《GitHub Copilot 學習手冊》好評推薦

《*GitHub Copilot 學習手冊*》是實用、有深度、貼近日常開發方法的一本書。

—— *Andrew Stellman*，開發者、團隊主管、講師與作者

每一位程式設計師都要瞭解 AI 輔助程式設計，GitHub Copilot 顯然是這個領域的龍頭，而 Brent 的這本書是帶你精通這款強大工具的終極指南。

—— *Tom Taulli*，《*AI-Assisted Programming*》作者

這本書帶你跨越「膚淺地使用 Copilot」與「真正發揮其潛能」之間的鴻溝。對於想要改善開發流程的工程師來說，本書是極為寶貴的資源。

—— *Balaji Dhamodharan*，資料科學領導者

這本書將帶領你開啟程式設計的未來。它透過清楚的課程與範例來幫助你熟悉 GitHub Copilot，不僅簡化整個程式開發流程，還讓程式更容易解讀與將它最佳化。無論你是剛接觸程式的新手，或是資深開發者，這本書都能讓你大幅提升效率。

—— *Aditya Goel*，*GenAI* 顧問暨全球銀行副總裁

《*GitHub Copilot 學習手冊*》詳盡地介紹了 GitHub Copilot 的功能與使用範例，其內容極具啟發性。

—— *Chhaya Methani*，*Microsoft*

碁峰 www.gotop.com.tw

序

我花了很多時間幫助開發者學習如何在日常工作中使用各種 AI 工具 —— 透過親自指導、舉辦培訓課程，以及在書中撰寫相關內容。當我自己設計真正的軟體時，也會依賴這些工具，其中有一樣工具對我而言特別重要：GitHub Copilot。

它是一款翻轉遊戲規則的工具，也是我的常備工具。

我已經撰寫程式數十年了，和每位開發者一樣，有時我也會卡住，可能是因為我想使用新技術或 API，卻害怕翻閱寫得很差的文件，可能是我要重構一段特別棘手的程式碼，而團隊每個人都避之唯恐不及，可能只是因為我不想寫那些非寫不可又枯燥乏味的樣板程式碼，但它將耗掉我生命中永遠逝去的半小時。

這就是 Copilot 的價值所在，或許它會給我完整的、可正確運作的程式碼，或許它只給我一個起點，但接下來還可以一起反覆修改。無論它給我什麼，對我來說都很有價值，而且它幾乎都能讓我繼續推動進度。

但使用 Copilot 也要負起責任。雖然 Copilot 能夠輸出可編譯、執行、甚至通過測試的程式碼，但如果你不瞭解它給你的東西，或不瞭解為什麼它能運作（更重要的是，你是怎麼讓它產生那個結果的），那麼，你就還沒有徹底掌握那些即將被併入 codebase（代碼庫）的內容，而這本書將帶領你掌控全局。

優秀的開發者不僅僅要學習 Copilot 的功能，還要學習與 AI 有效合作的方法，讓它在我們原本就在做的工作中支持我們，那些工作，就是撰寫程式碼、做決策，並且和他人一起解決問題。

碁峰 www.gotop.com.tw

這個學習旅程始於一個覺悟：Copilot 不是完美的，它也不想做到完美。它會根據程式碼的模式和背景資訊來提供建議，幫助你前進。而且很多時候，它的結果已經接近正確解且令人滿意了。

但是就和任何工具一樣，你必須知道如何引導它，以及懂得何時該停下來仔細檢查它提供的東西，這樣才能發揮它的最佳效果。

這就是出版這本書的重要原因。《GitHub Copilot 學習手冊》是一本實用、有深度，並根據日常開發方式來編寫的書。Brent Laster 完成一項出色的任務，因為他不只羅列了 Copilot 的功能，還示範了在真實的工作之中 —— 也就是在事情很混亂、非線性，且常常處於期限壓力下 —— 該如何使用這些功能。

他從 Copilot 的內在優勢談起，例如解釋程式碼和撰寫測試程式，然後逐步展示 Copilot 如何幫你解決令人頭痛的事情 —— SQL 查詢句、正規表示式、YAML 檔案，以及耗時的小任務（何況有時並不小！）。我很欣賞貫穿本書的範例，因為它們反映了開發者在真實生活中面對的問題，我想你也會欣賞它們。

