

De prompts a *loops*

Como hacer que Claude trabaje solo hasta terminar.

LAS PIEZAS DE UN LOOP

Antes de seguir, una aclaración sobre los conectores: por si solo, un loop solo puede trabajar en archivos locales. Con conectores, puede significar “leer mi email” o “revisar Google Drive”. En Claude Code los agregas con `claude mcp add` o desde el menu de conectores.

5 · SUBAGENTES

El agente que hace el trabajo no debería ser el que lo califica. Claude Code te deja definir agentes separados (en `.claude/agents/`) para que un agente haga la tarea y otro la verifique de forma independiente. Este paso es lo que hace que los loops sean lo bastante confiables como para correr sin supervision.

UN EJEMPLO REAL

El resumen de investigacion que se revisa a si mismo

Aqui esta la forma mas simple de ver por que un loop le gana a un prompt. Digamos que le pides a Claude que te escriba un resumen de una pagina sobre un tema. Esa es la tarea. La meta, la parte que lo convierte en un loop, es algo que se puede medir: cada afirmacion tiene al menos tres fuentes, y cada enlace de verdad abre a una pagina que respalda lo que dijo.

Sin un loop, Claude te entrega un resumen que se ve impecable con fuentes que suenan reales, pero que no existen, y no tienes ni idea de que se las invento. Un solo prompt no puede detectar su propia fuente inventada, porque confia en que tiene razon.

Con un loop, Claude escribe el resumen, y luego va enlace por enlace, abriendo cada fuente para confirmar que es real y que respalda sus datos. Asi sigue revisando hasta que cada fuente de la pagina es real y confiable. Eso es el loop haciendo la parte que antes hacias *a mano*.

COMO LO USARIAS EN LA PRACTICA

`/goal` escribe un caption de Instagram sobre [tu tema] donde cada dato o estadística que menciones tenga una fuente real que lo respalde. Antes de darlo por terminado, abre cada fuente para confirmar que el dato es correcto y esta actualizado. Si una fuente no existe o no respalda el dato, quitalo o reemplazalo por uno verificado. Detente solo cuando cada dato del caption este confirmado.

La unica parte que casi todos hacen mal

Tu objetivo tiene que ser algo medible para que Claude pueda calificarse a si mismo: numeros reales, un porcentaje o una lista de verificacion. No algo vago como “hazlo bueno”. Todo esto solo funciona si Claude puede puntuar su propio trabajo. Una meta difusa no le da nada que revisar, asi que simplemente para como un prompt normal.

Tu primer loop

Los dos comandos de los que hablamos disparan el loop. Un objetivo de una línea es más que suficiente para un trabajo pequeño, pero para uno más grande con muchos pasos, necesitas las reglas más extensas: donde encontrar la tarea a finalizar, como revisarse a sí mismo, como recordar y cuando parar. Rellena lo que está dentro de los corchetes, y luego pegas las reglas en Claude Code.

Copia estas reglas en Claude Code cuando tengas una meta con más de una pieza de trabajo dentro (una carpeta que limpiar, una lista que procesar). Convierte a Claude de una herramienta de una sola respuesta en un loop que descubre, ejecuta, verifica y rastrea el trabajo hasta que está terminado, así dejas de estar pendiente de cada paso.

Estas haciendo un loop, no respondiendo a un solo prompt. Aquí están las reglas:

META

Describe el estado terminado en una o dos frases. Ejemplo: “Cada uno de los 40 productos en mi catálogo tiene una descripción de entre 100 y 200 palabras, un precio actualizado y al menos 3 etiquetas.” Se específico sobre como se ve **TERMINADO**.

DONDE ESTA LA TAREA POR HACER

Dile donde se ubica la tarea. Ejemplos: “Abre mi archivo `catalogo.csv` y trata cada fila como un producto por revisar”, “busca en mi bandeja de entrada los correos sin responder de esta semana”, “revisa los comentarios de mi último post y trata cada uno sin responder como una tarea”.

COMO HACER EL TRABAJO

Trabaja en UN elemento a la vez. Terminalo por completo antes de empezar el siguiente. Sigue las normas del proyecto que encuentres en los archivos existentes. Imita el estilo, no inventes patrones nuevos. Si un elemento requiere una decisión que solo yo puedo tomar (gastar dinero, borrar cosas, contactar a una persona), DETENTE en ese elemento, agregalo a una lista de “necesita a Mariia” y pasa al siguiente.

COMO REVISARTE A TI MISMO

Despues de cada elemento, verificalo antes de marcarlo como terminado: [Elige lo que encaje: “vuelve a leer el archivo y confirma que cumple la meta”, “toma una captura de pantalla y confirma que se ve bien.”] Verificar significa evidencia, no confianza. Si la verificacion falla, arreglalo y verifica otra vez. Maximo 3 intentos por elemento, luego registralo como bloqueado y sigue.

COMO RECORDAR

Manten un archivo llamado LOOP-ESTADO.md. Despues de cada elemento, actualizalo con: nombre del elemento, estado (terminado / bloqueado / me necesita), que cambiaste y cualquier cosa que la proxima ejecucion necesite saber. Lee este archivo PRIMERO al inicio de cada ejecucion para que nunca repitas trabajo ya terminado.

CUANDO PARAR

Detente cuando: (a) cada elemento este terminado o registrado como bloqueado, o (b) hayas completado [N] elementos en esta ejecucion. Luego dame un reporte corto: que se termino, que esta bloqueado, que necesita mi decision. Empieza leyendo LOOP-ESTADO.md si existe, y luego encuentra el trabajo.

POR QUE IMPORTA LOOP-ESTADO.MD

El archivo de estado es lo que convierte diez ejecuciones separadas en un solo empleado continuo. Sin el, cada ejecucion empieza desde cero. Con el, puedes correr las mismas reglas segun un horario y el loop siempre retoma exactamente donde *se quedo*.

OJO · UN EJEMPLO MAS

Comparte un frame de Figma y pregunta: “Revisa el diseno de esta pantalla y dame todo lo que mi desarrollador necesita para construirla: medidas, espaciados, colores y los estados de cada boton. Luego escribe el codigo del front-end listo para que solo lo conecte.”

Cuando NO hacer un loop

Los loops no son gratis y no son para todo. Tres advertencias honestas antes de que construyas uno:

LAS TAREAS DE UNA SOLA VEZ NO NECESITAN LOOPS.

Si el trabajo es una sola respuesta (escribe este correo, resume este documento), un prompt es mas rapido. Los loops justifican su coste de configuracion en trabajo que se repite o que tiene varios elementos.

LOS LOOPS CONSUMEN MAS USO QUE LOS PROMPTS.

Un loop que se revisa a si mismo y reintenta se ejecuta varios turnos. En un plan de Claude, eso significa que vas a llegar a tus limites de uso mas rapido.

LA VERIFICACION ES TODO.

Un loop sin verificar solo comete errores mas rapido. Si no puedes describir como deberia revisar el loop su propio trabajo, la tarea todavia no esta lista para ser un loop. Dejala como un prompt hasta que puedas.

EMPIEZA ESTA SEMANA

Elige una tarea molesta de varios elementos que hayas estado haciendo a mano. Pega las reglas para terminarla, rellena los corchetes, usalo una vez mientras observas. Una vez que confies en el, programalo. Ese es todo el camino de prompts a loops.

de prompts a loops.