PK

LEARN CODING IN 2026

# Your 12-Week AI-Powered Guide

***"Stop watching tutorials. Start building."***

Go from zero to building full-stack web apps in 12 weeks using free AI coding assistants. No expensive courses. No passive video watching. Just real projects, real code, and real understanding.

**Python • HTML & CSS • Bootstrap • JavaScript • Flask • SQLite • OpenAI API**

**By Praneeth Kalluri**

Instagram: @praneeth_kalluri

`https://www.instagram.com/praneeth_kalluri/`

`LinkedIn: https://www.linkedin.com/in/praneethkalluri/`

2026 Edition

# Introduction

## The Method: Build to Learn

You don't need YouTube tutorials or expensive courses. In 2026, the fastest way to learn coding is by working directly with an AI coding assistant. You give it a prompt describing what you want to build. It writes the code. Then you study every line, ask deep questions, and truly understand what's happening. This isn't copy-paste coding — it's **guided discovery**.

In 12 weeks, you'll go from writing your very first line of code to building and deploying a full-stack web application with AI integration. Every week has a real project you can show employers, friends, or put on your portfolio.

## What You'll Learn

- **Python** — the most beginner-friendly programming language
- **HTML & CSS** — how every website is built
- **Bootstrap** — make websites look professional fast
- **JavaScript** — make websites interactive and dynamic
- **Flask** — build your own backend server with Python
- **SQLite** — store and retrieve data with databases
- **OpenAI API** — integrate AI into your own apps
- **Deployment** — put your app live on the internet

## Tools You Need (All Free)

Pick **any one** of these free AI coding assistants to get started. All of them have a free tier — you do not need to pay anything:

### Option 1: Antigravity by Google (Free Preview) — Recommended

`https://antigravity.google`

Google's agent-first IDE powered by Gemini. Currently free during public preview. Sign in with your Google account and start coding with AI agents that can plan and execute tasks. Best option for absolute beginners.

### Option 2: GitHub Copilot (Free Tier) — Recommended

`https://github.com/features/copilot/plans`

Built by GitHub (owned by Microsoft). Works inside VS Code. Sign up with a free GitHub account and select the free tier. 2,000 code completions and 50 chat requests per month for free.

### Option 3: Cursor (Free Tier)

`https://cursor.sh`

An AI-first code editor built on VS Code. Download Cursor, create a free account, and start coding with AI chat and autocomplete. Free Hobby plan available.

### Option 4: Windsurf (Free Tier)

`https://windsurf.com`

An agentic AI IDE with a powerful Cascade assistant for coding, debugging, and refactoring. Download Windsurf, create a free account, and start prompting.

### Option 5: Cline (Free — Open Source)

`https://marketplace.visualstudio.com/items?itemName=saoudrizwan.claude-dev`

A free, open-source AI coding agent for VS Code. Install it from the VS Code Extensions tab. You bring your own AI API key (like OpenAI). Great for learning how AI tools work under the hood.

## How to Use This Guide

Each week follows the same simple pattern:

**1. Copy the Project Prompt** — Each week has an exact prompt in a dark box. Copy it word-for-word and paste it into your AI coding assistant.

**2. Build the Project** — The AI will generate code with detailed comments. Read every line. Follow the setup instructions.

**3. Ask the Follow-Up Questions** — After your project works, go through the question list. Paste each question into the AI and read its explanation carefully.

**4. Go Deeper or Keep It Light** — Want more depth? Ask the AI: *"Why does this work under the hood?"* or *"What happens if I change this?"* Want simpler explanations? Ask: *"Explain this to me like I'm 5."*

After 12 weeks, you'll be **interview-ready** with real projects to show on your resume and GitHub profile. Companies care about what you can build — not which course you watched.

Follow **@praneeth_kalluri** on Instagram and connect on LinkedIn for more AI-powered learning tips!
`https://www.linkedin.com/in/praneethkalluri/`

# Table of Contents



**Free AI Coding Tools:** Antigravity | Copilot | Cursor | Windsurf | Cline

See the Introduction page for download links to all 5 free tools.

**Follow for updates:** Instagram @praneeth_kalluri | LinkedIn: https://www.linkedin.com/in/praneethkalluri/

WEEK 1

# Simple Calculator

Timeline: 1–2 days

**■■ BEFORE YOU START**

This is your very first project! You don't need any previous coding experience. All you need is a computer (Windows, Mac, or Linux) and an internet connection. This week will guide you through installing Python and writing your first program from scratch.

## ■ PROJECT PROMPT

*Copy and paste this entire prompt into your AI coding assistant:*

```
I am a complete beginner who has never written a single line of code before. Please create a simple calculator program in Python that does the following: 1. Asks the user to enter the first number 2. Asks the user to pick an operation: add (+), subtract (-), multiply (*), or divide (/) 3. Asks the user to enter the second number 4. Shows the result 5. Asks if they want to do another calculation or quit IMPORTANT SETUP INSTRUCTIONS (I need these because I've never done this before): - First, check if I have Python installed by telling me to open my Terminal (Mac) or Command Prompt (Windows) and type: python --version - If I don't have Python, give me step-by-step instructions to download and install it from python.org - Show me exactly how to create a new file called calculator.py, where to save it, and how to open it in a text editor - Show me the exact command to run the file from the terminal CODE REQUIREMENTS: - Add a detailed comment above EVERY single line of code explaining what it does and WHY in plain English, as if explaining to someone who has never seen code before - Handle the case where the user tries to divide by zero (show a friendly error message) - Handle the case where the user types letters instead of numbers (show a friendly error message) - Use a while loop so the calculator keeps running until the user types 'quit' - Keep the code as simple and readable as possible — no advanced tricks, just clean beginner code
```

## ? QUESTIONS TO ASK YOUR AI

*After your project is working, paste these questions one at a time into your AI assistant:*

- Explain each line of the calculator code in detail. What does every single line do, and why is it written that way?

- What are variables in Python? How are they different from variables in math class? Can you show me 5 different examples of creating variables?
- What are data types? What is the difference between a string, an integer, and a float? Why does Python care about the difference? Show examples.
- What are conditionals (if, elif, else)? Walk me through exactly how Python decides which branch to take. Can you draw out the logic flow?
- What is a while loop? How does it know when to stop? What would happen if I forgot to add a way to break out of the loop?
- What does the input() function do behind the scenes? Why do we need to convert it with int() or float()?
- What is error handling? What does try-except do, and why is it important for real programs? What would happen without it?
- How would I add more operations like power (**), modulus (%), and square root? Walk me through adding each one.
- What is the terminal/command prompt? Why do we run Python files from it instead of just double-clicking the file?
- What are the 10 most common beginner mistakes in Python, and how do I avoid them?

