خلاصهٔ دوره‌های آکادمی Anthropic

یادداشت‌های مطالعه به زبان فارسی

تهیه‌شده از anthropic.skilljar.com

18 دوره • هر دوره از صفحهٔ جدید

فهرست دوره‌ها

دوره ۱ — Claude 101 (مقدمات کلود)

موضوع: استفاده از Claude برای کارهای روزمرهٔ حرفه‌ای — قابلیت‌های اصلی، پروژه‌ها، آرتیفکت‌ها، اسکیل‌ها و کانکتورها. **تعداد درس:** ۱۴

خلاصهٔ کلی دوره

کلود صرفاً یک چت‌بات نیست؛ یک «شریک فکری» است. شما زمینه (context) و تخصص را می‌آورید و کلود هوش و سرعت را. دوره از آشنایی اولیه شروع می‌شود، به ابزارهای سازمان‌دهی کار (پروژه‌ها، آرتیفکت‌ها، اسکیل‌ها) می‌رسد، و با گسترش دسترسی کلود به ابزارها و وب (کانکتورها، جست‌وجوی سازمانی، Research) تمام می‌شود.

نکات مهم هر درس

۱. کلود چیست؟

- کلود یک دستیار هوش مصنوعی است که بر پایهٔ سه اصل طراحی شده: **مفید، بی‌ضرر، صادق** (helpful, harmless, honest).
- تفاوت با چت‌بات ساده: توانایی استدلال چندمرحله‌ای، کار با فایل، و تولید خروجی قابل استفاده.
- راه‌های دسترسی: **وب، اپ دسکتاپ، موبایل** — گفت‌وگوها بین دستگاه‌ها همگام‌سازی می‌شوند.

۲. اولین گفت‌وگو با کلود

- **ساختار یک پرامپت خوب:** زمینه (context) + وظیفه (task) + قالب و سبک خروجی (format/style).
- می‌توانید فایل و تصویر آپلود کنید تا کلود زمینهٔ بیشتری داشته باشد.
- با پیام‌های پیگیرانه (follow-up) پاسخ را اصلاح و دقیق‌تر کنید.
- **نکتهٔ کلیدی:** طوری با کلود حرف بزنید که انگار با یک همکار باهوش صحبت می‌کنید.

۳. گرفتن نتایج بهتر

- **مشکلات رایج:** پاسخ کلی و مبهم، برداشت اشتباه از خواسته، طولانی بودن بی‌مورد.
- **راه‌حل:** تکرار (iteration). اولین پاسخ را نقطهٔ شروع بدانید نه نتیجهٔ نهایی.
- مفهوم **AI Fluency**: مهارت کار روان و مؤثر با هوش مصنوعی.
- **Evals:** برای فهمیدن اینکه کلود در جریان کاری خاص شما چقدر خوب عمل می‌کند، آزمون‌های ارزیابی بسازید.

۴. کار با کلود روی دسکتاپ

سه شکل کار وجود دارد و باید بدانید کدام را انتخاب کنید: ۱. **نوبت‌به‌نوبت (turn by turn)** — پرسش و پاسخ تعاملی برای کارهای سریع. ۲. **واگذاری کل وظیفه** — کلود کار را مستقل اجرا می‌کند و شما کار دیگری می‌کنید. ۳. **ساخت نرم‌افزار در کدبیس** — کار روی کد پروژه.

۵. پروژه‌ها (Projects)

- پروژه = فضای کاری مستقل با **حافظه، تاریخچهٔ گفت‌وگو، پایگاه دانش و دستورالعمل‌های اختصاصی**.
- می‌توانید اسناد و فایل‌ها را به پایگاه دانش پروژه اضافه کنید.
- **دستورالعمل‌های پروژه (instructions)** رفتار کلود را در آن پروژه هدایت می‌کند.
- در پلن‌های Team و Enterprise می‌توان پروژه را با هم‌تیمی‌ها به اشتراک گذاشت.

۶. آرتیفکت‌ها (Artifacts)

- آرتیفکت = خروجی مستقل و تعاملی که در پنجره‌ای جدا کنار گفت‌وگو ساخته می‌شود.
- به‌جای بلوک طولانی کد یا متن داخل چت، محتوا **رندر شده و آمادهٔ استفاده** را می‌بینید: وب‌سایت کارا، نمودار تعاملی، سند قابل دانلود.
- قابل اشتراک‌گذاری با همکاران و انتشار عمومی.

۷. اسکیل‌ها (Skills)

- اسکیل = پوشه‌ای از **دستورالعمل‌ها، اسکریپت‌ها و منابع** که کلود به‌صورت پویا بارگذاری می‌کند.
- مثل «بسته‌های تخصص» عمل می‌کنند و به کلود یاد می‌دهند کار خاصی را چطور انجام دهد.
- اسکیل‌های داخلی Anthropic برای ساخت اسناد (Word، Excel، PowerPoint، PDF) موجود است.
- در تنظیمات فعال/غیرفعال می‌شوند. (فعلاً برای پلن‌های Pro، Max، Team و Enterprise)

۸. کانکتورها (Connectors)

- کانکتور، کلود را از یک «دستیار» به یک **همکار مطلع** تبدیل می‌کند.
- به کلود دسترسی می‌دهد به همان ابزارها و داده‌هایی که شما هر روز استفاده می‌کنید.
- کلود می‌تواند اطلاعات را **بخواند** و روی سیستم‌های شما **اقدام انجام دهد**.
- دیگر لازم نیست هر گفت‌وگو را از صفر شروع کنید.

۹. جست‌وجوی سازمانی (Enterprise Search)

- گزینهٔ «Ask {نام سازمان شما}» را به نوار کناری اضافه می‌کند.
- برای **یافتن و ترکیب دانش سازمانی** طراحی شده است.
- امنیت و سطح دسترسی‌ها از داده‌های سازمان محافظت می‌کند.
- فقط در پلن‌های Team و Enterprise و با فعال‌سازی توسط ادمین.

۱۰. Research برای بررسی عمیق

- Research یعنی **تحقیق سیستماتیک و چندمنبعی**.
- برخلاف یک جست‌وجوی ساده، کلود به‌صورت **عاملی (agentic)** عمل می‌کند: چند جست‌وجوی پیاپی که روی هم بنا می‌شوند.
- از **Thinking** استفاده می‌کند تا قبل از جمع‌آوری اطلاعات، رویکردش را برنامه‌ریزی کند.

• زوایای مختلف موضوع را بررسی می‌کند و خودش تصمیم می‌گیرد بعد چه چیزی را بکاود.

۱۱. کاربردها بر اساس نقش شغلی

• کلود در تقریباً هر نقشی کاربرد دارد؛ نمونه‌های عمومی: تولید گزارش وضعیت پروژه، خلاصه‌سازی، تحلیل داده، نگارش.

• Anthropic یک **گالری کاربردها (Use Case Gallery)** با راهنمای گام‌به‌گام دارد.

۱۲. راه‌های دیگر کار با کلود

• **Claude Code** — ابزار کدنویسی عاملی در ترمینال، IDE و مرورگر.

• **@Claude** — کلود در Slack.

• **Claude Design**، **Claude for Microsoft 365**، **Claude in Chrome**.

۱۳ و ۱۴. جمع‌بندی و گواهی پایان دوره

• مرور مطالب + آزمون برای دریافت گواهی قابل اشتراک در LinkedIn.

نکات کلیدی برای مرور سریع

• پرامپت خوب = زمینه + وظیفه + قالب.

• اولین پاسخ را نهایی ندانید؛ **تکرار کنید**.

• برای کار مستمر روی یک موضوع ← **پروژه** بسازید.

• برای خروجی قابل استفاده ← **آرتیفکت**.

• برای تخصص تکرارشونده ← **اسکیل**.

• برای اتصال به داده‌های واقعی خودتان ← **کانکتور**.

• برای تحقیق عمیق چندمنبعی ← **Research**.

دوره ۲ — Claude Code 101 (مقدمات کلود کد)

موضوع: استفادهٔ مؤثر از Claude Code در جریان کاری روزمرهٔ برنامه‌نویسی. **تعداد درس:** ۱۳ **پیش‌نیاز:** آشنایی پایه با ویرایشگر کد و خط فرمان + حساب Claude (Pro/Max/Enterprise) یا API key.

خلاصهٔ کلی دوره

Claude Code یک **ابزار کدنویسی عاملی (agentic)** است که کدبیس شما را می‌فهمد، فایل‌ها را ویرایش می‌کند، دستورات را اجرا می‌کند و به ابزارهای شما وصل می‌شود. تفاوت اصلی‌اش با Claude.ai این است که به‌جای کپی‌پیست کد، **خودش وارد پروژه می‌شود و کار را انجام می‌دهد.**

نکات مهم هر درس

۱. Claude Code چیست؟

- ابزار کدنویسی عاملی با دسترسی مستقیم به **فایل‌ها، ترمینال و کل کدبیس** شما.
- در دسترس در: ترمینال، VS Code، JetBrains، اپ دسکتاپ Claude، و وب.
- **عامل (Agent) چیست؟** نرم‌افزاری که با محیطش تعامل می‌کند و برای رسیدن به یک هدف مشخص، اقدام انجام می‌دهد. در هسته‌اش یک مدل زبانی بزرگ است که در یک **حلقه (loop)** کار می‌کند.

کارهایی که انجام می‌دهد: - خواندن و درک کدبیس (توضیح یک قابلیت، ردیابی یک باگ) - ویرایش فایل‌ها در کل پروژه (ریفکتور یک تابع + به‌روزرسانی همهٔ ارجاعات) - اجرای دستورات ترمینال (بیلد، تست، نصب پکیج) و تصمیم‌گیری بر اساس خروجی - جست‌وجوی وب برای مستندات و APIهای جدید

سه نکتهٔ مهم هنگام استفاده: 1. **پنجرهٔ زمینه (context window)** = حافظهٔ کاری کلود. ظرفیت زیاد دارد اما بی‌نهایت نیست. 2. **اجازه می‌گیرد** — به‌صورت پیش‌فرض قبل از اجرای دستور یا تغییر، از شما تأیید می‌گیرد. 3. **ممکن است اشتباه کند** — ممکن است منظور شما را بد بفهمد یا بیش‌ازحد پیچیده‌اش کند. در جریان بمانید.

۲. Claude Code چطور کار می‌کند — حلقهٔ عاملی

مراحل حلقه (Agentic Loop): 1. شما پرامپت را وارد می‌کنید. 2. کلود زمینهٔ لازم را جمع می‌کند؛ مدل یا متن برمی‌گرداند یا یک فراخوانی ابزار (tool call). 3. **اقدام** انجام می‌دهد — مثلاً ویرایش فایل یا اجرای دستور. 4. **نتیجه را راستی‌آزمایی می‌کند.** اگر هدف محقق شده باشد تمام می‌شود؛ وگرنه دوباره حلقه را تکرار می‌کند.

جمع‌بندی: حلقهٔ عاملی + پنجرهٔ زمینهٔ مدیریت‌شده + ابزارها + مجوزهای قابل تنظیم — همه داخل ترمینال شما. این چیزی است که آن را اساساً از یک پنجرهٔ چت متفاوت می‌کند.

۳. نصب Claude Code

- **macOS / Linux / WSL:** دستور `curl` (نصب یکجا). یا `brew install` — اما این روش **آپدیت خودکار ندارد**.
- **ویندوز:** در PowerShell از `Invoke-RestMethod` ، در CMD از `curl` ، یا `winget` (بدون آپدیت خودکار).
- بعد از نصب دستور `claude` باید کار کند؛ اگر نه، ترمینال را ری‌استارت کنید.

• وارد پوشهٔ پروژه شوید و `claude` را اجرا کنید.

۴. اولین پرامپت شما

حالت‌های تأیید — با `Shift + Tab` بین آن‌ها جابه‌جا شوید: - **Approval mode:** کلود برای هر ویرایش فایل یا اجرای دستور اجازه می‌گیرد. - **Auto-accept mode:** ویرایش فایل‌ها خودکار تأیید می‌شود، اما دستورات همچنان اجازه می‌خواهند. - **Plan Mode:** در همان منوی `Shift + Tab` .

> هیچ گزینه‌ای «درست» یا «غلط» نیست — هر کدام که با آن راحت‌ترید.

جمع‌بندی: تا می‌توانید پرامپت را **توصیفی و دقیق** بنویسید. از Plan Mode استفاده کنید تا کلود قبل از نوشتن هر کدی، جزئیات را بررسی کند.

۵. جریان کاری Explore → Plan → Code → Commit «مهم»

مهم‌ترین درس کل دوره. بدون آن، بیشتر افراد مستقیم می‌روند سراغ «کد بنویس» و بعداً مجبور به اصلاحات زیاد می‌شوند.

• **Explore (کاوش):** کلود فایل‌ها را می‌خواند تا زمینهٔ لازم پروژه را جمع کند.

• **Plan (برنامه):** در Plan Mode کلود **نمی‌تواند فایل ویرایش کند** — فقط می‌خواند و نقشهٔ اجرا می‌سازد. این نقشه معیار سنجش موفقیت می‌شود.

• **Code (کدنویسی):** رفت‌وبرگشت بین شما و کلود تا رسیدن به نتیجهٔ نهایی.

• **Commit:** بازبینی و push کد تا بتوانید سراغ قابلیت بعدی بروید.

مثال پرامپت در Plan Mode: «باید تبدیل WebP را به پایپ‌لاین آپلود تصویر اضافه کنم. بررسی کن کجای پایپ‌لاین باید انجام شود و...»

۶. مدیریت زمینه (Context Management)

• **زمینه = حافظهٔ کاری کلود.** هر فایلی که می‌خواند، هر دستوری که اجرا می‌کند و هر پیامی که می‌فرستید، فضا اشغال می‌کند.

• **وقتی پر شود چه می‌شود؟** به‌صورت خودکار **فشرده‌سازی (compaction)** انجام می‌شود: جزئیات مهم خلاصه و نتایج غیرضروری حذف می‌شوند.

دستورات کلیدی:

دستور	کاربرد
`/compact`	خلاصه‌کردن جلسات طولانی
`/clear`	شروع از نو با زمینهٔ خالی
`/context`	دیدن اینکه چه چیزی زمینه را مصرف کرده

جمع‌بندی: پرامپت‌ها را دقیق بنویسید، مصرف زمینه را بررسی کنید، و برای کارهایی که فقط نتیجه‌اش را می‌خواهید از **ساب‌ایجنت** استفاده کنید.

۷. بازبینی کد (Code Review)

- قبل از push کردن PR، از کلود بخواهید با یک **ساب‌ایجنت** تغییرات را بازبینی کند. ساب‌ایجنت در پنجرهٔ زمینهٔ مستقل خودش با «چشم تازه» کار می‌کند و **سوگیری عامل اصلی** را ندارد.
- ساب‌ایجنت بازبین را به ابزارهای **فقط-خواندنی** محدود کنید. بازبین باید ایراد بگیرد، نه فایل ویرایش کند.
- کانفیگ ساب‌ایجنت را در ریپو commit کنید تا کل تیم از همان بازبین استفاده کند.
- **اسکیل** `/commit-push-pr`: کامیت، push و ساخت PR را یکجا انجام می‌دهد.
- `--from-pr`: برای ادامهٔ کار روی یک PR در آینده.

۸. فایل CLAUDE.md «مهم»

- **مشکلی که حل می‌کند:** بدون آن، کلود هر بار از صفر شروع می‌کند — باید دوباره کدبیس را بکاود و فرض‌هایی بسازد که هدایتش را سخت می‌کند.
- یک فایل Markdown در **ریشهٔ پروژه** که کلود **هر بار در شروع جلسه خودکار می‌خواند**.
- مثل یک «اسکریپت آنبوردینگ» برای کدبیس شما.

چه چیزی بنویسیم؟ استک فناوری، ترجیحات کدنویسی، دستورات پرکاربرد — و به‌مرور تکمیلش کنید.

> تفاوت یک جلسهٔ آزاردهنده و یک جلسهٔ پربازده معمولاً به **زمینه** برمی‌گردد، و CLAUDE.md راه فراهم‌کردن آن است.

۹. ساب‌ایجنت‌ها (Subagents)

- کلود می‌تواند کارها را به ساب‌ایجنت‌ها **واگذار** کند تا موازی اجرا شوند.
- هر ساب‌ایجنت در **پنجرهٔ زمینهٔ ایزولهٔ خودش** کار می‌کند.
- **چرا مفید است؟** بخش زیادی از زمینه صرف کاوش کدبیس یا جست‌وجوی وب می‌شود که همه‌اش به قابلیت اصلی مربوط نیست. ساب‌ایجنت آن کار سنگین را انجام می‌دهد و **فقط پاسخ نهایی** را به زمینهٔ اصلی برمی‌گرداند.

۱۰. اسکیل‌ها (Skills)

- به دورهٔ اختصاصی «Introduction to agent skills» ارجاع داده شده.

۱۱. MCP (پروتکل زمینهٔ مدل)

- **MCP** یک **استاندارد باز** است که Claude Code را به ابزارها و منابع دادهٔ بیرونی وصل می‌کند.
- کلود خودش تشخیص می‌دهد چه زمانی باید از آن ابزارها استفاده کند.
- **چرا لازم است؟** بخش زیادی از زمینهٔ کاری شما بیرون از کدبیس است — در دیتابیس‌ها، اپ‌های بهره‌وری و ریپازیتوری‌های عمومی.

• **ابزار (tool) چیست؟** به عامل‌ها توانایی **انجام اقدام** می‌دهد، برخلاف هوش مصنوعی معمولی که فقط پاسخ متنی می‌دهد.

دستورات: - `claude mcp add` — افزودن سرور - فایل `.mcp.json` — محدودکردن به پروژه تا کل تیم خودکار آن را بگیرند - سرورهایی را که استفاده نمی‌کنید **غیرفعال** کنید تا زمینه مصرف نشود.

۱۲. هوک‌ها (Hooks)

• هوک اجازه می‌دهد در **نقاط مشخصی از چرخهٔ عمر** Claude Code دستور اجرا کنید.

• **تفاوت کلیدی: هوک‌ها قطعی (deterministic) هستند — همیشه اجرا می‌شوند.**

> اگر در CLAUDE.md بنویسید «بعد از هر ویرایش Prettier را اجرا کن»، بیشتر اوقات انجام می‌شود؛ اما گاهی نه. هوک آن را **بدون استثنا** اجرا می‌کند.

کاربردهای رایج: فرمت خودکار بعد از ویرایش فایل • لاگ‌کردن همهٔ دستورات برای انطباق (compliance) • مسدودکردن عملیات خطرناک مثل تغییر فایل‌های پروژه • اکشن • ارسال نوتیفیکیشن هنگام اتمام کار.

نوع هوک	کاربرد
`PostToolUse`	فرمت خودکار و لاگ‌گیری
`PreToolUse`	مسدودکردن عملیات خطرناک

• تنظیم با `/hooks` یا در `settings.json` . آن را در ریپو commit کنید تا کل تیم بگیرند.

> **قانون طلایی:** اگر چیزی باید **هر بار بدون استثنا** اتفاق بیفتد، آن را در پرامپت نگذارید — در **هوک** بگذارید.

۱۳. آزمون دوره

نکات کلیدی برای مرور سریع

• **حلقهٔ عاملی:** پرامپت → جمع‌آوری زمینه → اقدام → راستی‌آزمایی → تکرار.

• **جریان کاری طلایی:** Explore → Plan → Code → Commit.

• `Shift + Tab` = جابه‌جایی بین حالت‌های تأیید و Plan Mode.

• `/compact` ، `/clear` ، `/context` = ابزارهای مدیریت حافظه.

• **CLAUDE.md** = حافظهٔ دائمی پروژه.

• **ساب‌ایجنت** = زمینهٔ ایزوله برای کارهای سنگین؛ فقط نتیجه برمی‌گردد.

• **MCP** = اتصال به ابزارهای بیرونی.

• **هوک** = کنترل قطعی، همیشه اجرا می‌شود.

