

軟體工程師如何善用AI提升生產力



CLAIRE CHANG
2025/01/14



Blog

ABOUT ME

- 1 Flash遊戲開發者
- 2 H5網頁遊戲前端開發
- 3 .Net, PHP等後端程式開發
- 4 串流播放器、伺服器開發
- 5 OpenCV影像辨識程式開發
- 6 使用Tensorflow、YOLO等做即時影像辨識

生成式AI正融入日常生活，驅動未來科技爆發

生產力提升

生成式AI工具在真實商業任務中的應用顯著提高了用戶的工作效率，例如，在客戶服務領域，使用AI的客服代表可以每小時處理更多客戶查詢。

客戶服務效率	13.8%提升
文件撰寫效率	59%提升
程式設計任務完成率	126%提升

生成式AI可顯著提升生產力

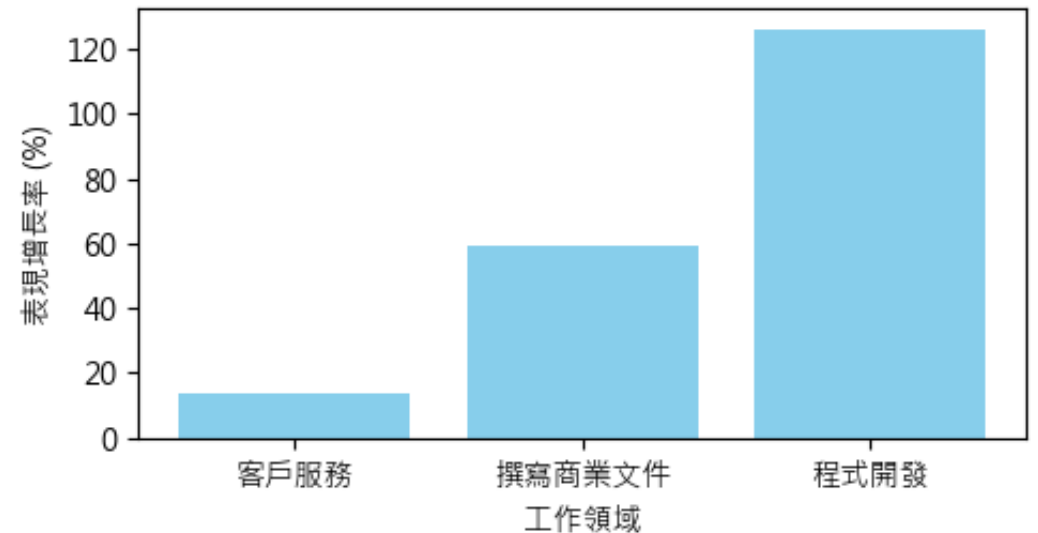
品質改善

生成式AI不僅提高了工作速度，還改善了工作的品質，例如，使用AI輔助的專業人員撰寫的商業文件品質得分顯著提高。

人機協作的未來

生成式AI不會取代人類，而是作為一種強大的工具，幫助人們更高效地完成工作，人機協作將成為未來工作環境的一部分。

不同工作領域，使用AI時增加的效率



生成式人工智慧的未來潛力



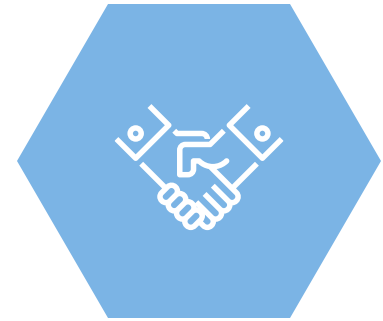
商業應用的快速採用

生成式人工智慧對於知識工作者可以提升其工作效率。可簡化複雜的工作，如稅法導讀和法律文件總結，提高專業勞工的效率。



易於獲得且成本低

生成式人工智慧工具不需要大量的硬體投資，也不需要建立實體基礎設施。企業能夠更容易地與人工智慧專家合作，設計針對特定需求的客製化工具。

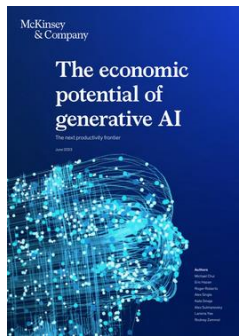
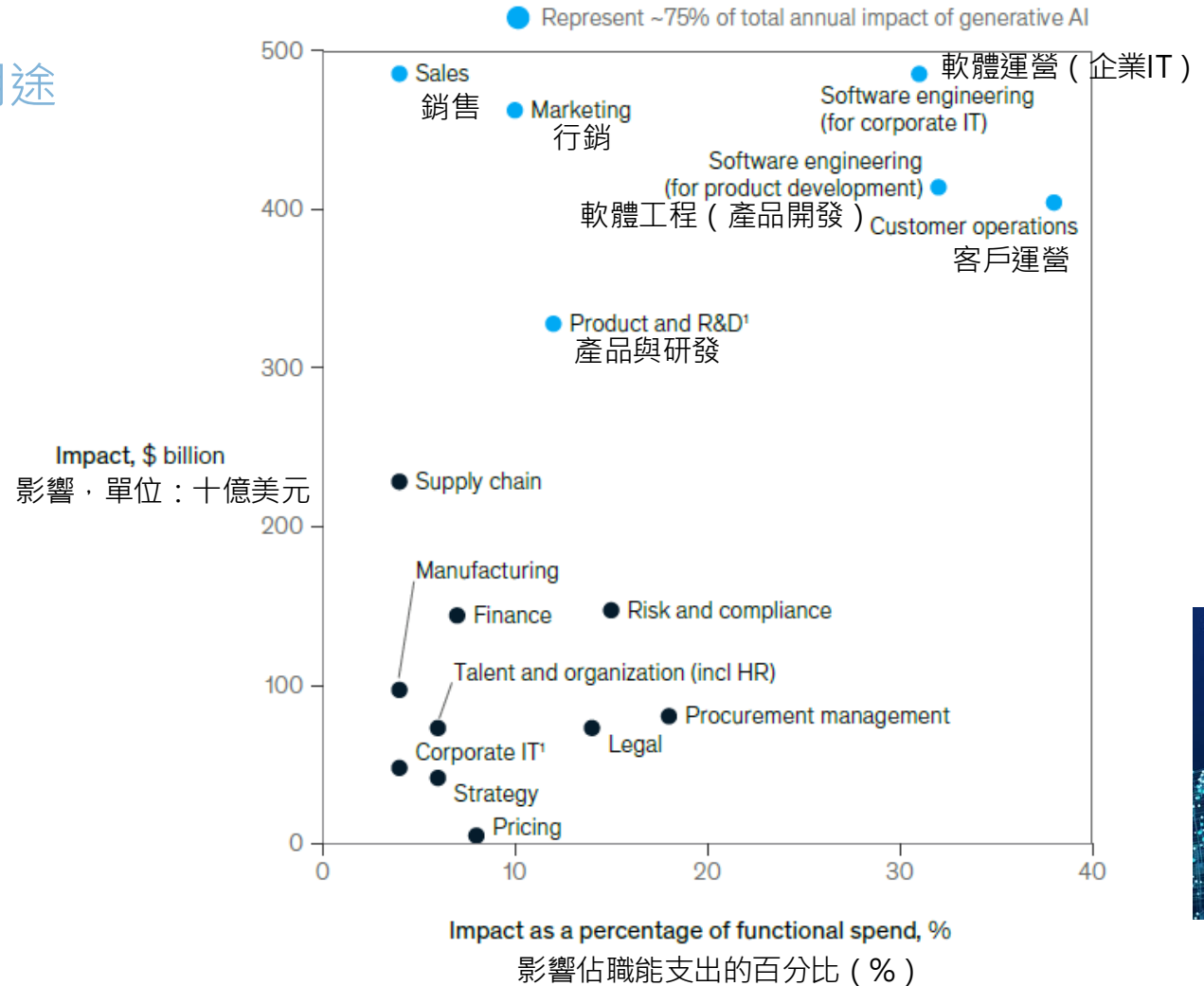


創新與挑戰的平衡

成功採用AI提升工作效率的關鍵，取決於如何平衡創新與挑戰。避免生成錯誤或帶有偏見的內容，每一個員工都應學習如何善用AI但不過度依賴所產出的內容。

麥肯錫：生成式人工智慧的經濟潛力

占了75%生成式人工智慧的業務用途





使用AI工具該注意些什麼？

AI 能自動化重複性任務，讓工程師專注於更具創造性與策略性的工作。但其結果可能包含錯誤、不符合需求或缺乏最佳實踐的部分。

為什麼人們說『AI賦能』

- 生成式AI 是一種革命性的技術，不僅改變了知識的傳遞方式，更可能重塑人類社會的結構。
- 我們應謹慎使用AI功能，以確保能從中受益而不是依賴。



正確使用AI的例子：輔助學習

在完成英文創意寫作後，可以使用AI工具(Grammarly)來檢查文法錯誤。學生完成寫作後，將其提交給AI檢查，然後將AI指出的錯誤列出並理解這些錯誤的原因，再進一步請教師解釋和指導。學生按照對自己現況能力的新了解而更有針對性的進行下一步的成長，方能達到提高能力的目標。

不正確使用AI的例子：完全依賴翻譯工具

在創意寫作時，直接使用Google翻譯將中文文本轉換為英文，然後將這些翻譯結果提交作為作業。完全沒有涉及到發自於學生本身的任何自主思考，錯失了練習英文的機會，沒有達到學習的目的，反而浪費了時間和資源，變成老師根本就是批改電腦生成內容的滑稽景象，也可能誤創造出學生有在學習的假象。

AI寫程式對開發者是危機，也是轉機



危機

- 即使完全沒有概念也能夠寫程式
- 初階工程師的角色容易被取代
- 更難從產出的程式去判斷工程師能力的優劣
- 可能產出帶有錯誤的程式碼
- 軟體開發團隊的結構可能會有巨大的改變，AI驅動的改變可能帶來多種職位的消失，尤其是在軟體測試方面



轉機

- 跨領域的學習更加容易
- AI對於學習基礎知識可以提供非常好的協助
- 開發 AI 應用的需求正在從數據科學團隊轉移到軟體工程和企業應用團隊
- 大幅增加軟體開發的效率並降低開發成本
- 能花更多時間在解決創造性的問題

軟體開發的環境正在劇烈變動

數位時代 | 861人追蹤 ☆ 追蹤

會寫程式不再是金飯碗！美國軟體工程師職缺5年腰斬，是AI惹的禍？



數位時代

2024年11月28日



一直以來，軟體工程師普遍被認為是待遇優渥、好找工作的專業，但在AI逐漸成熟的現在，這份印象正在逐漸被打破。《紐約時報》便在近日的文章中直指，過去程式設計一直被認為是工作有保障的金飯碗，但在AI浪潮來襲後，這樣的狀況開始出現變化。

在台灣，大學所學不好找工作、職涯中途想轉換跑道改做待遇更好的工作，有類似情況或念頭的人們，其中一部分會選擇修習速成的程式課程成為一名「工程師」。像這樣的短期課程在美國同樣有許多，然而開始有學生發現，即使上完課成為工程師，仍然很難找工作。

36歲的雷登（Florencio Rendon）在連續被建築公司開除後，決定找份更安穩的工作，於是參加程式訓練課程，希望能成為工程師。然而經歷四個月課程順利畢業後，他發現這身程式設計的本領，仍然很難找到工作。

2022年，Google旗下AI團隊曾聲稱，他們在程式比賽中測試了AI模型AlphaCode，發現效果已經媲美經過幾個月到一年訓練的新手工程師，而到了今年，Google已在財報會議上表示，超過4分之1的新程式碼現在都是由AI生成。



亞馬遜AWS CEO Matt Garman 圖/Amazon

亞馬遜AWS執行長麥特·加爾曼（Matt Garman）今年也曾告訴員工，未來24個月內，可能大多數開發人員都不會負責寫程式，這些基本任務會被AI給取代，並希望員工能夠繼續提升自己培養新能力，「這代表我們必須更了解客戶需求、知道我們要做什麼，比起在座位上實際寫程式，這會越來越成為我們的工作內容。」

能夠善用AI的工程師將如虎添翼



Photo Credit: Shutterstock / 達志影像

Google

Google 內部超過四分之一新程式碼由 AI 生成！CEO 表示：提高了公司效率

2024年度最佳: AI程式設計, “一人公司”時代的真正開端



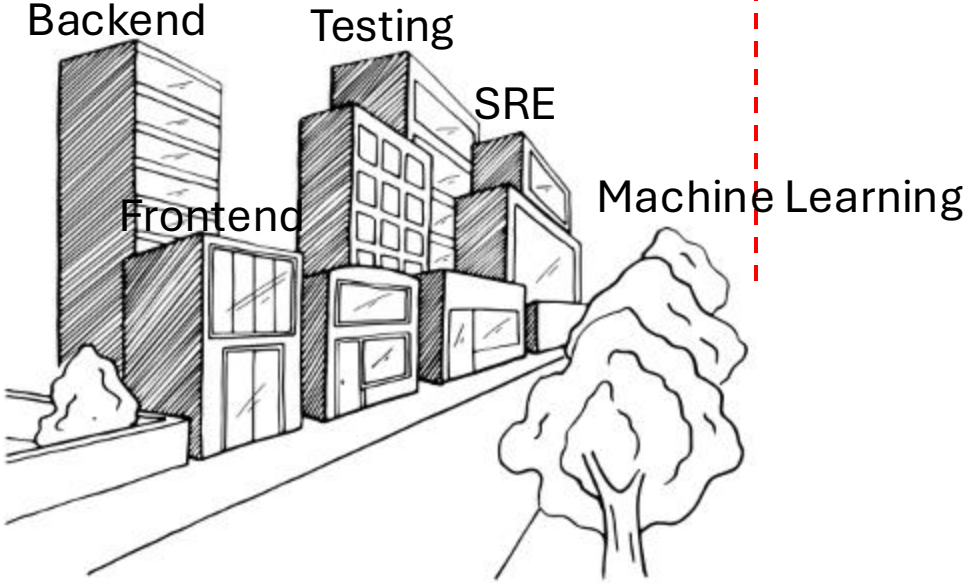
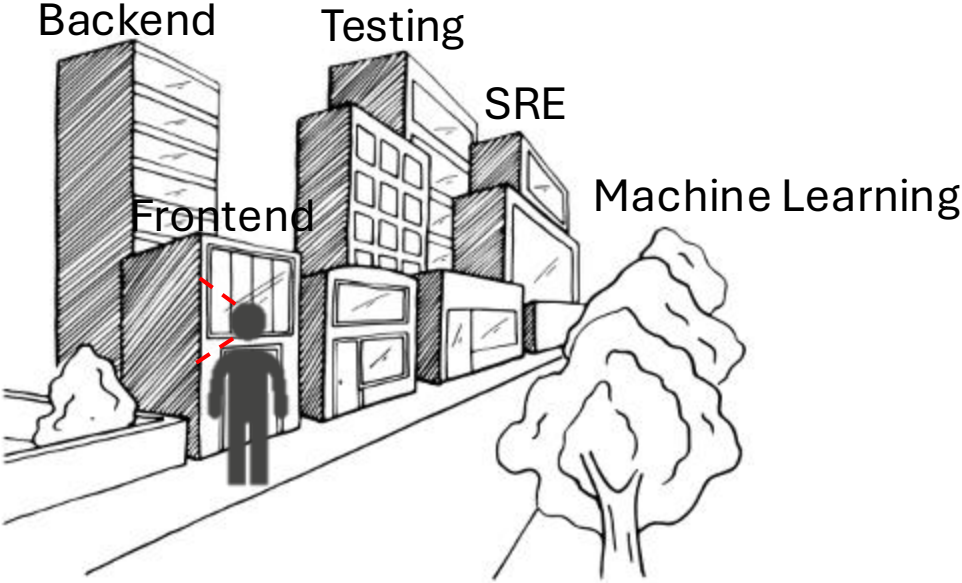
RexAA | 2025/01/01 • 0