## ✓ CONCEPTS YOU LEARNED

[ Variables ] [ Data types ] [ User input ] [ Conditionals (if/elif/else) ] [ While loops ] [ Basic operators ] [ Error handling (try/except) ] [ Terminal basics ]

---

WEEK 2

# To-Do List App

Timeline: 2–3 days

**■■ BEFORE YOU START**

You should have Python installed and working from Week 1. Open the same folder where you saved your calculator.py file. This week you will create a new file called todo.py in that same folder. You will build on the Python skills you learned (variables, loops, if/else) and learn new concepts like lists, dictionaries, and functions.

## ■ PROJECT PROMPT

*Copy and paste this entire prompt into your AI coding assistant:*

```
I am a beginner learning Python (I just built a simple calculator last week).
Please create a to-do list application in Python that does the following: 1. Shows
a menu with options: (1) Add Task, (2) View All Tasks, (3) Mark Task as Complete,
(4) Delete Task, (5) Quit 2. When adding a task, ask for the task name and save it
with a status of 'incomplete' 3. When viewing tasks, show each task with a number,
name, and status (incomplete or complete) in a nicely formatted way 4. When
marking complete, show all tasks and let the user pick which one to mark done 5.
When deleting, show all tasks and let the user pick which one to delete 6. Save
all tasks to a text file called 'tasks.txt' so they are still there when I close
and reopen the program SETUP INSTRUCTIONS: - Show me how to create a new file
called todo.py in the same folder as my calculator - Show me the exact command to
run it - Explain that no extra installations are needed for this project CODE
REQUIREMENTS: - Add a detailed comment above EVERY single line explaining what it
does in plain English - Organize the code into separate functions for each action
(add, view, complete, delete, save, load) - Use a list of dictionaries to store
tasks (each dictionary has 'name' and 'status' keys) - Handle edge cases: what if
there are no tasks? What if the user enters an invalid number? - Make the output
look clean and organized with proper spacing and formatting
```

## ? QUESTIONS TO ASK YOUR AI

*After your project is working, paste these questions one at a time into your AI assistant:*

- Explain each line of the to-do list code in detail. Walk me through what happens from the moment I run the program to when I quit.

- What are lists in Python? How do I add, remove, and access items? Show me 10 different things I can do with lists.
- What are dictionaries? Why did we use a list of dictionaries instead of just a list of strings? When should I use each one?
- What are functions? Why should I organize code into functions instead of writing everything in one block? What are parameters and return values?
- How does file reading and writing work in Python? What does 'open', 'read', 'write', and 'close' do? What is the 'with' statement and why is it better?
- What is CRUD? How do the four operations in our app (add, view, complete, delete) map to CRUD?
- What are for loops vs while loops? When should I use each one? Show me examples of both with our task list.
- What does enumerate() do and why is it useful when looping through lists?
- What happens to the data in my program when I close it? Why do we need to save to a file? What are other ways to store data?
- How would I add features like: task priorities (high/medium/low), due dates, and categories? Walk me through the changes needed.

## ✓ CONCEPTS YOU LEARNED

[ Lists ] [ Dictionaries ] [ For loops ] [ Functions ] [ Parameters & return values ] [ File I/O (read/write) ] [ CRUD operations ] [ String formatting ] [ Data persistence ]

---

WEEK 3

# Contact Book with Search

Timeline: 3–4 days

**■■ BEFORE YOU START**

You should have your calculator.py and todo.py files from previous weeks. Create a new file called contact_book.py in the same folder. This week builds on everything you have learned so far (variables, loops, functions, lists, dictionaries) and introduces JSON for saving data in a more structured way.

## ■ PROJECT PROMPT

*Copy and paste this entire prompt into your AI coding assistant:*

```
I am a beginner learning Python (I've built a calculator and a to-do list app).
Please create a contact book application in Python that does the following: 1.
Shows a menu: (1) Add Contact, (2) View All Contacts, (3) Search Contacts by Name,
(4) Edit a Contact, (5) Delete a Contact, (6) Quit 2. Each contact should store:
first name, last name, phone number, email address, and notes 3. When searching,
let the user type part of a name and show all contacts that match
(case-insensitive) 4. When editing, show the current value of each field and let
the user press Enter to keep it or type a new value 5. Save all contacts to a file
called 'contacts.json' using JSON format 6. When viewing contacts, display them in
a nicely formatted table-like layout SETUP INSTRUCTIONS: - Show me how to create
contact_book.py - Explain what JSON is in simple terms before we use it - No extra
installations needed — json is built into Python - Show the exact command to run
the file CODE REQUIREMENTS: - Add a detailed comment above EVERY single line
explaining what it does in plain English - Use functions for every operation -
Validate email format (must contain @ and a dot) - Validate phone number (must be
digits only and at least 10 characters) - Handle all edge cases with friendly
error messages - Use try-except blocks for file operations and user input - Make
the search work even if the user types in lowercase and the name is in uppercase
```

## ? QUESTIONS TO ASK YOUR AI

*After your project is working, paste these questions one at a time into your AI assistant:*

- Explain each line of the contact book code in detail. Trace through what happens when I add a contact, search for one, and edit one.

- What is JSON? Why is it better than a plain text file for storing structured data? Show me what the contacts.json file looks like inside.
- What are nested dictionaries and lists of dictionaries? How do we navigate through complex data structures? Show examples.
- How does searching through data work in Python? What is the 'in' keyword? How does the .lower() method help with case-insensitive search?
- What is data validation? Why should we check if an email has @ or if a phone number has only digits? What could go wrong without validation?
- What is error handling in depth? Explain try, except, else, and finally. When should I use each one? Show real examples.
- What is the difference between reading a file as text vs reading it as JSON? How does json.load() and json.dump() work internally?
- What are string methods in Python? Show me the 15 most useful ones (like .strip(), .lower(), .split(), .replace(), etc.) with examples.
- What is scope in Python? Why can't a variable created inside a function be used outside it? What are local vs global variables?
- How would I add an 'export to CSV' feature so I can open my contacts in Excel? Walk me through it step by step.