Copilot 成為一款重要的日常開發工具的原因，並不是它承諾提供神奇的效果，而是因為它尊重開發者的思考方式。本書也是如此。它預設你會動腦思考、有能力，並且熱衷學習，它給你使用這項工具來做出好決策的空間，而非只是照單全收。

在我自己的教學和寫作過程中，我曾經看過開發者在獲得清楚、坦誠的指導後，進步的速度會有多快，如果可以在他們實際工作時，幫助他們做出更好決策的話，更是如此。這正是本書的目標，這是一本為想要掌控程式碼的開發者而寫的實用指南，它不會浮誇地吹噓神奇的功能，也不會過度簡化用法。

如果你是 Copilot 的新手，這本書是個很好的起點。如果你已經使用它一段時間了，你會學到讓它更有效、更可信的技巧。如果你一直抱持著懷疑的態度，那也很健康。這本書會在你所在的位置與你接軌，並告訴你如何充分運用這個工具，而不失去那些優秀開發者所具備的特質。

—— *Andrew Stellman*
開發者、團隊主管、講師，
《*Agile* 學習手冊》、《深入淺出 *C#*》
以及許多其他 *O'Reilly* 書籍的作者 美國
紐約布魯克林，*2025* 年 *4* 月

碁峰 www.gotop.com.tw

前言

風與浪總是與最熟練的航海者同在。

—— *Edward Gibbon*

我寫了很久的程式 —— 久到我幾乎不想承認 —— 我見證過軟體開發產業的好幾波風潮與變革（我也曾經在其中的幾次撰文或授課），從網際網路到 CI/CD、到容器，一直到雲端。這些典範的變遷，顛覆了過去的做法，催生了創新、令人驚嘆的工具，以及軟體建構方法的轉折。更廣泛來說，由於人們試圖理解這些轉變，並摸索如何調整與運用自己的技能，這些轉變也帶來不確定性與混亂。

現在我們迎來生成式 AI，它可以說是我們在科技領域中遇過的最大變革浪潮，尤其對軟體開發者而言。就像其他浪潮一樣，它帶來大量的創新，並推動巨大的改變。但在某些方面，它更像鋪天而來的海嘯：新工具、新模型與新功能展現出驚人的能力，將程式設計師捲入其中，讓他們努力逃離被淹沒的下場。

在這本書中，我將試圖幫助你乘風破浪 —— 至少是在使用 GitHub Copilot 這個強大 AI 工具的層面上。我不敢說它能回答你的所有問題，但希望它能讓你自在地面對這項工具、幫助你理解它的能力（從基礎到進階的主題），並啟發你讓程式設計任務比以往更簡單、更輕易。

我已經使用 Copilot 好幾年了，我曾經在個人專案中使用它，也曾經作為一位 DevOps 主管在企業研發組織中試用它、作為培訓講師在多家廠商中教授它、懷疑它，也曾經是它的熱情粉絲。最後的兩種角色的意思是，在 Copilot 問世的早期，我對它的印象並不

碁峰 www.gotop.com.tw

像其他工具那樣深刻。但是在寫這本書、設計相關課程，以及觀察其演進的過程中，它變成我在 IDE 和 GitHub 中常用的主力工具之一。Copilot 有了長足的進展，即使只看過去幾個月也是如此。其功能和特性都有令人印象深刻的進步。我盡力將它的功能和特性收錄在本書中。

