

Vibe Coding 錯誤訊息急救手冊

看到紅字不再慌：Claude Code／Codex／Antigravity 常見錯誤速查

Vista Cheng ／ <https://www.vista.tw>

這本急救手冊是《零基礎 Vibe Coding》（基峰資訊出版）的免費延伸資源。書裡第七章教你除錯的心法與觀念，這本手冊是它的隨身速查版：把你最可能撞見的紅字，一條一條翻成人話，並且直接告訴你「這樣跟 AI 說」就能修。

我特別補上一整個章節，為大家專門講解三大會自己動手改程式的 AI 工具（Claude Code、Codex 與 Antigravity）最常見的卡關。

怎麼使用這本手冊

1. 看到紅字，先深呼吸。錯誤訊息是給 AI 看的，不是給你看的。
2. 翻到對應章節，找到長得像的那條。
3. 照著「這樣跟 AI 說」把〔填空〕換成你的情況，複製貼上。
4. 真的找不到對應條目？直接用開頭的「萬能救援句」，幾乎都能開場。

💡 你完全不需要看懂那些英文。你要做的只有一件事：把紅字原封不動複製全文，交給 AI。

目錄

看到紅字，照著這張圖走	3
使用說明與萬能救援句	3
一、環境與安裝類	4
錯誤：command not found: node 或 command not found: npm	4
錯誤：command not found: python 或 python: command not found	5
錯誤：The engine "node" is incompatible 或 requires Node version >=18	5
錯誤：permission denied 或 EACCES: permission denied	5
錯誤：'xxx' is not recognized as an internal or external command (Windows)	6
錯誤：command not found (明明剛裝好卻找不到)	6
二、套件與相依類	7
錯誤：Cannot find module 'X' 或 Module not found: Can't resolve 'X' ...	7

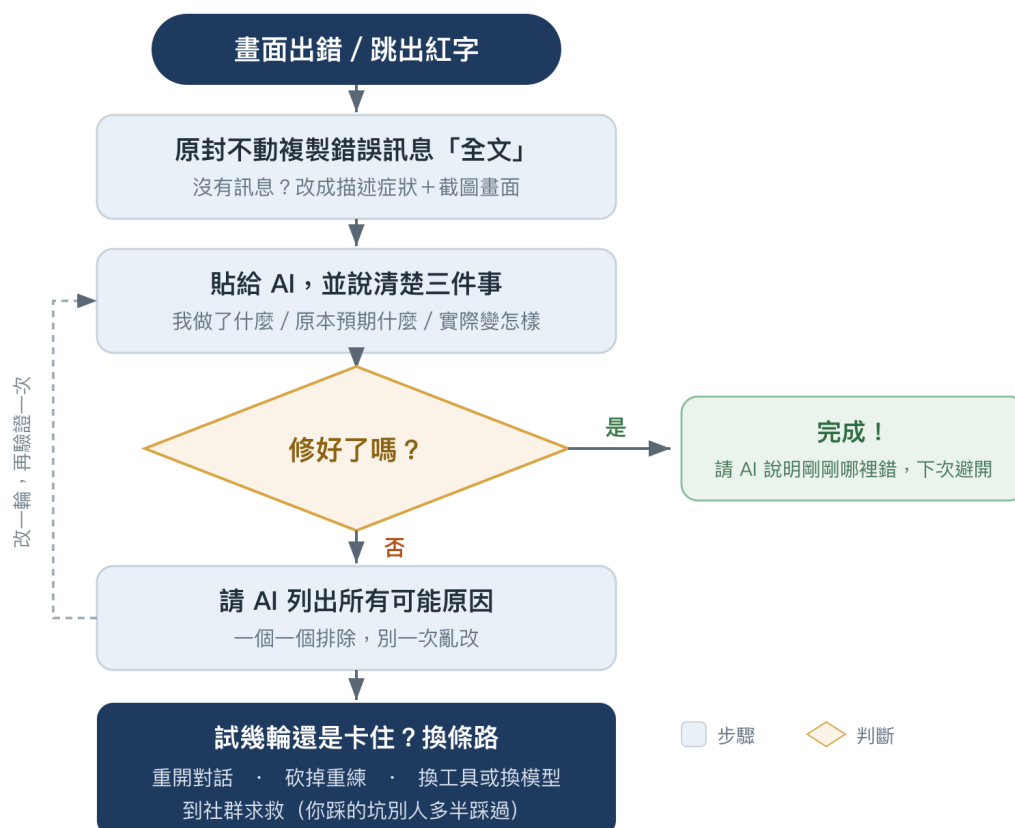
錯誤：npm ERR! (一大片紅字)	7
錯誤：unable to resolve dependency tree或peer dependency conflict..	7
錯誤：lockfile衝突 (package-lock.json出現<<<<<<標記)	8
錯誤：裝完套件後指令仍command not found	8
三、語法與型別類	9
錯誤：SyntaxError: Unexpected token.....	9
錯誤：TypeError: Cannot read properties of undefined (reading 'xxx')	9
錯誤：TypeError: X is not a function.....	9
錯誤：ReferenceError: X is not defined.....	10
錯誤：SyntaxError: Unexpected end of input	10
四、執行期與網路類	11
錯誤：EADDRINUSE: address already in use :::3000.....	11
錯誤：blocked by CORS policy	11
錯誤：Failed to fetch或fetch failed.....	11
錯誤：畫面一片空白 (沒有任何紅字)	12
錯誤：畫面一直轉圈、無限載入	12
錯誤：500 Internal Server Error.....	13
錯誤：Maximum call stack size exceeded.....	13
五、Git與版本控制類.....	13
錯誤：CONFLICT 或程式碼裡出現<<<<<<、=====、>>>>>>.....	13
錯誤：Updates were rejected/push rejected (remote有新東西)	14
錯誤：You are in 'detached HEAD' state.....	14
錯誤：不小心commit了大檔案或密鑰.....	15
錯誤：fatal: not a git repository.....	15
錯誤：Authentication failed/Permission denied (publickey)	15
六、部署類.....	16
錯誤：Build failed (部署時建置失敗)	16
錯誤：線上壞、本機好 (環境變數沒設定)	16
錯誤：部署後404或重新整理就失效	17
錯誤：部署成功了，但線上看不到更新 (快取)	17
錯誤：Function execution timed out (函式逾時)	17

錯誤：Module not found（只在部署時發生）	18
七、API 與金鑰類	18
錯誤：401 Unauthorized 或 403 Forbidden.....	18
錯誤：429 Too Many Requests	19
錯誤：API 金鑰失效或放錯位置.....	19
錯誤：金鑰不小心外洩（推上 GitHub 或貼出去了）	19
錯誤：API 呼叫逾時（ETIMEDOUT / request timeout）	20
八、資料庫類	20
錯誤：connection refused/could not connect to database	20
錯誤：migration failed（資料表建置/更新失敗）	21
錯誤：schema 不一致（drift，程式跟資料庫對不上）	21
錯誤：syntax error at or near "xxx"（SQL 語法錯誤）	21
錯誤：foreign key constraint 或 unique constraint 衝突.....	22
九、AI 工具本身的卡關.....	22
卡關：對話變慢、AI 開始忘東忘西、context 快爆了	22
卡關：AI 改到了你沒叫它改的檔案.....	23
卡關：AI 陷入無限修改迴圈，愈改愈糟	23
卡關：AI 宣稱修好了，但根本沒驗證（假成功）	24
卡關：AI 幻覺出不存在的套件、指令或 API.....	24
卡關：Claude Code 一直跳權限請求，卡住不動	25
卡關：MCP 伺服器連不上.....	25
卡關：AI 執行 git 或部署指令失敗，卻報成功.....	26
卡關：AI 讀不到某個檔案、或一直找錯路徑	26
卡關：AI 額度用完、或碰到速率限制	27
十、AI 幫倒忙類	27
幫倒忙：AI 悄悄刪掉了原本會動的功能	27
幫倒忙：過度工程，把簡單需求搞得超複雜.....	28
幫倒忙：改壞了，想要它還原.....	28
幫倒忙：它一直道歉，卻始終沒解決問題	29
幫倒忙：產出跟需求無關的東西	29
還是卡住的時候	29
想走得更深.....	30

看到紅字，照著這張圖走

看到紅字別慌：除錯決策流程

錯誤訊息是給 AI 看的，不是給你看的。照著箭頭走就好。



看到紅字別慌：除錯決策流程

使用說明與萬能救援句

先深呼吸。你現在看到的那一串紅字，不是在罵你，也不是你把電腦弄壞了。錯誤訊息（Error Message）其實是寫給 AI 看的線索，不是寫給你看的判決書，你不需要看懂每一個英文單字。你唯一要做的事，是把它原封不動、整段複製貼給你的 AI 助理，然後說一句「這是什麼意思，請告訴我怎麼修？」。就這樣，別緊張。

