Introduction to Linux Shell and Shell Scripting

Last Updated : 16 Apr, 2024

If we are using any major operating system, we are indirectly interacting with the **shell**. While running Ubuntu, Linux Mint, or any other Linux distribution, we are interacting with the shell by using the terminal. In this article we will discuss Linux shells and shell scripting so before understanding shell scripting we have to get familiar with the following terminologies:

- Kernel
- Shell
- Terminal

Table of Content

What is Kernel?

The kernel is a computer program that is the core of a computer's operating system, with complete control over everything in the system. It manages the following resources of the Linux system –

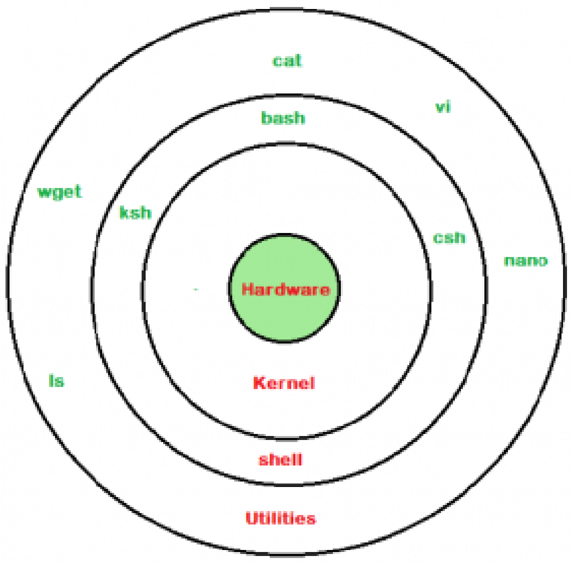
- File management
- Process management
- I/O management

It is often mistaken that Linus Torvalds has developed Linux OS, but actually, he is only responsible for the development of the Linux kernel.

Complete Linux system = Kernel + GNU system utilities and libraries + other management scripts + installation scripts.

What is Shell?

A shell is a special user program that provides an interface for the user to use operating system services. Shell accepts human-readable commands from users and converts them into something which the kernel can understand. It is a command language interpreter that executes commands read from input devices such as keyboards or from files. The shell gets started when the user logs in or starts the terminal.



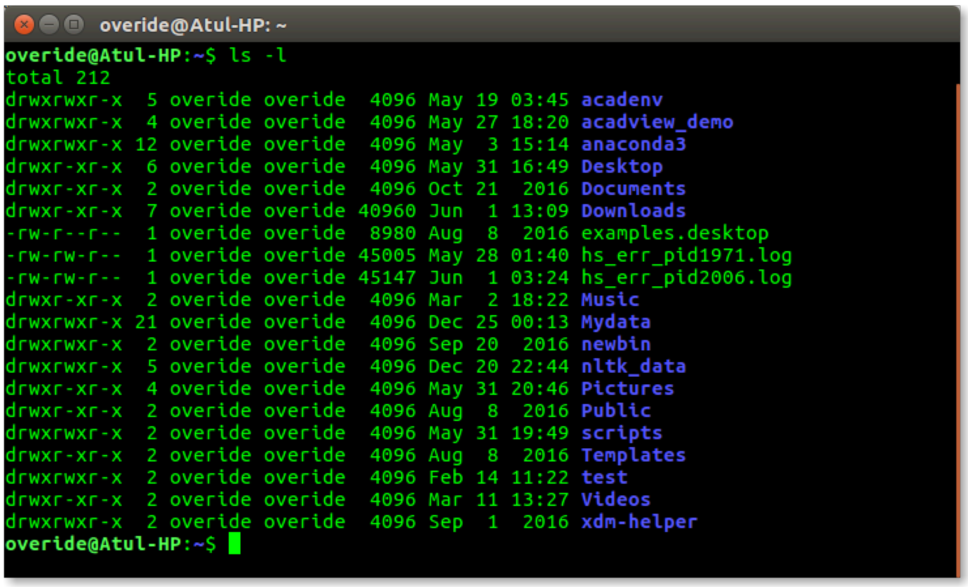
Linux Shell

Shell is broadly classified into two categories –

- Command Line Shell
- Graphical shell

Command Line Shell

Shell can be accessed by users using a command line interface. A special program called Terminal in Linux/macOS, or Command Prompt displayed on the terminal to the user. A terminal in Ubuntu 16.4 system looks like this –



