

Vibe Coding 提示詞模板庫

100+ 開發實戰範本，打開就能複製

Vista Cheng / <https://www.vista.tw>

這本模板庫是《零基礎 Vibe Coding》（碁峰資訊出版）的免費延伸資源。書裡教你提示詞的心法，這本手冊給你一套可以直接拿去用的開發現場說法：從專案起手、切版、做功能、串資料，到除錯、重構、測試與部署，每一個環節都幫你把話先想好、寫好。

怎麼用這本手冊

1. 遇到卡關的環節，先翻目錄找到對應類別。
2. 把範本裡的〔全形方括號〕換成你自己的內容。
3. 整段複製，貼進你的 AI 工具（Claude、Claude Code 或 Antigravity 都可以），送出。
4. 覺得好用的，收藏起來變成你自己的口袋範本。

你不需要背這 100 條提示詞。你只需要知道：卡住的時候，這裡有現成的說法可以參考。

💡 所有範本都通用於對話式 AI 與程式編輯器。若你已經在用 Claude Code、Codex 這類會自己動手改檔的 AI 開發工具，範本裡「請幫我做」的部分它會直接執行，效果更強。

目錄

提示詞的黃金結構	7
類別一、專案起手與規劃.....	8
1. 讓 AI 反問我，把模糊需求問清楚.....	8
2. 把大想法拆成一步步的里程碑	8
3. 請 AI 給我技術選型建議（並解釋給外行聽）	8

4. 要 AI 先寫規格書，再動手寫程式.....	9
5. 請 AI 估算工作量與難度.....	9
6. 開工前先盤點風險	9
7. 幫零基礎的我選最簡單能動的方案	9
8. 把一句話需求展開成完整功能清單.....	10
9. 決定第一個最小可用版本（MVP）做什麼.....	10
類別二、網站與前端切版.....	10
10. 產出一個 landing page 的整體骨架	10
11. 設計吸睛的 hero 主視覺區.....	11
12. 做一個好用的導覽列	11
13. 補上一個完整的頁尾	11
14. 讓網頁在手機上也好看（手機優先）.....	11
15. 加上深色模式切換.....	12
16. 加上克制、不干擾的進場動畫	12
17. 調出有質感的字體與間距	12
18. 檢查並修正無障礙對比.....	12
19. 複製一個網站給我的氣質	13
20. 把設計稿或擷圖描述成頁面.....	13
類別三、UI 元件與互動.....	13
21. 做一個含驗證與錯誤提示的表單.....	13
22. 做一個彈窗 modal.....	14
23. 做一個下拉選單	14
24. 做一個分頁 tab 切換.....	14
25. 做一個可展開的手風琴（accordion）	15
26. 做一個可排序、可篩選的資料表格	15

27. 加一張把資料視覺化的圖表.....	15
28. 做一個檔案上傳功能	15
29. 補上載入中與空狀態	16
30. 做一個通知 toast 提示	16
類別四、功能開發	16
31. 做登入與註冊功能.....	16
32. 做基本的增刪改查（ CRUD ）	17
33. 加一個搜尋功能	17
34. 做篩選與排序.....	17
35. 做分頁或無限捲動.....	17
36. 做一個購物車.....	18
37. 描述金流串接的需求（不碰機密）	18
38. 做寄信通知功能	18
39. 做匯出 CSV 或 PDF	19
40. 做多語系切換.....	19
41. 做權限分級	19
類別五、資料與後端.....	19
42. 設計資料庫 schema	19
43. 請 AI 幫我寫資料查詢	20
44. 建立一個 API 端點	20
45. 描述第三方 API 串接需求.....	20
46. 建立環境變數與金鑰觀念	21
47. 加上後端資料驗證.....	21
48. 加上快取讓速度變快	21
49. 建立一個排程任務.....	21

50. 規劃一次資料遷移.....	22
類別六、除錯與求救.....	22
51. 把錯誤原文原封不動貼給 AI	22
52. 用「預期 vs 實際」描述問題	22
53. 請 AI 幫我重現問題.....	23
54. 用二分法縮小問題範圍.....	23
55. 處理改了 A 卻壞了 B 的回歸.....	23
56. 應對 AI 說修好了但其實沒好.....	23
57. 卡住了，請 AI 換一個方向	24
58. 請 AI 白話翻譯這個錯誤在講什麼.....	24
59. 請 AI 加上防護與錯誤處理	24
60. 請 AI 列出所有可能原因再逐一排除	24
類別七、讀懂與學習程式.....	25
61. 白話翻譯這段程式.....	25
62. 逐行加上中文註解.....	25
63. 把邏輯畫成流程	25
64. 幫變數與函式取好名字.....	26
65. 給我兩種寫法並比較	26
66. 這段在做什麼，有沒有風險.....	26
67. 把術語翻成人話	26
68. 邊做邊教我	27
類別八、重構與優化.....	27
69. 把太長的檔案拆開.....	27
70. 提升可讀性但不改行為.....	27
71. 找出重複的程式碼.....	28

72. 改善效能與載入速度	28
73. 統一命名風格.....	28
74. 移除沒用到的東西.....	28
75. 加上清楚的結構與分層.....	29
76. 重構前先說明計畫.....	29
類別九、測試與品質把關.....	29
77. 幫我加基本測試	29
78. 列出邊界與極端情況	30
79. 驗證使用者輸入	30
80. 檢查無障礙	30
81. 基本資安檢查.....	30
82. 跨瀏覽器與手機檢查	31
83. 上線前檢查清單	31
84. 請它扮演找碴的 QA.....	31
85. 先自我檢查再回報.....	32
類別十、部署與上線.....	32
86. 帶我一步步部署	32
87. 選擇部署平臺.....	32
88. 設定環境變數.....	33
89. 綁自訂網域	33
90. 部署失敗時怎麼問.....	33
91. 確認線上版真的更新了.....	33
92. 上線後基本監控	34
93. 把部署步驟寫成我看得懂的清單.....	34
類別十一、內容行銷與 SEO	34

94. 寫 SEO 友善的標題與描述	34
95. 加上分享預覽（Open Graph）	35
96. 加上結構化資料	35
97. 寫多語版本	35
98. 搞懂 sitemap 與 robots	35
99. 把賣點寫成頁面文案	36
100. 無障礙與 SEO 一起顧	36
類別十二、AI 協作元技巧	36
101. 給 AI 設定角色人格	36
102. 限制它只能改某個範圍	37
103. 要它一步一步來	37
104. 要它先問清楚再動手	37
105. 指定輸出格式	38
106. 控制回覆長度	38
107. 要它每步報告在做什麼	38
108. 防止它亂加功能	38
109. 要它承認不確定，別硬掰	39
110. 長對話重開新局的交接	39
想走得更遠	39

提示詞的黃金結構

在開始翻範本之前，先記住這張圖。你會發現底下每一條範本，其實都是這六塊積木的不同組合。哪天沒有範本可抄，你也能自己疊出一句好提示詞。

提示詞的黃金結構

六塊積木疊出一句好指令：不用每塊都用上，缺哪塊就補哪塊



好提示詞 = 角色 + 情境 + 任務 + 約束 + 輸出格式 + 範例

提示詞的黃金結構：六塊積木疊出一句好指令

類別一、專案起手與規劃

不知道你有沒有過這種經驗：明明腦中有個很想做的東西，但一坐到電腦前就不知道第一步該打什麼字？這一關的重點不是急著寫程式，而是先把你到底要做什麼講清楚。就像蓋房子前要先畫藍圖，這幾條範本會幫你將模糊的靈感，變成 AI 聽得懂、也做得出來的計畫。

1. 讓 AI 反問我，把模糊需求問清楚

什麼時候用：你有個想法，但自己也說不清楚細節，怕一開口就講歪。

這樣說： > 我想做一個〔一句話描述你的想法，例如：幫我管理客戶回購提醒的小工具〕。我完全是零基礎，還沒想清楚細節。請你先不要寫程式，改成用問問題的方式，一次問我 3 到 5 個最關鍵的問題，幫我把需求釐清。問完後，用條列方式幫我整理成一份需求摘要讓我確認。

💡 這招叫反向訪談。與其自己硬想，不如讓 AI 當你的採訪記者。你會發現，光是回答它的問題，很多原本沒想到的細節就自己浮出來了。

2. 把大想法拆成一步步的里程碑

什麼時候用：想法很大很完整，但你不知道該從哪裡下手、先做哪塊？

這樣說： > 我最終想做出〔完整的產品願景，例如：一個能讓學員報名、上課、拿到證書的線上課程平臺〕。請幫我把這個大目標拆成 4 到 6 個里程碑，每個里程碑都是一個做完就能看到成果、也能實際使用的階段。請按照先簡單、能先用起來的順序排列，並說明為什麼這樣排。

💡 好的拆解是每一步都能單獨用，壞的拆解是全部做完才能用。如果 AI 給你的第一個里程碑還要等後面才能跑，請它重排。

3. 請 AI 給我技術選型建議（並解釋給外行聽）

什麼時候用：你不知道該用什麼工具或框架來做，看到一堆專有名詞就頭痛。

這樣說： > 我想做〔你的專案，例如：一個接受線上報名並收款的活動頁〕。我是零基礎、不懂寫程式，希望盡量用簡單、資源多、好維護的技術。請推薦一套適合我的技術組合，並用你會對一個完全外行的人怎麼解釋的方式，說明每個選擇是什麼、為什麼選它、有沒有更簡單的替代方案。

💡 一定要加上「解釋給外行聽」這句。不然 AI 很容易丟一堆術語，你複製貼上時根本不知道自己在寫些什麼？

4. 要 AI 先寫規格書，再動手寫程式

什麼時候用：你不想 AI 一上來就狂寫程式碼，想先確認方向對不對？

這樣說：> 接下來我們要做〔功能或專案名稱〕。在寫任何程式碼之前，請先幫我寫一份簡短的開發規格，包含這幾項：這個功能要解決什麼問題、使用者會怎麼操作、需要哪些畫面、資料要存哪些欄位、有哪些例外情況要處理。寫完等我確認沒問題，你再開始寫程式。

