MASTER TERMUX IN 7 DAYS

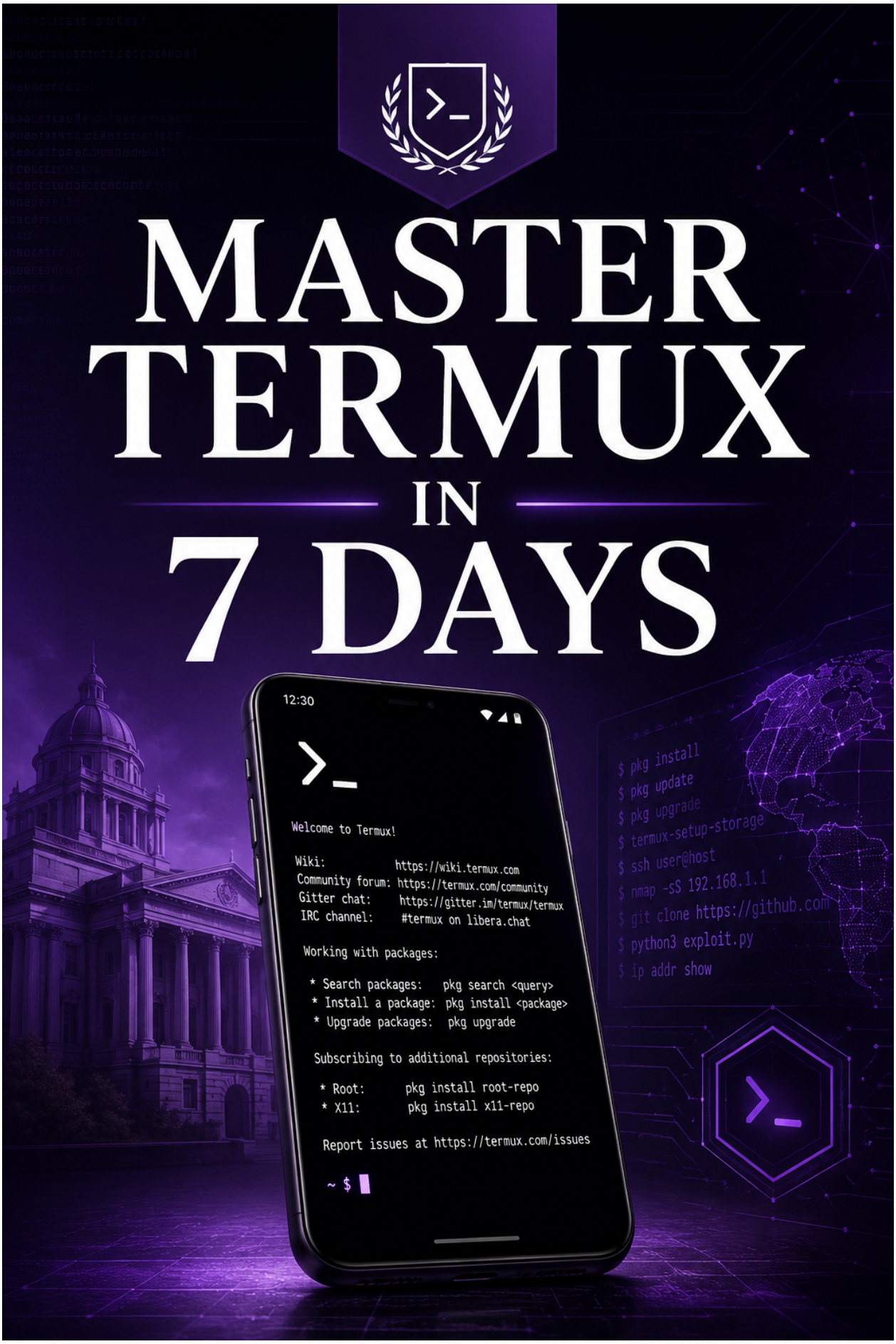

Table of Contents

This table shows where each lesson starts. The cover is not counted as a page. If you read on a phone, use the PDF search and type the lesson title.

Introduction - how this book teaches

This book is written for someone who may have never used a terminal before. It does not assume that the reader already knows where to tap, what an APK is, what a folder path means, or why an update command matters.

Each lesson is built like a practical classroom worksheet: you first learn the idea, then you perform tiny actions, then you check the result. The goal is not to memorize commands. The goal is to understand what you are doing before you run it.

Reading method

Read one small block, do it immediately on your phone, and write the result in your notes. If your screen looks different, stop and compare the exact words on your screen with the "what you should see" line.

- Never paste a command you do not understand.
- Do not install APK files from random sources.
- Keep screenshots or notes of errors; terminal errors are clues, not personal failures.
- Use the official links printed in the book, not copied social-media commands.

Part 1 - Lessons, explanations, and guided steps

Part 1 explains the environment and shows exactly what to do on the phone. Do not read it like an article. Read it with Termux open and a notes app next to you.

Part 1 goal

Understand settings, APK installation, paths, commands, packages, GitHub repos, Python scripts, local links, and terminal errors before the exercise workbook.

Deeper understanding - what Termux actually is

Termux is a Linux-like environment inside Android. It is not root, it does not replace Android, and it cannot automatically see every file on the phone. It gives you a separate workspace where you can install packages, download repositories, write scripts, and run local tools.

- Termux home is usually /data/data/com.termux/files/home.
- Android Downloads is usually /storage/emulated/0/Download.
- ~ means Termux home, not the whole phone.
- The prompt means the terminal is waiting for a command.
- Output is the command answer, and you should read it before continuing.

Thinking model

Where you are: Before every command.
What you do: Ask: where am I, which file do I want, which command am I running, what will change, what result do I expect?
What you should see: You can answer these before pressing Enter.
Why this matters: This reduces blind mistakes.
If it fails: If you cannot answer, run pwd, ls, and read README first.

Limits of Termux

Where you are: Before expecting a project to work.
What you do: Remember that permissions, packages, correct folder, and correct command matter.
What you should see: You do not treat every error as disaster.
Why this matters: Most errors are wrong path, missing dependency, or permission.
If it fails: If stuck, save the exact error and do the smallest check first.

How to read a command before running it

Every command has parts. Do not treat it like a magic phrase. Split it into program, action, options, files, paths, and URLs.

```
git clone https://github.com/dedsec1121fk/DedSec
```

Line-by-line explanation:

- git is the program.
- clone is the action: download a repo copy.
- The URL is the source.
- The result is a new DedSec folder.

```
cd DedSec && bash Setup.sh
```

Line-by-line explanation:

- cd DedSec changes folder.
- && means continue only if the previous part worked.
- bash Setup.sh runs the setup file.
- If cd fails, setup does not run.

Rule before Enter

If you see rm, unknown URL, curl pipe bash, sudo, or a path you do not understand, stop and explain the command line by line first.

More detail on Android APK and permissions

APK installation has stages: official source, download warning, unknown-app permission, installer, Play Protect, install result. The reader should know what each screen means.

Download warning

Where you are: Browser popup.
What you do: If you are on an official link, tap Download/OK. If you are on an unknown APK site, cancel.
What you should see: The file downloads and appears in notification or download list.
Why this matters: The warning exists because APKs can install apps.
If it fails: If you are not sure about the source, do not continue.

Allow from this source

Where you are: Android Settings after the prompt.
What you do: Enable Allow from this source only for the browser or My Files app that opened the APK. Then press Back.
What you should see: You return to the installer with Install button.
Why this matters: Android grants install permission per app.
If it fails: After installation you can disable it again.

App not installed

Where you are: Android installer.
What you do: Do not keep pressing install. Write the message and think source conflict.
What you should see: You understand whether old Termux from another source exists.
Why this matters: Termux add-ons should come from the same source.
If it fails: Uninstall old Termux/add-ons and reinstall all from F-Droid or all from official GitHub release.

Samsung example path

Settings -> Apps -> Special access -> Install unknown apps -> choose browser or My Files -> Allow from this source.

More useful everyday Termux habits

These habits make the reader stable: clean folders, logs, small commands, README first, backup before editing.

```
mkdir -p ~/Termux-Projects ~/Termux-Course/notes ~/Termux-Course/backups
printf "Termux course log\n" > ~/Termux-Course/notes/log.txt
ls ~/Termux-Course
```

Line-by-line explanation:

- Creates organized folders.
- Creates a starting log.
- Checks the structure.
- Do not work directly inside Downloads unless necessary.
- Before running a script, read README.
- Before deleting, run ls and check what exists.
- If a command fails, do not run ten random fixes. Read the first error.
- Keep screenshots or copy the last error lines.

Before starting - phone, internet, safety, and notes

This first setup matters because most beginner problems are not Termux problems. They are phone settings, storage permission, missing internet, wrong APK source, or a copied command from the wrong folder.

What to prepare before touching Termux

Charge the phone above 40%, connect to stable Wi-Fi, open a notes app, and create a note named "Termux course errors". Every time something fails, copy the exact error into that note.

- Use Android Settings to check free storage. Keep at least 2 GB free for the course; more is better if you clone large projects.
- Use a modern browser such as Chrome, Samsung Internet, Firefox, or Brave. The browser matters because Android gives APK install permission per app.
- Keep Google Play Protect enabled. It scans apps and may warn you about APKs installed outside the Play Store.
- Do not mix Termux from Google Play with add-ons from F-Droid or GitHub. Mixed sources often cause signature conflicts.

Step 1 - check storage

Where you are: Android home screen -> Settings.
What you do: Open Settings, search for Storage, and check available space.
What you should see: A storage screen showing used and free space.
Why this matters: Downloads, repositories, Python modules, and exported files need space. Low storage creates strange failures.
If it fails: Delete unnecessary files or move photos/videos before installing.

Step 2 - prepare notes

Where you are: Any notes app on the phone.
What you do: Create a note named Termux Course Log. Add today date and phone model.
What you should see: A blank note ready for commands and errors.
Why this matters: You need a place to record what changed. This makes troubleshooting possible.
If it fails: If you do not use notes, take screenshots of errors.

APK download and install - exact phone steps

An APK is an Android application package. Installing an APK outside the Play Store is called sideloading. It is normal for open-source tools, but it must be done carefully.

Official links

https://termux.dev/en/

https://f-droid.org/en/packages/com.termux/

https://github.com/termux/termux-app/releases

https://support.google.com/googleplay/answer/2812853

https://www.samsung.com/ae/support/mobile-devices/how-to-enable-permission-to-install-apps-from-unknown-source-on-my-samsung-phone/

Recommended source order

Use F-Droid or the official Termux GitHub releases. Do not use old Play Store Termux builds for this course. Install Termux, Termux:API, and Termux:Styling from the same source family when possible.

Step 1 - open official source

Where you are: Android browser.
What you do: Go to https://f-droid.org/en/packages/com.termux/ or https://github.com/termux/termux-app/releases.
What you should see: A page that clearly says Termux and belongs to F-Droid or github.com/termux.
Why this matters: You are verifying the source before installing code on your phone.
If it fails: If the page is an ad mirror or random APK site, close it.

Step 2 - download the APK

Where you are: F-Droid Termux page or GitHub Releases page.
What you do: On F-Droid, scroll to a version and tap Download APK. On GitHub Releases, open Assets and tap the APK file.
What you should see: The browser shows a download warning such as "This type of file can harm your device" or a download progress message.
Why this matters: Android warns because APK files can install apps. The warning is expected even for legitimate APKs.
If it fails: If there are many APK files and you are unsure which one to choose, use F-Droid Download APK first.

Step 3 - open the downloaded APK

Where you are: Browser download bar or Android Downloads/My Files.
What you do: Tap the downloaded .apk file.
What you should see: Android opens an install screen or asks permission to install unknown apps.
Why this matters: The phone is asking whether this browser/file manager is allowed to install apps.
If it fails: If nothing happens, open My Files -> Downloads and tap the APK there.

