4-MONTH GUIDED PROGRAM

PROGRAMMING & AI

From your first line of code to your first working product.

Learn Python, analyse data, understand AI and build useful web applications with live guidance in your browser.

01	02	03	04
CODE	DATA	AI	PRODUCT

4 months	6 months	Scholarships
Guided learning	Cloud Labs access	Available on review

Beginner-friendly. Practical. Built around understanding.

ASHISH CHAUDHARY
Founder and Instructor

AI EDUCATION FOR EVERYONE

WHO THIS IS FOR

Start where you are. Build toward your goal.

No previous coding experience is needed to begin. The program starts with the tools and gradually connects them to meaningful work.

01

Students and MBA applicants

Build projects you can explain in a college review or a relevant application discussion. Use evidence to compare business choices, discuss trade-offs and show how your reasoning developed.

02

Entrepreneurs and aspiring founders

Move beyond describing an idea. Learn how pages, forms, data and users fit together, then shape a small working product around one customer problem.

03

Working professionals

Develop practical Python-based analysis, reporting and software skills. Explore how repeatable calculations and interactive tools could improve a familiar workflow.

04

Self-taught learners and career explorers

Connect scattered tutorials into a coherent path. Practise reading code, debugging, saving versions and explaining why a solution works.

Your background is a starting point, not a barrier.

Bring a problem you care about, a willingness to practise and time to ask questions. Advanced mathematics is not an entry requirement; relevant concepts are introduced through the work.

Project work can strengthen how you present your learning. It does not guarantee MBA admission, employment, income or business growth.

Curriculum: six-phase learning path >

THE CURRICULUM AT A GLANCE

Six phases.
One connected learning path.

Four months of guided study, progressing from running code to a small application another person can use.

0 **Foundation**

Code, terminal commands, Git and GitHub

I can run code and keep track of changes.

1 **Thinking Like a Programmer**

Python, data structures, NumPy, pandas and visualisation

I can write programs and investigate a dataset.

2 **Building Software**

Text processing, web data, retrieval and Streamlit

I can combine components into a useful tool.

3 **Teaching Computers to Learn**

Machine learning, neural networks, images and text

I can evaluate a model instead of trusting a score.

4 **Build Web Applications**

Django, HTML, CSS, JavaScript, forms and CRUD

I can connect user actions to a working interface.

5 **Make Applications Real**

Databases, SQL, authentication and deployment

I can demonstrate a focused minimum working product.

Progress through the phases, rather than race through a tool list.

The cohort calendar, class timings and pace are confirmed during application review. Project depth is matched to preparation and practice; selected advanced examples extend the core path.

Curriculum: six-phase learning path >

PHASE 0 / FOUNDATION

Make the tools feel familiar.

Begin with a simple question: what happens when you give a computer an instruction?

Understand what code is

Run a small example, change an input and inspect the output. Learn the difference between the file containing your instructions and the place where you run them.

Use the terminal without the mystery

Navigate folders, locate files and run commands deliberately. Learn what a working directory means and how to investigate an error rather than repeatedly guessing.

Keep your work with Git and GitHub

Create a repository, save meaningful commits and understand how version history records a project. Practise explaining what changed and why before sharing your work.

A REPEATABLE LEARNING HABIT

READ > RUN > CHANGE > CHECK > SAVE

What you keep

A small Python script, a short README and a sequence of Git checkpoints that show how your work developed.

Why it matters

You are learning to investigate, recover and explain. Those habits support every later dashboard, model and website.

Curriculum: Foundation >

PHASE 1 / THINKING LIKE A PROGRAMMER

Learn Python.
Make sense of data.

Build the programming foundation before expecting AI-generated code or a dashboard to do the thinking for you.

Write clear instructions

Variables and data types; type conversion; arithmetic and comparisons; strings and formatted output.

Organise information

Lists, tuples, dictionaries and sets. Choose a useful structure, access values and combine related information.

Express decisions and repetition

Conditions, loops, functions and arguments. Introduce classes to organise related data and behaviour.

Work with numerical data

Use NumPy arrays to represent and calculate with numerical values instead of handling every value manually.