💡 這是 Vibe Coding 最關鍵的紀律：先對規格、再寫程式。先花五分鐘看規格，可以省下之後打掉重做的兩小時。

5. 請 AI 估算工作量與難度

什麼時候用：你想知道這個東西大概多難、要花多少力氣，好安排時間。

這樣說：> 我想做〔功能清單或專案描述〕。請以一個零基礎的人用 AI 輔助開發為前提，幫我評估每一項的難度（簡單／中等／困難），以及大概需要投入的心力。並特別標出哪幾項是新手最容易卡關的地雷，先讓我心裡有數。

💡 別問要花幾天的時間？因為 AI 很可能猜不準你的時間，所以，要問相對難度跟哪裡會卡？這才是對你有用的資訊。

6. 開工前先盤點風險

什麼時候用：專案要開始了，你想先知道哪裡可能出包，避免做到一半才發現。

這樣說：> 我準備開始做〔專案描述〕。在開工前，請幫我列出 5 個新手最常在這類專案踩到的坑，例如：技術限制、資料安全、費用暴增或上線後才發現的問題等。每一個都告訴我：可能怎麼發生、以及一個簡單的預防方法。

💡 把這份清單存起來，做到一半時拿出來對照。很多災難不是不會發生，只是你當初沒想到要防。

7. 幫零基礎的我選最簡單能動的方案

什麼時候用：同一件事有很多做法，你只想要最不容易出錯的那個。

這樣說：> 我要做〔具體功能，例如：讓使用者上傳一張照片並顯示出來〕。請給我對零基礎最友善的做法：步驟最少、需要設定的東西最少、最不容易出錯。如果有更強大但更複雜的做法，可以簡單提一下就好，但請明確告訴我你推薦哪一個以及原因。

💡 新手最大的陷阱是被更專業的做法誘惑。先求能動、能用，強大是以後的事。逼 AI 幫你二選一。

8. 把一句話需求展開成完整功能清單

什麼時候用：你只有一句模糊的目標，需要把它變成一張具體的要做哪些功能清單。

這樣說：> 我想做「〔一句話目標，例如：讓學員可以線上預約我的一對一諮詢〕」。請幫我把這句話展開成一份完整的功能清單，分成「一定要有（核心）」「有了更好（加分）」「以後再說（先不做）」三組。每項功能用一句白話描述使用者會怎麼用它。

💡 重點在那個以後再說的欄位。它幫你把野心先收進口袋，避免第一版就想做到十全十美，結果永遠做不完。

9. 決定第一個最小可用版本（MVP）做什麼

什麼時候用：功能一大堆，你想先做出一個小到能在一個週末完成、但已經有用的版本。

這樣說：> 這是我的完整功能清單：〔貼上清單〕。請幫我圈選出一個最小可用版本：只保留讓這個產品能被真正使用一次的最少功能，其他全部砍掉。告訴我這個版本包含哪些、砍掉哪些，以及為什麼留下的這些是不可或缺的核心。

💡 記住這句口訣：先做出醜但能用，再慢慢變好用又好看。第一版的目標是能上線給一個真人用，不是拿去參展。

類別二、網站與前端切版

如果說功能是房子的水電，那前端切版就是你看到、摸得到的裝潢與門面。這一關你要學會怎麼跟 AI 描述畫面長什麼樣，讓它幫你排出漂亮又好用的版面。你不用會畫圖，只要會用嘴巴形容，AI 就能幫你變出來。

10. 產出一個 landing page 的整體骨架

什麼時候用：你要做一個介紹產品或服務的單頁網站，想先有個完整的區塊架構。

這樣說：> 請幫我做一個〔產品或服務，例如：線上瑜伽課程〕的 landing page（單頁介紹網站）。請先規劃整個頁面由上到下有哪些區塊（例如主視覺、痛點、方案介紹、見證、常見問題、報名），每個區塊放什麼內容，再依這個結構做出來。整體風格要〔乾淨專業／溫暖親切／活潑年輕，擇一〕。

💡 先要區塊清單，再要成品。這樣你能在它動工前就調整順序，比做完再搬省事很多。

11. 設計吸睛的 hero 主視覺區

什麼時候用：訪客打開網頁第一眼看到的區塊，你想讓它一秒抓住人的注意力。

這樣說：> 請幫我設計網頁最上方的 hero 區（主視覺區）。主標題要傳達「〔你的核心賣點，例如：三個月從零學會做自己的網站〕」，下面放一句補充說明和一顆明顯的行動按鈕（例如「免費報名」）。版面要留白舒服、標題字夠大夠醒目，手機上也要好看。先給我 3 個不同語氣的主標題讓我挑。

💡 hero 區是全站最重要的黃金位置。讓 AI 先給你幾個標題選項，別讓它一次定稿，你會有更多好句子可挑。

12. 做一個好用的導覽列

什麼時候用：網站需要頂部選單，讓訪客能在不同區塊或頁面之間跳轉。

這樣說：> 請幫我做一個網站頂部的導覽列（navbar）。左邊放〔網站名稱或 logo 文字〕，右邊放這幾個連結：〔首頁、關於、方案、聯絡〕，最右邊一顆行動按鈕「〔立即報名〕」。手機上要收合成漢堡選單（點一下才展開）。捲動頁面時導覽列要固定在最上方。

💡 「手機上收合成漢堡選單」這句一定要講。不然導覽列在小螢幕上會擠成一團，這是新手最常忽略的細節。

13. 補上一個完整的頁尾

什麼時候用：網站最下方需要放版權、聯絡方式與次要連結的區塊。

這樣說：> 請幫我做網站最下方的頁尾（footer）。內容包含：〔品牌名稱與一句 slogan〕、幾個次要連結（隱私政策、服務條款、聯絡我們）、社群媒體圖示連結、以及最下面一行版權宣告「© 2026 〔你的名稱〕」。整體用比較低調的深色底，文字不要搶眼。

💡 頁尾雖然不起眼，卻是訪客找隱私政策、聯絡方式的地方。如果缺了它，網站會有種沒做完的感覺。

14. 讓網頁在手機上也好看（手機優先）

什麼時候用：你做好的頁面在電腦上正常，但在手機上跑版、擠成一團。

這樣說：> 我這個頁面在手機上顯示會跑版。請幫我改成手機優先的響應式設計（responsive）：先確保在窄螢幕（手機）上排版正常、文字不會太小、按鈕好按，再往上適配平板和電腦。請特別檢查圖片會不會超出畫面、以及左右會不會有多餘的橫向捲動。

💡 現在大多數訪客都用手機看網頁。養成先想手機、再想電腦的習慣，可以省下大量事後補救的功夫。

15. 加上深色模式切換

什麼時候用：你想讓網站能在明亮與深色兩種外觀之間切換，照顧不同使用者習慣。

這樣說：> 請幫我的網站加上深色模式（dark mode）。放一個切換按鈕，點一下就在亮色和深色之間切換，並記住使用者的選擇（下次來還是同一個模式）。切換時請確認：所有文字在兩種模式下都清楚可讀、背景和文字的對比夠強、按鈕和圖片也都要跟著調整，不要有某塊在深色下變成一片黑看不到字。

💡 深色模式最常翻車的地方是某段文字在深色底下還是深色，變成隱形。務必請 AI 兩種模式都逐一檢查文字顏色。

16. 加上克制、不干擾的進場動畫

什麼時候用：你想讓頁面元素在捲動時優雅地淡入，但又怕動得太浮誇很廉價。

這樣說：> 請幫我的頁面加上克制的進場動畫：當使用者往下捲動、某個區塊進入畫面時，讓它輕柔地淡入並微微上移。動畫要低調、快速（不要讓人等）、有質感，絕對不要那種閃爍、旋轉、彈跳的浮誇效果。同一個區塊只播一次，不要每次捲到都重播。

💡 動畫的最高境界是你感覺得到質感，卻說不出哪裡有改動。跟 AI 強調克制、有質感，比放任它自由發揮安全得多。

17. 調出有質感的字體與間距

什麼時候用：畫面內容都對，但整體看起來就是很陽春、少了設計感。

這樣說：> 我的頁面內容都對了，但看起來很陽春、沒有設計感。請只調整字體與間距來提升質感：選一組耐看的字體搭配、拉開標題和內文的字級差距、增加行距讓文字更好讀、放大區塊之間的留白讓版面呼吸。內容和功能都不要動，只優化視覺質感。

💡 留白和字級對比，是業餘與專業的分水嶺。很多時候頁面不好看，不是缺東西，而是東西塞太滿、擠太緊。

18. 檢查並修正無障礙對比

什麼時候用：你想確認網站上的文字，讓所有人（包含視力較弱者）都看得清楚。

這樣說： > 請幫我檢查整個頁面的文字可讀性（無障礙對比）。找出所有文字顏色和背景顏色對比不夠、看起來吃力的地方，例如淺灰字配白底、彩色按鈕上的文字看不清等。列出有問題的位置，並幫我調整到清楚好讀的程度，同時盡量保留原本的設計風格。

💡 淺灰配白底是最常見的地雷，設計師覺得高級，長輩和陽光下的手機使用者卻看得很痛苦。對比清楚永遠比淡淡的很美重要。

19. 複製一個網站給我的氣質

什麼時候用： 你看到某個網站的風格很喜歡，想做出類似感覺但屬於自己的版本。

這樣說： > 我很喜歡〔某個網站或風格，例如：**Apple** 官網那種簡潔高級感〕的氣質。請不要照抄，而是抓出它讓人覺得高級的幾個關鍵特徵（例如大量留白、克制的配色、精緻的字體、簡潔的排版），把這些特徵套用到我的〔你的網站主題〕上，做出屬於我自己、但有同樣質感的版面。

💡 講氣質和特徵，而不是照抄。這樣做出來是你的品味長進了，而不只是複製別人的畫面。

20. 把設計稿或擷圖描述成頁面

什麼時候用： 你腦中或手邊有個畫面樣子（草圖、擷圖或參考圖），想讓 AI 照著做出來。

這樣說： > 我想做一個像這樣的畫面：〔詳細描述版面，例如：最上面一張大圖，圖上疊白色大標題；中間三欄並排的方塊，每欄有一個圖示、一個小標、兩行說明；最下面一顆置中的按鈕〕。請照這個描述做出來，用〔你偏好的配色〕，並確保手機上三欄會自動變成上下堆疊。

