

/GOAL IN CLAUDE CODE: EIN ZIEL STATT 100 PROMPTS

6 PROMPTS

TOOL: CLAUDE CODE

STAND 2026-10-05

Du gibst Claude eine Aufgabe wie einem guten Mitarbeiter: „Fertig ist es, wenn alle Rechnungen sortiert im Jahresordner liegen.“ Danach arbeitet Claude Runde für Runde allein weiter und hört erst auf, wenn ein zweites Modell bestätigt, dass das Ziel erreicht ist. Das Herzstück ist die **Ziel-Formel** in Schritt 6, an der sich entscheidet, ob /goal glänzt oder im Kreis läuft. Dazu drei Use Cases zum Nachbauen, bis hin zum Loopsystem mit eigenem Executor.

3

USE CASES ZUM NACHBAUEN

ca. 10 Min

SETUP INKL. EXECUTOR

4.000

ZEICHEN PRO ZIEL

WAS DU BRAUCHST

Claude Pro oder Max. Pro kostet rund 18 € im Monat (rund 15 € im Jahresabo), Max ab rund 90 € im Monat, jeweils zzgl. Mehrwertsteuer. Im Free-Plan ist Claude Code nicht enthalten. Chat und Claude Code teilen sich dasselbe Nutzungskontingent.

Die Claude-Desktop-App (Mac oder Windows). Oben gibt es die Tabs Chat, Cowork und Code. /goal läuft im **Code-Tab**, ganz ohne Terminal. Für Fortgeschrittene: Claude Code gibt es auch als Kommandozeilen-Programm im Terminal, dort funktioniert /goal identisch.

Eine aktuelle Version: /goal gibt es ab Claude Code 2.1.139. Die Desktop-App hält sich normalerweise selbst aktuell.

Einen Projektordner. Claude Code arbeitet immer in einem Ordner, den du auswählst und freigibst. Nur dort wird gearbeitet.

Kurz erklärt: *Claude Code* ist der Teil von Claude, der direkt in einem Ordner auf deinem Rechner arbeitet: Dateien anlegen, ändern, Befehle ausführen. Ein *Turn* ist eine komplette Arbeitsrunde von Claude.

SO FUNKTIONIERT /GOAL

Normalerweise arbeitest du in Häppchen: Du promptest, Claude antwortet, du prüfst, du promptest wieder. Bei großen Aufgaben wird genau das Hin und Her zum Engpass. /goal dreht das um. Du beschreibst nur den Endzustand, dann läuft eine Schleife:

1. Claude arbeitet eine komplette Runde an der Aufgabe: plant, erstellt Dateien, führt Schritte aus.

2. Nach der Runde prüft ein zweites, unabhängiges Modell (ein kleines, schnelles) den Verlauf: Ist das Fertig-Kriterium erfüllt? Es antwortet mit Ja oder Nein plus einer kurzen Begründung.
3. Bei Nein startet sofort die nächste Runde. Claude nimmt die Begründung als Hinweis, was noch fehlt.
4. Bei Ja stoppt die Schleife, das Ziel wird abgehakt und du bekommst die Kontrolle zurück.

Wie ein Handwerker-Auftrag mit Abnahme: Der Handwerker arbeitet, ein unabhängiger Bauleiter schaut nach jedem Abschnitt drauf und sagt „noch nicht“ oder „abgenommen“. Es prüft nie derselbe, der gearbeitet hat.

KLASSISCHES PROMPTEN VS. /GOAL

	KLASSISCH	MIT /GOAL
Planung	Du zerlegst die Aufgabe selbst in Schritte	Du beschreibst nur den Endzustand
Weitermachen	Nach jeder Antwort musst du anstoßen	Claude startet die nächste Runde selbst
Wann ist es fertig	Dein Bauchgefühl	Ein zweites Modell prüft gegen dein Kriterium
Gut für	Kleine, schnelle Aufgaben	Projekte mit messbarem Endzustand

01 CODE-TAB ÖFFNEN

Starte die Claude-Desktop-App und klick oben auf **Code**.

Tippe zum Test `/goal` ins Eingabefeld. Erscheint der Befehl in der Vorschlagsliste, ist alles bereit. Erscheint ein Hinweis, sagt dir Claude Code direkt dort, woran es liegt (zum Beispiel veraltete Version oder Ordner-Zugriff noch nicht bestätigt).