這份手冊的用法很簡單：翻到跟你紅字最像的那一條，照著這樣跟 AI 說，把提示詞複製、把〔填空〕換成你的情況，貼給 AI 就好。找不到一模一樣的沒關係，錯誤訊息的關鍵字（像 `not found`、`undefined`、`refused`）就是分類線索，抓大方向對就行。

如果你懶得找、或情況不在表上，直接用下面這三句萬能救援提示，適用任何錯誤：

我在做一個專案時遇到這個錯誤，我是零基礎，請用最白話的方式告訴我：
這是什麼意思、為什麼會發生、我該怎麼一步步修好。錯誤訊息如下：
〔貼上整段紅字〕

剛剛的修法我照做了還是一樣，請不要重複同一個方法。請你先幫我判斷
「最可能的原因有哪幾個」，再一個一個帶我確認，找到真正原因再動手
改。錯誤訊息一樣是：〔貼上整段紅字〕

我不確定這段錯誤在講哪個檔案、哪一行。請你先幫我看懂它指向哪裡，
需要我提供哪些檔案或資訊你才能判斷，直接告訴我要給你什麼。

記住一個心法：報錯就像跟醫生描述症狀，你講得愈完整（哪一步做了什麼、跳出什麼、期待本來會怎樣），AI 這位超強助理就診斷得愈準。好，讓我們開始！

一、環境與安裝類

這一類的紅字，多半發生在你剛裝好工具、第一次要跑指令的時候。它們幾乎都在說同一件事：「我找不到某個程式，或我沒有權限。」別擔心，這是每個人開場的必經之路。

錯誤：`command not found: node` 或 `command not found: npm`

你會在哪看到：在終端機（Terminal）輸入 `node -v` 或 `npm` 想確認有沒有裝好時跳出來。

白話意思：電腦說「我不認識 `node`／`npm` 這個指令」，通常是還沒安裝，或裝了但電腦找不到它放在哪。

常見原因：- 根本還沒安裝 `Node.js`。- 裝完之後沒有重開終端機，新指令還沒生效。- 安裝路徑沒有被加進 `PATH`（電腦的「搜尋清單」）。

這樣跟 AI 說：> 我在〔`Mac`／`Windows`〕上輸入 `node -v`，跳出 `command not found: node`。我是零基礎，請一步一步帶我確認 `Node.js` 有沒有裝、要怎麼裝，裝完要不要重開終端機。每一步請告訴我要在哪裡打字、打什麼。

💡 裝完任何新工具，先把終端機完全關掉再重開一次，很多「找不到指令」其實只是舊視窗還沒更新。

錯誤：`command not found: python` 或 `python: command not found`

你會在哪儿看到：想跑 Python 程式、輸入 `python` 時出現。

白話意思：電腦找不到 Python，或你打的名字跟它實際的名字不一樣。

常見原因：- 尚未安裝 Python。- 新版系統要用 `python3` 而不是 `python`。- 安裝了但沒加進 `PATH`。

這樣跟 AI 說：> 我輸入 `python` 出現 `command not found`，但我記得好像有裝過。請幫我判斷是不是要改打 `python3`，還有怎麼確認到底裝了沒。我用的是〔Mac／Windows〕。

💡 先試著把 `python` 改成 `python3` 再跑一次，這是最常見的解法。

錯誤：`The engine "node" is incompatible or requires Node version >=18`

你會在哪儿看到：安裝套件或啟動專案時，提示你的 Node 版本太舊或不符。

白話意思：這個專案挑食，它要吃比較新（或特定）版本的 Node，你現在這版它不收。

常見原因：- Node 版本太舊，專案要求更新。- 同時裝了好幾個版本，現在生效的是舊的那個。

這樣跟 AI 說：> 我啟動專案時出現 `requires Node version >=18`，但我不知道我現在是哪一版、也不知道怎麼換。請先教我怎麼查目前版本，再一步步帶我升級或切換到它要的版本。我用〔Mac／Windows〕。

💡 把整段版本需求連同你 `node -v` 查到的數字一起貼給 AI，它就能直接告訴你差多少、怎麼補。

錯誤：`permission denied` 或 `EACCES: permission denied`

你會在哪儿看到：安裝套件、或執行某個指令、寫入某個資料夾時被擋下。

白話意思：電腦說「這個動作你沒有權限做」，像是想進一個上鎖的房間。

常見原因：- 想寫入的是系統保護的資料夾。- 用了需要管理員權限的安裝方式。- 檔案的擁有者不是你目前的帳號。

這樣跟 AI 說：> 我在做〔說明你剛剛在做什麼，例如 `npm` 全域安裝〕時出現 `permission denied`。請先告訴我這通常代表什麼、有沒有「不用硬加權限」的安全做法，再帶我操作。我不希望亂用 `sudo` 把系統弄壞，請提醒我風險。

💡 看到權限錯誤，先別急著在指令前面加 `sudo`。很多情況有更安全的作法，先問 AI 有沒有替代方案。

錯誤：`'xxx' is not recognized as an internal or external command` (Windows)

你會在哪儿看到：在 Windows 的命令提示字元輸入某個指令時，整句話這樣回你。

白話意思：這是 Windows 版的「找不到指令」，意思跟 Mac 的 `command not found` 一樣。

常見原因：- 程式沒裝，或裝了沒加進系統 PATH。- 需要重開命令提示字元或重開機。

這樣跟 AI 說：> 我在 Windows 輸入〔你打的指令〕，出現 `is not recognized as an internal or external command`。請一步步帶我確認這個程式有沒有裝、怎麼把它加進 PATH，最後要不要重開機。

💡 Windows 改完 PATH 之後，記得把命令提示字元整個關掉重開，不然新設定不會生效。

錯誤：`command not found` (明明剛裝好卻找不到)

你會在哪儿看到：你才剛用套件管理器裝好某個工具，馬上跑卻說找不到。

白話意思：東西是裝好了，但電腦的「搜尋清單」還沒把新工具的位置收進去。

常見原因：- 安裝路徑沒進 PATH。- 終端機還是舊的，沒重新載入設定。- 裝在專案內（本地）但你用全域的方式呼叫。

這樣跟 AI 說：> 我剛用〔你的安裝方式，例如 `npm install -g`、`brew`〕裝好〔工具名〕，但一跑就說 `command not found`。請幫我判斷是 PATH 沒設好、還是要重開終端機、還是我呼叫的方式錯了，一步步帶我排查。

💡 「裝好卻找不到」十之八九是 PATH 或沒重開視窗的問題，先關掉終端機重開再試一次。

二、套件與相依類

寫程式很少從零開始，你會用到很多別人寫好的積木（套件，**Package**）。這一類錯誤通常在說：「你要用的積木，我手上沒有，或裝壞了。」

錯誤：`Cannot find module 'X' 或 Module not found: Can't resolve 'X'`

你會在哪儿看到：啟動或執行專案時，畫面停在某一行說找不到某個模組。

白話意思：程式想拿某塊積木來用，但那塊積木沒被裝進來。

常見原因：- 這個套件還沒安裝。- 名字拼錯，或大小寫不對。- `node_modules` 資料夾遺失或壞掉。

這樣跟 AI 說：> 我執行專案時出現 `Cannot find module ' [ 它說找不到的名字 ] '`。請先判斷這是「還沒安裝」還是拼錯名字？如果要安裝，直接給我完整的安裝指令，並告訴我要在哪個資料夾裡打。

💡 把 'X' 裡那個確切名字複製給 AI，它常常一眼就看出是少裝了、還是打錯字。

錯誤：`npm ERR!`（一大片紅字）

你會在哪儿看到：執行 `npm install` 安裝套件失敗時，噴出一整片以 `npm ERR!` 開頭的訊息。

白話意思：安裝過程出事了，真正的原因藏在這堆紅字裡的某一行。

常見原因：- 網路不穩或來源被擋。- 版本互相打架。- 之前裝到一半壞掉，殘留檔案卡住。

這樣跟 AI 說：> 我跑 `npm install` 失敗，噴出一大片 `npm ERR!`。我把整段完整貼給你，請幫我找出「真正的關鍵那一行」是什麼、代表什麼問題，再給我對應的解法。訊息如下：〔貼上整段紅字，不要只貼第一行〕

💡 `npm ERR!` 會有很多行，最有用的線索常在中後段。整段貼、別自己挑，讓 AI 幫你找重點。

錯誤：`unable to resolve dependency tree 或 peer dependency conflict`

你會在哪儿看到：安裝套件時，說相依關係無法解決、有 `peer dependency` 衝突。

白話意思：兩塊積木對該用哪一版的第三塊積木意見不合，卡住了。

常見原因：- 新套件跟現有套件要求的版本對不上。- 專案裡某些套件太舊。

這樣跟 AI 說：> 我安裝套件時出現 `peer dependency conflict`（完整訊息如下）。請先幫我看懂是「哪兩個套件、為了哪個版本」在衝突，再告訴我最安全的解法是什麼，並提醒我硬繞過去會不會有後遺症。訊息：〔貼上整段〕