Explore tables with pandas

Read CSV files, inspect types and missing values, filter records, group observations and calculate summaries.

Explain patterns visually

Use Matplotlib and Seaborn to compare categories, inspect distributions and explore relationships between measures.

Example question: are higher sales also leaving more gross profit?

Read the table, establish what one row represents, calculate sales and product cost, then compare categories. Explain what your chart shows and which information is still missing.

Portfolio evidence

Keep a readable notebook or analysis script with the question, data checks, calculations, labelled charts and a short interpretation. A reproducible explanation matters more than decorative charts.

Curriculum: Python and exploratory data analysis >

PHASE 2 / BUILDING SOFTWARE

Turn small instructions into useful tools.

Combine text handling, information retrieval and a Python interface into something another person can try.

Extract information from text and webpages

Learn regular expressions for text patterns and Beautiful Soup for reading webpage structure. Practise selecting relevant content instead of treating a webpage as an unstructured wall of text.

Understand how information can be found

Explore embedding vectors as numerical representations and work through progressively richer chatbot examples. Learn to distinguish finding relevant information from generating a new answer.

Create an interface with Streamlit

Connect Python logic to inputs, tables, charts and results. The purpose is a usable experience: someone asks a question or selects data, and the application shows a meaningful response.

Featured reference: CookieBot, a website information assistant

A visitor asks a question about a website. The app reads a bounded set of pages, cleans the text and ranks passages with BM25, a text-retrieval method. It displays a selected passage and its source URL.

Your learning challenge

Adapt the content and questions for an appropriate, permitted source. Test questions it answers well and questions it handles poorly. Document the source, retrieval limits and a possible improvement.

This reference is a retrieval assistant, not a custom-trained language model. Use only data and websites you are permitted to access.

Curriculum: Building Software >

PHASE 3 / MACHINE LEARNING

Learn from examples. Then test what was learned.

Machine learning adds another way to solve a problem: fit a model to data, then check how it behaves on examples it did not train on.

REGRESSION

Estimate a numerical value

The housing example connects exploratory analysis with predictive modelling. Investigate which inputs are useful, compare predictions with actual values and discuss prediction error.

CLASSIFICATION

Distinguish between categories

The telecom churn notebook asks whether a customer is labelled as leaving the service. Work through cleaning, feature encoding, train/test splitting, model fitting and evaluation.

CLUSTERING

Explore groups without supplied labels

Use the clustering material to see how similarity can organise observations. Inspect the groups and explain whether they are useful, rather than treating every cluster as a business fact.

The habit that matters: separate learning from evaluation.

Understand train/test splitting, overfitting and underfitting. Use model reports and a confusion matrix where relevant, and explain the mistakes as well as the successes.

The telecom prediction exercise is different from the descriptive churn dashboard on page 13. Neither proves why a customer left, and no model accuracy or commercial result is guaranteed.

Curriculum: Teaching Computers to Learn >

PHASE 3 / DEEP LEARNING AND SELECTED EXTENSIONS

See how AI works with images and language.

Move from tables to pixels and text, using manageable learning examples rather than treating a model as a black box.

Handwritten digits

Explore MNIST digit images and a neural-network training example. Learn how an image becomes numerical input and how a predicted class is checked.

Image categories

The CIFAR-10 CNN example classifies images into ten categories. Inspect training curves, held-out results and the confusion matrix to understand errors.

Transfer learning

The cats-versus-dogs example uses VGG16. Explore how an existing visual model can support a narrower classification task and what preprocessing it expects.

Text and sentiment

Review the NLP and IMDB material: prepare text, represent inputs, train a review classifier and inspect results. A sentiment label is an estimate, not a fact about a person.

Additional exploration

OpenCV lessons introduce image manipulation and thresholding. Embeddings and other NLP tutorials offer further ways to connect numerical representations with real inputs.

Depth follows readiness and the chosen project.

These are selected advanced learning examples, not a promise to train every model in one cohort. Some exercises need external datasets or model downloads; compute access depends on the assigned lab.

Curriculum: Teaching Computers to Learn >

PHASE 4 / BUILD WEB APPLICATIONS

Give your software a usable interface.

Build with Python and Django, then add the HTML, CSS and JavaScript that help people understand and interact with it.