我撰寫本書的目標之一是分享使用 Copilot 的好處與挑戰，並試著幫助你避免意外。因此，在你開始閱讀之前，有幾點需要注意：

- 在過去大約一年裡，GitHub Copilot 幾乎每次發布版本都會至少改變一個對話框或控制元件的外觀或位置。我已盡力在寫書時更新截圖與敘述了。但可以確定的是，當你讀到這裡時，其中的一些元件又會被移動、改變外觀…等。
- Copilot 會定期推出新功能。因此，隨著時間過去，一定有一些新功能未被收錄於本書中。同理，有些功能在筆者撰稿時被標記為 *preview* 或 *experimental*。在你看這本書時，那些功能可能已經納入正式版，或者，如果沒有成功通過，已被移除。
- Copilot 可以在許多不同的環境中運行，包括多個 IDE。在每個 IDE 中，整合方式、外觀甚至功能可能有所不同。GitHub 一直以來皆優先支援 Visual Studio Code 以及具備相同介面的環境（例如 GitHub Codespaces），以推出新功能，並涵蓋一套完整的功能。因此，本書的所有截圖與範例均來自 VS Code 或 Codespace。
- 對於不使用 VS Code IDE 的人來說，GitHub 在你的環境中可能尚未實作一些功能，請參考 Copilot 與 / 或 IDE 文件（*https://docs.github.com/en/copilot*）以取得最新資訊。
- 同樣地，能與 Copilot 整合的 IDE 通常可以在多個平台上執行，包括 Windows、Mac 和 Linux。由於本書無法涵蓋所有變化，所以書中的範例皆來自 Mac，畫面與控制項可能呈現 Mac 風格。此外，不同平台的鍵盤快捷鍵也有所不同。為了避免每次都列出所有快捷鍵，我們使用 *Meta* 來代表你的平台所使用的快捷鍵。

以上注意事項皆適用於本書的整體架構。本書的撰寫目的，是帶領你從回答**那是什麼**的基本知識一路到進階的**如何跳脫核心功能範疇**。大部分章節都能單獨閱讀，你可以根據需求與熟悉程度自由跳讀各章。

本書架構

既然你已經在看這本書了，我想，你應該很想要將 GitHub Copilot 當成 AI 程式助理，或者至少想要瞭解它和類似的程式助理如何幫助你更迅速、有效率地完成工作。

碁峰 www.gotop.com.tw

以下是本書架構的概要。

第 1、2、3 章將讓你掌握理解 Copilot 所需的基本知識，並學習如何在它的兩種主要模式 —— 程式碼補全與交談（對話）介面中進行互動。

第 4 章將探討與 Copilot 互動的進階方式，例如進行更自動化的編輯，以及使用 Agent 模式，從輸入提示詞開始，一路到自動完成修改。本章還將探討 Copilot Vision 的獨特介面，以及如何使用 Copilot 來 debug。

第 5 章和第 6 章將示範如何利用 Copilot 來建立豐富的測試程式和多種文件，為你簡化這些工作，讓你有更多時間專心編寫其他程式。

第 7 章將說明如何提升 Copilot 結果的即時性與相關性。

第 8 章將探討如何使用 Copilot 來處理一些比較不常見但實用的任務。

第 9 章將探討 Copilot 在 GitHub 的交談介面，以及它如何幫助你更瞭解專案，和協助你處理 GitHub Issues 與 pull requests。

最後，第 10 章將說明如何整合 GitHub Extensions 來加強 Copilot 的功能，並示範如何建立你自己的擴充套件。

你不需要預先知道關於 Copilot 的任何事情即可開始閱讀，但我假設你已經熟悉基本的程式編寫技巧，並具備 Git 與 GitHub 的基本知識了。下一節會進一步解釋哪些讀者可以從本書受益。

目標讀者

如果你想要深入瞭解如何使用 AI 助理來開發軟體，本書就是為你而寫的。沒錯，本書的重點是 GitHub Copilot，但許多範例與流程在其他 AI 編程工具與環境中同樣適用。為了從本書獲得最大效益，你必須曾經使用 IDE 來寫程式，以及具備 GitHub 的基本知識。要特別聲明的是，這本書不會教你寫程式、不會教你使用 IDE，也不會教你使用 GitHub。但它會幫你透過 Copilot 在這些領域中大幅提升生產力。不論你是軟體開發者、品質工程師、SRE，還是單純想要瞭解 Copilot 這類的 AI 助理如何幫助你，我相信你都能從中獲得寶貴的資訊與見解。