دوره ۳ — Claude Code in Action (کلود کد در عمل)

موضوع: اجرای جلسات طولانی و بدون نظارت با Claude Code که بتوان به آن‌ها اعتماد کرد — هدایت، پیکربندی، خودکارسازی و راستی‌آزمایی. **تعداد درس:** ۱۰ **سطح:** پیشرفته‌تر از Claude Code 101.

خلاصهٔ کلی دوره

پرامپت‌زدن برای کارهای کوچک آسان است. اما کارهای طولانی — مثل ریفکتور روی ده‌ها فایل یا ساخت یک قابلیت کامل — بازی متفاوتی است. این دوره یاد می‌دهد چطور کار را **قبل از شروع محدوده‌بندی** کنید و **حین اجرا هدایتش** کنید، تا بتوانید بدون نظارت لحظه‌به‌لحظه به نتیجه اعتماد کنید.

نکات مهم هر درس

۱. هدایت جلسات طولانی (Steering Long Sessions)

دو عادت کلیدی: **اول کار را محدوده‌بندی کن، بعد حین اجرا هدایتش کن.**

- **Plan Mode:** قبل از اینکه کلود حتی یک خط کد بنویسد، از او نقشه بخواهید. در این حالت کلود در وضعیت **فقط-خواندنی** تحقیق می‌کند: کد را می‌خواند، می‌فهمد چه چیزی باید تغییر کند، و نقشه را تحویل می‌دهد.
- **منوی Rewind:** وقتی کلود از مسیر منحرف شد، با آن مسیر را اصلاح کنید.
- **تعیین هدف (goal):** وقتی توصیف «تمام‌شدن کار» برایتان راحت‌تر از توصیف «مراحل کار» است، هدف تعیین کنید نه مراحل.
- **Worktrees:** کارهای موازی را در worktreeهای جداگانهٔ گیت اجرا کنید.
- **فشرده‌سازی هوشمند:** خلاصه‌سازی آنچه اهمیت دارد را حفظ می‌کند.

> نتیجه: می‌توانید به یک اجرای طولانی اعتماد کنید بدون اینکه هر قدمش را بچه‌داری کنید.

۲. یک CLAUDE.md که واقعاً دنبال می‌شود «مهم»

تلهٔ رایج: فایل CLAUDE.md مدام بزرگ می‌شود. به مشکل می‌خورید، یک قانون اضافه می‌کنید. باز مشکل، باز قانون. بعد از مدتی یک فایل غول‌پیکر دارید و کلود بخش‌هایی از آن را نادیده می‌گیرد.

این باگ نیست — ماهیت کار فایل است: - CLAUDE.md یک **پیکربندی الزام‌آور نیست؛ راهنماست (guidance)**. - **هر خط با هر خط دیگر بر سر توجه کلود رقابت می‌کند.** - هرچه فایل طولانی‌تر، رقابت درونی بیشتر، و پیروی از هر قانون منفرد **کمتر قابل اتکا**.

> **هدف این نیست که همه‌چیز را بنویسید. هدف این است که فایل را جمع‌وجور نگه دارید. هرچه فایل لاغرتر، بخش بیشتری از آن واقعاً رعایت می‌شود.**

اول از خودتان بپرسید: آیا این واقعاً باید در CLAUDE.md باشد؟ (شاید جایش هوک یا اسکیل است.)

۳. اسکیل‌های راستی‌آزمایی (Verification Skills) «مهم»

اگر قرار است فقط یک اسکیل بسازید، این را بسازید.

مشکل: معمولاً چطور کار کلود را بررسی می‌کنید؟ می‌گویید ریفکتور کن، تمام می‌شود، و بعد **باید یادتان بماند** که دوباره چک کنید — تست‌ها را اجرا کنید یا diff را بخوانید. مشکل این است که بررسی وابسته به **یادآوری شماست**. یک بار فراموش کنید، کد بد رد می‌شود.

راه‌حل: اسکیل راستی‌آزمایی این وابستگی را حذف می‌کند.

ساختار یک اسکیل: - یک **پوشه** با فایل `SKILL.md` داخلش - یک **نام** - یک **توضیح (description)** که آن را فعال می‌کند - خود **رویه (procedure)**

نکتهٔ مهم عملکردی: فقط **توضیحات** در زمینه بارگذاری می‌شوند تا وقتی که اسکیل واقعاً لازم شود. یعنی **هزینه‌ای ندارد** که هر رویهٔ تکراری را در قالب اسکیل بسته‌بندی کنید.

اشتراک‌گذاری: اسکیل را در `.claude/skills` پروژه commit کنید → کل تیم همان روش را به ارث می‌برد. کار همه به یک شکل و **خودکار** بررسی می‌شود، بدون اینکه کسی لازم باشد یادش بماند درخواست کند.

۴. حالت‌های مجوز (Permission Modes)

حالت‌های مجوز می‌گذارند **یک‌بار** تصمیم بگیرید کلود چه چیزی را بدون پرسیدن اجرا کند.

سه حالتی که با `shift-tab` می‌چرخید و آن‌ها را می‌شناسید: **plan، accept edits، manual**. این‌ها برای کار روزمره و دستی‌اند.

در مجموع شش حالت مجوز وجود دارد. هر کدام مرز متفاوتی می‌کشد بین آنچه آزادانه اجرا می‌شود و آنچه تأیید شما را می‌خواهد: - **Manual:** فقط می‌خواند، بدون پرسیدن. بقیهٔ کارها اول اجازه می‌گیرند. - **Auto:** حالتی که کار واقعاً بدون نظارت (hands-off) در آن اتفاق می‌افتد — همان که برای اجراهای طولانی سراغش می‌روید.

۵. هوک‌ها (Hooks) — نگاه عمیق‌تر

مشکل CLAUDE.md: یک درخواست است، نه یک تضمین. می‌نویسید «همیشه بعد از ویرایش فرمت کن» و کلود **معمولاً** گوش می‌دهد. معمولاً. اما در یک اجرای طولانی که تماشا نمی‌کنید، «معمولاً» کافی نیست.

هوک = کد قطعی (deterministic) که در نقطهٔ ثابتی از حلقه اجرا می‌شود. قانون را از «کلود معمولاً گوش می‌دهد» به **«کلود نمی‌تواند از آن رد شود»** تبدیل می‌کند.

- Claude Code حدود **۳۰ رویداد هوک** در طول یک جلسه شلیک می‌کند. لازم نیست همه را بدانید؛ تعداد کمی از آن‌ها را بارها استفاده خواهید کرد.

هوک‌های کلیدی:

هوک	کاربرد
`PreToolUse`	محافظت از ابزارها / مسدودکردن عملیات خطرناک
`PostToolUse`	فرمت خودکار، لاگ‌گیری
`Stop`	کنترل پایان نوبت (gate the turn)
هوک compact	حفظ وضعیت هنگام فشرده‌سازی — خلاصه‌ای از فایل‌هایی که رویشان کار می‌کردید به زمینه برمی‌گردد تا کلود از همان‌جا ادامه دهد، نه از صفر

> راه‌اندازی اولش کمی زحمت دارد، اما همان بار اولی که چیزی را در اجرایی که اصلاً تماشا نمی‌کردید بگیرد، هزینه‌اش را پس می‌دهد.

۶. روتین‌ها و حالت Headless

وقتی به کلود برای انجام کاری اعتماد کردید، قدم بعدی این است که **دستی انجامش ندهید**.

یک **طیف** است: - **روتین‌ها (Routines):** یک پرامپت ذخیره‌شده که روی زیرساخت مدیریت‌شدهٔ Anthropic در فضای ابری اجرا می‌شود. **چیزی نمی‌سازید** — مستقیم‌ترین راه خودکارسازی. - **حالت Headless + Agent SDK:** در سر دیگر طیف — Claude Code را از **کد خودتان** اجرا می‌کنید و **کنترل کامل** دارید.

۷. GitHub Actions و بازبینی کد

بهترین جا برای واگذاری کار تکراری، **Pull Request** است — جایی که بازبینی انجام می‌شود و بیشتر کارهای تکراری آنجاست. دو راه:

1. **مسیر مدیریت‌شده — Code Review:** سرویس میزبانی‌شده توسط Anthropic که PRها را از طریق اپ گیت‌هاب Claude بازبینی می‌کند. **چیزی برای ساخت یا میزبانی نیست.** روشنش می‌کنید و شروع می‌کند به گذاشتن کامنت‌های inline دقیقاً روی خطوط مهم. ادمین سازمان فعالش می‌کند.
2. **GitHub Action:** خودتان سیم‌کشی می‌کنید — انعطاف بیشتر، کار بیشتر.

۸. اعتماد کن: راستی‌آزمایی اجراهای بدون نظارت «مهم»

کلود می‌گوید کار تمام است. قبل از ship کردن، باید چیزی را بررسی کنید که **حتی نظارتش نکردید**.

> **اصل کلیدی: به همان اندازه‌ای راستی‌آزمایی کن که به اجرا آزادی دادی. هرچه کمتر تماشا کردی، بیشتر راستی‌آزمایی کن.**

- جلسهٔ کوتاهی که پیام‌هایش را دیدید → یک نگاه سریع کافی است.
- اجرای بدون نظارت یا کاری که در CI بدون حضور کسی اجرا شده → **بررسی واقعی** لازم دارد. کسی ندیده چه شد، پس باید بعد از واقعه بازسازی‌اش کنید.
- اجراهای بدون نظارت را در **حالت auto** نگه دارید.

۹. پلاگین‌ها (Plugins)

مشکل: یک دایرکتوری `.claude` عالی با اسکیل‌ها، ساب‌ایجنت‌ها و هوک‌ها می‌سازید — بعد چه؟ همه فایل‌ها را بین دستگاه‌ها کپی‌پیست می‌کنند و امیدوارند همگام بماند.

پلاگین = یک واحد نصب‌شدنی که همهٔ آنچه را دستی به اشتراک می‌گذاشتید بسته‌بندی می‌کند: - اسکیل‌ها - ساب‌ایجنت‌ها - هوک‌ها - کانفیگ سرورهای MCP - و موارد جانبی دیگر

دو سمت دارد: **استفاده از پلاگین‌های دیگران** و **بسته‌بندی پلاگین خودتان** وقتی چیز ارزشمندی ساختید.

۱۰. آزمون دوره

نکات کلیدی برای مرور سریع

- CLAUDE.md را **لاغر** نگه دارید — هر خط اضافه، اثر بقیه را کم می‌کند.
- اولین اسکیلی که می‌سازید باید **اسکیل راستی‌آزمایی** باشد.
- توضیحات اسکیل‌ها ارزان‌اند؛ فقط هنگام نیاز کامل بارگذاری می‌شوند.
- برای تضمین قطعی → **هوک**، نه پرامپت.
- برای اجرای بدون نظارت → **حالت auto** + راستی‌آزمایی متناسب.
- برای خودکارسازی بدون کدنویسی → **روتین**؛ برای کنترل کامل → **headless/Agent SDK**.
- برای توزیع کل ستاپ در تیم → **پلاگین**.

دوره ۴ — Introduction to Subagents (آشنایی با ساب‌ایجنت‌ها)

موضوع: استفاده و ساخت ساب‌ایجنت در Claude Code برای مدیریت زمینه، واگذاری وظایف و ساخت جریان‌های کاری تخصصی. **تعداد درس:** ۴

خلاصهٔ کلی دوره

ساب‌ایجنت‌ها **دستیارهای تخصصی** هستند که Claude Code می‌تواند کارها را به آن‌ها واگذار کند. هر کدام در **پنجرهٔ زمینهٔ گفت‌وگوی مستقل خودش** اجرا می‌شود، کارش را انجام می‌دهد، و یک **خلاصه** به رشتهٔ اصلی برمی‌گرداند. مراحل میانی — همهٔ خواندن فایل‌ها، جست‌وجوها و فراخوانی ابزارها — ایزوله می‌مانند و هرگز گفت‌وگوی اصلی شما را شلوغ نمی‌کنند.

نکات مهم هر درس

۱. ساب‌ایجنت چیست؟

چرا اهمیت دارد: هر بار که با Claude Code چت می‌کنید، به پنجرهٔ زمینهٔ اصلی اضافه می‌شود. هر فراخوانی ابزار، هر خواندن فایل، هر نتیجهٔ جست‌وجو آنجا ذخیره می‌شود. **این فضا محدود است** و وقتی پر شود، کلود بخش‌های قبلی گفت‌وگو را از دست می‌دهد.

ساب‌ایجنت با **راه‌اندازی یک پنجرهٔ زمینهٔ جداگانه** این مشکل را حل می‌کند.

سه مزیت اصلی: 1. کار را به **قطعات متمرکز** تقسیم می‌کنند — هر ساب‌ایجنت روی یک وظیفهٔ مشخص تمرکز می‌کند. 2. پنجرهٔ زمینهٔ اصلی شما را **تمیز نگه می‌دارند** با ایزوله‌کردن همهٔ کارهای میانی. 3. **فقط اطلاعات مورد نیازتان** را به شکل خلاصهٔ فشرده برمی‌گردانند.

> هرچه سروصدا در زمینهٔ اصلی کمتر باشد، طولانی‌تر و مؤثرتر می‌توانید کار کنید.

۲. ساخت یک ساب‌ایجنت

Claude Code ساب‌ایجنت‌های **داخلی** دارد، اما می‌توانید خودتان هم بسازید — مثلاً برای بازبینی کد، نوشتن تست، یا بررسی مستندات.

ساختار: فایل‌های **Markdown با frontmatter از نوع YAML** که به کلود می‌گویند **چه زمانی** از ساب‌ایجنت استفاده کند و **چطور** رفتار کند.

روش ساخت: 1. دستور اسلش `/agents` — رابط اصلی مدیریت ساب‌ایجنت‌ها را باز می‌کند. 2. گزینهٔ **Create new agent** را انتخاب کنید. 3. **دامنه (scope)** را انتخاب کنید: - **Project-level** — فقط در پروژهٔ فعلی در دسترس - **User-level** — در همهٔ پروژه‌های دستگاه شما مشترک

۳. طراحی ساب‌ایجنت‌های مؤثر «مهم»

یک ساب‌ایجنت با پیکربندی ضعیف **سرگردان می‌شود، بیش‌ازحد طول می‌کشد، یا خروجی‌ای می‌دهد که عامل اصلی نمی‌تواند استفاده کند.**

چهار عامل اصلاح:

۱) نوشتن توضیحات خوب (descriptions) - وقتی به عامل اصلی پیام می‌دهید، **نام و توضیح همهٔ ساب‌ایجنت‌های موجود** در پرامپت سیستمی گنجانده می‌شود. - عامل اصلی **بر همین اساس** تصمیم می‌گیرد کدام ساب‌ایجنت را و کِی اجرا کند. - پس اگر می‌خواهید کنترل بهتری روی زمان فعال‌شدن خودکار داشته باشید، **نام و توضیح** همان چیزی است که باید رویش کار کنید.

۲) تعریف قالب خروجی (output format) — تا عامل اصلی بتواند از نتیجه استفاده کند.

۳) گزارش موانع (obstacles) — ساب‌ایجنت باید بگوید کجا گیر کرده.

۴) محدودکردن دسترسی ابزارها (tool access) — مثلاً بازبین کد را فقط-خواندنی کنید.

۴. استفادهٔ مؤثر از ساب‌ایجنت‌ها

سؤال اصلی: کِی کمک می‌کنند و کِی سر راه قرار می‌گیرند؟

> **پاسخ به یک چیز برمی‌گردد: آیا کارِ میانی برای رشتهٔ اصلی شما اهمیت دارد یا نه.**

کِی ساب‌ایجنت می‌درخشد — وقتی **کاوش از اجرا جداست**: - شما به یک **نتیجه** نیاز دارید، نه به گزارش لحظه‌به‌لحظهٔ اینکه چطور پیدا شد - کار اکتشافی، زمینهٔ رشتهٔ اصلی را شلوغ می‌کند - کار از **دیدگاه تازه** یا یک پیکربندی سفارشی سود می‌برد

کِی ساب‌ایجنت مناسب نیست: - اگر **هر قدم به کشف قدم قبلی وابسته است**، آن کار را در رشتهٔ اصلی نگه دارید.

نکات کلیدی برای مرور سریع

- ساب‌ایجنت = زمینهٔ ایزوله + خلاصهٔ برگشتی.
- `/agents` برای ساخت و مدیریت.
- دامنه: Project-level یا User-level.
- **نام + توضیح** تعیین می‌کند کِی خودکار فعال شود.
- ابزارها را محدود کنید (بازبین = فقط-خواندنی).
- **قانون سرانگشتی:** فقط نتیجه را می‌خواهید → واگذار کنید. هر قدم به قدم قبل وابسته است → در رشتهٔ اصلی بمانید.

دوره ۵ — Introduction to Agent Skills (آشنایی با اسکیل‌ها)

موضوع: ساخت، پیکربندی و اشتراک‌گذاری اسکیل‌ها در Claude Code — دستورالعمل‌های Markdown قابل استفادهٔ مجدد که کلود خودکار در زمان مناسب اعمال می‌کند. **تعداد درس:** ۶ (عمدتاً ویدیویی)

خلاصهٔ کلی دوره

اسکیل = **فایل Markdown قابل استفادهٔ مجدد** که به Claude Code یاد می‌دهد یک کار مشخص را چطور خودکار انجام دهد. به‌جای تکرار دستورالعمل‌ها هر بار که می‌خواهید کلود یک PR را بازبینی کند یا پیام کامیت بنویسد، **یک بار اسکیل را می‌نویسید** و کلود هر وقت آن کار پیش آمد اعمالش می‌کند.

نکات مهم هر درس

۱. اسکیل چیست؟ (~۱۵ دقیقه)

اهداف یادگیری: - تعریف اسکیل‌های Claude Code و نحوهٔ کارشان - **کجا زندگی می‌کنند:** دایرکتوری شخصی (personal) در برابر دایرکتوری پروژه (project) - تفاوت **اسکیل** با **CLAUDE.md** و **دستورات اسلش (slash) commands)** - تشخیص موقعیت‌هایی که اسکیل ابزار درست شخصی‌سازی است

۲. ساخت اولین اسکیل (~۲۰ دقیقه)

اهداف یادگیری: - ساخت اسکیل از صفر با **ساختار frontmatter** صحیح - تست و تأیید اینکه اسکیل درست بارگذاری می‌شود - اینکه Claude Code چطور **درخواست‌های ورودی را با اسکیل‌های موجود تطبیق می‌دهد** - **سلسله‌مراتب اولویت اسکیل‌ها:**

> **Enterprise ← Personal ← Project ← Plugins**

مثال عملی ویدیو: ساخت یک اسکیل شخصی «توضیح PR» که در همهٔ پروژه‌ها کار می‌کند — ساختاردهی فایل `SKILL.md` ، تستش، و فهم نحوهٔ کشف و تطبیق اسکیل‌ها.

۳. پیکربندی و اسکیل‌های چندفایلی (~۲۰ دقیقه)

تکنیک‌های پیشرفته‌ای که اسکیل‌ها را قدرتمندتر می‌کند:

- **فیلدهای متادیتای پیشرفته:** `allowed-tools` و `model`
- **نوشتن توضیحاتی که قابل‌اتکا فعال (trigger) می‌شوند** — روی درخواست‌های درست
- **`allowed-tools` :** محدودکردن کارهایی که کلود هنگام فعال‌بودن اسکیل می‌تواند انجام دهد (برای جریان‌های کاری حساس از نظر امنیتی)
- **افشای تدریجی (progressive disclosure):** سازمان‌دهی اسکیل‌های پیچیده در **ساختار چندفایلی**

۴. اسکیل در برابر سایر قابلیت‌های Claude Code «مهم»

انتخاب گزینهٔ اشتباه منجر به پیچیدگی غیرضروری می‌شود.

