以人工智慧（AI）輔助自學程式編碼：以Google Colaboratory整理Google Scholar資料為例

蔡明蓉 [1]、張維亙 [4]、蔡崇煌 [2、3、4]、陳鶴元 [1]
東海大學 教育研究所 [1]、高齡健康與運動科學學士學位學程 [2]、
共同學科暨通識教育中心 [3]、澄清醫院中港分院 家庭醫學科 [4]

摘要

目的

隨著人工智慧（Artificial Intelligence, AI）技術的發展，自學程式編碼（Coding）變得高效且便捷。本研究探討如何利用 AI 工具和 Google Colaboratory（Colab）平台輔助自學 Python 程式編碼，並以整理 Google Scholar 搜尋到的資料爲例做説明。

方法

以 Microsoft Copilot、DeepSeek、ChatGPT 及 Google Gemini 等 AI 模型交替使用，輔助 Python 編碼，於 Google 雲端硬碟 /Colab 平台執行程式。使用電腦型號爲 ASUS 筆記型 LAPTOP-TSDMU3MP，配備 Intel Core i7-1165G7 2.80GHz 處理器、運行 Windows 11。

結果

本研究探討 AI 輔助程式編碼應用的可行性，發現 AI 可作爲開發者的得力助手。透過 Colab 雲端運算能力，完全沒有基礎的初學者亦能快速掌握 Python 程式應用技巧。以整理 Google Scholar 學術資料爲例，發現給予 AI 提示詞，即可自動生成 Python 編碼，有些會有使用步驟、功能説明、程式碼説明、範例運行及注意事項，且會除錯及優化程式碼等，於 Colab 平臺，對於沒有 Python 程式語言基礎的初學者可輕易上手。再者透過不同的輸入條件，及於不同 AI，可產生不同的結果以資比較，可達到 Python 程式語言的自我學習。

結論

以 AI 輔助，可大大降低進入程式編碼的門檻，於 Colab 平臺可輕易上手執行 Python。因爲 AI 會有註解及程式碼解釋，因此也可達到初學程式語言者的自學參考。（澄清醫護管理雜誌 2025；21（3）：13-22）

關鍵詞：人工智慧、Google Colaboratory、Google Scholar、Python、自學

前言

人工智慧（Artificial Intelligence, AI）之 ChatGPT（聊天生成式預訓練轉換器，Chat Generative Pre-trained Transformer），是基於 GPT-3.5、GPT-4[1]、GPT-4o 及後來又陸續發表更新的版本 [2] 等架構，並以強化學習訓練的大型語言模型（Large Language Model, LLM）[3]，除了可用人類自然對話方式互動外，還有編寫和除錯（Debug）電腦程式碼的功能 [3,4]。之後微軟（Microsoft）公司也發佈 Windows Copilot，其亦有生成電腦程式碼的功能 [5]。於 2024 年底中國大陸發佈的 DeepSeek R1 後，因訓練成本低，但功能媲美 ChatGPT，因此引起西方資訊科技業及媒體的諸多討論及大震撼 [6,7]。蔡等人曾使用 Excel 配合其內建的 Visual Basic For Application（VBA）程式語言設計家庭檔案 [8]，發現 AI 具超強生成程式碼的能力，AI 技術的進步為自學程式編碼

通訊作者：蔡崇煌
通訊地址：台中市西屯區台灣大道四段 966 號
E-mail：m333834@yahoo.com.tw

受理日期：2025 年 2 月；接受刊載：2025 年 3 月

提供了新的可能性。有關程式語言的學習，過去我們主要是參考書本或使用 Google 查詢，目前有 ChatGPT、Microsoft Copilot、Gemini 及 DeepSeek 等等之後，使用更方便，其皆可依提示詞（Prompts）自動生成程式編碼。推估程式編碼將成為未來不可或缺的技能，然而對於初學而言，傳統的學習方式往往耗時且缺乏實踐機會，又自學程式編碼時常面臨的挑戰，如資源過載、技術門檻高等，本文探索 AI 如何降低這些障礙，及增強學習者的能力。Python 是目前最流行的程式語言，透過 Google Colaboratory（Colab）的免費雲端平台執行，不必再額外下載安裝，不需要編譯（Compile）成執行檔（.exe），對於初學語言者而言，是很友善的學習環境，且馬上可看到執行的成果，能夠顯著提升自學效率。有以上的優勢環境，因此我們嘗試以 AI 輔助程式編碼於如何利用人工智慧簡化與優化編碼學習過程，提供利用 Colab 整理 Google Scholar 資料的具體實例作說明。