💡 描述版面時，用「上面、中間或下面」「左邊、右邊」「幾欄並排」這種空間方位詞，AI 最容易抓對位置。附上擷圖效果更好。

類別三、UI 元件與互動

網站不是一張靜態海報，訪客會點、會填、會滑。這一關講的是那些會動、會回應你的小零件：表單、彈窗、選單或表格。就像家裡的開關和門把，它們不起眼，但只要一個不好用，整個體驗就毀了。這幾條範本教你把細節交代清楚，讓每個互動都順手。

21. 做一個含驗證與錯誤提示的表單

什麼時候用： 你需要一個讓使用者填資料的表單，且要防止他們填錯或漏填。

這樣說： > 請幫我做一個〔用途，例如：活動報名〕表單，欄位有：〔姓名、Email、電話、備註〕。要有即時驗證：Email 格式不對、必填欄位空白時，就在該欄位下方顯示紅色提示文字，告訴使用者哪裡錯、該怎麼改。全部填對之前，送出按鈕不能按。送出成功後顯示一句感謝訊息。

💡 好的錯誤提示會說「Email 好像少了 @」，壞的只會說「錯誤」。請 AI 給出具體、友善、教人怎麼改的提示，不要只是冷冰冰的警告訊息。

22. 做一個彈窗 modal

什麼時候用： 你想在不換頁的情況下，跳出一個小視窗讓使用者確認或填資料。

這樣說： > 請幫我做一個彈窗（modal）。當使用者點〔某個按鈕，例如：立即預約〕時，畫面中央跳出一個小視窗，背景變暗且不能點。視窗裡放〔內容，例如：一個簡短的預約表單〕，右上角有關閉的叉叉，點視窗以外的暗色區域或按 **Esc** 鍵也能關閉。關閉時要有輕微的淡出動畫。

💡 「點外面或按 **Esc** 能關」是彈窗的基本禮貌。少了它，使用者會有種被困住的煩躁感，這是新手最常漏掉的一環。

23. 做一個下拉選單

什麼時候用： 選項太多不適合全部攤開，你想收進一個點開才展開的選單。

這樣說： > 請幫我做一個下拉選單（dropdown）。平常只顯示〔目前選項或按鈕文字〕，點一下往下展開所有選項：〔選項清單〕，選好之後選單收起、顯示所選項目。點選單以外的地方要能自動收起。選單展開時要能用鍵盤上下鍵移動、**Enter** 選取，對用鍵盤操作的人也友善。

💡 下拉選單最容易忘記做點外面自動收合。測試時故意點一下旁邊空白處，看它會不會乖乖收起來？

24. 做一個分頁 tab 切換

什麼時候用： 同一塊區域要放好幾組內容，你想用頁籤讓使用者切換著看。

這樣說： > 請幫我做一個分頁切換（tabs）。上方有幾個頁籤：〔頁籤名稱，例如：課程介紹、講師、常見問題〕，點哪個頁籤，下方就只顯示對應的內容，其他隱藏。目前選中的頁籤要有明顯的視覺標示（例如底線或變色）。切換內容時不要整頁重新載入，只換下方區塊。

💡 分頁很適合同一主題的多面向資訊。但如果內容彼此無關，硬塞進分頁反而讓人找不到，那種情況不如直接分區塊攤開。

25. 做一個可展開的手風琴（accordion）

什麼時候用：像常見問題那樣，一長串標題平常收起，點了才展開答案。

這樣說：> 請幫我做一個手風琴（accordion）樣式的常見問題區。每個問題是一行標題，平常收合，點一下往下平順展開答案，再點一下收起。展開時標題旁的箭頭要跟著轉向。可以設定成〔一次只能展開一個／可以同時展開多個，擇一〕。展開收合要有滑順的動畫，不要瞬間跳出。

💡 一次只展開一個比較整齊，適合 FAQ；可同時展開多個，適合使用者需要對照比較的情況。想清楚再告訴 AI。

26. 做一個可排序、可篩選的資料表格

什麼時候用：你有一堆資料要用表格呈現，還想讓使用者自己排序或篩選來找資料。

這樣說：> 請幫我做一個資料表格，欄位有：〔例如：日期、客戶名稱、金額、狀態〕。功能要包含：點欄位標題可以升冪／降冪排序、上方有一個搜尋框可以即時篩選、狀態欄用不同顏色的標籤呈現。資料很多時要能分頁，每頁顯示〔20〕筆。手機上請改成好讀的卡片式排列。

💡 表格在手機上最容易爆版。「手機改成卡片式」這句能救你一命，不然一堆欄位在小螢幕上會擠到沒法看。

27. 加一張把資料視覺化的圖表

什麼時候用：你有一組數字，想用長條圖或折線圖讓人一眼看懂趨勢。

這樣說：> 我有這樣一組資料：〔描述資料，例如：過去六個月每月的報名人數〕。請幫我用〔長條圖／折線圖／圓餅圖，擇一〕呈現。圖表要有清楚的標題、座標軸標籤、以及滑鼠移上去會顯示該筆數字的提示。配色要簡潔專業、手機上也看得清楚。如果我還沒有真實資料，先用假資料讓我看效果。

💡 選圖有訣竅：比較大小用長條圖，看趨勢變化用折線圖，看占比用圓餅圖。選對圖，資料才會替你說話。

28. 做一個檔案上傳功能

什麼時候用：你需要讓使用者上傳圖片或文件，還想把體驗做得順一點。

這樣說：> 請幫我做一個檔案上傳功能。使用者可以點按鈕選檔案，或直接把檔案拖進指定區域。只接受〔檔案類型，例如：jpg、png 圖片〕，單檔不超過〔5MB〕，超過或格式不對就顯示清楚的提示。選好後顯示檔名和一個小預覽（如果是圖片），上傳中顯示進度，成功後給明確的完成提示。

💡 限制類型與大小並給提示，是安全與體驗的雙重防線。少了它，使用者傳了 50MB 的檔案卡住又不知道為什麼，體驗會很差。

29. 補上載入中與空狀態

什麼時候用：資料還在載入、或根本沒有資料時，你不想讓使用者對著空白畫面發呆。

這樣說： > 請幫我的〔某個列表或區塊，例如：訂單列表〕補上兩種狀態的畫面。載入中：顯示一個轉圈或骨架佔位效果，讓使用者知道正在讀取，不要空白一片。沒有資料時（空狀態）：顯示一句友善的說明和一個引導動作，例如「你還沒有任何訂單，去逛逛吧」加一顆按鈕，而不是留一片空白讓人以為壞了。

💡 空白畫面會讓使用者以為當機。載入中和空狀態這兩塊常被新手忽略，補上它們，作品立刻從半成品變成品。

30. 做一個通知 toast 提示

什麼時候用：使用者完成某個動作後，你想給一句短暫、不擋路的成功或失敗提示。

這樣說： > 請幫我做一個通知提示（toast）。當使用者〔完成某動作，例如：儲存成功／刪除失敗〕時，畫面〔右上角或下方〕輕輕滑出一小條訊息，成功用綠色、失敗用紅色，顯示 3 秒後自動淡出消失，也可以手動點叉關掉。它不能擋住使用者正在操作的內容。

💡 toast 適合「講一聲就好」的短訊息。如果是需要使用者做決定的重要事情（例如「確定刪除？」），該用彈窗，不要用會自己消失的 toast。

類別四、功能開發

前面做的是外觀，這一關做的是本事：能登入、能存資料、能搜尋以及能結帳。這些是讓你的作品從好看的樣板，變成真的能用的產品的關鍵。別被這些聽起來很硬的詞嚇到，你要做的一樣是把需求講清楚，剩下的交給 AI。

31. 做登入與註冊功能

什麼時候用：你的產品需要讓使用者建立帳號、登入後看到專屬內容。

這樣說： > 請幫我做使用者註冊與登入功能。註冊要有：Email、密碼、確認密碼，並檢查 Email 格式與密碼強度。登入後要能記住登入狀態，也要有登出和忘記密碼的入口。請用安全的方式處理密碼（不要用明碼存），並告訴我這部分你用了什麼機制、我還需要注意什麼。

💡 帳號密碼牽涉安全，是最不該自己土法煉鋼的部分。明確要求 AI 用安全的既有方案，並請它說明，別讓它自己發明一套。

32. 做基本的增刪改查（CRUD）

什麼時候用：你要管理一批資料，需要能新增、查看、修改、刪除它們。

這樣說：> 請幫我做一組〔資料名稱，例如：客戶資料〕的管理功能，包含四個基本操作：新增一筆、列出全部、修改某一筆與刪除某一筆。每筆資料的欄位有：〔欄位清單〕。刪除前要跳出確認，避免手滑誤刪。修改和新增用同一個表單。做好後給我一個簡單的管理頁面，可以看到全部資料並操作。

💡 CRUD（增刪改查）是幾乎每個軟體的骨幹。學會用一句話交代這四件事，你就掌握了大半個後臺管理系統。

33. 加一個搜尋功能

什麼時候用：資料變多了，你想讓使用者用關鍵字快速找到想要的東西。

這樣說：> 請幫我在〔某個列表，例如：文章列表〕加上搜尋功能。使用者在搜尋框輸入關鍵字，就即時篩選出〔要搜尋的欄位，例如：標題或內文〕包含該關鍵字的結果。搜尋不分大小寫，沒有結果時顯示找不到符合的項目的友善提示。搜尋框旁放一個清除按鈕，可以一鍵還原全部。

💡 先做單一關鍵字即時篩選，就很夠用了。別一開始就想做那種很聰明的模糊搜尋，把基本款做穩再說。

34. 做篩選與排序

什麼時候用：資料有很多類別或屬性，你想讓使用者依條件過濾、依欄位排序。

這樣說：> 請幫我的〔列表名稱〕加上篩選和排序。篩選：讓使用者可以依〔條件，例如：分類、狀態、日期區間〕縮小範圍，可以多個條件同時套用。排序：可以依〔欄位，例如：時間新到舊、價格低到高〕切換。篩選和排序要能疊加使用，並在畫面上清楚顯示目前套用了哪些條件，也能一鍵清除。