01 PROJEKTORDNER WÄHLEN

Wähle den Ordner, in dem gearbeitet werden soll, zum Beispiel den Ordner mit deinen Rechnungen oder deinem Website-Projekt, und bestätige den Zugriff. Für den ersten Test legst du am besten einen eigenen **Übungsordner** an.

01 AUTO-MODUS EINSCHALTEN

Neben dem Senden-Button sitzt die **Modus-Auswahl**. Wähle dort **Auto**.

Im Auto-Modus laufen Arbeitsschritte ohne ständige Einzel-Nachfragen durch. Im Hintergrund prüft trotzdem eine Sicherheitsebene jede Aktion und blockiert Riskantes, etwa Löschen außerhalb des Projekts. Ohne Auto-Modus fragt Claude vor bestimmten Befehlen nach, und die Schleife wartet auf dich.

01

ZIEL SETZEN

Tippe `/goal`, dahinter dein Ziel, und schick es ab. Die Schleife startet sofort, ein zusätzlicher Prompt ist nicht nötig. Solange das Ziel aktiv ist, zeigt eine Anzeige, wie lange es schon läuft.

Dein erster Testlauf im Übungsordner:

TIPP

Beobachte beim ersten Lauf die Begründungen des Prüf-Modells nach jeder Runde. So bekommst du schnell ein Gefühl dafür, wie es dein Ziel liest. Genau daraus leitet sich die Ziel-Formel ab.

DEIN ERSTER TESTLAUF

```
/goal Im Ordner liegt eine Datei ideen.md mit 10 nummerierten Content-Ideen zum Thema "KI im Alltag", jede mit einem Satz Beschreibung. Beweis: Zeige am Ende den Inhalt der Datei. Stoppe spätestens nach 5 Turns.
```

01

KONTROLLE BEHALTEN

`/goal` ohne Text zeigt den Status: das Ziel, die bisherigen Runden, den Verbrauch und die letzte Begründung des Prüf-Modells.

`/goal clear` bricht sofort ab.

Ein neues `/goal ...` ersetzt das alte. Es läuft immer nur ein Ziel gleichzeitig.

01

DAS PERFEKTE ZIEL FORMULIEREN

Ob `/goal` brilliant oder frustrierend ist, entscheidet sich fast nur hier. Das Prüf-Modell führt selbst nichts aus und öffnet keine Dateien. Es liest nur den Gesprächsverlauf. Claude muss den Beweis also sichtbar in den Verlauf bringen. Ein Ziel, das zuverlässig funktioniert, hat vier Bausteine:

1. **Messbarer Endzustand:** Was ist am Ende wahr? Eine Dateianzahl, ein leerer Ordner, ein fehlerfreier Testlauf. Etwas, das man eindeutig mit Ja oder Nein beantworten kann.
2. **Beweis:** Womit zeigt Claude, dass es stimmt? Eine Liste ausgeben, ein Prüfskript laufen lassen, ein Testergebnis zeigen.
3. **Grenzen:** Was darf sich nicht ändern? „Lösche nichts“, „ändere keine Dateiinhalte“, „nur in diesem Unterordner“.
4. **Turn-Limit:** „Stoppe spätestens nach 20 Turns.“ Deine Notbremse gegen Endlos-Schleifen, und sie schont dein Kontingent.

Platz hast du genug: bis zu 4.000 Zeichen pro Ziel.

PROMPT 01: DIE ZIEL-FORMEL ZUM AUSFÜLLEN

```
/goal [ENDZUSTAND: Was ist am Ende wahr?]. Beweis: [Womit zeigt Claude, dass es stimmt, z. B. ein Testlauf, eine Dateiliste, ein Prüfskript?]. [GRENZEN: Was darf sich dabei nicht ändern?]. Stoppe spätestens nach [N] Turns.
```

WEICHE ZIELE VS. MESSBARE ZIELE

	ZU WEICH	MESSBAR
Website	Mach die Website besser	Alle 5 Sektionen existieren, das Prüfskript läuft fehlerfrei durch
Ordner	Räum den Ordner auf	Der Ordner Eingang ist leer, alle Dateien liegen umbenannt in Jahresordnern
Texte	Schreib gute Texte	Pro Stellenanzeige im Ordner liegt genau ein Anschreiben als eigene Datei

DIE FAUSTREGEL

„Besser“, „schöner“, „professioneller“ kann das Prüf-Modell nicht messen. Wenn ein Kollege am Telefon nicht eindeutig mit Ja oder Nein antworten könnte, ob dein Kriterium erfüllt ist, formuliere es um.