研究方法

文獻題目以關鍵詞（“Colaboratory”OR“Colab”）AND“Google Scholar”搜尋 Google Scholar。以 DeepSeek 或其他 AI 及 Google 瀏覽器交替使用，輔助 Python 編碼，執行前先完成 DeepSeek（https://www.deepseek.com/）及 ChatGPT（https://chatgpt.com/）註冊，使用 Google Chrome 可直接進入 Gemini（https://gemini.google.com/app?hl=zh-TW），使用 Edge 瀏覽器即有內建 Copilot，不必額外再安裝，使用 Colab（https://colab.research.google.com/）需先到 Google 雲端硬碟設定。使用電腦型號為 ASUS 筆記型 LAPTOP-TSDMU3MP，配備 Intel Core i7-1165G7 2.80GHz 處理器、32GB RAM、運行 Windows 11、64 位元電腦。

本研究實施步驟如下，首先開啟 Google 雲端硬碟 / 直接讀取舊的檔案（.ipynb）或“+ 新增”/ 更多 /Google Colaboratory/ 出現 Colab Notebook/ 貼上程式碼。第一次使用，如果沒有出現 Colaboratory，則使用“+ 連結更多應用程式”加入 Colab，完成的結果如圖一。在 Colab 中建立一個新的 Notebook 後，執行以下步驟：