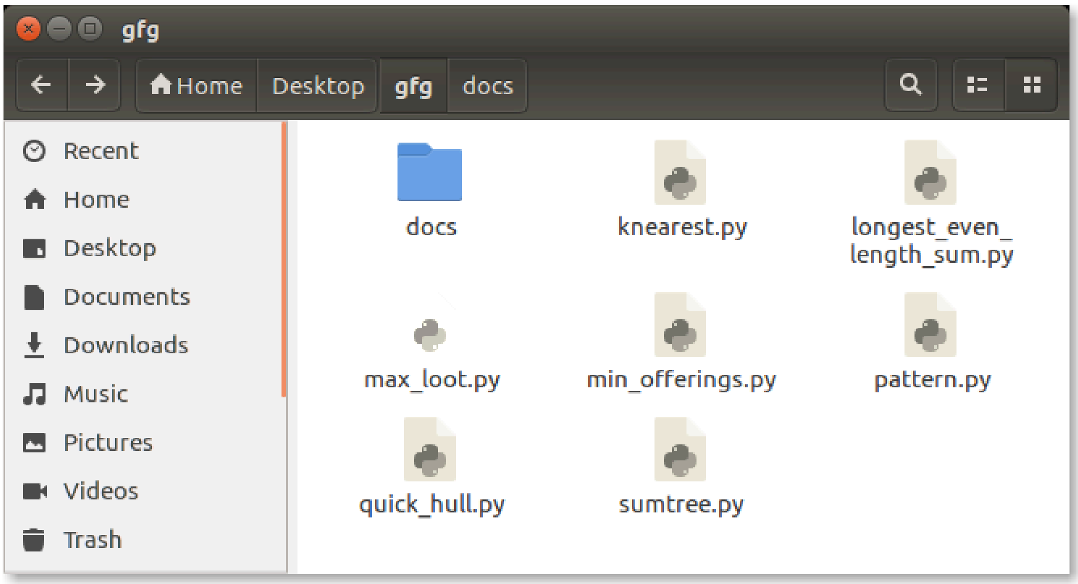
```
overide@Atul-HP: ~
overide@Atul-HP:~$ ls -l
total 212
drwxrwxr-x  5 overide overide  4096 May 19 03:45 acadenv
drwxrwxr-x  4 overide overide  4096 May 27 18:20 acadview_demo
drwxrwxr-x 12 overide overide  4096 May  3 15:14 anaconda3
drwxr-xr-x  6 overide overide  4096 May 31 16:49 Desktop
drwxr-xr-x  2 overide overide  4096 Oct 21  2016 Documents
drwxr-xr-x  7 overide overide 40960 Jun  1 13:09 Downloads
-rw-r--r--  1 overide overide  8980 Aug  8  2016 examples.desktop
-rw-rw-r--  1 overide overide 45005 May 28 01:40 hs_err_pid1971.log
-rw-rw-r--  1 overide overide 45147 Jun  1 03:24 hs_err_pid2006.log
drwxr-xr-x  2 overide overide  4096 Mar  2 18:22 Music
drwxrwxr-x 21 overide overide  4096 Dec 25 00:13 Mydata
drwxrwxr-x  2 overide overide  4096 Sep 20  2016 newbin
drwxrwxr-x  5 overide overide  4096 Dec 20 22:44 nltk_data
drwxr-xr-x  4 overide overide  4096 May 31 20:46 Pictures
drwxr-xr-x  2 overide overide  4096 Aug  8  2016 Public
drwxrwxr-x  2 overide overide  4096 May 31 19:49 scripts
drwxr-xr-x  2 overide overide  4096 Aug  8  2016 Templates
drwxrwxr-x  2 overide overide  4096 Feb 14 11:22 test
drwxr-xr-x  2 overide overide  4096 Mar 11 13:27 Videos
drwxrwxr-x  2 overide overide  4096 Sep  1  2016 xdm-helper
overide@Atul-HP:~$
```

linux command line

In the above screenshot "**ls**" command with "**-l**" option is executed. It will list all the files in the current working directory in a long listing format.
Working with a command line shell is a bit difficult for beginners because it's hard to memorize so many commands. It is very powerful; it allows users to store commands in a file and execute them together. This way any repetitive task can be easily automated. These files are usually called batch files in Windows and **Shell** Scripts in Linux/macOS systems.

