
  - Making HTTP requests
  - Handling responses, JSON parsing
- Geolocation API
- Web Workers
  - Multithreading with web workers

Modular JavaScript

Modules

ES6 modules (export, import)

Default and named exports

CommonJS and AMD

require() and module.exports

Bundlers

Using tools like Webpack or Parcel

Regular Expressions

Basics of Regular Expressions

Syntax and pattern matching

Common Methods

test(), exec()

String methods using regex (match(), replace())

Flags and Modifiers

Module 15: Bootstrap

Introduction to Bootstrap

Overview of Bootstrap

History and evolution of Bootstrap

Importance of responsive design in web development

Installation and setup of Bootstrap (via CDN, npm, or manual download)

File structure of Bootstrap

Bootstrap Grid System

Understanding the Bootstrap grid system

Grid layout and breakpoints

Building responsive layouts with the grid system

- Understanding container, row, and column classes
- Nesting grids and offsetting columns

Typography and Basic Elements

- Bootstrap's typography system
- Headings, paragraphs, and text utilities
- Lists, blockquotes, and code elements
- Inline elements and contextual text classes

Bootstrap Components

- Overview of Bootstrap components
- Buttons and button groups
- Forms: Form controls, input groups, layout options, and validation
- Navigation: Navbar, navs, and tabs
- Dropdowns and modals
- Alerts, badges, and breadcrumbs
- Cards and media objects

Utilities and Helpers

- Utility classes in Bootstrap
- Margin, padding, and spacing utilities
- Display and visibility classes
- Sizing utilities for width, height, and viewport settings
- Flexbox utilities for alignment, distribution, and order
- Text alignment and font utilities
- Background and color utilities

Advanced Components

- Carousel and image sliders
- Collapse and accordions
- Tooltips and popovers
- Pagination and progress bars
- Scrollspy and sticky navigation

Bootstrap Icons and Customization

Introduction to Bootstrap Icons
Adding and customizing Bootstrap Icons
Customizing Bootstrap with Sass variables
Overriding Bootstrap styles
Creating custom themes with Bootstrap

Module 16: React JS

Introduction to React JS

What is React?
History and evolution of React
Key features of React
Understanding Single Page Applications (SPAs)
React vs Other Frontend Frameworks (Vue, Angular)

Setting up the Development Environment

Node.js and npm installation
Installing React using Create React App (CRA)
Project folder structure in React
Overview of development tools (VS Code, React Developer Tools)

JSX (JavaScript XML)

Introduction to JSX
JSX vs HTML
Embedding JavaScript expressions in JSX
JSX attributes and children

Components in React

Types of Components: Functional and Class-based
Component lifecycle (Introduction)
Creating and exporting components
- Component reusability

Props in React

- Passing data with props
- Default props
- Prop types (validating props)

State in React

- What is state in React?
- Managing local state in functional components
- The useState hook
- Updating and manipulating state

Event Handling

- Handling events in React
- Passing arguments to event handlers
- Synthetic events

Conditional Rendering

- Using if-else for conditional rendering
- Ternary operators and logical && for rendering

Lists and Keys

- Rendering lists in React
- Using keys in lists
- Handling dynamic data in lists

Forms in React

- Controlled vs Uncontrolled components
- Handling form inputs
- Form submission and validation

Lifting State Up

- Lifting state to a common ancestor
- Sharing state between components

React Router

Introduction to React Router
Setting up routing in a React application
Route parameters and navigation
Nested routes and redirection

React Hooks

Introduction to Hooks in React
useState, useEffect, useContext hooks
Rules of Hooks
Custom hooks and when to use them

Managing Side Effects with useEffect

Introduction to side effects
Fetching data with useEffect
Cleaning up effects
Dependency arrays in useEffect

Context API

Introduction to React Context
Creating a Context
Providing and consuming context
When to use Context vs props

Performance Optimization

Introduction to React performance optimizations
Memoization with React.memo and useMemo
Reducing unnecessary re-renders
Lazy loading with React.lazy and Suspense

Higher-Order Components (HOCs)

- Introduction to HOCs
- Creating and using HOCs
- Use cases for HOCs

Redux (State Management)

- Introduction to Redux
- Setting up Redux in a React application
- Actions, Reducers, and Store
- Connecting Redux to React components with react-redux
- Understanding the Redux flow

Deployment of React Applications

- Building a React application for production
- Hosting React apps on platforms like Netlify, Vercel, or GitHub Pages
- Optimizing bundle size and performance for deployment