一、輸入提示詞到 AI，例如 DeepSeek 或其他 AI 獲取程式碼。

二、選取“+程式碼”/複製貼上程式碼/執行安裝開頭為“!pip install…”必要資料庫，此步驟也可附在其他程式碼上，如果沒有就可省略此步驟。

三、執行其他主程式。

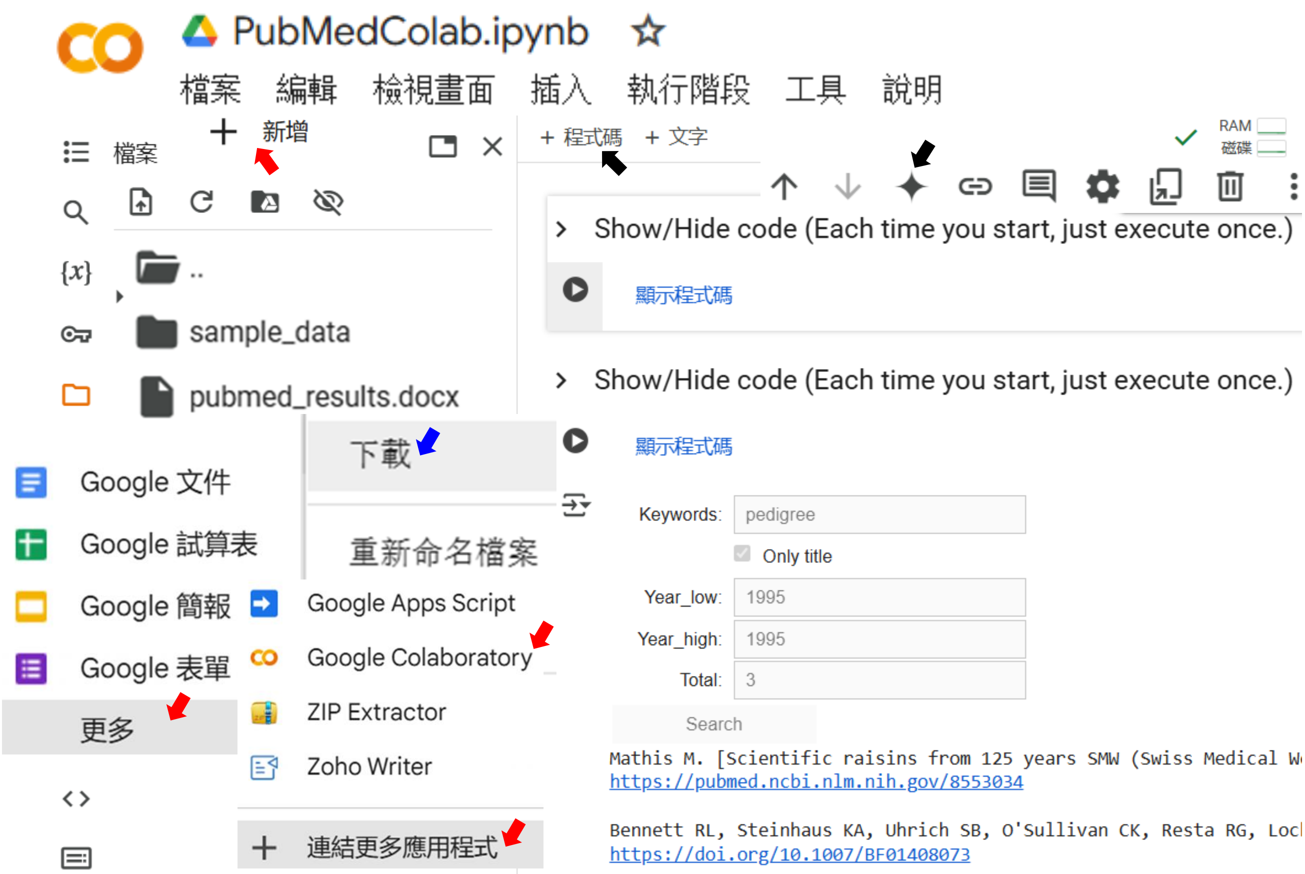
四、執行的結果可下載儲存，否則退出或電腦關機，資料即消失。

五、不符合需求的結果可再與AI溝通做修正，或自己做修改程式碼。

六、有不了解之處，可將程式碼片段輸入AI工具，獲取解釋和改進建議，或直接在Notebook使用“可用的AI功能（✦）”/說明程式碼，會出現程式碼的解釋。

結果

文獻題目以關鍵詞（“Colaboratory” OR“Colab”）可搜尋到2,330篇文章（allintitle:（Colaboratory OR Colab）），同時又符合Google scholar 0篇（（allintitle:（Colaboratory OR Colab））AND Google scholar）。使用AI，例如DeepSeek輸入提示詞（Colab，使用關鍵字搜尋Google Scholar文獻的Title，需要呈現文獻的Author、Title、Year、Volume、Pages、文獻的網址，結果儲存成csv檔），即可自動生成程式碼，經過多次的互動做修改，最後的程式碼呈現在圖二。修改程式搜尋關鍵詞為“Tirzepatide”，搜尋的輸出結果在圖三。點擊Google Scholar可以直接到學術搜尋網頁，點選可用的AI功能（✦）/說明程式碼，會出現程式碼的功用解釋，再者，AI也會提供清晰的程式碼解釋，幫助使用者理解每一程式碼的功能。由於程式碼或執行結果很長，透過AI詢問如何隱藏結果及程式碼，發現可以使用數種方法，我們採用#@title後面再加上要顯示的文字，例如“Show/Hide code（Each time you start, just execute once.）”點擊v符號可隱藏程式碼，!pip install可以獨立執行或加在其他程式碼的前面（圖二）。



