



BUILDER • CORE PYTHON & SOFTWARE THINKING

Build Bigger. Think Like a Developer.

6 Months • Weekly 60-Minute LIVE Class • Small Skill-Based Cohorts

Designed for students who know the basics and are ready to build larger programs, solve harder problems and work more independently.

100% LIVE

CODING GYM

PROJECT-BASED

DEVELOPER MINDSET

Strong Python skills are essential to building with AI — not just using it.

Core Python → structured programs → APIs & automation → AI-ready development.



PROGRAM OVERVIEW

Six months. From coder to confident Builder.

24 LIVE CLASSES

Approximately 24 weekly sessions
60 minutes each
Hands-on coding every week
Real-time instructor guidance

CORE PYTHON

Functions → collections → files → modules →
OOP basics → APIs → structured applications

Students move beyond isolated exercises.

LEARN BY BUILDING

Coding Gym • Debugging Gym • utilities •
data apps • API projects • automation •
capstone

Every concept becomes part of a larger
system.

OUTCOME

Students finish able to design, structure, debug and extend multi-feature Python programs with growing independence.



MONTHS 1-2

CORE PYTHON + STRUCTURED THINKING

01 • FUNCTIONS + PROGRAM DESIGN

functions • parameters • return values • scope • decomposition • reusable code

Coding Gym: refactor small scripts into clean functions

02 • COLLECTIONS + ALGORITHMS

lists • dictionaries • nested data • searching • sorting concepts • validation

Build: score systems, trackers and menu-driven apps

GOAL Students stop thinking line-by-line and begin designing programs as reusable pieces that work together.



MONTHS 3–4

FILES + OOP BASICS + DEBUGGING GYM

03 • FILES + DATA PERSISTENCE

read/write files • CSV/JSON concepts • exceptions • validation • persistent data

Build: apps that remember data between runs

04 • OOP + PROGRAM STRUCTURE

classes • objects • attributes • methods • modeling real things • modules

Build: structured multi-file applications



DEBUGGING GYM

Trace the Bug • isolate the cause • use print/logging strategically • read tracebacks • live debugging • repair broken multi-function programs

AI challenge: verify and fix generated code instead of trusting it.

GOAL Build programs that persist data, use clear structure and can be debugged systematically instead of by guessing.



MONTHS 5–6

APIs + AUTOMATION + REAL APPLICATIONS

05 • APIs + EXTERNAL DATA

HTTP/API concepts • requests • JSON • parsing responses • error handling • API keys & safe habits

Build programs that connect to real services and data.

06 • AUTOMATION + BUILDER CAPSTONE

automation workflows • multi-feature design • GitHub workflow • documentation • testing mindset • final Builder capstone

Plan → design → build → debug → document → present.

GOAL Combine core Python, APIs, files and structured design into a useful application the student can defend and extend.



LEARNING BY BUILDING

Not just exercises — Builders create useful Python systems.

Data & Tracker Apps

collections • files • validation

API-Powered Apps

requests • JSON • error handling

Automation Tools

files • functions • workflows

Structured Apps

modules • classes • persistence

Coding + Debugging Gym

reasoning + refactoring + diagnosis

Builder Capstone

design • build • test • document

GitHub becomes part of the workflow: organize versions, document projects and present work like a developing software engineer.



BUILDER CAPSTONE

Real problems. Different Builder solutions.

SMART STUDY PLANNER

Assignments • priorities • file storage • summaries • modular design

Optional API extension for richer data.

API DATA DASHBOARD

Fetch live/public data • parse JSON • filter results • save favorites • handle failures

Great for learning real application flow.

MY OWN IDEA

Student proposes a useful app or automation, scopes it with Ankit, builds a realistic Builder version and documents it on GitHub.

The capstone proves the student can combine multiple concepts, make design decisions, debug independently and explain how the system works.



WHAT COMES NEXT

Builder is where coding starts becoming software development.

1 • EXPLORER

Foundation checkpoint
Python fundamentals
Coding + Debugging Gym
Explorer-level projects

2 • BUILDER

Core Python depth
Files + OOP basics
APIs + automation
Builder capstone

3 • CREATOR

Advanced builds
AI applications
Workflows + agents
Portfolio projects

Recommended placement: students should demonstrate Explorer-level fundamentals before Builder. Experienced students can take a readiness assessment for direct placement.

Know where your child stands in the age of AI → thinkpythonai.com/will-ai-take-my-kids-job



THINKPYTHONAI YOUNG CODERS

From “I can code” to “I can build software.”

BUILDER • 6 MONTHS • LIVE • SMALL SKILL-BASED COHORTS

**Weekly coding • Coding Gym • Debugging Gym • APIs • automation •
GitHub • final capstone**

Young Coders Coordinator: Crystal
+1 (650) 444-3020

THE BUILDER PROMISE

We move beyond syntax.

Students learn to structure larger programs.

They debug systematically, connect to data and APIs, and build with increasing independence.

Then they are ready for Creator-level Python and AI applications.

Strong Python skills are essential to building with AI — not just using it.