📘 Course 1: Programming Logic & Algorithms with Python

Course Overview

🎯 Course Title

Programming Logic & Algorithms with Python

🎓 Course Description

This course builds the **foundational thinking systems** required to become a professional software engineer and computer scientist. Students will master:

- Computational thinking
- Logic & problem decomposition
- Flowcharts & pseudocode
- Python programming fundamentals
- Algorithmic thinking (loops, conditions, functions)
- Debugging & testing
- Data structures & file handling
- Time complexity fundamentals

This is not “learn Python basics quickly.” It is a **deep cognitive training program** designed to rewire students to think like expert programmers.

By the end, a student with zero programming experience will be able to:

- Design and implement Python programs
- Transform real-world problems into algorithms
- Debug, test, and optimize beginner-intermediate code
- Work independently on structured programming tasks

🧠 Skills You Will Gain

- ✅ Computational & algorithmic thinking
- ✅ Problem decomposition & abstraction
- ✅ Python programming fundamentals
- ✅ Debugging & testing techniques
- ✅ Time complexity intuition
- ✅ Beginner-level software engineering habits

🧩 Key Topics Covered

- Breaking problems into steps
- Flowcharts & pseudocode
- Variables, types, expressions
- Control flow: `if`, `for`, `while`
- Functions, parameters, return values
- Lists, tuples, dictionaries
- Basic file I/O
- Algorithm design patterns
- Error handling fundamentals
- Intro to Big-O efficiency thinking

🧰 Tools & Technologies

- **Python 3.x**
- VS Code / Jupyter Notebook
- Flowchart tools (draw.io / Mermaid)
- Git basics (Week 8+)

Course Format

Component	Delivery
Theory	Deep lectures + intuitive explanations
Practice	Weekly coding labs, exercises, & take-home tasks
Projects	Mini-projects + Final Capstone
Feedback	Solutions + debugging notes
Visualization	Flowcharts, diagrams, memory traces

17 Weekly Breakdown (High-Level Overview — Detailed per week later)

Week	Theme
1	Thinking like a computer scientist
2	Variables, memory, data types
3	Expressions & control flow
4	Loops & iteration
5	Functions & modular design
6	Lists, tuples, strings
7	Dictionaries & sets
8	Algorithm patterns + Mini-project
9	Debugging & testing
10	File I/O & formats
11	Searching & sorting basics
12	Complexity basics + Mini-project
13	Problem-solving systems
14	Applied algorithm building
15	Capstone development
16	Capstone completion + assessment

📦 Expected Outputs

At the end of this course the student will have:

- ✅ 200+ solved exercises
- ✅ 48 take-home homework tasks
- ✅ Flowcharts & pseudocode sets
- ✅ 3 mini-projects
- ✅ 1 final capstone
- ✅ Strong Python + logic foundation

🧪 Assessment

Component	Weight
Weekly exercises	30%
Mini-projects	20%
Capstone project	40%
Reflection + understanding checks	10%

👨‍🎓 Who This Course Is For

- Beginners with zero coding experience
- Self-learners preparing for CS degree or bootcamp
- Students learning algorithmic thinking before AI/ML

🚫 Who This Course Is NOT For

- People seeking “just syntax” tutorials
- People unwilling to solve many exercises

This course builds **thinkers, not typists**.

📘 Required Materials

- Laptop
- Python installed
- VS Code
- Internet access for docs

🎯 Learning Outcomes

By end of the course students will be able to:

- ✅ Take a real-world problem → write algorithm → code solution
- ✅ Decide when to use loops, functions, or data structures
- ✅ Debug, test, and improve code
- ✅ Understand time-efficiency trade-offs