Graphical Shells

Graphical shells provide means for manipulating programs based on the graphical user interface (GUI), by allowing for operations such as opening, closing, moving, and resizing windows, as well as switching focus between windows. Window OS or Ubuntu OS can be considered as a good example which provides GUI to the user for interacting with the program. Users do not need to type in commands for every action. A typical GUI in the Ubuntu system –



GUI Shell

There are several shells are available for Linux systems like –

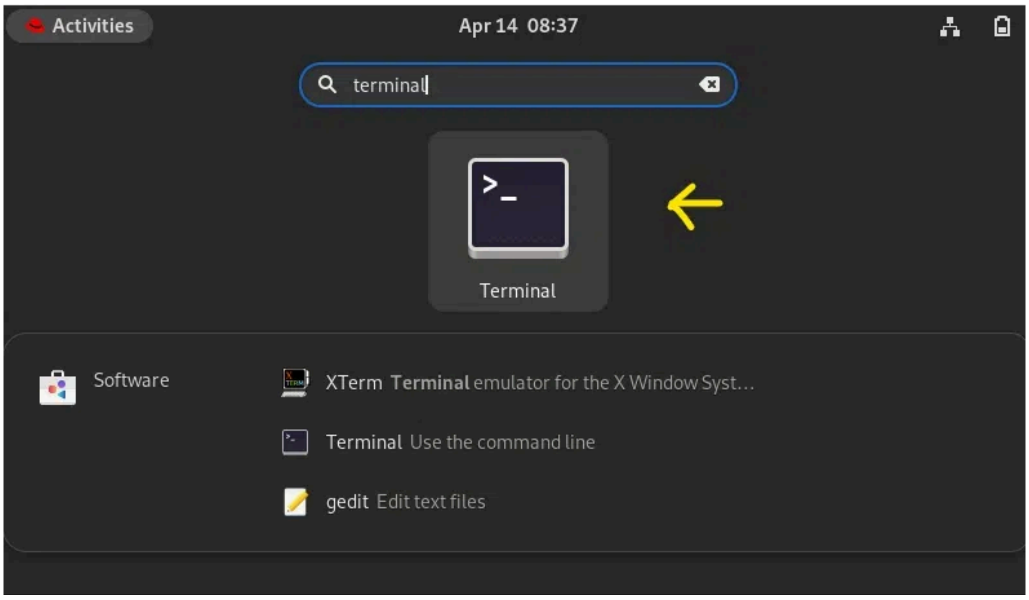
- BASH (Bourne Again SHell) – It is the most widely used shell in Linux systems. It is used as default login shell in Linux systems and in macOS. It can also be installed on Windows OS.
- CSH (C SHell) – The C shell's syntax and its usage are very similar to the C programming language.
- KSH (Korn SHell) – The Korn Shell was also the base for the POSIX Shell standard specifications etc.

Each shell does the same job but understands different commands and provides different built-in functions.

What is a terminal?

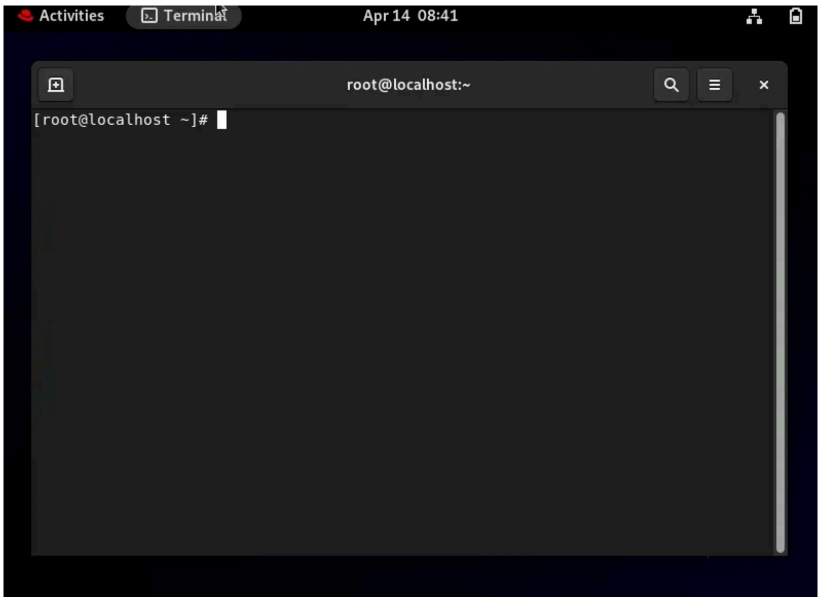
A program which is responsible for providing an interface to a user so that he/she can access the shell. It basically allows users to enter commands and see the output of those commands in a text-based interface. Large scripts that are written to automate and perform complex tasks are executed in the terminal.