اهداف یادگیری: - مقایسهٔ اسکیل با **CLAUDE.md، ساب‌ایجنت، هوک و سرور MCP** - انتخاب قابلیت درست برای هر مورد استفاده - طراحی ستاپی که چند قابلیت را **مکمل هم** ترکیب کند

نکتهٔ کلیدی مطرح‌شده: «CLAUDE.md در **هر** زمینه بارگذاری می‌شود» — یعنی هزینهٔ دائمی دارد، برخلاف اسکیل که فقط توضیحش بارگذاری می‌شود.

راهنمای انتخاب (جمع‌بندی از کل دوره‌ها):

نیاز	ابزار درست
زمینهٔ دائمی پروژه (استک، دستورات)	CLAUDE.md — کوتاه نگه دارید
رویهٔ تکرارشونده و مشروط	Skill
کار سنگین اکتشافی با زمینهٔ جدا	Subagent
تضمین قطعی، همیشه اجرا شود	Hook
اتصال به داده/ابزار بیرونی	MCP

۵. اشتراک‌گذاری اسکیل‌ها (~۲۰ دقیقه)

اسکیل‌ها وقتی در تیم یا سازمان به اشتراک گذاشته شوند بسیار ارزشمندتر می‌شوند.

سه روش اصلی توزیع: 1. **Commit در ریپازیتوری Git** — برای تیم پروژه 2. **پلاگین‌ها و مارکت‌پلیس‌ها** — برای توزیع بین پروژه‌ها 3. **تنظیمات مدیریت‌شدهٔ سازمانی (enterprise managed settings)** — برای استقرار در کل سازمان

«هشدار» نکتهٔ مهم (gotcha): ساب‌ایجنت‌ها به‌صورت خودکار اسکیل‌ها را به ارث نمی‌برند — باید ساب‌ایجنت‌های سفارشی را طوری پیکربندی کنید که از اسکیل‌های مشخصی استفاده کنند.

۶. عیب‌یابی اسکیل‌ها (~۱۵ دقیقه)

وقتی اسکیل‌ها طبق انتظار کار نمی‌کنند، مشکل معمولاً در چند دستهٔ قابل پیش‌بینی است:

- **اعتبارسنج اسکیل (skills validator):** برای گرفتن مشکلات **ساختاری** قبل از شروع دیباگ
- **مشکلات فعال‌سازی (triggering) و بارگذاری** — اسکیل اصلاً فعال نمی‌شود
- **تعارض اولویت** بین اسکیل‌های project، personal، enterprise و plugin
- **خطاهای زمان اجرا:** وابستگی‌های نصب‌نشده، مشکلات مجوز، و مسیرهای اشتباه (path issues)

نکات کلیدی برای مرور سریع

- اسکیل = فایل `SKILL.md` + frontmatter (نام، توضیح، و اختیاری `allowed-tools` و `model`).
- **فقط توضیح** در زمینه بارگذاری می‌شود → ساخت اسکیل زیاد هزینه‌ای ندارد.

- **توضیح** مهم‌ترین بخش است — تعیین می‌کند اسکیل کِی فعال شود.
- اولویت: Plugins → Project → Personal → Enterprise.
- برای اسکیل‌های بزرگ از **افشای تدریجی** و ساختار چندفایلی استفاده کنید.
- ساب‌ایجنت‌ها اسکیل‌ها را خودکار به ارث نمی‌برند.
- برای عیب‌یابی، اول **validator** را اجرا کنید.

دوره ۶ — Introduction to Claude Cowork (آشنایی با کلود کووُرک)

موضوع: کار کردن در کنار کلود روی فایل‌ها و پروژه‌های واقعی — حلقهٔ وظیفه، پلاگین‌ها و اسکیل‌ها، جریان‌های کاری فایل و تحقیق، و هدایت مسئولانهٔ کارهای چندمرحله‌ای. **تعداد درس:** ۱۵

خلاصهٔ کلی دوره

Cowork یعنی **کلود در محیط شما** — روی فایل‌های شما، در اپ‌های شما، با ابزارهای شما. این یک چت‌بات باهوش‌تر نیست؛ **شیوهٔ کاملاً متفاوتی از کار با کلود است.**

نکات مهم هر درس

ماژول ۱ — آشنایی با Cowork

۱. Cowork چیست؟ (۸ دقیقه)

- تفاوت **Chat**، **Cowork** و **Code** را بشناسید و بدانید کِی سراغ کدام بروید.
- Cowork نه به‌عنوان چت‌بات، بلکه به‌عنوان **روش دیگری از کار** عمل می‌کند.

۲. راه‌اندازی Cowork (۸ دقیقه)

- Cowork را در **اپ دسکتاپ Claude روی مک و ویندوز** اجرا کنید، یا **در فضای ابری** (بتا، پلن‌های واجد شرایط) از وب و اپ موبایل.
- روی دسکتاپ، کلود مستقیماً با **فایل‌های محلی** شما کار می‌کند.
- Cowork را به یک **پوشهٔ کاری** وصل کنید.
- اپ‌هایی که کارتان در آن‌هاست را متصل کنید.
- **مهم:** بشناسید کلود **قبل از چه کارهایی می‌پرسد** و قبل از چه کارهایی نمی‌پرسد — تا با اطمینان کار را واگذار کنید.

۳. Cowork چه کاری برایتان انجام می‌دهد (۱۱ دقیقه)

سه الگوی کاری که مناسب Cowork است: 1. کارهای **چندمرحله‌ای** (multi-step) 2. کارهای **مبتنی بر فایل** (file-based) 3. کارهای **چندابزاری** (multi-tool)

- `/schedule` — تنظیم وظیفهٔ زمان‌بندی‌شده.
- اجرای وظایف در **فضای ابری** (بتا) و شروع یا بررسی آن‌ها از گوشی یا هر مرورگری.

> هر کاری مناسب Cowork نیست؛ وقتی الگوها را بشناسید، بازدهی بیشتری می‌گیرید.

۴. اولین وظیفه را به Cowork بسپارید (۱۵ دقیقه)

یک کار واقعی از ابتدا تا انتها: **واگذاری → هدایت میان‌کار → بازبینی نهایی**. - به سؤالات شفاف‌سازی کلود طوری پاسخ دهید که خروجی بهتری تولید شود. - وقتی کلود مسیر را اشتباه رفت، **مسیر را اصلاح کنید** (course-correct). - **ریتم کار** را حس کنید — کمتر شبیه پرامپت‌زدن است.

ماژول ۲ — Cowork را مال خودتان کنید

۵. نتایج بهتر و سریع‌تر (۵ دقیقه)

چهار بلوک سازنده که کلود را به‌مرور در کار شما بهتر می‌کند: 1. **دستورالعمل‌های سراسری (global instructions)** 2. **پروژه‌ها (projects)** 3. **اسکیل‌ها (skills)** 4. **پلاگین‌ها (plugins)**

> هرچه کلود دربارهٔ شما، کارتان و روش تیمتان بیشتر بداند، خروجی بعدی بهتر است — و سریع‌تر به آن می‌رسید. **این اثر انباشتی (compounding) دارد.**

۶. زمینهٔ ثابت: دستورالعمل‌های سراسری و پروژه‌ها (۷ دقیقه)

تفاوت مهم با Chat: - در **Chat** حافظه **خودش ساخته می‌شود** — روشنش می‌کنید و کلود از گفت‌وگوها یاد می‌گیرد. - در **Cowork** فرق دارد: زمینه‌ای که از کاری به کار دیگر منتقل می‌شود، عمدتاً **زمینه‌ای است که خودتان تنظیم می‌کنید**.

- **دستورالعمل‌های سراسری:** روی **هر** جلسهٔ Cowork اعمال می‌شود — کلود هر جلسه را با دانستن روش کار شما شروع می‌کند.
- تصمیم بگیرید چه کاری متعلق به یک **پروژه** است.
- سه راه برای شروع یک پروژه وجود دارد.

۷. اسکیل‌ها: روش خودتان را به Cowork یاد بدهید (۱۲ دقیقه)

- **اسکیل = دفترچهٔ راهنمای قابل استفادهٔ مجدد (playbook)** — پوشه‌ای از فایل‌ها و منابع که به کلود یاد می‌دهد یک نوع کار مشخص را **آن‌طور که شما می‌خواهید** انجام دهد.
- وقتی کاری منطبق با اسکیل شروع می‌کنید، کلود playbook را بارگذاری و دنبال می‌کند.
- **اسکیل‌ها خودکار و دقیقاً وقتی لازم‌اند استفاده می‌شوند.**
- یک اسکیل **چهار بلوک سازنده** می‌تواند داشته باشد.
- تمرین: یک اسکیل از یکی از فرایندهای تکراری خودتان بسازید.

۸. پلاگین‌ها: تخصص تیم را کدگذاری کنید (۱۲ دقیقه)

- **پلاگین = مجموعهٔ بسته‌بندی‌شده‌ای از اسکیل‌ها** که حول یک «شغل/کار» ساخته شده.
- **تفاوت با اسکیل:** اسکیل **یک** playbook است؛ پلاگین **چندتاست** — به‌علاوهٔ **کانکتورها و ساب‌ایجنت‌هایی** که به آن‌ها وابسته‌اند.
- پلاگین‌ها **دو شکل** دارند.
- وقتی جریان کاری یک نفر به **جعبه‌ابزار مشترکی** تبدیل می‌شود که هرکسی می‌تواند نصب کند، همه‌چیز تغییر می‌کند.

ماژول ۳ — گسترش دامنه

۹. Claude in Chrome (۱۰ دقیقه)

- کلود می‌تواند در کروم **بخواند، کلیک کند و در سایت‌ها پیمایش کند**.
- **Claude in Chrome پلی است برای ابزارهایی که کانکتور ندارند.** هر چیزی که در مرورگر زندگی می‌کند، کلود می‌تواند بخواند و رویش اقدام کند.
- **Chrome و Cowork با هم کار می‌کنند:** کلود در مرورگر اطلاعات جمع می‌کند و اقدام انجام می‌دهد، سپس نتایج را می‌گیرد و در Cowork چیزی می‌سازد.

۱۰. Claude for Microsoft 365 (۵ دقیقه)

- کلود به‌صورت **افزونه (add-in)** داخل **PowerPoint، Excel، Word و Outlook** ظاهر می‌شود و روی فایل‌ها و فضاهای بازِ شما کار می‌کند.
- **یک گفت‌وگو می‌تواند زمینه را بین اپ‌ها منتقل کند** — تحلیل را در Excel بسازید و به اپ بعدی تحویل دهید.
- تصمیم بگیرید کار را در **Cowork** انجام دهید یا **داخل خود سند**.

ماژول ۴ — کار ایمن و اشتراک‌گذاری

۱۱. بهترین شیوه‌ها برای کار ایمن (۸ دقیقه)

کف امنیت (که خود Cowork تضمین می‌کند): - کلود **همیشه قبل از حذف می‌پرسد**. - در **حالت مجوز پیش‌فرض**، قبل از **ارسال یا اشتراک‌گذاری** هم می‌پرسد.

آنچه شما باید اضافه کنید: - **فضای کاری‌تان را طوری تنظیم کنید که چیزهای مهم محافظت شوند.** - پرامپت‌هایی بنویسید که **جا برای اقدام اشتباه باقی نگذارند**. - لحظاتی را بشناسید که **«توقف و تفکر» از سرعت مهم‌تر است**.

۱۲. اعتبارسنجی اسکیل‌ها برای پلاگین‌ها (۸ دقیقه)

- **eval چیست و چرا قبل از اشتراک‌گذاری یا اتکا به یک اسکیل اهمیت دارد.**
- وقتی اسکیل می‌سازید یا در پلاگین بسته‌بندی می‌کنید، در واقع **یک محصول کوچک** می‌سازید که دیگران استفاده می‌کنند.
- مثل هر چیزی که به همکار تحویل می‌دهید (قالب، مدل اکسل، چک‌لیست) **ارزش یک تست‌درایو را دارد** قبل از اینکه میزتان را ترک کند.
- **نکته:** وقتی از اسکیلی که خودتان ساختید استفاده می‌کنید، می‌دانید چطور دور مشکلاتش بچرخید — دیگران نمی‌دانند.
- اجرای یک **eval سبک** از طریق `skill-creator`.

۱۳. آنچه می‌سازید را با تیم به اشتراک بگذارید (۷ دقیقه)

- مسیر طبیعی: اسکیل‌ها **از روش شخصی یک نفر شروع می‌شوند** → از **eval** رد می‌شوند → روی موارد استفادهٔ بیش از یک نفر **دوام می‌آورند**.
- برای مقیاس‌دادن به کل تیم، آن‌ها را در یک **پلاگین** بسته‌بندی کنید.

- نحوهٔ **توزیع پلاگین در یک سازمان Enterprise**.
- عادت‌های خوب برای **سالم نگه‌داشتن یک پلاگین مشترک** در طول زمان.

۱۴ و ۱۵. جمع‌بندی، قدم‌های بعدی و آزمون

نکات کلیدی برای مرور سریع

- Cowork = کلود در **محیط شما** (فایل‌ها، اپ‌ها، ابزارها).
- سه الگوی مناسب: **چندمرحله‌ای، فایل‌محور، چندابزاری**.
- برخلاف Chat، **حافظه خودکار ساخته نمی‌شود** — شما زمینه را تنظیم می‌کنید.
- چهار بلوک سازنده: **دستورالعمل سراسری → پروژه → اسکیل → پلاگین** (به‌ترتیب افزایش ساختار).
- **اسکیل = یک playbook؛ پلاگین = چند playbook + کانکتور + ساب‌ایجنت.**
- کلود **همیشه قبل از حذف** می‌پرسد؛ در حالت پیش‌فرض قبل از **ارسال/اشتراک** هم.
- قبل از اشتراک‌گذاری اسکیل، **eval** بگیرید.
- `/schedule` برای وظایف تکرارشونده.

دوره ۷ — Claude Platform 101 (مقدمات پلتفرم کلود)

موضوع: ساخت روی پلتفرم کلود از پایه — برای توسعه‌دهندگانی که چند فراخوانی API انجام داده‌اند یا فقط از پنجرهٔ چت استفاده کرده‌اند. **تعداد درس:** ۱۴

خلاصهٔ کلی دوره

پلتفرم کلود، زیرساخت Anthropic برای **ساخت برنامه‌نویسی‌شده (programmatic)** با کلود است. به‌جای چت در مرورگر، درخواست‌های **ساختاریافته** از کد خود می‌فرستید و پاسخ ساختاریافته می‌گیرید — با کنترل روی هر جزئیات: کدام مدل، چند توکن، کلود چه ابزارهایی داشته باشد، و چه دستورالعمل سیستمی دنبال کند.

اجزای پلتفرم: - یک **REST API** که از هر زبانی قابل فراخوانی است - **SDK** برای زبان‌های مختلف - **رابط‌های خط فرمان (CLI)** - یک **کنسول** برای مدیریت کلیدهای API، پایش مصرف و استقرار عامل‌های مدیریت‌شده

نکات مهم هر درس

۱. پلتفرم کلود چیست؟

(بالا توضیح داده شد.)

۲. اولین فراخوانی API

راه‌اندازی: 1. کلید API را از `platform.claude.com` بگیرید (نیاز به خرید اعتبار دارید). 2. کلید را در فایل `.env.local` ذخیره کنید تا از version control بیرون بماند.

> **هاردکد کردن کلید در سورس، همان راهی است که کلیدها روی GitHub لو می‌روند.** 3. نصب SDK:
> `npm install @anthropic-ai/sdk`

آناتومی یک درخواست: - تابع `messages.create` با: - `model` (مدل) - محدودیت توکن (`max_tokens`) - `messages` (پیام‌ها) - **system prompt** را اضافه کنید تا رفتار کلود را شکل دهید. - **پاسخ، آرایه‌ای از بلوک‌هاست (blocks)** — روی آن‌ها حلقه بزنید و `type` هر بلوک را بررسی کنید.

> از اینجا به بعد، همه‌چیز روی همین الگو ساخته می‌شود.

۳. انتخاب مدل درست «مهم»

> اگر پیش‌فرض را باهوش‌ترین مدل بگذارید، صورتحساب API غافلگیرتان می‌کند. ارزان‌ترین را بگذارید، ممکن است خروجی کافی نباشد.

سطوح مدل:

مدل	کاربرد
Opus	مسائل سخت
Sonnet	کار روزمره
Haiku	حجم بالا (volume)

انتخاب با پارامتر `model` در فراخوانی API.

روش عملی انتخاب: 1. یک **ارزیابی ساده (eval)** بسازید — **۲۰ تا ۳۰ نمونهٔ نماینده** از بار کاری واقعی‌تان — **قبل از** نوشتن کد پروداکشن. 2. eval را **از Haiku به بالا** اجرا کنید. 3. **در ارزان‌ترین مدلی متوقف شوید که خروجی‌اش را واقعاً حاضرید ship کنید.**

هزینه: `response.usage` توکن‌های ورودی و خروجی را گزارش می‌کند — صورتحساب بر همین اساس است.

در پروداکشن: به‌جای انتخاب یک مدل برای همه‌چیز، **وظایف مختلف را به مدل‌های مختلف مسیریابی کنید** — حتی داخل یک endpoint.

۴. حلقهٔ عامل (Agent Loop) توضیح داده شد «مهم»

یک فراخوانی تنها، فقط یک پاسخ برمی‌گرداند. برای خودکارسازی یک جریان کاری، کلود باید **اقدام کند، نتیجه را ببیند، تصمیم بگیرد بعد چه، و ادامه دهد.**

عامل چیست؟ نسخهٔ خودمختار کلود که **هر دو طرف حلقهٔ پیام‌رسانی را بدون انسان در وسط** اجرا می‌کند. عامل یک وظیفه می‌گیرد، ابزاری انتخاب می‌کند، و کد را در یک حلقه اجرا می‌کند تا کلود تصمیم بگیرد کار تمام است.

پیاده‌سازی سادهٔ حلقه: 1. پیامی همراه با **tools** به کلود بفرست 2. هر ابزاری که کلود درخواست کرد را **اجرا کن** 3. **نتیجه را برگردان** به کلود 4. **وقتی `stop_reason` برابر `end_turn` شد، متوقف شو**

> **شما مالک حلقه و ابزارها هستید. کلود مالک استدلال است.**

- همین شکل حلقه از یک دموی ساختگی آب‌وهوا تا یک عامل انطباق (compliance) پروداکشن **مقیاس می‌گیرد** — فقط ابزارها و لوله‌کشی عوض می‌شود.
- اگر نمی‌خواهید مالک حلقه باشید، **عامل‌های مدیریت‌شده (managed agents)** دقیقاً همین حلقه را برایتان اجرا می‌کنند.

۵. استفاده از ابزار (Tool Use) چیست؟

- **ابزار = تابعی که شما تعریف و به کلود معرفی می‌کنید.** توصیف می‌کنید چه کار می‌کند و چه ورودی‌هایی می‌گیرد؛ کلود تصمیم می‌گیرد کِی صدایش بزند.

> **«مهم» نکتهٔ کلیدی که باید درونی کنید: کلود ابزار را اجرا نمی‌کند — کد شما اجرا می‌کند.**

جریان کار: 1. کلود یک **فراخوانی ابزار** درخواست می‌کند. 2. **کد شما** تابع را اجرا می‌کند. 3. نتیجه به کلود برمی‌گردد.

۶. Thinking (تفکر گسترده) چیست؟

حالت شکستی که می‌خواهیم از آن اجتناب کنیم: یک سؤال چندمرحله‌ای بپرسید و مدل فوراً جواب دهد — با اطمینان اشتباه می‌کند.

- **Extended thinking** به کلود اجازه می‌دهد **قدم‌به‌قدم استدلال کند** قبل از تولید پاسخ نهایی.
- توکن‌های استدلال داخلی تولید می‌کند (**زنجیرهٔ تفکر / chain of thought**) و بعد پاسخ می‌دهد. **استدلال در پاسخ قابل مشاهده است.**

فعال‌سازی (با Opus 4.7):

```
thinking: { "type": "adaptive" }
```

> بودجهٔ توکن لازم نیست — **خود کلود تصمیم می‌گیرد کِی و چقدر فکر کند.**

تنظیم عمق با پارامتر `effort` داخل `output_config`: `low` • `medium` • `high` (پیش‌فرض) • `xhigh` • `max`

کِی استفاده کنیم؟ مسائل سخت و پر از trade-off. **کِی نه؟** مسائل ساده — آنجا فقط **تأخیر و توکن** هزینه می‌کند.