碁峰 www.gotop.com.tw

在我撰寫本書時，我認為能夠從中受益的讀者群包括：

- 對 AI 編碼助理還很陌生（或剛開始接觸），並希望瞭解它們是什麼，以及如何善加利用的人。
- 已經理解 AI 編碼助理的概念與流程，並想進一步瞭解 GitHub Copilot 之功能的人。
- 想要瞭解並評估 GitHub Copilot，有機會的話，想在組織或企業中大規模使用它的人。
- 已經有一些 GitHub Copilot 使用經驗，並希望充分發揮其功能的人。
- 想知道如何彌補 GitHub Copilot 的一些不足的人。
- 想要瞭解 Copilot 新功能（例如 Agent 模式）的人。
- 想要自己建立 GitHub Copilot 擴充套件的人。
- 已經在工作中使用 GitHub，想要利用它所整合的 Copilot 的人。

如果以上的任何一點符合你的情況，希望這本書能提供你所尋求的價值。如果你讀完本書，有機會的話，歡迎隨時透過評論或在未來的研討會與培訓場合中給予回饋。我們將在第 1 章開始介紹如何在編寫程式的過程中搭上生成式 AI 的浪潮，祝你衝浪愉快！

本書編排慣例

本書使用下列的編排方式：

斜體字（*Italic*）

代表新術語、URL、email 地址、檔名，與副檔名。中文以楷體表示。

定寬字（`Constant width`）

在長程式中使用，或是在文章中代表變數、函式名稱、資料庫、資料型態、環境變數、陳述式、關鍵字等程式元素。

定寬粗體字（**`Constant width bold`**）

代表應由使用者親自輸入的命令或其他文字。

定寬斜體字（*`Constant width italic`*）

應換成使用者提供的值的文字，或由上下文決定的值的文字。

碁峰 www.gotop.com.tw

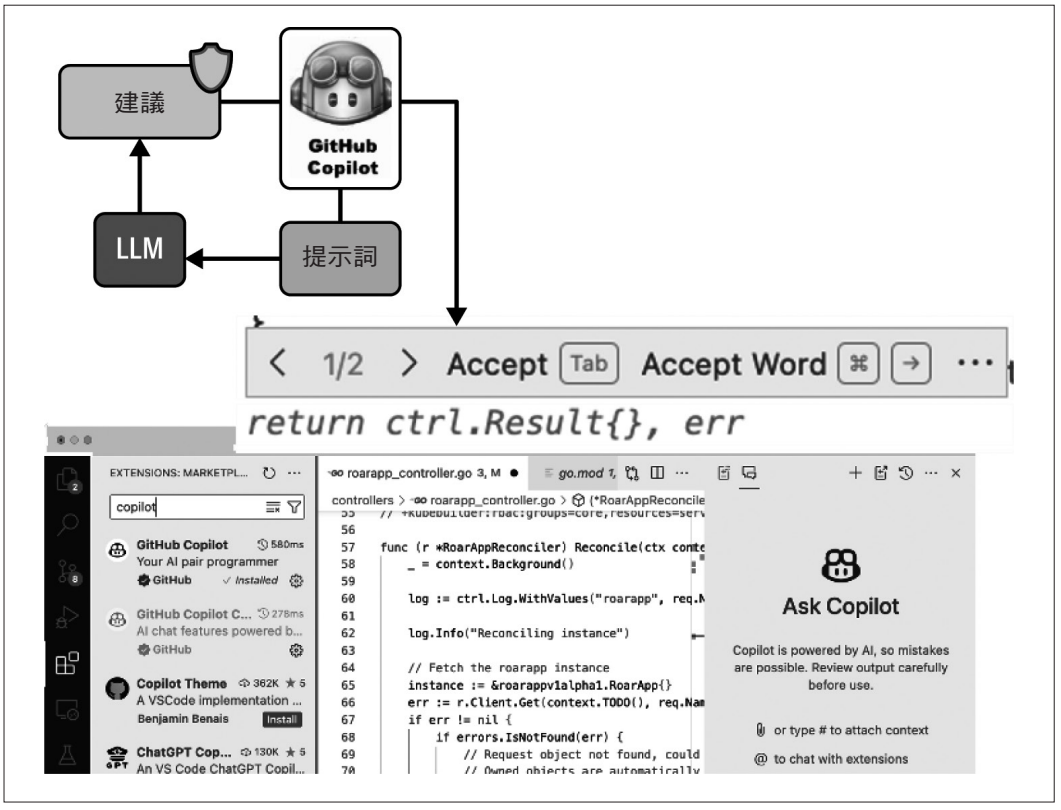

圖 1-4　從提示模型到生成建議的過程

後續的章節將討論如何使用 Copilot，你將更明白整個流程是怎麼運作的。但目前，我們要先瞭解利用 Copilot 來開發軟體時需要注意的一些事項。

使用時的注意事項

知道 Copilot 是什麼以及它的運作方式之後，有一件重要的事情要特別注意：Copilot 和任何 AI 一樣，可能提供錯誤或不完整的答案，它可能會出錯，或是生成令人意外的結果。雖然這種情況並不常見，但在使用時，你仍然要謹記一些重要事項，以便保持警覺。在本節中，我們將探討以下注意事項：

碁峰 www.gotop.com.tw

在預設情況下，使用者互動資料會被追蹤，可能包含匿名化的資料，不過具體細節會依你所使用的 Copilot 方案而有所不同（更多資訊請參閱 Trust Center〔*https://oreil.ly/rC8Rm*〕的文件）。有一些方法可以讓你在資料傳輸過程與資料處於靜止狀態時加密。GitHub 也有控制措施，嚴格限制誰能在他們那一邊存取資料。