Step 4 - allow unknown apps for that source only

Where you are: Android settings screen opened by the install prompt.
What you do: Turn on Allow from this source for the browser or file manager you used. Then press Back.
What you should see: You return to the APK install screen with an Install button.
Why this matters: Android grants install permission per app, not globally on modern Android versions.
If it fails: After installing, you can return here and turn the permission off again.

Step 5 - install and verify

Where you are: APK install screen.
What you do: Tap Install. Wait. Tap Open only when it says App installed.
What you should see: Termux opens and shows a black terminal screen with text and a prompt.
Why this matters: The app is installed. The first prompt proves the app can start.
If it fails: If it says App not installed, uninstall old Termux/add-ons from other sources and install all from one source.

Security rule

Allowing unknown apps is not a habit; it is a temporary permission for a trusted source. After installing Termux, go back to Settings -> Apps -> Special app access -> Install unknown apps and disable the permission if you do not need it.

First opening of Termux - what the screen means

The first Termux screen can look empty or confusing. It is actually waiting for text commands. The important part is the prompt: the prompt marks the place where you type.

Step 1 - identify the prompt

Where you are: Inside Termux after opening it.
What you do: Look at the last line. It may end with $ or show a path like ~ $.
What you should see: A blinking cursor after the prompt.
Why this matters: The cursor means Termux is ready. Commands are typed there.
If it fails: If there is installation text still moving, wait until it stops.

Step 2 - test simple input

Where you are: Termux prompt.
What you do: Type echo hello and press Enter.
What you should see: The next line prints hello, then a new prompt appears.
Why this matters: This confirms keyboard input and Enter work.
If it fails: If your keyboard covers the prompt, rotate the phone or use a terminal-friendly keyboard.

Step 3 - understand output

Where you are: Termux prompt.
What you do: Run pwd.
What you should see: A path such as /data/data/com.termux/files/home.
Why this matters: pwd means print working directory. It tells you where commands will operate.
If it fails: If you expected Downloads, you are not there yet. You must cd to that path.

```
echo hello
```

Line-by-line explanation:

- echo prints text to the screen.
- hello is the text you asked it to print.
- This command is safe because it does not change files.

```
pwd
```

Line-by-line explanation:

- pwd means print working directory.
- The result is your current folder.
- When a command cannot find a file, pwd is one of the first checks.

Storage access - Downloads, shared storage, and paths

Android apps do not automatically see every file on the phone. Termux has its own private home folder. To work with Downloads, you must request storage permission.

Official links

https://wiki.termux.com/wiki/Termux-setup-storage

```
termux-setup-storage
```

Line-by-line explanation:

- This asks Android to give Termux access to shared storage.
- A permission popup should appear.
- After approval, Termux creates storage shortcuts in ~/storage.

Step 1 - run storage setup

Where you are: Termux prompt.
What you do: Type termux-setup-storage and press Enter.
What you should see: Android asks to allow Termux to access photos/media/files or storage.
Why this matters: Without this permission, scripts cannot easily read Downloads or save exports there.
If it fails: If no popup appears, open Android Settings -> Apps -> Termux -> Permissions and allow storage/files.

Step 2 - check shortcuts

Where you are: Termux prompt.
What you do: Run ls ~/storage.
What you should see: You should see folders like downloads, shared, dcim, pictures.
Why this matters: These shortcuts are easier than typing long Android paths every time.
If it fails: If ~/storage does not exist, rerun termux-setup-storage and fully close/reopen Termux.

Step 3 - understand real Downloads path

Where you are: Termux prompt.
What you do: Run ls /storage/emulated/0/Download.
What you should see: You should see files from your phone Downloads folder.
Why this matters: Some scripts use the real Android path. You must recognize it.
If it fails: If permission denied appears, storage permission is not approved.

```
ls ~/storage
ls /storage/emulated/0/Download
```

Line-by-line explanation:

- ls lists files and folders.
- ~/storage is a Termux shortcut folder.
- /storage/emulated/0/Download is the common Android internal Downloads path.

Package management - update, upgrade, install, search

Termux uses packages. A package is a prepared tool that Termux knows how to download and install. The pkg command is a Termux-friendly wrapper around the package manager.

Official links

https://wiki.termux.com/wiki/Package_Management

https://github.com/termux/termux-packages/wiki/package-management

```
pkg update
```

Line-by-line explanation:

- Downloads the newest package lists.
- It does not install everything by itself.
- It tells Termux what versions are available.

```
pkg upgrade
```

Line-by-line explanation:

- Installs available upgrades for packages you already have.
- You may be asked to confirm with Y or N.
- Read the prompt before pressing Enter.

```
pkg install git nano python
```

Line-by-line explanation:

- Installs Git, Nano, and Python.
- git downloads repositories.
- nano edits text files.
- python runs Python scripts.

Step 1 - update lists

Where you are: Termux prompt, internet connected.
What you do: Run pkg update.
What you should see: Lines showing repositories and package lists.
Why this matters: You are refreshing the catalog before installing tools.
If it fails: If a repo error appears, change mirror or check internet.

Step 2 - upgrade safely

Where you are: Termux prompt.
What you do: Run pkg upgrade and read confirmation prompts.
What you should see: A list of packages to upgrade or a message that everything is current.
Why this matters: Upgrades reduce weird errors caused by old tools.
If it fails: If it asks about config files, keep the default unless you know why.

Step 3 - install core tools

Where you are: Termux prompt.
What you do: Run pkg install git nano python.
What you should see: Termux downloads and installs the tools.
Why this matters: These are the core tools for the course projects.
If it fails: If a package is not found, run pkg search keyword first.

Navigation - folders, files, spaces, and safe deletion

Most beginner terminal mistakes are location mistakes. The terminal only acts in the current folder unless you give it a full path.

```
pwd
ls
cd folder-name
cd ..
```

Line-by-line explanation:

- pwd shows where you are.
- ls shows what is inside the current folder.
- cd folder-name enters a folder.
- cd .. goes one folder back.

```
mkdir -p ~/Termux-Projects
cd ~/Termux-Projects
pwd
```

Line-by-line explanation:

- mkdir creates a folder.
- -p means do not complain if parent folders already exist.
- cd enters the practice folder.
- pwd confirms where you landed.

Spaces in file names

If a file or folder has spaces, wrap it in quotes. Example: python "Corrupted Files.py". Without quotes, the shell thinks each word is a separate argument.

```
ls -la
cp file.txt backup-file.txt
rm -i old-file.txt
```

Line-by-line explanation:

- ls -la shows hidden files and details.
- cp copies a file; it is safer than editing the only copy.
- rm -i asks before deleting; use it while learning.

Never use destructive commands blindly

Do not teach yourself with rm -rf. If you do not understand the target path, do not run the command. Deletion in Termux is not a game.

Text editing with nano - open, write, save, exit

Nano is a beginner-friendly terminal editor. It is not fancy, but it is predictable. The bottom of the screen shows shortcuts. The ^ symbol means Ctrl.

```
nano notes.txt
```

Line-by-line explanation:

- Opens notes.txt.
- If the file does not exist, nano creates it when you save.
- You type normal text in the editor area.

Step 1 - create a note

Where you are: Inside your course folder.
What you do: Run nano notes.txt. Type one sentence about what you learned.
What you should see: Nano opens, and you can write text.
Why this matters: Editing files is necessary for scripts, notes, and configs.
If it fails: If nano is missing, run pkg install nano.

Step 2 - save

Where you are: Inside nano.
What you do: Press Ctrl+O, then Enter.
What you should see: Nano writes the file and stays open.
Why this matters: Ctrl+O means Write Out, which means save.
If it fails: If the keyboard lacks Ctrl, enable extra keys in Termux or use a keyboard app.

Step 3 - exit and check

Where you are: Inside nano.
What you do: Press Ctrl+X. Then run cat notes.txt.
What you should see: Your sentence prints in the terminal.
Why this matters: You prove the file was saved before moving on.
If it fails: If cat says file missing, you saved in another folder or did not confirm Enter.

```
cat notes.txt
```

Line-by-line explanation:

- cat prints the file content.
- It is useful for small files.
- For long files, use less file.txt if installed.

GitHub from browser and from Termux

GitHub is a place where projects are stored. You can use it in two ways: download a ZIP from the browser, or clone the repository in Termux using git.

Official links

https://docs.github.com/en/repositories/creating-and-managing-repositories/cloning-a-repository

https://docs.github.com/en/authentication/connecting-to-github-with-ssh

Browser method - Download ZIP

Where you are: Android browser on a GitHub repository page.
What you do: Tap Code, then Download ZIP. Wait for the file to appear in Downloads.
What you should see: A .zip file in Downloads.
Why this matters: This is easiest for beginners who only want files.
If it fails: If the Code button is hard to see on mobile, rotate the phone or use desktop site.

Termux method - clone

Where you are: Termux inside ~/Termux-Projects.
What you do: Run git clone followed by the repository URL.
What you should see: A new folder appears with the repository name.
Why this matters: Clone keeps Git history and makes updates easier with git pull.
If it fails: If git is missing, run pkg install git.

Update method - pull

Where you are: Inside a cloned repo folder.
What you do: Run git pull.
What you should see: Git says Already up to date or downloads changes.
Why this matters: This updates the project without deleting your whole folder.
If it fails: If you edited files, Git may ask you to commit, stash, or reset. Read the message carefully.

```
mkdir -p ~/Termux-Projects
cd ~/Termux-Projects
git clone https://github.com/dedsec1121fk/DedSec.git
ls
```

Line-by-line explanation:

- Creates a clean practice folder.
- Moves into it.
- Downloads the DedSec repository.
- Lists the folder to prove it exists.

Running Python scripts safely

A Python script is a text file that Python executes line by line. Before running a script from any project, read the README, list the files, and identify the main file.

Official links

https://docs.python.org/3/

https://docs.python.org/3/library/venv.html

https://pip.pypa.io/en/stable/

```
python --version
python script.py
python "File With Spaces.py"
```

Line-by-line explanation:

- python --version confirms Python is installed.
- python script.py runs a script in the current folder.
- Quotes are needed when the file name has spaces.

Step 1 - inspect first

Where you are: Inside a project folder.
What you do: Run ls, then read README.md if it exists with less README.md or open it in browser.
What you should see: You see the main files and instructions.
Why this matters: This prevents running the wrong file.
If it fails: If README is too long, search within it for Install, Run, Usage.

Step 2 - check Python

Where you are: Termux prompt.
What you do: Run python --version.
What you should see: A version number such as Python 3.x.x.
Why this matters: You need to know Python exists before troubleshooting scripts.
If it fails: If missing, run pkg install python.

Step 3 - run carefully

Where you are: Project folder.
What you do: Run the exact command from the README. Do not invent file names.
What you should see: The script opens a menu, prints help, or gives a specific error.
Why this matters: The first run teaches what the script expects.
If it fails: If an error appears, copy it exactly. Do not summarize it.

Requirements rule

If a project has requirements.txt, read it before pip installing random modules. In Termux, some Python packages need system packages first. Random pip commands can waste time.

Project 1 - DedSec Project: what it does and how to study it

DedSec Project is an educational Android + Termux toolkit. It contains scripts, utilities, local web interfaces, games, settings/configuration tools, and bilingual English/Greek documentation.

Official links

https://github.com/dedsec1121fk/DedSec

https://ded-sec.space/

What the user learns from this project

Repository structure, scripts organization, setup commands, local web UIs, README reading, safe project inspection, updates with git pull, and how a large Termux toolkit is organized.

```
mkdir -p ~/Termux-Projects
cd ~/Termux-Projects
git clone https://github.com/dedsec1121fk/DedSec.git
cd DedSec
ls
```

Line-by-line explanation:

- Creates your learning projects folder.
- Downloads the repository.
- Enters the DedSec folder.
- Lists files so you can see what exists before running anything.

Step 1 - read the README

Where you are: Inside ~/Termux-Projects/DedSec.
What you do: Open README.md in GitHub browser or run less README.md if available.
What you should see: You see installation sections, categories, and disclaimers.
Why this matters: The README is the map of the project. Running random files first is a beginner mistake.
If it fails: If less is missing, use cat README.md for short viewing or open GitHub in browser.