۷. ابزارهای داخلی (Built-in Tools)

برخی قابلیت‌ها آن‌قدر رایج‌اند که Anthropic از پیش ساخته‌شان. **کد نمی‌نویسید، سندباکس میزبانی نمی‌کنید — فقط ابزار را اعلام می‌کنید و Anthropic اجرایش می‌کند.**

ابزارهای سرور (Server tools) — اعلام‌شده توسط شما، اجراشده توسط Anthropic: **Web search** — اینترنت را جست‌وجو می‌کند و نتایج را با **استناد (citation)** برمی‌گرداند - **Code execution** - **Web fetch**

> **برای این‌ها به حلقهٔ عامل نیاز ندارید.** کلود خودش ابزارها را صدا می‌زند و نتیجه **در همان پاسخ** برمی‌گردد. دنبال بلوک‌های `server_tool_use` و tool result کنار بلوک‌های متنی معمولی بگردید.

ابزارهای کلاینت (Client tools) مثل `memory` و `bash` **جایی که کد شما اجرا می‌شود** اجرا می‌شوند — اما SDK **schema و runner** را برایتان فراهم می‌کند.

> ایدهٔ «میزبانی‌شده توسط Anthropic» تا بالاترین سطح مقیاس می‌گیرد: **عامل‌های مدیریت‌شده** آن را به **کل عامل** اعمال می‌کنند، نه فقط یک ابزار.

۸. اسکیل‌ها (Skills)

- پوشه‌هایی از دستورالعمل‌ها، اسکریپت‌ها و منابع که کلود **به‌صورت پویا** بارگذاری می‌کند.
- هستهٔ هر اسکیل، فایل `SKILL.md` است — **یک بار آپلود می‌کنید** و بعد به هر فراخوانی `messages.create` **پیوست** می‌کنید.
- شما به کلود یاد می‌دهید **کارها را چطور انجام می‌دهید:** قالب گزارش وضعیت، چک‌لیست بازبینی، یادداشت‌های انتشار.

تفاوت اسکیل با ابزار:

	کاربرد
ابزار (Tool)	کلود را به **داده و اقدام** وصل می‌کند
اسکیل (Skill)	به کلود **رویه و شکل خروجی** را یاد می‌دهد

۹. MCP

سؤال: ما ابزار، اسکیل و کانکتور داریم. پس چرا MCP وجود دارد؟

پاسخ به یک چیز برمی‌گردد: چه کسی کد یکپارچه‌سازی را نگهداری می‌کند.

مشکل نگهداری: فرض کنید عامل شما باید تسک‌ها را از Asana بکشد، Google Calendar را چک کند و Slack را جست‌وجو کند. با ابزارهای سفارشی باید **سه یکپارچه‌سازی** بنویسید — این بخشش شدنی است. **بخش دردناک بعدش است:** باید هر بار که یکی از این سرویس‌ها API خود را تغییر می‌دهد (که مکرر اتفاق می‌افتد) آن‌ها را نگهداری کنید.

> MCP این بار نگهداری را از دوش شما برمی‌دارد.

۱۰. مدیریت زمینه (Context Management)

> یک میلیون توکن زیاد به نظر می‌رسد، اما وقتی یک عامل واقعی را ship می‌کنید، **سریع‌تر از آنچه فکر می‌کنید تمام می‌شود.**

چه چیزهایی زمینه حساب می‌شوند؟ - پرامپت سیستمی - تاریخچهٔ پیام‌ها - تعریف ابزارها و **نتایج ابزارها** - فایل‌های پیوست و اسکیل‌ها - بلوک‌های thinking

> این ورودی **هر** فراخوانی API است. **هم موقع ورود پول می‌دهید، هم موقع خروج.** و وقتی پنجره پر شود، درخواست شکست می‌خورد.

۱۱. عامل‌های مدیریت‌شده (Managed Agents) چیستند؟

- **Claude Managed Agents** مجموعه‌ای از APIها برای **ساخت و استقرار عامل در مقیاس بزرگ**.
- شما عامل‌ها را با **ابزارها، پرسوناها و قابلیت‌های مشخص** تعریف می‌کنید.
- **محیط‌های سندباکس** را با پکیج‌های درست و کنترل‌های شبکه پیکربندی می‌کنید.
- سپس از اپلیکیشن خودتان **session** شلیک می‌کنید و کلود کار را داخل یک **کانتینر ایزوله** انجام می‌دهد — با دسترسی کامل به فایل‌سیستم، اجرای bash، و جست‌وجوی وب.
- **زیر کاپوت، همان حلقهٔ عامل است — اما میزبانی‌شده برای شما.**

۱۲. ساخت اولین عامل مدیریت‌شده

اگر حلقهٔ عامل را دستی ساخته باشید، این‌ها را می‌شناسید: حلقهٔ `while`، سوئیچ روی `stop_reason`، اجرای ابزارها. **این کار می‌کند و برای خیلی از قابلیت‌ها شکل درستی است.**

اما گاهی حلقه قرار است خیلی طولانی اجرا شود — دقایق، شاید ساعت‌ها — در طول ابزارهای متعدد، با **وضعیتی که باید حفظ شود**، فایل‌هایی که باید نوشته شوند، و کاری که باید **بعد از قطعی شبکه از سر گرفته شود**.

> در آن نقطه، نمی‌خواهید حلقه روی سرور شما اجرا شود. **می‌خواهید واگذارش کنید.** این همان عامل مدیریت‌شده است.

۱۳. ساخت با Claude Code

- نوشتن دستی کد فراخوانی API خوب است، اما مسیر سریع‌تری هم هست: **بگذارید کلود برایتان بنویسد.**
- **مثال عملی:** یک فایل TypeScript که آب‌وهوا می‌گیرد، با دو stub:
- `getWeather` — شهر می‌گیرد، دما و شرایط برمی‌گرداند
- `run` — باید از **tool runner** و SDK تایپ‌اسکریپت کلود استفاده کند
- **tool runner** قطعه‌ای است که فراخوانی ابزار را مدیریت می‌کند.

نکات: - اسکیل داخلی **Claude API** وقتی Claude Code وجود SDK تایپ‌اسکریپت را تشخیص دهد **خودکار بارگذاری می‌شود** — یا با `/claude-api` صدایش بزنید. - پرامپتی بدهید که **نام فایل، الگو و وضعیت نهایی** را مشخص کند — کلود کد را می‌نویسد، اجرا می‌کند و خطاها را همان‌جا رفع می‌کند. - **شکل آشنای کد Claude API:** یک ابزار تعریف کن → به runner بدهش → نتیجه را برگردان. - **روش کار: Stub کن، واگذار کن، diff را بازبینی کن.**

۱۴. آزمون دوره

نکات کلیدی برای مرور سریع

- کلید API را در `.env.local` بگذارید، **هرگز** در سورس.
- `messages.create` = مدل + messages (+ system) + max_tokens.
- پاسخ = **آرایه‌ای از بلوک‌ها**؛ `type` هرکدام را چک کنید.
- **Opus** سخت • **Sonnet** روزمره • **Haiku** حجم بالا. با eval از پایین به بالا انتخاب کنید.
- **حلقهٔ عامل:** بفرست → ابزار را اجرا کن → نتیجه را برگردان → تا `end_turn`.
- **کد شما ابزار را اجرا می‌کند، نه کلود.**
- **ابزارهای سرور** (web fetch، code execution، web search) حلقهٔ عامل نمی‌خواهند.
- `thinking: {type:"adaptive"}` + `effort` برای عمق تفکر.
- **MCP** مشکل **نگهداری یکپارچه‌سازی‌ها** را حل می‌کند.
- **عامل مدیریت‌شده** = حلقهٔ عامل روی زیرساخت Anthropic، برای اجراهای طولانی و باحالت.

دوره ۸ — AI Fluency: Framework & Foundations (سواد هوش مصنوعی: چارچوب و مبانی)

موضوع: همکاری مؤثر، کارآمد، اخلاقی و ایمن با سیستم‌های هوش مصنوعی. **تعداد درس:** ۱۵ **نکته:** این دورهٔ **مادر** خانوادهٔ AI Fluency است؛ نسخه‌های ویژهٔ معلمان، دانشجویان، کسب‌وکارهای کوچک، NGOها و سازندگان همگی بر همین چارچوب بنا شده‌اند.

خلاصهٔ کلی دوره

تمرکز دوره روی **ساختن همکاری معنادار با هوش مصنوعی** است، نه صرفاً استفاده از آن. قلب دوره، **چارچوب ۴D** است.

«مهم» چارچوب ۴D — چهار شایستگی اصلی

شایستگی	تعریف
Delegation (واگذاری)	تصمیم‌گیری اندیشیده دربارهٔ اینکه چه کاری را با AI انجام دهیم و چه کاری را خودمان
Description (توصیف)	ارتباط شفاف با سیستم‌های هوش مصنوعی
Discernment (تشخیص)	ارزیابی نقادانهٔ خروجی‌ها و فرایندهای AI
Diligence (وظیفه‌شناسی)	همکاری مسئولانه و اخلاقی

نکات مهم هر درس

۱. مقدمه‌ای بر AI Fluency (۱۵–۱۰ دقیقه)

- تمرکز دوره: **توسعهٔ همکاری معنادار با AI**، نه فقط استفاده از آن.
- [دانلود] فایل دانلودی: `AI Fluency vocabulary cheat sheet.pdf`

۲. چرا به AI Fluency نیاز داریم؟

سه شیوهٔ نوظهور همکاری با AI: 1. **Automation (خودکارسازی)** — AI کار را انجام می‌دهد 2. **Augmentation (تقویت)** — AI و انسان با هم کار می‌کنند 3. **Agency (عاملیت)** — AI مستقل عمل می‌کند

- «روان بودن» با AI یعنی توسعهٔ **مهارت‌ها، دانش، بینش‌ها و ارزش‌های عملی**.

۳. چارچوب ۴D (۵ دقیقه) «مهم»

- [دانلود] دانلودها: `AI Fluency Summary One-Pager.pdf` (جدول بالا.) - `Framework_for_AI_Fluency_V_1.5.pdf`

۴. مبانی هوش مصنوعی مولد (۶ دقیقه)

- تعریف **AI مولد (generative)** و تفاوتش با سایر انواع AI: توانایی **خلق محتوای جدید** به‌جای صرفاً تحلیل آنچه موجود است.
- نحوهٔ کار مدل‌های زبانی بزرگ (**LLM**) مثل کلود.
- [دانلود] دانلود: `Overview of Generative AI.pdf`

۵. قابلیت‌ها و محدودیت‌ها (۷ دقیقه)

قابلیت‌ها: - **همه‌کاره بودن** در وظایف زبانی - حفظ **جریان گفت‌وگو** - **جابه‌جایی بین وظایف متنوع** بدون آموزش اضافه

محدودیت‌ها: - **تاریخ قطع دانش (knowledge cutoff)** - و سایر محدودیت‌های بررسی‌شده در دوره

۶. نگاه نزدیک‌تر به Delegation (۶ دقیقه)

سه مؤلفهٔ واگذاری: 1. **Problem Awareness (آگاهی از مسئله)** — واقعاً چه مشکلی را حل می‌کنم؟ 2. **Platform Awareness (آگاهی از پلتفرم)** — کدام ابزار برای این کار مناسب است؟ 3. **Task Delegation (واگذاری وظیفه)** — چه بخشی را به AI بسپارم؟

- توسعهٔ آگاهی از ملاحظات **وظیفه، پلتفرم و حالت (mode)**.
- [دانلود] دانلود: `Delegation Summary.pdf`

۷. برنامه‌ریزی پروژه و Delegation (۲۰ دقیقه) — تمرین عملی

گام ۱: پروژه‌تان را انتخاب کنید. یک پروژهٔ **متوسط و چندمرحله‌ای** که تا آخر دوره رویش کار کنید: - به‌قدر کافی **بزرگ** که چند نوع وظیفه را در بر بگیرد - به‌قدر کافی **قابل مدیریت** که در بازهٔ دوره تمام شود

۸. نگاه نزدیک‌تر به Description (۴ دقیقه)

سه نوع توصیف: 1. **Product (محصول)** — چه خروجی‌ای می‌خواهم؟ 2. **Process (فرایند)** — چطور به آن برسد؟ 3. **Performance (عملکرد)** — با چه لحن/نقش/کیفیتی؟

> Description فراتر از نوشتن یک درخواست ساده است — **هنر ارتباط مؤثر** با سیستم‌های AI.

- [دانلود] دانلود: `Description Summary.pdf`

۹. تکنیک‌های مؤثر پرامپت‌نویسی

- **مهندسی پرامپت (prompt engineering)** چیست و چرا برای همکاری پربازده مهم است.
- **شش تکنیک بنیادی پرامپت‌نویسی** برای بهبود تعامل.
- شناسایی **الگوهای رایج موفقیت**.
- **عیب‌یابی و اصلاح پرامپت** وقتی پاسخ‌ها مطلوب نیستند.
- [دانلود] دانلود: `6 Effective Prompting Techniques.pdf`

۱۰. نگاه نزدیک‌تر به Discernment (۵ دقیقه)

- ارزیابی اندیشمندانهٔ **خروجی‌ها، فرایندها و رفتارهای** AI.
- توسعهٔ **تفکر نقادانه** در تعاملات.
- شناسایی و رفع نگرانی‌های کیفی.
- **Discernment روی دیگر سکهٔ Description است.**

۱۱. حلقهٔ Description–Discernment (۶۰-۳۰ دقیقه) «مهم» تمرین عملی

- مهارت‌های توصیف و تشخیص را روی پروژهٔ واقعی‌تان به کار ببرید.
- درگیر شدن در **حلقه‌های بازخورد پربازدهِ توصیف–تشخیص**.

> هدف: خلق نتایجی از همکاری انسان-AI که **از آنچه هرکدام به‌تنهایی می‌توانستند بهتر باشد**.

۱۲. نگاه نزدیک‌تر به Diligence (۲۰ دقیقه)

- درک **پیامدهای اخلاقی** همکاری با AI.
- اهمیت **شفافیت (transparency)** در کار با AI.
- شناخت **مسئولیت خودتان** در قبال تعاملات و خروجی‌ها.

> در حالی که سه شایستگی دیگر عمدتاً به **اثربخشی** می‌پردازند، **Diligence به مسئولیت‌پذیری** می‌پردازد.

۱۳. جمع‌بندی (۶ دقیقه)

مرور چهار شایستگی و اتصال AI Fluency به مسیر توسعهٔ مستمر شما.

۱۴. گواهی پایان دوره

۱۵. فعالیت‌های تکمیلی (خودآموز)

ساخت برنامهٔ شخصی AI Fluency: - برای هر یک از چهار شایستگی ۴D، **سطح فعلی مهارت خود را ارزیابی کنید.** - با کمک یک دستیار AI، یک برنامهٔ ساختاریافته برای توسعهٔ تدریجی بسازید.

نکات کلیدی برای مرور سریع

- **۴D = Delegation، Description، Discernment، Diligence.**
- سه شیوهٔ همکاری: **Automation، Augmentation، Agency**.
- Delegation سه مؤلفه دارد: آگاهی از مسئله، آگاهی از پلتفرم، واگذاری وظیفه.
- Description سه نوع دارد: **محصول، فرایند، عملکرد**.
- **Discernment روی دیگر سکهٔ Description است** — این دو یک **حلقهٔ بازخورد** می‌سازند.
- **Diligence** = اخلاق، شفافیت، مسئولیت‌پذیری.

دوره ۹ — AI Capabilities and Limitations (قابلیت‌ها و محدودیت‌های هوش مصنوعی)

موضوع: دورهٔ مقدماتی دربارهٔ **اینکه AI چگونه کار می‌کند** — چهار ویژگی بنیادی ماشین و اینکه هر کدام کجا به قابلیت و کجا به محدودیت تبدیل می‌شود. **تعداد درس:** ۱۴

خلاصهٔ کلی دوره

> **چارچوب ۴D به شما یاد می‌دهد چطور با AI همکاری کنید. این دوره به شما یاد می‌دهد ماشین چطور کار می‌کند.**
> این دو، **دو نیمهٔ یک سیستم** هستند.

مطالب این دوره **بادوام (durable)** است — حتی وقتی مدل‌ها و محصولات مدام تغییر می‌کنند، این ویژگی‌های پایه ثابت می‌مانند.

«مهم» چهار ویژگی اصلی

1. **پیش‌بینی توکن بعدی (Next Token Prediction)**
2. **دانش (Knowledge)**
3. **حافظهٔ کاری (Working Memory)**
4. **هدایت‌پذیری (Steerability)**

> **نکتهٔ کلیدی: ویژگی‌های AI مولد روی یک پیوستار (continuum) از قابلیت تا محدودیت قرار دارند** — یعنی همان چیزی که قدرتش است، منشأ ضعفش هم هست.

نکات مهم هر درس

۱. مقدمه

- چرا این مطالب بادوام است.
- چطور چارچوب «قابلیت‌ها و محدودیت‌ها» با چارچوب ۴D کار می‌کند.
- [دانلود] دانلود: `AI Capabilities & Limitations Framework Overview.pdf`

۲. منظور ما از AI چیست؟ (۲۰ دقیقه)

- **بیشتر AI موجود در جهان** (فیلترهای اسپم، سیستم‌های توصیه‌گر و...) **طبقه‌بندی و پیش‌بینی** است، نه تولید.
- **AI مولد** با آن‌ها فرق دارد.

۳. AI شخصیتش را چطور به دست می‌آورد؟ (۲۵ دقیقه) «مهم»

> **ادب، کمک‌رسانی و احتیاط یک AI، ویژگی‌های ذاتی نیستند — نتیجهٔ آموزش‌اند.**

فرایند دومرحله‌ای آموزش: 1. **پیش‌آموزش (Pretraining)** 2. **تنظیم دقیق (Fine-tuning)**

«اثر انگشت رفتاری» که هر مرحله باقی می‌گذارد: - **Sycophancy (چاپلوسی)** — تمایل به تأیید کاربر - **Verbosity (پرگویی)** — پاسخ‌های بلندتر از لازم - **Over-caution (احتیاط بیش از حد)** - **Loose confidence calibration (کالیبراسیون شل اطمینان)** — با اطمینان اشتباه گفتن

از این درک استفاده کنید تا رفتارهایی را که در تعاملات خودتان می‌بینید **تفسیر کنید**.

۴. پیش‌بینی توکن بعدی (۳۰ دقیقه) «مهم»

• **مکانیزم هستهٔ AI مولد** — و دلیل اینکه **هم روانی (fluency) و هم توهم (hallucination)** تولید می‌کند.

پیوستار Next Token Prediction:

	ناحیه
مسیر کوبیده‌شده (well-worn path)	موضوعات پرتکرار → عملکرد قوی
قلمرو نوظهور (novel territory)	موضوعات کمیاب → خطر جعل

> **«هشدار» ناحیهٔ تمرکز جعل (fabrication): جزئیات مشخص — نام‌ها، تاریخ‌ها، استنادها، آمار.**

قابلیت‌های محصول که این را جبران می‌کنند: استنادها (citations)، سیگنال‌دهی عدم‌قطعیت، تولید محدودشده (constrained generation)، الگوی generator-verifier.

۵. تمرین تعاملی: Text Your Friend Markov

• یک **تولیدکنندهٔ توکن بعدی ۱۰۰٪ قابل‌تفسیر**.

• شبیه بازی‌ای که با کلمات پیشنهادی کیبورد گوشی پیام می‌ساختید.

• نشان می‌دهد کلمات بر اساس **الگو** پیشنهاد می‌شوند، نه فهم.