## ✓ CONCEPTS YOU LEARNED

[ JSON data format ] [ Nested data structures ] [ Search algorithms ] [ Data validation ] [ Error handling (try/except/else/finally) ] [ String methods ] [ Variable scope ] [ Import statements ]

---

WEEK 4

# Personal Profile Page

Timeline: 2–3 days

### ■■ BEFORE YOU START

Big switch this week! You are moving from Python (terminal programs) to web development (things you see in a browser). You do NOT need any of your previous Python files for this project. Create a brand new file called profile.html. You will open this file directly in your web browser (Chrome, Firefox, etc.) instead of running it in the terminal.

## ■ PROJECT PROMPT

*Copy and paste this entire prompt into your AI coding assistant:*

```
I am a beginner who has been learning Python for 3 weeks and now I want to learn
web development. I have never created a website before. Please create a personal
profile webpage using only HTML and CSS that includes: 1. A profile photo section
(use a placeholder image from https://placehold.co/200x200 for now) 2. My name in
a large heading 3. A short bio paragraph about me 4. A 'Skills' section that shows
skills as styled tags/badges 5. An 'About Me' section with 2-3 paragraphs 6. A
'Contact' section with email and social media links (use # as placeholder links)
7. A clean footer at the bottom SETUP INSTRUCTIONS (I've never made a webpage
before): - Explain what HTML and CSS are in simple terms before we start - Show me
how to create a file called profile.html (explain that we put HTML and CSS in the
same file using a style tag) - Show me how to open it in my web browser (just
double-click the file or drag it into Chrome) - Explain that I don't need to
install anything — the browser IS the tool CODE REQUIREMENTS: - Put ALL CSS inside
a <style> tag in the HTML file (one single file, no separate CSS file) - Add a
detailed comment above EVERY HTML tag explaining what it is and what it does - Add
a detailed comment above EVERY CSS rule explaining what property it changes and
why - Use a modern, clean color scheme (suggest specific hex colors) - Use Google
Fonts (show me how to include one via a link tag) - Center the content on the page
with a max-width container - Make the profile photo circular using CSS - Add
subtle hover effects on the skill badges and links - Make sure it looks good on
both desktop and mobile (basic responsiveness)
```

## ? QUESTIONS TO ASK YOUR AI

*After your project is working, paste these questions one at a time into your AI assistant:*

- Explain each line of the HTML and CSS code in detail. What does every single tag and CSS property do?
- What is HTML? What are tags, elements, and attributes? Walk me through the anatomy of an HTML tag with 5 different examples.
- What is CSS? How does it know which HTML element to style? Explain selectors (tag, class, ID) with examples for each.
- What is the CSS box model? Draw it out and explain content, padding, border, and margin. Why does every web developer need to understand this?
- What are divs, sections, headers, and footers? When should I use each one? What is semantic HTML and why does it matter?
- How do colors work in CSS? What are hex codes, RGB, and HSL? How do I pick colors that look good together? Suggest 5 color palettes.
- How do fonts work on the web? What are Google Fonts? What is the difference between serif and sans-serif? What are good font combinations?
- What is the display property? Explain block, inline, inline-block, and flex with visual examples of each.
- What does 'responsive' mean? Why does my website need to look good on phones? How do media queries work? Show me a simple example.
- How do I add hover effects, transitions, and simple animations in CSS? Show me 5 different hover effects I can use.
- What is the difference between margin and padding? When should I use each one? Show visual examples.
- How do I use Chrome DevTools to inspect and debug my webpage? What are the most useful features for a beginner?

## ✓ CONCEPTS YOU LEARNED

[ HTML tags & structure ] [ CSS selectors & properties ] [ Box model ] [ Semantic HTML ] [ Google Fonts ] [ CSS Flexbox basics ] [ Hover effects & transitions ] [ Basic responsiveness ] [ Chrome DevTools ]

---

WEEK 5

# Portfolio Website with Bootstrap

Timeline: 3–4 days

### ■■ BEFORE YOU START

This is a brand new file — create a new file called portfolio.html (separate from Week 4's profile.html). You will use the same HTML and CSS skills from Week 4, but this time you will also use Bootstrap, a CSS framework that makes websites look professional quickly. Keep your profile.html file — you can link to it from your portfolio later!

## ■ PROJECT PROMPT

*Copy and paste this entire prompt into your AI coding assistant:*

```
I am a beginner learning web development (I built a profile page with HTML and CSS last week). Please create a full portfolio website using HTML, CSS, and Bootstrap that includes: 1. A responsive navigation bar at the top with links: Home, Projects, About, Contact — that collapses into a hamburger menu on mobile 2. A hero section with my name, a tagline like 'Aspiring Full-Stack Developer', and a call-to-action button that scrolls to Projects 3. A Projects section showing 3 project cards in a grid (each card has a placeholder image, project title, short description, and a 'View Project' button) 4. An About Me section with a photo on one side and text on the other, in two columns that stack on mobile 5. A Contact section with a form (name, email, message, submit button) 6. A footer with copyright and social media icon links SETUP INSTRUCTIONS: - Explain what Bootstrap is and why developers use CSS frameworks (saves time, looks professional, already responsive) - Explain what a CDN is in simple terms - Show me exactly how to include Bootstrap 5 using CDN links (both CSS and JavaScript) — just add two lines to the HTML head and one before closing body - Create a single HTML file called portfolio.html - Show me how to open it in my browser CODE REQUIREMENTS: - Add a detailed comment above EVERY HTML element explaining what it does - Add a detailed comment next to EVERY Bootstrap class explaining what that class does (e.g., 'container' centers content, 'row' creates a horizontal group, 'col-md-4' makes 3 equal columns on medium screens) - Use Bootstrap's grid system (container, row, col) for layout - Use Bootstrap components: navbar, cards, buttons, forms, and grid - Add a small custom CSS section for personalization (custom colors, fonts, spacing) - Use a Google Font for headings - Use placeholder images from https://placehold.co/ - Make everything fully responsive — test at desktop, tablet, and mobile sizes
```

## ? QUESTIONS TO ASK YOUR AI

*After your project is working, paste these questions one at a time into your AI assistant:*