To access the terminal, simply search in search box "terminal" and double-click it.


Activities
Apr 14 08:37
terminal
Terminal
Software
XTerm Terminal emulator for the X Window Syst...
Terminal Use the command line
gedit Edit text files

open terminal

Here you can see how the terminal looks of Red Hat Linux.



```
[root@localhost ~]#
```

terminal

Shell Scripting

Usually, shells are interactive, which means they accept commands as input from users and execute them. However, sometimes we want to execute a bunch of commands routinely, so we have to type in all commands each time in the terminal.

As a shell can also take commands as input from file, we can write

Shell scripts are similar to the batch file in MS-DOS. Each shell script is saved with **`.sh`** file extension e.g., **myscript.sh.**

A shell script has syntax just like any other programming language. If you have any prior experience with any programming language like Python, C/C++ etc. It would be very easy to get started with it.

A shell script comprises the following elements –

- Shell Keywords – if, else, break etc.
- Shell commands – cd, ls, echo, pwd, touch etc.
- Functions
- Control flow – if..then..else, case and shell loops etc.

Why do we need shell scripts?

There are many reasons to write shell scripts:

- To avoid repetitive work and automation
- System admins use shell scripting for routine backups.
- System monitoring
- Adding new functionality to the shell etc.

Some Advantages of shell scripts

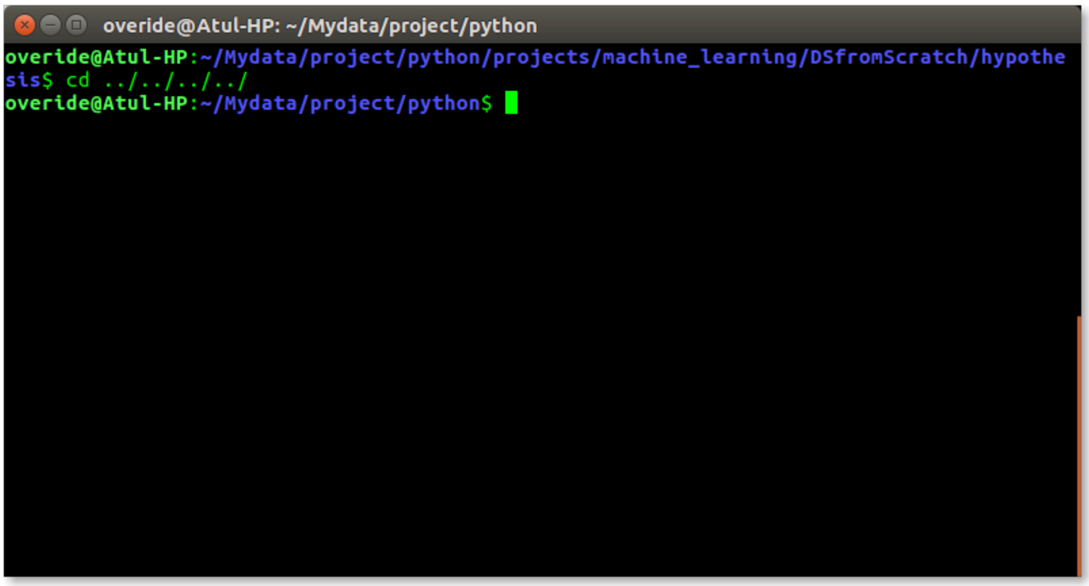
- The command and syntax are exactly the same as those directly entered in the command line, so programmers do not need to switch to entirely different syntax
- Writing shell scripts are much quicker
- Quick start
- Interactive debugging etc.

Some Disadvantages of shell scripts

- Prone to costly errors, a single mistake can change the command which might be harmful.

- Provide minimal data structure unlike other scripting languages. etc.