۶. دانش (Knowledge) (۳۰ دقیقه)

• دانش مدل **در حین آموزش شکل می‌گیرد** و **تاریخ قطع (cutoff) ثابتی** دارد.

پیش‌بینی کنید کدام موضوعات کجا قرار می‌گیرند:

ناحیهٔ قابلیت	ناحیهٔ لبه (خطر)
پرتکرار	کمیاب
اخیر در دادهٔ آموزش	بعد از cutoff
سازگار (consistent)	تخصصی (niche) / مورد مناقشه (contested)

شکست‌های مشخصهٔ دانش: - **کهنگی (staleness)** - **پوشش ناهموار (uneven coverage)** - **سوگیری به‌ارث‌رسیده (inherited bias)** - **فراموشی منبع (source amnesia)** — می‌داند اما نمی‌داند از کجا

قابلیت‌های محصول که جبران می‌کنند: جست‌وجوی وب، بازیابی/**RAG**، استفاده از ابزار.

۷. تمرین تعاملی: تجسم فضای ۱۰۲۴ بُعدی

مشکل رشته‌ها (strings): «car» را جست‌وجو کنید و هر سندی که کلمهٔ «car» دارد پیدا می‌شود — اما «automobile»، «vehicle» یا «ترمز سیویک من خراب است» پیدا نمی‌شود. - دهه‌ها جست‌وجو بر پایهٔ **شباهت رشته‌ای** بود، نه **معنا**. - راه‌حل‌های قدیمی: دیکشنری مترادف‌ها، قواعد ریشه‌یابی (stemming). - راه‌حل امروز: **نزدیکی در فضای چندبعدی (embeddings)**.

۸. حافظهٔ کاری (Working Memory) (۳۰ دقیقه)

• **پنجرهٔ زمینه = ظرفی با اندازهٔ ثابت.** پیامدش برای اسناد طولانی، گفت‌وگوهای طولانی و حافظهٔ بین‌جلسه‌ای.

> **«هشدار» این ویژگی ماهیت «پرتگاه» (cliff) دارد**، برخلاف بقیه که **افت تدریجی** دارند. یعنی تا یک نقطه خوب کار می‌کند و بعد ناگهان می‌افتد.

راهبردهای «زمینه به‌عنوان اهرم»: - **جلو انداختن (front-loading)** مطالب مهم - **تکه‌تکه کردن (chunking)** کارهای طولانی - **تأمین مجدد (re-supplying)** زمینهٔ حیاتی

قابلیت‌های محصول: حافظه (memory)، فشرده‌سازی (compaction)، پروژه‌ها/فضاهای کاری، پنجره‌های زمینهٔ بزرگ‌تر.

۹. تمرین تعاملی: افت زمینه — وقتی «بیشتر» یعنی «کمتر»

> غریزهٔ طبیعی این است که همه‌چیز را بدهید: کل سند را پیست کنید، همهٔ پیام‌ها را بگذارید. **اطلاعات بیشتر یعنی پاسخ بهتر، درست؟ همیشه نه.**

• پدیده‌ای که هرکس برای امتحان شب‌امتحان خوانده باشد شهودی می‌فهمد: **حدی برای آنچه می‌توان یک‌جا در ذهن نگه داشت وجود دارد — و چیزهای وسط اول ناپدید می‌شوند.**

۱۰. هدایت‌پذیری (Steerability) (۳۰ دقیقه) «مهم»

چرا کار می‌کند؟ fine-tuning به مدل **پیروی از دستور** را آموخته است. **چرا محدودیت دارد؟** دستورها از طریق **تطبیق الگو (pattern-matching)** دنبال می‌شوند، **نه فهم**.

کنترل محکم‌تر	کنترل شل‌تر
دستورهای کوتاه، عینی، قابل راستی‌آزمایی	زنجیره‌های استدلال طولانی
	درخواست‌های انتزاعی
	وظایف نیازمند دقت ذاتی (native precision)

شکست‌های مشخصه: - انحراف استدلال (reasoning drift) - حرف در برابر روح (letter-over-spirit) - حساب‌وکتاب شکننده (brittle arithmetic)

۱۱. تمرین تعاملی: حرف در برابر روح «مهم»

دستورها از طریق تطبیق الگو رعایت می‌شوند، نه فهم. همیشه فاصله‌ای بین کلمات شما و منظور شما هست. راه بستن این فاصله: هدف را صریح بیان کنید.

گفتید	منظورتان بود
«کوتاه‌ترش کن»	درخواست اصلی را برای کسی که سریع می‌خواند برجسته کن
«حرفه‌ای‌ترش کن»	برای مخاطب و کانال متفاوتی بازقاب‌بندی کن
«جزئیات بیشتری اضافه کن»	فقط آنچه به تصمیم‌گیری کمک می‌کند را بسط بده

۱۲. وقتی ویژگی‌ها با هم برخورد می‌کنند (When Properties Collide)

تعامل بین چهار ویژگی و اینکه چطور مشکلات ترکیبی می‌سازند.

۱۳. قدم‌های بعدی (۲۰ دقیقه)

- ترکیب چهار ویژگی و اثر انگشت‌های آموزش در یک **مدل ذهنی کاربردی**.
- اتصال این چارچوب به چارچوب ۴D به‌عنوان **دو نیمهٔ یک سیستم**.
- **یک تغییر عینی** در روش کارتان که همین هفته اعمال کنید.

استفادهٔ روان از AI یعنی حفظ کردن همهٔ حالت‌های شکست نیست. یعنی داشتن یک مدل کوچک و شفاف از ماشین.

۱۴. آزمون دوره

نکات کلیدی برای مرور سریع

- چهار ویژگی: **پیش‌بینی توکن بعدی، دانش، حافظهٔ کاری، هدایت‌پذیری**.
- هر ویژگی روی **پیوستار قابلیت↔محدودیت** است.

• جعل در **نام‌ها، تاریخ‌ها، استنادها و آمار** متمرکز می‌شود.

• شخصیت AI از **pretraining + fine-tuning** می‌آید → چاپلوسی، پرگویی، احتیاط زیاد، اطمینان بد کالیبره‌شده.

• **حافظهٔ کاری «پرتگاهی» است، نه تدریجی** — وسط متن اول گم می‌شود.

• **بیشتر همیشه بهتر نیست** — زمینهٔ اضافه می‌تواند کیفیت را پایین بیاورد.

• دستورها با **الگو** دنبال می‌شوند نه فهم → **هدف را صریح بگویید، نه فقط دستور را**.

دوره ۱۰ — Introduction to Model Context Protocol (آشنایی با MCP)

موضوع: ساخت سرور و کلاینت MCP از صفر با **پایتون**. تسلط بر سه پریمیتیو اصلی MCP — **ابزارها، منابع و پرامپت‌ها** — برای اتصال کلود به سرویس‌های بیرونی. **تعداد درس:** ۱۴ (ویدیویی + پروژهٔ عملی)

خلاصهٔ کلی دوره

MCP یک لایهٔ ارتباطی است که به کلود **زمینه و ابزار** می‌دهد، بدون اینکه مجبور باشید انبوهی از کد یکپارچه‌سازی خسته‌کننده بنویسید.

> آن را راهی بدانید برای **انتقال بار تعریف و اجرای ابزارها از سرور شما به سرورهای تخصصی MCP**.

پروژهٔ عملی دوره: یک **سرور مدیریت اسناد** با ابزارهای خواندن و به‌روزرسانی سند.

«مهم» سه پریمیتیو اصلی سرور — و اینکه هرکدام را چه کسی کنترل می‌کند

این مهم‌ترین نکتهٔ کل دوره است. **هر پریمیتیو توسط بخش متفاوتی از استک برنامهٔ شما کنترل می‌شود:**

پریمیتیو	کنترل‌کننده	توضیح
Tools (ابزارها)	**مدل (Model-Controlled)**	کلود تصمیم می‌گیرد کِی صدایشان بزند؛ نتایج مستقیماً استفاده می‌شوند
Resources (منابع)	**برنامه (Application-Controlled)**	کد شما تصمیم می‌گیرد چه داده‌ای در پرامپت گنجانده شود
Prompts (پرامپت‌ها)	**کاربر (User-Controlled)**	کاربر انتخاب می‌کند از کدام قالب استفاده کند

نکات مهم هر درس

۱. خوش‌آمدگویی

- لینک مستقیم راهنمای نصب **UV** (مدیر پکیج پایتون).

۲. معرفی MCP

- معماری پایه: یک **MCP Client** (برنامهٔ شما) که با یک **MCP Server** صحبت می‌کند.
- هدف: کاهش کد یکپارچه‌سازی تکراری.

۳. کلاینت‌های MCP

- **کلاینت MCP = پل ارتباطی** بین سرور شما و سرورهای MCP.
- نقطهٔ دسترسی شما به همهٔ ابزارهایی که یک سرور MCP فراهم می‌کند.
- **تبادل پیام و جزئیات پروتکل** را مدیریت می‌کند تا برنامهٔ شما مجبور نباشد.

«هشدار» ویژگی کلیدی: ارتباط مستقل از حامل (Transport Agnostic)

> یکی از نقاط قوت اصلی MCP این است که **کلاینت و سرور می‌توانند از راه‌های مختلف با هم ارتباط بگیرند** — بدون اینکه کد شما تغییر کند.

۴. راه‌اندازی پروژه

- [دانلود] دانلودها: `cli_project.zip` (شروع) و `cli_project_COMPLETE.zip` (نسخهٔ کامل)

۵. تعریف ابزار با MCP

- ساخت سرور MCP با **SDK رسمی پایتون** بسیار ساده‌تر است.
- به‌جای نوشتن **schemaهای پیچیدهٔ JSON با دست**، ابزارها را با **دکوراتور (decorator)** تعریف می‌کنید و SDK کار سنگین را انجام می‌دهد.
- **مثال دوره:** سرور مدیریت اسناد با دو ابزار اصلی — یکی برای **خواندن** سند، یکی برای **به‌روزرسانی**. همهٔ اسناد در حافظه (in memory) به‌شکل یک ساختار ساده نگه‌داری می‌شوند.

۶. بازرس سرور (Server Inspector)

- هنگام ساخت سرور MCP، به راهی نیاز دارید که **بدون اتصال به یک برنامهٔ کامل** عملکرد را تست کنید.
- SDK پایتون MCP یک **بازرس مبتنی بر مرورگر** داخلی دارد برای دیباگ و تست **بلادرنگ**.
- **راه‌اندازی:** ابتدا محیط پایتون را فعال کنید (دستور دقیق در README پروژه)، سپس بازرس را اجرا کنید.

۷. نظرسنجی رضایت از دوره

۸. پیاده‌سازی یک کلاینت

- کلاینت همان چیزی است که به کد برنامهٔ شما اجازه می‌دهد با سرور MCP ارتباط بگیرد.
- **نکتهٔ واقع‌گرایانه:** در بیشتر پروژه‌های دنیای واقعی، شما **یا** کلاینت MCP را پیاده می‌کنید **یا** سرور را — **نه هر دو**. در این پروژه هر دو را می‌سازیم فقط برای اینکه ببینید چطور کنار هم کار می‌کنند.

۹. تعریف منابع (Resources)

- **منابع** به شما اجازه می‌دهند **داده را در معرض کلاینت‌ها بگذارید** — شبیه **هندلرهای درخواست GET** در یک سرور HTTP معمولی.
- برای وقتی عالی‌اند که نیاز دارید **اطلاعات بگیرید** نه اینکه **اقدامی انجام دهید**.
- **مثال:** قابلیت «منشن سند» که کاربر تایپ می‌کند `@document_name` تا به فایلی ارجاع دهد.

۱۰. دسترسی به منابع

- منابع اجازه می‌دهند سرور اطلاعاتی را در معرض بگذارد که **مستقیماً در پرامپت گنجانده می‌شود**، به‌جای اینکه برای دسترسی به داده نیاز به **فراخوانی ابزار** باشد.
- این راه **کارآمدتری** برای تأمین زمینه به مدل‌های AI است.
- **جریان کار:** کاربر تایپ می‌کند `What's in the @...` → کد شما آن را به‌عنوان درخواست منبع تشخیص می‌دهد → یک `ReadResourceRequest` می‌فرستد.

۱۱. تعریف پرامپت‌ها (Prompts)

- **پرامپت‌ها** اجازه می‌دهند **دستورالعمل‌های از پیش ساخته‌شده و باکیفیت** تعریف کنید که کلاینت‌ها به‌جای نوشتن پرامپت از صفر استفاده کنند.
- آن‌ها را **قالب‌های با دقت ساخته‌شده** بدانید که نتایج بهتری از آنچه کاربران خودشان می‌نویسند می‌دهند.

چرا پرامپت؟ بینش کلیدی: کاربران **می‌توانند** بیشتر کارها را مستقیماً از کلود بخواهند — اما نتیجه‌اش به‌خوبی یک قالب مهندسی‌شده نیست.

۱۲. پرامپت‌ها در کلاینت

- آخرین گام ساخت کلاینت: پیاده‌سازی قابلیت پرامپت — **فهرست‌کردن همهٔ پرامپت‌های موجود** از سرور و **بازیابی پرامپت مشخص با متغیرهای پرشده**.
- متد `list_prompts` ساده است: تابع list prompts سشن را صدا می‌زند و پرامپت‌ها را برمی‌گرداند.

```
async def list_prompts(self) -> list[types.Prompt]:
    ...
```

۱۳. ارزیابی نهایی MCP

۱۴. مرور MCP

مرور سه پریمیتیو و اینکه کِی از هرکدام استفاده کنیم (جدول بالا).

نکات کلیدی برای مرور سریع

- **MCP = لایهٔ ارتباطی** که بار یکپارچه‌سازی را از سرور شما بردارد.
- **Tools → مدل کنترل می‌کند • Resources → برنامه کنترل می‌کند • Prompts → کاربر کنترل می‌کند.**
- MCP **مستقل از حامل (transport agnostic)** است.
- با **SDK پایتون** و **دکوراتورها** ابزار تعریف کنید — نه JSON Schema دستی.
- **بازرس مرورگری** برای تست سرور بدون برنامهٔ کامل.
- منابع = خواندن داده (مثل GET)؛ ابزارها = انجام اقدام.
- معمولاً یا کلاینت می‌سازید یا سرور، نه هر دو.

دوره ۱۱ — Model Context Protocol: Advanced Topics (MCP): مباحث پیشرفته

موضوع: الگوهای پیشرفتهٔ پیاده‌سازی MCP — **Sampling، اعلان‌ها (Notifications)، دسترسی به فایل‌سیستم (Roots)** و **مکانیزم‌های حامل (Transports)** برای توسعهٔ سرور MCP در سطح پروداکشن. **تعداد درس:** ۱۵ (ویدیو + پروژه‌های عملی قابل دانلود)

خلاصهٔ کلی دوره

این دوره ادامهٔ «آشنایی با MCP» است و روی قابلیت‌هایی تمرکز دارد که سرور را از یک نمونهٔ آزمایشی به یک سرویس **قابل استقرار در پروداکشن** تبدیل می‌کند.

نکات مهم هر درس

۱. شروع کنیم!

۲. Sampling (نمونه‌گیری) «مهم»

تعریف: Sampling به یک **سرور** اجازه می‌دهد از طریق یک **کلاینت MCP متصل** به یک مدل زبانی مثل کلود دسترسی پیدا کند.

> **به‌جای اینکه سرور مستقیماً کلود را صدا بزند، از کلاینت می‌خواهد که این کار را از طرف او انجام دهد.**

مشکلی که حل می‌کند: فرض کنید سروری با ابزار تحقیق دارید که اطلاعات را از ویکی‌پدیا می‌گیرد و باید خلاصه شود. اگر سرور خودش کلود را صدا بزند، **هزینه و مسئولیت تولید متن روی دوش سرور** است.

> **با Sampling، مسئولیت و هزینهٔ تولید متن از سرور به کلاینت منتقل می‌شود.**

۳. راهنمای عملی Sampling

- [دانلود] دانلود: `sampling.zip`

۴. اعلان‌های لاگ و پیشرفت (Log & Progress Notifications)

- **پیاده‌سازی ساده، اما تأثیر بزرگ روی تجربهٔ کاربری.**
- **مشکل:** وقتی کلود ابزاری را صدا می‌زند که زمان می‌برد — مثل تحقیق دربارهٔ یک موضوع یا پردازش داده — کاربر معمولاً **هیچ چیزی نمی‌بیند** و نمی‌داند آیا چیزی خراب شده یا نه.
- **راه‌حل:** اعلان‌ها به کاربر نشان می‌دهند **در طول عملیات طولانی چه اتفاقی می‌افتد**.

۵. راهنمای عملی اعلان‌ها

• [دانلود] دانلود: `notifications.zip`

۶. Roots (ریشه‌ها) «مهم»

تعریف: Roots راهی است برای **اعطای دسترسی سرورهای MCP به فایل‌ها و پوشه‌های مشخص** روی دستگاه محلی شما.

> آن را یک **سیستم مجوز** بدانید که می‌گوید: «هی سرور MCP، تو می‌توانی به این فایل‌ها دسترسی داشته باشی» — اما **کاری بیشتر از صرفاً اعطای مجوز** انجام می‌دهد.

مشکلی که حل می‌کند: بدون roots به مشکل رایجی برمی‌خورید. فرض کنید سروری با ابزار تبدیل ویدیو دارید که یک **مسیر فایل** می‌گیرد — سرور از کجا بداند کدام مسیرها معتبر و مجازند؟

۷. راهنمای عملی Roots

• [دانلود] دانلود: `roots.zip`

۸. نظرسنجی

۹. انواع پیام JSON

• **همهٔ ارتباطات MCP از طریق پیام‌های JSON انجام می‌شود.**

• هر نوع پیام هدف مشخصی دارد.

• درک این انواع **حیاتی** است، به‌ویژه هنگام کار با حامل‌های مختلف مثل **streamable HTTP**.

۱۰. حامل STDIO (The STDIO Transport)

• کلاینت‌ها و سرورهای MCP با تبادل پیام JSON ارتباط می‌گیرند — اما **این پیام‌ها واقعاً چطور منتقل می‌شوند؟**

• کانال ارتباطی را **حامل (transport)** می‌نامند. روش‌های مختلفی هست: از درخواست HTTP تا WebSocket.

• **STDIO** حاملی است که وقتی تازه شروع می‌کنید با آن کار می‌کنید — ساده و محلی.

۱۱. حامل StreamableHTTP

• **StreamableHTTP** به کلاینت‌های MCP اجازه می‌دهد به **سرورهای میزبانی‌شده از راه دور** روی اتصالات HTTP وصل شوند.

• برخلاف STDIO که **هر دو طرف باید روی یک دستگاه باشند**، این حامل امکان **سرورهای عمومی MCP** را باز می‌کند که هرکسی می‌تواند به آن‌ها دسترسی داشته باشد.

• **«هشدار» نکتهٔ مهم:** برخی تنظیمات پیکربندی می‌توانند قابلیت‌ها را به‌شدت محدود کنند.

۱۲. StreamableHTTP به‌صورت عمیق «مهم»

مشکل بنیادی: برخی قابلیت‌های MCP نیاز دارند **سرور به کلاینت درخواست بفرستد** — اما **HTTP این کار را دشوار می‌کند** (HTTP ذاتاً یک‌طرفه است: کلاینت می‌پرسد، سرور جواب می‌دهد).

قابلیت‌هایی که به ارتباط سرور→کلاینت وابسته‌اند: Sampling، اعلان‌ها (notifications)، و لاگ‌گیری.

راه‌حل: دور زدن مبتنی بر SSE (Server-Sent Events).

نکات کلیدی این درس: - StreamableHTTP **پیچیده‌تر از سایر حامل‌های MCP** است، چون باید محدودیت‌های HTTP را دور بزند. - راه‌حل مبتنی بر SSE، **قابلیت کامل MCP روی HTTP** را ممکن می‌کند. - **درک مدل دو-اتصالی (dual-connection model) برای دیباگ و بهینه‌سازی حیاتی است.** - `session ID` **برای همهٔ درخواست‌های بعد از مقداردهی اولیه (initialization) الزامی است.** - سیستم به‌صورت خودکار **چند اتصال SSE** را مدیریت می‌کند تا انواع مختلف ارتباط سرور→کلاینت را پوشش دهد.