💡 顯示目前套用了哪些條件，很重要。不然使用者篩到後來會忘記自己設了什麼，看到奇怪結果還以為壞了。

35. 做分頁或無限捲動

什麼時候用：資料筆數很多，你不想一次全部載入，要分批呈現。

這樣說：> 我的〔列表〕資料很多，一次全載入會太慢。請幫我改成〔分頁：底部有頁碼可以一頁一頁翻／無限捲動：捲到底自動載入下一批，擇一〕。每

批顯示〔20〕筆。載入下一批時要有載入中的提示，全部載完時顯示已經到底了。請告訴我這兩種做法的差別，幫我確認我選的適合我的情況。

💡 內容型網站（文章、貼文）適合無限捲動；需要精準找到第幾筆的後臺管理則適合分頁。選錯會讓使用者找資料找到抓狂。

36. 做一個購物車

什麼時候用：你的產品要賣多樣商品，使用者需要先加入購物車再一起結帳。

這樣說：> 請幫我做購物車功能。使用者可以把商品加入購物車、在購物車裡調整數量或移除、即時看到小計與總金額。購物車內容在重新整理或關掉再打開後還要記得（不能一刷新就清空）。右上角的購物車圖示要顯示目前有幾件商品。先做到「結帳」按鈕為止，實際付款下一步再處理。

💡 刷新後購物車還記得，是新手最容易漏的體驗細節。先跟 AI 講清楚要保留狀態，不然使用者手滑重整就得重挑一輪。

37. 描述金流串接的需求（不碰機密）

什麼時候用：你要讓使用者線上付款，需要 AI 幫你規劃該怎麼接金流。

這樣說：> 我想讓使用者可以線上付款購買〔商品／課程〕，需要支援信用卡。請先不要寫串接程式，而是幫我規劃：常見的金流服務有哪些選擇、串接大概要經過哪些步驟、我需要準備什麼資料、有哪些安全與法規重點要注意。金鑰之類的機密資訊我會自己保管，請你只在需要填入的地方留清楚的〔佔位符〕（placeholder）提醒我。

💡 金流牽涉錢和機密，第一步是搞懂流程，而不是急著貼程式。務必強調機密自己保管、只留佔位符，別把任何金鑰貼進對話。

38. 做寄信通知功能

什麼時候用：某件事發生時（報名成功、密碼重設），你想自動寄一封信通知使用者。

這樣說：> 請幫我做一個「當〔某事件，例如：使用者完成報名〕時，自動寄一封通知信」的功能。信件內容包含：〔要放的資訊，例如：報名項目、時間、聯絡方式〕，語氣親切專業。請規劃我需要用哪種寄信服務、要準備什麼設定，並把寄信的帳號密碼等機密留成〔佔位符〕，由我自己填。先給我一封信件內容的範本讓我確認文案。

💡 先確認信件文案，再處理怎麼寄這件事。畢竟，文案是使用者真正會看到的東西，值得你花時間跟 AI 打磨語氣。

39. 做匯出 CSV 或 PDF

什麼時候用：你想讓使用者把畫面上的資料，下載成 Excel 能開的檔或 PDF。

這樣說：> 請幫我加一個匯出功能，讓使用者把〔某份資料，例如：訂單列表〕下載下來。**CSV：**包含這些欄位〔欄位清單〕，用 Excel 打開不會亂碼（請注意中文編碼）。**PDF：**排版乾淨、有標題和日期。匯出按鈕放在列表上方，如果使用者有套用篩選，就只匯出篩選後的結果。

💡 中文匯出 CSV 最常見的災難，就是用 Excel 開就變亂碼。一定要特別跟 AI 點名這件事，請它處理好編碼。

40. 做多語系切換

什麼時候用：你的網站要同時服務中文與其他語言的使用者。

這樣說：> 請幫我的網站加上語言切換，支援〔繁體中文、英文〕。放一個語言切換的選單，切換後整個介面的文字都跟著變，並記住使用者的選擇。請把所有文字集中管理（不要寫死在各處），方便我之後新增語言或修改翻譯。先幫我把現有頁面的文字整理成一份對照表讓我填翻譯。

💡 多語系的關鍵是文字集中管理。一開始就做對，日後加語言只要補一份對照表；做錯了就得滿場翻找散落各處的文字，很痛苦。

41. 做權限分級

什麼時候用：不同身份的使用者（例如管理員、一般會員）能看到和操作的東西不一樣。

這樣說：> 請幫我做權限分級。我有這幾種角色：〔例如：管理員、一般會員或訪客〕。各自的權限是：〔管理員可以管理所有資料；會員只能看和改自己的；訪客只能瀏覽公開內容〕。請確保沒有權限的人不只是看不到按鈕，就算他直接嘗試操作也會被擋下來。告訴我你怎麼確保這道防線是安全的。

💡 新手常以為把按鈕藏起來，就叫做權限控制。務必強調就算硬闖也要擋住，因為藏起來的功能，懂的人還是有辦法直接呼叫。

類別五、資料與後端

如果前端是餐廳的用餐區，後端就是廚房：客人看不到，但菜是那裡做出來的，食材也存在那裡。這一關談的是資料怎麼存、怎麼取、怎麼保護。名詞聽起來很硬，但你要做的還是老樣子：把我想存什麼、想拿什麼講清楚，AI 會幫你吧廚房蓋好。

42. 設計資料庫 schema

什麼時候用：你要開始存資料了，需要先規劃資料長什麼樣、有哪些欄位？

這樣說： > 我要設計〔專案，例如：一個線上課程平臺〕，需要儲存這些東西：〔例如：學員、課程、報名紀錄〕。請幫我設計資料庫結構（**schema**）：每一種資料該有哪些欄位、各是什麼型態、彼此之間怎麼關聯（例如一個學員可以報名多門課）。請用白話解釋你的設計，並提醒我有沒有漏掉常見會用到的欄位。

💡 資料結構是地基，蓋歪了以後很難改。花時間跟 AI 對這一步，並主動問「我以後想加 X 功能，現在的設計擋不擋得住」。

43. 請 AI 幫我寫資料查詢

什麼時候用： 你想從資料裡撈出特定的結果，但不知道查詢語法怎麼寫？

這樣說： > 我的資料表〔表名與欄位說明〕。我想查出「〔用白話描述你要什麼，例如：這個月報名人數最多的前五門課，含各自的報名人數〕」。請幫我寫出這段查詢，並逐句用白話解釋它在做什麼，讓我看得懂、以後能自己微調。如果有更有效率的寫法也請告訴我。

💡 查詢用白話描述你要的結果最好，別自己猜語法。重點是請 AI 逐句解釋，這樣你會慢慢看懂，而不是永遠只能複製貼上。

44. 建立一個 API 端點

什麼時候用： 你要讓前端畫面能跟後端要資料或送資料，需要一個溝通的接口。

這樣說： > 請幫我建立一個 API 端點（**endpoint**），功能是「〔做什麼，例如：接收前端送來的報名資料，存進資料庫，回傳成功或失敗〕」。請說明這個端點的網址、該用哪種請求方式、前端要送什麼格式的資料、會收到什麼回應。並幫我加上基本的錯誤處理（例如資料不完整時回傳清楚的錯誤訊息）。

💡 把 API 想成點餐窗口：前端遞單子（送資料）進去，後端把菜（回應）端出來。講清楚送什麼、回什麼，這個窗口就打通了。

45. 描述第三方 API 串接需求

什麼時候用： 你想接一個現成的外部服務（地圖、簡訊或 AI 等），但不知道怎麼開始。

這樣說： > 我想在我的專案裡串接〔某個外部服務，例如：把地址轉成地圖座標的服務〕。請先幫我規劃：這類服務常見的選擇有哪些、串接大致要哪些步驟、我需要去申請什麼、免費額度大概到哪、要注意什麼限制。金鑰我會自己申請保管，你在程式裡需要用到的地方請留〔佔位符〕並註明。

💡 串接外部服務前，先問免費額度到哪、超過會怎樣？很多新手不小心接了會計費的服務，用一用才發現帳單來了。

46. 建立環境變數與金鑰觀念

什麼時候用：你手上有 API 金鑰、密碼這類機密，想知道該怎麼安全地放。

這樣說： > 我有一些機密資訊（例如 API 金鑰、資料庫密碼）需要在程式裡使用，但我知道不能直接寫死在程式碼裡。請用零基礎能懂的方式教我：什麼是環境變數（**environment variable**）、為什麼機密要放在這裡、我該怎麼設定、以及要怎麼確保它不會不小心被上傳到公開的地方。請幫我把專案設定成正確的做法。

💡 金鑰寫死在程式碼裡、又推上公開的地方，是新手最貴的一課（可能被盜刷）。務必一開始就把環境變數 + 不上傳機密這件事做對。

47. 加上後端資料驗證

什麼時候用：你想確保存進資料庫的資料是乾淨、合法的，不被亂七八糟的輸入污染。

這樣說： > 前端表單雖然有做檢查，但我知道不能只信任前端。請幫我在後端（收到資料、存進資料庫之前）也加一層驗證：檢查〔必填欄位有沒有缺、格式對不對、數字有沒有在合理範圍、有沒有奇怪的內容〕。驗證不過就擋下來、回傳清楚的錯誤，不要讓髒資料進到資料庫。

💡 請謹記這句話：「永遠不要只相信前端的檢查。」前端驗證是為了體驗，後端驗證才是真正的防線，因為前端的檢查是能被繞過的。

48. 加上快取讓速度變快

什麼時候用：某些資料每次都重算或重抓很慢，你想把結果暫存起來加速。

這樣說： > 我的〔某個功能，例如：首頁的熱門文章排行〕每次載入都要重新計算，有點慢。這份資料其實不用即時、每〔10 分鐘〕更新一次就好。請幫我加上快取（**cache**）：把結果暫存起來，時間內直接用暫存的，時間到了再重算。請用白話解釋你的做法，並提醒我快取可能帶來的副作用（例如資料會有點延遲）。

💡 快取是用一點點資料新鮮度，換很多速度。先確認這份資料晚幾分鐘更新有沒有關係，如果沒關係的才適合快取，會即時變動的資料別亂快取。

49. 建立一個排程任務

什麼時候用：你想讓某件事在固定時間自動執行，不用你手動去按。

這樣說： > 我想讓〔某件事，例如：每天早上 9 點自動寄出前一天的報名統計信〕自動定時執行。請幫我規劃這個排程任務（**scheduled task**）：該用什麼