01 **First Django website**

Connect a URL, a view and a template. Understand how a request becomes a page.

02 **Business and dynamic websites**

Organise multiple pages and display changing information rather than hard-coding every result.

03 **Forms and interaction**

Collect user input, validate it and provide useful feedback when something is missing or invalid.

04 **JavaScript in the browser**

Explore browser-side behaviour and the difference between frontend actions and server-side processing.

05 **CRUD and a mini product**

Create, read, update and delete records, then combine the pieces into a small application around one purpose.

Example adaptation: a founder website that does more than introduce you.

Present your offer, show project demonstrations and let a visitor submit a structured enquiry. The learning goal is to connect a real user action to a meaningful response.

The web phase uses Django with HTML, CSS and JavaScript. Independent reference projects are taught incrementally; your own project grows through changes you can explain.

Curriculum: Build Web Applications >

PHASE 5 / MAKE APPLICATIONS REAL

Add the pieces real users need.

A working page is a beginning. A useful product also needs stored information, appropriate access and a reliable way to run.

Databases and persistence

Understand how application models map to stored records and why data must remain available after a request ends.

Practical SQL

Use database questions to learn how to inspect, select and work with data. Connect a query to the application problem it answers.

Users and authentication

Add registration and login where the product needs them. Learn why a user should see their own records, not everyone else's.

Production configuration

Separate configuration from code. Understand environment settings, secrets, allowed hosts and static files before exposing an application.

Deployment

Follow the project's deployment workflow and verify the core experience in the target environment. A development server is not a production plan.

A minimum working product

Connect one user need to one complete flow. Refine the Phase 4 idea instead of repeatedly starting a larger, unfinished application.

The reference product is deliberately small.

The final example is a service-booking application with registration, services, booking and confirmation. You learn a reusable structure, then adapt the user, content and data model to your own idea.

Deployment learning is included in the curriculum. Public hosting, domains, third-party services and their costs are separate arrangements to confirm for your project.

Curriculum: Make Applications Real >

FEATURED PRODUCT / SERVICE BOOKING

From scattered messages to one clear customer journey.

Imagine offering consultations or tutoring. Requests arrive in different chats, and it is hard for a customer to see what happens next.

The product question

Can a visitor find a service, submit a booking and see confirmation? Start with that complete journey before adding payments, notifications or a large feature list.

1	2	3	4
ACCOUNT	SERVICE	BOOKING	CONFIRM
Register or sign in	Choose a service	Submit a request	Review the result

What the reference includes

Service list and detail pages, a booking form, user-specific bookings, a local SQLite database and environment-based deployment configuration.

What you personalise

The user problem, service information, form fields and content. A consultant, tutor or campus club could adapt the same pattern to a different small workflow.

What you demonstrate

Show one complete journey. Explain where the information is stored, which user can access it and what happens when input is invalid. Keep setup instructions and a list of known limits.

A prototype that someone can try, not a finished business.

The goal is a minimum working product. Payments, scheduling automation, enterprise-scale security and other advanced features are later decisions, not implied features of this reference.

Reference project: service-booking minimum working product >

BUSINESS PROJECT LIBRARY / 1 OF 3

Follow the money. Ask a better question.

Guided Streamlit practice, selected to fit your goals. All data is fictional.

01 Retail sales & profit

Higher sales are not automatically healthier sales.

A retail manager must decide where to focus promotions or stock. Compare regions, categories and months before recommending the next investigation.

WHAT YOU BUILD

Use order-level quantity, prices, discounts and product costs. Customise net-sales, gross-profit, margin and average-order cards with charts and filters.

ILLUSTRATIVE EXAMPLE

Two INR 1,000 items, a 10% discount and INR 600 unit cost produce INR 1,800 sales and INR 600 gross profit.

YOUR ANALYTICAL TAKEAWAY

Explain one opportunity with two observations and one missing fact. Gross profit excludes rent, salaries and other operating expenses.

02 Marketing campaigns

Which channel deserves a controlled budget test?

The team celebrates impressions and clicks, but the manager wants to understand conversion volume, spending and attributed results.