Step 2 - identify categories

Where you are: Repo folder or GitHub page.
What you do: Look for folders such as Scripts, Games, Developer Base, or documented categories.
What you should see: You understand the project is a collection, not one single command.
Why this matters: Large repositories must be navigated, not guessed.
If it fails: If file names differ, trust the current repo listing, not memory.

Step 3 - run only documented entry points

Where you are: Termux prompt.
What you do: Use only commands described in README or inside clear script comments.
What you should see: The tool opens a menu or prints instructions.
Why this matters: A documented entry point is easier to troubleshoot.
If it fails: If a script asks for permissions or network access, pause and read why.

Project 2 - Offline Survival Project: offline knowledge base

Offline Survival Project is a bilingual offline-first knowledge project for Android + Termux. It combines a JSON knowledge database, a terminal app, a local browser interface, TXT export tools, and update logs.

Official links

https://github.com/dedsec1121fk/Offline-Survival-Project

What the user learns from this project

How data folders work, how JSON databases are used by Python, how offline apps search knowledge, how exports are saved, and why storage permission matters.

```
mkdir -p ~/Termux-Projects
cd ~/Termux-Projects
git clone https://github.com/dedsec1121fk/Offline-Survival-Project.git
cd Offline-Survival-Project
ls
```

Line-by-line explanation:

- Creates or reuses your project area.
- Clones the offline survival project.
- Enters its folder.
- Lists the database, script, and support files.

```
python "Offline Survival.py"
```

Line-by-line explanation:

- Runs the main script.
- Quotes protect the space in the file name.
- The expected result is a menu or interface described by the project.

Step 1 - confirm database exists

Where you are: Inside Offline-Survival-Project.
What you do: Run ls and look for Offline Survival Database or similar database folder.
What you should see: A folder containing JSON files or database shards.
Why this matters: The script needs the database; without it the reader has nothing to search.
If it fails: If missing, the download is incomplete; re-clone or re-extract the ZIP.

Step 2 - understand offline use

Where you are: Project folder.
What you do: Disconnect internet after setup only as a test, then open the script again.
What you should see: Search/browse still works if the database is local.
Why this matters: This proves the project is useful without internet after download.
If it fails: If it tries to download data, read the README to see what part requires internet.

Step 3 - export safely

Where you are: Inside the app or terminal.
What you do: Use export features only after storage permission is approved.
What you should see: Exported TXT files appear in a documented folder or Downloads.
Why this matters: Exports help keep notes outside the app.
If it fails: If files do not appear, check /storage/emulated/0/Download and ~/storage/downloads.

Project 3 - Corrupted Files Project: offline incident archive

Corrupted Files Project is a bilingual offline incident archive focused on Greece and the USA. It organizes public-interest incidents, institutional-failure cases, disasters, corruption-related files, and historical dossiers into browsable folders, JSON records, indexes, and media references.

Official links

https://github.com/dedsec1121fk/Corrupted-Files-Project

What the user learns from this project

How structured archives work, how country/year/incident folders are organized, how indexes make browsing faster, and how a Python reader can navigate a dataset.

```
mkdir -p ~/Termux-Projects
cd ~/Termux-Projects
git clone https://github.com/dedsec1121fk/Corrupted-Files-Project.git
cd Corrupted-Files-Project
ls
```

Line-by-line explanation:

- Creates or reuses the projects folder.
- Downloads the archive.
- Enters the project.
- Shows the country folders, indexes, and main script.

```
python "Corrupted Files.py"
```

Line-by-line explanation:

- Runs the main reader script.
- Quotes are required because the file name contains a space.
- The expected result is a terminal-guided way to browse/search the archive.

Step 1 - browse manually first

Where you are: Inside the project folder or file manager.
What you do: Open Greece or USA, choose a year, then open one incident folder.
What you should see: You see text files, JSON records, metadata, and possibly media references.
Why this matters: Manual browsing teaches the structure before using a script.
If it fails: If folders are missing, the clone/download did not finish.

Step 2 - check indexes

Where you are: Root of project.
What you do: Look for files such as Master Incident Index, dates by country, statistics, or JSON/CSV indexes.
What you should see: Index files summarize the archive.
Why this matters: Indexes help you search without opening every folder.
If it fails: If index files are missing, use current README instructions for validation/rebuild if provided.

Step 3 - run reader

Where you are: Termux in project folder.
What you do: Run python "Corrupted Files.py".
What you should see: A menu or guided terminal output appears.
Why this matters: The reader is easier than manual browsing when the archive is large.
If it fails: If Python error appears, copy it exactly and check missing files or modules.

Verified project links and README guides

This section collects the three project links and their verified README links with installation/usage instructions in one place. First open the repository, then the README, then the install/use section, and only then run the Termux commands.

Official links

DedSec Project repository: https://github.com/dedsec1121fk/DedSec

DedSec Project README: https://github.com/dedsec1121fk/DedSec/blob/main/README.md

DedSec README file - scroll to "How To Install And Setup The DedSec Project": https://github.com/dedsec1121fk/DedSec/blob/main/README.md

Offline Survival repository: https://github.com/dedsec1121fk/Offline-Survival-Project

Offline Survival README: https://github.com/dedsec1121fk/Offline-Survival-Project/blob/main/README.md

Offline Survival README file - scroll to "How To Install And Run Offline Survival": https://github.com/dedsec1121fk/Offline-Survival-Project/blob/main/README.md

Corrupted Files repository: https://github.com/dedsec1121fk/Corrupted-Files-Project

Corrupted Files README: https://github.com/dedsec1121fk/Corrupted-Files-Project/blob/main/README.md

Corrupted Files README file - scroll to "How To Download": https://github.com/dedsec1121fk/Corrupted-Files-Project/blob/main/README.md

Important

If the GitHub README changes later, trust the current README. The book teaches the safe method: official repo -> README -> install/use guide -> Termux command.

DedSec Project - README install guide with exact steps

The DedSec Project README describes a first-time full install. This section rewrites it as practical phone and Termux steps.

Official links

README file - installation section inside this README:
https://github.com/dedsec1121fk/DedSec/blob/main/README.md

Step 1 - F-Droid and Termux

Where you are: Android browser and F-Droid.
What you do: Download F-Droid, open it, search Termux, and install it. If you install Termux:API or Termux:Styling, use the same source family.
What you should see: Termux appears in the app drawer and opens without App not installed.
Why this matters: The README recommends F-Droid for compatibility. One source family reduces signature conflicts.
If it fails: If App not installed appears, uninstall old Termux/add-ons from other sources and reinstall all from one source.

Step 2 - open Termux before pasting

Where you are: Termux app.
What you do: Open Termux and wait until the first text stops moving.
What you should see: You see a prompt with a cursor, usually something like ~ $.
Why this matters: The commands must run inside Termux, not in the browser.
If it fails: If the keyboard is hidden, tap the black terminal area.

Step 3 - prepare packages

Where you are: Termux prompt.
What you do: Run the README command for update/upgrade/git/nano/storage.
What you should see: You see download/update messages and a storage permission popup.
Why this matters: This prepares Termux to download repos, edit files, and access shared storage.
If it fails: If no storage popup appears, open Settings -> Apps -> Termux -> Permissions and check Files/Storage.

Step 4 - clone and setup

Where you are: Termux prompt.
What you do: Clone the repo, enter the DedSec folder, and run bash Setup.sh.
What you should see: The setup/menu starts or prints instructions.
Why this matters: Setup.sh is the documented entry point from the README.
If it fails: If it says No such file, run pwd and ls. You are probably in the wrong folder.

```
pkg update -y
pkg upgrade -y
```

```
pkg install git nano -y
termux-setup-storage
```

Line-by-line explanation:

- Updates Termux packages.
- Installs git and nano.
- Requests storage access.

```
git clone https://github.com/dedsec1121fk/DedSec
cd DedSec
bash Setup.sh
```

Line-by-line explanation:

- Downloads the DedSec Project.
- Enters the project folder.
- Starts the setup mentioned in the README.

Offline Survival Project - README install/run guide with exact steps

The README says this is an offline-first knowledge base with a JSON database, terminal app, browser interface, TXT exports, and update logs. To work, the main Python file and database folder must exist locally.

Official links

README file - install/run section inside this README:
https://github.com/dedsec1121fk/Offline-Survival-Project/blob/main/README.md

Step 1 - download the project

Where you are: GitHub page or Termux.
What you do: Use Code -> Download ZIP and extract it, or use git clone from Termux.
What you should see: You have an Offline-Survival-Project or Offline-Survival-Project-main folder.
Why this matters: Without local files, there is no offline database for the app to read.
If it fails: If the ZIP will not open, download it again with stable internet.

Step 2 - storage for exports

Where you are: Termux prompt.
What you do: Run termux-setup-storage if you want exports/Downloads access.
What you should see: Android asks permission and ~/storage is created.
Why this matters: The README mentions storage access for exports and files in Downloads.
If it fails: If Permission denied appears, check Android permissions for Termux.

Step 3 - enter the correct folder

Where you are: Termux prompt.
What you do: Run cd with the real folder name, then ls.
What you should see: You see Offline Survival.py and the database folder.
Why this matters: Before running Python, you must be in the folder that contains the script.
If it fails: If ls does not show the script, you are in the wrong path.

Step 4 - run the main script

Where you are: Inside the project folder.
What you do: Run python "Offline Survival.py" or python3 "Offline Survival.py" if needed.
What you should see: A menu/app output appears.
Why this matters: Quotes are required because the filename contains a space.
If it fails: If ModuleNotFoundError appears, copy the exact module name and check README/requirements.

Step 5 - browser UI

Where you are: Inside the terminal menu.
What you do: If you choose the browser option, open http://127.0.0.1:8765/ in the browser.
What you should see: A local interface opens on the same phone.
Why this matters: 127.0.0.1 means local phone, not a public website.
If it fails: If it does not open, keep the Termux process running and check whether the port is 8765 or another printed port.

```
termux-setup-storage
cd ~/Offline-Survival-Project-main
python "Offline Survival.py"
```

Line-by-line explanation:

- Requests storage access.
- Enters the extracted project folder.
- Runs the main script with quotes.

```
cd ~/Offline-Survival-Project-main
python3 "Offline Survival.py"
```

Line-by-line explanation:

- Alternative only if python3 is needed.
- Launches the same script with a different Python command.

Corrupted Files Project - README download/use guide with exact steps

The README describes this project as a bilingual offline incident archive for Greece and the USA. You can use it with manual browsing or with the main Python script.

Official links

README file - download section inside this README: https://github.com/dedsec1121fk/Corrupted-Files-Project/blob/main/README.md

README: https://github.com/dedsec1121fk/Corrupted-Files-Project/blob/main/README.md

Step 1 - browser download

Where you are: GitHub repository page.
What you do: Tap Code -> Download ZIP and extract it anywhere you want.
What you should see: You see a Corrupted-Files-Project-main folder.
Why this matters: This is the simplest method if you do not want Git commands.
If it fails: If the ZIP is incomplete, download it again.

Step 2 - or clone with Termux

Where you are: Termux inside ~/Termux-Projects.
What you do: Run git clone with the official URL and enter the folder.
What you should see: You see Greece, USA, database/tools, or root index files.
Why this matters: Clone makes later updates easier.
If it fails: If git is missing, run pkg install git.

Step 3 - manual browsing

Where you are: File manager or Termux.
What you do: Open Greece or USA, choose a year folder, then an incident folder.
What you should see: You see paired EN/EL files, JSON records, metadata, and media references.
Why this matters: Manual browsing teaches the structure before using the reader.
If it fails: If folders are missing, extraction/clone did not finish.

Step 4 - run reader

Where you are: Termux inside the project folder.
What you do: Run python "Corrupted Files.py".
What you should see: A guided terminal reader/menu appears.
Why this matters: The script helps when the archive is large.
If it fails: If it says cannot open file, check pwd, ls, and quotes.