GitHub 版本庫與法律問題

使用 GitHub 上的公開版本庫來訓練 LLM 的合法性在輿論與法院中都遭受質疑。這些問題就留給他人去解決，本書不會專門探討或評論在訓練初期的這些面向。

另一個常見的疑慮是，當 Copilot 使用「以授權版本庫訓練出來的模型」時，可能會產生與該 codebase 之內容極為相似的建議，這意味著使用者可能在不知情的情況下納入與之重複的程式碼，因而違反授權條款或智慧財產權。Copilot 有一個選項可讓個人或管理員封鎖這類符合公開程式碼的結果（圖 1-5 的上面那一個選項）。如果啟用該選項，Copilot 會提醒你，並濾除來自公開 codebase 的匹配程式碼。

為什麼要允許 matching content ？

相符的內容（*matching content*）是指 GitHub 版本庫中的程式碼與 Copilot 建議的程式碼完全或幾乎相符。你可能會好奇，「允許 matching content」這個選項有什麼價值？價值實際上不大，但或許你只想要研究如何完成某件事，而且不打算使用 matching content，或是打算在它是最佳做法時，於遵守授權條款的情況下使用它。

安全性

最後但同樣重要的是 Copilot 回傳的結果的安全性。安全性是任何產品或應用程式的重要關注點，而處理這個關注點的第一步就是寫出安全的程式碼。

之前說過，當 Copilot 從 LLM 收到結果之後，GitHub 會額外處理那些結果，額外處理的事情包括執行演算法來迅速檢查可能的漏洞。GitHub 不會完整地掃描結果是否安全，

碁峰 www.gotop.com.tw

因為那會耗掉太多時間，而是會快速掃描是否存在象徵漏洞的模式，或不安全的寫法，如果發現問題，它就會提出相關的建議。

即使有這些措施，我們仍無法保證所有問題都被一網打盡。由於這個處理方式不等於使用一個專門尋找漏洞的應用程式來執行完整的掃描，因此，你用來檢查其他程式碼是否安全的任何手段也要用在 Copilot 生成的結果上。

在使用生成式 *AI* 時，有一個明確且一直存在的要求：你必須隨時檢查並評估它回傳的任何建議，不該假設它完全且正確地解讀了背景資訊。除了最簡單的情況外，你不能假設結果是完美的。

換句話說，雖然 Copilot 經常被稱為 ***AI* 結對程式設計師**，但它並不像真正的人類結對程式設計師那樣，能夠理解與熟悉你的程式碼，把 Copilot 當成一位剛加入專案的熟練程式設計師比較恰當，這樣的程式設計師能根據觀察到的內容、以及他人分享的資訊，寫出有用的程式碼，但他無法掌握完整的背景資訊、專案歷史或背景。因此，你必須謹慎地確保他們產生的程式碼是正確、安全，並且適合合併的。你也應該以同樣的方式看待 Copilot 的答案與建議。