- Explain each line of the code in detail, especially every Bootstrap class. What does each class do and why did we use it?
- What is Bootstrap and why do developers use CSS frameworks? What are the alternatives to Bootstrap? When should I use a framework vs writing CSS myself?
- What is a CDN (Content Delivery Network)? Why do we load Bootstrap from a CDN instead of downloading it? What are the pros and cons?
- How does the Bootstrap grid system work? Explain container, row, and col. What do the breakpoints (sm, md, lg, xl) mean? How do I make a 2-column layout that becomes 1 column on phones?
- How do Bootstrap components work? Walk me through how a navbar, card, button, and form are built. How do I customize their colors?
- What is responsive design in depth? How does Bootstrap make things responsive automatically? What are media queries and how does Bootstrap use them?
- What is mobile-first design? Why does Bootstrap use a mobile-first approach? How does this affect how I write my code?
- How do I customize Bootstrap? How do I override Bootstrap styles with my own CSS? Why does the order of CSS files matter?
- What are Bootstrap utility classes? Show me the 20 most useful ones for spacing (m-, p-), text (text-center, fw-bold), display (d-flex, d-none), and colors.
- What is a CSS framework vs a CSS library vs writing from scratch? What are Tailwind CSS and Material UI? How do they compare to Bootstrap?

## ✓ CONCEPTS YOU LEARNED

[ CSS frameworks ] [ Bootstrap 5 grid system ] [ Responsive design ] [ CDN ] [ Bootstrap components ] [ Mobile-first design ] [ Navbar & navigation ] [ Cards & forms ] [ Utility classes ]

---

WEEK 6

# Interactive Portfolio with JavaScript

Timeline: 3–4 days

### ■■ BEFORE YOU START

Open your portfolio.html file from Week 5. This week you will add JavaScript to that same file to make it interactive. You are NOT creating a new file — you are enhancing the portfolio website you already built. Make sure your Week 5 portfolio is working and looks good before starting this week.

## ■ PROJECT PROMPT

*Copy and paste this entire prompt into your AI coding assistant:*

```
I am a beginner learning web development (I built a portfolio website with
Bootstrap last week). Now I want to add interactivity with JavaScript. Please take
my portfolio website and add the following features: 1. A dark mode toggle button
in the navbar that switches between light and dark themes. When clicked, ALL
colors should change — background, text, cards, navbar. Remember the user's
choice. 2. Smooth scrolling: when clicking a nav link, the page smoothly scrolls
to that section instead of jumping 3. Form validation: before submitting the
contact form, check that the name is not empty, email contains @ and a dot, and
message is at least 10 characters. Show a red error message under each invalid
field. 4. A typing animation on the hero section tagline that types out the text
letter by letter, waits, deletes it, then types the next phrase from a list of 3
phrases 5. A 'scroll to top' button that appears when the user scrolls down, and
smoothly scrolls back to the top when clicked SETUP INSTRUCTIONS: - Explain what
JavaScript is and how it makes websites interactive (HTML = structure, CSS =
style, JS = behavior) - Show me how to add JavaScript to my HTML file using a
<script> tag at the bottom (explain why it goes at the bottom) - Keep everything
in one single HTML file - Show me how to open the browser console (F12 or
right-click > Inspect > Console) to see errors CODE REQUIREMENTS: - Add a detailed
comment above EVERY line of JavaScript explaining what it does in plain English -
Use modern JavaScript (let/const, arrow functions, template literals) but explain
what each one means - Use addEventListener instead of inline onclick - Use
querySelector and querySelectorAll for selecting elements - Keep the code
organized with clear sections for each feature - Add CSS transitions so the dark
mode switch feels smooth (not instant)
```

## ? QUESTIONS TO ASK YOUR AI

*After your project is working, paste these questions one at a time into your AI assistant:*

- Explain each line of the JavaScript code in detail. Walk me through exactly what happens when I click the dark mode button, step by step.
- What is the DOM (Document Object Model)? How does JavaScript 'see' and interact with HTML? Show me a mental model of how the DOM tree works.
- What are events and event listeners? What is the difference between addEventListener and onclick? Show me the 10 most common events (click, submit, scroll, keydown, etc.).
- What is querySelector and how does it find elements? What is the difference between querySelector and querySelectorAll? How do CSS selectors work in JavaScript?
- What is the difference between let, const, and var? Why did JavaScript add let and const? When should I use each one? What is 'hoisting'?
- What are functions in JavaScript? What is the difference between regular functions and arrow functions (=>)? When should I use each?
- What are template literals (backtick strings)? How are they better than regular strings? Show me 5 examples of using them.
- How does JavaScript handle the 'typing animation' step by step? What are setTimeout and setInterval? How does timing work in JavaScript?
- What is the 'this' keyword in JavaScript? Why does it confuse so many people? Show me examples where it behaves differently.
- What are the 15 most important JavaScript methods for beginners? (like getElementById, classList.add, textContent, innerHTML, style, etc.) Show a cheat sheet.
- How do CSS transitions and JavaScript work together for smooth animations? How do I animate things like color, opacity, and position?

## ✓ CONCEPTS YOU LEARNED

[ DOM manipulation ] [ Event listeners ] [ querySelector ] [ let/const/var ] [ Arrow functions ] [ Template literals ] [ setTimeout/setInterval ] [ classList API ] [ Form validation ] [ CSS transitions with JS ]

---

WEEK 7

# Quiz App in Browser

Timeline: 3–4 days

### ■■ BEFORE YOU START

This is a brand new project! Create a new file called quiz.html. You will use the HTML, CSS, and JavaScript skills from Weeks 4–6 to build a standalone quiz app from scratch. No previous project files are needed — everything will be in one fresh file.