💡 網路上常叫人加 `--force` 或 `--legacy-peer-deps` 硬裝，但那是繞過問題不是解決。先問 AI 有沒有更乾淨的作法。

錯誤：lockfile 衝突（`package-lock.json` 出現 <<<<<< 標記）

你會在哪儿看到：多人協作或你合併版本後，安裝時出錯，打開 lockfile 看到奇怪的箭頭符號。

白話意思：記錄「每塊積木裝哪一版」的清單被兩邊各改一次，打架了。

常見原因：- 合併（merge）不同版本時 lockfile 被雙方都動過。- 手動改了這個檔案。

這樣跟 AI 說：> 我的 `package-lock.json`（或 `pnpm-lock.yaml`）出現 <<<<<< 這種衝突標記，安裝就失敗。我是零基礎，這個檔案我看不懂也不敢亂改。請告訴我最安全的處理方式，最好是能直接重新產生一份乾淨的，一步步帶我做。

💡 lockfile 是電腦自動產生的檔案，通常最省事的解法是刪掉它重新安裝，讓工具重生一份，不用自己去改箭頭。

錯誤：裝完套件後指令仍 `command not found`

你會在哪儿看到：明明 `npm install` 成功了，想用那個套件提供的指令卻說找不到。

白話意思：積木裝在這個專案的工具箱裡，但你用全電腦通用的方式在叫它，兩邊對不上。

常見原因：- 套件裝在本地（專案內）而非全域。- 該用專案內的執行方式（例如透過 `npx` 或 `package.json` 的 `script`）。

這樣跟 AI 說：> 我用 `npm install` 裝了〔套件名〕，安裝顯示成功，但我直接打它的指令卻 `command not found`。請幫我判斷它是裝在本地還是全域，還有正確的呼叫方式應該是什麼（例如是不是要用 `npx` 或 `package.json` 裡的 `script`）。

💡 專案內裝的工具，通常要用 `npx` 指令名或寫進 `package.json` 的 `scripts` 來跑，而不是直接打指令名。

三、語法與型別類

這一類是程式碼本身寫法出了小差錯，通常是漏字、拼錯或括號沒打完整。好消息是它幾乎都會告訴你出錯的檔案跟行數，AI 修起來很快。

錯誤：SyntaxError: Unexpected token

你會在哪儿看到：一存檔或一執行就立刻報錯，程式根本跑不起來。

白話意思：程式的文法有地方不通順，電腦讀到某個字看不懂。

常見原因：- 少了逗號、引號、括號，或多打了一個。- 中英文標點混用（例如不小心打成全形的引號）。- 複製貼上時混進了奇怪字元。

這樣跟 AI 說：> 我存檔後出現 **SyntaxError: Unexpected token**，它指到〔檔名與行號，紅字裡會寫〕。請幫我看這附近是不是少了或多了標點、括號。我把那一段程式碼貼給你：〔貼上錯誤指到的那幾行〕

💡 語法錯誤最常見的元兇，是少一個括號或引號。把它指的那一行連同上下各兩三行一起貼給 AI，通常一秒就抓到。

錯誤：TypeError: Cannot read properties of undefined (reading 'xxx')

你會在哪儿看到：程式跑到一半、或某個按鈕點下去時當掉，這是最常見的紅字之一。

白話意思：程式想從一個其實根本不存在的東西身上拿資料，撲了個空。

常見原因：- 資料還沒載入完成就先被使用。- 名字打錯，拿到一個空值。- 預期有內容的清單其實是空的。

這樣跟 AI 說：> 我遇到 **TypeError: Cannot read properties of undefined (reading '〔它說的那個名字〕')**。這在我〔說明你剛做的動作，例如按下送出、開啟頁面〕時發生。請幫我判斷是不是某個資料還沒準備好就被用了，並教我怎麼加保護讓它不要一空就爆掉。相關程式碼：〔貼上〕

💡 這個錯誤幾乎都跟時機有關：東西還沒到，你就先去拿了。跟 AI 說清楚「我點了什麼才出問題的」，它會更快定位。

錯誤：TypeError: X is not a function

你會在哪儿看到：呼叫某個功能時，說它不是一個函式。

白話意思：你把一個不能被當成動作來用的東西，硬當成動作去執行了。

常見原因：- 名字拼錯，或大小寫不符。- 套件沒正確引入，拿到的不是預期的功能。- 版本更新後，那個功能改名或被移除。

這樣跟 AI 說：>我呼叫〔那個名字〕時出現 `is not a function`。請幫我確認這個名字有沒有拼錯、是不是引入的方式不對、還是它在新版被改名了。我用的套件是〔套件名與版本，若知道〕，程式碼片段：〔貼上〕

💡 如果是最近升級過套件才壞的，跟 AI 講一聲「我剛更新過版本」，這常常就是主因。

錯誤：ReferenceError: X is not defined

你會在哪儿看到：執行時說某個名字沒有被定義。

白話意思：你用了一個從來沒宣告過、電腦不認識的名字。

常見原因：- 變數名打錯字。- 忘了先宣告就直接使用。- 該引入的東西沒引入。

這樣跟 AI 說：>我遇到 `ReferenceError`：〔那個名字〕 `is not defined`。請幫我判斷是我把名字打錯了、還是漏了宣告、還是少 `import` 了某個東西。這是相關程式碼：〔貼上〕

💡 `is not defined`（沒定義）跟前面的 `is not a function`（不是動作）不一樣：前者是「這個名字根本不存在」，後者是「名字存在但用錯方式」。跟 AI 講清楚看到的是哪一個。

錯誤：SyntaxError: Unexpected end of input

你會在哪儿看到：存檔或執行時報錯，指的行號常常是檔案的最後一行。

白話意思：電腦讀到檔案結尾了，卻發現有個括號或引號開了沒關，還在等你補上。

常見原因：- 少了一個結尾的 `}`、`)` 或 `]`。- 引號只打了前半沒打後半。- 刪程式碼時不小心刪掉一半。

這樣跟 AI 說：>我出現 `Unexpected end of input`，它指到檔案最後。我猜是有括號或引號沒關好。請幫我檢查這個檔案的括號有沒有成對，把該補的補上。整個檔案內容如下：〔貼上整個檔案〕

💡 這類錯誤把整個檔案貼給 AI 最有效，因為問題常常不在它指的那一行，而是前面某處少關了一個括號。

四、執行期與網路類

程式跑得起來，但一操作就出狀況，或畫面怪怪的：畫面空白、一直轉圈或連不到伺服器。這一類跟執行當下和連線有關，也是設計網頁最常撞到的一區。

錯誤：EADDRINUSE: address already in use :::3000

你會在哪看到：啟動專案時，說某個 **port**（連接埠，例如 **3000**）已經被占用。

白話意思：你要用的那扇門，已經有另一個程式站在那裡了，擠不進去。

常見原因： - 之前開的專案沒關乾淨，還占著同一個 **port**。 - 另一個程式剛好也用這個 **port**。

這樣跟 AI 說： > 我啟動專案時出現 **EADDRINUSE: address already in use :::**〔那個數字〕。請告訴我兩條路：一是怎麼把占用這個 **port** 的舊程式關掉，二是怎麼改用另一個 **port** 啟動。我用〔**Mac/Windows**〕，一步步帶我。

💡 最快的解法通常是把之前那個沒關的終端機視窗找出來關掉，或直接換一個 **port** 號啟動。

錯誤：blocked by CORS policy

你會在哪看到：網頁想跟後端或某個 **API** 拿資料時，在瀏覽器的開發者工具（**Console**）看到這行紅字，畫面資料出不來。

白話意思：這是瀏覽器的安全門禁，它擋住了「從這個網站去跟另一個網站要資料」的動作，因為對方沒說「我允許你」。

常見原因： - 後端沒設定允許你的前端來存取。 - 前端呼叫的網址不對。 - 本地開發跟線上網址對不上。

這樣跟 AI 說： > 我的網頁抓資料時，**Console** 出現 **blocked by CORS policy**。請用白話跟我解釋 **CORS** 是什麼，然後幫我判斷該改的是前端呼叫的方式，還是後端要開放權限，並分別告訴我怎麼做。我的前端跑在〔網址〕，要打的 **API** 在〔網址〕。