就像與新成員合作時一樣，你提供的細節越多（不論是作為背景資訊的程式碼，或具體指示詞），得到的結果通常就會越理想。在交談中給予 Copilot 更多的程式碼範例與更具體的提示，將大大提升 AI 的輸出品質。

當你開始使用或考慮使用 Copilot 這類的助手時，另一個問題可能會浮現：為什麼不直接使用 ChatGPT 或類似的交談機器人來產生程式碼呢？答案是：你當然可以。然而，這兩種 AI 的方法與介面有幾個關鍵的差異。下一節要簡單地比較它們，我們用 ChatGPT 來代表能夠產生程式碼的其他通用型 AI。

比較 Copilot 與 ChatGPT

你可能會好奇，Copilot 或類似的程式設計助手與 ChatGPT 或其他交談機器人有何不同，畢竟它們可能都使用同一個底層模型來回應你。一般來說，兩者的差異在於 Copilot 專注於程式設計領域，並提供針對此一領域的專門功能；而 ChatGPT 的範疇更廣，涵蓋任何領域，但無法提供相同程度的整合。表 1-1 展示幾個具體層面的差異。

碁峰 www.gotop.com.tw

第三章

在 IDE 中與 Copilot 交談

在 ChatGPT 這類受歡迎的 AI 平台裡，交談介面是你和 LLM 互動的入口。就像和人類交談一樣，與 AI 進行連續的對話讓人感覺比較自然，也能讓對話延伸下去，深入探討主題。

Copilot 的交談介面也有相同的優點，但它是專門為程式設計領域而打造的。我們在第 2 章討論過的 inline 功能可讓 Copilot 提供直接、快速、精準的程式碼建議。這些程式碼補全是根據你所處理的內容來生成的。但如果能以互動的方式自由討論程式設計會很有幫助，就像與同事討論程式設計挑戰或想法一樣。事實上，你很快就會難以想像如何使用 Copilot 而不使用它的交談功能。

本章將探討如何使用 Copilot 的交談功能（Copilot Chat）並與之互動。我們將介紹 Chat 的基本與進階面向。

我們會先帶你瞭解如何操作 Chat 的主要介面、如何解析它的輸出，以及如何管理多個 Chat 交談。

接下來，我們會簡單看看在 Chat 介面中如何進行提示，並探索各種不同的 Chat 介面。為了確保你在介面中正確使用提示詞，我們將檢視 Copilot 內建的各種捷徑與修飾器（modifiers），包括參與者（participants）與變數。我們也會探討如何在 IDE 的終端機中使用 Chat。