Step 5 - validation tools if present

Where you are: Termux inside the project folder.
What you do: If the README-listed files exist, run validate or rebuild indexes.
What you should see: You see validation/rebuild output.
Why this matters: Useful for checking structure and indexes.
If it fails: If the file does not exist, do not invent it. Check current README and ls.

```
git clone https://github.com/dedsec1121fk/Corrupted-Files-Project.git
cd Corrupted-Files-Project
python "Corrupted Files.py"
```

Line-by-line explanation:

- Downloads the project.
- Enters the folder.
- Runs the main reader with quotes.

```
python "Corrupted Files Tools/validate_rebuilt_structure.py"
python "Corrupted Files Tools/rebuild_indexes.py"
```

Line-by-line explanation:

- Run these only if the files exist.
- The first checks structure.
- The second rebuilds indexes.

Useful daily workflow - the same routine for every project

The safest way to learn Termux is to repeat the same inspection routine for every new folder. This routine prevents most beginner mistakes.

```
pwd
ls
ls -la
cat README.md
python --version
```

Line-by-line explanation:

- Confirm location.
- List visible files.
- List hidden files and details.
- Read the README if it is short enough.
- Confirm Python exists before running Python scripts.

Project inspection checklist

1. What is the project name? 2. What is the main file? 3. Does it need storage permission? 4. Does it need internet after setup? 5. Where does it save output? 6. What command starts it? 7. What does a successful start look like?

- Create one folder: ~/Termux-Projects. Keep projects there.
- Do not run scripts from Downloads forever; move or clone projects into a stable folder.
- Keep a notes.txt file inside each project with setup commands and errors.
- Before deleting a project, check pwd twice.

Terminal troubleshooting - what errors mean and what to do

Use this section like a dictionary. Find the exact phrase, read what it means, then do the first safe fix.

No such file or directory

Meaning: You are in the wrong folder, or the file name is typed differently.
First fix: Run pwd, then ls. If the file has spaces, use quotes: python "Offline Survival.py".

Permission denied

Meaning: The file exists but cannot execute, or Android storage permission is missing.
First fix: Use bash script.sh instead of ./script.sh, or run termux-setup-storage and allow permission.

command not found

Meaning: The package that provides the command is not installed, or the command name is wrong.
First fix: Run pkg search name, then pkg install package. For Python modules, use pip install module.

Unable to locate package

Meaning: The package name is wrong, repositories are not updated, or mirrors are broken.
First fix: Run pkg update. Then run pkg search keyword. Do not guess package names.

repository is under maintenance or down

Meaning: The mirror may be temporary broken.
First fix: Run termux-change-repo, choose Main repository, select another mirror, then pkg update.

Hash Sum mismatch

Meaning: Package lists are inconsistent or your connection interrupted.
First fix: Run pkg clean, pkg update, then retry on stable Wi-Fi.

pkg update stuck

Meaning: Network, mirror, VPN, or DNS problem.
First fix: Turn off VPN temporarily, switch Wi-Fi/mobile data, change repo mirror.

ModuleNotFoundError

Meaning: Python can run, but a library imported by the script is missing.
First fix: Read the exact missing module name. Install with pip install module-name or read the README requirements.

SyntaxError

Meaning: The Python file is broken, incomplete, copied badly, or edited incorrectly.
First fix: Do not guess. Re-download the file, compare line number, and check missing quotes/brackets.

ImportError from pip package

Meaning: Package installed, but wrong version or platform mismatch.
First fix: Update pip only if README says so. Try pip install --upgrade package. In Termux, prefer pkg packages when available.

git: not found

Meaning: Git is not installed.
First fix: Run pkg install git.

fatal: repository not found

Meaning: Wrong URL, private repo, or no access.
First fix: Open the repo URL in browser first. If private, login or check permissions.

Authentication failed

Meaning: GitHub password authentication is not used for Git operations.
First fix: Use HTTPS with token or SSH keys. Read GitHub SSH docs.

Connection timed out

Meaning: Internet, DNS, firewall, VPN or GitHub availability problem.
First fix: Open the link in browser. Retry later or switch network.

Address already in use

Meaning: A local server is already running on that port.
First fix: Stop the old process with Ctrl+C, or choose another port.

Cannot open file in Downloads

Meaning: Termux does not have storage access or path is wrong.
First fix: Run termux-setup-storage, allow permission, then use /storage/emulated/0/Download.

App not installed

Meaning: APK conflict, old Play Store build, wrong architecture, or signature mismatch.
First fix: Uninstall old Termux/add-ons from other source. Install all Termux apps from the same source.

Play Protect warning

Meaning: Android wants to scan the APK or warns about sideloading.
First fix: Use official sources only. Scan if asked. Do not disable Play Protect just to install random files.

nano cannot save

Meaning: Folder is read-only, wrong storage permission, or file opened without write permission.
First fix: Save inside your home folder first. Use cp to move later.

rm deleted wrong file

Meaning: The command did exactly what you typed.
First fix: Stop. Do not create new files there. Restore from backup if available. Use rm -i in learning phase.

APK and Android troubleshooting - install problems

APK problems usually come from source mixing, blocked unknown-app permission, old Play Store builds, Play Protect warnings, or incomplete downloads.

Problem - Install button is blocked

Where you are: Android APK install screen.
What you do: Wait a few seconds, scroll the permission dialog if needed, or close overlay apps.
What you should see: The Install button becomes tappable.
Why this matters: Android blocks accidental taps and overlay attacks.
If it fails: Disable screen filter/overlay apps temporarily if needed.

Problem - App not installed

Where you are: Android installer.
What you do: Uninstall old Termux and all Termux add-ons from other sources, then reinstall from one source.
What you should see: The new install completes.
Why this matters: Android signatures must match. Mixing sources causes signature mismatch.
If it fails: Back up files first if you already used the old app.

Problem - Cannot open APK

Where you are: Downloads/My Files.
What you do: Re-download the file from official source, make sure the filename ends in .apk, then tap it again.
What you should see: Android opens package installer.
Why this matters: A partial download is not a valid APK.
If it fails: Try another browser or the F-Droid page.

Problem - Play Protect warning

Where you are: Install flow.
What you do: Read the warning. If the APK came from official Termux/F-Droid/GitHub, scan if offered.
What you should see: Play Protect may allow install after scan.
Why this matters: Warnings are part of Android safety. They should not be ignored blindly.
If it fails: If the file came from a random site, delete it and use official links.

After installing

Go back to Settings -> Apps -> Special app access -> Install unknown apps -> your browser/file manager. Turn off Allow from this source if you do not need more APK installs.

Git and GitHub troubleshooting - clone, pull, access

Git errors are usually very literal. The message often tells you whether the problem is URL, network, authentication, or local changes.

```
git status
git remote -v
git pull
```

Line-by-line explanation:

- git status shows local changes and branch state.
- git remote -v shows where the repo connects.
- git pull downloads updates from the remote.
- If repository not found appears, open the repo URL in browser first.
- If authentication fails, remember GitHub does not use account passwords for Git operations.
- If local changes block pull, save your edits somewhere before resetting anything.
- If a repo is private, cloning needs a logged-in method with access.

HTTPS vs SSH

HTTPS is simpler for public clones. SSH is better when you push often, but it requires generating an SSH key and adding it to GitHub. Beginners can start with HTTPS for public projects.

Python troubleshooting - modules, paths, and script names

Python problems usually come from running the wrong file, being in the wrong folder, missing a module, or using a package that does not build easily on Android.

```
python --version
which python
pip --version
python -m pip --version
```

Line-by-line explanation:

- Confirms Python version.
- Shows the Python executable path.
- Checks pip command.
- Uses Python to call pip directly, which is often clearer.

Problem - ModuleNotFoundError

Where you are: After running a Python script.
What you do: Read the exact missing module name. Search the README for requirements.
What you should see: You know which dependency is missing.
Why this matters: Installing random packages can make the situation worse.
If it fails: Use pip install exact-name only if the README or error makes it clear.

Problem - file with spaces

Where you are: Termux prompt.
What you do: Use quotes around the filename: python "Corrupted Files.py".
What you should see: Python starts the file instead of saying cannot open file.
Why this matters: The shell splits words at spaces unless you quote them.
If it fails: You can also type the first letters and use Tab completion.

Problem - script cannot find database

Where you are: Inside project folder.
What you do: Run pwd and ls. Confirm database folders are beside the script.
What you should see: Script and database are in same project tree.
Why this matters: Many scripts use relative paths from the current folder.
If it fails: cd into the project folder before running the script.

Command library - what to type and when

The command library explains not just the command, but the situation where it is useful.

pwd

Use when you feel lost. It tells you the current folder.

ls

Use before opening/running files. It shows what exists here.

ls -la

Use when hidden files or permissions matter.

cd folder

Use to enter a folder. Replace folder with the exact folder name.

cd ..

Use to go one level back.

mkdir -p folder

Use to create folders safely, including parent folders.

cp source destination

Use to make a copy before editing.

mv old new

Use to rename or move files.

rm -i file

Use to delete with confirmation while learning.

cat file

Use to print short text files.

nano file

Use to edit a text file.

pkg search word

Use before installing when you do not know the exact package name.

pkg install package

Use to install Termux packages.

python file.py

Use to run Python file in current folder.

git clone URL

Use to download a repository into a folder.

git pull

Use inside a cloned repository to update it.

clear

Use to clear screen only; it does not delete files.

history

Use to see recently typed commands.

du -sh folder

Use to check folder size.

find . -name "*.py"

Use to find Python files under current folder.

More useful skills - backups, logs, exports, and cleanup

Error journal

Create ~/Termux-Course/errors.txt. Every time something fails, write the date, command, exact error, and what fixed it. This becomes your personal manual.

Clean project folders

Keep downloaded ZIP files in Downloads, but keep working clones in ~/Termux-Projects. This prevents confusion between archives and active folders.

Backup before experiments

Before editing a working script, copy it: cp script.py script.py.bak. If the edit breaks it, restore the backup.

Use Tab completion

Type the first letters of a file or folder and press Tab. This prevents spelling mistakes and handles long names faster.

Use quotes for spaces

If a file name has spaces, quote it. This applies to Python files, folders, and paths.

Check output location

When a script says it saved a file, immediately run ls in the expected folder or check Downloads. Do not assume it saved correctly.

Phone heat and battery

Long installs can heat a low/mid-range phone. Keep screen brightness moderate and do not run heavy builds while charging under a pillow or blanket.

Stable internet first

If pkg or git errors happen during weak Wi-Fi, do not troubleshoot the project yet. Fix the network first.

One source per app family

Install Termux and add-ons from the same source to avoid signature mismatch.

Learn from successful output

When a command works, read the output too. Success output teaches normal behavior; errors are easier to spot later.

Seven-day plan with exercises and checks

The seven days are not meant to be rushed. A strong beginner can spend two focused hours per day. A complete beginner may need longer, and that is normal.

- Day 1: Install Termux correctly, allow storage, understand prompt, run safe commands.
- Day 2: Learn folders, files, quotes, nano, copying, moving, and safe deletion.
- Day 3: Learn pkg update, pkg upgrade, package search, and install core tools.
- Day 4: Learn GitHub browser download, ZIP extraction, git clone, git pull, and README reading.
- Day 5: Learn Python scripts, missing modules, project folders, and running quoted filenames.
- Day 6: Study DedSec Project, Offline Survival Project, and Corrupted Files Project.
- Day 7: Build a clean workspace, write notes, troubleshoot errors, and complete a final practice run.

Daily proof of work

At the end of each day, write three lines in notes.txt: what you did, what worked, what failed. This turns mistakes into a personal troubleshooting database.

```
mkdir -p ~/Termux-Course/day1
cd ~/Termux-Course/day1
nano notes.txt
```

Line-by-line explanation:

- Creates a folder for day 1.
- Moves into the folder.
- Creates a notes file for the day.

Final checklist - the reader should be able to do this alone

By the end, the reader should not only copy commands. The reader should understand how to install Termux, manage APK settings safely, navigate folders, install packages, clone projects, run Python scripts, and troubleshoot common terminal errors.