## ■ PROJECT PROMPT

*Copy and paste this entire prompt into your AI coding assistant:*

```
I am a beginner learning JavaScript (I've added interactivity to a portfolio website). Please create a complete quiz application that runs entirely in the browser using HTML, CSS, and JavaScript. Here's what I need: 1. A start screen with the quiz title, a short description, and a 'Start Quiz' button 2. At least 10 multiple-choice questions about general coding knowledge (4 options each) 3. Show one question at a time with a question counter ('Question 3 of 10') 4. Highlight the selected answer in blue, then show if it was correct (green) or wrong (red) after clicking 'Submit Answer' 5. If wrong, highlight the correct answer in green so the user learns 6. Show a progress bar that fills up as the user moves through questions 7. A results screen at the end showing: total score, percentage, number correct vs wrong, and a message based on score (Excellent, Good, Keep Practicing) 8. A 'Restart Quiz' button that shuffles the questions and starts over SETUP INSTRUCTIONS: - Create a single file called quiz.html - No installations or libraries needed — pure HTML, CSS, and JavaScript - Show me how to open it in the browser CODE REQUIREMENTS: - Add a detailed comment above EVERY line of code explaining what it does - Store all quiz questions in an array of objects at the top of the JavaScript section - Use a 'state' object to track: current question index, score, selected answers, and quiz status - Create separate functions for: rendering the current question, checking the answer, moving to next question, showing results, and restarting - Use DOM manipulation to dynamically create and update the HTML (don't hide/show pre-built sections — build them with JavaScript) - Add smooth CSS transitions between questions - Make it fully responsive and look great on mobile - Use a modern, clean design with a consistent color scheme
```

## ? QUESTIONS TO ASK YOUR AI

*After your project is working, paste these questions one at a time into your AI assistant:*

- Explain each line of the quiz app code in detail. Walk me through the complete lifecycle: from clicking Start to seeing Results.
- What are arrays of objects and why are they perfect for storing quiz data? How do I access properties inside objects inside arrays? Show me examples with our quiz data.
- What is 'state management' and why does it matter? How does our quiz app know which question we're on, what the score is, and whether the quiz is over?
- How do we dynamically create HTML elements with JavaScript? What is innerHTML vs createElement vs textContent? What are the pros and cons of each?
- What is the difference between == and === in JavaScript? Why should I almost always use ===? Show examples where they give different results.
- How does the array shuffle algorithm work? What is the Fisher-Yates shuffle? Why can't we just use sort(() => Math.random() - 0.5)?
- What are array methods in depth? Explain .map(), .filter(), .find(), .forEach(), .reduce(), .some(), and .every() with examples related to our quiz.
- What are callback functions? How do event listeners use callbacks? Why is JavaScript built around callbacks?
- What is the difference between passing a function and calling a function? Why do we write addEventListener('click', handleClick) without parentheses?
- How would I add these features: a countdown timer for each question, a high score system using localStorage, and different difficulty levels? Walk me through each.
- What is the 'spread operator' (...)? What is 'destructuring'? Show me 5 examples of each and how they make code cleaner.

## ✓ CONCEPTS YOU LEARNED

[ Arrays of objects ] [ State management ] [ Dynamic HTML rendering ] [ innerHTML vs createElement ] [ === strict equality ] [ Array methods (map, filter, find) ] [ Callback functions ] [ CSS animations ] [ Application logic & flow control ]

---

WEEK 8

# Weather App with API

Timeline: 4–5 days

### ■■ BEFORE YOU START

This is a brand new project! Create a new file called weather.html. This week you will learn to fetch real data from the internet using APIs. You will need to sign up for a free API key at openweathermap.org (instructions are in the prompt). No previous project files are needed.

## ■ PROJECT PROMPT

*Copy and paste this entire prompt into your AI coding assistant:*

```
I am a beginner learning web development (I've built several projects including a quiz app). Now I want to learn how to fetch real data from the internet. Please create a weather application using HTML, CSS, and JavaScript that does the following: 1. Has a search bar where I type a city name and press Enter or click a Search button 2. Shows the current weather for that city: temperature (in both Fahrenheit and Celsius with a toggle), weather condition (sunny, cloudy, rainy, etc.), humidity percentage, wind speed, and a weather icon 3. Shows a 'feels like' temperature 4. Shows a 5-day forecast with day name, icon, and high/low temperature 5. Changes the background color/gradient based on weather (blue for clear, gray for cloudy, dark for rain) 6. Shows the city name, country, and current date/time 7. Has a loading spinner while fetching data 8. Shows a clean error message if the city is not found API SETUP INSTRUCTIONS (step by step): - Go to https://openweathermap.org/api and click 'Sign Up' - Create a free account (you only need an email address) - After signing up, go to 'API Keys' in your profile and copy your key - Explain that the free tier gives 1000 API calls per day (more than enough) - Show me exactly where to paste my API key in the code (put a clear placeholder like YOUR_API_KEY_HERE) SETUP INSTRUCTIONS: - Create a single file called weather.html - No installations needed — we use the browser's built-in fetch() function - Show me how to test it in my browser CODE REQUIREMENTS: - Add a detailed comment above EVERY line of code explaining what it does - Use the fetch() API with async/await (explain what these mean in the comments) - Use the OpenWeatherMap 'Current Weather' API and '5 Day Forecast' API - Handle errors gracefully: no internet, invalid city, API key issues - Create separate functions for: fetching data, displaying current weather, displaying forecast, handling errors - Make it responsive and beautiful on both desktop and mobile - Use weather-appropriate icons (you can use the icons provided by OpenWeatherMap's API)
```

## ? QUESTIONS TO ASK YOUR AI

*After your project is working, paste these questions one at a time into your AI assistant:*

- Explain each line of the weather app code in detail. Walk me through exactly what happens from the moment I type a city and press Search.
- What is an API? Explain it like I'm 5. Then explain it technically. How does our code 'talk' to the OpenWeatherMap server? What goes back and forth?
- What is the fetch() function? How does it work behind the scenes? What is an HTTP request? What are GET vs POST requests?
- What is async/await? Why can't we just write regular code to get data from an API? What are Promises? Explain the three states: pending, fulfilled, rejected.
- What is JSON? How do we go from a URL to a JavaScript object we can use? Walk me through: fetch URL → get response → parse JSON → use data.
- What are HTTP status codes? What do 200, 301, 400, 401, 403, 404, and 500 mean? How do we check for errors in our API response?
- What is an API key? Why do APIs require them? How should we keep them safe? Why is it bad to put API keys in public code on GitHub?
- What is the difference between synchronous and asynchronous code? Why is JavaScript single-threaded but can still handle APIs? What is the event loop?
- What are template literals and how do we use them to build URLs with variables? How does string interpolation work?
- How do APIs work with rate limits and quotas? What happens if I make too many requests? How do real companies handle millions of API calls?
- How would I add: geolocation to automatically detect my city, a search history feature, and the ability to save favorite cities? Walk me through each.