💡 **CORS** 幾乎都要在「被呼叫的那一端（後端／**API**）」設定允許，光改前端通常沒用。把兩邊網址都告訴 **AI**，它才好判斷。

錯誤：Failed to fetch 或 fetch failed

你會在哪看到：程式要去拿網路資料時失敗，畫面可能卡住或顯示錯誤。

白話意思：程式伸手去某個網址拿東西，但那隻手根本沒碰到對方。

常見原因： - 網址打錯，或後端根本沒啟動。 - 網路斷線。 - 被 CORS 或防火牆擋下（會伴隨其他訊息）。

這樣跟 AI 說： > 我抓資料時出現 **Failed to fetch**。請幫我一項一項排查：後端有沒有在跑、網址對不對、是不是網路或 CORS 問題。告訴我每一項要怎麼自己確認。我要打的網址是〔網址〕。

💡 先確認「你要連的那個後端或服務，現在到底有沒有在運作」。很多 **Failed to fetch** 只是因為後端沒開。

錯誤：畫面一片空白（沒有任何紅字）

你會在哪儿看到： 打開網頁只有全白，什麼都沒有，看起來像沒報錯。

白話意思： 程式其實在背後出錯了，只是錯誤躲在瀏覽器的開發者工具裡，沒顯示在畫面上。

常見原因： - 有 JavaScript 錯誤讓整頁沒渲染出來。 - 某個資料是空的導致畫面崩掉。 - 路徑或引入出錯。

這樣跟 AI 說： > 我的網頁打開是一片空白，畫面上沒有任何錯誤。請教我怎麼打開瀏覽器的開發者工具、切到 **Console** 分頁去看有沒有紅字，我把看到的紅字貼給你。如果 **Console** 是空的，也請告訴我下一步該看哪裡。

💡 畫面空白不等於沒錯，只是錯誤藏起來了。按下鍵盤的 **F12** 打開開發者工具、看 **Console** 分頁，紅字通常就在那裡等你。

錯誤：畫面一直轉圈、無限載入

你會在哪儿看到： 頁面卡在載入動畫，資料永遠出不來。

白話意思： 程式在等一個永遠等不到的回應，於是是一直卡在載入中的狀態。

常見原因： - 要拿的資料請求失敗了，但沒有處理失敗的情況。 - 後端沒回應或回應很慢。 - 載入的判斷邏輯有誤，永遠停在還在載入。

這樣跟 AI 說： > 我的頁面一直轉圈、載不出來也不報錯。請幫我判斷是不是「資料請求失敗了但程式沒處理」，並教我怎麼加上「載入失敗就顯示提示」而不是一直轉。同時教我怎麼在開發者工具的 **Network** 分頁看那個請求是成功還失敗。相關程式碼：〔貼上〕

💡 開發者工具的 **Network**（網路）分頁能看到每個請求的狀態，紅色的那筆通常就是卡住的元兇。

錯誤：500 Internal Server Error

你會在哪儿看到：呼叫後端或打開頁面時，回傳 500。

白話意思：不是你的錯，是伺服器那一端自己出事了，它自己也處理不了。

常見原因：- 後端程式碼有 bug。- 資料庫連不上或查詢出錯。- 環境變數（設定值）沒設好。

這樣跟 AI 說：> 我呼叫後端時回 500 Internal Server Error。我知道這代表伺服器端出錯。請教我去哪裡看後端的錯誤紀錄（log），找出真正的錯誤那一行，我再貼給你。我的後端是〔平臺或框架名，若知道〕。

💡 500 只是「表面代碼」，真正的原因寫在伺服器的 log 裡。跟 AI 要「怎麼看 log」，找到裡面那行真正的錯誤才是關鍵。

錯誤：Maximum call stack size exceeded

你會在哪儿看到：程式突然當掉，噴出這行，通常伴隨一大串重複的訊息。

白話意思：程式陷入了自己一直呼叫自己的無底洞，像照鏡子照到無限層，記憶體撐爆了。

常見原因：- 函式在沒有出口的情況下不斷呼叫自己（無限遞迴）。- 兩個功能互相呼叫，繞不出來。- 資料結構意外形成迴圈。

這樣跟 AI 說：> 我遇到 Maximum call stack size exceeded。我知道這通常是「無限循環呼叫」造成的。請幫我從這段程式碼找出是哪裡在無限呼叫自己、要怎麼加上停止條件。程式碼：〔貼上你懷疑的那段〕

💡 這個錯誤幾乎都跟某段程式沒有終點、一直重複有關。把最近改動的那段貼給 AI，它很擅長抓出這種迴圈。

五、Git 與版本控制類

打個比方，Git 就好像是你的遊戲存檔系統，讓你隨時能回到之前的版本。但存檔合併時偶爾會打結，這一類就是教你怎麼解結，不要慌張刪東西。

錯誤：CONFLICT 或程式碼裡出現 <<<<<<< 、===== 、>>>>>>>

你會在哪儿看到：合併（merge）或拉取（pull）別人的更新後，Git 說有衝突，打開檔案看到這些箭頭符號夾著兩段程式碼。

白話意思：同一個地方，你跟別人（或另一個版本）各改了一次，Git 不知道該聽誰的，請你來決定。

常見原因： - 兩個版本改到了同一行。 - 拉取遠端更新前，本地也動過同一處。

這樣跟 AI 說： > 我合併時出現 **CONFLICT**，檔案裡有 <<<<<<<、=====、>>>>>>> 這些符號。我是零基礎，看不懂哪段是誰的。請幫我解讀：
===== 上面那段是誰的、下面那段是誰的，我該保留哪個、還是兩個都要。
我把整段貼給你：〔貼上含箭頭符號的那一整段〕

💡 那些箭頭符號是 **Git** 給你的選擇題，不是壞掉。<<<<<<< 到 ===== 之間是你的版本，===== 到 >>>>>>> 之間是對方的。決定好之後要把三行箭頭符號一起刪掉。

錯誤：Updates were rejected / push rejected (remote 有新東西)

你會在哪儿看到： 你想把改動推上去 (**push**) 時，被 **Git** 擋下來拒絕。

白話意思： 雲端上有你電腦裡還沒有的更新，**Git** 怕你蓋掉別人的東西，所以先擋住，要你先把新的拉下來。

常見原因： - 別人（或你在另一臺裝置）先推了新東西上去。 - 你本地的版本落後於雲端。

這樣跟 AI 說： > 我 **push** 時出現 **Updates were rejected**，看起來是雲端有我還沒有的更新。請一步步帶我安全地先把遠端的更新拉下來、跟我的合併好，再推上去。過程中如果會有衝突，請提醒我怎麼處理。

💡 看到 **push** 被拒，標準動作是先 **pull** 再 **push**。別急著用 **--force** 硬推，那可能會蓋掉別人的心血。

錯誤：You are in 'detached HEAD' state

你會在哪儿看到： 切換版本後，**Git** 提示你進入 **detached HEAD** 狀態。

白話意思： 你現在站在一個歷史上的某個舊存檔點，而不是在正常的工作線上，這裡改東西不會被好好記錄。

常見原因： - 直接切換到某個舊的提交 (**commit**) 而不是分支。 - 誤操作跳到了歷史的某一點。

這樣跟 AI 說： > **Git** 說我在 **detached HEAD** 狀態。我不太確定自己是怎麼跑到這裡的，也怕改的東西不見。請用白話解釋這是什麼狀態、我現在該怎麼安全回到正常的分支，如果我剛剛有改東西，怎麼保住它。

💡 **detached HEAD** 不危險，只是站錯位置。先別繼續改東西，讓 **AI** 帶你回到正常分支（通常是 **main**）再說。

錯誤：不小心 `commit` 了大檔案或密鑰

你會在哪看到：`push` 時因為檔案太大被拒，或你突然發現把密碼、金鑰之類的機密提交進去了。

白話意思：有個不該進存檔的東西被記進版本歷史了，得把它請出去。

常見原因： - 忘了設定忽略清單（`.gitignore`）。 - 把裝相依套件的資料夾、影片、金鑰檔一起提交了。

這樣跟 AI 說： > 我不小心把〔大檔案／含密碼的檔案〕`commit` 進去了，還沒（或已經）`push`。請告訴我：一、怎麼把它從版本記錄裡移除；二、怎麼設定 `.gitignore` 以後不再提交它；三、如果裡面有密鑰，我是不是必須立刻去把那把金鑰作廢重發。請按嚴重程度排順序帶我做。

💡 如果 `commit` 進去的是金鑰或密碼，就算你之後刪掉，也要當它「已經外洩」，第一件事是去平臺把那把金鑰作廢重發，不能只靠刪檔。

錯誤：`fatal: not a git repository`

你會在哪看到：在某個資料夾下打 `Git` 指令時，被回這句。

白話意思：你現在站的這個資料夾，還沒被設定成 `Git` 的存檔專案，所以 `Git` 不知道要管哪裡。

常見原因： - 還沒對這個專案做初始化（`git init`）。 - 你人在錯的資料夾，真正的專案在別的地方。

這樣跟 AI 說： > 我打 `Git` 指令時出現 `fatal: not a git repository`。請幫我判斷是「這個專案還沒被 `Git` 初始化」，還是「我站錯資料夾了」。教我怎麼確認目前在哪個資料夾、以及該怎麼做才對。