- I can download Termux from an official source and explain why the source matters.
- I can enable Allow from this source only for the app I used to install the APK.
- I can turn that permission off again after installing.
- I can run termux-setup-storage and find Downloads from Termux.
- I can explain the difference between home folder and Android Downloads.
- I can use pwd, ls, cd, mkdir, cp, mv, rm -i, cat, nano.
- I can run pkg update, pkg upgrade, pkg install, and pkg search.
- I can clone a public GitHub repository and update it with git pull.
- I can run a Python script and handle filenames with spaces.
- I can read an error message and choose the right first check instead of guessing.

Course completion rule

The course is complete when the reader can set up a new Termux install, clone the three practice projects, run their documented entry points, and write a troubleshooting note for any error without panic.

Android settings appendix - exact places to look

Android menus can be different between Samsung, Xiaomi, Pixel, OnePlus, and older phones. Do not memorize one path only. Learn the words to search inside Settings.

Search words to use in Android Settings

Search these words one by one: Install unknown apps, Unknown apps, Special app access, Permissions, Storage permission, Battery optimization, App info, Downloads.

Find Install unknown apps

Where you are: Android Settings search bar.
What you do: Type Install unknown apps. Open the result. Choose your browser or My Files.
What you should see: A list of apps with switches or a page with Allow from this source.
Why this matters: This is where Android controls which app may install APK files.
If it fails: If search finds nothing, go Settings -> Apps -> Special app access -> Install unknown apps.

Find Termux permissions

Where you are: Settings -> Apps -> Termux -> Permissions.
What you do: Open Termux permissions and check Files/Storage/Photos and videos depending on Android version.
What you should see: Permission screen shows Allowed or Not allowed.
Why this matters: Scripts cannot save to Downloads unless Android permission allows it.
If it fails: If storage is missing, run termux-setup-storage inside Termux first.

Find battery restrictions

Where you are: Settings -> Apps -> Termux -> Battery.
What you do: Set Termux to unrestricted or not optimized only while doing long installs.
What you should see: Battery page shows optimization mode.
Why this matters: Android may stop long terminal work in the background.
If it fails: Return to normal optimization later if you want maximum battery saving.

Find Downloads

Where you are: Open My Files or Files app.
What you do: Open Internal storage -> Download.
What you should see: You see APKs, ZIPs, PDFs, and exported files.
Why this matters: This is where the browser usually saves downloads.
If it fails: If you cannot find a file, sort Downloads by newest first.

Important safety note

Do not enable unknown-app install for every app. Enable it only for the browser or file manager you used, install the trusted APK, then disable it again.

APK install walkthrough - every screen the reader may see

This lesson explains the install flow as a sequence of screens. A beginner should be able to compare the phone screen with this list and know whether things are normal.

Screen A - browser download warning

Where you are: Browser after tapping Download APK.
What you do: Read the warning and confirm only if the URL is official.
What you should see: Download starts and a progress bar or notification appears.
Why this matters: The warning is generic; it appears because APKs can install apps.
If it fails: If the URL is not official, cancel the download.

Screen B - download complete

Where you are: Browser download bar or notification shade.
What you do: Tap Open, or open Downloads and tap the APK manually.
What you should see: Android package installer opens.
Why this matters: Opening the APK begins the install process.
If it fails: If Open does nothing, use My Files -> Downloads.

Screen C - permission required

Where you are: Android says this app cannot install unknown apps.
What you do: Tap Settings, enable Allow from this source, then press Back.
What you should see: You return to the installer.
Why this matters: Android is not blocking Termux specifically; it is protecting APK installs.
If it fails: If you enabled the wrong app, go back and enable the browser/file manager used to open the APK.

Screen D - install screen

Where you are: Android package installer.
What you do: Tap Install. Do not tap random popups.
What you should see: Progress appears, then App installed or App not installed.
Why this matters: This is the actual app installation step.
If it fails: If it fails, read the exact error; signature conflicts are common with Termux source mixing.

Screen E - first open

Where you are: App installed screen.
What you do: Tap Open.
What you should see: Termux opens to a terminal prompt.
Why this matters: This proves the app starts.
If it fails: If it closes immediately, reboot phone and open Termux again. If still broken, reinstall from one official source.

What to write in notes

Write the source used, exact APK filename if visible, whether Android asked for Allow from this source, and whether Play Protect scanned it.

Termux keyboard and touch controls - small things beginners miss

The terminal is harder on a phone because the keyboard hides keys that are common on a computer. Learn these controls early, because they affect nano, Ctrl+C, Tab completion, and long commands.

- Tap the terminal once before typing if the keyboard does not appear.
- Long-press inside Termux to open extra options such as paste, more, and style depending on version.
- Ctrl+C stops many running commands or local servers. It does not copy text in the terminal context.
- Ctrl+O in nano saves. Ctrl+X exits nano.
- Tab completion helps complete file and folder names. It prevents spelling mistakes.
- If paste adds strange line breaks, paste one command block at a time, not a whole page of commands.

Practice Ctrl+C safely

Where you are: Termux prompt.
What you do: Run sleep 20, then press Ctrl+C.
What you should see: The sleep command stops and prompt returns.
Why this matters: You learn how to stop a process before needing it during a real problem.
If it fails: If you cannot press Ctrl, enable Termux extra keys or use Hacker Keyboard/OpenBoard-style keyboard.

Practice Tab completion

Where you are: Inside any folder with files.
What you do: Type the first letters of a filename, then press Tab.
What you should see: Termux completes the name or shows options.
Why this matters: This prevents mistakes with long project file names.
If it fails: If nothing happens, type more letters or check that the file exists with ls.

Practice paste control

Where you are: Termux prompt.
What you do: Copy one short command, paste it, read it, then press Enter.
What you should see: Only the command you expected runs.
Why this matters: Reading before Enter prevents accidents.
If it fails: If multiple commands pasted, stop and delete the line before pressing Enter.

ZIP files, extraction, and moving projects from Downloads

Many beginners download a ZIP from GitHub and then do not know where it went or how to use it. A ZIP is an archive. You must extract it before running scripts inside it.

```
pkg install unzip
mkdir -p ~/Termux-Projects
unzip /storage/emulated/0/Download/project.zip -d ~/Termux-Projects
```

Line-by-line explanation:

- Installs unzip if missing.
- Creates a clean projects folder.
- Extracts the ZIP from Android Downloads into your Termux projects folder.

Step 1 - confirm ZIP exists

Where you are: Termux prompt.
What you do: Run ls /storage/emulated/0/Download | grep .zip.
What you should see: You see the ZIP filename.
Why this matters: You must know the exact filename before extracting.
If it fails: If nothing appears, check My Files -> Downloads or re-download.

Step 2 - extract to a clean place

Where you are: Termux prompt.
What you do: Run unzip with the full ZIP path and -d ~/Termux-Projects.
What you should see: Files extract and a folder appears.
Why this matters: Running projects from a stable folder is cleaner than running from compressed Downloads.
If it fails: If unzip says command not found, run pkg install unzip.

Step 3 - enter extracted folder

Where you are: Termux prompt.
What you do: Run cd ~/Termux-Projects, then ls, then cd exact-folder-name.
What you should see: You are inside the extracted project folder.
Why this matters: Most scripts expect to be started from their project root folder.
If it fails: If folder name is long, use Tab completion.

ZIP vs folder

Do not try to run Python files while they are still inside an unextracted ZIP. Extract first, then run from the extracted folder.

How to read a README like instructions, not decoration

The README is not just a description. For a beginner, it is the instruction sheet, safety note, map, and troubleshooting entry point.

README reading order

Read in this order: project purpose, requirements, installation, first run command, folder structure, where output saves, update instructions, known errors, license/disclaimer.

Step 1 - find the run command

Where you are: GitHub README page or local README.md.
What you do: Use browser Find in page and search: install, run, usage, python, Termux.
What you should see: You find the exact command expected by the project.
Why this matters: The correct run command saves time.
If it fails: If there is no clear run command, inspect filenames but do not run random files blindly.

Step 2 - find requirements

Where you are: README page.
What you do: Search for requirements, dependencies, pkg install, pip install.
What you should see: You see needed tools or modules.
Why this matters: Missing dependencies cause most Python errors.
If it fails: Install only what is listed or clearly required by an error.

Step 3 - find output path

Where you are: README page.
What you do: Search for save, export, output, Downloads, storage.
What you should see: You know where generated files go.
Why this matters: If you cannot find output, you may think the script failed even when it worked.
If it fails: After running, check that folder with ls.

Project worksheets - exact tasks for the three practice projects

These worksheets turn the three projects into learning exercises, not just downloads. The reader should complete each line and write the result.

DedSec worksheet

1. Clone the repo. 2. Write the repo path. 3. List top-level folders. 4. Find README install section. 5. Find a safe documented entry point. 6. Write what the project is for in your own words. 7. Run only a documented command. 8. Copy any error exactly.

Offline Survival worksheet

1. Clone or extract the repo. 2. Confirm the main Python file exists. 3. Confirm database folder exists. 4. Run the main script. 5. Search/browse one entry. 6. Find where exports save. 7. Test offline after setup. 8. Write what data files the script depends on.

Corrupted Files worksheet

1. Clone or extract the repo. 2. Open Greece or USA manually. 3. Choose one year. 4. Open one incident folder. 5. Identify text, JSON, metadata, and index files. 6. Run the reader script. 7. Search for an incident. 8. Write how manual browsing differs from the script reader.

```
cd ~/Termux-Projects
find . -maxdepth 2 -type f -name "README.md"
find . -maxdepth 3 -type f -name "*.py"
```

Line-by-line explanation:

- Moves to the projects folder.
- Finds README files near the top of projects.
- Finds Python files near the top of projects.

Terminal decision trees - choose the first safe check

A decision tree stops guessing. Start from the symptom, do the smallest safe check, then move to the next check only if needed.

When a file does not run

1. Run pwd. 2. Run ls. 3. Check spelling. 4. If name has spaces, use quotes. 5. If Python file, run python "file name.py". 6. If database missing, re-extract/re-clone. 7. If still failing, copy exact error.

When pkg fails

1. Check internet in browser. 2. Run pkg update. 3. If mirror error, run termux-change-repo. 4. Run pkg search package. 5. Install exact package. 6. If still failing, wait and retry; repositories can have temporary issues.

When git clone fails

1. Open repo URL in browser. 2. Check spelling. 3. Confirm public/private access. 4. Check internet/VPN. 5. Install git if missing. 6. For private repos, use an authenticated method. 7. Save exact error.

When Python module is missing

1. Read exact ModuleNotFoundError name. 2. Search README for requirements. 3. Search project for requirements.txt. 4. Install only needed module. 5. If build fails, check whether Termux package exists. 6. Copy full error.

Extra practice labs - make the reader stronger

These labs are small on purpose. Small labs build confidence without damaging files.

```
mkdir -p ~/Termux-Course/lab-files
cd ~/Termux-Course/lab-files
echo "first note" > note1.txt
cp note1.txt note1-backup.txt
ls
```

Line-by-line explanation:

- Creates lab folder.
- Moves into it.
- Creates a text file.
- Copies it as backup.
- Lists files to confirm both exist.

```
nano checklist.txt
cat checklist.txt
```

Line-by-line explanation:

- Opens a checklist file.
- After saving in nano, cat proves it exists.

```
mkdir folder-a folder-b
mv note1-backup.txt folder-a/
ls folder-a
```

Line-by-line explanation:

- Creates two folders.
- Moves backup file into folder-a.
- Confirms the file is there.

```
find . -type f
du -sh .
history | tail
```

Line-by-line explanation:

- Finds files below current folder.
- Shows approximate size of the lab folder.
- Shows recent commands.

Part 2 - Exercises and practical labs

This is the second part. Part 1 explains. Part 2 makes the reader do the work. Every exercise has a goal, commands, expected result, and understanding check.

Self-grade
Green: you completed it and can explain it. Yellow: you needed help. Red: it failed but you saved the exact error.