## ✓ CONCEPTS YOU LEARNED

[ APIs & HTTP requests ] [ fetch() function ] [ async/await & Promises ] [ JSON parsing ] [ HTTP status codes ] [ Error handling with APIs ] [ API keys & security ] [ Asynchronous JavaScript ] [ Event loop basics ]

---

WEEK 9

# Expense Tracker

Timeline: 4–5 days

**■■ BEFORE YOU START**

This is a brand new project! Create a new file called expenses.html. You will use all your frontend skills (HTML, CSS, JavaScript, Bootstrap) to build a complete expense tracker app. This project brings together everything you have learned about frontend development so far. No previous project files are needed.

## ■ PROJECT PROMPT

*Copy and paste this entire prompt into your AI coding assistant:*

```
I am a beginner learning web development (I've built apps including a weather app
with API). Please create a full expense tracker web application using HTML, CSS,
JavaScript, and Bootstrap that does the following: 1. A form to add a new expense:
amount (number input), category (dropdown: Food, Transport, Entertainment,
Shopping, Bills, Health, Other), description (text), and date (date picker,
defaults to today) 2. A dashboard section showing: total spending this month,
total spending today, and spending by category with amounts 3. A list of all
expenses showing date, description, category, and amount — with delete buttons 4.
Filter expenses by: category (dropdown), date range (from and to date pickers),
and a search box for descriptions 5. Sort expenses by: date (newest/oldest),
amount (highest/lowest) 6. A simple bar chart or visual breakdown showing spending
per category (you can build this with CSS — colored bars with widths proportional
to amounts) 7. Save all data to localStorage so it persists when the browser is
closed 8. An 'Export to CSV' button that downloads all expenses as a .csv file
SETUP INSTRUCTIONS: - Create a single file called expenses.html - Include
Bootstrap 5 via CDN for styling - No other installations needed - Show me how to
open it in the browser CODE REQUIREMENTS: - Add a detailed comment above EVERY
line of code explaining what it does - Store expenses as an array of objects in
localStorage - Use array methods: .map(), .filter(), .reduce(), .sort(),
.forEach() - Create separate, well-named functions for each feature - Use event
delegation where appropriate - Handle edge cases: empty fields, negative amounts,
invalid dates - Make it responsive and look professional with Bootstrap - Use
number formatting (show $ with two decimal places)
```

## ? QUESTIONS TO ASK YOUR AI

*After your project is working, paste these questions one at a time into your AI assistant:*

- Explain each line of the expense tracker code in detail. Walk me through the data flow: adding an expense, filtering, and seeing the updated dashboard.
- What is localStorage? How does it work? How is it different from cookies and sessionStorage? What are its limits? What can and can't I store in it?
- Deep dive into array methods: explain .map(), .filter(), .reduce(), .sort(), .find(), .some(), .every(), and .findIndex() with real examples from our expense data.
- What is event delegation? Why is it more efficient than adding event listeners to every button? How does event bubbling work?
- How does the filter and sort system work? How do we chain multiple filters together? What happens under the hood when we filter by category AND date range?
- How do we create a CSV file with JavaScript and trigger a download? What is a Blob? What is URL.createObjectURL? Walk me through the entire process.
- What is the Date object in JavaScript? How do date comparisons work? How do I format dates for display? What are common date pitfalls?
- What is number formatting? How does .toFixed(2) work? How do I format currency properly? What is the Intl.NumberFormat API?
- What is the difference between localStorage, a JSON file, and a real database? When should I use each? Why would this app need a real database at scale?
- How would I add: monthly budget limits with warnings, recurring expenses, income tracking, and charts using a library like Chart.js? Walk me through each.

## ✓ CONCEPTS YOU LEARNED

[ localStorage ] [ Array methods (map, filter, reduce, sort) ] [ Event delegation ] [ Data filtering & sorting ] [ CSV export ] [ Date handling ] [ Number formatting ] [ Blob & file download ] [ State management patterns ]

---

WEEK 10

# Backend with Flask + Database

Timeline: 5–6 days

### ■■ BEFORE YOU START

Big switch this week! You are moving from frontend-only apps (HTML files in the browser) to backend development (a Python server). You will rebuild the expense tracker concept from Week 9, but this time with a real Python backend and database. You will need to install Flask by running: pip install flask. Create a new folder called 'expense-tracker-backend' and work inside it. This is the same Python you used in Weeks 1–3, but now it powers a web server!

## ■ PROJECT PROMPT

*Copy and paste this entire prompt into your AI coding assistant:*

```
I am a beginner learning web development (I've been building frontend apps for 9 weeks). Now I want to learn backend development. Please create a full expense tracker with a Python Flask backend and SQLite database, plus a simple HTML/CSS/JavaScript frontend that talks to it. Here's what I need: BACKEND (Python Flask + SQLite): 1. A Flask server with these API endpoints: - POST /api/expenses — add a new expense (amount, category, description, date) - GET /api/expenses — get all expenses (support optional query params: category, start_date, end_date) - GET /api/expenses/<id> — get a single expense by ID - PUT /api/expenses/<id> — update an expense - DELETE /api/expenses/<id> — delete an expense - GET /api/summary — get total spending and spending by category 2. A SQLite database with an 'expenses' table 3. Input validation on all endpoints 4. Proper error responses with status codes FRONTEND (HTML/CSS/JS): 1. A simple but clean interface that lets me add, view, filter, and delete expenses 2. Uses fetch() to call the Flask API endpoints 3. Displays data from the API on the page SETUP INSTRUCTIONS (very detailed, I've never used Flask before): - Explain what a backend server is in simple terms (like a waiter at a restaurant) - Show me how to install Flask: open terminal, type: pip install flask - Explain what pip is (Python's package manager, like an app store for Python libraries) - Create two files: app.py (the backend) and a templates/index.html (the frontend) - Explain the folder structure I need - Show me how to run the Flask server: python app.py - Explain that the server runs at http://127.0.0.1:5000 and what that means - Show me how to test API endpoints in the browser and with curl commands CODE REQUIREMENTS: - Add a detailed comment above EVERY line of code in BOTH files - In app.py: explain what routes, decorators, request, jsonify, and each Flask concept means - Create the SQLite database and table automatically when the app starts - Use parameterized SQL queries (explain why — SQL injection prevention) - Return proper HTTP status codes (200, 201, 400, 404, 500) - Add CORS support (explain what CORS is and why we need it) - Keep the code well-organized with comments separating each section
```

## ? QUESTIONS TO ASK YOUR AI

*After your project is working, paste these questions one at a time into your AI assistant:*

- Explain each line of both app.py and index.html in detail. Walk me through what happens when I add an expense: from clicking Submit to seeing it in the list.
- What is a server? What does Flask do exactly? How does it listen for requests and send responses? Explain the request-response cycle with a restaurant analogy.
- What is a REST API? What do the HTTP methods (GET, POST, PUT, DELETE) mean? How do they map to CRUD? Why is this the standard way to build APIs?
- What is a database? Why SQLite? What is the difference between SQLite, MySQL, and PostgreSQL? When should I use each one?