💡 先確認你的終端機「現在在哪個資料夾」（可以請 AI 教你用 `pwd` 或 `cd` 確認），很多時候只是走錯房間。

錯誤：`Authentication failed / Permission denied (publickey)`

你會在哪看到：`push` 或 `pull` 到 GitHub 等平臺時，說身分驗證失敗。

白話意思：`Git` 不確定你是不是有權限的本人，把門關上了。

常見原因： - 密碼或存取權杖（`token`）過期、輸入錯誤。 - 平臺已不接受密碼，改用 `token` 或金鑰（`SSH key`）。 - 帳號沒有這個專案的權限。

這樣跟 AI 說： > 我 `push` 到〔GitHub／其他平臺〕時出現 `Authentication failed`。我是零基礎。請幫我判斷現在該用哪種驗證方式（密碼、存取權杖、還是 `SSH` 金鑰），並一步步帶我設定好。我用〔Mac／Windows〕。

💡 現在多數平臺已經不收帳號密碼，改用存取權杖或 SSH 金鑰。先問 AI 你的平臺現在要哪一種，再照著設定。

六、部署類

部署（Deploy）就是把你本機做好的東西搬到網路上，讓全世界都連得到。經典魔咒是我電腦上明明好好的，一上線就壞。這一類專治這種水土不服。

錯誤：Build failed（部署時建置失敗）

你會在哪儿看到：把專案推上部署平臺（例如 Vercel、Cloudflare 或 Netlify）後，建置步驟中斷、顯示失敗。

白話意思：平臺在幫你打包上線的過程中撞到錯誤，包不起來就上不了線。

常見原因：- 程式碼裡其實有錯誤，只是本機沒觸發。- 少裝了某個套件，或版本對不上。- 環境變數沒設定。

這樣跟 AI 說：> 我部署到〔平臺名〕時出現 **Build failed**。我把完整的建置紀錄（**build log**）貼給你，請幫我找出真正卡住的那一行、代表什麼問題、怎麼修。紀錄如下：〔貼上整段 **build log**，尤其是紅字附近〕

💡 部署平臺都會有一份完整的建置紀錄，真正的原因在裡面。把那整段貼給 AI，比只說「**build** 失敗」有用一百倍。

錯誤：線上壞、本機好（環境變數沒設定）

你會在哪儿看到：本機跑得好好的，上線後某些功能壞掉，尤其是連 API 或資料庫的部分。

白話意思：你電腦裡有一份祕密設定檔（環境變數）沒被搬上線，線上版少了鑰匙就開不了那些門。

常見原因：- 本機的 **.env** 設定沒有同步到部署平臺。- 部署平臺上的變數名稱拼錯或漏設。

這樣跟 AI 說：> 我的專案本機正常，但部署後〔某功能，例如登入、抓資料〕就壞了。我懷疑是環境變數沒設。請教我怎麼比對本機的 **.env** 跟部署平臺上設定的變數，找出漏掉哪些、怎麼在平臺上補齊。我用的平臺是〔平臺名〕。

💡 若遇到本機好、線上壞的狀況，十之八九是環境變數沒搬上去。**.env** 檔通常不會被上傳，你得手動到部署平臺的設定裡一個一個補。

錯誤：部署後 404 或重新整理就失效

你會在哪儿看到：首頁打得開，但點進子頁面、或在子頁面按重新整理，就出現 404 找不到頁面。

白話意思：伺服器不知道這些網站內部的路線該怎麼走，一重新整理就找不到路。

常見原因：- 單頁應用（SPA）的路由沒在伺服器端設定好。- 部署設定裡少了「把所有路徑導回首頁」的規則。

這樣跟 AI 說：> 我的網站首頁正常，但點進內頁或按重新整理就 404。我猜是路由沒設定好。請幫我判斷這是不是單頁應用要加的「rewrite／redirect 規則」，並告訴我在〔平臺名〕上具體要加什麼設定。我用的框架是〔框架名，若知道〕。

💡 這是前端網站部署的超經典坑。跟 AI 講清楚你用哪個平臺、哪個框架，它通常能直接給你那幾行設定貼上去就好。

錯誤：部署成功了，但線上看不到更新（快取）

你會在哪儿看到：你明明改好推上去了，打開網站卻還是舊的樣子。

白話意思：你看到的是暫存的舊版本，新版還沒被你這邊的瀏覽器或平臺更新過來。

常見原因：- 瀏覽器快取（暫存）了舊檔案。- 平臺或 CDN 的快取還沒刷新。- 部署其實沒成功，你以為成功了。

這樣跟 AI 說：> 我部署後線上還是舊的。請先教我怎麼強制重新整理、清掉瀏覽器快取，確認不是我這邊的問題，再教我怎麼確認部署到底有沒有真的成功、以及怎麼清掉平臺的快取。我用〔平臺名〕。

💡 先用強制重新整理的方式，排除你自己瀏覽器的快取（Mac 按 Cmd+Shift+R，Windows 按 Ctrl+Shift+R）。如果還是舊的，才往平臺快取或部署沒成功去查。

錯誤：Function execution timed out（函式逾時）

你會在哪儿看到：線上某個功能（尤其是後端運算、呼叫 AI、處理大量資料）跑很久後失敗，說逾時。

白話意思：這段程式跑太久，超過平臺允許的時間上限，被強制中斷。

常見原因：- 呼叫的外部服務回應太慢。- 一次處理太多資料。- 免費方案的時間限制較短。

這樣跟 AI 說：>我線上的〔功能名〕出現 **timed out**（逾時）。請幫我判斷是不是「這段做太多事、跑太久」，並給我兩種方向：一是怎麼讓它跑快一點，二是能不能把工作拆小或改成背景處理。我用〔平臺名〕，這段大概在做〔說明它在做什麼〕。

💡 逾時通常代表一口氣做太多。跟 AI 討論能不能拆成小步驟，往往比硬調時間上限更根本。

錯誤：Module not found（只在部署時發生）

你會在哪儿看到：本機好好的，一部署就說找不到某個模組。

白話意思：本機因為某些寬鬆設定沒發現的問題，在嚴格的線上環境現形了，常常是大小寫或漏裝。

常見原因：- 檔名大小寫不一致（你的電腦不分大小寫，但伺服器會分）。
- 某個套件只裝在本機、沒被記進安裝清單。

這樣跟 AI 說：>我本機正常，但部署時出現 **Module not found**：〔它說找不到的名字〕。請重點幫我檢查兩件事：一、我 **import** 的檔名大小寫跟實際檔名有沒有完全一致；二、這個套件有沒有被正確記進 **package.json**。我用〔平臺名〕。

💡 這個坑超常見：Mac 不分大小寫，但線上伺服器分。**Header** 跟 **header** 在你電腦上是同一個，上線後就變兩個。請 AI 特別幫你比對大小寫。

七、API 與金鑰類

當你的程式要用別人的服務（例如：AI、地圖或金流），通常需要一把鑰匙（API 金鑰）來證明身分。這一類的錯誤，多半跟這把鑰匙有關：沒帶、帶錯、用太兇、或不小心弄丟。

錯誤：401 Unauthorized 或 403 Forbidden

你會在哪儿看到：呼叫某個 API 或服務時，被回 401 或 403。

白話意思：401 是你沒表明身分或身分不對，403 是知道你是誰，但這件事不准你做。兩者都是門禁問題。

常見原因：- API 金鑰沒帶、帶錯、或過期。- 帳號權限不足。- 金鑰放錯位置，沒被正確送出。

這樣跟 AI 說：>我呼叫〔服務名〕的 API 時回 401（或 403）。請先幫我分清楚這兩個的差別，再帶我一步步排查：金鑰是不是有效、有沒有正確被送出、帳號權限夠不夠。我串接的方式大概是〔簡述，例如在前端 **fetch**、後端呼叫〕。

💡 401 先查鑰匙對不對、有沒有帶，403 先查這個帳號有沒有這個權限？方向不同，別搞混。

錯誤：429 Too Many Requests

你會在哪儿看到：短時間內密集呼叫某服務後，被回 429。

白話意思：你敲門太快太多次，對方請你先冷靜一下，這叫速率限制（Rate Limit）。

常見原因：- 短時間內請求次數超過方案允許。- 程式不小心在迴圈裡狂打 API。- 免費方案額度用完。

這樣跟 AI 說：> 我呼叫〔服務名〕時出現 429 Too Many Requests。請幫我判斷是不是我的程式在短時間內打太多次（例如迴圈裡重複呼叫），並教我怎麼加上稍等再重試的機制，避免一直撞牆。相關程式碼：〔貼上〕