Exercise 1 - prompt and output

Goal: separate what you type from what the terminal answers.

```
echo "I control the input"
pwd
date
whoami
```

Line-by-line explanation:

- echo prints text.
- pwd shows current folder.
- date shows system date/time.
- whoami shows the environment user.

Check

Write the expected result in your notes before moving on.

Exercise 2 - folders, files, and safe deletion

Goal: handle folders without deleting the wrong things.

```
mkdir -p ~/Termux-Course/exercises/files
cd ~/Termux-Course/exercises/files
printf "line one\nline two\n" > notes.txt
ls
cat notes.txt
cp notes.txt notes-backup.txt
rm -i notes-backup.txt
```

Line-by-line explanation:

- Creates a safe folder.
- Creates a file.
- Reads it.
- Creates backup.
- rm -i asks confirmation.

Check

Write the expected result in your notes before moving on.

Exercise 3 - storage and Downloads

Goal: see the difference between Termux home and Android Downloads.

```
termux-setup-storage
ls ~/storage
ls /storage/emulated/0/Download
mkdir -p /storage/emulated/0/Download/Termux-Course-Test
echo "saved from Termux" > /storage/emulated/0/Download/Termux-Course-Test/test.txt
```

Line-by-line explanation:

- Requests storage permission.
- Shows shortcuts.
- Shows Downloads.
- Writes a test file in Downloads.

Check

Write the expected result in your notes before moving on.

Exercise 4 - packages

Goal: update, search, install, and verify.

```
pkg update
pkg search python
pkg install python -y
python --version
```

Line-by-line explanation:

- Updates package lists.
- Searches Python.
- Installs Python.
- Checks version.

Check

Write the expected result in your notes before moving on.

Exercise 5 - nano

Goal: write and save a terminal text file.

```
cd ~/Termux-Course/exercises
nano checklist.txt
cat checklist.txt
```

Line-by-line explanation:

- Open nano.
- Write 3 checklist lines.
- Ctrl+O saves, Enter confirms, Ctrl+X exits.
- cat verifies it saved.

Check

Write the expected result in your notes before moving on.

Exercise 6 - README as instructions

Goal: find install/run/requirements in a repo.

```
pkg install git -y
git clone https://github.com/dedsec1121fk/Offline-Survival-Project.git
cd Offline-Survival-Project
ls
cat README.md | head
```

Line-by-line explanation:

- Installs git.
- Downloads a verified repo.
- Shows files.
- Reads the start of README.

Check

Write the expected result in your notes before moving on.

Exercise 7 - DedSec Project worksheet

Goal: inspect a large repo before running setup.

```
mkdir -p ~/Termux-Projects
cd ~/Termux-Projects
git clone https://github.com/dedsec1121fk/DedSec
cd DedSec
ls
cat README.md | head -40
```

Line-by-line explanation:

- Downloads DedSec repo.
- Shows structure.
- Reads README.
- Find the documented setup command before running anything.

Check

Write the expected result in your notes before moving on.

Exercise 8 - Offline Survival worksheet

Goal: run a project that depends on a main script and database folder.

```
cd ~/Termux-Projects
git clone https://github.com/dedsec1121fk/Offline-Survival-Project.git
cd Offline-Survival-Project
ls
python "Offline Survival.py"
```

Line-by-line explanation:

- Downloads repo.
- Enters folder.
- Verifies files.
- Runs script with quotes because of space.

Check

Write the expected result in your notes before moving on.

Exercise 9 - Corrupted Files worksheet

Goal: manual browsing and Python reader.

```
cd ~/Termux-Projects
git clone https://github.com/dedsec1121fk/Corrupted-Files-Project.git
cd Corrupted-Files-Project
ls
find . -maxdepth 2 -type d | head
python "Corrupted Files.py"
```

Line-by-line explanation:

- Downloads archive.
- Shows folders.
- Runs reader.
- Compare manual browsing with reader.

Check

Write the expected result in your notes before moving on.

Exercise 10 - small Python script

Goal: create, run, edit, fix.

```
mkdir -p ~/Termux-Course/exercises/python
cd ~/Termux-Course/exercises/python
printf 'name = input("Name: ")\nprint("Hello " + name)\n' > hello.py
python hello.py
```

Line-by-line explanation:

- Creates Python file.
- Asks input.
- Prints answer.
- Then edit with nano and rerun.

Check

Write the expected result in your notes before moving on.

Exercise 11 - local server

Goal: understand local link.

```
mkdir -p ~/Termux-Course/exercises/web
cd ~/Termux-Course/exercises/web
printf "<h1>Hello from Termux</h1>" > index.html
python -m http.server 8080
```

Line-by-line explanation:

- Creates HTML.
- Starts server.
- Open http://127.0.0.1:8080 in browser.
- Ctrl+C stops server.

Check

Write the expected result in your notes before moving on.

Exercise 12 - troubleshooting drills

Goal: practice errors without fear.

```
cd ~/Termux-Course/exercises
python missing-file.py
cd folder-that-does-not-exist
cat not-here.txt
```

Line-by-line explanation:

- Creates controlled errors.
- Read what is missing.
- Then run pwd and ls for diagnosis.

Check

Write the expected result in your notes before moving on.

Final assignment - clean Termux workspace

Goal: combine basics into a reusable structure.

```
mkdir -p ~/Termux-Workspace/{projects,notes,backups,exports,labs}
printf "Workspace created\n" > ~/Termux-Workspace/notes/status.txt
cd ~/Termux-Workspace
find . -maxdepth 2 -type d
cat notes/status.txt
```

Line-by-line explanation:

- Creates workspace.
- Writes status.
- Shows folders.
- Add one repo, one script, one export, and one backup.

Check

Write the expected result in your notes before moving on.

Self-assessment checklist

Use this page before saying you completed the book.

- I can install an APK from an official source.
- I can explain Allow from this source.
- I can find Downloads from Termux.
- I can explain pwd, ls, cd, cat, cp, mv, rm.
- I can git clone a verified repo.
- I can read README and find run command.
- I can run a Python file with spaces using quotes.
- I can start a local server and open 127.0.0.1.
- I can save exact errors and avoid guessing fixes.

Final rule

You do not need to memorize everything. You need to know where to look, what to check first, and how not to damage your environment.

Part 3 - Android on a PC or USB for Termux and Android apps

This part is for a reader who wants to try Android on a computer, run it from a USB drive, or install it on a PC so they can use Android apps and Termux on a larger screen. It does not replace the previous lessons. The previous lessons teach Termux. This part shows how to create an Android environment on a PC without guessing. The safest starting point is always live USB testing, not installing over the main drive.

First safety rule

Do not install an Android OS onto the Windows drive unless you have a backup and you know exactly which partition you are selecting. Start with live boot from USB. Live boot shows whether Wi-Fi, keyboard, display, touchpad, audio, and apps work without touching the disk.

What you learn here

You will understand ISO, bootable USB, live boot, disk install, install to a second USB/SSD, BIOS/UEFI, Secure Boot, partition, GRUB, F-Droid, Termux inside Android-x86 style systems, and why some APKs do not work on PCs.

Which options exist and which one to choose

There are several ways to run Android apps on a PC, but they are not the same. Some are Android-x86 operating systems, some are ChromiumOS-like systems with an Android subsystem, and some are containers inside Linux. Choose by goal, not by name.

Official links

Bliss OS official site: https://blissos.org/

Bliss OS documentation: https://docs.blissos.org/

Bliss OS live boot guide: https://docs.blissos.org/installation/live-boot-bliss-os/

Bliss OS bootable USB install guide: https://docs.blissos.org/installation/install-from-bootable-usb/

Android-x86 official site: https://www.android-x86.org/

Android-x86 download page: https://www.android-x86.org/download

Android-x86 installation guide: https://www.android-x86.org/installhowto.html

FydeOS download page: https://fydeos.io/download/

FydeOS for PC: https://fydeos.io/download/pc/

FydeOS Android apps guide: https://fydeos.io/help/manual/manage-your-apps/add-apps-and-extensions/install-android-apps-on-fydeos/

Waydroid install instructions: https://docs.waydro.id/usage/install-on-desktops

Waydroid install/run apps: https://docs.waydro.id/usage/install-and-run-android-applications

ChromeOS Flex differences: https://support.google.com/chromeosflex/answer/11542901

If you want clean Android on PC

Where you are: Before downloading anything.
What you do: Start with Android-x86 or Bliss OS. Download only from official sites/docs. Create a bootable USB and live boot first.
What you should see: You see an Android desktop or Android setup wizard from USB.
Why this matters: These options are closest to Android on a computer.
If it fails: If Wi-Fi or graphics fail in live mode, do not install yet.

If you want a laptop-like experience

Where you are: Before choosing the OS.
What you do: Check FydeOS for PC. It is closer to a ChromeOS-like desktop with Android apps on supported hardware.
What you should see: You see a ChromiumOS-like desktop and Android app section, if supported.
Why this matters: It can be friendlier for daily desktop use, not only testing Android.
If it fails: If the Android subsystem is unsupported on your PC, it is not a good Termux/Android-app target.

If you already have a Linux laptop

Where you are: On Ubuntu/Debian/Fedora etc.
What you do: Check Waydroid instead of installing a new OS. It is not an Android OS install. It is an Android container inside Linux.
What you should see: An Android environment runs inside your Linux session.
Why this matters: It is less risky than changing partitions.
If it fails: If you only have Windows, Waydroid is not the simple path.

What not to choose for this goal

Where you are: Before wasting time with the wrong ISO.
What you do: Do not choose ChromeOS Flex if your goal is general Android apps and Termux. It is not the right path for this book.
What you should see: You understand that ChromeOS Flex is a different thing.
Why this matters: This book wants Android apps/Termux, not only a lightweight browsing OS.
If it fails: If you already use ChromeOS Flex, use Linux environment only if your model supports it, but do not expect a full Android app experience.

Why random OS names are not included

Older projects like Remix OS are discontinued, and other projects often have stale or confusing pages. This book prefers verified official links over fake downloads. If there is no clear official download/install page, it is not used as a main guide.

Vocabulary before touching USB or disk

If you do not understand these words, do not jump to install. Read them as a small dictionary. When you see them in an installer, you will know what is happening.

- ISO/IMG: operating system image. It is not a normal file you copy to the USB. It must be written with a tool such as Rufus, Etcher, or dd.
- Bootable USB: a USB drive that the computer can start from before Windows opens.
- Live boot: testing an operating system from USB without installing to disk. Changes usually disappear after shutdown.
- Installer mode: mode that writes Android to a disk or partition. It can erase data if you choose the wrong target.
- BIOS/UEFI: firmware settings for startup. You change boot order, Secure Boot, and storage mode there.
- Boot menu: quick device selection menu, often F12, F9, F10, F11, Esc, or Del depending on brand.
- Secure Boot: UEFI protection that can block non-Windows boot media. If you disable it, write down the original setting.
- BitLocker/recovery key: if Windows uses drive encryption, Secure Boot/TPM changes can request the recovery key. Save it before BIOS changes.
- Partition: a piece of a disk. Wrong partition can mean lost Windows or files.
- EXT4: Linux filesystem often recommended for Android-x86/Bliss installs.
- NTFS: Windows filesystem. Some installers can use it, but follow instructions exactly.
- GRUB: bootloader that shows startup choices, such as Windows or Android.
- x86_64: 64-bit PC architecture. Many Android APKs are ARM-only and may not run properly on x86_64 Android.

Tiny exercise before continuing

Write in a notes app: which PC you have, RAM size, SSD/HDD, free space, how to open boot menu, and whether BitLocker is enabled.

Before downloading an OS - hardware check and backup

The same ISO can work perfectly on one laptop and badly on another. The reason is drivers: Wi-Fi, Bluetooth, GPU, audio, touchpad, sleep, brightness. That is why you check first and install later.

Step 1 - write PC information

Where you are: Inside Windows or Linux.
What you do: On Windows go to Settings -> System -> About. Write processor, RAM, and system type. On Linux run uname -m and lscpu.
What you should see: You know whether it is x64/x86_64 and how much memory you have.
Why this matters: Android PC builds usually target x86/x86_64, not ARM laptops.
If it fails: If you do not know the architecture, do not download an ISO blindly.