۱۳. وضعیت (State) و حامل StreamableHTTP

دو پرچم (flag) که جنبه‌های بنیادی رفتار سرور را کنترل می‌کنند:

پرچم	کاربرد
`stateless_http`	سرور را بدون‌حالت می‌کند
`json_response`	نحوهٔ برگرداندن پاسخ را کنترل می‌کند

کِی به Stateless HTTP نیاز دارید؟

> فرض کنید سرور MCPای می‌سازید که محبوب می‌شود. اولش شاید فقط چند کاربر داشته باشید... اما وقتی بخواهید **مقیاس بدهید** یا در **پروداکشن مستقر** کنید، حالت‌داری (statefulness) مشکل‌ساز می‌شود — چون درخواست‌ها باید همیشه به همان نمونهٔ سرور برسند.

- [دانلود] دانلود پروژهٔ همراه موجود است.

۱۴. ارزیابی مفاهیم MCP

۱۵. جمع‌بندی

نکات کلیدی برای مرور سریع

- **Sampling** = سرور از کلاینت می‌خواهد مدل را صدا بزند → **هزینه و مسئولیت به کلاینت منتقل می‌شود.**
- **Notifications** = لاگ و پیشرفت؛ ساده اما تجربهٔ کاربری را نجات می‌دهد.
- **Roots** = مجوز دسترسی به فایل/پوشهٔ مشخص روی دستگاه محلی.
- **همهٔ ارتباط MCP از طریق JSON است.**
- **STDIO** → محلی، هر دو طرف روی یک دستگاه.

• **StreamableHTTP** → از راه دور، سرورهای عمومی.

• StreamableHTTP از **SSE** استفاده می‌کند تا محدودیت یک‌طرفه بودن HTTP را دور بزند؛ **session ID الزامی است**.

• برای مقیاس‌پذیری در پروداکشن → `stateless_http` .

دوره ۱۲ — AI Fluency for Educators (سواد هوش مصنوعی برای معلمان)

موضوع: توانمندسازی اعضای هیئت علمی، طراحان آموزشی و رهبران آموزشی برای به‌کارگیری AI Fluency در **عمل تدریس** و **راهبرد نهادی**. **تعداد درس:** ۵

خلاصهٔ کلی دوره

چارچوب ۴D را از حالت عمومی به **کاربرد مشخص در تدریس** می‌برد: طراحی دوره، تدوین اهداف یادگیری، و ساخت مواد آموزشی و تکالیف.

> **«مهم» تأکید محوری دوره: AI Fluency دربارهٔ Augmentation (افزودن به کار شما) است، نه Automation (اینکه AI کار را به‌جای شما انجام دهد)** — و این تمایز **به‌ویژه در بافت‌های یادگیری** حیاتی است.

نکات مهم هر درس

۱. مقدمه‌ای بر AI Fluency برای معلمان (۳۰ دقیقه)

اهداف یادگیری: - توضیح ارتباط چارچوب ۴D با **تدریس** - **ساخت یک «سند زمینهٔ تدریس» قابل استفادهٔ مجدد (teaching context document)** که درک مشترکی با AI برقرار کند - **بیان صریح ارزش‌های آموزشی و محدودیت‌های خود (pedagogical values and constraints)** برای همکاری مؤثرتر

> **نکتهٔ کاربردی کلیدی:** به‌جای اینکه هر بار از صفر توضیح دهید چه‌کاره‌اید و چه می‌خواهید، **یک سند زمینه بسازید** و آن را در هر گفت‌وگو استفاده کنید.

- [دانلود] دانلود: رونوشت ویدیو (`AIF4E_TRANSCRIPT_01_WELCOME.txt`)

۲. مرور چارچوب AI Fluency (۱۵ دقیقه، ویدیو ۱۳ دقیقه‌ای)

- تعریف AI Fluency و هر یک از **۴D**: Delegation، Description، Discernment، Diligence.
- **تأکید صریح: تقویت (augmentation) در برابر خودکارسازی (automation).**

چرا در آموزش این تمایز مهم‌تر است؟ چون هدف آموزش، **تولید خروجی** نیست — **رشد یادگیرنده** است. اگر AI کار را انجام دهد، یادگیری اتفاق نیفتاده.

- [دانلود] دانلودها: `AI Fluency Summary.pdf` • `Key Terminology Cheat Sheet.pdf` • `Framework_for_AI_Fluency_V_1.5.pdf`

۳. کاربرد AI Fluency در طراحی دوره و اهداف یادگیری (۶۰ دقیقه) «مهم»

اهداف یادگیری: - به‌کارگیری **نظام‌مند هر چهار D** در طراحی دوره - حرکت از **برنامه‌ریزی سطح‌بالا** به **اهداف یادگیری مشخص** - ساخت **ساختارهای منسجم دوره** از طریق همکاری با AI - **حفظ یکپارچگی آموزشی (pedagogical integrity)** در عین بهره‌گیری از توانایی‌های AI

> **بینش کلیدی ویدیو: ساختن زمینهٔ غنی و مشترک، همکاری با AI را دگرگون می‌کند.**

۴. کاربرد AI Fluency در مواد آموزشی و تکالیف (۶۰ دقیقه)

اهداف یادگیری: - ساخت مواد آموزشی با دستیار AI با به‌کارگیری چارچوب ۴D - **بهره‌گیری از زمینهٔ از پیش تثبیت‌شده** برای توسعهٔ منسجم مواد - اعمال **کنترل کیفیت نظام‌مند از طریق Discernment** - ساخت موادی که **به‌عنوان یک تجربهٔ یادگیری یکپارچه** با هم کار کنند

دامنهٔ کار: از اسلایدهای درس تا ابزارهای ارزیابی — یعنی موادی که **دانشجویان واقعاً با آن‌ها کار می‌کنند**.

۵. گواهی پایان دوره

نکات کلیدی برای مرور سریع

- **Augmentation نه Automation** — به‌ویژه در یادگیری.
- اولین کار عملی: **سند زمینهٔ تدریس** بسازید (ارزش‌ها، محدودیت‌ها، سبک، مخاطب).
- زمینهٔ غنی و مشترک → بزرگ‌ترین اهرم بهبود کیفیت خروجی.
- **Discernment = کنترل کیفیت نظام‌مند**، نه بازبینی تصادفی.
- مواد آموزشی باید **به‌عنوان یک مجموعهٔ یکپارچه** طراحی شوند، نه تکه‌تکه.
- **یکپارچگی آموزشی را قربانی سرعت نکنید.**

دوره ۱۳ — AI Fluency for Students (سواد هوش مصنوعی برای دانشجویان)

موضوع: توسعهٔ مهارت‌های AI Fluency که **یادگیری، برنامه‌ریزی شغلی و موفقیت تحصیلی** را از طریق همکاری مسئولانه با AI تقویت می‌کند. **تعداد درس:** ۶

خلاصهٔ کلی دوره

> این دوره روی **نکات موقتی یا ابزارهای خاص** تمرکز نمی‌کند؛ **اصول ماندگاری** را بررسی می‌کند که به شما کمک می‌کند **تصمیم‌های اندیشیده دربارهٔ اینکه کِی و چگونه از AI استفاده کنید** بگیرید.

تعریف AI Fluency در این دوره: **استفاده از AI به‌شکل مؤثر، کارآمد، اخلاقی و ایمن.**

نکات مهم هر درس

۱. خوش‌آمدید (۵ دقیقه)

- چرا AI Fluency یک **مهارت ضروری برای دانشجویان** است.
- تمرکز بر **اصول بادوام** نه ترفندهای زودگذر.

۲. چارچوب AI Fluency (۳۰ دقیقه)

اهداف یادگیری: - تعریف AI Fluency و اینکه چرا برای **آیندهٔ تحصیلی و حرفه‌ای** شما اهمیت دارد - به‌کارگیری چارچوب ۴D در تعاملات‌تان - **تشخیص تفاوت استفاده از AI برای automation در برابر augmentation** - **ساخت یک «سند زمینهٔ یادگیری شخصی»** برای هدایت همکاری‌های آینده

- [دانلود] دانلودها: `AI Fluency Summary.pdf` • `Key Terminology Cheat Sheet.pdf` • `Framework_for_AI_Fluency_V_1.5.pdf`

۳. AI به‌عنوان شریک یادگیری (۵۵ دقیقه) «مهم» — مهم‌ترین درس دوره

تمایز حیاتی:

> **استفاده از AI برای «انجام دادن کار» در برابر استفاده از AI برای «یاد گرفتن».**

اهداف یادگیری: - تمایز بین این دو حالت - به‌کارگیری ۴D **به‌طور مشخص در بافت یادگیری** - ساخت یک **شریک یادگیری AI** که یادگیری را **تقویت کند نه جایگزینش شود** - ساخت یک **دفترچهٔ یادگیری زنده (living learning journal)** که رشد شما را در طول زمان ردیابی کند

هر D در بافت یادگیری: - **Delegation:** حفظ **آگاهی از مسئله (problem awareness)** — یعنی خودتان بفهمید مسئله چیست، قبل از اینکه بسپاریدش. - (و به همین ترتیب برای سه D دیگر در بافت یادگیری.)

> **«هشدار» خطر اصلی برای دانشجو:** اگر AI تکلیف را انجام دهد، نمرهٔ خوب می‌گیرید اما **مهارتی به دست نمی‌آورید**. هدف باید **یادگیری** باشد.

۴. AI در برنامه‌ریزی شغلی (۵۵ دقیقه)

اهداف یادگیری: - استفادهٔ اندیشیده از AI برای **کاوش شغلی، مهارت‌سازی، جست‌وجوی کار و پشتیبانی از فرایند درخواست شغل** - ساخت **مواد درخواست شغل اصیل (authentic)** با کمک AI و در عین حال **حفظ صدای منحصربه‌فرد خودتان** - **تمرین مهارت‌های مصاحبه** با AI به‌عنوان مربی (coach) - ساخت یک **رویکرد راهبردی به توسعهٔ شغلی**

> **اصل کلیدی: از AI به‌عنوان شریک فکری استفاده کنید، نه ماشینی که تصمیم می‌گیرد یا محتوای کلیشه‌ای (generic) تولید می‌کند.**

۵. «انسان در حلقه» بودن (۴۰ دقیقه) «مهم»

تعریف: «انسان در حلقه» (human in the loop) یعنی **تصمیم‌گیرنده‌ای که تعاملات AI را با قضاوت، خلاقیت و اخلاق هدایت می‌کند.**

اهداف یادگیری: - درک معنای «انسان در حلقه» بودن - **ساخت یک سیاست شخصی همکاری با AI** که بازتاب ارزش‌های شما باشد - تدوین **دستورالعمل‌هایی برای استفادهٔ مسئولانه** در زندگی تحصیلی و حرفه‌ای - ساخت چارچوبی برای **رشد مستمر** در AI Fluency

۶. گواهی پایان دوره

نکات کلیدی برای مرور سریع

- AI Fluency = **مؤثر + کارآمد + اخلاقی + ایمن.**
- **AI برای انجام کار ≠ AI برای یادگیری.** این تمایز، قلب دورهٔ دانشجویی است.
- اولین کار عملی: **سند زمینهٔ یادگیری شخصی** بسازید.
- **دفترچهٔ یادگیری زنده** بسازید تا رشدتان را ببینید.
- در برنامه‌ریزی شغلی: **صدای خودتان را حفظ کنید**؛ AI مربی است نه نویسنده.
- **انسان در حلقه = تصمیم‌گیرندهٔ نهایی با قضاوت و اخلاق.**
- یک **سیاست شخصی همکاری با AI** بنویسید و به آن پایبند باشید.

دوره ۱۴ — Teaching AI Fluency (تدریس سواد هوش مصنوعی)

موضوع: توانمندسازی اعضای هیئت علمی و طراحان آموزشی برای **تدریس و ارزیابی** AI Fluency در محیط‌های حضوری و کلاسی. **تعداد درس:** ۸ **پیش‌نیاز:** آشنایی قبلی با چارچوب AI Fluency (دورهٔ ۸).

خلاصهٔ کلی دوره

این دوره برای معلمانی است که **خودشان با چارچوب کار کرده‌اند و حالا آماده‌اند آن را به دیگران بیاموزند**. تمرکز روی **روش‌های تدریس، حلقه‌های آموزشی، ارزیابی و طراحی تکلیف** است.

«مهم» دو حلقهٔ اصلی چارچوب ۴D

این دوره مهم‌ترین بازتعریف چارچوب را ارائه می‌دهد: **۴D در واقع دو حلقهٔ جفت‌شده است.**

حلقه	جفت	تمرکز
حلقهٔ Delegation–Diligence	واگذاری ↔ وظیفه‌شناسی	سطح **راهبردی و اخلاقی** — آیا، کِی و چگونه از AI استفاده کنیم
حلقهٔ Description–Discernment	توصیف ↔ تشخیص	سطح **عملیاتی و لحظه‌به‌لحظه** — ساخت محیط شناختی مشترک با AI

نکات مهم هر درس

۱. خوش‌آمدگویی و رویکردهای تدریس AI Fluency (۶۰ دقیقه)

چهار رویکرد متمایز برای تدریس چارچوب: 1. **خطی (linear)** — ترتیبی، D به 2 D. **غیرخطی (non-linear)** — بر اساس نیاز 3. **متمرکز (focused)** — تعمیق روی یک 4 D. **حلقه‌محور (loop-based)** — تدریس بر پایهٔ دو حلقهٔ بالا

هدف: انتخاب و **تطبیق** رویکرد متناسب با دانشجویان خودتان.

- [دانلود] دانلودها: `AI Fluency Summary.pdf` • `Key Terminology Cheat Sheet.pdf`

۲. حلقهٔ Delegation–Diligence (۴۰ دقیقه) «مهم»

- چارچوبی برای **طراحی و تصمیم‌گیری مسئولانه** در همکاری با AI.
- به پرسش‌های **راهبردی و اخلاقی** می‌پردازد: **آیا، کِی و چگونه** از AI استفاده کنیم.

حرکت از «چطور از AI استفاده کنم؟» به «چطور دربارهٔ AI تصمیم‌های خوب بگیرم؟»

• ویدیو **رابطهٔ دوطرفه** بین این دو شایستگی را نشان می‌دهد: واگذاری بدون وظیفه‌شناسی بی‌مسئولیت است؛ وظیفه‌شناسی بدون واگذاری فلج‌کننده است.

۳. حلقهٔ Description–Discernment (۴۰ دقیقه) «مهم»

• روی **صنعت لحظه‌به‌لحظهٔ ساختن محیط‌های شناختی (cognitive environments)** تمرکز دارد که در آن انسان و AI مؤثر با هم کار کنند.

• **حرکت فراتر از تک‌پرامپت به گفت‌وگوهای پایدار (sustained conversations).**

• به دانشجویان می‌آموزد **زمینه و درک مشترک** با AI بسازند.

• تمرکز روی سه نوع توصیف: **Product، Process، Performance**.

۴. چگونه ۴D را ارزیابی کنیم؟ (۴۰ دقیقه) «مهم»

سه رویکرد مکمل ارزیابی:

رویکرد	تمرکز
مبتنی بر نتیجه (outcome-based)	آنچه دانشجویان از طریق همکاری با AI **تولید می‌کنند**
مبتنی بر فرایند (process-based)	**چگونگی** کار دانشجویان با AI در طول زمان
مبتنی بر تأمل (reflection-based)	**آگاهی فراشناختی (metacognitive)** دانشجو

هدف عملی: ساخت **روبریک (rubric)** هایی که چهار شایستگی را در بافت تدریس خاص شما بسنجند.

• [دانلود] **چهار ماتریس ارزیابی قابل دانلود:** `Description • Delegation assessment matrix.pdf` `Diligence assessment • Discernment assessment matrix.pdf` `• assessment matrix.pdf` `matrix.pdf`

۵. طراحی تکلیف برای AI Fluency (۴۰ دقیقه)

سه اصل کلیدی: 1. **اصالت (authenticity)** — تکالیفی که **همکاری واقعی با AI در دنیای واقعی** را بازتاب دهند 2. **تکرار (iteration)** — گنجاندن فرصت‌های اصلاح که **رشد را نمایان کند** 3. (اصل سوم در ویدیو ارائه می‌شود)

چالش عملی: مدیریت حجم و پیچیدگی افزایش‌یافتهٔ کار دانشجویی تقویت‌شده با AI — وقتی همه با AI کار می‌کنند، حجم خروجی به‌شدت بالا می‌رود.

• [دانلود] دانلود: `Assignment component guide.pdf`

۶. تأثیر AI و رشتهٔ شما (۴۰ دقیقه)

• AI روی رشته‌های مختلف **به شیوه‌های منحصربه‌فرد** اثر می‌گذارد — در **برنامهٔ درسی، روش تدریس و ارزیابی.**

تخصص رشته‌ای شما — فهم ارزش‌های محوری، روش‌ها و شیوه‌های اندیشیدن در حوزه‌تان — برای پیمایش این اختلال‌ها ضروری است.

• **خروجی عملی:** نگارش یک **مقالهٔ موضع (position paper)** که دیدگاه شما دربارهٔ نقش AI در رشته‌تان را بیان کند.

۷. به‌کارگیری تخصص رشته‌ای در AI Fluency (۴۰ دقیقه)

• چگونه **دانش عمیق رشته‌ای** خود را به کار ببرید تا چارچوب را **مخصوص حوزه‌تان** کنید.

AI Fluency ویژهٔ رشته از طریق «صریح‌کردن دانش ضمنی (making tacit knowledge explicit)» شکل می‌گیرد — یعنی بیان‌کردن آنچه متخصصان حوزه‌تان می‌دانند اما معمولاً نمی‌گویند.

• **همکاری با همکاران** برای ساخت درک مشترک از AI Fluency در گروه آموزشی.

۸. گواهی پایان دوره

نکات کلیدی برای مرور سریع

• **۴D = دو حلقه:** Delegation↔Diligence (راهبردی/اخلاقی) و Description↔Discernment (عملیاتی).

• سؤال محوری تغییر می‌کند: از «چطور استفاده کنم؟» به **«چطور تصمیم خوب بگیرم؟»**.

• چهار رویکرد تدریس: **خطی، غیرخطی، متمرکز، حلقه‌محور**.

• ارزیابی سه‌گانه: **نتیجه + فرایند + تأمل**.

• طراحی تکلیف: **اصالت + تکرار**.

• **دانش ضمنی رشته‌تان را صریح کنید** — این کلید بومی‌سازی چارچوب است.

• چهار **ماتریس ارزیابی** آماده برای دانلود موجود است.

دوره ۱۵ — AI Fluency for Nonprofits (سواد هوش مصنوعی برای سازمان‌های غیرانتفاعی)

موضوع: توسعهٔ AI Fluency برای افزایش **اثرگذاری و کارایی سازمانی** در عین وفاداری به **مأموریت و ارزش‌ها**. **تعداد درس:** ۱۰

خلاصهٔ کلی دوره

دوره صراحتاً **شکاف بین شور و اشتیاق نسبت به AI و پیاده‌سازی معنادار** در این بخش را به رسمیت می‌شناسد. تمرکز روی کاربردهای عملی است: **تحقیق، نگارش، حریم خصوصی، تحلیل داده، خودکارسازی جریان کار و سیاست سازمانی.**

نکات مهم هر درس

۱. خوش‌آمدید (۲۰ دقیقه)

- تعریف AI Fluency.
- **بیان صریح ارزش‌ها و محدودیت‌های خودتان** برای همکاری مؤثرتر.
- شناسایی **اهداف‌تان** برای یکپارچه‌سازی ابزارها و توانمندی AI در کار.