圖一 Google Colaboratory之Colab Notebook
開啟Google雲端硬碟/新增/更多/Google Colaboratory/貼上程式碼
*如果沒有出現Colaboratory，連結更多應用程式加入

```
#@title Show/Hide code （ Each time you start, just execute once. ） [Google Scholar]（ https://scholar.google.com.tw/?hl=zh-TW ）
# 安裝必要套件
!pip install scholarly ipywidgets requests beautifulsoup4 -q IPython
from bs4 import BeautifulSoup
from scholarly import scholarly
import ipywidgets as widgets
from IPython.display import display, clear_output
import csv
import re # Importing the 're' module for regular expression operations

# Import the requests library here
import requests

# 自訂函數：取得 Google Scholar 真實總文章數
def get_total_results（ keyword, year_low, year_high, search_title_only ）:
    """直接解析 Google Scholar 網頁獲取總結果數"""
    headers = {
        'User-Agent': 'Mozilla/5.0 （ Windows NT 10.0; Win64; x64 ） AppleWebKit/537.36 （ KHTML, like Gecko ）
Chrome/91.0.4472.124 Safari/537.36'
    }
    # 根據是否只搜尋標題來調整查詢語法
```

```
    query_prefix = "allintitle:" if search_title_only else ""
            url = f"https://scholar.google.com/scholar?q={query_prefix}{keyword}&as_ylo={year_low}&as_yhi={year_high}"
    response = requests.get（url, headers=headers）
    soup = BeautifulSoup（response.text, 'html.parser'）

    # 定位總結果數元素
    result_stats = soup.find（'div', {'id': 'gs_ab_md'}）
    if result_stats:
        # 解析數值 （範例文字："約 1,230 項結果"）
        match = re.search（r'（\d{1,3}（,\d{3}）*）', result_stats.text）
        if match:
            return int（match.group（1）.replace（',', ''））
    return 0

# 修改搜尋函數
def search_scholar（keyword, limit=10, year_low=None, year_high=None, search_title_only=True）:
    """整合總結果數與詳細資訊"""
    # 取得總文章數
    total = get_total_results（keyword, year_low, year_high, search_title_only）

    # 根據是否只搜尋標題來調整查詢語法
    query_prefix = "allintitle:" if search_title_only else ""
    search_query = scholarly.search_pubs（
        query=f"{query_prefix}{keyword}",
        year_low=year_low,
        year_high=year_high,
    ）
    results = []
    for _ in range（limit）:
        try:
            pub = next（search_query）
            bib = pub.get（'bib', {}）
            authors = [f"{name.split（）[-1]} {' '.join（name.split（）[:-1]）}" for name in bib.get（'author', []）]
            venue = bib.get（'venue', ''）
            venue_abbr = venue.replace（'Journal of ', 'J. '）.replace（'Proceedings of the ', 'Proc. '）.replace（'Transactions on ', 'Trans. '）.replace（'Conference on ', 'Conf. '）.replace（'Symposium on ', 'Symp. '）.replace（'International ', 'Int. '）.replace（'National ', 'Nat. '）.replace（'Annual ', 'Annu. '）.replace（'Meeting of the ', 'Meet. '）.replace（'Workshop on ', 'Worksh. '）.replace（' and ', ' & '）.replace（'IEEE ', ''）.replace（'ACM ', ''）
            results.append（{
                'title': bib.get（'title', ''）,
                'authors': ", ".join（authors）,
                'year': bib.get（'pub_year', ''）,
                'venue': venue_abbr,
                'url': pub.get（'pub_url', ''）
            }）
        except StopIteration:
            break
    return total, results

# 修改按鈕點擊處理
def on_search_button_clicked（b）:
    clear_output（wait=True）
```

```
    # 創建動態顯示元件
    total_html = widgets.HTML（
        value="<div style='color: #1e88e5; font-weight: bold; margin-left: 10px;'>總文章數 ：載入中...</div>"
    ）
    search_ui = widgets.HBox（[search_button, total_html]）
    display（keyword_input, limit_input, year_low_input, year_high_input, search_title_only_checkbox, search_ui）

    # 執行搜尋
    total, papers = search_scholar（
        keyword=keyword_input.value,
        limit=limit_input.value,
        year_low=year_low_input.value,
        year_high=year_high_input.value,
        search_title_only=search_title_only_checkbox.value,
    ）

    # 更新總數顯示
    total_html.value = f"""
    <div style='color: #1e88e5; font-weight: bold; margin-left: 10px;'>
        總文章數 ：{total:,} 篇 （已取得 {len（papers）} 篇）
    </div>
    """

    # 顯示詳細結果
    for idx, paper in enumerate（papers）:
        print（f"\n📄 第 {idx+1} 篇結果"）
        print（f"作者 ：{paper['authors']}"）
        print（f"標題 ：{paper['title']}"）
        print（f"出處 ：{paper['venue']}"）
        print（f"年份 ：{paper['year']}"）
        print（f"連結 ：{paper['url']}"）
        print（"—" * 50）

    # 將結果儲存成 CSV 檔案
    with open（'search_results.csv', 'w', newline='', encoding='utf-8'） as csvfile:
        fieldnames = ['Authors','Title', 'Venue', 'Year', 'URL']
        writer = csv.DictWriter（csvfile, fieldnames=fieldnames）

        writer.writeheader（）
        for paper in papers:
            writer.writerow（{
                'Authors': paper['authors'],
                'Title': paper['title'],
                'Venue': paper['venue'],
                'Year': paper['year'],
                'URL': paper['url']
            }）

# 創建介面元件
keyword_input = widgets.Text（value='', description='🔍 關鍵字:'）
limit_input = widgets.IntText（value=10, description='📚 搜尋數量:'）
year_low_input = widgets.IntText（value=2020, description='📅 年份下限:'）
```

```
year_high_input = widgets.IntText（value=2025, description='📅年份上限:'）
search_title_only_checkbox = widgets.Checkbox（
    value=True,
    description='僅搜尋標題',
    disabled=False,
    indent=False
）
search_button = widgets.Button（
    description='開始搜尋',
    button_style='success',
    icon='search'
）

# 綁定事件與顯示
search_button.on_click（on_search_button_clicked）
display（keyword_input, limit_input, year_low_input, year_high_input, search_title_only_checkbox, search_button）
```