最後，我們會示範如何建立 Chat 的自訂指令，以便在產生回應時使用，並討論如何妥善處理 Chat 帶來的幻覺與不良回應。

碁峰 www.gotop.com.tw

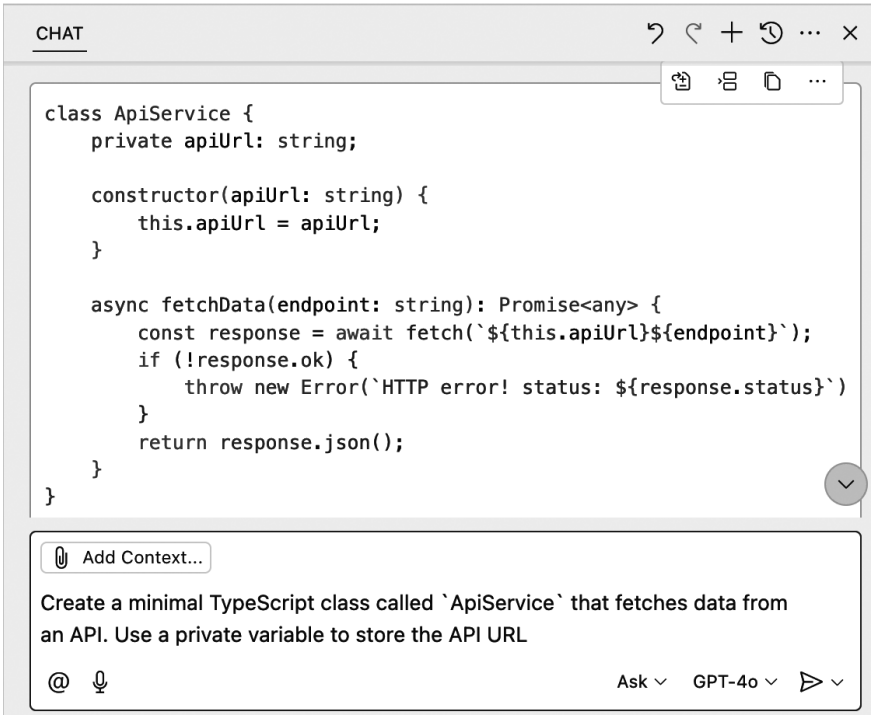

圖 3-41　沒有使用自訂程式碼生成指令的 TypeScript 簡單類別

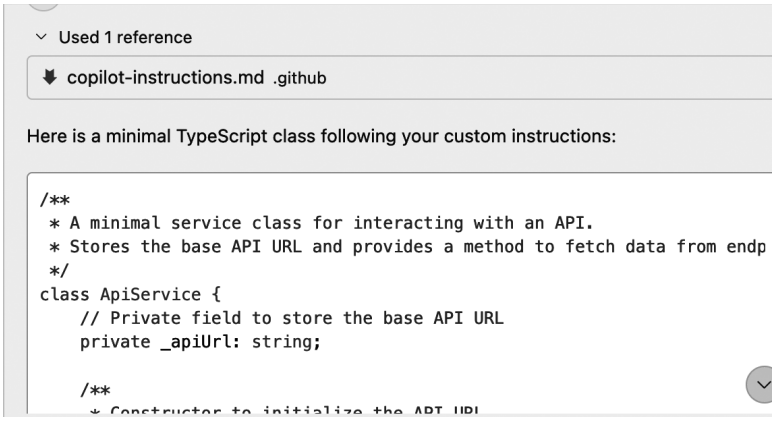

圖 3-42　使用自訂程式碼生成指令的 TypeScript 簡單類別

碁峰 www.gotop.com.tw

使用 Next Edit Suggestions 來做預測編輯

當你重構程式碼時，是不是經常在檔案中四處尋找需要修改的地方？Copilot 的 Next Edit Suggestions（NES）就是為了解決這樣的需求而設計的，它能簡化尋找與進行下一個修改的過程。

NES 可以根據你在程式中持續做了哪些修改、錯字和邏輯錯誤來預測並建議編輯。它建議的修改範圍可能從單一符號到多行程式碼不等。在重構的情境中，NES 能分析你正在進行的修改，並根據邏輯找出接下來需要修改的位置，再接著往下處理。

啟用 Next Edit Suggestions

在筆者撰稿時，你可以透過 VS Code 裡的一個彈出對話框來啟用 NES，你可以按下狀態列的 Copilot 控制項來進入該對話框（圖 4-1）。

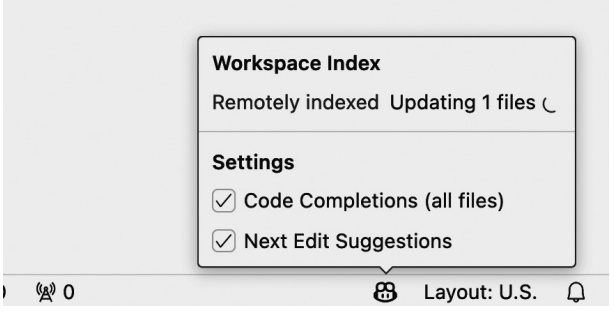

圖 4-1　使用 Copilot 控制項來啟用 NES

你也可以在 VS Code 的設定中直接啟用它，方式是設定 `github.copilot.nextEditSuggestions.enabled=true`。

啟用後，當你編輯程式時，NES 就會檢視你的程式碼，並嘗試預測還有哪些相關部分需要修改。NES 會在編輯器的 *gutter*（邊欄，在編輯器的行號左邊的空白區域）裡使用箭頭來突顯它的建議。舉例來說，假設我們用一個 Python 類別來表示二維座標點，它裡面有一個用來計算兩點距離的方法，並且附帶一個簡單的範例。程式碼可能長這樣：