۲. چارچوب 4D (۳۰ دقیقه) «مهم»

دو حالت تعامل (دو حلقه):

حلقه	کاربرد
Delegation–Diligence	تصمیم‌گیری دربارهٔ **اینکه آیا و کِی** از AI استفاده کنیم
Description–Discernment	درگیر شدن **در حین** کار با AI

۳. تحقیق با AI (۵۰ دقیقه)

- **Description:** ساخت پرامپت‌های مؤثر که AI را به جمع‌آوری و ترکیب اطلاعات مرتبط با کار غیرانتفاعی شما هدایت کند.
- **Discernment:** ارزیابی تحقیق تولیدشده توسط AI از نظر **دقت، مرتبط بودن و تناسب** با بافت سازمان شما.
- **سناریوی عملی:** ماریا، مدیر اجرایی، در حال گسترش برنامهٔ مسکن سازمانش.

۴. نگارش با AI (۵۰ دقیقه)

- **Description:** پرامپت‌های شفاف و غنی از زمینه برای تولید یا بهبود مواد نوشتاری **هم‌راستا با مأموریت سازمان و نیاز مخاطب**.

• **Discernment:** ارزیابی متون تولیدشده از نظر **دقت، تناسب و هم‌راستایی با ارزش‌های سازمانی و استانداردهای ارتباطی**.

۵. درک حریم خصوصی و داده (۳۰ دقیقه) «مهم»

> **یکی از رایج‌ترین نگرانی‌های حرفه‌ای‌های بخش غیرانتفاعی: حریم خصوصی داده.**

اهداف یادگیری: - بیان نگرانی‌های حریم خصوصی و **ارزیابی ابزارهای AI بر اساس سیاست‌های مدیریت دادهٔ آن‌ها** - تمرین **راهبردهای بهداشت داده (data hygiene)** برای کار ایمن با اطلاعات حساس

محتوا: واقعاً چه اتفاقی برای داده‌ای که با ابزارهای AI به اشتراک می‌گذارید می‌افتد، و چطور پلتفرم‌ها و پلن‌های مختلف را از این نظر ارزیابی کنید.

۶. تحلیل داده با AI (۵۰ دقیقه) «مهم»

پرسش حیاتی: از کجا بدانم می‌توانم به تحلیل AI اعتماد کنم؟

اهداف یادگیری: - استفاده از **حلقهٔ Delegation–Diligence** برای **اعتبارسنجی نظام‌مند توانایی‌های تحلیلی AI** برای کار خاص شما - به‌کارگیری Description و Discernment برای شناسایی الگو در داده‌ها، **در عین شناخت محدودیت‌های AI**

> **«مهم» تکنیک کلیدی: با آزمودن AI روی داده‌ای که خودتان از قبل می‌فهمید، اعتماد بسازید.** اگر روی داده‌ای که جوابش را می‌دانید درست عمل کرد، می‌توانید کمی بیشتر اعتماد کنید.

۷. تقویت جریان کار (۴۰ دقیقه)

• به‌کارگیری **هر چهار بُعد 4D با هم** برای ساخت یک **رویهٔ تکرارپذیر** با AI.

• **سناریوی عملی:** امیلی، هماهنگ‌کنندهٔ توسعه، در حال آماده‌سازی گالای سالانهٔ سازمان — راه‌اندازی یک **سیستم پاسخ ایمیل با کمک AI**.

• **تکنیک: دسته‌بندی وظایف بر اساس اینکه AI چه چیزی را باید انجام دهد** و چه چیزی را نه.

۸. یکپارچه‌سازی (۴۵ دقیقه) «مهم»

دو نگرانی رایج دربارهٔ پذیرش سازمانی AI: 1. **چطور از وابستگی بیش‌ازحد به AI اجتناب کنم؟** 2. **چطور ارتباط انسانی را حفظ کنیم؟**

اهداف یادگیری: - یکپارچه‌سازی AI به شیوه‌هایی که **ظرفیت انسانی را تقویت** و **مأموریت را پیش ببرد** - **ساخت یک سیاست سازمانی AI** که بازتاب ارزش‌های شما باشد و **استفادهٔ پایدار** را تضمین کند

۹. قدم‌های بعدی (۱۵ دقیقه)

• دسترسی به **blueprintها و منابع** برای ادامهٔ مسیر.

• **به‌اشتراک‌گذاری آموخته‌ها با همکاران.**

• **چالش عملی:** یک کار واقعی از محیط کارتان انتخاب کنید و هر چهار شایستگی را رویش پیاده کنید.

نکات کلیدی برای مرور سریع

- ابتدا **ارزش‌ها و محدودیت‌های سازمان** را صریح کنید.
- **حریم خصوصی داده** بزرگ‌ترین نگرانی این بخش است ← **بهداشت داده** را تمرین کنید.
- برای اعتماد به تحلیل AI: **روی داده‌ای که جوابش را می‌دانید تست کنید**.
- جریان کار = دسته‌بندی وظایف + رویهٔ تکرارپذیر.
- **یک سیاست سازمانی AI بنویسید.**
- هدف: **تقویت ظرفیت انسانی**، نه جایگزینی آن.

دوره ۱۶ — AI Fluency for Small Businesses (سواد هوش مصنوعی برای کسب‌وکارهای کوچک)

موضوع: توسعهٔ AI Fluency برای افزایش اثرگذاری و کارایی کسب‌وکار، در عین وفاداری به مأموریت و ارزش‌ها. **تعداد درس:** ۱۰

خلاصهٔ کلی دوره

نسخهٔ ویژهٔ صاحبان کسب‌وکار کوچک، با **مطالعهٔ موردی واقعی: Mak، مدیرعامل شرکت MAKS Enterprises (TIPM Rebuilders)** که در چند درس نشان می‌دهد چطور هر دو حلقه را در کسب‌وکار واقعی‌اش به کار برده است.

«مهم» ساختار دو حلقه‌ای (تعریف دقیق این دوره)

حلقه	نام	نقش
حلقهٔ درونی (inner loop)	**Description & Discernment**	**هدایت** می‌کند — کار لحظه‌به‌لحظه با AI
حلقهٔ بیرونی (outer loop)	**Delegation & Diligence**	**قاب‌بندی** می‌کند — هر تعامل AI را چارچوب می‌دهد

نکات مهم هر درس

۱. AI Fluency برای کسب‌وکارهای کوچک (۱۵ دقیقه)

- تعریف AI Fluency و **چرا برای صاحبان کسب‌وکار کوچک اهمیت دارد**.
- **بیان صریح ارزش‌ها، اهداف و محدودیت‌ها** برای همکاری مؤثرتر.
- شناسایی اهداف برای یکپارچه‌سازی AI در **عملیات کسب‌وکار**.

۲. چارچوب 4D (۱۵ دقیقه)

- تعریف هر چهار شایستگی و کاربردشان در کار روزمره.
- **نکتهٔ کلیدی:** ساختار **حلقهٔ درونی/بیرونی** (جدول بالا).

۳. قابلیت‌ها و محدودیت‌های AI (۲۵ دقیقه)

- تعریف **AI مولد** و تفاوتش با انواع دیگر AI.
- **چگونگی ساخت و آموزش مدل‌های زبانی بزرگ** مثل کلود، و اینکه این برای کاربرد در کسب‌وکار **چه معنایی دارد** — چه کاری را قابل اتکا انجام می‌دهند و چه کاری را نه.

۴. کاوش کنید! — تمرین تعاملی Markov

- **«Text Your Friend Markov»** — تولیدکنندهٔ توکن بعدی **۱۰۰٪ قابل تفسیر**.

- همان بازی قدیمی ساخت پیام با کلمات پیشنهادی کیبورد.
- نشان می‌دهد پیشنهادها بر اساس **الگوی استفاده** ساخته می‌شوند، نه فهم.

۵. اصلاح و پالایش با AI (۳۰ دقیقه) — حلقهٔ Description–Discernment

- **Description:** ساخت پرامپت‌های مؤثر برای جمع‌آوری و ترکیب اطلاعات مرتبط با کسب‌وکار کوچک شما.
- **Discernment:** ارزیابی تحقیق تولیدشده از نظر **دقت، ارتباط و تناسب — پیش از اقدام بر اساس آن**.

مطالعهٔ موردی: Mak نشان می‌دهد چطور با نمونه‌های واقعی **پرامپت‌هایش را پالایش کرد** و با **تخصص حوزه‌ای و قضاوت خودش** با AI کار کرد.

> **«مهم» درس اصلی: تخصص شما جایگزین نمی‌شود — تبدیل به ابزار هدایت AI می‌شود.**

۶. استفادهٔ شفاف از AI (۳۰ دقیقه) — حلقهٔ Delegation–Diligence

- بیان نگرانی‌های حریم خصوصی و ارزیابی ابزارها بر اساس **سیاست‌های مدیریت داده**.
- تمرین **بهداشت داده** برای کار ایمن با اطلاعات حساس کسب‌وکار.

مطالعهٔ موردی Mak: - **چه چیزی را به AI واگذار کرد** - **چطور با مشتریان دربارهٔ استفاده از AI شفاف بود** - **چه چیزی را عمداً انسان‌محور نگه داشت**

> **حلقهٔ بیرونی هر تعامل AI را قاب‌بندی می‌کند.**

۷. جمع‌بندی همه‌چیز (Tying it all together)

۸. انسان در حلقه (۲۰ دقیقه)

دو نگرانی رایج: **وابستگی** و **حفظ ارتباط انسانی**.

اهداف: - یکپارچه‌سازی AI به شیوه‌هایی که **ظرفیت انسانی را تقویت** و **اهداف کسب‌وکار را پشتیبانی** کند. - **ساخت سیاست سازمانی AI** که بازتاب ارزش‌های شما باشد.

۹. جمع‌بندی و نگاه به جلو (۱۵ دقیقه)

> **نکتهٔ کلیدی: همین هفته با یک کار واقعی شروع کنید — چیزی که همین حالا منتظرتان است را انتخاب کنید و آن را زمین تمرین‌تان قرار دهید.**

- دسترسی به blueprintها و منابع.
- به‌اشتراک‌گذاری با همکاران و شرکای کسب‌وکار.

نکات کلیدی برای مرور سریع

- **حلقهٔ درونی = Description + Discernment** (هدایت) • **حلقهٔ بیرونی = Delegation + Diligence** (قاب‌بندی).
- **تخصص حوزه‌ای شما اهرم اصلی است**، نه مانع.
- **شفافیت با مشتری** دربارهٔ استفاده از AI — بخشی از Diligence.
- عمداً تصمیم بگیرید **چه چیزی انسان‌محور بماند**.
- **بهداشت داده** برای اطلاعات حساس کسب‌وکار.
- **همین هفته با یک کار واقعی شروع کنید.**

دوره ۱۷ — AI Fluency for Builders (سواد هوش مصنوعی برای سازندگان)

موضوع: ویژهٔ **سازندگان** — افراد محصول، طراحان و سازندگان — کسانی که **کل قوس از مسئله تا راه‌حل shipشده** را مالک‌اند. **تعداد درس:** ۱۰

خلاصهٔ کلی دوره

قوی‌ترین نسخهٔ تخصصی خانوادهٔ AI Fluency. تمرکز روی **چه چیزی را واگذار کنیم و چه چیزی را نه**، **زنجیرهٔ توصیف**، و **پنج عدسی ارزیابی کار تولیدشده توسط AI**.

نکات مهم هر درس

۱. خوش‌آمدید

اهداف: - تعریف AI Fluency و کاربردش در نقش شما به‌عنوان سازنده. - **ساخت یک «بریف قابل استفادهٔ مجدد»** که ارزش‌ها، محدودیت‌ها و زمینهٔ شما را به AI بدهد. - تصمیم‌گیری دربارهٔ **اینکه آیا، کِی و کجا** در فرایند ساخت واگذار کنید.

۲. چارچوب 4D (۲۰ دقیقه)

- تشخیص **حلقهٔ درونی از حلقهٔ بیرونی** و دانستن اینکه کِی هرکدام را اعمال کنید.
- **نگاشت کار فعلی‌تان به 4D** و یافتن **بزرگ‌ترین فرصت** خودتان.

۳. قابلیت‌ها و محدودیت‌های AI (۲۵ دقیقه)

- تعریف AI مولد و تفاوتش با انواع دیگر.
- **سه تحولی که مدل‌های زبانی مدرن را ممکن کرد.**
- شناسایی اینکه AI کنونی **چه کاری را خوب انجام می‌دهد و کجا قابل‌پیش‌بینی شکست می‌خورد.**

۴. Delegation و جعبه‌ابزار سازنده (۴۵ دقیقه) «مهم»

> **پیش از آنکه حتی یک خط کد بنویسید، ده‌ها تصمیم می‌گیرید که تعیین می‌کند آنچه می‌سازید اصلاً اهمیت دارد یا نه.**

اهداف یادگیری: - معرفی **جعبه‌ابزار سازنده** و اینکه AI در **هر مرحله** کجا ارزش اضافه می‌کند. - **«مهم» قاعدهٔ طلایی:** توضیح اینکه **چرا واگذاری «پیاده‌سازی» ایمن است اما واگذاری «قضاوت» نیست.** - نوشتن **تست‌های پذیرش (acceptance tests)** که **«تمام‌شدن» را تعریف کنند — پیش از آنکه حتی یک خط کد وجود داشته باشد.**

این درس، Delegation را **بازقاب‌بندی** می‌کند: سؤال «آیا باید...؟» نیست، بلکه **چه چیزی** را واگذار کنیم است.

۵. Description و ساختن چیزهای عالی (۴۵ دقیقه) «مهم»

> **بیشتر آموزش‌های AI به شما یاد می‌دهند پرامپت بهتری بنویسید. این لازم است اما کافی نیست برای ساختن محصولات عالی.**

زنجیرهٔ توصیف (The Description Chain): - ترجمهٔ **نیاز کاربر** به **دستور دقیق AI** - تشخیص اینکه **کِی یک شکست در توصیف آبشاری می‌شود** و ردیابی آن تا **حلقه‌ای که شکسته** - **بیان قصد از طریق تست‌ها** — تست‌هایی که دقیقاً به AI می‌گویند موفقیت چه شکلی است

۶. Discernment برای کد (۳۰ دقیقه) «مهم»

> **وقتی AI می‌تواند در چند دقیقه یک محصول کارا بسازد، «کار می‌کند» دیگر معیار نیست.**

«مهم» پنج عدسی ارزیابی کار تولیدشده توسط AI: 1. **درستی (correctness)** 2. **کیفیت (quality)** 3. **تناسب (fit)** 4. **تجربه (experience)** 5. **مسئولیت‌پذیری (responsibility)**

اهداف: - بالا بردن معیار کیفیت **فراتر از «کار می‌کند»** با یادگیری **حالت‌های شکست مخصوص محصولات ساخته‌شده با AI** - گرفتن **نقاط کور فنی قابل‌پیش‌بینی AI** پیش از رسیدن به پروداکشن

۷. Discernment برای تجربهٔ کاربری (۴۵ دقیقه)

> **همان‌طور که AI پیاده‌سازی را سرعت می‌دهد، طراحی به عامل تمایز تبدیل می‌شود.**

اصول UX که هنگام کار با AI بیشترین اهمیت را دارند: - **سلسله‌مراتب (hierarchy)** - **جریان‌های کاربر (user flows)** - **دسترس‌پذیری (accessibility)** - **الگوهای بازخورد (feedback patterns)**

مهارت‌ها: - نقد طراحی‌های تولیدشده توسط AI با **بازخوردی به‌قدر کافی مشخص که قابل اقدام باشد** - تصمیم‌گیری **آگاهانه دربارهٔ trade-off** بین **سرعت، پرداخت (polish) و نیاز کاربر**

۸. پشت آنچه می‌سازید بایستید (۴۵ دقیقه) «مهم»

> **Diligence در مدل سازنده یعنی مالکیت کامل: شما مسئول محصول هستید — از اینکه آیا اصلاً باید وجود داشته باشد، تا اینکه آیا بعد از عرضه به کاربران خدمت می‌کند.**

اهداف: - بیان اینکه **وقتی چیزی را که AI در ساختش کمک کرده ship می‌کنید، مالک چه چیزی هستید** - شناسایی **واقعیت‌های فنی که در زمان ship شدن رو می‌شوند** و ساخت **حلقه‌های بازخورد** برای گرفتن آن‌ها - **گرفتن تصمیم نهایی: ship کن، درستش کن، یا متوقفش کن**

۹. جمع‌بندی و نگاه به جلو (۱۵ دقیقه)

> **نکتهٔ کلیدی: AI در «وسط» قوی‌ترین است** — یعنی در پیاده‌سازی، نه در تعریف مسئله (ابتدا) و نه در قضاوت نهایی (انتها).

نکات کلیدی برای مرور سریع

- **«مهم» واگذاری پیاده‌سازی ایمن است؛ واگذاری قضاوت نیست.**
- **تست‌های پذیرش را قبل از کد بنویسید** — «تمام‌شدن» را تعریف کنید.
- **زنجیرهٔ توصیف:** نیاز کاربر → دستور دقیق. شکست در یک حلقه، آبشاری می‌شود.
- **پنج عدسی Discernment:** درستی، کیفیت، تناسب، تجربه، مسئولیت‌پذیری.
- **«کار می‌کند» معیار نیست.**
- UX عامل تمایز است: سلسله‌مراتب، جریان کاربر، دسترس‌پذیری، بازخورد.
- **Diligence = مالکیت کامل**، از «آیا باید وجود داشته باشد» تا بعد از عرضه.
- **AI در وسط قوی‌ترین است** — ابتدا و انتها مال شماست.

دوره ۱۸ — ساخت با Claude API (به‌همراه Amazon Bedrock و Google Cloud)

این بخش، سه دورهٔ زیر را در یک فصل ادغام می‌کند (چون حدود ۸۵٪ محتوایشان یکسان است):

دوره	تعداد درس
Building with the Claude API	۸۵
Claude with Amazon Bedrock	۸۳
Claude on Google Cloud (Vertex AI)	۹۳
مجموع	**۲۶۱**

> **تفاوت اصلی سه دوره فقط در «لایهٔ دسترسی» است** — نحوهٔ احراز هویت و فراخوانی مدل. **مفاهیم، ابزارها، RAG، MCP و معماری عامل‌ها یکسان‌اند.**

نقشهٔ کلی — هشت ماژول

۱. دسترسی به API و مبانی • ۲. ارزیابی پرامپت (Evals) • ۳. مهندسی پرامپت • ۴. استفاده از ابزار (Tool Use) • ۵. RAG • ۶. قابلیت‌های کلود • ۷. MCP • ۸. عامل‌ها و جریان‌های کاری

ماژول ۱ — دسترسی به API و مبانی

چرخهٔ عمر درخواست — پنج مرحله

درک کامل مسیر از لحظه‌ای که کاربر «ارسال» را می‌زند تا وقتی پاسخ کلود روی صفحه ظاهر می‌شود.

چرا مهم است؟ - طراحی **معماری امن** که کلیدهای API را محافظت کند - تنظیم **محدودیت توکن (token limits)** مناسب - مدیریت **دلایل توقف (stop reasons)** مختلف در منطق برنامه - **دیباگ** با فهم اینکه مشکل کجای خط لوله رخ داده

گرفتن کلید API

1. به کنسول Anthropic API بروید و وارد شوید
2. دکمهٔ **Get API Keys** (بالا سمت راست داشبورد)
3. دکمهٔ **Create Key**

اولین درخواست

- نصب پکیج‌ها و **پیکربندی امن کلید API**.

• نمونه‌های دوره با **پایتون** است.

گفت‌وگوهای چندنوبتی (Multi-Turn) «مهم»

> **مفهوم حیاتی: کلود هیچ بخشی از تاریخچهٔ گفت‌وگوی شما را ذخیره نمی‌کند. هر درخواست کاملاً مستقل است و هیچ حافظه‌ای از تبادل‌های قبلی ندارد.**

اگر می‌خواهید گفت‌وگوی چندنوبتی داشته باشید که کلود زمینهٔ قبلی را به یاد بیاورد، **خودتان باید کل تاریخچه را در هر درخواست بفرستید**.