軟體工程師角色的轉變

執行者

監督與創新者



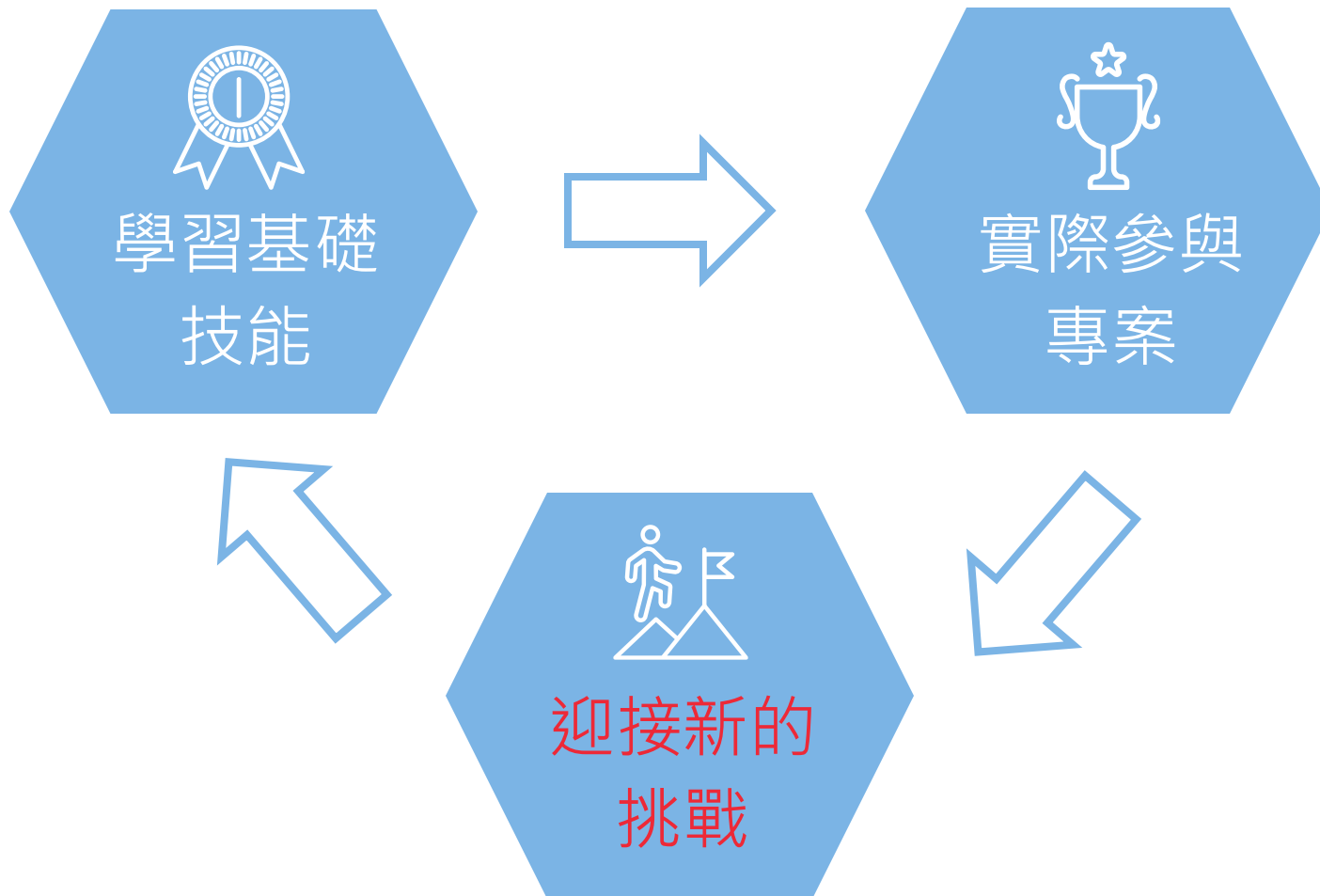
軟體工程師角色的轉變



- 透過AI工具，可以顯著縮短工程師學習新語言、新框架的時間，降低跨域門檻。
- 對**基礎軟體開發概念的理解**和對**整體技術架構**的掌握度，將變得更重要。
- 撰寫程式的時間縮短，**創造性、解決問題的能力**變得更加重要。
- 工程師的技能要求將會有很大的改變，軟體工程師需要**更多元化的技能集**，以適應角色變化和團隊結構的重新組合。

我們需要學習更多元、跨域的技能，以應付軟體產業可預見、即將迎來的巨大改變

AI職場賦能的 三個關鍵步驟



奔跑吧! 兄弟!

讓我們正面迎戰充滿挑戰與不確定性的軟體開發新世界



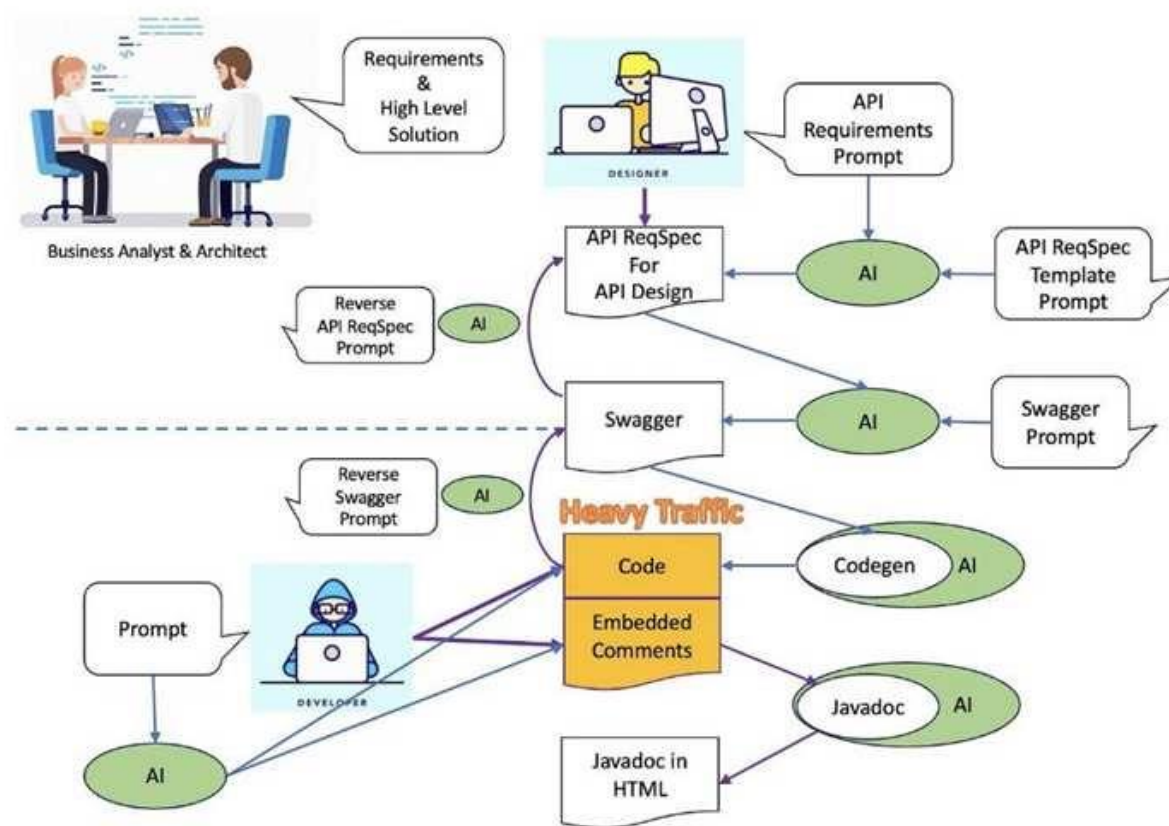
使用AI工具效率真的有提升到126%嗎？

- Sida Peng 和來自微軟、GitHub 以及麻省理工學院的研究夥伴們進行了主文中提到的第三項研究。
- 在這次研究中，70 位程式設計師被要求使用 JavaScript 開發一個 HTTP 伺服器。研究將參與者分成兩組，一組是使用 GitHub Copilot AI 工具的「實驗組」，另一組則是完全不使用任何 AI 工具的「對照組」。
- 參與者的程式設計經驗平均有 6 年，每天寫程式的時間約 9 小時，為經驗豐富的開發者。
- 研究結果顯示，沒有使用 AI 工具的程式設計師平均完成任務的時間是 160.89 分鐘（約 2.7 小時），而使用 AI 工具的程式設計師則平均只花 71.17 分鐘（約 1.2 小時）。這兩組的表現差異在統計上達到顯著水準，p 值為 0.0017。

當人工智慧出錯時，除錯可能會花費比人工智慧實際為我們節省的時間還要多。

AI-Assisted Design & Development for API (A2D2-API)

Without testing flow



AI 程式碼生成的挑戰與風險

除錯過程的隱性成本

AI 能快速完成 70% 的工作，但最後 30% 的修正和優化卻可能非常耗時

錯誤循環與效率損失

AI 提供的修正建議可能導致其他模組出現錯誤，開發者需要不斷與 AI 互動來解決問題，最終耗費更多時間

安全性與品質問題

AI 生成的程式碼可能包含安全漏洞或不符合最佳實踐，可能導致應用程式在生產環境中出現重大錯誤



因此，掌握如何與AI協作開發是非常重要的！

我們需要了解**AI擅長哪些項目**。在使用AI除錯時，需要**掌握大方向**來引導AI，使他能給出正確的除錯方向。我們應同時結合**AI學習**並利用AI來生成程式碼，才能更好的找出與AI協作的最佳實踐

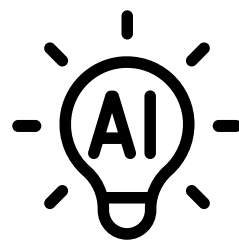
結合傳統和AI工具的優勢

- 使用AI工具完成重複性任務，將較複雜的功能或決策交由人類完成。
- 較單純的任務使用AI工具，關鍵核心功能由人手動開發，大型項目結合兩種方式進行。
- 透過AI工具提供即時建議，輔助人類開發，並促進新手學習與知識傳遞。
- 由人類來審查AI工具輸出的程式碼，也使用AI提供建議學習軟體開發規範，確保符合專案需求與高品質標準。
- 混和人工智慧和人類智慧，綜合AI工具的效率與知識庫優勢，搭配人類創意與靈活性，共同提升專案效能。