💡 看到 429，先檢查是不是有段程式在迴圈裡不小心狂呼叫 API。修好迴圈通常比升級方案更該做。

錯誤：API 金鑰失效或放錯位置

你會在哪儿看到：串接服務時一直失敗，訊息提到 invalid key、missing key，或就是連不上。

白話意思：那把鑰匙不是打錯、就是沒插進正確的鑰匙孔。

常見原因：- 金鑰複製時多了空格或少了字。- 放進了錯的環境變數名稱。- 前端後端該用的金鑰種類搞混。

這樣跟 AI 說：> 我串接〔服務名〕一直失敗，感覺是 API 金鑰的問題。請帶我逐項檢查：金鑰有沒有複製完整（頭尾有沒有多空格）、放的環境變數名稱對不對、以及這個服務前端跟後端該用的金鑰是不是不同把。

💡 複製金鑰時最容易頭尾多帶一個空格或換行。重新乾淨地複製一次，常常就好了。

錯誤：金鑰不小心外洩（推上 GitHub 或貼出去了）

你會在哪儿看到：你發現把金鑰寫死在程式碼裡推上了公開倉庫，或收到平臺寄來「偵測到你的金鑰外洩」的警告信。

白話意思：你家鑰匙的照片被貼到公開布告欄了，任何人都可能拿去用，甚至幫你花錢。

常見原因：- 把金鑰直接寫在程式碼裡而非環境變數。- 忘了把設定檔加進 .gitignore。

這樣跟 AI 說：> 我不小心把〔服務名〕的 API 金鑰外洩了（寫在程式碼裡推上了公開倉庫）。請按照緊急程度帶我處理：一、立刻去哪裡把這把金鑰作廢、重新產生一把；二、怎麼把金鑰改放到環境變數、不再寫死在程式裡；三、怎麼設定 .gitignore 避免再犯。

💡 金鑰一旦外洩，「刪掉那行程式碼」是沒用的，因為歷史記錄還在。唯一的解法是「立刻作廢舊金鑰、重發新的」。這件事要第一時間做，別拖。

錯誤：API 呼叫逾時（ETIMEDOUT / request timeout）

你會在哪儿看到：呼叫某個外部服務時，等很久後回報逾時。

白話意思：你敲了門，但對方一直沒開，等到你這邊先放棄了。

常見原因：- 對方服務當下很慢或掛了。- 網址、埠號或網路設定有誤。- 一次要的資料量太大。

這樣跟 AI 說：> 我呼叫〔服務名〕的 API 一直逾時。請幫我排查：是對方服務本身慢／掛了，還是我這邊網址設定有問題，還是我一次要太多資料。教我怎麼分別確認，以及怎麼加上「逾時重試」讓它更穩。

💡 先去那個服務的官方狀態頁（status page）看看是不是對方正處於掛掉的狀態？如果是對方的問題，你怎麼改都沒用，等它恢復就好。

八、資料庫類

資料庫（Database）是你的程式存放資料的倉庫。這一類錯誤多半在說：倉庫連不上、倉庫的格局跟你程式想的不一樣、或你放進去的東西違反了倉庫的規矩。

錯誤：connection refused / could not connect to database

你會在哪儿看到：程式啟動或第一次要存取資料時，說連不上資料庫。

白話意思：你的程式去敲資料庫的門，結果沒人應門，可能是地址錯了，或倉庫根本沒開。

常見原因：- 資料庫沒啟動。- 連線字串（地址、帳號密碼或埠號）有誤。- 網路或防火牆擋住。

這樣跟 AI 說：> 我的程式出現 connection refused，連不上資料庫。請帶我逐項排查：資料庫服務有沒有在跑、連線設定（主機、埠號、帳號、密碼、資料庫名稱）對不對、是不是被防火牆擋。我用的資料庫是〔資料庫種類，例如 PostgreSQL、MySQL 或 Supabase〕。

💡 先確認資料庫本身有沒有在運作？本機的資料庫常常只是忘了啟動，雲端的則要確認連線字串一字不差。

錯誤：migration failed（資料表建置/更新失敗）

你會在哪看到：執行資料庫的 migration（把資料表結構的變更套用上去）時失敗。

白話意思：你想重新裝潢倉庫的格局，但施工到一半撞到問題，改不下去。

常見原因： - 新的結構跟已有資料衝突（例如要加一個「不能為空」的欄位，但舊資料是空的）。 - migration 檔本身有錯。 - 之前跑到一半的 migration 卡住了。

這樣跟 AI 說： > 我跑資料庫 migration 時失敗，訊息如下。請幫我看懂它卡在哪一步、為什麼，最安全的處理方式是什麼（尤其我不想弄丟現有資料）。
訊息：〔貼上整段〕。我用〔資料庫或工具名〕。

💡 涉及 migration 一定要小心，動到的是真實資料。跟 AI 強調「我不想弄丟資料」，請它先解釋清楚再動手，重要資料先備份。

錯誤：schema 不一致（drift，程式跟資料庫對不上）

你會在哪看到：程式抓資料時說某個欄位不存在，或型別對不上，但你覺得明明有。

白話意思：程式手上那張倉庫平面圖，跟倉庫的實際格局長得不一樣了，兩邊沒對齊。

常見原因： - 改了程式的資料結構定義，但沒同步更新資料庫。 - 本機跟線上的資料庫結構不一致。

這樣跟 AI 說： > 我遇到程式跟資料庫對不上的問題（它說某欄位不存在／型別不符，但我以為改過了）。請幫我判斷是不是「我改了定義但沒同步到資料庫」，並帶我確認兩邊目前的實際結構、怎麼把它們對齊。我用〔工具名，例如 Prisma、Drizzle、Supabase〕。

💡 這種「明明改了卻沒生效」的問題，先分清楚你改的是程式裡的定義，還是資料庫的實際結構？兩者要同步，光改一邊不夠。

錯誤：syntax error at or near "xxx"（SQL 語法錯誤）

你會在哪看到：執行一段資料庫查詢（SQL）時，說語法有錯。

白話意思：你給倉庫管理員的指令，文法寫錯了，它看不懂。

常見原因： - 少了逗號、括號或引號。 - 用了資料庫的保留字當欄位名。 - 不同資料庫的語法有細微差異。

這樣跟 AI 說：> 我執行這段 SQL 時出現 `syntax error at or near "`〔它指的那個字〕"。請幫我找出文法錯在哪、改好給我。我用的資料庫是〔種類〕。這是我的 SQL：〔貼上整段查詢〕。

💡 不同資料庫的 SQL 有小差異。一定要跟 AI 講清楚你用哪一種（PostgreSQL、MySQL、SQLite…），它才能給對版本的寫法。

錯誤：foreign key constraint 或 unique constraint 衝突

你會在哪儿看到：新增或刪除資料時被擋，說違反外鍵或唯一鍵限制。

白話意思：倉庫有規矩。唯一鍵是說這個值不能重複（例如同一個 E-mail 不能註冊兩次）；外鍵是說這筆資料牽連著別筆資料，不能亂刪亂加。

常見原因：- 想存入一個已經存在的、不該重複的值。- 想刪掉一筆還被其他資料引用著的資料。- 想引用一筆根本不存在的關聯資料。

這樣跟 AI 說：> 我在〔新增／刪除〕資料時出現〔unique／foreign key〕constraint 衝突。請用白話解釋這條規矩在保護什麼，並幫我判斷正確的做法是什麼（例如是不是該先檢查有沒有重複、或先處理關聯的資料）。完整訊息：〔貼上〕。

💡 這類錯誤其實是資料庫在幫你把關，避免髒資料。別想著硬繞過限制，先問 AI 這條規矩想擋的是什麼，順著它做才安全。

九、AI 工具本身的卡關

這一區是這份手冊最想補給你的部分。前面八區都在講程式的錯，但當你用 Claude Code、Codex 或 Antigravity 這類會自己動手做事的 AI 工具時，卡關的常常是 AI 這個夥伴本身：它改壞了、它鑽牛角尖、它說謊或是連不上。這些書裡不會教，但你天天會遇到。這一區教你怎麼管理你的 AI 夥伴。

卡關：對話變慢、AI 開始忘東忘西、context 快爆了

你會在哪儿看到：同一個對話聊很久之後，AI 回應變慢、開始忘記你前面講過的設定，或提示你 context（上下文記憶）快滿了。

白話意思：AI 的短期記憶像一張桌子，東西堆太滿就放不下，舊的會被擠掉，它就開始忘事。

常見原因：- 單一對話進行太久、塞了太多檔案和來回。- 一次貼了超大量的程式碼或紀錄。

這樣跟 AI 說：> 我們這個對話已經很長了，我覺得你開始忘記前面的東西。請幫我把目前的進度、已經做了什麼決定、還沒做什麼、關鍵的檔案和設定，總結成一段交接筆記，我要拿它到新對話重新開始。