Step 2 - backup before any install

Where you are: Windows or Linux file manager.
What you do: Copy important files to an external disk or cloud. If BitLocker exists, save the recovery key.
What you should see: Your important files exist outside the disk you may modify.
Why this matters: Partition mistakes can erase data. Backup is your safety.
If it fails: If you do not have backup, do live boot only, not install.

Step 3 - find boot menu key

Where you are: Browser or PC manual.
What you do: Search: "your laptop model boot menu key". Write the key.
What you should see: You know how to choose USB at startup.
Why this matters: Without boot menu, you may think the USB is broken when you simply did not select it.
If it fails: If you do not know the model, check the laptop sticker or Windows About.

Step 4 - prepare a separate USB

Where you are: On your desk before opening Rufus/Etcher.
What you do: Use a 16GB+ USB that contains no important files.
What you should see: The USB can be erased safely.
Why this matters: Creating a bootable USB erases the USB content.
If it fails: If it has files, copy them elsewhere first.

```
# Linux checks, only if you are already on Linux
uname -m
lscpu | head
lsblk
```

Line-by-line explanation:

- uname -m shows architecture such as x86_64.

- lscpu shows CPU information.
- lsblk shows disks/USBs and helps avoid choosing the wrong drive.

Bliss OS - what to know before using it

Bliss OS is a known option for Android on PC, but use it carefully. The official page says the project is under maintenance/lockdown and that older public releases are snapshots without further updates/support. So this book does not tell you to install Bliss blindly. It says: check the current official page, read the docs, live boot, then decide.

Official links

Bliss OS: https://blissos.org/

Bliss OS docs: https://docs.blissos.org/

Live boot guide: https://docs.blissos.org/installation/live-boot-bliss-os/

Bootable USB install guide: https://docs.blissos.org/installation/install-from-bootable-usb/

Windows installer docs: https://docs.blissos.org/installation/using-the-windows-installer/

Step 1 - read status

Where you are: Browser.
What you do: Open blissos.org and docs.blissos.org. Check for new warnings, new downloads, or new instructions.
What you should see: You know whether older instructions still apply or whether you should wait for a new build.
Why this matters: The project changes and older instructions can become obsolete.
If it fails: If the page says docs are obsolete, do not install to the main drive.

Step 2 - live boot first

Where you are: Windows with Rufus or another USB writer.
What you do: Create a bootable USB and choose live boot from boot menu.
What you should see: Android loads from USB without installation.
Why this matters: You check hardware without touching Windows.
If it fails: If you get black screen, try safe graphics/VESA/another boot option if available.

Step 3 - check before install

Where you are: Inside live Bliss OS.
What you do: Test Wi-Fi, keyboard, mouse/touchpad, sound, brightness, sleep, browser, F-Droid/Termux install.
What you should see: You know what works and what fails.
Why this matters: Do not install an OS that is not usable on your PC.
If it fails: If something basic fails, stop and keep notes instead of installing.

Android-x86 - a simple option for trying Android on PC

Android-x86 is a classic project for Android on PC. The official install guide explains ISO download, bootable USB, live boot, partition install, GRUB, read-write /system, and auto installation. For a beginner, the most important rule is: live boot first, install only to a safe target.

Official links

Android-x86 official site: https://www.android-x86.org/

Android-x86 download: https://www.android-x86.org/download

Android-x86 install howto: https://www.android-x86.org/installhowto.html

Step 1 - download ISO

Where you are: PC browser.
What you do: Open the official download page and download the latest suitable ISO from an official mirror.
What you should see: You have a .iso file in Downloads.
Why this matters: The ISO is the image that will be written to USB.
If it fails: If you downloaded HTML or the wrong file, return to the official mirror and wait for completion.

Step 2 - bootable USB

Where you are: Rufus/Etcher on Windows or dd on Linux.
What you do: Write the ISO to USB. Do not copy-paste the ISO file to the USB.
What you should see: The USB becomes bootable and its old files are erased.
Why this matters: The computer needs bootable structure, not a normal file.
If it fails: If the USB does not appear in the boot menu, rewrite the ISO with another tool or USB port.

Step 3 - live mode

Where you are: PC boot menu.
What you do: Choose Live CD/Run Android-x86 without installation if available.
What you should see: Android starts without writing to disk.
Why this matters: Live mode is a compatibility test.
If it fails: If the screen goes black, try VESA/no hardware acceleration if available.

Step 4 - install only when sure

Where you are: Installer menu.
What you do: If installing, choose the correct partition/drive and read format prompts. Do not choose auto installation on the main disk if it has Windows you want to keep.
What you should see: The installer writes Android and possibly a bootloader.
Why this matters: This is where data-loss risk becomes real.
If it fails: If you do not know which partition is which, cancel and learn with Disk Management/lsblk first.

FydeOS - laptop-style option with Android apps

FydeOS for PC is closer to a ChromeOS-like experience. It can be friendlier for daily use, but Android subsystem support is not guaranteed on every PC. The official Android apps page notes that the Android subsystem and related functions can only be adapted to Intel graphics cards, so you must check compatibility before treating it as a Termux solution.

Official links

FydeOS download: https://fydeos.io/download/

FydeOS for PC: https://fydeos.io/download/pc/

FydeOS Android apps guide: https://fydeos.io/help/manual/manage-your-apps/add-apps-and-extensions/install-android-apps-on-fydeos/

Step 1 - choose the right build category

Where you are: FydeOS for PC download page.
What you do: Choose Intel Slim, Intel Modern, AMD Graphics, or another category that matches your PC.
What you should see: You do not download a random image.
Why this matters: The correct image improves the chance of working graphics and Android subsystem.
If it fails: If you do not know CPU/GPU, return to the hardware checklist.

Step 2 - test before relying on Android apps

Where you are: Live USB or test install.
What you do: After boot, open Launcher -> Android and see if the Android environment starts.
What you should see: The Android environment starts and you can install apps.
Why this matters: If this fails, the Termux/Android-app goal fails.
If it fails: If Android is missing or does not start, do not continue as a Termux PC solution.

Step 3 - install Termux

Where you are: Inside the Android environment.
What you do: Use an app store or F-Droid/APK only from official source and install Termux.
What you should see: Termux opens and shows a prompt.
Why this matters: Having an Android subsystem does not automatically mean every APK works.
If it fails: If Termux will not install, check architecture/support and try the F-Droid source.

Waydroid - Android apps inside Linux without a new Android OS

Waydroid is not a separate Android OS to boot from USB. It is a way to run an Android environment inside a Linux desktop, usually on a Wayland session. It is useful if you already have a Linux laptop and do not want to change partitions for Android-x86/Bliss.

Official links

Waydroid docs: https://docs.waydro.id/

Install on desktops: https://docs.waydro.id/usage/install-on-desktops

Install/run Android applications: https://docs.waydro.id/usage/install-and-run-android-applications

Step 1 - confirm Linux/Wayland

Where you are: Linux desktop.
What you do: Open a terminal and check your session. Ubuntu 22.04+ can usually use Wayland.
What you should see: You know whether the environment is suitable.
Why this matters: Waydroid needs the right Linux desktop/container setup.
If it fails: If you are on Windows only, this is not the first choice.

Step 2 - install on Ubuntu/Debian-like systems

Where you are: Linux terminal.
What you do: Follow official commands for the curl repo script and apt install waydroid.
What you should see: The package installs and Waydroid appears in the app menu.
Why this matters: You use the official repo instead of random blog scripts.
If it fails: If the repo script fails, check distro codename and official docs.

Step 3 - install APK

Where you are: Linux terminal or app menu.
What you do: Use waydroid app install xyz.apk or F-Droid inside the Android environment.
What you should see: The app is installed in the Android container.
Why this matters: This lets you test Android apps without dual boot.
If it fails: If the APK is ARM-only, it may not work on x86_64.

```
sudo apt install curl ca-certificates -y
curl -s https://repo.waydro.id | sudo bash
sudo apt install waydroid -y
```

Line-by-line explanation:

- Installs prerequisites.
- Adds the official Waydroid repo script.
- Installs Waydroid. Run only in a suitable Linux environment after reading the official docs.

```
waydroid app install xyz.apk
waydroid app launch com.example.app
```

Line-by-line explanation:

- Installs an APK in the Waydroid container.
- Launches the app if you know the package name.

ChromeOS Flex and names to be careful with

Do not confuse every lightweight PC OS with Android OS. ChromeOS Flex can be useful for turning an old PC into a ChromeOS-like device, but it is not the right choice for this part if your goal is general Android apps and Termux. Google official docs explain that ChromeOS Flex differs from ChromeOS and does not provide the same general Google Play/Android app support as Chromebooks.

Official links

ChromeOS Flex differences official page: https://support.google.com/chromeosflex/answer/11542901

- ChromeOS Flex: good for browser/cloud use, not the main target for a Termux Android app lab.
- Remix OS: discontinued, so it is not used as a modern install guide.
- Phoenix OS or old Android PC builds: often stale/unclear sources. Do not use random ISO reupload sites.
- PrimeOS: the checked official site promotes PrimeOS 3.0/Primebook. Without a clear verified generic PC install page, it is not included here as an install path.

OS choice rule

If you cannot find an official site, official download, official docs, and recent support indication, do not put it on your main computer. Test only on a spare machine or VM if you accept the risk.

Creating a bootable USB on Windows with Rufus

Rufus writes the ISO to the USB in a way your computer can boot. This erases the USB. Do not use a USB that contains photos or important files.

Official links

Rufus official site: https://rufus.ie/

Step 1 - download Rufus

Where you are: Windows browser.
What you do: Open https://rufus.ie/ and download the portable or normal version.
What you should see: You have a Rufus .exe in Downloads.
Why this matters: Rufus creates bootable USB drives.
If it fails: If you downloaded it from another site, cancel and get the official one.

Step 2 - choose USB and ISO

Where you are: Rufus window.
What you do: Plug in the USB, open Rufus, choose the USB in Device, press Select, and choose the Android ISO.
What you should see: Rufus shows the USB and ISO filename.
Why this matters: You must confirm you are writing to the correct USB.
If it fails: If you see an external disk with data, stop.

Step 3 - start and warnings

Where you are: Rufus window.
What you do: Keep default settings unless the official guide says otherwise. Press Start. If it asks ISO mode, choose ISO mode when the guide recommends it.
What you should see: Rufus warns that it will erase the USB and then writes the image.
Why this matters: Warnings protect you from the wrong device.
If it fails: If a Syslinux download prompt appears, follow the official guide for that OS.

Step 4 - safe eject

Where you are: Windows taskbar.
What you do: When it says Ready, close Rufus and safely eject the USB.
What you should see: The USB is ready for boot.
Why this matters: Safe eject reduces corrupted USB risk.
If it fails: If boot fails, rewrite the USB and try another port.

Creating a bootable USB on Linux with dd without erasing the wrong disk

dd is powerful and dangerous. It does not ask if you are sure. If you use the wrong of=/dev/sdX, you can erase the main disk. Use it only if you understand which device is the USB.

Step 1 - find the USB device

Where you are: Linux terminal.
What you do: Run lsblk before and after plugging the USB. Compare which new device appeared.
What you should see: You know whether the USB is /dev/sdb or /dev/sdc.
Why this matters: You must not write to /dev/sda or nvme0n1 if that is the main disk.
If it fails: If you are not 100% sure, do not use dd.

Step 2 - write the ISO

Where you are: Linux terminal.
What you do: Run dd with if=the ISO and of=the USB device, not a partition like /dev/sdb1.
What you should see: The terminal may take time, then returns to prompt.
Why this matters: This writes the raw image to the USB.
If it fails: If permission error appears, use sudo. If the device is wrong, cancel before pressing Enter.