- What is SQL? Walk me through these commands in detail: CREATE TABLE, INSERT INTO, SELECT, UPDATE, DELETE, WHERE, ORDER BY, GROUP BY. Show examples with our expenses table.
- What is SQL injection and why is it dangerous? Show me an example of how a hacker could attack our app without parameterized queries. How do parameterized queries prevent this?
- What is CORS? Why does the browser block requests from one origin to another? How does Flask-CORS fix this? What is the 'same-origin policy'?
- What are HTTP status codes in detail? Walk me through all the common ones (200, 201, 204, 400, 401, 403, 404, 405, 500) with real examples.
- What are Flask decorators like @app.route()? What is a decorator in Python? How does Flask know which function to call when a URL is requested?
- How does the frontend talk to the backend using fetch()? What is the full journey of a request: from JavaScript fetch() → HTTP request → Flask route → database query → response → back to JavaScript?
- What would I need to learn to add user authentication (login/signup)? What are sessions, cookies, and JWT tokens? Give me a high-level overview.

## ✓ CONCEPTS YOU LEARNED

[ Backend servers & Flask ] [ REST API design ] [ HTTP methods & status codes ] [ SQLite & SQL basics ] [ CRUD with databases ] [ SQL injection prevention ] [ CORS ] [ Frontend-backend communication ] [ Decorators & routes ] [ API testing with curl ]

---

WEEK 11

# AI Chatbot with OpenAI

Timeline: 5–6 days

### ■■ BEFORE YOU START

This is a brand new project that builds on your Flask skills from Week 10. Create a new folder called 'ai-chatbot' for this project. You will need to install a few libraries by running: pip install flask openai python-dotenv. You will also need to sign up for an OpenAI API key at platform.openai.com (instructions are in the prompt). This project combines your Flask backend skills with a real AI API!

## ■ PROJECT PROMPT

*Copy and paste this entire prompt into your AI coding assistant:*

```
I am a beginner learning full-stack web development (I just built a Flask backend with database last week). Now I want to build something really cool — an AI chatbot. Please create a chatbot web application with a Python Flask backend and an HTML/CSS/JavaScript frontend that uses the OpenAI API. Here's what I need: FEATURES: 1. A beautiful chat interface that looks like a real messaging app (messages on left for AI, right for user, with different colors) 2. A text input at the bottom with a Send button (and pressing Enter also sends) 3. The AI should remember the conversation context (not just answer one question at a time) 4. A 'typing...' indicator while waiting for the AI response 5. A system prompt that gives the chatbot a personality (e.g., 'You are a friendly coding tutor who explains things simply') 6. A button to clear the conversation and start fresh 7. Display timestamps on each message 8. Auto-scroll to the latest message API SETUP INSTRUCTIONS (very detailed): - Go to https://platform.openai.com and create an account - Explain that OpenAI charges for API usage but gives free credits for new accounts (or has a pay-as-you-go model that's very cheap for small projects) - Go to API Keys section and create a new secret key - IMPORTANT: Explain that this key is like a password and should NEVER be shared or put in public code - Show me how to store the API key safely in a .env file - Show me how to install the required libraries: pip install flask openai python-dotenv FILE STRUCTURE: - Create these files: app.py (Flask backend), templates/index.html (frontend), .env (API key), .gitignore (to hide .env) - Explain why we have a .gitignore file CODE REQUIREMENTS: - Add a detailed comment above EVERY line in ALL files - In app.py: create a /api/chat POST endpoint that receives the user's message and conversation history, sends it to OpenAI, and returns the AI response - Use the openai Python library with the ChatCompletion API (use the gpt-4o-mini model — it's the cheapest and works great for this project) - Include a system message that defines the chatbot's personality - Handle API errors gracefully (rate limits, invalid key, network errors) - In the frontend: use fetch() to send messages to the Flask backend - Store conversation history in a JavaScript array and send it with each request - Make the chat interface responsive and beautiful with CSS - Add smooth animations for new messages appearing
```

## ? QUESTIONS TO ASK YOUR AI

*After your project is working, paste these questions one at a time into your AI assistant:*

- Explain each line of code in both app.py and index.html in detail. Walk me through the complete flow: I type a message → it goes to Flask → Flask calls OpenAI → response comes back → appears in chat.
- What is the OpenAI API? How do Large Language Models (LLMs) like GPT work at a high level? What is the difference between GPT-4o-mini and GPT-4o? Which one should I use for my projects?
- What is an API key and why must we keep it secret? What could happen if someone steals my key? How does the .env file and python-dotenv keep it safe?

- What is a POST request and why do we use POST instead of GET for sending chat messages? What is the request body? How do we send JSON data to an API?
- How does conversation context work with the OpenAI API? Why do we send the entire conversation history with each request? What are the 'system', 'user', and 'assistant' roles?
- What is prompt engineering? How does the system message affect the AI's behavior? Show me 5 different system prompts and how they change the chatbot's personality.
- What are tokens? How does OpenAI count them? Why do they matter for cost and context limits? How do I estimate how many tokens my conversation will use?
- What is rate limiting? What happens if I send too many requests to OpenAI? How should a real app handle rate limits and errors?
- What is the .gitignore file? Why is it critical for security? What other files should always be in .gitignore? What happens if I accidentally push my API key to GitHub?
- What are environment variables? How are they different from regular variables in code? How do they work on different operating systems?
- How would I add: streaming responses (seeing the AI type word by word), multiple chatbot personalities the user can choose, and saving conversation history to a database?