圖二 以DeepSeek建立Python程式碼

討論

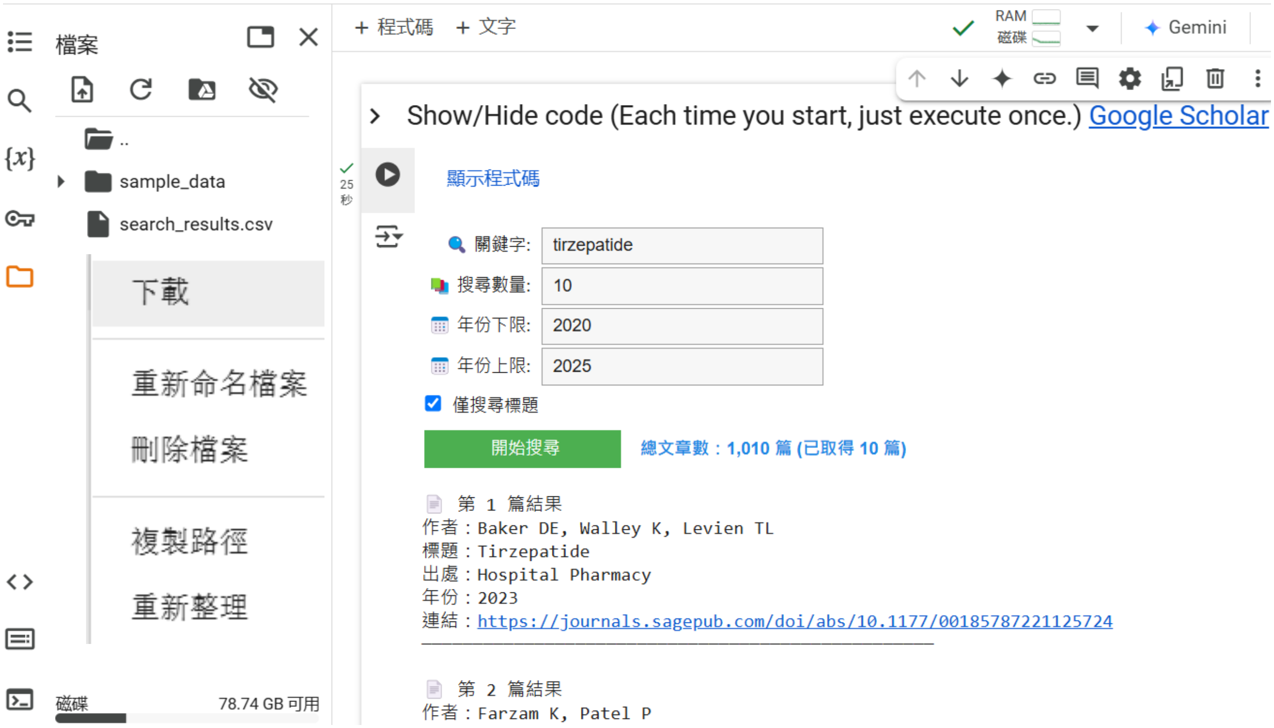
本文探討AI輔助程式語言編碼的應用，並以Google Colab整理Google Scholar搜尋到的資料為例，展示如何利用Python整理成逗號分隔值（Comma-separated Values, CSV）檔案，方便Excel快速讀取及後續整理備用，及透過AI提供的資訊的自學經驗，結果顯示AI具有可依照輸入提示詞，自動生成編碼的能力，頗適合自學及完全沒有程式語言基礎者。

