

Month 1 – Ultra-Detailed, Day-by-Day Execution Plan (Zero Ambiguity)

This version is designed so you never ask: "What should I do today?"

Every day has: **prerequisites** → **exact steps** → **time estimates** → **quick checks** → **pitfalls**.

Fully aligned with the provided roadmap: Month 1 = Python fluency + Git discipline + LeetCode habit + backlog priority.

Assumed Background

- You know what variables, loops, and functions roughly are (basic programming exposure).
- You have used a computer normally (file system, browser, installs).
- You are serious about following instructions exactly.

If any assumption is false, you must slow down and repeat earlier days instead of skipping.

Operating Rules (Strict)

- **Mon–Fri (every day):**
 - Learn one concept
 - Implement in notebook
 - Solve exactly 1 LeetCode Easy
 - Push exactly 1 commit
 - **Saturday:** cleanup + refactor + better explanations
 - **Sunday:** backlog subject first priority
 - **Never skip commits** (even improvement counts)
-

Daily Time Budget

- Weekdays: **2.5–3 hours**
- Saturday: **3–4 hours**
- Sunday: **2–3 hours (lighter + backlog heavy)**

Breakdown per weekday:

- 50–60 min: Learn concept
 - 45–60 min: Implement in notebook
 - 30–40 min: LeetCode
 - 15–20 min: Git commit + README update
-

Required Setup (Must exist by end of Day 1)

Repo structure:

```
ml-foundations/  
  notebooks/  
  leetcode/  
  README.md
```

README must contain:

- Daily learning log
 - Topics covered
 - Reflections
-

WEEK 1 – Environment, Syntax, Discipline

Goal: Build foundation correctly, avoid bad habits, build consistency.

Day 1 – Setup, Python, First Notebook (Environment-first order)

Time target: 2.5–3 hrs

Prerequisites: Computer with internet

Rationale: You asked whether venv should come before learning Python. The correct order is: install Python → verify → create venv → install packages inside venv → then write code. This prevents bad habits later.

Steps (execute strictly in order)

1. **Install Python 3.10+** from python.org
 - After install: open terminal → `python --version` must show 3.x
2. **Install Git** → verify with `git --version`
3. **Create GitHub account** (if not exists)
4. **Create repo on GitHub:** `ml-foundations`
5. **Clone locally:**
 - `git clone <your-repo-url>`
 - `cd ml-foundations`
6. **Create virtual environment (venv)**
 - `python -m venv .venv`
7. **Activate venv**
 - Linux/macOS: `source .venv/bin/activate`
 - Windows: `.venv\Scripts\activate`
 - Check: terminal should show `(.venv)`
8. **Upgrade pip (inside venv only)**
 - `pip install --upgrade pip`
9. **Install Jupyter inside venv**
 - `pip install notebook`
10. **Create folders**
 - `mkdir notebooks leetcode`
11. **Create first notebook**
 - Run: `jupyter notebook`
 - Create: `notebooks/python_basics.ipynb`

Learn (30–40 min)

- `print()`
- variables
- int, float, string

Implement (30–40 min)

In notebook write and run:

- `print("Hello World")`

- store your name in variable
- add two numbers
- convert string to int using `int()`

LeetCode (30–40 min)

- Problem: Two Sum
- Save solution as: `leetcode/day01_two_sum.py`

Git (10–15 min)

- `git add .`
- `git commit -m "Day 1: setup, venv, first notebook"`
- `git push`

Quick Check

- Can you explain what venv isolates?
- Does your terminal show `(.venv)` when active?
- Does GitHub show today's commit?

Common Pitfalls + Fixes

- **python not found** → reinstall Python and check "Add to PATH"
 - **pip installs globally** → you forgot to activate venv
 - **Jupyter not opening** → run `pip install notebook` inside venv
-

Day 2 – Conditionals

Learn

- if, elif, else

Implement

Notebook:

- Check even/odd

- Compare two numbers
- Grade system example

LeetCode

- Palindrome Number

Quick Check

- When does elif run?
-

Day 3 – Loops

Learn

- for loop
- while loop

Implement

- Sum of numbers 1-100
- Multiplication table
- Print pattern

LeetCode

- FizzBuzz

Pitfall

- Infinite while loop (always change loop variable)
-

Day 4 – Lists

Learn

- list creation

- append, remove
- indexing, slicing

Implement

- Store 5 names, print longest
- Reverse list manually

LeetCode

- Contains Duplicate
-

Day 5 – Dicts + Tuples

Learn

- dict key/value
- tuple immutability

Implement

- Phonebook using dict

LeetCode

- Valid Anagram
-

Saturday (Week 1 Review)

- Reopen notebook
- Add markdown headings
- Delete useless experiments

Sunday

- Backlog study
- Redo hardest LC

- Write 3–4 lines reflection in README
-

WEEK 2 – Functions + Better Structure

Day 6 – Functions

- Learn: def, return
- Write 5 small utility functions
- LC: Best Time to Buy/Sell Stock

Day 7 – Arguments

- Learn default args
- Build calculator using functions
- LC: Merge Strings Alternately