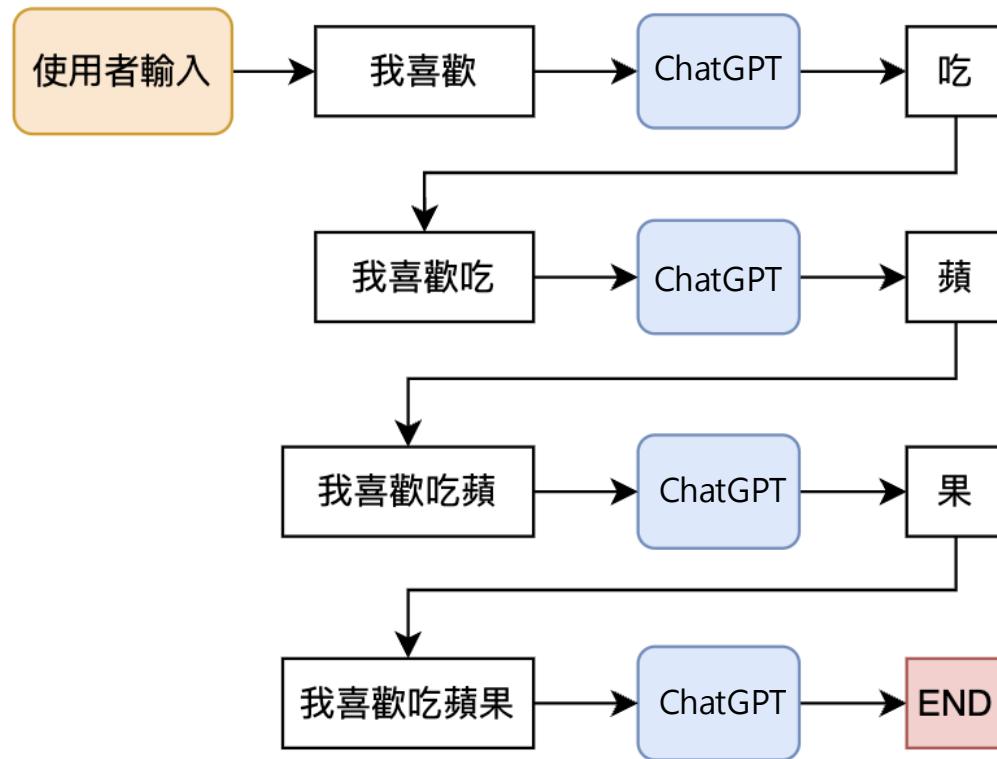
理解 AI 的特性 是善用 AI 的關鍵要素

每個生氣中的女人



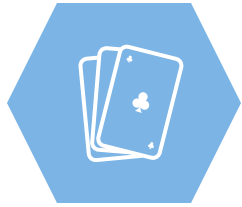
背後往往有一位
不知道自己做錯什麼的男人

文字生成模型是根據前一個字元或詞彙 來預測下一個字

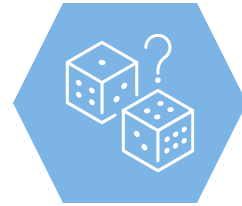


根據計算出的機率選擇最可能出現的字

理解生成式AI背後原理，會更能善用它

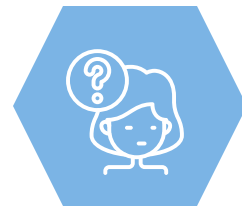


核心為「文字接龍」機率計算。



基於大量文字資料，計算每個字詞出現的機率。

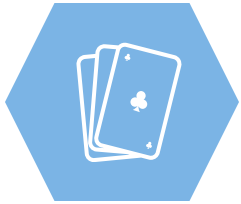
例：「香榭 __」，後面接大道的機率遠高於其他詞組。



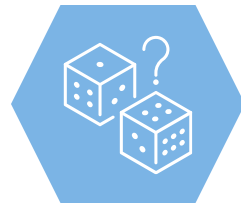
結論：透過猜測高機率字詞組合完成回答。

因此越多人討論、資料越多且不容易混淆的主題，LLM就越擅長

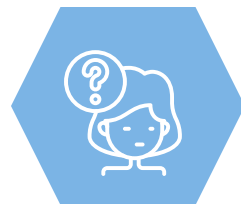
AI擅長哪些領域？



翻譯、文本生成與總結、常識、基礎知識



可以讓模型自己驗證回答的領域（例如寫程式、語意分析、情感分析、文本分類、語音識別與合成等）。這類任務可以用一些過去訓練好的深度學習模型，來訓練基礎模型。



大數據的整理、分析、視覺化，以及SEO

★ 麥肯錫預測，最主要受生成式AI影響的領域
會是在程式碼編寫、行銷與銷售、客戶服務以及產品研發。

生成式AI擅長什麼？

擅長項目	註記
受歡迎且被廣為討論的議題	對於被廣為討論的議題，若資料不會因時間而彼此衝突，且任務與你的商業領域知識無關時，生成式AI大多表現良好。
巨集類型的程式編寫	例如Excel巨集或者是Google Apps Script，這類邏輯性較少的流程自動化程式，生成式AI幾乎不會出錯
該領域的大概念與方向	生成式AI對於各領域的常識是擅長的，例如教科書內容，我們可以想像它是個通才，在各種領域都有至少入門60分的程度。
錯誤訊息的分析	很多時候錯誤訊息很長，真的很懶得看完，ChatGPT可以快速找到錯誤訊息的重點並且給予建議。



生成式AI不擅長什麼？

ChatGPT對於他不擅長的領域，不會說他不知道，而會一本正經地胡說八道！

不擅長項目	註記
同領域很多相似概念的狀況	例如在討論中國哲學時，AI可能不清楚「儒家」與「道家」兩者的具體區別，例如儒家的倫理重視與道家的自然無為。
更新速度很快的資訊	例如像是現在哪一個模型的Coding能力最強，這種可能現在是Claude、改天又變成OpenAI的模型，這類資訊就很容易搞混
冷門或較少人討論的項目	對於該領域較困難、較少人討論的議題，生成式AI回答錯誤的機率會提高很多。
和業務緊密相關的邏輯	零售業的客戶消費行為分析或旅遊業的訂票規則，生成式AI可能因缺乏上下文理解，無法完整反映實際需求。



SEO公司如何善用AI擅長的領域



Jemmy Ko

10月31日下午7:16 · 🌐

SEO專家

【頂尖內容公司，運用 AI 內容的 5 大技巧】

— 有家公司灌 200 篇文章到官網、撈進一堆流量

我這幾天飛新加坡，花了好幾萬買門票，參加研討會之後學到：

原來，有個公司用 AI 內容，灌了 221 篇文章到官網。

但這些不是亂灌，而是精心佈局過的——

它像編字典，從字母 A 到 Y，把「名詞解釋」一條一頁塞進去。

成果很神。

整批內容，有 927 個重點關鍵字排進 Google 前 3 名。

這些流量如果用買的，每個月要投超過 \$43,953 美金的廣告費。

這家公司叫「Ahrefs」，是國際知名的內容權威公司。

1. 名詞解釋

「名詞解釋」是客觀事實、有標準答案的知識，就這些東西不管真人來寫，或者機器來寫都差不多，AI 產出的版本甚至更精準、更好。

內容權威公司 Ahrefs 說，有這些類似特性的內容

- 名詞解釋
- 有客觀答案的知識
- 指南、百科、說明文件

頂尖內容公司，運用 AI 內容的 5 大技巧

1. 灌入「名詞解釋」頁面
2. 關鍵字研究：列出「種子關鍵字」
3. 分析 AI 內建的資料庫
4. 判斷搜尋意圖
5. 翻譯、分析上下文



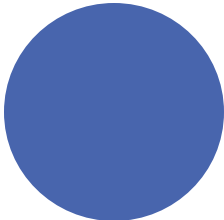
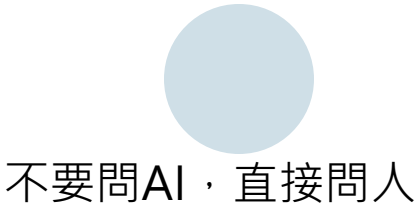
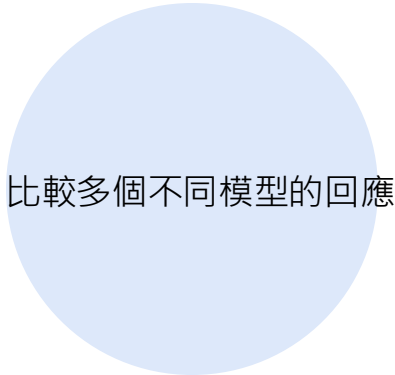
如何應對AI不擅長的領域

- 學習該領域的基礎知識，提高判斷回應正確性的能力
- 對於冷門知識，可善用Felo Search、Perplexity等搜尋式AI，以及Google等傳統搜尋引擎，針對回應做查驗
- 比對多個不同模型的回應，如Gemini、ChatGPT、Claude、Grok，若多個模型回答一致，則可信度高。
- 針對更迭頻繁的最新技術，可積極參與該領域的社群活動，獲得最新的第一手消息
- 直接問AI你確定是這樣嗎？請他再檢查一下



辨識LLM回應正確性的方法

回覆難以驗證



回覆易於驗證



AI擅長領域

AI不擅長領域



認識AI，學習與AI一同工作

AI 在程式開發中的應用範圍廣泛，除了撰寫測試、初始專案內容與程式碼審查外，還能在程式碼生成、除錯、測試自動化、文件生成、預測分析、使用者體驗設計等方面提供強大的支持。這些功能不僅提升了開發效率，還改善了程式碼品質與專案管理的整體效能。

GitHub Copilot

> 已使用 1 個參考

manifest.json

新增 manifest.json 檔案，設定 Chrome Extension 的基本資訊和權限。

manifest.json +32 -1

popup.html

新增 popup.html 檔案，設定彈出視窗的基本結構和樣式。

popup.html +23 -1

popup.js

新增 popup.js 檔案，實現按鈕點擊事件和訊息傳遞。

popup.js +10 -1

工作集 (8 個檔案)

接受

捨棄



background.js

content.js

manifest.json

options.html

options.js

popup.html

+ 新增檔案...

編輯工作區中的檔案



GPT 4o



AI在軟體開發中擅長什麼？

各種場景的程式碼生成



HTML/CSS/JavaScript

生成網頁的結構、樣式和互動邏輯，特別適合處理響應式設計與動態效果



框架程式碼

如 React、Vue、Angular 的元件範例與應用模板



API 開發程式

RESTful API 的端點邏輯與數據處理，支援 Express、Django、Flask 等框架。



資料庫操作程式

生成 SQL 查詢語句或 ORM (如 Sequelize、SQLAlchemy) 的資料庫操作程式碼。

聊天 + ↻ ...

cochiachang
@workspace /explain
manifest.json:1-32

Workspace
已使用 2 個參考

這段 `manifest.json` 文件是用來配置一個 Chrome 擴展程序的。以下是對這段代碼的詳細解釋：

- `manifest_version`: 這個字段指定了 manifest 文件的版本，這裡使用的是版本 3，這是目前最新的版本，提供了更好的安全性和性能。
- `name`: 這是擴展程序的名稱，這裡命名為「網頁內容摘要」。
- `version`: 這是擴展程序的版本號，這裡設置為 1.0。
- `description`: 這是對擴展程序的簡短描述，這裡說明了這個擴展程序會呼叫 OPEN AI 的 API 來總結 Chrome 正在瀏覽的網頁的內容。
- `permissions`: 這個字段列出了擴展程序所...

有關 API 調用的細節？

詢問 Copilot

- JS options.js Current file
- JS background.js:24

@ 🗣️ 📄 GPT 4o ▶️

AI在軟體開發中擅長什麼？

加速新技術的學習與應用

- 學習與關鍵技術尋找
快速檢索與彙整最新的技術文檔、教學資源與最佳實踐
- 針對錯誤提供解釋與解決方案
在學習新技術時，遇到錯誤訊息或疑惑，可提供解釋與解決方案，減少摸索時間。
- 生成簡單應用範例
根據需求，AI可快速生成某種方法或技術的實際應用範例，例如演算法的基本實現或框架功能的簡單應用，幫助理解與實踐。
- 加速原型設計與創意實現
協助快速完成小型專案或原型，有助於實驗新技術的可行性

聊天 + ...

cochiachang

#selection /tests

Selection .copilot-test-instructions.md:1-23

Workspace

> 已使用 2 個參考

讓我們來設置 Mocha 測試框架，並新增一個 "hello world" 測試：

tests

JS helloWorld.test.js

套用變更

接著，我們需要在工作區中安裝 Mocha：

```
npm install --save-dev mocha
```

最後，您可以使用以下命令來運行測試：

```
npx mocha
```

cochiachang

/clear

詢問 Copilot

JS apiController.test.js Current file


GPT 4o

AI在軟體開發中擅長什麼？

簡化測試工作


- 生成測試案例
根據程式碼的邏輯與結構生成測試案例，減少開發時間提升覆蓋率。
- 設定測試環境
自動生成執行腳本並設定好測試環境
- 整合測試腳本
自動生成 API 測試腳本，適用於 Postman 或 pytest。

聊天 + ↻ ...

 cochiachang

請審查我剛剛選擇的這段代碼，並提供任何潛在的問題或改進建議。

JS userService.js:1-34

 GitHub Copilot

> 已使用 2 個參考

這段程式碼的主要功能是檢查使用者是否存在，如果不存在則創建一個新的使用者。以下是一些潛在的問題和改進建議：

- 錯誤處理：**在 `catch` 區塊中，當無法取得使用者個人資料時，應該考慮記錄更多的錯誤資訊以便於調試。
- 重複程式碼：**在 `try` 和 `catch` 區塊中都有 `this.db.User.create` 的呼叫，可以將這部分程式碼提取出來，減少重複。
- 日誌記錄：**在 `catch` 區塊中創建使用者後，應該記錄一條日誌以便於追蹤。
- 異步錯誤處理：**在 `catch` 區塊中，應該考慮如何處理異步錯誤，避免未處理的異步錯誤導致應用程式崩潰。

以下是改進後的程式碼：

```
// services/userService.js

class UserService {
  constructor(client, db) {
    this.client = client;
    this.db = db;
  }
}
```

詢問 Copilot

JS apiController.test.js Current file

@ 📎 🗣️ GPT 4o ↵

AI在軟體開發中擅長什麼？

程式碼重構與優化



提升可讀性

程式碼經過重構後，能明顯提升可讀性，讓開發者更容易理解，同時也提高程式碼的執行效率



結構優化

AI 工具協助開發者進程式碼重構，優化整體結構，使程式碼邏輯更清晰、更易於維護。



改進與建議

根據需求提供優化建議，包括演算法改進或方法簡化，協助簡化冗長的程式碼，提升程式碼品質。



AI在軟體開發中擅長什麼？

文件生成與維護

自動生成文檔

AI能自動生成程式碼文件與技術文檔

產生清晰的註解

根據程式碼內容生成清晰的註解與說明

API 文檔

自動生成符合 OpenAPI Swagger 規範的 API 文檔

架構圖與技術說明

根據專案結構或專案需求，生成架構圖或技術報告

AI在軟體開發中不擅長什麼？

高度依賴業務邏輯的程式碼

AI 的不足

AI無法深刻理解特定領域的業務邏輯或從程式中的上下文了解完整的邏輯。因此對於需要融合專業領域知識的項目容易出錯。例如財務計算、醫療數據處理等專業領域。

範例

財務計算系統中的稅務計算程式碼，根據不同國家的法律和政策，需要非常精確地處理稅率、扣除項目和優惠政策。AI可能無法全面理解這些規則，生成正確的程式碼。



AI在軟體開發中不擅長什麼？

管理複雜跨模組的依賴關係

AI 的不足

AI 難以掌握整個系統的全貌，若系統需要在多個模組間處理細緻依賴或整合，可能會忽略某些依賴或產生不一致的程式碼。

範例

開發一個大型電子商務平台，需整合庫存管理、訂單處理和支付系統。這些模組之間有多層依賴關係，若未正確管理，可能導致訂單重複扣款或庫存錯誤。



AI在軟體開發中不擅長什麼？

快速變更的技術或函式庫

AI 的不足

AI 的知識基於訓練數據，無法即時反映最新版本的變更。

影響

對於更新頻率極高的開源框架或函式庫，容易產生過時或無效的建議。

範例

一些快速變動中的框架如LangChain的版本更新可能引入新的 API 或棄用舊的功能，AI 生成的程式碼可能基於過時的版本，導致無法正常運作。



AI在軟體開發中不擅長什麼？

冷門語言的程式碼

AI 的不足

AI 的訓練數據中對較少使用的技術的支援有限。

影響

較少人使用的語言或框架（如 COBOL），可能生成不準確或不完整的程式碼。

範例

使用 COBOL 撰寫銀行交易處理系統，這種語言在現代開發中已不常見，AI 可能生成錯誤的語法或忽略重要的語言特性。





與 AI 協作前 你必須具備的能力

了解生成式AI擅長與不擅長的部分之後，我們必須學習如何和AI協作，以及避免被AI所生成具有漏洞的程式碼影響整個專案的可靠信

與 AI 協作的挑戰

- AI 可以幫助完成 70% 的工作，但最後的 30% 讓人非常挫折。它總是邁進一步，卻伴隨著新的 bug、問題等，讓進展退兩步。
- 因此，雖然 AI 工具能快速完成大部分的基礎工作，但剩下的細節調整與錯誤處理，仍需要人的智慧和經驗來解決，特別是當程式變得複雜時。



Peter Yang ✓

@petergyang

Honest reflections from coding with AI so far as a non-engineer:

It can get you 70% of the way there, but that last 30% is frustrating. It keeps taking one step forward and two steps backward with new bugs, issues, etc.