本研究在程式設計之初，原先有加上下載PDF功能，而且程式會預設隨機延遲以模擬人類行為，避免大量下載時會被斷線，結果導致搜尋的速度相當的慢，因此我們再與AI溝通，輸入提示詞“速度太慢，可否刪除不必要的資料進一步加速”，及“關鍵詞放在最上方方便修改”，經過多次的溝通最後的結果呈現在圖二。AI提供一些建議，要加速程式，我們可以從以下幾個方面進行優化，移除不必要的功能，例如下載PDF檔案的功能，這會大幅增加程式的執行時間、減少資料庫操作，如果不需要將資料存到資料庫，可以移除資料庫相關的程式碼、減少隨機延遲，雖然隨機延遲可以模擬人類行為，但會顯著降低速度、可以根據需求調整或移除只提取必要的欄位，如摘要、引用次數等可移除。其中的隨機延遲模擬人類行為，用意是當要大量的下載資料時，要讓電腦無法判別是利用機器在下載，如果是被認定是機器在大量下載會被斷線。我們設定程式執行之後有2種呈現方式，直接呈現在Notebook平臺及csv檔案（圖三），後者在離開雲端硬碟後會消失，所以可先下載到硬碟保存。逗號分隔值也稱字元分隔值，因為分隔字元也可以不是逗號，其檔案以純文字形式儲存表格資料（數字和文字），所以任何程式皆可讀取，包括MS-Excel會自動轉換成表格。

由於程式碼或執行結果很長，要透過輸入框輸入資料比較不方便，透過AI詢問如何隱藏程式碼，發現可以使用多種方法，有些方法較複雜或無法執行，最後採用#@title 後面再加上要顯示的文字，例如“Show/Hide Code”，點擊v符號可隱藏程式碼（圖二）。

一般自學程式編碼可能會面臨以下的挑戰，例如一、傳統學習方式偏重理論，缺乏實際應用場景；二、環境配置複雜，初學者常因程式環境配置問題而放棄學習；三、學習資源分散，網路上的學習資源質量參差不齊，難以系統化學習等等。

利用AI輔助程式學習中的應用，例如Copilot、DeepSeek、ChatGPT、Gemini等工具能夠提供即時的程式碼解釋和錯誤修正建議，因編碼文字一般都較淺顯易懂，設計者都可依樣畫葫蘆做修正，再者，可以將編碼複製貼上，請AI幫忙除錯或修改、AI工



圖三 Python程式可在Google Colaboratory執行
程式執行之後有2種呈現方式，直接呈現在平臺及逗號分隔值（Comma-separated Values, CSV），
後者在離開雲端硬碟後會消失，但可下載到硬碟作保存。
點擊Google Scholar可以直接到學術搜尋網頁