💡 與其在快爆掉的對話裡硬撐，不如請它寫一份交接筆記，然後開新對話貼上去。乾淨的記憶體讓 AI 反應更準。

工具差異：Claude Code 可用 `/compact` 壓縮對話、`/clear` 清空重開，長專案建議把重點寫進 `CLAUDE.md` 讓它每次自動讀到 | Codex 開新的 session 重啟最乾脆，重點寫進專案的說明檔 | Antigravity 也是開新對話最省事，把關鍵脈絡重新貼一次。

卡關：AI 改到了你沒叫它改的檔案

你會在哪儿看到：你只叫它修一個小地方，結果它順手動了好幾個其他檔案，甚至改壞了本來沒問題的功能。

白話意思：這位助理太熱心，自作主張多做了事，超出你交代的範圍。

常見原因：- 指令講得太籠統，它自行腦補了範圍。- 它認為順手一起改比較好。

這樣跟 AI 說：> 你剛剛改的檔案超出我要求的範圍了。我只要你改〔指定的那個檔案或功能〕，其他一律不要動。請先列出你剛剛總共改了哪些檔案，把不該動的那些還原回去，之後每次動手前先跟我說「你打算改哪些檔案」，等我同意再做。

💡 對付自作主張，最好的辦法是事先劃界線：「只准改這個檔，其他碰都別碰。」下指令時把範圍講死，能省掉很多善後。

工具差異：Claude Code 改檔前通常會請求你同意，看到它要動不相關的檔案時直接拒絕即可；也可以先用它的計畫模式（Plan Mode）讓它只出計畫不動手 | Codex 建議在有版本控制（Git）的乾淨狀態下讓它跑，出包用 `git` 一鍵還原 | Antigravity 動手前多半會列出計畫，養成先看計畫再放行的習慣。

卡關：AI 陷入無限修改迴圈，愈改愈糟

你會在哪儿看到：同一個 bug，它改了又改，每次都說「這次應該好了」，但問題還在，甚至衍生出新問題，愈弄愈亂。

白話意思：它在原地打轉、鬼打牆，用類似的思路一直試，卻沒跳出來重新想。

常見原因：- 它一直在症狀上補強，沒找到真正的病根。- 它被錯誤的假設困住，不肯換方向。

這樣跟 AI 說：> 停。我們已經來回改了好幾次都沒好，請不要再直接改程式了。請你退一步，先列出這個問題最可能的 2 到 3 個根本原因，我們一個

一個來確認，找到真正的病根再動手。也請你老實說，是不是有哪個假設你一直沒驗證。

💡 AI 一旦鬼打牆，最有效的一句話是「停，先別改，我們重新診斷」。逼它從一直試，切換到先想清楚，往往一講就可以搞定。

工具差異：Claude Code 若愈改愈糟，善用它的還原能力回到修改前的乾淨版本，再換個說法重講一次問題 | Codex 卡住時，把整段來回丟給它請它重新審視自己的推理很有效 | Antigravity 可以直接開新對話、換一個角度重新描述問題，跳出舊思路。

卡關：AI 宣稱修好了，但根本沒驗證（假成功）

你會在哪儿看到：它信誓旦旦說「已完成、功能正常」，你一實際去點、去跑，發現根本沒好。

白話意思：它以為自己做好了，但沒有真的去執行、去確認，就先來跟你報喜了。

常見原因：- 它只在腦中推演，沒實際跑程式或看結果。- 它把程式碼寫完，誤當成功能正常。

這樣跟 AI 說：> 你說修好了，但我這邊測試還是不行。請不要只是「認為」好了，請你實際去執行／驗證，並把「你怎麼驗證的、看到什麼結果」告訴我。如果你沒辦法親自驗證，請老實說「我沒辦法驗證，需要你幫我確認以下這幾點」，而不是直接說完成。

💡 **養成習慣：**AI 說「完成」時，你自己一定要親手驗一次。並且明確要求它「給我驗證的證據」，而不是收下它的口頭保證。

工具差異：Claude Code、Codex 或 Antigravity 都能實際執行指令來驗證，所以你可以理直氣壯要求它跑給我看、把輸出貼出來，別接受純文字的「應該可以了」。

卡關：AI 幻覺出不存在的套件、指令或 API

你會在哪儿看到：它很有自信地叫你安裝某個套件、或用某個功能，結果一裝一用就報說找不到，Google 也查無此物。

白話意思：AI 有時會一本正經地編造看起來很合理、實際上不存在的東西，這叫幻覺（Hallucination）。

常見原因：- 它把好幾個相似的東西記混了。- 它猜了一個聽起來合理的名字。- 它引用的是過時或根本沒有的版本。

這樣跟 AI 說： > 你剛剛叫我用的〔套件／指令／功能名〕，我裝／用了之後說不存在，我也查不到。請你確認這東西是不是真的存在、有沒有官方文件可查。如果你不確定，請明講「我不確定」，我們去找真正存在的替代方案，不要用猜的。

💡 對聽起來很合理但你沒聽過的名字要有戒心。請 AI 附上官方文件連結，是戳破幻覺最快的方法：講不出來源的，八成是編的。

工具差異： Claude Code、Codex 或 Antigravity 都能實際上網或讀官方文件查證，所以遇到可疑名稱，直接叫它去查證、附上出處，而不是照單全收。

卡關：Claude Code 一直跳權限請求，卡住不動

你會在哪儿看到： Claude Code 每要做一個動作（改檔、跑指令）就停下來問你同不同意，你按到手痠，或它卡在等你回應。

白話意思： 它為了安全，做每件事前都先問過你，這是保護機制，不是壞掉。

常見原因： - 預設就是動手前先問。 - 某些指令被判定為需要確認。 - 它在等你的回覆，但你沒注意到。

這樣跟 AI 說： > 你一直停下來問我同不同意，我想讓流程順一點。請告訴我：哪些「安全、可逆」的動作我可以一次性允許、不用每次問；哪些「有風險」的動作（例如刪檔、部署、動到金鑰）務必每次都先問我。幫我拿捏這個界線。

💡 權限請求是保護你的好設計，別急著全部關掉。可以把讀檔、查資料這類安全動作設成免問，但遇到刪除、部署或付費這類動作，請一定保留每次確認。

工具差異： Claude Code 可在設定裡把常用又安全的指令加進允許清單（allowlist），減少重複詢問；也有一次性的核准選項 | Codex 有不同的核准模式可調整它「多常問你」 | Antigravity 同樣可在設定裡調整自動執行的範圍。共通原則：把方便留給安全動作，把把關留給危險動作。

卡關：MCP 伺服器連不上

你會在哪儿看到： 你想讓 AI 連上某個外部工具（透過 MCP，一種讓 AI 接外部服務的標準），結果它說連不上、工具不可用，或那些功能整組消失。

白話意思： AI 要接的那個「外掛工具」沒接上線，可能是沒開、設定錯、或鑰匙不對。

常見原因： - 設定檔寫錯（路徑、名稱或金鑰）。 - 那個 MCP 服務本身沒啟動。 - 需要重啟 AI 工具才會生效。

這樣跟 AI 說： > 我想用的〔MCP 工具名〕連不上，相關功能都不能用。請帶我逐項排查：設定檔有沒有寫對、需要的金鑰有沒有填、那個服務有沒有在跑、要不要重啟工具才生效。我用的是〔Claude Code／Codex／Antigravity〕，錯誤訊息（若有）：〔貼上〕。

💡 MCP 連不上，八成問題在設定檔的某個小地方（拼字、路徑、金鑰）。改完設定記得完全重啟 AI 工具，很多時候只是沒重開而已。

工具差異： Claude Code 用 /mcp 指令可以查看每個 MCP 伺服器的連線狀態，一眼看出哪個掛了 | Codex 與 Antigravity 也各有 MCP 設定檔，排查方向一樣：先看狀態、再核對設定、最後重啟。

卡關：AI 執行 git 或部署指令失敗，卻報成功

你會在哪看到： 你叫 AI 幫你提交、推送或部署，它說已完成、已上線，但你看根本沒動、或線上沒更新。

白話意思： 它跑的指令其實失敗了，但它沒仔細看回傳結果，就先跟你報成功了。

常見原因： - 指令失敗但 AI 沒解讀錯誤輸出。 - 驗證身分、權限或網路的環節出錯。 - 它把指令送出去了，當成成功了。

這樣跟 AI 說： > 你說已經〔push／部署〕成功，但我去看根本沒更新。請你重新執行一次，並把「指令的完整回傳訊息」原封不動貼給我，我們一起看它到底是成功還是失敗。不要只跟我說結論，要給我證據。