「AI 速度」的隱性成本 – 除錯



AI加速了實現過程，但資深工程師的專業知識確保程式碼是可維護且穩健的。

資深工程師在參考 AI 建議的同時，還會...

- 將程式碼重構為更小的模組
- 添加 AI 遺漏的邊界案例處理
- 優化類型定義和介面
- 質疑並優化AI給出的架構決策
- 增加全面的錯誤處理

他們使用AI工具來

- 快速做出產品POC，實現他們已經理解的想法
- 生成程式碼之後，有能力進行改進
- 探索已知問題的替代方法
- 自動化例行編碼任務



三種實用的AI輔助開發模式

初步生成程式碼

- 讓AI快速完成功能的雛形
- 手動審查並重構以提高模塊化
- 添加全面的錯誤處理
- 編寫徹底的測試
- 記錄關鍵決策



在開發初期需要快速實現功能原型時，這種模式能幫助節省時間，同時確保程式碼的品質。

持續對話細化功能

- 將大任務拆解為多個小任務
- 為每個不同的任務開始新的AI對話
- 保持上下文集中和最小化
- 頻繁審查和提交更改
- 保持緊密的反饋循環



適合在多任務或多模組協作開發中，確保每個部分的輸出都有快速回應與即時調整。

驗證與強化程式碼

- 使用AI進行初始代碼生成
- 手動審查所有關鍵路徑
- 對邊緣情況進行自動化測試
- 定期進行安全審計



適合用於需要高度可靠性與安全性的任務，例如金流、身份驗證或高流量系統。

三種模式應混合使用，確保產出的品質

範例情境：開發一個商品管理系統

目標

開發一個商品管理 API，實現：

- 商品的新增、查詢、更新、刪除（CRUD 功能）。
- 支援按類別篩選和分頁功能。
- 確保數據庫的安全性和 API 的穩定性。



初步生成程式碼

- 1 工程師向 AI 提問**

"用 Flask 幫我開發一個商品管理 API，包括 CRUD 功能，使用 SQLite 資料庫。"
- 2 AI 輸出**

基本的 CRUD 路由和資料庫模型。
。可快速運行的初步版本程式碼。
- 3 修正顯而易見的錯誤**

查看官方文件或手冊，或詢問AI來修正執行這個初步版本的錯誤，

快速生成完整的初稿，涵蓋基礎功能與框架。

持續對話細化功能

- 1 操作 1**

啟動一個新對話，請 AI 「新增一個功能，讓查詢 API 支援按類別篩選，並實現分頁功能。」

AI 輸出程式碼，新增 category 和 page 參數的處理邏輯。
- 2 操作 2**

發現篩選功能中的 SQL 語句未考慮 SQL 注入，提出改進需求「如何防止 SQL 注入？幫我修正這段程式碼。」AI 優化查詢語句，改用參數化查詢。

每個功能細化都單獨開啟新對話，保持上下文專注並即時反饋修正。

驗證與強化程式碼


- 1 手動審查核心邏輯**

對 AI 生成的核心邏輯進行**手動審查**，**檢查關鍵部分**。接著發現分頁邏輯在極端情況下出錯，手動修正。
- 2 生成單元測試**

啟動一個新對話，請 AI **生成單元測試**：“幫我撰寫測試案例，覆蓋新增、查詢、更新和刪除功能的所有主要場景和邊界情況。”接著手動補充 AI 未考慮到的邊界測試。
- 3 安全性檢查**

進行安全性檢查，針對以下部分進行強化：**確保 API 請求的驗證邏輯完整**（如正確的身份驗證）。確保資料庫的訪問權限不會導致敏感數據洩露。

對關鍵邏輯和安全性進行手動審查與測試，確保可靠性與穩定性。



與 AI 協作的必備能力 - 1

對 AI 工具有效提問的能力

- 應能夠清晰描述需求或問題的上下文。
- 使用具體的語言與範圍限制，例如：「如何在 React 中實現搜尋功能？」。
- 熟悉如何使用指令 (Prompts) 來引導 AI 提供更準確的答案。

評估與驗證能力


- 能閱讀和理解 AI 生成的程式碼，確保其符合專案需求和最佳實踐。
- 熟悉單元測試、功能測試，能驗證程式碼是否運行正確。
- 判斷 AI 提供的資訊是否準確，尤其是涉及業務邏輯或冷門技術時，需額外檢查。

基礎技術與領域知識

- 技術基礎：如程式語言、框架、工具的基本知識 (例如 HTML、Python、Git 等) 。
- 業務知識：熟悉你的工作領域或專案背景，能辨識 AI 輸出是否符合需求。

問題分解與拆解能力

- 能將大範圍的問題拆解成多個小問題。
- 逐步詢問 AI，最終拼接成完整的解決方案。
- 清晰地描述需求與上下文，提出具體問題，避免模糊或過於開放的提問 (如「幫我寫個網站」) 。



與 AI 協作的必備能力 - 2

■ AI工具的操作與熟練度

- 熟悉主流 AI 協作工具 (如 : ChatGPT 、 Gemini 、 Claude 、 Perplexity) 。
- 能有效地使用程式AI工具 (如 : GitHub Copilot 、 Cursor 、 Windsurf) 進行開發 。

■ 持續優化的能力

- 熟悉如何給 AI 提供反饋並優化提問方式 。
- 善於從 AI 的多次回應中篩選出最有效的答案 。

■ 官方文件閱讀與查驗能力

- 熟悉使用搜尋引擎 (如 Google) 查找技術文件或範例 。
- 能快速理解和學習技術文檔或教學資源 。

■ 持續學習能力

- 需保持對 AI 技術的興趣與學習態度，掌握新工具與框架 。
- 在學習過程中共享經驗，促進整個團隊的成長與適應力 。

如何同時兼顧快速開發以及專業成長

1

使用AI進行快速原型設計

2

花時間理解生成的程式碼如何運作

3

在使用AI的同時學習新的技術或框架概念

4

在與AI的協作中建立知識基礎

5

將AI作為學習工具，而不僅僅是代碼生成器

需要耐心和熱情，與許多人最初希望通過使用AI工具自動寫程式的目標恰恰相反。



最關鍵的技能 – 在與AI的協作中學習

★ 注意到了嗎?有好多部分是新出現的, 對於原本工程師應有的技能也有更多的要求(例如單元測試、重構、敏捷方法)

對AI工具有效提問的能力

AI工具的操作與熟練度

評估與驗證能力

持續優化的能力



官方文件閱讀與查驗能力

基礎技術與領域知識

問題分解與拆解能力

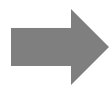
持續學習能力



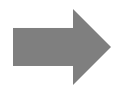
和 AI 協作 快速學習新技術

我是一位軟體工程師

遊戲



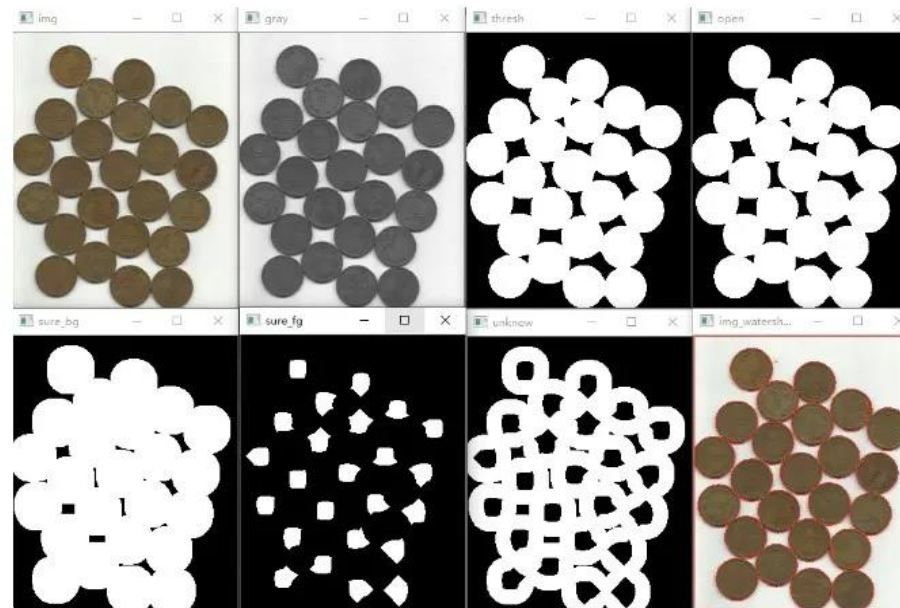
串流



串流影像偵測

- 2019年開始使用OpenCV進行影像辨識。
- 純粹的OpenCV對於環境光線變化的適應能力不佳，尤其是相黏、被部分遮蓋的物件，會嚴重影響偵測狀況。

公司開始想用ML來取代OpenCV



一直以來我對 AI 都很有興趣...

我覺得自己頗擅長歸納與邏輯思考，總覺得自己應該很適合走AI的路

第一次嘗試自己學習AI是在 2014年初...

結果：被一堆數學符號困住

全機率公式



$$\begin{aligned} P(S) &= 0.7 & P(H|S,R) &= 1 \\ P(R) &= 0.01 & P(H|^S,R) &= 0.9 \\ P(R|S) & & P(H|S,^R) &= 0.7 \\ &= 0.01 & P(H|^S,^R) &= 0.1 \\ P(R|H,S) &= 0.0142 \\ &= \frac{P(H|R,S) \cdot P(R)}{P(H|R,S) \cdot P(R) + P(H|^R,S) \cdot P(^R)} \end{aligned}$$

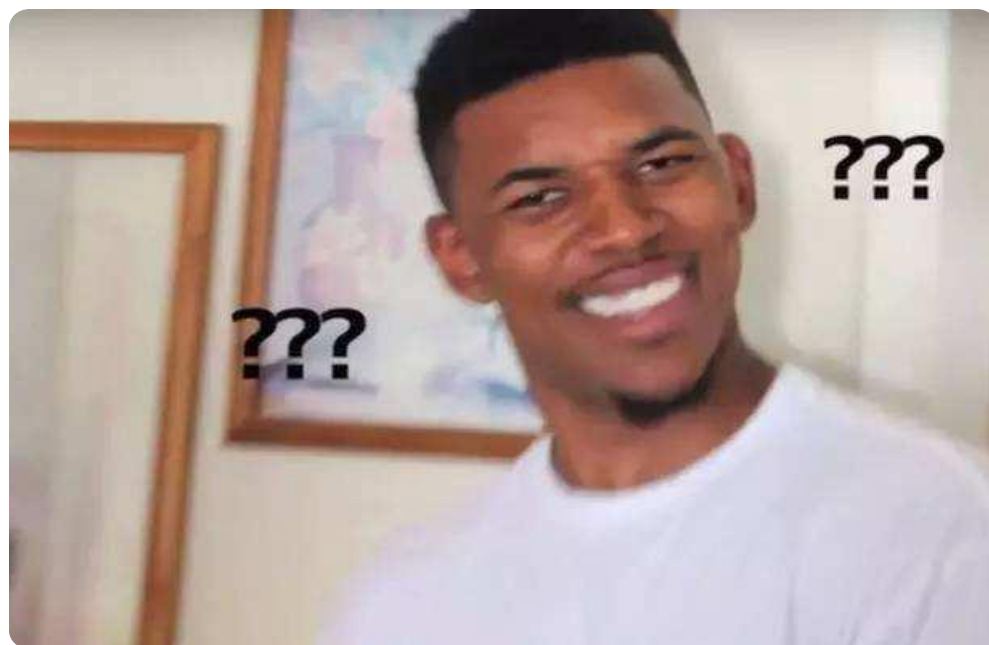
是在探討今天若心情很好 $P(H)$ 可能原因有天氣好 $P(S)$ 或是被加薪 $P(R)$
假使天氣好 $P(S)$ 的機率為0.7
被加薪 $P(R)$ 的機率為0.01

$P(H|S,R) = 1 \gg$ 天氣好且被加薪，心情好機率為1
 $P(H|^S,R) = 0.9 \gg$ 天氣不好但被加薪，心情好的機率為0.9
 $P(H|S,^R) = 0.7 \gg$ 天氣好但沒被加薪，心情好的機率為0.7
 $P(H|^S,^R) = 0.1 \gg$ 天氣不好又沒被加薪，心情好的機率為0.1

則求出 $P(R|H,S) >$ 也就是求出在心情好且天氣好的狀況下，是被加薪的機率

兩天後....