碁峰 www.gotop.com.tw

比較 Copilot 的進階編寫功能

我們已經介紹幾種可以用來修改程式的方法了，表 4-1 能幫助你判斷什麼時候該使用哪一個方法。

表 4-1　比較 Copilot 的進階編輯功能

功能	說明	使用時機	使用情境範例
Next Edit Suggestions (NES)	在手動修改之後預測後續的編輯	• 簡單重構 • 編輯現有程式碼 • 在小規模更新中維持一致性	• 在 Python dataclass 中新增欄位 • 修正變數名稱裡的錯字 • 更新 API 端點 URL
Copilot Edits	透過自然語言提示詞與迭代檢查，來進行多檔案編輯	• 進行跨檔案修改 • 中等規模的重構任務	• 改變 React app 元件的名稱 • 更新設定格式 • 加入全域的錯誤處理機制
Copilot Agent 模式	能執行多步驟編程任務並自我修正的自主 AI agent	• 複雜的專案任務 • 全流程功能實作 • 初步嘗試問題解決方案 • 為程式碼建立額外介面	• 在既有程式中新增功能 • 從零開始建立 Flask web 應用程式 • 將 JavaScript 移植到 TypeScript • 試著針對問題產生修正方案

Copilot Vision

你是否曾經將 IDE 裡的問題（issue）截圖並傳給同事，因為這比親口解釋更有效率？或者，你是否曾經想要實作一個外觀或風格與看過的應用程式介面相似的版本？

若是如此，或是你預期將來可能遇到這種情況，Copilot 提供一種能幫助你的視覺功能。*Copilot Vision* 是一種進階功能，它能夠讓你在某些 IDE 的 Copilot Chat 中直接附加與處

碁峰 www.gotop.com.tw

圖 5-12　在主交談介面中使用 /tests 命令產生的測試程式

更改圖示與控制項

Copilot 經常更改各種介面使用的控制按鈕，有時也會重新排列位置。當你讀到這裡時，Copilot 很可能又做了新的調整。

一般來說，以下幾點在各種版本裡通常都會成立：

- 它會提供控制按鈕來讓你接受（保留）與拒絕（撤銷）變更。
- 打勾符號也是用來保留（keep）的控制按鈕。
- 它會提供一個控制按鈕來讓你在編輯器中以檔案形式查看建議的 diff，或將建議的 diff 套用至檔案。
- diff 控制按鈕可能會將檔案設為合理的名稱，也可能不會。務必在檢查並選擇要保留的變更後，自行儲存檔案。

這些回應涵蓋了基本的使用案例。但如果你希望生成更多測試，並涵蓋更多情況呢？你可以提供更明確的提示詞來做到，這就是下一節的主題。

碁峰 www.gotop.com.tw

第九章

在 GitHub 中使用 Copilot

GitHub 生態系統也和其他重要的軟體開發工具一樣，透過 Copilot 來整合 AI。這意味著你可以直接在 GitHub 中與 Copilot 交談，也能利用 AI 的功能直接處理諸如 pull request 和 issue 等核心功能。

在本章，我們將概述 Copilot 是怎麼被整合到 GitHub 的，並解釋在使用 GitHub 時，如何充分發揮它的功能。我們將討論以下內容：

- 在 GitHub 版本庫中使用 Copilot Chat
- 利用 Copilot 來變更流程
- Copilot 與 pull request
- Copilot 與 GitHub issues

GitHub 介面的演進

就像本書中的其他所有資訊一樣，本章的螢幕截圖，以及關於有哪些項目存在、它們位於何處的資訊，都是筆者撰稿時的最新狀態。就像 Copilot 被整合到 IDE 裡面的情況一樣，它被整合到 GitHub 裡的方式也可能會逐漸改變。

首先，我們來看看 Copilot 是怎麼和你的 GitHub 裡的版本庫整合在一起的。

碁峰 www.gotop.com.tw