方式設定、要執行什麼、失敗了怎麼辦（例如要不要重試、要不要通知我）。請用零基礎能懂的方式解釋，並提醒我測試時該怎麼確認它真的有照時間跑。

💡 排程任務最怕你以為它有跑，其實默默失敗了。一定要問 AI 失敗時怎麼通知我，別讓它在背後無聲無息地掛掉。

50. 規劃一次資料遷移

什麼時候用：你要改資料結構、或把資料搬到新地方，怕把現有資料搞壞。

這樣說：> 我需要〔要做的變更，例如：在現有的客戶資料表新增一個來源管道欄位，並把舊資料的這個欄位預設為「其他」〕。這裡面已經有正在使用的真實資料，我很怕改壞。請幫我規劃一個安全的資料遷移（migration）步驟：先做什麼、怎麼備份、怎麼驗證改完後資料沒問題、萬一出錯怎麼還原。

💡 動到真實資料前，第一件事永遠是先備份。就像遊戲打魔王前先存檔，出事了才有回頭路。務必請 AI 把備份和還原步驟寫進去。

類別六、除錯與求救

寫程式遇到出錯，不是你笨，是全世界工程師的日常。真正的差別在於會不會求救：把問題描述清楚的人，AI 三兩下就幫他解決；描述得含糊的人，來回半天還在鬼打牆。這一關教你怎麼把卡關的情況講到 AI 一聽就懂，就像你去看醫生，症狀說得愈準，愈快對症下藥。

51. 把錯誤原文原封不動貼給 AI

什麼時候用：程式跑出一段紅色錯誤訊息，你想請 AI 幫你看。

這樣說：> 我的程式出現了錯誤，以下是完整的錯誤訊息，我原封不動貼給你：>〔把整段錯誤訊息貼上，不要只貼一句，前後都要〕> 我當時正在做的事情是：〔例如：點下「送出」按鈕的時候〕。請幫我看這個錯誤是什麼意思、可能是什麼原因造成的、以及該怎麼修。

💡 別只複製一句紅字，也別自己轉述。錯誤訊息像病歷，完整貼上最有用。很多關鍵線索都藏在你覺得看起來不重要的那幾行裡。

52. 用「預期 vs 實際」描述問題

什麼時候用：程式沒報錯，但行為不對，跟你想的不一樣。

這樣說：> 我遇到一個問題，程式沒有跳錯誤，但行為不對。> 我預期：〔點下按鈕後，應該跳出成功訊息〕。> 實際發生：〔點下按鈕後，什麼都沒發生〕。> 重現步驟：〔1. 打開頁面 2. 填好表單 3. 點送出〕。> 請幫我分析為什麼會這樣，該從哪裡開始查。

💡「預期 vs 實際 vs 重現步驟」是求救的黃金公式。沒報錯的問題最難講，用這三格把它框起來，AI 就抓得到方向。

53. 請 AI 幫我重現問題

什麼時候用：問題只是偶爾發生、或你不確定什麼情況下會出錯。

這樣說：> 我這個功能〔描述功能〕有時候會出錯、有時候正常，我抓不到規律。請幫我想想：在什麼情況下最可能出這個問題？請列出幾個可能觸發問題的情境，並告訴我每一個該怎麼實際操作去測試，好讓我穩定地重現這個 bug，才能對症下藥。

💡 抓不到規律的 bug 最惱人。與其瞎試，不如請 AI 幫你列出可疑情境，然後逐一測試。只要能穩定重現，問題就解決一半了。

54. 用二分法縮小問題範圍

什麼時候用：一大段程式或一整個流程出錯，你不知道是哪一段的問題。

這樣說：> 我這個〔功能或流程〕出錯了，但它牽涉的環節很多，我不知道問題出在哪一段。請教我用二分法縮小範圍：怎麼把整個流程切成前後兩半、分別怎麼確認哪一半是好的、哪一半有問題，然後再把有問題的那半繼續切。請一步一步帶我找出真正出錯的那個點。

💡 二分法就像玩猜數字：每次砍一半，很快就逼近答案。與其從頭到尾一行行看，不如先確認問題在前半還後半，效率高得多。

55. 處理改了 A 卻壞了 B 的回歸

什麼時候用：你明明是去修某個地方，結果別的、本來好好的功能反而壞了。

這樣說：> 我剛才為了〔目的，例如：修正報名表單的驗證〕改了〔改動內容〕，結果原本正常的〔另一個功能，例如：登入功能〕反而壞了。我改動前它是好的。請幫我分析：我的改動為什麼會連累到這個看似無關的功能？它們之間可能有什麼隱藏的關聯？該怎麼修才不會又弄壞別的地方。

💡 這種按下葫蘆浮起瓢的狀況，叫回歸（regression）。關鍵線索是改之前是好的，一定要講給 AI 聽，它才會往你剛才的改動這個方向查。

56. 應對 AI 說修好了但其實沒好

什麼時候用：AI 信誓旦旦說改好了，你一測還是一樣壞。

這樣說：> 你上一步說已經修好了，但我實際測試，問題完全沒有改善，狀況跟之前一模一樣：〔描述現在的實際情形，或貼上一樣的錯誤〕。請不要再假

設上次的修法是對的。我們重新開始：請先跟我確認你認為問題的根本原因是什麼，我們對齊了原因，再決定怎麼修。

💡 AI 會有過度自信的時候。這時別讓它在錯的方向上繼續修正，直接喊停、要求它重講一次你認為的根本原因，逼它回到診斷。

57. 卡住了，請 AI 換一個方向

什麼時候用：同一個方法試了好幾次都修不好，你陷在死胡同裡。

這樣說： > 這個問題我們已經用〔目前的方法〕試了好幾次都沒成功：〔簡述試過什麼、結果如何〕。我不想再往同一個方向鑽了。請你先退一步，用全新的角度重新思考這個問題，提出 2 到 3 個跟剛才完全不同的解法或思路讓我選，並說明各自的優缺點。

💡 一個方法試三次沒好，就別再試第四次了。明確告訴 AI 換個完全不同的方向，並附上你試過什麼，避免它又繞回原路。

58. 請 AI 白話翻譯這個錯誤在講什麼

什麼時候用：你看到一串英文錯誤或術語，完全看不懂它在說什麼。

這樣說： > 我看到這個訊息完全看不懂：〔貼上錯誤訊息或那段看不懂的術語〕。請你當我的翻譯，用講給一個完全不懂程式的朋友聽的方式，白話解釋這句話到底在講什麼、它在抱怨什麼、大方向上通常跟什麼有關。我先求聽懂它在說什麼，還不急著修。

💡 看懂錯誤訊息在講什麼，你就從怕它變成懂它。多用這招當翻譯機，久而久之你會發現常見的錯誤就那幾種。

59. 請 AI 加上防護與錯誤處理

什麼時候用：程式在遇到意外狀況時就整個崩掉，你想讓它更耐操、不要動不動就掛。

這樣說： > 我的〔某個功能〕遇到意外狀況（例如網路斷線、使用者輸入怪東西、外部服務沒回應）時就整個當掉。請幫我加上防護和錯誤處理：讓程式在出狀況時不要崩潰，而是優雅地擋下來、給使用者一句友善的提示、必要時記錄下來方便我查。請告訴我你補了哪些防護、還有哪些狀況值得防。

💡 好的程式不是不會出錯，而是出錯時不會爆炸。主動請 AI 想還有哪些狀況值得提防，它常能點出你沒想到的意外。

60. 請 AI 列出所有可能原因再逐一排除

什麼時候用：問題原因不明，你想有系統地排查，而不是東試一個西試一個。

這樣說： > 我遇到這個問題：〔描述問題，附上預期 vs 實際〕。我不想亂猜亂試。請先幫我列出所有可能造成這個問題的原因，從最常見、最可能的排到最少見的。然後針對每一個原因，告訴我該怎麼快速確認是不是它？我們一個一個排除，找到真正的兇手。

💡 這是專業工程師的除錯心法：先列清單、依可能性排序、逐一驗證。比起靈光一閃亂試，這種系統性排查的方式，慢慢來反而最快。

類別七、讀懂與學習程式

AI 幫你寫出來的程式，你不見得看得懂，這很正常。但如果你連它做了什麼都不知道，就等於把方向盤完全交出去，出事只能乾瞪眼。這一類的提示詞不是要你變成工程師，而是讓 AI 一邊做、一邊當你的家教，把每一段程式翻成你聽得懂的話，你才有底氣做決定。

61. 白話翻譯這段程式

什麼時候用： AI 貼了一段你完全看不懂的程式，想先知道它在幹嘛再決定要不要用。

這樣說： > 請用完全不懂程式的人也能聽懂的白話，解釋下面這段程式在做什麼。不要用專業術語，如果一定要用，請馬上用生活例子解釋。最後用一句話總結它的目的。 > 〔貼上程式碼〕

💡 記得加一句「假設你在跟我阿嬤解釋」，AI 會自動把難度再降一階。

62. 逐行加上中文註解

什麼時候用： 想把一段程式留著日後複習，需要每一行都有說明。

這樣說： > 請幫下面這段程式每一行加上繁體中文註解，說明那一行在做什麼、為什麼需要它。註解要簡短，放在該行的上方或後方，不要改動任何程式邏輯。 > 〔貼上程式碼〕

💡 明講不要改動程式邏輯很重要，否則 AI 常常順手幫你改寫，反而動到本來好好的東西。

63. 把邏輯畫成流程

什麼時候用： 程式有很多判斷與分支，你想看懂它的決策路線。

這樣說： > 請把下面這段程式的執行邏輯，畫成一張文字版流程圖，用箭頭和縮排表示先後順序與判斷分支（例如：「如果 X 成立就往這走，否則往那走」）。讓我不用讀程式也能看懂它的流程。 > 〔貼上程式碼〕

💡 看不懂文字流程圖的話，補一句「請改用條列步驟，像食譜一樣一步一步寫」。

64. 幫變數與函式取好名字

什麼時候用：程式裡出現一堆 `a`、`b`、`data1`、`temp`，你根本認不出誰是誰。

這樣說： > 下面這段程式的變數和函式名稱太籠統，我看不懂。請幫我改成一看就懂用途的名稱（例如把 `data` 改成 `customerList`），並附一張新舊名稱對照表說明每個名字代表什麼。改名時請一併更新所有用到它的地方。 > [貼上程式碼]