到底你的天氣、心情、加薪和我有甚麼關係啦！！！！



(崩潰)

$$\begin{aligned}
 P(R|S) &= 0.01 \\
 P(H|S, R) &= 0.7 \\
 P(H|\bar{S}, R) &= 0.1 \\
 P(R|H, S) &= 0.0142 \\
 &= \frac{P(H|R, S) \cdot P(R|S)}{P(H|R, S) \cdot P(R) + P(H|\bar{R}, S) \cdot P(\bar{R})}
 \end{aligned}$$

是在探討今天若心情很好(H)可能原因有天氣好(S)或是被加薪(P(R))
假使天氣好(S)的機率為0.7
被加薪(P(R))的機率為0.01

$P(H|S, R) = 1$ >> 天氣好且被加薪，心情好機率為1
 $P(H|\bar{S}, R) = 0.9$ >> 天氣不好但被加薪，心情好的機率為0.9
 $P(H|S, \bar{R}) = 0.7$ >> 天氣好但沒被加薪，心情好的機率為0.7
 $P(H|\bar{S}, \bar{R}) = 0.1$ >> 天氣不好又沒被加薪，心情好的機率為0.1

請幫忙解釋這一題的解法，一步步解釋

現在，同樣的問題我們可以詢問ChatGPT直到我們理解為止



這題是要解釋貝氏定理(Bayes' theorem)的應用，針對特定條件下的機率進行計算。目的是計算在「心情好(H)且天氣好(S)」的情況下，被加薪(R)的機率 $P(R|H, S)$ 。

以下逐步說明解法：

1. 已知條件與機率

根據題目，給出的條件是：

- $P(S) = 0.7$ (天氣好的機率)
- $P(R) = 0.01$ (被加薪的機率)
- 條件機率：
 - $P(H|S, R) = 1$ (天氣好且被加薪，心情好機率為1)
 - $P(H|\bar{S}, R) = 0.9$ (天氣不好但被加薪，心情好機率為0.9)

Message ChatGPT



貝氏定理 (Bayes' Theorem) 是一個在機率論中非常重要的公式，它的由來與條件機率的基本概念有關。貝氏定理的目的在於描述在已知某些條件發生後，更新對事件發生機率的看法。以下是它的來源與意義：

1. 條件機率的基本定義

條件機率 $P(A|B)$ 表示在事件 B 發生的前提下，事件 A 發生的機率。它的基本定義為：

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

也就是說， $P(A|B)$ 是 A 和 B 同時發生的機率 (聯合機率 $P(A \cap B)$)，除以事件 B 發生的機率 $P(B)$ 。

同理， $P(B|A)$ 也可以寫成：

$$P(B|A) = \frac{P(A \cap B)}{P(A)}$$

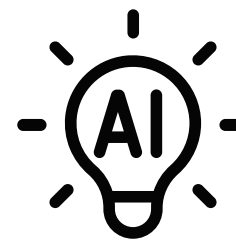
2. 聯合機率的對稱性

因為聯合機率 $P(A \cap B)$ 是事件 A 和事件 B 同時發生的機率，所以它可以用兩種方式表達：

$$P(A \cap B) \downarrow P(A|B) \cdot P(B)$$

Message ChatGPT





軟體工程師轉往AI建模之路
ChatGPT是我的學習救星！

對於不懂的概念，生成式AI可用白話解釋

User

請解釋深度學習裡的反向傳播與連鎖率

ChatGPT

反向傳播 (Backpropagation) 是一種在神經網絡中執行反向傳播演算法的方法，用於計算每個神經元的貢獻度，以及更新每個神經元的權重和偏差。

反向傳播的過程分為兩個步驟：

1. 計算損失函數的梯度：首先，我們會計算
2. 更新權重和偏差：接下來，我們會使用

連鎖率 (Chain rule) 是一種微積分中的定理，用於計算多元函數的偏微分。偏微分是指對單個變數求微分的過程，而忽略其他變數的變化。

User

解釋一下微積分的連鎖率的用處與價值

ChatGPT

連鎖率 (Chain rule) 是一種微積分中的定理，用於計算多元函數的偏微分。偏微分是指對單個變數求微分的過程，而忽略其他變數的變化。

例如，假設我們有一個二元函數 $f(x, y)$ ，那麼我們可以使用連鎖率來計算對 x 求偏微分的結果，即 $\partial f / \partial x$ 。連鎖率的公式是：

User

激活函數幹嘛用的

ChatGPT

激活函數 (Activation function) 是一種在神經網絡中使用的函數，用於決定一個

在神經網絡中，每個神經元都有一個權重和一個偏差。當神經元接收到輸入信號時，

激活函數的主要作用有以下幾點：

User

列出激活函數的可用參數，以及每個參數的特點和用處

ChatGPT

常見的激活函數可以分為兩類：非線性激活函數和線性激活函數。

非線性激活函數包括：

Sigmoid 函數：Sigmoid 函數是一種常見的激活函數，公式為 $f(x) = 1 / (1 + \exp(-x))$ 。

Tanh 函數：Tanh 函數是另一種常見的激活函數，公式為 $f(x) = (\exp(x) - \exp(-x)) / (\exp(x) + \exp(-x))$ 。

ReLU 函數：ReLU (Rectified Linear Unit) 函數是另一種常見的激活函數，公式為 $f(x) = \max(0, x)$ 。

線性激活函數包括：

User

Data augmentation是什麼

ChatGPT

數據增強是指在訓練機器學習模型時使用虛擬數據的過程。這些虛擬數據是通過對真實數據進

數據增強的目的是增加模型可用的訓練數據量，同時防止過擬合。過擬合是指模型對訓練數據

數據增強通常用於訓練計算機視覺模型，但也可以用於訓練其他類型的模型，如自然語言處理

可以從對話中針對我們專案的需求來學習

生成式AI可以針對任務給予學習方向

User
如何避免model誤分類沒看過的圖 (直接提出現在遇到的問題)

ChatGPT
有幾種方法可以幫助模型避免誤分類沒看過的圖像:

1. 增加訓練數據: 模型越看過的訓練數據越多, 就越能夠很好地辨識出未看過的圖像
2. 調整超參數: 調整超參數(如學習率、正則化系數等)可以改善模型的泛化能力
3. 使用不同的模型架構: 使用不同的模型架構(如卷積神經網絡、長短期記憶網絡等)
4. 使用集成學習方法: 集成學習方法(如隨機森林、梯度提升樹等)通過將多個模型
5. 使用數據增強: 數據增強是指通過改變圖像的亮度、尺寸、旋轉等來增加訓練數

User
如何設計資料集, 才能提高模型泛化能力 (進一步追問)

ChatGPT
如果你想要設計資料集以提高模型的泛化能力, 你可以考慮以下幾個方面:

1. 增加訓練數據的多樣性: 資料集中的樣本越多樣, 模型就有更多的機會學習到更多不同類型的圖像, 從而提高泛化能力
2. 增加訓練數據的數量: 越多的訓練數據可以幫助模型學習到更多的圖像特徵, 從而提高泛化能力。
3. 對訓練數據進行抽樣: 在抽取訓練數據時, 可以考慮使用分層抽樣(stratified sampling)的方法, 確保訓練數據中各
4. 使用數據增強: 數據增強是指通過改變圖像的亮度、尺寸、旋轉等來增加訓練數據的數量。這可以幫助模型學習更多的圖

User
對訓練數據進行抽樣怎麼實作 (進一步追問)

ChatGPT
在 Python 中, 可以使用 `scikit-learn` 中的 `train_test_split` 函數對訓練數據進行抽樣

例如, 假設你有一個包含 1000 個樣本的訓練數據集, 你希望將其分成兩個子集: 訓練子集和驗證

```
...  
from sklearn.model_selection import train_test_split
```

至scikit-learn官網或網路搜尋此函數使用範例

User
數據集量太多如何清理

ChatGPT
如果你的數據集量太大, 你可以考慮以下幾種方法來清理數據:

1. 刪除無用的特徵: 刪除不相關或者具有噪聲的特徵可以幫助減少數據集的大小, 同時也可以改善模型
2. 刪除重複的樣本: 在訓練數據集中如果有重複的樣本, 可以考慮刪除這些樣本, 以免造成訓練模型的
3. 刪除缺失值較多的樣本: 如果某些樣本中缺失值較多, 可以考慮刪除這些樣本, 以免造成模型學習的
4. 使用數據壓縮技術: 數據壓縮技術(如 PCA)可以幫助減少數據集中特徵的數量, 從而減小數據集的大

AI是一個很適合用來找【關鍵字】的工具

詢問一些抽象的方向與概念

User

我想在七萬多張的圖片數據集裡砍掉一些冗餘資料，篩選條件和方針怎設定才能增加訓練

ChatGPT

如果你想在七萬多張的圖片數據集中砍掉一些冗餘資料

1. 篩選條件應該基於數據品質：例如，如果有一些圖
2. 篩選方針應該基於訓練目標：例如，如果你的訓練
3. 篩選條件和方針應該基於數據平衡：例如，如果

根據

User

機器學習裡使用彩圖和灰度圖做資料集有何不同

ChatGPT

在機器學習中使用彩圖和灰度圖做資料集有以下幾

1. 維度不同：灰度圖只有一個維度（即灰度值）
2. 計算複雜度不同：使用彩圖做資料集時，模型
3. 訓練準確度不同：在某些情況下，使用彩圖做資料集可能會帶來更好的訓練準確

所以，在使用彩圖或灰度圖做資料集時，需要根據自己的需要和實際情況進行選擇。

User

當目標在畫面中被截掉一半時容易

ChatGPT

如果目標在畫面中被截掉一半，
那麼，在這種情況下，該怎麼辦呢？

有幾種可能的解決方案：

1. 對訓練資料進行增強：可以對訓練資料進行各種轉換（例如旋轉、翻轉、縮放等），以便模型能夠學習到
2. 增加訓練資料的數量：可以增加訓練資料的數量，以便模型能夠更好地學習到目標的各種特徵。
3. 調整模型的複雜度：可以嘗試調整模型的複雜度，以便模型能夠更好地處理輸入數據中只包含目標的一
4. 對輸入數據進行預處理：可以對輸入數據進行預處理，以便更好地保留輸入數據中的所有有效信息。

User

在資料集內把要辨識的目標以外的東西都留白是否有助於計算模型

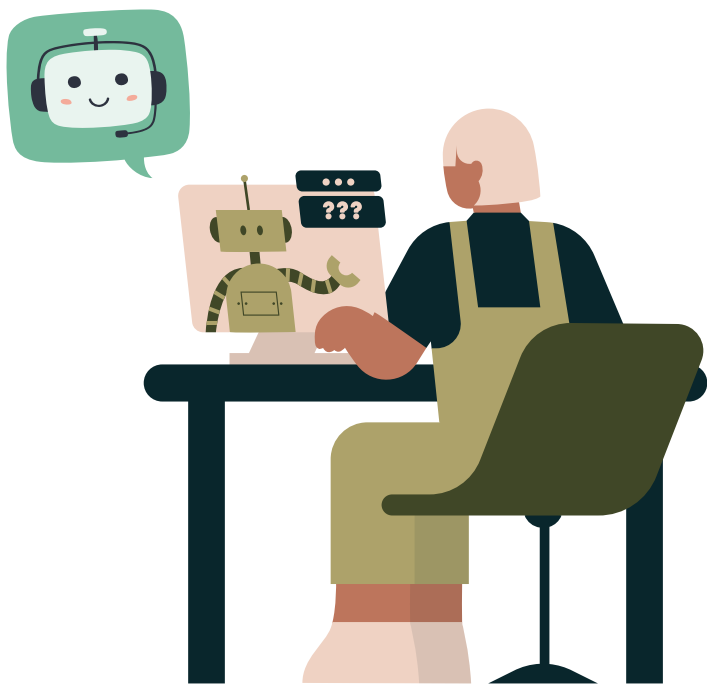
ChatGPT

在資料集中將非目標部分留白並不會對模型的訓練有明顯的幫助。
實際上，在訓練模型時，我們最好提供所有可能的資料，以便模型在訓練
有些人可能會擔心，如果資料集中包含了那些與目標無關的資訊，會不