## ✓ CONCEPTS YOU LEARNED

[ LLM APIs & OpenAI ] [ API key security ] [ Environment variables & .env ] [ POST requests with JSON body ] [ Conversation context & roles ] [ Prompt engineering ] [ Tokens & rate limits ] [ Full-stack integration ] [ .gitignore & security ] [ Error handling patterns ]

---

WEEK 12

# Deployment — Going Live

Timeline: 5–6 days

### ■■ BEFORE YOU START

In this final week, you will deploy the AI chatbot you built in Week 11 to the internet so anyone can use it. Make sure your Week 11 chatbot project is fully working on your computer before starting this week. Open your 'ai-chatbot' folder from Week 11 — that is the project you will be deploying. You will learn Git, GitHub, and Render to put your app live online!

## ■ PROJECT PROMPT

*Copy and paste this entire prompt into your AI coding assistant:*

```
I am a beginner who just built an AI chatbot with Flask, HTML/CSS/JS, and the OpenAI API. Now I want to put it live on the internet so anyone can use it. Please walk me through deploying my chatbot project. Here's what I need: PART 1 — GIT AND GITHUB (version control): 1. Explain what Git is in simple terms (a save system for code, like checkpoints in a video game) 2. Show me how to install Git (if not already installed) 3. Walk me through these commands step by step: - git init (start tracking this project) - git add . (stage all files) - git commit -m 'Initial commit' (save a checkpoint) - Create a new repository on GitHub.com (show the steps on the website) - git remote add origin [URL] (connect local project to GitHub) - git push -u origin main (upload code to GitHub) 4. Explain what .gitignore is and make sure .env is in it BEFORE we push PART 2 — PREPARE FOR DEPLOYMENT: 1. Create a requirements.txt file (explain what it is — a list of libraries your project needs) Show the exact command: pip freeze > requirements.txt 2. Create a render.yaml configuration file for Render 3. Modify app.py if needed for production (explain any changes) 4. Explain the difference between development (localhost) and production (real server) PART 3 — DEPLOY TO RENDER: 1. Go to https://render.com and sign up with your GitHub account (free) 2. Click 'New Web Service' and connect your GitHub repository 3. Set the build command to: pip install -r requirements.txt 4. Set the start command to: python app.py 5. Show me where to add environment variables (the API key) in Render's dashboard — NEVER put the API key in the code 6. Click Deploy and wait for it to go live 7. Show me my live URL and how to share it IMPORTANT NOTES TO INCLUDE: - Explain what happens behind the scenes when someone visits my URL - Explain what a domain name is and how to buy a custom one if I want - Show me how to check logs on Render if something breaks - Explain what CI/CD means in simple terms CODE/FILE REQUIREMENTS: - Add detailed comments in every configuration file - Show me the exact folder structure I should have before deploying - Include a README.md file for the GitHub repo that explains what the project is
```

## ? QUESTIONS TO ASK YOUR AI

*After your project is working, paste these questions one at a time into your AI assistant:*

- Explain every step and command in detail. Walk me through the entire flow from code on my computer to a live website anyone can visit.
- What is Git? Why is version control important? What is the difference between Git (tool on your computer) and GitHub (website for sharing code)?
- Walk me through Git commands in depth: init, add, commit, push, pull, clone, branch, merge, status, log, diff. Show me what each one does with examples.
- What is a repository? What is a commit? What is a branch? Explain these with a mental model like 'branches on a tree' or 'parallel universes of your code'.
- What is requirements.txt and why is it critical for deployment? How does the server know which libraries to install? What would happen without it?

- What is deployment? What does it mean for my code to run on a server somewhere? What is Render doing behind the scenes when I click Deploy?
- What is the difference between development and production? Why do we need different settings? What are common things that work locally but break in production?
- What are environment variables on a server? How do I set them in Render? Why are they the correct way to handle secrets in production?
- What is a domain name? How does DNS work? How does typing 'google.com' eventually reach a specific server? How do I buy and connect a custom domain?
- What is CI/CD (Continuous Integration / Continuous Deployment)? How does it work with GitHub and Render? Why do professional teams use it?
- What should I learn next? What is the roadmap from here to becoming a junior developer? What are the most in-demand skills in 2026?
- How do I debug a deployed app that's crashing? What are server logs? How do I read error messages from Render?

## ✓ CONCEPTS YOU LEARNED

[ Git & version control ] [ GitHub ] [ Repositories & commits ] [ requirements.txt ] [ Deployment to Render ] [ Environment variables in production ] [ Domain names & DNS ] [ CI/CD basics ] [ Development vs production ] [ README documentation ]

---

# What's Next?

Congratulations! You've completed 12 weeks of real, project-based coding. You didn't just watch tutorials — you **built things**. You now have a portfolio of projects, a solid understanding of full-stack development, and the ability to learn any new technology by asking the right questions.

## Suggested Next Steps

- **Learn a Frontend Framework** — React, Vue, or Svelte. React is the most in-demand. Start with the official React tutorial and build a project with it.
- **Learn an Advanced Backend** — Try Django (Python) or Express.js (JavaScript/Node.js) for more powerful backend features like authentication, middleware, and database ORMs.
- **Contribute to Open Source** — Go to GitHub, find beginner-friendly projects (look for 'good first issue' labels), and make your first contribution. This is resume gold.
- **Build a Personal Project** — Think of something you wish existed, and build it. A habit tracker, a recipe app, a budget tool. Real projects born from real needs impress employers more than anything.
- **Create a Developer Portfolio** — Put your best 3–4 projects on a clean portfolio site (you already know how to build one!). Link it on your resume and LinkedIn.
- **Practice for Interviews** — Start solving problems on LeetCode (Easy level), learn basic data structures (arrays, objects, linked lists), and practice explaining your projects out loud.

## The One Rule: Keep Building

The developers who grow fastest are the ones who build one project a week, even small ones. Every project teaches you something new. Every bug you fix makes you sharper. Keep your AI assistant open, keep asking questions, and keep shipping code.

**Free AI Coding Tools (pick any one):**

```
Copilot: https://github.com/features/copilot/plans
Cursor: https://cursor.sh
Windsurf: https://windsurf.com
Cline: https://marketplace.visualstudio.com/items?itemName=saoudrizwan.claude-dev
Antigravity: https://antigravity.google
```

**Connect with me for more AI-powered learning tips**

Instagram: @praneeth_kalluri — https://www.instagram.com/praneeth_kalluri/

LinkedIn: https://www.linkedin.com/in/praneethkalluri/

Have questions? DM me on Instagram or connect on LinkedIn — I read every message.

*"The best time to start coding was yesterday. The second best time is right now."*