Step 3 - sync and remove

Where you are: Linux terminal.
What you do: Run sync and then eject/unmount.
What you should see: Writes finish before you remove the USB.
Why this matters: Removing USB early can break the boot media.
If it fails: If it does not boot, rewrite from the beginning.

```
lsblk
# Replace android.iso and /dev/sdX with your real ISO and USB device
sudo dd if=android.iso of=/dev/sdX bs=4M status=progress oflag=sync
sync
```

Line-by-line explanation:

- lsblk shows disks and partitions.
- dd writes the ISO to the USB device.
- sync waits for writes to finish. Be careful with /dev/sdX.

Booting from USB without getting lost in BIOS

After creating the bootable USB, you must tell the PC to start from USB instead of the internal disk. You do this with a boot menu or BIOS/UEFI boot order.

Step 1 - quick boot menu

Where you are: PC off or restarting.
What you do: Insert USB. Press power and repeatedly tap F12/F9/F10/F11/Esc/Del depending on model.
What you should see: A boot menu opens with devices.
Why this matters: You do not need to permanently change BIOS settings.
If it fails: If it does not open, search exact model + boot menu key.

Step 2 - choose USB correctly

Where you are: Boot menu.
What you do: Choose the USB. If you see UEFI: USB and plain USB, start with UEFI on modern PCs.
What you should see: GRUB or Android boot menu loads.
Why this matters: UEFI is modern startup for most computers.
If it fails: If it does not start, try the other USB entry or rewrite the image.

Step 3 - Secure Boot/TPM carefully

Where you are: BIOS/UEFI settings.
What you do: If the official guide requires it, temporarily disable Secure Boot and write down the original setting. Do not touch TPM/drive encryption without the BitLocker recovery key.
What you should see: The USB may boot.
Why this matters: Some Android PC builds are not signed for Secure Boot.
If it fails: If Windows asks for BitLocker recovery, use the saved key or restore previous BIOS settings.

Do not change everything at once

Change one setting, test, and write the result. If you change Secure Boot, TPM, AHCI, boot order, and partitions all together, you will not know what caused the problem.

Live boot test checklist - before installing

Live boot is your filter. If something basic does not work in live USB, it will probably bother you after installation too. Do not skip this checklist.

- Display: correct resolution, no black screen, no flicker.
- Keyboard: can type in search/input fields.
- Mouse/touchpad: click, scroll, right click if needed.
- Wi-Fi: sees networks, connects, opens browser.
- Bluetooth: optional, but note whether it works.
- Audio: plays video or sound test.
- Battery/charging on laptop: shows status or not.
- Sleep/wake: sleeps and returns without freezing.
- USB storage: sees another USB if inserted.
- Android apps: can install F-Droid or APK from official source.
- Termux: opens, shows prompt, runs echo hello and pkg update.

```
echo hello
uname -m
getprop ro.product.cpu.abi
getprop ro.build.version.release
```

Line-by-line explanation:

- echo hello checks terminal input.
- uname -m shows kernel architecture.
- getprop ro.product.cpu.abi shows Android ABI.
- getprop ro.build.version.release shows Android version.

Checklist result

If 8/11 work and the basics you need are okay, you can consider installing. If Wi-Fi, keyboard, or graphics fail, do not install on the main PC.

Installing to disk or second USB/SSD - what each option means

Permanent installation is the most dangerous point. The installer may write partitions and bootloader. For beginners, the best practice is a second disk, external SSD, or old laptop without important data.

Option 1 - live USB only

Where you are: No install.
What you do: Each time, boot from USB and test.
What you should see: The disk does not change.
Why this matters: Safest option for learning.
If it fails: It may not save data after reboot depending on OS/persistence.

Option 2 - install to separate USB/SSD

Where you are: Installer target selection.
What you do: Choose an external drive you emptied and know is correct.
What you should see: The Android OS lives on external media.
Why this matters: This reduces risk to the main disk.
If it fails: If you choose the wrong disk, you can erase Windows.

Option 3 - dual boot on internal disk

Where you are: Installer partition selection.
What you do: Only if you have backup, a free partition, and understand GRUB/EFI.
What you should see: At startup, choose Windows or Android.
Why this matters: More practical but riskier.
If it fails: If bootloader breaks, you need a Windows/Linux recovery USB.

Option 4 - whole disk Android

Where you are: Auto installation or wipe disk.
What you do: Only on a spare PC.
What you should see: The PC boots directly into Android OS.
Why this matters: Simple but destructive for existing data.
If it fails: Do not do it on your main laptop with personal files.

Red line

If the installer says erase disk, format, write partition table, or auto install, stop and verify the disk. These options can erase everything.

Installing Termux inside the Android PC environment

After Android opens on the PC, Termux is installed like on an Android device, with one important difference: PC Android is usually x86_64. Some apps/APKs are ARM-only and do not work. Termux from F-Droid/GitHub official source is the cleanest start.

Official links

Termux official website: https://termux.dev/en/

Termux F-Droid package: https://f-droid.org/en/packages/com.termux/

Termux GitHub releases: https://github.com/termux/termux-app/releases

Step 1 - open browser inside Android OS

Where you are: Android PC browser.
What you do: Go to the official Termux/F-Droid link. Do not search random APK sites.
What you should see: You see an official package/release page.
Why this matters: The source is the most important security step.
If it fails: If you have no internet, fix Wi-Fi/Ethernet first.

Step 2 - install F-Droid or Termux APK

Where you are: Android downloads/installer.
What you do: Download APK, open it, allow from this source only for the browser/file manager, tap Install.
What you should see: You see App installed.
Why this matters: This is the same sideload logic you learned on phone.
If it fails: If App not installed appears, check old install, architecture, or corrupted download.

Step 3 - first Termux test

Where you are: Termux app.
What you do: Open Termux and run echo hello, uname -m, pkg update.
What you should see: You see prompt, output, and package update lines.
Why this matters: This proves Termux works on the Android PC OS.
If it fails: If pkg update fails, check internet/date/time/DNS and Termux source.

```
echo hello
uname -m
pkg update
pkg install git python nano -y
python --version
```

Line-by-line explanation:

- Basic input test.
- Shows architecture.
- Refreshes package lists.
- Installs core tools.

- Checks Python.

Why some Android apps do not work on PC Android

The reader must know that Android on PC does not mean every APK works. Differences include CPU architecture, Google Play Services, sensors, camera, GPS, DRM, graphics acceleration, and input method.

- ARM-only APK: many Android apps are built for ARM phones. On x86_64 PC Android they may fail or require a translation layer.
- Google Play Services: apps requiring GMS may not open without Play Store/Services or proper certification.
- Camera/GPS/sensors: laptop hardware does not always have the same sensors as a phone.
- Games: may require Vulkan/OpenGL drivers that do not work well in the PC build.
- Termux packages: on x86_64, some packages may differ from an ARM Android phone.
- Storage paths: Downloads/shared storage can behave differently depending on OS.

Check architecture before blaming the app

Where you are: Inside Termux or Android terminal.
What you do: Run uname -m and getprop ro.product.cpu.abi.
What you should see: You know if you are x86_64/x86 or something else.
Why this matters: APK compatibility depends on ABI.
If it fails: If the app is ARM-only, look for x86 compatible version or another method.

Check Google dependency

Where you are: Inside the Android app.
What you do: If the app says Google Play Services required, write it down.
What you should see: You know it is not a Termux error.
Why this matters: You do not solve Google dependency with pkg install.
If it fails: Choose F-Droid alternatives or a build with proper Play support if legally/appropriately needed.

Troubleshooting for Android OS on PC/USB

These are common problems. Do not change ten things at once. Make one fix, retry, and write the result.

USB does not appear in boot menu

Where you are: Boot menu/BIOS.
What you do: Try another USB port, rewrite ISO with Rufus/Etcher, check UEFI/Legacy mode.
What you should see: The USB appears as UEFI: USB name or USB HDD.
Why this matters: Bad image writing or wrong boot mode is common.
If it fails: If it continues, try another USB stick.

Black screen after boot

Where you are: Android boot menu.
What you do: Try safe graphics/VESA/no hardware acceleration if available. Try another build.
What you should see: The system reaches desktop/setup.
Why this matters: GPU drivers are a common problem in Android-x86 builds.
If it fails: If no option works, the hardware is not a good target.

Wi-Fi does not work

Where you are: Inside live Android.
What you do: Try USB tethering from phone, USB Ethernet adapter, or another build.
What you should see: You get temporary internet.
Why this matters: Many Wi-Fi chipsets lack drivers in a specific kernel.
If it fails: If you need Wi-Fi daily, do not install permanently until solved.

Termux will not install

Where you are: Android installer.
What you do: Check source, architecture, previous install, and whether the APK fully downloaded.
What you should see: App installed or a clear error message.
Why this matters: App not installed is a clue, not an answer.
If it fails: Download again from F-Droid/GitHub official and do not mix sources.

pkg update fails on Android PC

Where you are: Termux.
What you do: Check internet, date/time, DNS, proxy/VPN, then try termux-change-repo.
What you should see: Repositories respond and update finishes.
Why this matters: Wrong time or DNS can break HTTPS/repositories.
If it fails: If all fails, save exact error and try another network.

Windows does not boot after install

Where you are: Boot menu/BIOS/GRUB.
What you do: Do not panic. Check whether Windows exists in boot menu. If not, you may need Windows recovery USB or bootloader repair.
What you should see: You can return to Windows or repair boot.
Why this matters: GRUB/EFI may have changed boot order.
If it fails: If inexperienced, ask for help before writing commands to the EFI partition.

BitLocker recovery screen

Where you are: Windows boot.
What you do: Enter the saved recovery key or restore BIOS settings you changed.
What you should see: Windows unlocks.
Why this matters: Secure Boot/TPM changes can trigger BitLocker recovery.
If it fails: If you do not have the key, stop making changes and find it from your Microsoft account/backup.

Part 3 exercises - Android PC/USB lab

These exercises do not require permanent installation. The goal is to learn research, bootable USB, and live testing. Permanent install happens only if you have backup and a spare disk/PC.

Exercise 13 - OS choice without fake links

Where you are: Browser and notes app.
What you do: Open official links for Bliss OS, Android-x86, FydeOS, and Waydroid. For each one write: official download, official docs, target hardware, can it run Android apps/Termux?
What you should see: You have a comparison table of 4 options.
Why this matters: You learn to choose with evidence, not YouTube hype.
If it fails: If you cannot find official download/docs, write "not safe choice for now".

Exercise 14 - create bootable USB

Where you are: Windows/Rufus or Linux/dd.
What you do: Download an ISO from official source and write it to an empty USB.
What you should see: The USB appears as bootable in the boot menu.
Why this matters: You learn the difference between copying an ISO and writing an image.
If it fails: If it does not boot, rewrite the USB and note the tool/settings.

Exercise 15 - live boot checklist

Where you are: PC from USB.
What you do: Enter live mode and complete the hardware checklist. Do not install.
What you should see: You know what works on your PC.
Why this matters: Live test protects you from bad installation.
If it fails: If three basics fail, try another OS/build.

Exercise 16 - Termux on Android PC

Where you are: Live Android environment.
What you do: Open browser, download F-Droid/Termux from official source, install, and run echo hello + pkg update.
What you should see: Termux opens and runs basic commands.
Why this matters: This proves the goal of Part 3.
If it fails: If it fails, write exact error and architecture.

Exercise 17 - compare phone and PC

Where you are: Notes app.
What you do: Write differences: keyboard, screen, paths, performance, app compatibility, storage, errors.
What you should see: You have a personal conclusion about which environment helps you.
Why this matters: The good choice depends on use, not OS name.
If it fails: If unsure, keep PC setup as test lab and phone as main Termux.

Exercise 18 - final install report

Where you are: Notes app or text file.
What you do: Write one page: which OS you tested, where you downloaded it, what worked, what failed, which errors appeared, what you would do differently.
What you should see: You have a clean technical report.
Why this matters: This is real learning, not just installation.
If it fails: If you ask for help, this report gives all useful context.

Final result

After Part 3, you do not need to have installed anything permanently. You need to explain which option is safe for your PC and complete at least one correct live boot test.