透過同時詢問多個AI平台，可驗證這種概念性問題回覆的正確性

興趣是驅動AI學習的好幫手

那段時間真的是每天和ChatGPT聊天到半夜



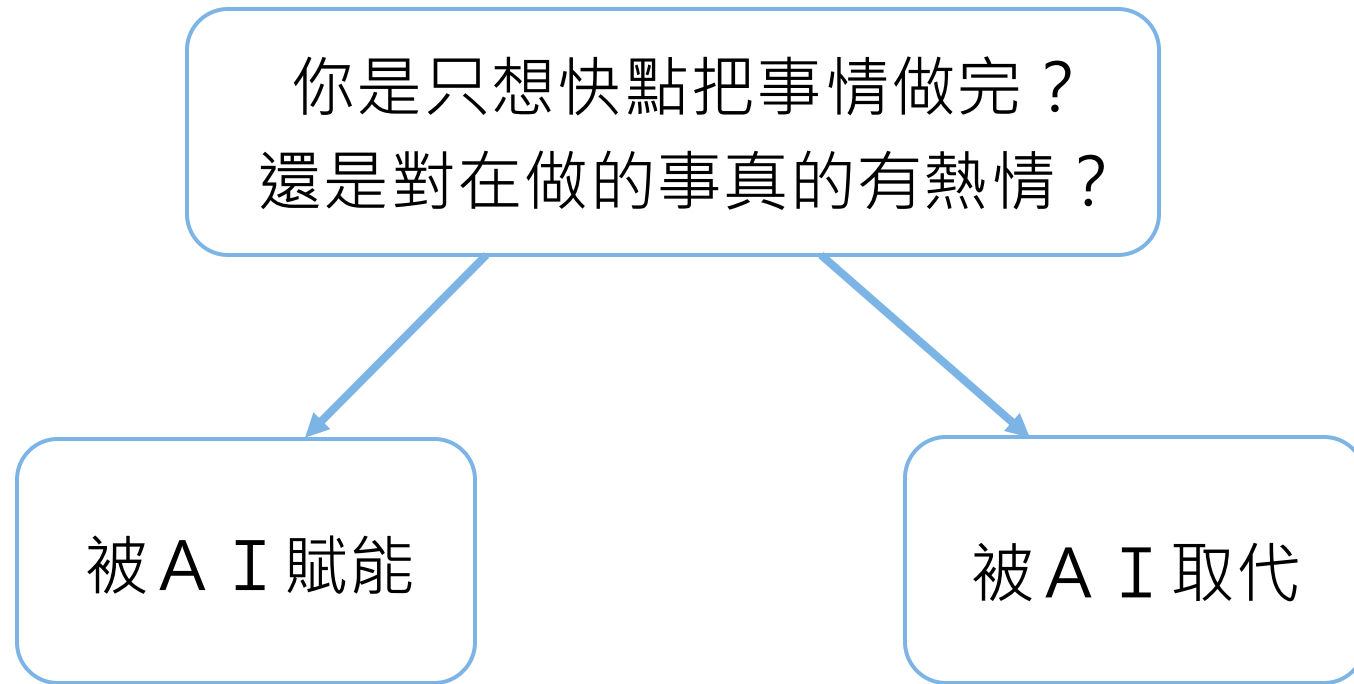
交叉熵誤差和均方誤差是什麼？

DNN、KNN、CNN、SVM有啥不同、應用在哪？

要怎麼知道是否有梯度消失的狀況？

真的就像是有一個專屬的老師隨時在旁邊幫我解惑，讓我快速通關、打通卡住我的所有問題！

興趣在 AI 賦能上，占據重要角色



我們是否有認真的去思考、理解AI的回應，並一步步與AI互動、深入探究。

這非常重要



這種學習方式有什麼好處？

可採最短路徑學習

以終為始，目的明確

在實戰中快速獲得大量經驗

即學即用

快速達成目的



在互動、問答的過程深化理解

這種學習方式有什麼缺乏？

缺乏學習環境

缺乏對領域系統性的了解

無管道取得最新資訊

無法認識共學夥伴

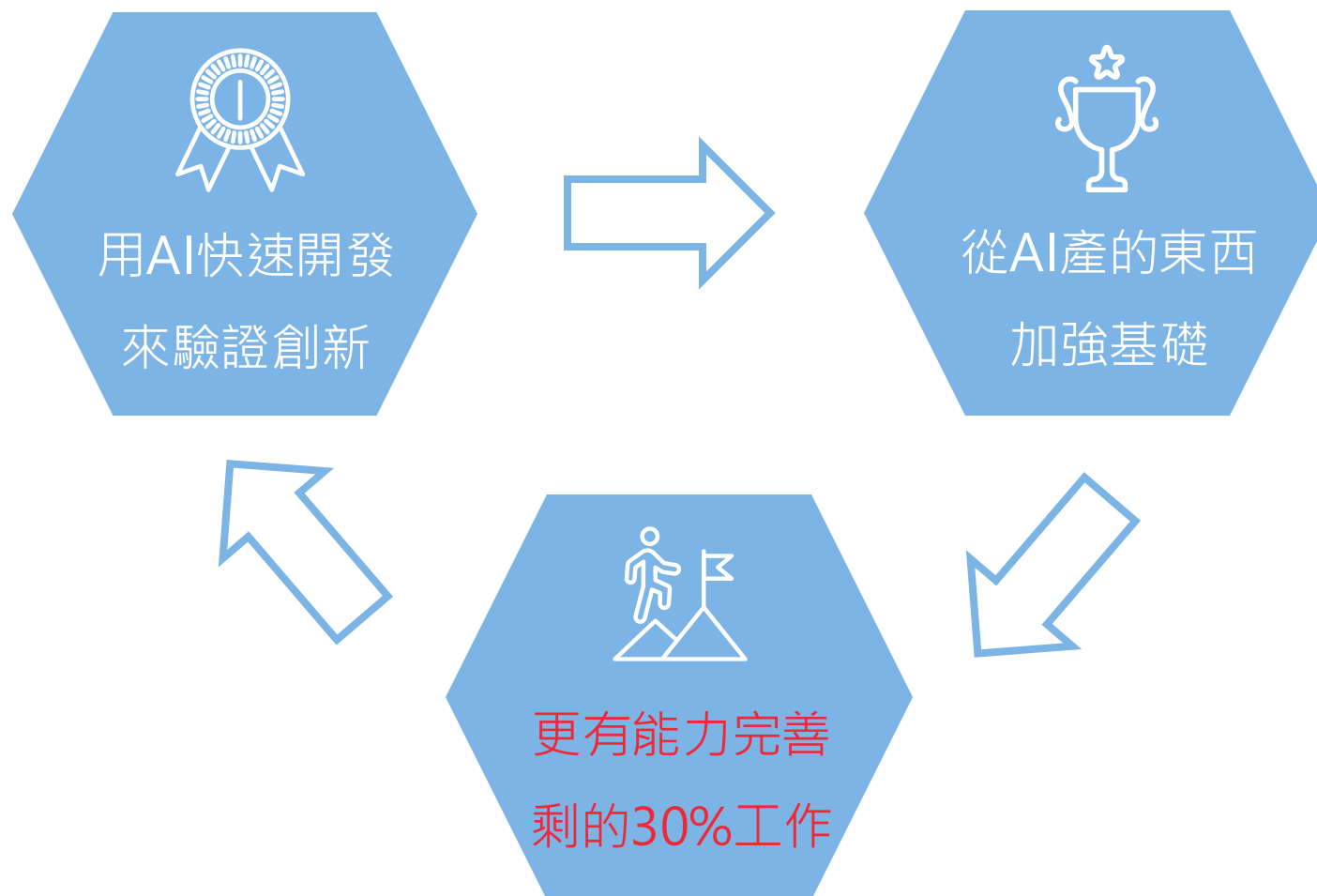


最終我還是報名了

 台灣人工智慧學校
Taiwan AI Academy

補齊系統知識
並踏進該領域的生態圈

懂得善用 AI 的工程師將脫穎而出



- 加速創意落地與原型開發，縮短從構想到實現的時間，幫助工程師更專注於核心創新。
- AI 不僅是工具，更是學習助力，協助工程師快速理解技術概念，並在解決問題的過程中提升專業能力。
- 有經驗的工程師能運用 AI 加速開發，同時憑藉專業知識處理細節（如錯誤處理、邊界案例與架構優化），最終產出穩定、可維護的程式碼。



如何與 AI 共同開發

初步生成程式碼

- 利用像 Bolt、v0 和「螢幕截圖轉程式碼 AI」這類工具，快速啟動專案的方式正在被徹底改變。
- 典型的工作流程：
 1. 從設計或初步概念開始
 2. 使用 AI 自動生成完整的初始程式碼基底
 3. 在數小時或數天內完成一個可運行的原型（而非數週時間）專
 4. 注於快速驗證和持續改進
- 使用 Bolt，可將 Figma 設計快速轉換成一個可運行的網頁應用，足以獲得初期用戶的反饋以驗證概念。

DEMO 工具： Bolt 網址：<https://bolt.new/>

使用REACT生成一個具購物車功能的商店網站，可將購物車列表生成EXCEL






Introducing bolt.new: Dev sandbox with AI from StackBlitz

What do you want to build?

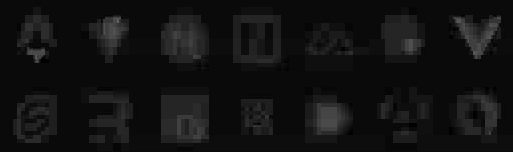
Prompt, run, edit, and deploy full-stack web apps.

How can Bolt help you today?



- [Start a blog with Astro](#)
- [Build a mobile app with NativeScript](#)
- [Create a docs site with Vitepress](#)
- [Scaffold UI with shadcn](#)
- [Draft a presentation with Slidv](#)
- [Code a video with Remotion](#)

or start a blank app with your favorite stack.



持續對話細化功能

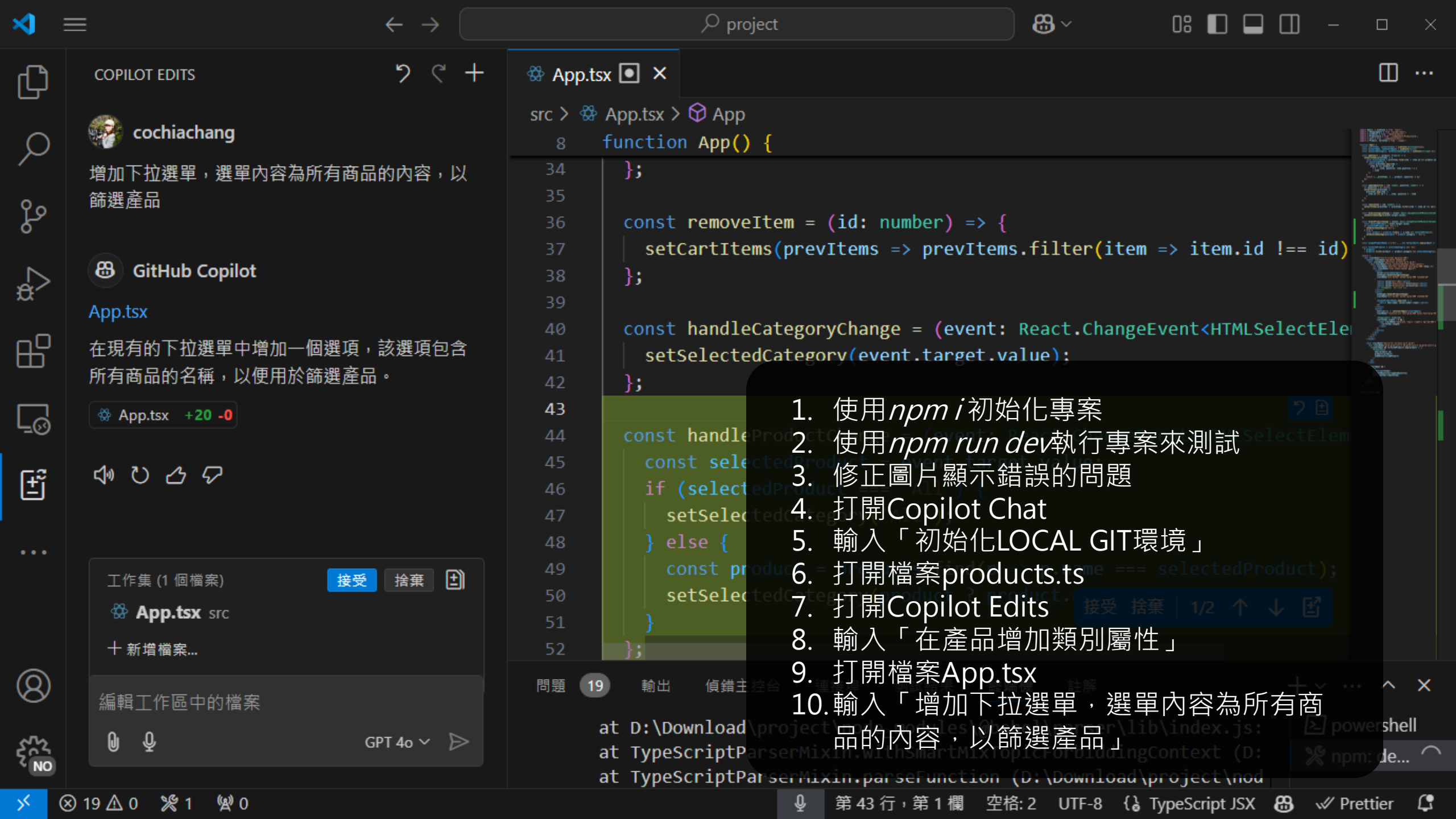
- 透過 Cursor、Cline、Copilot 和 WindSurf 等工具，AI 已成為日常開發工作流程中的核心。
- 開發者的工作方式：
 1. 使用 AI 提供程式碼自動完成與建議
 2. 借助 AI 完成複雜的重構任務
 3. 自動生成測試和文件
 4. 將 AI 作為「配對程式員」，共同解決問題
- 雖然不如原型開發（Prototype Development）那麼華麗，但對開發流程的影響更加深遠。

DEMO

工具：Github Copilot

- 在產品增加類別屬性
- 增加下拉選單，選單內容為所有商品的內容，以篩選產品





COPILOT EDITS

cochiachang

增加下拉選單，選單內容為所有商品的內容，以篩選產品

GitHub Copilot

App.tsx

在現有的下拉選單中增加一個選項，該選項包含所有商品的名稱，以使用於篩選產品。

App.tsx +20 -0

工作集 (1 個檔案)

接受

捨棄

App.tsx src

+ 新增檔案...

編輯工作區中的檔案

GPT 4o

App.tsx

src > App.tsx > App

```
8 function App() {
```

```
34 };
```

```
35
```

```
36 const removeItem = (id: number) => {
```

```
37   setCartItems(prevItems => prevItems.filter(item => item.id !== id)
```

```
38 };
```

```
39
```

```
40 const handleCategoryChange = (event: React.ChangeEvent<HTMLSelectEle
```

```
41   setSelectedCategory(event.target.value);
```

```
42 };
```

```
43
```

```
44 const handleProductChange = (event: React.ChangeEvent<HTMLSelectElem
```

```
45   const selectedProduct = event.target.value === selectedProduct);
```

```
46   if (selectedProduct !== event.target.value) {
```

```
47     setSelectedProduct(event.target.value);
```

```
48   } else {
```

```
49     const product = event.target.value === selectedProduct);
```

```
50     setSelectedProduct(event.target.value);
```

```
51   }
```

```
52 };
```

1. 使用 `npm i` 初始化專案
2. 使用 `npm run dev` 執行專案來測試
3. 修正圖片顯示錯誤的問題
4. 打開 Copilot Chat
5. 輸入「初始化 LOCAL GIT 環境」
6. 打開檔案 `products.ts`
7. 打開 Copilot Edits
8. 輸入「在產品增加類別屬性」
9. 打開檔案 `App.tsx`
10. 輸入「增加下拉選單，選單內容為所有商品的內容，以篩選產品」

問題 19

輸出

偵錯主控台

```
at D:\Download\project\node_modules\react-dom\lib\index.js:  
at TypeScriptParserMixin.withSmartMixingOfBiddingContext (D:  
at TypeScriptParserMixin.parseFunction (D:\Download\project\nod
```



第 43 行, 第 1 欄

空格: 2

UTF-8

TypeScript JSX



Prettier



驗證與強化程式碼

- 透過 Cursor、Cline、Copilot 和 WindSurf 等工具，AI 已成為日常開發工作流程中的核心。
- 確保生成的程式碼的品質：
 1. 將生成程式碼重構為更小、更專注的模組
 2. 添加 AI 遺漏的邊界情況處理
 3. 加強類型定義與介面設計
 4. 檢視並質疑架構決策的合理性
 5. 加入全面的錯誤處理機制
- 資深工程師透過多年積累的工程智慧，約束並完善 AI 的輸出，確保程式碼的可維護性和穩健性。
- AI 加速的是實作過程，但真正的價值來自於工程師的專業知識。

DEMO

工具：Github Copilot



聊天 + 🕒 ...

cochiachang

@workspace /setupTests 建立playwright的測試環境

package.json:1-13

Workspace

> 已使用 1 個參考

讓我們來建立一個 Playwright 測試環境，並新增一個 "hello world" 測試：

- tests
 - TS example.spec.ts
 - TS playwright.config.ts

套用變更

接著，我們需要在你的工作區安裝 Playwright：

```
npm install --save-dev @playwright/test
```

如何撰寫第一個測試？

詢問 Copilot

package.json Current file

@ 📎 🗣️

GPT 4o

package.json x

package.json > ...

```

1  {
2    "name": "vite-react-typescript-starter",
3    "private": true,
4    "version": "0.0.0",
5    "type": "module",
6    "scripts": {
7      "dev": "vite",
8      "build": "vite build",
9      "lint": "eslint .",
10     "preview": "vite preview"
11   },
12   "dependencies": {
13     "lucide-react": "0.344.0",