01

USE CASE 1: RECHNUNGS-CHAOS AUFRÄUMEN

Ein Ordner voller PDFs mit Namen wie `scan_final_2.pdf` oder `Rechnung(3).pdf`. Claude Code öffnet die PDFs, liest Firma und Datum heraus, benennt die Dateien einheitlich um und sortiert sie in Jahresordnern. Das Fertig-Kriterium ist herrlich eindeutig: Der Eingangsordner ist leer.

Selbst prüfen: Öffne danach 3 bis 4 Dateien und check, ob Firma und Monat im neuen Namen stimmen. Die `uebersicht.md` zeigt dir auf einen Blick, welche alte Datei zu welchem neuen Namen wurde.

PROMPT 02: RECHNUNGEN SORTIEREN

```
/goal Alle PDF-Rechnungen im Ordner "Eingang" sind nach dem Schema JJJJ-MM-Firmenname.pdf umbenannt und in Jahresordner (z. B. 2025, 2026) verschoben. Beweis: Der Ordner "Eingang" ist leer und im Hauptordner liegt eine uebersicht.md mit alter und neuer Bezeichnung jeder Datei. Ändere keine Dateiinhalte und lösche nichts. Stoppe spätestens nach 15 Turns.
```

USE CASE 2: LANDINGPAGE, BIS SIE FERTIG IST

Mit `/goal` liefert Claude nicht die erste Version ab, sondern baut, prüft und bessert nach, bis ein Prüfskript objektiv bestätigt, dass alles da ist. Das Prüfskript schreibt Claude selbst. Sein fehlerfreier Durchlauf ist der Beweis, den das Prüf-Modell sehen will.

TIPP

Leg vor dem Start eine Datei `inhalt.md` mit deinen echten Texten, Angeboten und Kontaktdaten in den Ordner und ergänze im Ziel: „Verwende ausschließlich die Inhalte aus `inhalt.md`.“ So wird die Seite deine und nicht generisch.

PROMPT 03: LANDINGPAGE BIS FERTIG

```
/goal Im Ordner liegt eine fertige einseitige Landingpage: index.html mit den Sektionen Hero, Vorteile, Über mich, FAQ und Kontakt, komplett auf Deutsch, responsiv und ausschließlich mit echten, fertig formulierten Texten. Beweis: Ein von dir geschriebenes Prüfskript läuft fehlerfrei durch und bestätigt, dass alle 5 Sektionen existieren und alle internen Links ein Ziel haben. Stoppe spätestens nach 20 Turns.
```

USE CASE 3: DAS LOOPSYSTEM MIT EXECUTOR

Die stärkste Variante ist ein Rollen-Split: Das Hauptgespräch bleibt Planer und Kontrolleur, die eigentliche Arbeit erledigt ein **Executor-Subagent**. Ein Subagent ist ein Mini-Claude mit eigenem Gedächtnis für die Aufgabe und eigener Arbeitsanweisung, den das Hauptgespräch pro Aufgabe losschickt. Der Hauptverlauf bleibt übersichtlich, und wer prüft und abhakt, ist nicht derselbe, der ausgeführt hat.

Executor einrichten (einmalig, 2 Minuten):

1. Die Datei `executor.md` von der Guide-Seite herunterladen oder Claude im Code-Tab bitten, sie nach der Beschreibung anzulegen. Am schnellsten geht das mit dem Prompt darunter.
2. Im Projektordner liegt sie dann im Unterordner `.claude/agents/`. Soll der Executor in allen Projekten verfügbar sein, kommt sie stattdessen in `~/.claude/agents/` in deinem Benutzerordner.
3. Findet Claude den Subagent danach nicht: Claude Code einmal neu starten, dann ist er da.

EXECUTOR ANLEGEN LASSEN

Lege im Projektordner die Datei `.claude/agents/executor.md` an, mit genau diesem Inhalt:

```
---
name: executor
description: Führt genau eine klar umrissene Aufgabe aus der Aufgabenliste aus und
meldet das Ergebnis kompakt zurück. Wird vom Hauptgespräch pro Aufgabe losgeschickt.
---
```

Du bist der Executor. Regeln:

1. Du bearbeitest pro Einsatz genau eine Aufgabe.
2. Du erledigst sie vollständig, bevor du zurückmeldest.
3. Du erfindest nichts dazu und änderst nichts außerhalb der Aufgabe.
4. Am Ende meldest du kompakt: was du getan hast, welche Dateien entstanden oder geändert sind und wie man das Ergebnis prüfen kann.