💡 好名字是最便宜的註解。這一步做完，很多程式你不必看註解也能猜個八九不離十。

65. 給我兩種寫法並比較

什麼時候用：AI 給了一種做法，你想知道有沒有更簡單的路，順便學判斷。

這樣說： > 針對「[要達成的功能]」這個需求，請給我兩種不同的寫法。用表格比較它們的優缺點（好懂程度、效能、日後好不好維護），最後以零基礎新手該選哪一個的角度給我一句建議。 > [貼上程式碼或描述需求]

💡 這招能幫你養出判斷力：看多了比較，你會慢慢知道什麼叫乾淨的程式？

66. 這段在做什麼，有沒有風險

什麼時候用：從網路上或別人那裡拿到一段程式，貼進專案前想先確認安全。

這樣說： > 我打算把下面這段程式加進我的專案。請先用白話說明它會做什麼，再幫我檢查有沒有潛在風險（例如會不會刪掉資料、會不會把我的資訊送到外部、有沒有看起來可疑的地方）。如果有疑慮，請明確標出是哪一行。 > [貼上程式碼]

💡 別人給的程式不要照單全收。這一問，等於請 AI 幫你先聞一下有沒有壞掉的味道。

67. 把術語翻成人話

什麼時候用：AI 的回覆裡塞滿看不懂的名詞，你被 `API`、非同步或快取這些字擋在門外。

這樣說： > 你剛剛的說明裡出現這些我不懂的詞：[列出你看不懂的詞]。請幫每一個詞用一句生活化的比喻解釋，就像跟朋友聊天那樣，不要再用其他專業術語來解釋專業術語。

💡 把不懂的詞隨手記進一個檔案，累積成你的個人詞彙表，下次遇到就不慌了。

68. 邊做邊教我

什麼時候用：你想在完成專案的同時，順便把觀念學起來，而不是做完什麼都沒學到。

這樣說：> 接下來我們一起做「〔專案名稱〕」。請每完成一個步驟，就用三句話內告訴我：這一步在做什麼、為什麼要這樣做、以及我學到了哪個可以重複使用的觀念。我看懂了會說「繼續」，你再往下走。

💡 這是把 AI 從代工變成教練的關鍵句。速度會慢一點，但一個月後你會差很多。

類別八、重構與優化

程式能跑，不代表它整齊。就像房間住久了，東西愈堆愈亂，程式改著改著也會變成一團，連 AI 自己都會在裡面迷路。重構就是不改變功能、只把裡面收乾淨。你不必自己動手，但要會開口請 AI 幫你整理，而且要盯著它別在整理時把東西弄壞。

69. 把太長的檔案拆開

什麼時候用：某個檔案愈長愈大，AI 每次修改都要花很久，還常常改錯地方。

這樣說：> 這個檔案太長了，AI 每次改都很吃力。請幫我依照功能把它拆成幾個小檔案（例如畫面、資料處理、共用工具各自分開），並說明每個新檔案負責什麼。拆完之後功能必須和原本完全一樣，不能有任何行為改變。>〔貼上程式碼或說明檔案位置〕

💡 拆檔前先請它只講計畫、先不要動手，你確認方向對了再讓它執行，見範本 76。

70. 提升可讀性但不改行為

什麼時候用：程式可以跑但很亂，你想讓它變乾淨，又怕一改就壞。

這樣說：> 請幫我重構下面這段程式，目標是讓它更好讀、更好維護：包括改善命名、整理排版、把重複的邏輯抽出來。硬性要求：功能和輸出結果必須和原本一模一樣，這是純整理，不是改功能。改完請告訴我你動了哪些地方、為什麼。>〔貼上程式碼〕

💡 「純整理，不改功能」這八個字要反覆強調。重構最怕的就是 AI 順手多做，把好的也一起換掉。

71. 找出重複的程式碼

什麼時候用：你覺得同樣的東西好像在很多地方各寫了一遍，改一個要改十個。

這樣說：> 請幫我掃過這個專案，找出重複或非常相似的程式碼段落，列成清單（在哪些地方、重複了什麼）。針對每一組重複，建議該怎麼合併成一個共用的部分，讓我以後只要改一個地方就好。先給我清單和建議，我確認後你再動手。>〔貼上程式碼或說明範圍〕

💡 有個好記的原則叫「同一件事只寫一次」。重複愈少，日後改起來愈省力，出錯機會也愈低。

72. 改善效能與載入速度

什麼時候用：網頁開很慢、按鈕按下去要等很久，你想讓它變順。

這樣說：> 我的〔頁面或功能〕載入或反應很慢。請幫我找出可能拖慢速度的原因，列出前三個最值得優化的地方，並依效果大、風險小和好實作來排序。先解釋每一項為什麼會慢、改了會怎樣，我選定後你再逐一實作。>〔貼上程式碼或描述症狀〕

💡 先問最值得改的前三個，避免 AI 一次改一大片。慢的原因通常集中在少數幾處，抓大放小最划算。

73. 統一命名風格

什麼時候用：專案是東拼西湊做起來的，命名一下英文一下中文、一下大寫一下小寫，很亂。

這樣說：> 這個專案的命名風格不一致（有的用中文、有的用英文、大小寫也很亂）。請先幫我訂一套簡單一致的命名規則並說明，再依這套規則把整個專案的名稱統一。改動請一次到位，並確認所有相關的地方都同步更新，不要漏掉。>〔貼上程式碼或說明範圍〕

💡 統一風格看起來只是美觀問題，其實能大幅降低你和 AI 之後的溝通成本，因為大家對東西叫什麼有共識了。

74. 移除沒用到的東西

什麼時候用：專案裡堆了一堆試過又放棄的程式、匯入卻沒用的東西，愈看愈亂。

這樣說：> 請幫我找出這個專案裡沒有被使用到的東西：沒被呼叫的函式、匯入了卻沒用的套件、永遠不會執行到的程式。列成清單讓我確認，並標註你判斷沒用到的理由。我逐一確認後，再請你刪除。>〔貼上程式碼或說明範圍〕

💡 一定要先看清單再刪。有些程式看似沒用，其實是別處偷偷在用，讓 AI 給理由能幫你攔下誤刪。

75. 加上清楚的結構與分層

什麼時候用：專案長大了，所有東西擠在一起，你想把它整理成有條理的資料夾結構。

這樣說：> 我的專案愈做愈大，檔案都擠在一起很難找。請建議一套適合新手、清楚好懂的資料夾結構（例如：畫面、資料、共用工具或設定各自分層），畫給我看，並說明每一層放什麼。我同意後，請你幫我搬移檔案並更新所有連動的路徑。

💡 好的結構就像圖書館的分類，東西各就各位，你和 AI 找起來都快。搬移後記得請 AI 確認網站還能正常開。

76. 重構前先說明計畫

什麼時候用：要做比較大的整理，你怕 AI 一頭栽下去改一大片，收不回來。

這樣說：> 我想重構〔要處理的部分〕，但在你動手之前，請先只用文字告訴我你的計畫：你打算分幾個步驟、每步會動到哪些檔案、可能有什麼風險。先不要寫任何程式。我看過計畫、回覆「開始」之後，你再一步一步做，每做完一步停下來讓我確認。

💡 這是重構最重要的一句護身符。先看地圖再上路，你隨時能喊停，不會等到全部改完才發現走錯方向。

類別九、測試與品質把關

「奇怪！在我電腦上看，明明好好的？」是新手最常踩的坑。你自己點一遍覺得沒問題，換個手機、換個奇怪的操作順序就爆了。測試與品質把關，就是在東西交到別人手上之前，先請 AI 扮演那個最愛亂按、最會挑毛病的使用者，把問題在家裡先抓出來。

77. 幫我加基本測試

什麼時候用：功能做好了，你想有個自動檢查，之後改東西才不會不小心弄壞它。

這樣說：> 請幫「〔功能名稱〕」加上基本的自動測試，涵蓋最重要的幾個使用情境（正常情況、常見的錯誤情況）。用白話在每個測試上方註解它在驗證什麼。也請告訴我怎麼執行這些測試，以及看到什麼結果代表通過。>〔貼上程式碼或描述功能〕

💡 有了測試，就像遊戲有了存檔點。以後每次改完跑一遍，綠燈就安心，紅燈就知道剛剛動壞了什麼。

78. 列出邊界與極端情況

什麼時候用：功能一般狀況能動，但你擔心遇到怪怪的輸入就出包。

這樣說：> 針對「〔功能名稱〕」，請幫我列出所有可能出問題的邊界與極端情況（例如：欄位空白、輸入超長、負數、貼上表情符號、網路斷線、同時按很多下）。做成清單，每一項說明「如果沒處理會發生什麼」，並依嚴重程度排序。

💡 使用者永遠會做出你想不到的事。先把這張清單攤開，你就知道哪些洞非補不可、哪些可以先放著。

79. 驗證使用者輸入

什麼時候用：頁面上有表單或輸入框，你想確保使用者亂填時不會出事。

這樣說：> 我的〔表單或輸入框〕需要檢查使用者的輸入。請幫我加上驗證：說明每個欄位該接受什麼、不該接受什麼，遇到不合規定時要顯示清楚友善的中文提示（而不是讓程式壞掉）。請以新手也看得懂的錯誤訊息為原則設計提示文字。> 〔貼上程式碼或描述表單〕

💡 好的驗證不只是擋錯，還要好好告訴使用者哪裡填錯、該怎麼改，這決定了別人用起來是順手還是想砸電腦。

80. 檢查無障礙

什麼時候用：你想確認視障者、用鍵盤操作或色弱的使用者也能正常使用你的網頁。

這樣說：> 請幫我檢查這個頁面的無障礙（**Accessibility**）問題，站在視障、只能用鍵盤操作以及色弱使用者的角度。列出具體問題（例如圖片缺少替代文字、顏色對比不足、無法用鍵盤操作），每一項說明影響到誰、該怎麼修，並依重要性排序。> 〔貼上程式碼或網址〕

