3 продвинутых приёма для Claude Agent Skills из официального гайда Anthropic

Большинство советов по skills сводятся к «добавь SKILL.md и опиши, что он делает». Но у Anthropic в инженерном блоге и в документации по best practices описаны более тонкие приёмы, которые отличают skill, который реально работает без сюрпризов, от skill, который ломается на первом нестандартном случае.

1. Держи отдельную секцию с частыми ошибками

- Официальные skills от Anthropic держат отдельный блок вида «Avoid (Common Mistakes)» с конкретными пунктами провалов — не общими принципами, а буквально «не центрируй текст», «не забывай про отступы текстового блока», «не используй одинаковый layout на всех слайдах».
- В общем гайде по написанию skills этот же принцип называется **«Anti-patterns to avoid»** — два конкретных примера оттуда: не используй Windows-стиль путей (`scripts\file.py`) даже на Windows, только `/` -слэши; и не предлагай сразу пять вариантов инструмента («можно pypdf, или pdfplumber, или PyMuPDF…») — дай один вариант по умолчанию с явной оговоркой, когда переключаться на альтернативу.
- **Зачем:** явный список известных провалов экономит итерации — Claude сразу видит, на какие грабли уже наступали другие, вместо того чтобы находить их заново в каждой сессии.

2. Progressive disclosure — skill это папка, а не один длинный промпт

- Skill физически устроен как папка с файлом `SKILL.md` ; при старте агент подгружает в системный промпт только `name` и `description` каждого установленного skill — это первый уровень раскрытия информации.
- Если задача подходит под skill, Claude читает весь `SKILL.md` целиком — второй уровень; а если внутри есть ссылки на дополнительные файлы вроде `forms.md` для узкого сценария, Claude открывает их только когда они реально нужны — третий уровень и далее.
- Благодаря такой многоуровневой загрузке общий объём материала, который можно упаковать в skill, практически не ограничен — тяжёлые детали просто не попадают в контекст, пока не понадобятся.
- **Практика:** когда `SKILL.md` разрастается, лишний контент выносится в отдельные файлы и просто указывается ссылкой — это держит ядро skill компактным и читаемым.

3. Не зарельсовывай — подбирай степень свободы под задачу

- Уровень детализации инструкций должен соответствовать тому, насколько задача хрупкая и вариативная, а не быть одинаково жёстким всегда.
- **Высокая свобода:** обычные текстовые инструкции — подходит там, где годятся разные подходы и решение зависит от контекста, например ревью кода.
- **Низкая свобода:** конкретный скрипт почти без параметров — для операций, где ошибка дорого стоит и нужна строгая дисциплина, например миграция базы данных: «запусти именно этот скрипт, не меняй команду и не добавляй флаги».
- Между этими крайностями — «средняя свобода»: шаблон или псевдокод с параметрами для задач, где есть предпочтительный паттерн, но допустима вариация.

> *Claude как робот на тропе: узкий мост с обрывами по бокам требует точных инструкций и ограждений — низкая свобода. Открытое поле без препятствий — достаточно общего направления, дальше модель сама найдёт путь.*