```

現在這個專案的邏輯拆分得不够好
需要重構之後才有辦法自動生成單元測試

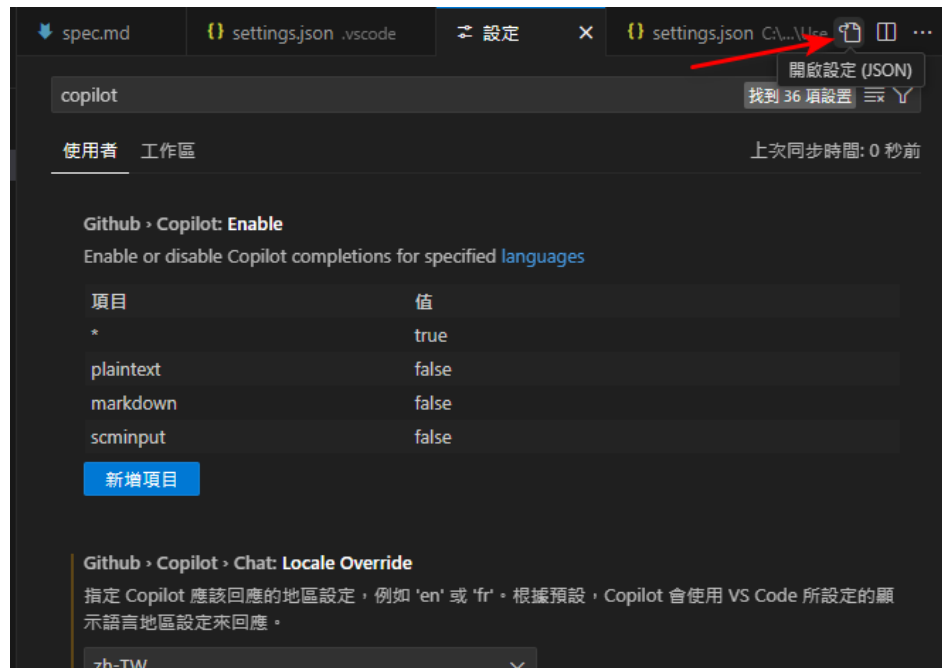
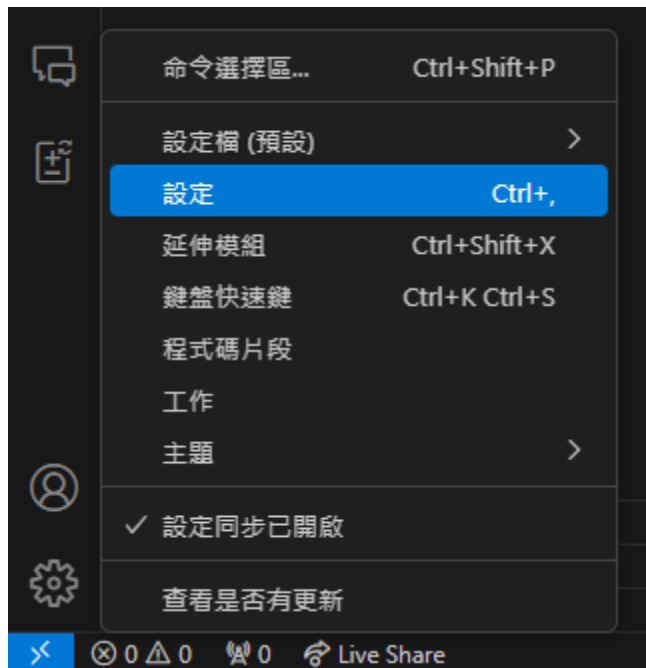
問題 1 輸出 偵錯主控台 更換 測試結果 縮編機 註釋

警告：PowerShell 偵測到您可能正在使用非正統的 PowerShell 執行環境，這可能導致某些命令已基於相容性而停用。請參閱 [PowerShell 執行環境](#) 以了解如何重新啟用它們。要重新啟用它們，請執行 'Import-Module PSReadLine'。

PS D:\Download\project>

1. Git commit儲存修改
2. 打開Copilot Chat
3. 輸入 /setupTests
4. 輸入「jest」
5. 輸入 #terminalLastCommand
6. 修改jest.config.js為export default {
7. npm install --save-dev jest @types/jest ts-jest

讓團隊使用Copilot時有更一致的 產出風格



<https://vscode.dev.org.tw/docs/copilot/copilot-customization>

DEMO 工具：Github Copilot



設定Copilot產生程式碼的規則

github.copilot.chat.codeGeneration.instructions

簡單範例

```
// GitHub Copilot Chat - 自訂提示
"github.copilot.chat.codeGeneration.useInstructionFiles": true,
"github.copilot.chat.codeGeneration.instructions": [
  {
    "text": "永遠用繁體中文回應"
  },
  {
    "file": ".copilot-instructions.md"
  }
],
```

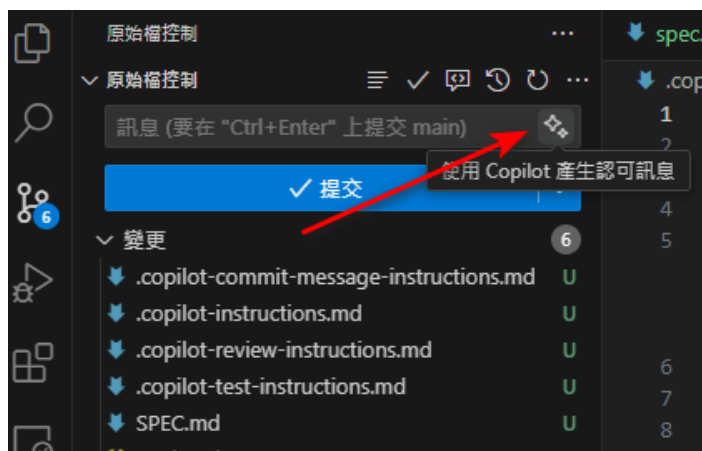
在 settings.json 檔案中定義，有以下兩種方式：

- 直接在設定中定義指示：使用 text 屬性來設定內容。
- 參考外部檔案：使用 file 屬性來指定檔案位置。

```
↓ .copilot-instructions.md > ## 目標
1  ## 目標
2
3  確保 GitHub Copilot Chat 生成的代碼符合以下標準：
4
5  - **一致性**：與項目現有代碼風格保持一致。
6  - **可讀性**：代碼清晰、易於理解和維護。
7  - **效率**：高效執行，資源利用合理。
8  - **安全性**：避免常見的安全漏洞和風險。
9  - **可測試性**：易於編寫單元測試和集成測試。
10
11 ## 代碼風格指導
12
13 ### 語言與框架
14
15 - **語言**：JavaScript (ES6+) 等。
16 - **框架**：React、Node.js、Express、Mocha 等。
17
18 ### 命名規則
19
20 - **變數與函數**：使用駝峰式命名法 (e.g., getUserData)。
21 - **類與組件**：使用帕斯卡命名法 (e.g., UserService)。
22 - **常量**：使用全大寫字母和下劃線分隔 (e.g., MAX_LIMIT)。
23
24 ### 格式化
25
```

自訂 Git Commit 訊息提示

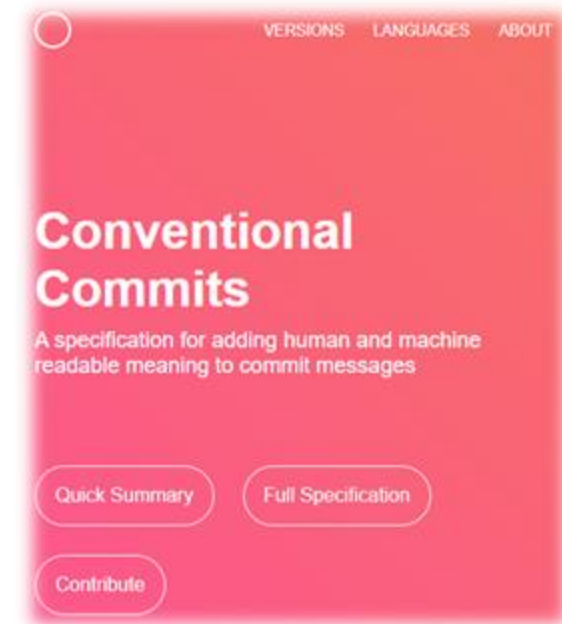
github.copilot.chat.commitMessageGeneration.instructions



規範Copilot在我們點選這個按鈕時，產生的Commit訊息的訊息格式

```
// GitHub Copilot Chat - 自訂 Git Commit 訊息提示
"github.copilot.chat.commitMessageGeneration.instructions": [
  {
    "file": ".copilot-commit-message-instructions.md"
  }
]
```

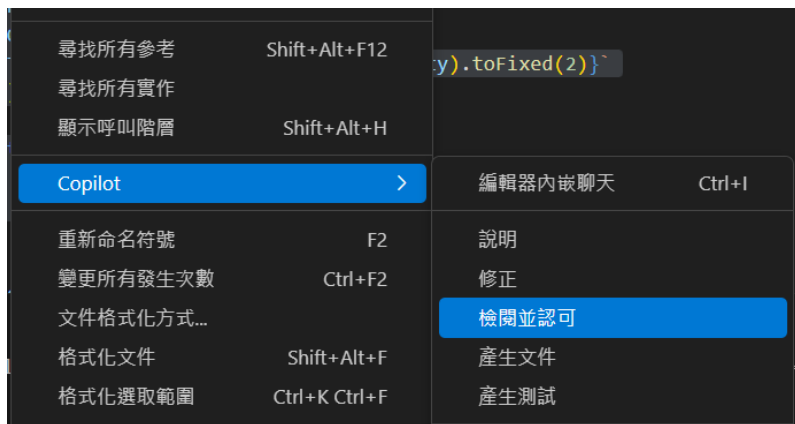
把這個設定寫在.vscode\settings.json
裡面並存到Git，可讓團隊有一致的
commit規範



這是範本所使用的規範
更多資料：
<https://www.conventionalcommits.org/zh-hant/v1.0.0/>

自訂程式碼審查提示

github.copilot.chat.reviewSelection.instructions



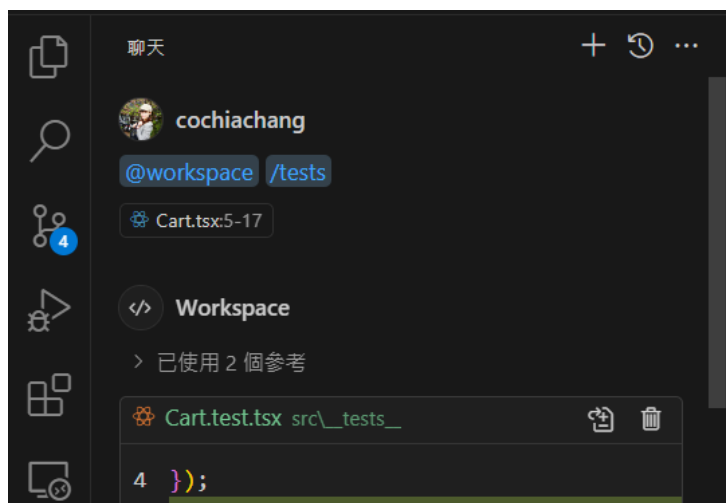
滑鼠右鍵 > Copilot > 檢閱並認可
(Review and Comment)

```
// GitHub Copilot Chat - 自訂程式碼審查提示
"github.copilot.chat.reviewSelection.instructions": [
  {
    "text": "永遠用繁體中文回應"
  },
  {
    "file": ".copilot-review-instructions.md"
  }
],
```

```
▼ .copilot-review-instructions.md > ...
1
2 ## **JavaScript Code Review 指南規範**
3
4 ### **1. 語法與格式 (Syntax and Formatting)**
5
6 1.1 **縮排**
7
8 - 使用 2 或 4 個空格進行縮排，避免使用制表符 (Tab)。
9 - 確保縮排一致，所有代碼層級清晰易讀。
10
```

測試相關設定

github.copilot.chat.testGeneration.instructions



在「聊天」中使用 @workspace /tests 提示，將依據這邊的設定來產生測試

```
// GitHub Copilot Chat - 測試相關設定
"github.copilot.chat.setupTests.enabled": true,
"github.copilot.chat.fixTestFailure.enabled": true,
"github.copilot.chat.generateTests.codeLens": true,
"github.copilot.chat.testGeneration.instructions": [
  {
    "file": ".copilot-test-instructions.md"
  },
  {
    "text": "一律使用繁體中文描述測試目標"
  }
],
```

在 settings.json 檔案中定義一組指示。若要在設定中直接定義指示，請設定 text 屬性。若要參考外部檔案，請設定 file 屬性



使用AI學習撰寫Chrome插件

我自己三小時從完全沒碰過Chrome插件，到完成一款可摘要網頁內容的插件的心路歷程

初始提示

開發一個簡單的 Chrome 瀏覽器插件，可以與網站互動，並提供一個簡單的 UI 讓使用者使用 OpenAI API。比如，設計一個插件，當使用者選中網頁內容時，點擊插件按鈕，可以用 OpenAI API 來生成該內容的摘要或評論。

因為對怎麼開發都不了解，自然下的指令就會很空泛，不太能產生正確結果

要怎麼debug

產出的結果有錯
詢問除錯方法



在開發 Chrome 插件時，進行有效的 debug 是確保功能正常運作的關鍵。以下是一些具體步驟和技巧，幫助你 debug 你的 Chrome 插件：

1. 使用 Chrome 開發者工具 (DevTools)

Chrome 提供了強大的開發者工具，可以幫助你檢查 JavaScript 錯誤、網路請求、DOM 結構等。

開啟插件的開發者工具：

- 在 Chrome 中，輸入 `chrome://extensions/` 並開啟擴充功能頁面。
- 找到你的插件，點選右上角的「背景頁面」或「服務工作者」連結來開啟開發者工具。
 - 如果是使用 `background.js`，選擇「背景頁面」。
 - 如果是使用 `service worker` (manifest v3)，選擇「服務工作者」。
- 這樣會打開 Chrome 的開發者工具，可以在「Console」面板查看錯誤訊息和日誌，並在「Network」面板檢查網路請求。

建議透過環境變數或更安全的方式來管理。

```
javascript Copy code  
  
// 在真正上線時，切勿直接把金鑰硬編碼在這裡！  
const OPENAI_API_KEY = "在這裡放你的 OpenAI API Key";  
  
chrome.runtime.onMessage.addListener(async (request, sender, sendResponse) => {  
  if (request.type === "OPENAI_REQUEST") {  
    const { text, action } = request.payload;
```

請介紹chrome extension 有哪些可用的執行單元



Chrome Extension (Chrome 擴充功能) 中，執行單元 (Execution Units) 指的是它內部不同的程式碼執行環境或模組。以下是 Chrome Extension 中主要的執行單元及其功能介紹：

1. Background Scripts (背景腳本)

- 用途：持續執行以監控事件 (如網頁載入、瀏覽器操作)。
- 特性：
 - 可存取全域的 Chrome API。
 - 可以與其他執行單元 (如內容腳本、Popup) 通信。
 - 類型：
 - **Persistent (常駐型)**：一直在記憶體中執行。

4. Options Page (設定頁面)

- 用途：提供擴充功能的設定介面。
- 特性：
 - 通常用於儲存使用者偏好。
 - 可存取 Chrome 的儲存 API。
- 範例應用：改變擴充功能的主題、設定選項。