💡 無障礙不是額外功德，它讓更多人用得了你的東西，順帶對搜尋引擎也友善，一舉兩得。

81. 基本資安檢查

什麼時候用：東西要上線給別人用了，你想先做一次基礎的安全體檢。

這樣說：> 請幫我的專案做一次基礎資安檢查，重點放在新手最容易犯的錯：有沒有把密碼或金鑰（**API Key**）直接寫死在程式裡、有沒有把不該公開的資

訊送到前端、使用者輸入有沒有可能被拿來搞破壞。列出問題與修法，用白話說明每個風險最壞會怎樣？>〔貼上程式碼或說明範圍〕

💡 金鑰外洩是零基礎最常見的意外。請 AI 特別確認金鑰是放在安全的設定裡，而不是躺在任何會被別人看到的檔案中。

82. 跨瀏覽器與手機檢查

什麼時候用：你只在自己的電腦和瀏覽器測過，擔心別人用手機或別的瀏覽器會跑版。

這樣說：> 我只在自己的電腦上測過這個頁面。請幫我檢查它在手機、平板、以及不同瀏覽器（Chrome、Safari）上可能出現的問題（跑版、按鈕太小、文字被切掉、橫向捲動）。列出風險點與建議修法，並告訴我可以怎麼自己快速驗證。>〔貼上程式碼或網址〕

💡 一個簡單的自測法：在電腦瀏覽器把視窗拉到很窄，模擬手機畫面。很多跑版問題當場就現形。

83. 上線前檢查清單

什麼時候用：東西快做完了，你想要一張發布前該確認什麼的清單，避免漏東漏西。

這樣說：> 我的「〔專案名稱〕」快要上線了。請幫我做一張新手也能照著走的上線前檢查清單，涵蓋功能、手機顯示、錯誤處理、資安、速度、以及金鑰設定等面向。每一項用一句白話寫成可打勾的動作，並標出哪些是「沒做到絕對不能上線」的紅線項目。

💡 把這張清單存起來，每個專案上線前都跑一遍。清單化的好處是不靠記性，你不會因為一時興奮就忘了關鍵一步。

84. 請它扮演找碴的 QA

什麼時候用：你想在上線前，讓 AI 用最挑剔的眼光把整個東西攻擊一遍。

這樣說：> 現在請你扮演一位最龜毛、最愛挑毛病的品質測試人員（QA），目標是把我的「〔功能或頁面〕」玩壞。請列出你會怎麼刁難它：亂填、亂點、用意想不到的順序操作、丟奇怪的資料進去。每一種攻擊寫出「你預期會發生什麼問題」，讓我逐一去防。

💡 換個角色，AI 的視角就變了。當它從幫你做切換成來拆你的臺，就會挖出你自己怎麼也想不到的破綻。

85. 先自我檢查再回報

什麼時候用：你希望 AI 交東西給你之前，先自己認真檢查一遍，減少你來回抓錯。

這樣說：> 完成這個任務後，先不要急著說做好了。請你自己扮演審查者，把成果重新檢查一遍：有沒有沒處理到的情況、有沒有明顯的錯、有沒有偏離我的需求。把你自己抓到的問題和修正一起回報給我，如果都沒問題，請說明你檢查了哪些項目。

💡 這一句能大幅提升 AI 交付的品質。你等於在它按下送出前，多安插了一道它自己的品管關卡。

類別十、部署與上線

做好的東西還躺在你電腦裡，別人是看不到的。部署（Deploy）就是把它搬到網路上，讓全世界用網址就能打開。這一步對新手最有魔法感，也最容易卡關。訣竅是不要自己硬闖，把 AI 當成牽著你手、一步一步走完流程的導遊。

86. 帶我一步步部署

什麼時候用：你第一次要把做好的東西放上網，完全不知道從哪開始。

這樣說：> 我完全沒有部署過任何東西，想把「〔專案名稱〕」放到網路上讓別人用網址打開。請當我的導遊，一步一步帶我做，每次只給我一個步驟，用白話說明這一步要做什麼、我會看到什麼畫面、成功的樣子長怎樣？我完成並回報後，你再給下一步。

💡 關鍵是每次只給一步。道理很簡單，如果一次給你十個步驟，你很可能會在第三步就迷路。所以慢慢來，一步一確認，反而最快。

87. 選擇部署平臺

什麼時候用：你聽過一堆部署服務的名字，但不知道自己的專案該用哪個。

這樣說：> 我做的是「〔簡述你的專案類型，例如純網頁、有登入功能、有資料庫〕」，預算希望是「免費／每月可接受金額」。請推薦兩到三個適合新手的部署平臺，用表格比較它們的難易度、免費額度以及適合的情境，最後給我一句「以你的狀況我推薦哪一個」的結論。

💡 新手起步先求免費、好上手，別一開始就追求功能最強。等你真的需要更多，再換也不遲。

88. 設定環境變數

什麼時候用：程式需要用到密碼或金鑰，你知道不能寫死在程式裡，但不會設定。

這樣說： > 我的專案有一些不能公開的資訊（例如 API 金鑰、密碼），聽說要放在環境變數裡，而不是寫在程式碼中。請先用白話解釋環境變數是什麼、為什麼比較安全，再一步一步教我在〔你用的部署平臺〕上設定它，並告訴我程式裡要怎麼改成去讀取這些變數？

💡 環境變數就像保險箱：程式知道去保險箱拿鑰匙，但鑰匙本身不會被寫進到處流傳的程式碼裡。這是資安的基本功。

89. 綁自訂網域

什麼時候用：你的網站現在是一串又長又醜的網址，你想換成自己的品牌網域。

這樣說： > 我想把我的網站從現在的預設網址，換成我自己的網域〔你的網域，例如 `vista.tw`〕。我的網域是在〔購買網域的地方〕買的，部署在〔你的部署平臺〕。請一步一步教我怎麼設定，包含我需要去哪裡改什麼、大概要等多久生效、以及怎麼確認綁定成功了。

💡 網域設定改完通常不會立刻生效，可能要等幾分鐘到幾小時，這叫生效等待。別以為沒成功就一直亂改，先等一下再驗。

90. 部署失敗時怎麼問

什麼時候用：你按下部署，結果跳出一堆紅字錯誤，整個人慌了。

這樣說： > 我在部署時失敗了，跳出下面這段錯誤訊息。請先用白話告訴我這大概是什麼問題、嚴不嚴重，再列出最可能的兩到三個原因，並從最可能的開始，一次給我一個解法讓我試。我試完會回報結果，你再決定下一步。 > 〔完整貼上錯誤訊息〕

💡 部署錯誤訊息一定要整段完整貼給 AI，別自己擷取一半。那串看似天書的紅字，往往第一行就藏著真正的病因。

91. 確認線上版真的更新了

什麼時候用：你改了東西又重新部署，但打開網頁好像還是舊的，不確定到底更新了沒。

這樣說： > 我重新部署後，不確定線上的版本有沒有真的更新（畫面看起來還是舊的）。請教我幾個確認方法：怎麼判斷部署是否真的成功、瀏覽器的快取

(Cache) 可能怎麼騙我、以及怎麼強制看到最新版。請用新手能操作的步驟說明。

💡 常見的假象是瀏覽器把舊畫面存起來給你看。用無痕視窗打開，或強制重新整理，通常就能看到真正的最新版。

92. 上線後基本監控

什麼時候用：東西上線了，你想知道萬一壞掉或沒人用，怎麼第一時間發現？

這樣說： > 我的網站已經上線。我不是工程師，但想做最基本的監控，避免它壞掉了我卻不知道。請推薦一到兩個適合新手、最好有免費方案的簡單做法，能在網站掛掉時通知我。用白話說明怎麼設定、以及它會怎麼提醒我。

💡 對新手來說，網站掛了會自動寄信通知我，這一件事就夠用了。先求有、再求好，別一開始就想搭一套複雜的儀表板。

93. 把部署步驟寫成我看得懂的清單

什麼時候用：這次好不容易部署成功了，你想把流程記下來，下次自己也能重來一遍。

這樣說： > 我們剛剛成功把專案部署上線了。請把整個過程整理成一張我看得懂的步驟清單，用白話寫，每一步包含做什麼、在哪裡做、成功長怎樣？特別標出這次卡關過、下次要小心的地方。我要把它存起來，當作以後每次上線的說明書。

💡 部署流程很容易做完就忘。趁記憶新鮮請 AI 幫你寫成說明書，下次就不必再從零摸索，也不用每次都重新問。

類別十一、內容行銷與 SEO

網站做好了、也上線了，接下來的問題是：別人怎麼找到它？分享到社群時，為什麼有的連結會跳出漂亮的預覽圖，有的只有一串冷冰冰的網址？這一類提示詞不談通用文案，而是教你請 AI 幫網站做好被看見的技術設定，讓搜尋引擎和社群平臺都願意把你推出去。

94. 寫 SEO 友善的標題與描述

什麼時候用：你想讓網頁在 Google 搜尋結果裡，標題和說明文字看起來吸引人又切題。

這樣說： > 我的頁面是關於「〔頁面主題與目標客群〕」。請幫我寫出適合放進網頁的 SEO 標題 (title) 和描述 (meta description)，要包含大家會搜尋的關鍵字、控制在搜尋結果不會被截斷的長度、並且讓人有想點進去的衝動。給我兩組選擇，並告訴我這些文字實際上要放進程式的哪個位置。

💡 標題和描述是你在 Google 上的門面。同一個排名，寫得誘人的那個就是比較多人點，這是免費卻常被忽略的關鍵。

95. 加上分享預覽（Open Graph）

什麼時候用：你把網址貼到 LINE 或臉書，卻只顯示乾巴巴的連結，沒有圖也沒有標題。

這樣說：> 我希望我的頁面被分享到臉書、LINE 或 Threads 時，能顯示漂亮的預覽（標題、說明或縮圖）。請教我加上 Open Graph 標籤：告訴我需要哪幾個標籤、每個代表什麼、實際的程式碼長怎樣、該放在檔案的哪裡，以及縮圖建議做多大尺寸。

💡 設定完可以用臉書或 LINE 的官方分享偵錯工具驗證。有時預覽會被舊資料卡住，用偵錯工具重新抓一次通常就會更新。

96. 加上結構化資料

什麼時候用：你想讓 Google 更看得懂你的頁面內容（例如這是文章、商品還是活動），有機會拿到更好的呈現。

這樣說：> 我的頁面是「〔頁面類型，例如一篇文章／一個商品／一場活動〕」。請幫我加上對應的結構化資料（Structured Data），讓 Google 更容易理解頁面內容。用白話說明它是什麼、對我有什麼好處，給我可以直接用的程式碼，並告訴我怎麼驗證它有沒有寫對。