Simple demo of shell scripting using Bash Shell
If you work on a terminal, something you traverse deep down in directories. Then for coming few directories up in path we have to execute a command like this as shown below to get to the "python" directory:



```
overide@Atul-HP:~/Mydata/project/python/projects/machine_learning/DSfromScratch/hypothe
sis$ cd ../../../../
overide@Atul-HP:~/Mydata/project/python$
```

get to the "python" directory:

It is quite frustrating, so why not we can have a utility where we just have to type the name of directory and we can directly jump to that without executing the "**cd ../**" command again and again. Save the script as "**jump.sh**"

```
# !/bin/bash

# A simple bash script to move up to desired directory level
directly

function jump()
{
    # original value of Internal Field Separator
    OLDIFS=$IFS

    # setting field separator to "/"
    IFS=/

    # converting working path into array of directories in path
```

```
        IFS=$OLDIFS

        local pos=-1

        # ${path_arr[@]} gives all the values in path_arr
        for dir in "${path_arr[@]}"
        do
            # find the number of directories to move up to
            # reach at target directory
            pos=$[$pos+1]
            if [ "$1" = "$dir" ];then

                # length of the path_arr
                dir_in_path=${#path_arr[@]}

                #current working directory
                cwd=$PWD
                limit=$[$dir_in_path-$pos-1]
                for ((i=0; i<limit; i++))
                do
                    cwd=$cwd/..
                done
                cd $cwd
                break
            fi
        done
    }
```

For now, we cannot execute our shell script because it does not have permissions. We have to make it executable by typing following command –

> *$ chmod +x path/to/our/file/jump.sh*

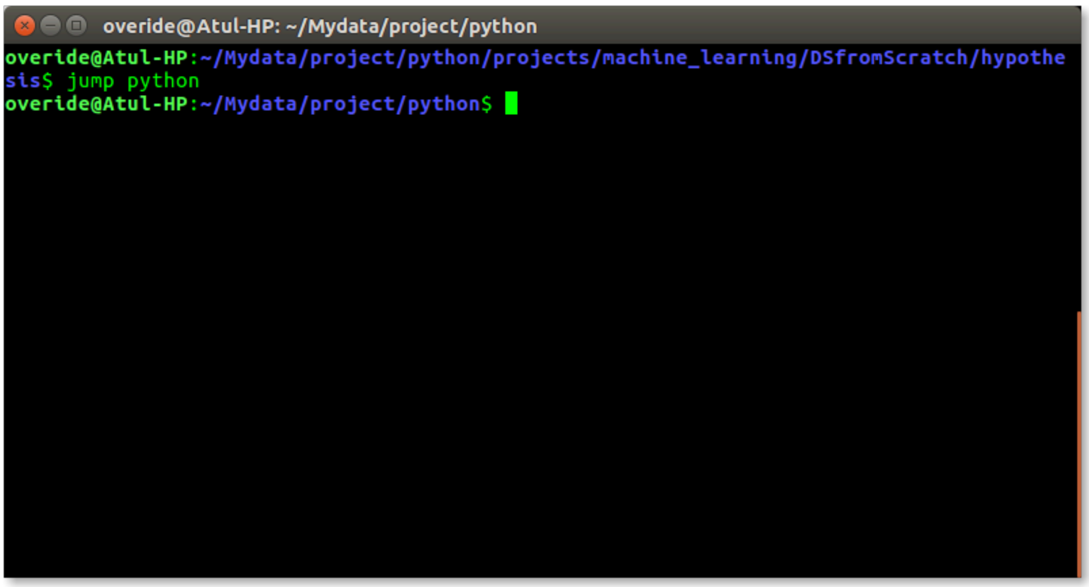
Now to make this available on every terminal session, we have to put this in "**.bashrc**" file.

"**.bashrc**" is a shell script that Bash shell runs whenever it is started interactively. The purpose of a .bashrc file is to provide a place where you can set up variables, functions, and aliases, define our prompt, and define other settings that we want to use whenever we open a new terminal window.

Now open your terminal and try out new "jump" functionality by typing following command-

$ jump dir_name

just like the below screenshot:



```
overide@Atul-HP: ~/Mydata/project/python
overide@Atul-HP:~/Mydata/project/python/projects/machine_learning/DSfromScratch/hypothe
sis$ jump python
overide@Atul-HP:~/Mydata/project/python$
```

jump dir_name

Conclusion

In this article, we learned about the essential parts of Linux systems: the kernel, which controls everything, the shell, which lets us interact with the operating system, and the terminal, our interface to give commands. We explored command line and graphical shells, like BASH, and understood that the terminal is where we type in our commands. The article introduced shell scripting, a way to automate tasks using simple scripts, and discussed the advantages and disadvantages of using them. A practical example demonstrated creating a script for quick directory navigation. Finally, we saw how to make the script work and become accessible in every terminal session. This article is a

Comment More info

Advertise with us

Next Article

How to Create a Shell Script in linux

Similar Reads

Shell Scripting - Difference between Korn Shell and Bash shell

Korn Shell: Korn Shell or KSH was developed by a person named David Korn, which attempts to integrate the features of other shells like C shell...