發現有程式碼安全性警示

直接詢問AI來了解整個 Chrome Extension的大架構 有哪些，了解是否有直接可用的方式，讓不同的使用者儲存他自己專屬的API KEY

> 決定使用Options Page的模組來設定此變數

```
3 應產生以下檔案的檔案結構  
4  | - manifest.json  
5  | - popup.html  
6  | - popup.js  
7  | - content.js  
8  | - background.js  
9  | - options.html  
10 | - options.js  
11 | - icon.png
```

直接詢問錯誤原因

```
Unchecked runtime.lastError: Could not establish connection.  
Receiving end does not exist.  
Context  
popup.html  
Stack Trace  
popup.html:0 (anonymous function)
```



Thought about Chrome extension error for 12 seconds

當你在 `popup.html` 中執行 `chrome.tabs.sendMessage(...)` 出現

```
arduino Copy code  
  
Unchecked runtime.lastError: Could not establish connection. Receiving end does not exist.
```

通常代表以下幾種常見原因：

1. 該分頁沒有執行 content script

- 若 `contentScript.js` 並沒有被注入到當前分頁，就沒有「接收端」發送的訊息，所以就會跳出這個錯誤。
- 常見情況：
 - `manifest.json` 中的 `matches` 設置不正確，導致 content scri

檢查方向與解法

1. 確認 `contentScript.js` 是否真的有被注入

- 在 `manifest.json` 的 `content_scripts` 配置中，建議先用

```
json Copy code  
  
"matches": ["<all_urls>"]
```

進行測試，確保無論去哪個網站都能載入你的 content script。

- 或者如果只想在特定網域進行測試，請確保該網域（例如 `https://example.com/*`）跟目前開啟的網頁是相同的。



Thought about manifest and content script issues for a couple of seconds

在新版 Manifest V3 中，**content script** 不是在擴充功能的背景頁 (Background / Service Worker) 裡執行，而是直接注入到「網頁」本身。因此，你不會在擴充功能管理介面 (或 Service Worker 的 devtools) 中看到 content-script.js 的檔案或執行緒。

要查看 content script 的執行情況，你必須到「被注入的網頁」開啟開發者工具。以下是幾個重點說明：

1. content script 在網頁裡執行

- 當你在特定網頁 (如 <https://google.com>) 上開發者工具 (F12) 時，就能在 Sources 分頁的某個區塊 (通常會顯示「Content Scripts」或以你的擴充功能 ID 命名的檔案夾) 看到 content-script.js。

2. Service Worker 與 Content Script 分離

- Manifest V3 把背景程式改成「Service Worker」。它沒有傳統的「背景頁 (page)」介面，所以在擴充功能的管理介面中，你只能看到一個「Service Worker」窗口，顯示 background.js。
- content script 則是注入到瀏覽器可允許的網頁中，不會在這裡顯示。

3. 部分頁面無法注入 content script

- 如果你是在 <chrome://extensions/> 或 <chrome://settings/> 等 Chrome 系統頁面 (<chrome.google.com/webstore>) (Chrome Web Store) 等頁面測試，是無法注入 content script 的。
- 請確保你在一般可載入 script 的網頁上測試，例如一個普通的網站

開始學習要被注入到現在正在瀏覽的網頁的這個部分如何去除錯。

並開始注意到Manifest的設定似乎很關鍵

具體如何檢查看到 content script ?

- 在一般網頁上打開開發者工具
 - 前往任意普通網站 (例如 Google 首頁)，並在該網頁按下 F12 或右鍵 → 「檢查」。
- 切換到 Sources 分頁
 - 在左方或上方的檔案清單中，尋找一個像是 `chrome-extension://<擴充功能 ID>/content-script.js` 的項目，或是有個「Content scripts」的資料夾。
- 確認是否真的有被注入
 - 如果找不到 content-script.js，表示在該網頁並沒有成功注入。請再次確定你的擴充功能已安裝並啟用、以及頁面網址不在 Chrome 限制的範圍內。



Thought about debugging popup in Chrome extension for 6 seconds

要在 Chrome 擴充功能中除錯 (debug) `popup.html` ，可以透過以下幾種方式進行：

1. 直接在 Chrome 擴充功能管理頁面操作

- 打開 `chrome://extensions/` 進入擴充功能管理頁面。
- 找到你的擴充功能後，點選「Service Worker」或「背景頁」旁邊可能會有個「Inspect views: `popup.html`」連結（如果有顯示）。
- 有些情況會顯示一個「Inspect popup」按鈕或類似的連結，按下後就能開啟 DevTools 直接對 `popup.html` 進行除錯。

2. 從瀏覽器工具列直接開啟擴充功能的彈出視窗 (Popup)

- 在瀏覽器工具列中，點擊該擴充功能的圖示讓 `popup.html` 顯示出來。
- 在彈出視窗上按右鍵 → 選擇「檢查」(Inspect)，就能打開該彈出視窗的 DevTools。
- 這種方式最直接，可以看到 `popup.html` 中的 `console.log` 與網路請求等資訊。

3. 加上除錯訊息

- 在 `popup.js` 或 `popup.html` 裡加上 `console.log()` 等除錯訊息，然後透過上述方式打開 DevTools，即可查看 console 中的輸出。

4. 重新整理與快速測試

- 在開啟的 DevTools 裡，可以直接對 `popup.html` 做即時檢查與修改（包括 DOM 結構、樣式等等）。
- 如需重新載入整個 popup，可以關閉或切換頁籤，再次點選擴充功能圖示，即會再次載入



剛剛詢問嵌入網頁中的 `content.js` 的 debug 方式，仍然無法讓我看到 `popup.html` 的具體錯誤

因此詢問他怎麼 debug `popup.html`

這時，我已經學習到了三個 Chrome 的執行單元：

1. `background.js`: 在 Chrome 後面執行的 Service worker
2. `content.js`: 直接被注入現在正在閱讀的網頁原始碼裡面
3. `popup.html`: 點選插件圖示會出現的操作介面

Thought about chrome.runtime.sendMessage for a few seconds

在 Chrome 擴充功能的架構裡，`chrome.runtime.sendMessage` 是用來在擴充功能的不同部分之間（例如 popup、background script、content script 等）傳遞訊息的 API。以下是它的主要用途與運作流程說明：

1. 發送訊息：

- 你可以在 `popup.js`、`content script` 或其他任意能使用 `chrome.runtime` 的地方呼叫 `chrome.runtime.sendMessage()`，來發送一個訊息物件（通常包含 `type` 和 `payload`）給擴充功能的背景程式（`background / service worker`）。

2. 接收訊息：

- 在背景程式（`background.js` 或 `service_worker.js`）或其他有註冊 `chrome.runtime.onMessage.addListener(...)` 的腳本中，就能收到這個訊息並進行對應處理。
- 例如，你可以在背景程式裡 `chrome.runtime.onMessage.addListener((message, sender, sendResponse) => { ... })`，檢查 `message.type` 來判斷要做什麼事情（像是呼叫外部 API、存取瀏覽器的特殊功能等）。

3. 回傳結果：

- 在 `onMessage` 事件裡，你可以同步或非同步地做一些運算。
- 若是非同步操作，需要在事件處理函式的最末端 `return true;`，表示會在稍後再呼叫 `sendResponse(...)` 回傳最終結果。↓
- 呼叫 `sendResponse(...)` 就能把結果「回傳」給最初呼叫

開始透過AI工具學習

了解如何開發Chrome擴充功能

記得，這類大語言模型對於「教科書等級」知識的回答不太會出錯，他頂多會回答得不夠完整，但是不太會錯誤。

直接拿LLM的回應去交差是不行的，因為不夠完整。但透過頻繁的詢問、互動、深入探究，用LLM來學習基礎知識，是非常好的工具。

```
// 回傳 true 以表示會使用非同步的方式呼叫 sendResponse
return true;
}
});
```

這邊因為先傳回了// 回傳 true 以表示會使用非同步的方式呼叫 sendResponse return true; 導致popup.js的response失敗

學習認識錯誤訊息所代表的意義



Thought about async message handling in Chrome for 8 seconds

在 **Manifest V3** 的環境下，如果你在 `chrome.runtime.onMessage.addListener(...)` 裡面執行了 非同步操作 (像是 `await callOpenAI(...)`)，就必須在函式末尾 `return true;`，才能讓瀏覽器知道「我會在之後再用 `sendResponse(...)` 回傳結果」，否則 Chrome 會在事件處理結束後就丟棄這次的回傳管道，導致結果無法送回去。

不過，有時候我們在 **popup** 介面中使用 `chrome.runtime.sendMessage(...)` 呼叫背景程式，如果使用者 很快就關掉了 **popup** 或發生了其他原因 (例如延遲太久、被其他機制中斷)，可能會在回傳結果前就關閉了通訊管道，造成 `popup.js` 收不到回傳資料。這在 `console` 上常見的錯誤訊息是：

less

Copy code

`Unchecked runtime.lastError: The message port closed before a response was received.`

或類似提示。以下提供幾個排查方向：

所以讀取網頁內容要使用chrome.tabs還是chrome.runtime.



Thought about `chrome.tabs` vs `chrome.runtime` for reading page content for 4 seconds

在 **Manifest V3** 環境下，如果你想「讀取網頁內容」(例如 `document.body.innerText`)，有兩種常見做法，跟你用 `chrome.tabs` 還是 `chrome.runtime` 其實不是直接對應，重點在於你打算在哪個執行環境中讀取：

AI反覆給出不同卻都不能使用的修改建議，此時用詢問的方式和AI釐清這兩種方式的差異，學習了解這些方法的使用時機。

一、在 Content Script 中直接讀取

1. 把程式碼寫在 content-script.js

- content script 本身就執行在該網頁的 DOM 環境，能直接存取 `document`、`window` 等。
- 例如：

```
js Copy code  
  
// content-script.js  
const pageContent = document.body.innerText;  
// 接著送訊息給 background 或 popup  
chrome.runtime.sendMessage({  
  type: "PAGE_CONTENT",  
  data: pageContent  
});
```

篩選器

Manifest V3

Manifest V2

accessibilityFeatures

action

alarms

audio

bookmarks

browsingData

certificateProvider

commands

API

權限

資訊清單

translated by Google 本頁面由 Cloud Translation

首頁 > Docs > Chrome Extensions > Reference >

API 參考資料

多數擴充功能都需要存取一或多個 Chrome 擴充

常見的 Extensions API 功能

Extensions API 由包含執行擴充功能的方法和欄位。舉例來說，`chrome.action` 命名空間：

除非另有說明，否則擴充功能 API 中的方法為使用承諾取得這些非同步方法的結果。

AI開始鬼打牆...

根據剛剛和AI聊天所學習到的東西，去查看說明文件和網路教學

從文件的分類，可以知道在開發插件時，最重要的項目包括：

1. API
2. 權限
3. manifest.json的設定

接著發現是因為權限不足，沒有開啟activeTab的權限，導致一直取不到tab資訊

LLM在這邊最大的價值在於它大大的提升了我們的學習效率，我們不是要AI取代我們，而是要被它賦能

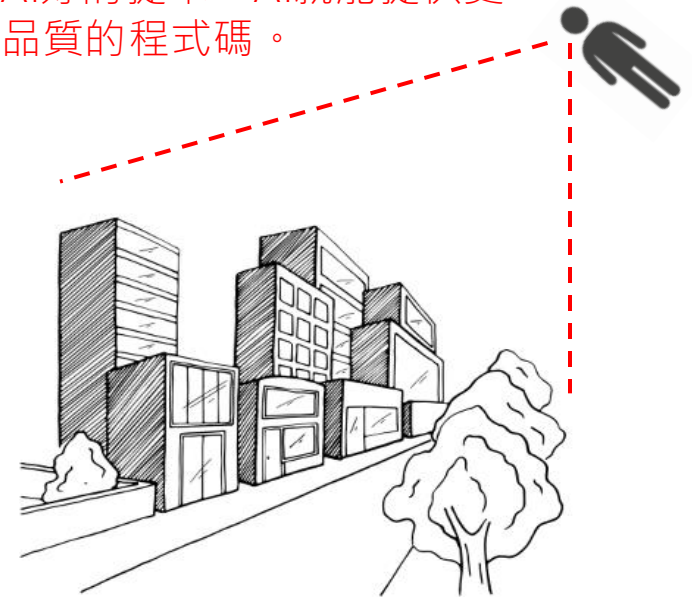
```
1 開發一款Chrome Extension，可以呼叫OPEN AI的API來總結Chrome正在瀏覽的網頁的內容
2
3 在Google Chrome 工具列中顯示擴充功能圖示，使用icon.png作為icon，default_popup為popup.html
4 options_page為options.html
5
6 這個插件一共需要要求下列權限
7 1. activeTab: 才能取得現在網頁的內容
8 2. contextMenus: 在CHROME插件工具列新增一個ICON，按下後彈出popup.html
9 3. storage: 儲存OPENAI API KEY
10 4. scripting: 在網頁內插入script以便將網頁內容回傳
11
12 在每個網頁增加content.js的內容
13 "content_scripts": [
14   {
15     "matches": ["<all_urls>"],
16     "js": ["content.js"]
17   }
18 ]
19
20 每個檔案的功能
21 1. popup.html - 寬度300px，標題為"網頁內容摘要"，下方有按鈕寫著"產生摘要"，摘要結果顯示於按鈕下方
22 產生摘要的流程如下
23 - sendMessage: REQUEST_PAGE_CONTENT
24 - 取得網頁內容後sendMessage: OPENAI_REQUEST取得摘要
25 - 顯示OPENAI回傳的摘要內容
26 1. content.js - 監聽REQUEST_PAGE_CONTENT事件，並且回傳現在頁面的內容(前五百字)
27 2. background.js - 聽OPENAI_REQUEST事件，確認應回傳true來保持連線，接著從storage取得儲存的API KEY，並呼叫
OpenAI API
28   requestBody = {
29     model: "gpt-3.5-turbo",
30     messages: [
31       {
32         role: "system",
33         content: "你是一位精通中文的 AI 助手。"
34       },
35       {
36         role: "user",
37         content: prompt
38       }
39     ],
```

重新撰寫規格書

要求應取得的權限
以及在那些位置插入content.js
的script的manifest.json設定

並且針對檔案來描述在該檔案應
該要有的功能

當我們用更高的視角看待專案，
給予AI好的提示，AI就能提供更
好的品質的程式碼。





未來的可能性
AI自己能夠操作電腦

由Claude所開發的
Computer use，已經可
以由AI自己來操作電腦



我是Alex，我在Anthropic擔任開發者關係主管。

想更多認識我嗎？

快加我的粉專和部落格來追蹤我吧！



F B



I G



B L O G