💡 結構化資料就像幫你的頁面貼上標籤紙，主動告訴搜尋引擎我是什麼。寫完務必用 Google 的測試工具驗一次，格式錯了它會直接忽略。

97. 寫多語版本

什麼時候用：你想把網站做成中、英文（或加日文）版本，讓不同語言的訪客都能看懂。

這樣說：> 我想把「〔頁面名稱〕」做成中文和〔英文／日文〕的多語版本。請先用白話說明多語網站有哪幾種常見做法、各自的優缺點，再以新手好維護為原則推薦一種。然後教我怎麼設定，讓搜尋引擎知道這些是同一頁的不同語言版本，不會被當成重複內容。

💡 多語不是只把文字翻過去就好，還要讓搜尋引擎知道版本之間的對應關係。這一步做對，你的英文頁才有機會被英文使用者搜到。

98. 搞懂 sitemap 與 robots

什麼時候用：你聽過網站地圖和爬蟲設定，但不知道自己的小網站到底需不需要、該怎麼弄？

這樣說： > 請用白話解釋 **sitemap**（網站地圖）和 **robots.txt** 這兩個東西是什麼、分別在幫我做什麼。以我這種「〔簡述你的網站規模，例如只有幾頁的個人網站〕」來說，需不需要設定？如果需要，請一步一步教我怎麼產生和放置，並告訴我怎麼確認 **Google** 有讀到。

💡 網站地圖等於把你所有頁面列一張清單交給 **Google**，省得它自己慢慢找。就算是小網站，設一下也沒壞處。

99. 把賣點寫成頁面文案

什麼時候用： 你有一個產品或服務頁，但只列了功能，缺一打動人、又對搜尋友善的文案。

這樣說： > 這是我的產品頁，目前只列了功能：〔貼上現有內容〕。目標客群是〔你的客群〕，他們最在意〔客群的痛點或渴望〕。請幫我把它改寫成能打動人的頁面文案，聚焦在這對客人有什麼好處，並自然帶入他們會搜尋的關鍵字，不要塞得很生硬。給我可直接替換的版本。

💡 好文案講的是你會得到什麼，不是這個功能叫什麼？關鍵字要像鹽巴一樣自然融進句子，撒太多反而會被搜尋引擎當作作弊。

100. 無障礙與 SEO 一起顧

什麼時候用： 你想一次把更多人用得了和更容易被搜到這兩件事一起做好。

這樣說： > 請幫我檢查這個頁面，同時從無障礙和 **SEO** 兩個角度找問題，因為我聽說它們常常互相加分。重點看：圖片的替代文字、標題層級的結構、連結文字有沒有意義。列出問題清單，每一項說明它同時影響無障礙和 **SEO** 的哪個部分，並給我修法。 > 〔貼上程式碼或網址〕

💡 圖片替代文字是最典型的一石二鳥：視障者靠它讀圖，搜尋引擎也靠它認圖。把該寫的都寫好，兩邊同時受惠。

類別十二、AI 協作元技巧

前面十一類教你請 **AI** 做各種事，這一類更上一層：教你怎麼指揮 **AI** 這件事本身。同樣的任務，會下指令的人拿到的成果，往往跟不會下的人差了好幾個檔次。這些是進階玩家的內功心法：設定角色、劃定範圍、控制節奏、要它誠實。學會這些，**AI** 才真正變成聽你調度的軍團，而不是各自亂跑的散兵。

101. 給 **AI** 設定角色人格

什麼時候用： 你希望 **AI** 用某個專業角色的視角回答，而不是泛泛的通用答案。

這樣說： > 從現在開始，請你扮演一位「〔角色，例如有十年經驗、很重視新手體驗的資深工程師〕」。用這個角色的專業和口吻跟我對話：〔角色該有的態度，例如務實、不賣弄術語、會主動提醒風險〕。在我說跳出角色之前，都請維持這個設定。

💡 設定角色不只是好玩，它會實質改變 AI 的判斷重點。一個重視新手的角色，自然會少用術語、多給提醒。

102. 限制它只能改某個範圍

什麼時候用： 你只想讓 AI 動某一個檔案或某一段，最怕它雞婆去改別的地方。

這樣說： > 接下來的修改，請你只能動〔指定的檔案或區塊〕，其他任何檔案和程式都不准碰。如果你認為需要改到範圍以外的地方，先停下來告訴我原因、問過我同意，不要自己動手。開始前，請先複述一次你被允許改動的範圍，讓我確認。

💡 「請先複述你被允許改的範圍」是很強的一招。讓 AI 自己講一遍界線，它接下來越界的機率會明顯下降。

103. 要它一步一步來

什麼時候用： 任務有點複雜，你怕 AI 一口氣全做完，中間錯了你也來不及攔。

這樣說： > 這個任務請你分步驟進行，一次只做一步。每做完一步就停下來，用一句話告訴我這步做了什麼，然後等我說「繼續」你再往下。不要一次把全部做完。如果你覺得步驟可以拆，就拆得愈細愈好，我寧可多確認幾次。

💡 一步一停看似慢，其實省時間。錯誤在第一步就被你抓到，總比全部做完才發現要從頭來過划算得多。

104. 要它先問清楚再動手

什麼時候用： 你的需求還沒講得很完整，不想讓 AI 自己腦補一堆假設就開工。

這樣說： > 在你開始動手之前，如果我的需求有任何不清楚、或有多種做法可選的地方，請先把問題一次列出來問我，不要自己假設。等我回答完、你確定完全理解了，再開始做。如果你其實已經很清楚、沒有問題要問，也直接告訴我「沒有疑問，我要開始了」。

💡 這一句能擋掉大量做完之後，才發現不是我要的。與其事後重做，不如讓 AI 在開工前把不確定的地方一次問齊。

105. 指定輸出格式

什麼時候用：你希望 AI 的回覆有固定格式，方便你直接拿去用或存檔。

這樣說：> 請用以下固定格式回覆我：〔描述你要的格式，例如先一句結論、再用表格列出項目、最後給三個可執行的下一步〕。不論內容多寡，都請維持這個格式。如果某個欄位沒有內容，就寫「無」，不要自己改變結構。

💡 事先框好格式，等於幫 AI 鋪好軌道。它回得整齊、你讀得省力，還能一份份接起來變成你的專屬模板。

106. 控制回覆長度

什麼時候用：AI 每次都回一大篇，你其實只想要一個乾脆的答案。

這樣說：> 接下來請你回答簡短、直接講重點，每個回覆盡量控制在〔例如三句話／五行〕以內，先給結論再說原因。如果某個問題真的需要展開，先給我摘要，並問我要不要看完整說明？如果我說要，你再展開。

💡 太長的回覆會稀釋重點，你也讀得很累。先要摘要、需要再展開，主動權在你手上，節奏你來定。

107. 要它每步報告在做什麼

什麼時候用：AI 一動就是一長串操作，你想全程知道它到底在幹嘛，不要黑箱。

這樣說：> 在整個過程中，請你邊做邊用白話向我回報：現在正在做哪一步、為什麼要做、以及做完後結果是什麼。把它想成你在向一個看不到你螢幕的新手同事實況轉播。遇到任何你不確定或有風險的決定，先停下來說明，讓我決定。

💡 讓 AI 開啟實況轉播模式，你就能在它跑偏的當下即時喊停，而不是等到最後看著一團亂才後悔沒早點問。

108. 防止它亂加功能

什麼時候用：你只要求做 A，AI 卻自作主張連 B、C、D 一起做了，把簡單的事弄複雜。

這樣說：> 請只完成我這次明確要求的事，不要自己多加任何我沒提到的功能、設計或優化。如果你有想到值得做的額外建議，先用清單列出來給我看就好，絕對不要直接動手做。保持最小改動，是我這次最重要的原則。

💡 AI 的熱心常常是災難的開始。多做的每一分，都是你日後要理解、要維護、可能要除錯的負擔。先擋住，需要再加。

109. 要它承認不確定，別硬掰

什麼時候用：你怕 AI 為了給答案，把不確定的東西講得像真的一樣，害你被誤導。

這樣說： > 接下來如果你對某件事沒有把握、或這是你不確定的資訊，請直接告訴我你不確定，並說明你為什麼不確定、我可以怎麼自己查證。我寧可你老實說不知道，也不要你為了給答案而猜一個聽起來很篤定的錯誤內容給我。

💡 AI 最危險的不是答錯，而是很有自信地答錯。事先給它誠實的許可，它就比較願意說這個我不確定，你也才知道哪些答案需要再查一下？

110. 長對話重開新局的交接

什麼時候用：一段對話拖太久，AI 開始忘東忘西、愈來愈遲鈍，你想開個新對話重新來。

這樣說： > 我們這段對話有點長了，我打算開一個全新的對話重新開始。在那之前，請你幫我寫一份交接筆記：我們在做什麼專案、目前進度到哪、已經決定了哪些事、還有哪些待辦、以及有哪些重要背景是新對話一開始就必須知道的。寫成我可以直接複製貼到新對話的格式。

💡 對話愈長，AI 反而愈容易糊塗。學會請它寫交接筆記、開新局重啟，是老手維持 AI 頭腦清醒的祕訣，就像遊戲存檔後重開一局，狀態更好。

想走得更遠

這本手冊只是工具卡。真正把 Vibe Coding 從土法煉鋼變成系統性能力，靠的是完整方法論。

《零基礎 Vibe Coding》用 14 章帶你走完整條路：從認識這個時代、各家 AI 助手的脾氣，到規劃專案、寫提示詞、看懂 AI 產出、系統性除錯，再到親手做出網站、名單磁鐵、品牌銷售頁、數據儀表板，最後替作品把關品質、走向專業。

《零基礎 Vibe Coding》，Vista Cheng 著，碁峰資訊出版，2026 年 8 月上市。

更多免費資源與上市通知：vista.tw/books/vibe-coding

版權所有 © 2026 Vista Cheng (<https://www.vista.tw>)。本手冊為《零基礎 Vibe Coding》（碁峰資訊）之免費延伸資源，請勿直接轉傳檔案。