Zeig mir danach den Inhalt der Datei.

DER LOOP: PLANEN, DELEGIEREN, PRÜFEN

Leg eine Datei `aufgaben.md` mit deiner Aufgabenliste an: eine Aufgabe pro Zeile, mit `[ ]` davor. Dann startest du das Ziel darunter.

Drei Kontrollebenen greifen ineinander: Der Executor arbeitet, das Hauptgespräch prüft jedes Teilergebnis, und das Prüf-Modell von `/goal` entscheidet unabhängig, ob das Gesamtziel erreicht ist. Jedes `[x]` ist ein sichtbarer Haken im Verlauf.

Selbst prüfen: Steht nach dem Lauf hinter jeder Aufgabe ein `[x]` und liegt zu jeder ein Ergebnis im Ordner? Hängt eine Aufgabe, findest du im Verlauf die Rückmeldung des Executors, woran es lag.

PROMPT 04: LOOPSYSTEM MIT EXECUTOR

`/goal` Alle Aufgaben in `aufgaben.md` sind erledigt und mit `[x]` abgehakt. Ablauf: Erstelle zuerst einen kurzen Plan, delegiere dann jede Aufgabe einzeln an den Subagent "executor" und prüfe jedes Ergebnis selbst nach, bevor du die Aufgabe abhakst. Beweis: `aufgaben.md` enthält keine offene Aufgabe mehr und pro Aufgabe liegt ein Ergebnis im Projektordner. Stoppe spätestens nach 30 Turns.

KOSTEN, EHRlich GERECHNET

Eine Schleife, die 20 Runden dreht, verbraucht das Kontingent von 20 Runden. Die Prüfschritte selbst kosten fast nichts, die eigentliche Arbeit aber schon. Fang mit kleinen Zielen an, schreib immer ein Turn-Limit ins Ziel und schneide große Vorhaben lieber in drei kleine Ziele nacheinander. Den aktuellen Verbrauch siehst du jederzeit mit `/goal` ohne Text.

WENN DAS NICHT KLAPPT

Die Schleife wartet ständig auf mich: Du bist noch im Standard-Modus. Auto-Modus einschalten (Schritt 3).

Das Ziel wird zu früh als erreicht gemeldet: Dein Kriterium ist zu weich oder ohne Beweis. Bau einen konkreten Nachweis ein, etwa „Prüfskript läuft fehlerfrei durch“ oder „zeige die Dateiliste“.

Die Schleife dreht sich im Kreis: Status mit `/goal` ansehen. Die letzte Begründung verrät, woran sich das Prüf-Modell stört. Mit `/goal clear` stoppen, präziser formulieren, neu starten.

/goal erscheint nicht: App aktualisieren (Claude Code 2.1.139 oder neuer) und den Ordner-Zugriff bestätigen.

Der Executor wird nicht gefunden: Prüf, ob die Datei wirklich in `.claude/agents/` liegt, und starte Claude Code einmal neu.

QUICK REFERENCE

`/goal [dein Ziel]` — Startet die Schleife sofort. Ein neues Ziel ersetzt das alte.

`/goal (ohne Text)` — Status: Ziel, Runden, Verbrauch, letzte Prüf-Begründung.

/goal clear — Bricht die Schleife sofort ab.

Messbarer Endzustand — Ein Kriterium, das eindeutig mit Ja oder Nein prüfbar ist.

Beweis einbauen — Das Prüf-Modell sieht nur, was Claude im Verlauf zeigt.

Turn-Limit — „Stoppe spätestens nach N Turns“ als Notbremse ins Ziel schreiben.

Auto-Modus — Modus-Auswahl neben dem Senden-Button.

HINWEIS

Hinweis: Dieser Guide dient zu Informations- und Bildungszwecken. Alle Inhalte spiegeln den Stand bei Erstellung wider, Oberflächen, Funktionen und Preise der genannten Tools können sich ändern. Prüfe Ergebnisse von KI-Tools immer selbst nach. @GPTMarlon übernimmt keine Haftung für Entscheidungen auf Grundlage dieser Inhalte. Dieser Guide stellt keine Anlage-, Steuer-, Rechts- oder Gesundheitsberatung dar. Lass Ergebnisse bei Bedarf fachlich prüfen.

Datenschutz (DSGVO): Claude Code arbeitet nur im freigegebenen Ordner. Leg dort nur Rechnungen und Dateien ab, die du verarbeiten darfst.