Day 8 – Sets

- Union, intersection
- LC: Jewels and Stones

Day 9 – Strings deeper

- Methods, slicing
- LC: Valid Palindrome

Day 10 – Rebuild notebook

- Rewrite one notebook cleanly
- LC: Roman to Integer

Saturday: Merge notebooks

Sunday: Backlog + revise LC

WEEK 3 – OOP + Modules + venv

Day 11 – Classes

- Student class
- LC: Climbing Stairs

Day 12 – Methods

- Add methods to class
- LC: Ransom Note

Day 13 – Inheritance

- Animal → Dog
- LC: Isomorphic Strings

Day 14 – Modules

- Create two .py files and import
- LC: Majority Element

Day 15 – venv

- `python -m venv venv`
- Activate venv
- Document why it matters
- LC: Add Digits

Saturday: Refactor

Sunday: Backlog focus

WEEK 4 – File I/O, Errors, Polish

Day 16 – Reading files

- `open()`, `read()`
- LC: Length of Last Word

Day 17 – Writing files

- `write()`, `with`
- LC: Power of Two

Day 18 – Exceptions

- `try/except/finally`
- LC: Missing Number

Day 19 – Cleanup day

- Rename variables
- Better comments
- LC: Single Number

Day 20 – README overhaul

- Project description
- Structure
- LC: Move Zeroes

Saturday: Final polish

Sunday: Reflection + backlog

WEEK 4 (continued) – File I/O, Errors, Polish (Days 16–23)

Day 16 – Reading Files (File I/O I)

Steps

- Create a file `sample.txt` with 5 lines of text
- Learn: `open()` , modes(`'r'` , `'w'` , `'a'`), `read()` , `readline()` , `readlines()`
- Write code to:
 - Count number of lines
 - Count number of words
- LC: Length of Last Word

Quick Check

- Difference between `read()` and `readlines()` ?

Pitfall

- Forgetting to close files (fix using `with` statement tomorrow)
-

Day 17 – Writing Files (File I/O II)

Steps

- Learn: `with open(...) as f:`
- Write program that logs today's learning into `log.txt`
- Append new line every run
- LC: Power of Two

Quick Check

- Why is `with` safer than manual `close()` ?
-

Day 18 – Error Handling

Steps

- Learn: `try` , `except` , `else` , `finally`
- Write program that safely handles:
 - Wrong input
 - Division by zero
 - File not found
- LC: Missing Number

Pitfall

- Catching broad `Exception` blindly
-

Day 19 – Code Quality Day

Steps

- Reopen all notebooks
- Rename bad variables (`x` , `temp` , `abc` → meaningful names)
- Add docstrings to functions
- LC: Single Number

Quick Check

- Can a stranger understand your code without asking you?
-

Day 20 – README Overhaul

Steps

- Rewrite README with:
 - What this repo is
 - What you learned
 - How to run notebooks
 - Folder structure
 - LC: Move Zeroes
-

Day 21 (Saturday) – Deep Refactor Day

Steps (3–4 hrs)

- Merge related notebooks into fewer, cleaner files
- Ensure each notebook has:
 - Title
 - Sections
 - Explanation
 - Clean outputs
- Remove duplicate examples

- Commit: `Week 4 refactor and cleanup`
-

Day 22 (Sunday) – Backlog + Reflection

Steps

- 90–120 min backlog study (highest priority)
 - Reattempt 2 hardest LC problems
 - Write reflection in README:
 - What was hardest this month?
 - What improved most?
-

WEEK 5 (Days 23–30) – Consolidation + Mini Project

Goal: Prove you can build something small independently

Day 23 – Mini Project Planning

Steps

- Choose one:
 - CLI Calculator
 - Student Marks Manager
 - Expense Tracker (CLI)
 - Write plan in notebook:
 - Features
 - Inputs/outputs
 - LC: Valid Parentheses
-

Day 24 – Project v1

- Implement basic version (works but messy)
 - LC: Intersection of Two Arrays
-

Day 25 – Project v2

- Add functions
 - Add input validation
 - LC: Plus One
-

Day 26 – Project v3

- Add file saving/loading
 - LC: Happy Number
-

Day 27 – Project Cleanup

- Refactor project code
 - Add comments + docstrings
 - LC: Ugly Number
-

Day 28 – Project Documentation

- Add project section to README
 - Explain:
 - What it does
 - How to run
 - Example usage
 - LC: Remove Duplicates from Sorted Array
-

Day 29 (Saturday) – Final Polish

- Review entire repo
 - Ensure commit history looks consistent
 - Fix formatting everywhere
-

Day 30 (Sunday) – Final Review + Backlog

- Backlog study
- Month review in README:

- Skills gained
 - Weak areas
 - Goals for Month 2
-

End of Month Checklist (Strict)

- GitHub repo looks professional
 - 3–5 clean notebooks
 - 1 small project included
 - 15–20 LC Easy solved
 - README readable by strangers
 - Daily commit habit formed
-

Optional Enrichment (only if ahead of schedule)

- Learn `pip freeze > requirements.txt`
- Learn difference: `venv` vs `virtualenv`
- Explore `argparse` for CLI programs