3 min read

Bash Scripting - Introduction to Bash and Bash Scripting

Bash is a command-line interpreter or Unix Shell and it is widely used in GNU/Linux Operating System. It is written by Brian Jhan Fox. It is used a...

10 min read

Shell Scripting - Interactive and Non-Interactive Shell

A shell gives us an interface to the Unix system. While using an operating system, we indirectly interact with the shell. On Linux distribution...

3 min read

Shell Scripting - Restricted Shell

Shell is one of the most important and powerful tools available in GNU/Linux-based systems. One can control the entire system if used...

A shell gives us an interface to the Unix system. While using an operating system, we indirectly interact with the shell. On Linux distribution...

3 min read

Shell Scripting - Shell Signals Values

Prerequisites: Processes, Bash Scripting, Shell Function Library Signals help the operating system to communicate with processes and vice-vers...

6 min read

Shell Scripting - Shell Variables

A shell variable is a character string in a shell that stores some value. It could be an integer, filename, string, or some shell command itself....

6 min read

Bash Scripting - How to Run Bash Scripting in Terminal

In this article, we are going to see how to run bash script in terminal. For example, we create a bash file set of instructions or commands (known ...

2 min read

Bash Scripting - Working of Bash Scripting

Bash Scripting is a crucial component of Linux's process automation. You can write a series of commands in a file and then execute them using...

6 min read

Shell Script to Show the Difference Between echo “$SHELL” and ech...

In shell scripting and Linux, the echo command is used to display text on the terminal or console. When used with the $SHELL variable, which...

4 min read

Django Tutorial
Python Projects
Python Tkinter
Web Scraping
OpenCV Tutorial
Python Interview Question

Operating Systems
Computer Network
Database Management System
Software Engineering
Digital Logic Design
Engineering Maths

DevOps

Git
AWS
Docker
Kubernetes
Azure
GCP
DevOps Roadmap

System Design

High Level Design
Low Level Design
UML Diagrams
Interview Guide
Design Patterns
OOAD
System Design Bootcamp
Interview Questions

School Subjects

Mathematics
Physics
Chemistry
Biology
Social Science
English Grammar

Commerce

Accountancy
Business Studies
Economics
Management
HR Management
Finance
Income Tax

Databases

SQL
MYSQL
PostgreSQL
PL/SQL
MongoDB

Preparation Corner

Company-Wise Recruitment Process
Resume Templates
Aptitude Preparation
Puzzles
Company-Wise Preparation
Companies
Colleges

Competitive Exams

JEE Advanced
UGC NET
UPSC
SSC CGL
SBI PO
SBI Clerk
IBPS PO
IBPS Clerk

More Tutorials

Software Development
Software Testing
Product Management
Project Management
Linux
Excel
All Cheat Sheets
Recent Articles

Code Converters

Currency Converter

Random Number Generator

Random Password Generator

Share your Experiences

Internships

DSA/Placements

DSA - Self Paced Course

DSA in JavaScript - Self Paced Course

DSA in Python - Self Paced

C Programming Course Online - Learn C with Data Structures

Complete Interview Preparation

Master Competitive Programming

Core CS Subject for Interview Preparation

Mastering System Design: LLD to HLD

Tech Interview 101 - From DSA to System Design [LIVE]

DSA to Development [HYBRID]

Placement Preparation Crash Course [LIVE]

Development/Testing

JavaScript Full Course

React JS Course

React Native Course

Django Web Development Course

Complete Bootstrap Course

Full Stack Development - [LIVE]

JAVA Backend Development - [LIVE]

Complete Software Testing Course [LIVE]

Android Mastery with Kotlin [LIVE]

Machine Learning/Data Science

Complete Machine Learning & Data Science Program - [LIVE]

Data Analytics Training using Excel, SQL, Python & PowerBI - [LIVE]

Data Science Training Program - [LIVE]

Mastering Generative AI and ChatGPT

Data Science Course with IBM Certification

Programming Languages

C Programming with Data Structures

C++ Programming Course

Java Programming Course

Python Full Course

Clouds/Devops

DevOps Engineering

AWS Solutions Architect Certification

Salesforce Certified Administrator Course

GATE

GATE CS & IT Test Series - 2025

GATE DA Test Series 2025

GATE CS & IT Course - 2025

GATE DA Course 2025