WHAT YOU BUILD

Compare campaign and channel filters, clicks, conversions, spend, cost per acquisition and return on ad spend. Calculate rates from totals, not an average of percentages.

ILLUSTRATIVE EXAMPLE

INR 5,000 spend and INR 20,000 attributed revenue give 4x ROAS. If there are 25 conversions, acquisition cost is INR 200.

YOUR ANALYTICAL TAKEAWAY

Propose a bounded campaign test and how to judge it. ROAS is not profit, and attribution alone does not prove the ad caused the sale.

Project library: six Streamlit business analytics cases >

BUSINESS PROJECT LIBRARY / 2 OF 3

Understand customers. Plan the next action.

Guided Streamlit practice, selected to fit your goals. All data is fictional.

03 Customer retention & churn

Who is leaving, and what should you investigate?

A subscription manager sees new sign-ups alongside departures. Explore customer groups before proposing the same retention offer for everyone.

WHAT YOU BUILD

Compare contract types, tenure, satisfaction and support-ticket patterns. Show group sizes alongside the proportion marked as churned.

ILLUSTRATIVE EXAMPLE

Twenty marked departures among 100 selected customers means a 20% snapshot proportion. The same percentage can describe very different sample sizes.

YOUR ANALYTICAL TAKEAWAY

Write a segment investigation with a specific unanswered question. This is descriptive analysis, not a monthly churn rate or the predictive model on page 7.

04 Inventory planning

Enough stock in one place. Too much in another.

A warehouse manager needs to balance availability against money tied up in products. Buying more of everything does not solve the problem.

WHAT YOU BUILD

Compare stock on hand, assumed daily demand, replenishment lead time, safety stock and inventory value. Explore stock cover and an extra buffer scenario.

ILLUSTRATIVE EXAMPLE

A stock level of 100 units at an assumed demand of 10 per day gives 10 days of cover. A 12-day replenishment time deserves attention.

YOUR ANALYTICAL TAKEAWAY

Produce a prioritised stock-review note. Check demand and delivery assumptions before ordering; the scenario is not a guaranteed forecast.

Project library: six Streamlit business analytics cases >

BUSINESS PROJECT LIBRARY / 3 OF 3

Bring everyday operations into the picture.

Guided Streamlit practice, selected to fit your goals. All data is fictional.

05 Finance & budget analysis

Is lower spending a saving or unfinished work?

A department head needs to explain where expenses differ from the plan. A red number is the start of a question, not proof of poor performance.

WHAT YOU BUILD

Compare monthly budgets and actual expenses by department and expense category. Use utilisation, variance and cumulative trends to locate differences.

ILLUSTRATIVE EXAMPLE

Budget INR 50,000 minus actual expense INR 56,000 gives variance of -INR 6,000. In this project, negative variance means overspend.

YOUR ANALYTICAL TAKEAWAY

Prepare a short budget-review brief that connects the difference to an investigation. This is an expense report, not a cash-flow statement or a profit calculation.

06 Restaurant operations

More orders, but what happens to service?

A restaurant manager sees strong sales and customer complaints. Explore whether waiting times or food-cost contribution vary by branch, meal period or channel.

WHAT YOU BUILD

Use orders, bill amounts, food costs, waiting times and ratings. Compare like-for-like segments and inspect whether a problem is concentrated in a shift.

ILLUSTRATIVE EXAMPLE

A bill of INR 1,000 with INR 350 food cost leaves INR 650 before staffing, rent, delivery fees, tax and other expenses.

YOUR ANALYTICAL TAKEAWAY

Recommend a small staffing or workflow test and a way to evaluate it. A relationship between waiting and ratings does not establish causation.

Project library: six Streamlit business analytics cases >

MAKE THE WORK YOUR OWN

A project is stronger when you can explain it.

Build a collection of completed work around your goals, then give it a clear home in a personal resume, portfolio or founder website.

For students and MBA applicants

Turn a project into a case study: the problem, the data, alternatives considered, evidence, decision and limitations. Use it to discuss your interests and initiative where relevant, without presenting a teaching exercise as paid client work.

For entrepreneurs

Use the website skills to present your founder story, offer, demonstrations and contact route. Link to a working prototype so a prospective collaborator or customer can see the idea rather than only read about it.