پرامپت‌های سیستمی (System Prompts)

• راهی قدرتمند برای **سفارشی‌سازی نحوهٔ پاسخ کلود** — شکل‌دادن به **لحن، سبک و رویکرد**.

• **مثال:** ساخت چت‌بات معلم ریاضی. وقتی دانش‌آموز می‌پرسد «چطور `5x + 2 = 3` را برای x حل کنم؟»، می‌خواهید کلود مثل یک معلم رفتار کند، نه اینکه فقط جواب بدهد.

دما (Temperature) «مهم»

پارامتری که کنترل می‌کند پاسخ‌های کلود چقدر **قابل‌پیش‌بینی یا خلاقانه** باشند.

نیاز	دما
پاسخ‌های ثابت و واقع‌محور	**پایین**
طوفان فکری خلاقانه	**بالا**
بیشتر کارهای عمومی	**متوسط**

> **«هشدار» نکتهٔ مهم: دما خروجی متفاوت را تضمین نمی‌کند — فقط احتمال آن را تغییر می‌دهد.** حتی در دمای بالا، کلود ممکن است گاهی پاسخ‌های مشابه تولید کند.

استریم پاسخ (Response Streaming)

• **مشکل:** تولید پاسخ می‌تواند **۱۰ تا ۳۰ ثانیه** طول بکشد و کاربر به اسپینر لودینگ خیره بماند.

• **راه‌حل:** استریم — متن **تکه‌تکه (chunk by chunk)** همان‌طور که تولید می‌شود ظاهر می‌شود.

دادهٔ ساختاریافته (Structured Data)

• **مشکل رایج:** وقتی JSON، کد پایتون یا لیست می‌خواهید، کلود تمایل دارد **متن توضیحی دور محتوا اضافه کند**. گاهی فقط دادهٔ خام را می‌خواهید.

• تکنیک‌هایی برای گرفتن **فقط خروجی خام**.

ماژول ۲ — ارزیابی پرامپت (Prompt Evaluation) «مهم»

> **نوشتن پرامپت خوب فقط شروع کار است.** مهندسی پرامپت تکنیک نوشتن بهتر می‌دهد؛ **ارزیابی پرامپت اندازه می‌گیرد که آن پرامپت‌ها واقعاً چقدر خوب کار می‌کنند.**

جریان کاری معمول ارزیابی — پنج گام

1. **تدوین (Draft)** مجموعه‌داده
2. تولید **مجموعه‌دادهٔ آزمون (test dataset)**
3. ترکیب هر مورد آزمون با **قالب پرامپت**
4. ارسال به کلود
5. **نمره‌دهی (grading)** نتایج

انواع نمره‌دهنده (Graders)

نمره‌دهنده خروجی مدل را می‌گیرد و بازخورد قابل اندازه‌گیری برمی‌گرداند — معمولاً **عددی بین ۱ تا ۱۰** (۱۰ = کیفیت بالا).

سه رویکرد اصلی:

نوع	توضیح
نمره‌دهی مبتنی بر مدل (Model-based)	یک مدل، خروجی مدل دیگر را قضاوت می‌کند
نمره‌دهی مبتنی بر کد (Code-based)	اعتبارسنجی برنامه‌نویسی‌شده — مثلاً **صحت نحوی (syntax) کد تولیدشده و پیروی از قالب درست**
نمره‌دهی انسانی	قضاوت مستقیم انسان

- [دانلود] دانلود: `001_prompt_evals_complete.ipynb`

ماژول ۳ — مهندسی پرامپت

> **مهندسی پرامپت یعنی گرفتن پرامپتی که نوشته‌اید و بهبود آن برای خروجی‌های قابل‌اتکاتر و باکیفیت‌تر — از طریق «پالایش تکراری» (iterative refinement).**

فرایند: پرامپت پایه → ارزیابی عملکرد → اعمال نظام‌مند تکنیک‌ها → تکرار.

چهار تکنیک اصلی

۱) شفاف و مستقیم بودن (Being clear and direct)

۲) مشخص بودن (Being specific) - به‌جای اینکه همه‌چیز را به تفسیر مدل بسپارید، **دستورالعمل‌ها یا گام‌های روشن** بدهید.

۳) ساختاردهی با تگ‌های XML «مهم» - وقتی پرامپت شما محتوای زیادی دارد، کلود گاهی نمی‌فهمد کدام تکه‌ها به هم مربوط‌اند. - **تگ‌های XML** راهی ساده برای افزودن **ساختار و وضوح** هستند — به‌ویژه وقتی حجم زیادی داده را درج (interpolate) می‌کنید.

۴) ارائهٔ مثال (Providing examples) «مهم» - یکی از **مؤثرترین** تکنیک‌ها. - معروف به **one-shot** یا **multi-shot prompting**: دادن جفت‌های **ورودی/خروجی نمونه** برای هدایت پاسخ. - **مثال:** تحلیل احساسات — چند نمونه بدهید تا کلود دقیقاً دسته‌بندی مورد نظرتان را یاد بگیرد.

ماژول ۴ — استفاده از ابزار (Tool Use)

> **به‌صورت پیش‌فرض، کلود فقط اطلاعات دادهٔ آموزشی‌اش را می‌داند و به رویدادهای جاری، دادهٔ بلادرنگ یا سیستم‌های بیرونی دسترسی ندارد. ابزارها این محدودیت را حل می‌کنند.**

پروژهٔ عملی

ساخت سیستمی که به کلود یاد می‌دهد **برای تاریخ‌های آینده یادآور تنظیم کند**. مثال: «برای قرار دکترم یادآور بگذار. یک هفته بعد از پنجشنبه است.»

گام‌های پیاده‌سازی

۱) توابع ابزار (Tool Functions) — توابع پایتونی که کلود می‌تواند صدا بزند.

۲) شمای ابزار (Tool Schemas) «مهم» - بعد از نوشتن تابع، یک **JSON Schema** بسازید که به کلود بگوید تابع **چه آرگومان‌هایی می‌خواهد و چطور استفاده شود**. - این شما به‌عنوان **مستندات** عمل می‌کند که کلود می‌خواند. - **JSON Schema مخصوص AI نیست** — یک استاندارد اعتبارسنجی دادهٔ رایج است.

۳) مدیریت بلوک‌های پیام (Message Blocks) - با ابزارها، کلود **پیام‌های چندبلوکی** برمی‌گرداند که هم **متن** و هم **اطلاعات استفاده از ابزار** دارند.

۴) ارسال نتایج ابزار - وقتی کلود بلوک `tool_use` برمی‌گرداند، پارامترهای ورودی را استخراج و تابع را اجرا کنید، سپس نتیجه را برگردانید.

۵) گفت‌وگوهای چندنوبتی با ابزار «مهم» - گاهی کلود باید **چند ابزار را پشت سر هم** صدا بزند. - **مثال:** «۱۰۳ روز بعد از امروز چه روزی است؟» → اول تاریخ فعلی را بگیر، بعد ۱۰۳ روز اضافه کن.

۶) پیاده‌سازی حلقه

> **کلید تشخیص اینکه کلود ابزار می‌خواهد، در `stop_reason` است.** حلقه را ادامه دهید تا وقتی کلود دیگر درخواست ابزار نکند — آن یعنی پاسخ نهایی آماده است.

ابزارهای داخلی

ابزار	توضیح
Text Editor Tool	تنها ابزار داخلی که لازم نیست از صفر بسازید — به کلود توانایی کار با **فایل‌ها و دایرکتوری‌ها** مثل یک ویرایشگر متن می‌دهد
Web Search Tool	جست‌وجوی اینترنت برای اطلاعات جاری. **«هشدار» سازمان شما باید آن را در کنسول تنظیمات فعال کند.** برخلاف ابزارهای دیگر، **پیاده‌سازی را شما نمی‌نویسید**

فراخوانی ریزدانه‌ای ابزار (Fine-grained tool calling)

- ترکیب **استفاده از ابزار با استریم** → به‌روزرسانی بلادرنگ همان‌طور که AI آرگومان‌های ابزار را تولید می‌کند.

اختصاصی Bedrock / Vertex

- **Batch tool use** (استفادهٔ دسته‌ای از ابزار) — در Bedrock و Vertex
- **Flexible tool extraction** و **Structured data with tools** — در Bedrock
- **Controlling model output** — در Bedrock و Vertex

ماژول ۵ — RAG (تولید تقویت‌شده با بازیابی)

> **RAG تکنیکی است برای کار با اسناد بزرگ که در یک پرامپت جا نمی‌شوند.** به‌جای چپاندن همه‌چیز در یک پرامپت غول‌پیکر، سند را به **تکه (chunk)** می‌شکند و **فقط مرتبط‌ترین بخش‌ها** را هنگام پاسخ‌دهی وارد می‌کند.

۱) راهبردهای تکه‌تکه کردن متن (Text Chunking) «مهم»

> **یکی از حیاتی‌ترین گام‌ها. نحوهٔ شکستن اسناد مستقیماً روی کیفیت کل سیستم اثر می‌گذارد. راهبرد بد → زمینهٔ نامرتبط وارد پرامپت می‌شود → AI پاسخ کاملاً اشتباه می‌دهد.**

۲) امبدینگ متن (Text Embeddings)

- بعد از تکه‌کردن، باید بفهمید **کدام تکه‌ها به سؤال کاربر مرتبط‌اند** — این اساساً یک **مسئلهٔ جست‌وجو** است.
- رایج‌ترین روش: **جست‌وجوی معنایی (semantic search)** — بر پایهٔ **معنا** نه تطبیق رشته.

۳) جریان کامل RAG — پنج گام

1. **تکه‌کردن** متن منبع
2. تولید **امبدینگ‌ها**
3. ذخیره در **پایگاه دادهٔ برداری (vector database)**
4. انجام **جست‌وجوی شباهت (similarity search)**

5. درج تکه‌های مرتبط در پرامپت

۴) جست‌وجوی لغوی BM25 «مهم»

- **مشکل جست‌وجوی معنایی به‌تنهایی:** گاهی به **تطبیق دقیق کلمه** نیاز دارید که جست‌وجوی معنایی از دستش می‌دهد (مثلاً جست‌وجوی یک شمارهٔ حادثهٔ مشخص).
- **راه‌حل:** ترکیب جست‌وجوی معنایی با **جست‌وجوی لغوی (lexical)** به روش **BM25**.

۵) خط لولهٔ RAG چنداندیسی (Multi-Index)

- کلاس‌های `VectorIndex` و `BM25Index` **API تقریباً یکسانی** دارند (هر دو `add_document()` و `search()`).
- آن‌ها را در یک **خط لولهٔ جست‌وجوی یکپارچه** ترکیب کنید تا از **قوت هر دو رویکرد** بهره ببرید.

اختصاصی Bedrock / Vertex

- **Reranking results** (رتبه‌بندی مجدد نتایج)
- **Contextual retrieval** (بازیابی زمینه‌ای)

ماژول ۶ — قابلیت‌های کلود

تفکر گسترده (Extended Thinking)

- «کاغذ چرک‌نویس» کلود — به مدل زمان می‌دهد پیش از پاسخ نهایی روی مسائل پیچیده کار کند، و **استدلال قابل مشاهده است**.
- **«هشدار» با برخی قابلیت‌های دیگر سازگار نیست** — به‌ویژه **پیش‌پرکردن پیام (message pre-filling)** و **temperature**.

پشتیبانی از تصویر (Image Support)

- می‌توانید تصاویر را در پیام‌ها بگنجانید: توصیف محتوا، مقایسهٔ چند تصویر، شمارش اشیا، تحلیل بصری پیچیده.
- **«هشدار» محدودیت: حداکثر ۱۰۰ تصویر** در یک درخواست.

پشتیبانی از PDF

- کلود می‌تواند فایل‌های PDF را **مستقیماً بخواند و تحلیل کند**.
- کد تقریباً مشابه پردازش تصویر است؛ تفاوت اصلی در **مشخص‌کردن نوع فایل** است.

استنادها (Citations) «مهم»

- وقتی کلود بر اساس اسنادی که می‌دهید پاسخ می‌دهد، کاربران ممکن است فکر کنند از دادهٔ آموزشی‌اش می‌گوید.
- **Citations** به کلود اجازه می‌دهد **به بخش‌های مشخصی از اسناد منبع ارجاع دهد** و به کاربر نشان دهد **هر تکه اطلاعات دقیقاً از کجا آمده**.

کش پرامپت (Prompt Caching) «مهم»

> **کش پرامپت با استفادهٔ مجدد از کار محاسباتی درخواست‌های قبلی، پاسخ‌ها را سریع‌تر و هزینهٔ تولید متن را کمتر می‌کند.**

قواعد کش: - درخواست **اول** کار پردازشی را **در کش می‌نویسد** - درخواست‌های **بعدی** از کش **می‌خوانند** → هم سریع‌تر، هم ارزان‌تر - **«هشدار» فقط وقتی کار می‌کند که مکرراً محتوای یکسان بفرستید** - **طول عمر کش: یک ساعت**

اختصاصی Claude API

Code Execution + Files API - Files API: راه جایگزین برای آپلود فایل — به‌جای انکود کردن تصاویر/اسناد در هر درخواست. - ترکیب این دو، امکانات جالبی برای **واگذاری کارهای پیچیده** به کلود باز می‌کند.

ماژول ۷ — MCP (پروتکل زمینهٔ مدل)

> این ماژول عیناً همان محتوای **دورهٔ ۱۰** است. برای جزئیات کامل به آن فصل مراجعه کنید.

خلاصهٔ سریع: - **MCP = لایهٔ ارتباطی** که بار تعریف و اجرای ابزارها را از سرور شما به **سرورهای تخصصی MCP** منتقل می‌کند. - **پروژهٔ عملی:** یک **چت‌بات CLI** که از طریق خط فرمان با مجموعه‌ای از اسناد تعامل می‌کند — شامل هر دو بخش **کلاینت و سرور MCP**. - **سه پریمیتیو:** Tools (مدل کنترل می‌کند) • Resources (برنامه کنترل می‌کند) • Prompts (کاربر کنترل می‌کند). - **مستقل از حامل (transport agnostic)**. - **بازرس مرورگری** برای تست بلادرنگ سرور.

ماژول ۸ — عامل‌ها و جریان‌های کاری «مهم»

> **جریان‌های کاری (workflows) و عامل‌ها (agents) هر دو راهبردهایی برای انجام کارهایی هستند که کلود نمی‌تواند در یک درخواست تمام کند.**

«مهم» قاعدهٔ انتخاب

وضعیت	انتخاب
می‌توانید گام‌های لازم را دقیقاً مشخص کنید	**Workflow** — زنجیرهٔ از پیش تعریف‌شدهٔ فراخوانی‌ها
نمی‌دانید گام‌ها چه باید باشند	**Agent** — هدف + ابزارها بدهید، بگذارید خودش بفهمد

> نکتهٔ ظریف: **شما در طول این دوره هر دو را ساخته‌اید** — هر وقت ابزار دادید و گذاشتید کلود خودش راه را پیدا کند، یک **عامل** ساخته‌اید.

سه الگوی جریان کاری

۱) جریان‌های موازی‌سازی (Parallelization) - **مشکل:** پرامپت‌های تکی پیچیده وقتی سعی می‌کنید کار پیچیده‌ای را مؤثر پیاده کنید، از هم می‌پاشند. - **راه‌حل:** شکستن کار پیچیده به **قطعات متمرکز و موازی**.

۲) جریان‌های زنجیره‌ای (Chaining) - شکستن یک کار بزرگ و پیچیده به **زیرکارهای کوچک و متوالی**. - به‌ویژه ارزشمند برای **پرامپت‌های طولانی** که کلود در مدیریت یکنواخت آن‌ها مشکل دارد.

۳) جریان‌های مسیریابی (Routing) - **مشکل:** انواع مختلف درخواست کاربر به **رویکردهای متفاوت** نیاز دارند؛ یک پرامپت «یک‌اندازه برای همه» جواب نمی‌دهد. - **راه‌حل: دسته‌بندی درخواست‌های ورودی** و مسیریابی آن‌ها به **خط لوله‌های پردازش تخصصی**. - **مثال:** ابزار بازاریابی شبکه‌های اجتماعی که از موضوع کاربر اسکریپت ویدیو می‌سازد.

بازرسی محیط (Environment Inspection) «مهم»

> **مفهومی حیاتی که اغلب نادیده گرفته می‌شود: کلود کورکورانه عمل می‌کند.** برای مؤثر بودن، **باید بتواند نتایج اقدامات خودش را مشاهده و درک کند.**

مثال: در Computer Use، هر بار که کلود متنی تایپ یا دکمه‌ای کلیک می‌کند، **بلافاصله باید نتیجه را ببیند** (اسکرین‌شات جدید) تا بداند بعد چه کند.

اپلیکیشن‌های Anthropic به‌عنوان نمونهٔ عامل

Claude Code - دستیار کدنویسی مبتنی بر ترمینال. - **ابزارها:** عملیات فایل (جست‌وجو، خواندن، ویرایش) • دستورات ترمینال • و بیشتر. - **دستور `/init` :** اولین کاری که در شروع پروژه انجام می‌دهید. - **کلاینت MCP داخلی دارد** → با اتصال سرورهای MCP می‌توانید توانایی‌هایش را به‌شدت گسترش دهید.

** Computer Use **(در دوره‌های Bedrock و Vertex) - نحوهٔ کار Computer Use و** ویژگی‌های عامل‌ها**.

تفاوت‌های اختصاصی هر پلتفرم

Claude API (مستقیم از Anthropic)

- `Getting an API key` — کلید از کنسول Anthropic
- **Code execution و Files API**
- **Fine grained tool calling**
- ماژول کامل **Agents and workflows** (موازی‌سازی، زنجیره‌ای، مسیریابی)

Amazon Bedrock

- بدون گام مجزای «گرفتن کلید API» — از **احراز هویت AWS** استفاده می‌کند
- **Batch tool use** • **Flexible tool extraction** • **Structured data with tools**
- **Reranking results** • **Contextual retrieval**

- **Computer Use** و نحوهٔ کارش
- **Automated debugging** • **Parallelizing Claude Code**

Google Cloud (Vertex AI) — کامل‌ترین نسخه

- درس اختصاصی `Vertex AI Setup`
- **هم** Computer Use را دارد **هم** ماژول کامل Agents and workflows
- **The batch tool** • **Contextual retrieval** • **Reranking results**
- **Automated debugging** • **Parallelizing Claude Code**

نکات کلیدی برای مرور سریع

- **کلود تاریخچه را ذخیره نمی‌کند** — شما باید هر بار کل گفت‌وگو را بفرستید.
- **دما:** پایین = ثابت • بالا = خلاق. تضمین نمی‌کند، فقط احتمال را تغییر می‌دهد.
- **Evals قبل از پروداکشن** — ۵ گام، سه نوع نمره‌دهنده (مدل‌محور، کدمحور، انسانی).
- **چهار تکنیک پرامپت:** شفاف و مستقیم • مشخص • تگ XML • مثال (one/multi-shot).
- `stop_reason` کلید حلقهٔ ابزار است.
- **JSON Schema** = مستندات ابزار برای کلود.
- **RAG:** تکه‌کردن → امبدینگ → پایگاه برداری → جست‌وجوی شباهت. **کیفیت تکه‌کردن حیاتی‌ترین عامل است.**
- **BM25** را با جست‌وجوی معنایی ترکیب کنید (چنداندیسی).
- **کش پرامپت:** فقط با محتوای تکراری یکسان؛ عمر یک‌ساعته.
- **Extended thinking با temperature و pre-filling سازگار نیست.**
- **Workflow وقتی گام‌ها را می‌دانید؛ Agent وقتی نمی‌دانید.**
- سه الگوی جریان کاری: **موازی‌سازی، زنجیره‌ای، مسیریابی**.
- **عامل باید نتیجهٔ اقدامش را ببیند** (بازرسی محیط) وگرنه کور است.