💡 涉及 git 和部署，一定要眼見為憑。叫 AI 把指令的原始輸出貼出來，或你自己去線上、去 GitHub 確認，別只信它的口頭報告。這是踩過無數次的教訓。

工具差異： Claude Code、Codex 或 Antigravity 都會實際執行這些指令，你有權要求它把終端機的完整輸出貼出來；部署類尤其要自己再開網站用強制重新整理確認一次。

卡關：AI 讀不到某個檔案、或一直找錯路徑

你會在哪看到： 它說找不到你明明存在的檔案，或一直在錯的資料夾裡打轉。

白話意思： 它對檔案放在哪的認知跟實際對不上，像拿著錯的地圖找路。

常見原因： - 它在錯的資料夾（工作目錄）下操作。 - 檔名、路徑的大小寫或拼字不符。 - 檔案其實在別的地方，或還沒建立。

這樣跟 AI 說： > 你說找不到〔檔名〕，但它確實存在。請先告訴我你現在認為的「工作目錄」是哪裡，再幫我確認這個檔案的正確完整路徑。我們先把「你站在哪、檔案在哪」對齊，再繼續。

💡 檔案找不到，多半是 AI 搞錯資料夾。先跟它確認彼此認知的路徑不一致，把地圖對齊，問題常常就消失了。

工具差異： Claude Code、Codex 都在你開啟的專案資料夾內作業，可請它先列出目前目錄與檔案清單確認位置 | Antigravity 同理，先讓它顯示目前在哪、看得到哪些檔，然後再往下走。

卡關：AI 額度用完、或碰到速率限制

你會在哪看到： 用到一半，AI 工具說你這段時間的用量到上限了，或要你稍後再試。

白話意思： 你的方案在這個時段能用的量用完了，需要等額度回復或升級方案。

常見原因： - 短時間高強度使用，撞到速率上限。 - 訂閱方案的當期額度用盡。 - 用了很吃資源的大型任務。

這樣跟 AI 說： > 我碰到用量／額度限制了。在等額度回復的這段時間，請幫我把目前的進度和接下來的計畫整理成一份筆記，這樣我等一下（或換個工具）可以無縫接上。另外也給我幾個省用量的建議，避免太快又撞上限。

💡 撞到額度別乾等，先請它整理進度筆記，你可以換到另一個 AI 工具繼續，或利用空檔自己測試、規劃。長任務盡量拆小段做，比較不會一次燒光額度。

工具差異： Claude Code、Codex 或 Antigravity 各有自己的額度與速率規則，撞牆時可考慮切換到手邊另一個工具接續；把進度筆記存成檔案，換工具時貼上去就能接手。

十、AI 幫倒忙類

AI 很強，但它偶爾會好心辦壞事。這一區跟第九區不同：這裡不是工具連線出問題，而是 AI 在完成任務的過程中，做了你不想要的事。學會辨識這幾種狀況並即時喊停，你就能穩穩地當它的主管。

幫倒忙：AI 悄悄刪掉了原本會動的功能

你會在哪看到： 它幫你加了新功能，或改了某處之後，你發現「本來好好的另一個功能」不見了或壞了。

白話意思： 它為了完成新任務，順手把它認為用不到的舊東西清掉了，但那其實還在用。

常見原因： - 它誤判某段程式是多餘的。 - 改動時牽連到相關的部分卻沒察覺。

這樣跟 AI 說： > 你加新功能之後，我發現〔原本的某功能〕不見／壞了。請比對你這次的改動，找出是不是把某段原本有用的程式刪掉或改壞了，把它恢復回來。之後請記住：沒有我的同意，不要刪除任何現有功能，就算你覺得它沒用。

💡 用版本控制（Git）就是為了這種時刻：隨時能還原到上一個好的存檔點。動大工程前先存檔（commit），出事一鍵回復，不用慌張。

幫倒忙：過度工程，把簡單需求搞得超複雜

你會在哪看到： 你只想要一個小功能，它卻給你一大套架構、裝一堆套件、拆成好多檔案，你完全看不懂也不敢動。

白話意思： 它用蓋一棟大樓的規格，來幫你釘一個層板，殺雞用了牛刀。

常見原因： - 它預設要做得專業、完整與可擴充。 - 你沒說我要最簡單的版本。

這樣跟 AI 說： > 這個需求其實很簡單，你做得太複雜了，我看不懂也維護不了。請用最簡單、最少檔案或最少套件的方式重做一次，目標是我這個零基礎的人也能看懂每一步。我真正要的只是〔一句話講清楚最小需求〕。

💡 下指令時主動加一句「請用最簡單的方式做，不要過度設計」。對零基礎的你來說，看得懂、改得動，比專業架構重要得多。

幫倒忙：改壞了，想要它還原

你會在哪看到： AI 一連串改動之後東西壞了，你只想回到還沒改之前那個能動的狀態。

白話意思： 你想按下回到上一個存檔，把剛剛的變動整組撤銷。

常見原因： - 一次改太多，難以逐一挑出哪裡出錯。 - 愈修愈亂，不如整組退回重來。

這樣跟 AI 說： > 剛剛這一連串改動把東西弄壞了，我不想再一點一點修。請幫我把這次對話所做的所有改動整組還原，回到我們開始改之前那個能正常運作的版本。還原後先跟我確認一切正常，我們再重新規劃。

💡 「與其修一團亂，不如退回重來」常常更省時間。這也是為什麼動手前先 commit 存檔這麼重要：你永遠有路可退。

工具差異： Claude Code 可用它的還原功能（例如 rewind）退回較早的狀態 | Codex 建議靠 Git 還原到乾淨的提交 | Antigravity 同樣可搭配 Git 回到上一個存檔點。共通心法：讓 Git 當你的安全網。

幫倒忙：它一直道歉，卻始終沒解決問題

你會在哪看到：每次回你都是「抱歉、是我的疏忽、我馬上改」，然後又是同樣的錯，問題原地踏步。

白話意思：它把力氣花在道歉上，但沒真的換方法，禮貌很多、進展卻很少。

常見原因：- 它困在同一個錯誤思路裡出不來。- 它在迎合你的情緒，而非解決技術問題。

這樣跟 AI 說：> 不用道歉，我要的是解決。請你先停下來，用一句話清楚說明「這個問題目前真正的卡點是什麼」，還有「你接下來要試的方法，跟前面失敗的那些有什麼不同」。如果你其實也不確定，就直說，我們一起換個方向找資料。

💡 道歉對你沒有幫助，別被它的客氣安撫住。直接把它拉回技術層面：「這次的方法跟上次哪裡不一樣？」逼它給出新思路，而不是新道歉。

幫倒忙：產出跟需求無關的東西

你會在哪看到：你要的是 A，它洋洋灑灑給了你 B，答非所問或做錯方向。

白話意思：它誤會了你的意思，往錯的方向全速前進。

常見原因：- 你的需求有模糊地帶，它自己猜了一個。- 它抓錯了重點。

這樣跟 AI 說：> 這不是我要的。在你繼續做之前，請先用你自己的話，把你理解我要的是什麼，複述一遍給我確認。我真正要的是〔用一兩句話重講一次，愈具體愈好，最好舉個例子〕。等我說對了，你再動手。

💡 發現方向歪掉，第一時間喊停，別讓它繼續往錯的路跑。請它先複述你的需求再開工，能在源頭擋掉大部分的誤會，省下大量的重複修改。

還是卡住的時候

如果你把整份手冊翻過一遍，紅字還是在那裡瞪著你，這裡有四條退路，任選一條，別跟它死磕。

第一，開一個全新的對話重來：AI 的記憶清空後，常常一下就看出剛才鑽牛角尖的盲點，很多「怎麼修都修不好」換個乾淨對話就通了。

第二，砍掉重練那一小段：與其在一團亂麻上繼續打結，不如把出問題的那一小塊還原、甚至刪掉重做一次，這不是失敗，是老手都在用的正常手段。

第三，換個工具或換個模型問問看：同一個問題，Claude Code、Codex 或 Antigravity 的角度不一樣，A 卡住的地方，B 可能一眼就破。

第四，去專業社群求救：把你的錯誤訊息（記得先把金鑰之類的機密塗掉）貼到相關的社群或論壇（<https://www.facebook.com/groups/vibecoding.club>），你遇到的坑，幾乎一定有人先踩過、也一定有人願意拉你一把。

最後想跟你說：會遇到這些紅字，正代表你真的在動手做，而不是只在旁邊看。每一個被你解掉的錯誤，都在幫你吧遇到問題怎麼跟 AI 描述、怎麼判斷方向這套本事練起來，這才是 Vibe Coding 真正值錢的地方。卡住很正常，我們都是這樣一路撞過來的。深呼吸，複製紅字，貼給你的 AI 夥伴，繼續往前。我相信，你可以的。

想走得更深

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：<https://www.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，歡迎分享領取頁連結，請勿直接轉傳檔案。