For working professionals

Document a reporting or workflow problem, the analysis you performed and a reproducible result. Use approved or anonymised data; do not place confidential workplace records in a public portfolio.

A USEFUL CASE STUDY CONTAINS

QUESTION / METHOD / EVIDENCE

RESULT / LIMITATIONS / NEXT STEP

Keep code, a README, meaningful commits and a short demo.

Credit reference material and AI assistance. Claim only the work you completed and tested. Public portfolio hosting is separate from temporary, private Cloud Labs access.

Curriculum: Make Applications Real >

THE LEARNING EXPERIENCE

Four months of guidance. Six months of Cloud Labs.

A prepared browser workspace supports the taught program and gives you additional time to revisit, improve and finish your work.

SIX-MONTH ACCESS WINDOW

Month 1	Month 2	Month 3	Month 4	Month 5	Month 6

Months 1-4: guided program **Months 5-6: continued lab practice**

Code in your browser

Use VS Code for files and applications, and JupyterLab for notebooks and experiments. Prepared environments reduce local setup work. Keep projects in your workspace and save independent copies with Git.

Learn with a person, not only a playlist

Join live sessions in our virtual classroom, ask questions and work through examples. Practise between sessions, make small changes and return with the point you want to understand.

Use AI as an assistant you can check

Ask for explanations, a small implementation or help interpreting an error. Run the result, compare it with the requirement and test edge cases. Never submit secrets or confidential records to an AI tool.

Access and revision

Six months means total Cloud Labs access, not six extra months after the program. Exact activation and expiry dates are confirmed at enrolment. Class recordings are available for seven days; keep your own notes and code.

Bring a laptop or desktop, a modern browser and stable internet. Lab compute is shared and assigned for learning; ongoing live mentoring, public hosting and third-party API credits are not implied by lab access.

Explore the free learning material >

FEES AND SCHOLARSHIP CONSIDERATION

Invest in the guidance. Ask about support.

The published learning material remains free. This offering adds four months of live guidance and the managed learning experience described here.

PROGRAM FEE / BEFORE GST

₹79,999

+ applicable GST | Six months of Cloud Labs included

Scholarships are available

Request consideration when you apply. Tell us about your background, learning goals, achievements or early experiments and why this program would help you take the next step.

Each application is personally reviewed

We discuss your readiness, goals and the kind of project you hope to build. A previous coding portfolio is not required to begin. Scholarship requests are considered individually, subject to the available support and applicable criteria.

A clear decision before you enrol.

Any approved scholarship, its conditions and the final payable amount will be confirmed before enrolment. Requesting consideration does not guarantee an award or a particular amount.

Choose from the available cohort timings

Share your preferred time slots and contact us to discuss a suitable batch. We will confirm the timetable, class frequency, fees, payment terms and access dates before you commit.

Ask for the upcoming cohort calendar and weekly practice expectations. Any eligible alumni benefit, and whether it can be combined with other support, will be confirmed individually.

YOUR NEXT STEP

Tell us what you want to build.

Let us discuss your goals and a project that fits.

Can I join without a technical background?

Yes. The sequence starts with running code and Python foundations. Consistent practice matters more than arriving with a list of technologies.

Will every project need AI?

No. You learn how AI works so you can decide when it adds value. A straightforward form, database or calculation can be the better solution.

What should I have by the end?

Aim for working exercises, documented analysis and a focused product you can demonstrate. The exact portfolio reflects your chosen scope, practice and completed work.

What remains available after the teaching ends?

Cloud Labs access lasts six months in total. Save your source files, data you may retain, notes and Git history before access ends. The public learning material remains free.

YOUR APPLICATION, PERSONALLY REVIEWED

Share your background, goal, preferred time slots and any scholarship request.

Ashish Chaudhary | Founder and Instructor

+91 9319619889 **ashish@cookiesensei.com**

Explore the curriculum **Start at cookiesensei.com**

Educational program. No guarantee of admission, employment, funding or business success. Curriculum reference: programming-and-ai, main c10a61c (October 2026). Examples and depth may be adapted to the cohort.

Curriculum: six-phase learning path >