具可以自動生成程式碼片段，幫助初學者快速上手。本文介紹以Colab執行Python的優勢，例如無需配置環境，提供即開即用的Python環境，有別於一般的編譯式語言（Compiled Language），例如C++必須先編譯為.exe檔案才可執行，初學者可以專注於學習程式設計、免費的GPU/TPU資源使得資料處理和分析更加高效、協作與分享功能，方便多人協作和知識分享等優勢。因免費AI有些可能會有使用量及輸入文字多少的限制，建議可以彼此交替使用，再者，Colab有內建Gemini，也可直接使用生成程式碼，但有字數的限制。

ChatGPT是OpenAI公司開發的人工智慧（能）聊天機器人，其基於GPT-3.5及後來的GPT-4架構的大型語言模型[1]，輔以部分人工的參與修正，並以強化學習訓練（訓練資料為2021年9月前），除了可用人類自然對話方式互動外，還有編寫和除錯電腦程式碼的能力，於2022年11月一推出，即造成舉世的轟動，上線兩個月即有上億使用者[3]，有關AI的話題一時蔚為風潮，後來又發表了GPT-4o[2]。

DeepSeek（深度求索）是一家成立於2023年7月的中國人工智慧公司，專注於開發先進的大語言模型和相關技術。在2024年底，其發佈最新的模型DeepSeek V3，引起了廣泛關注，在性能上與頂尖閉源模型如GPT-4o和Claude-3.5-Sonnet不相上下，使西方科技及媒體大感意外[7]。DeepSeek V3在試點基準測試中表現優異，在某些方面甚至超越了其他閉源模型，例如，在美國數學競賽（AIME 2024）和中國全國高中數學聯賽（CNMO 2024）中，DeepSeek-V3明顯領先於其他模式[9]。但其最大對話長度仍有限制，如果超過，只要開啟另一個新的對話即可繼續。其呈現的語言為簡體字，但可以請它改用繁體字。

Python是目前最受歡迎的程式設計語言，可以做網頁開發、資料科學與機器學習，特別是最後者，使它成為最流行的語言，最近幾年皆是排行榜冠軍。目前有Colab提供的優越環境使用，對自學

或初學者是一大福音，大大降低了進入的學習門檻。Python是一種高階、直譯式、通用且易於學習的程式語言，由Guido van Rossum於1991年首次發布，其以其簡潔的語法、強大的功能和廣泛的應用領域而聞名。其是直譯式語言，不需要編譯，程式碼可以直接運行，這使得開發和測試更加快速，但也導致執行速度較慢。

Google Colab是由Google提供的免費雲端開發環境，基於Jupyter Notebook，專為機器學習、數據分析和教育所做的設計。允許用戶在瀏覽器中編寫和執行Python程式碼，並提供免費的GPU和TPU資源，相當適合初學者、學生和研究人員。其已預裝TensorFlow、PyTorch、Pandas、Numpy等前人已設計完整的常用數據科學和機器學習庫，再者，亦可使用!pip install安裝Colab未預裝的資料庫，這些過程都可以請AI協助，!pip install可以獨立執行或如果加在其他程式碼的前面，就不會多一欄結果。

Google Scholar有嚴格的反爬蟲機制，頻繁下載可能會導致IP被封鎖，Serpapi是一個付費的API服務，專門用於爬取Google搜尋結果，包括Google Scholar，它提供了簡單的API接口，並且可以繞過Google的反爬蟲機制。本文只是探討如何利用AI和Colab平台輔助自學Python程式編碼，並以整理Google Scholar搜尋到的資料為例做說明，在此建議遵守Google Scholar的使用條款，不要非法進行大規模爬取資料。

研究限制

本文僅列出最後完成的結果，在中間的過程，AI對於不明確的描述提示，可能生成與需求不符的代碼[10]，由於篇幅的關係，尚有少部分的異常或錯誤並未在此顯示。再者，AI不斷的再學習，所以在不同的時間給予相同的條件，結果可能也會不大一樣，此時可以再不斷的對話溝通，或使用另一種AI工具。目前AI工具相當多元[3,11,12]，各有優缺點，然大部分免費的皆可生成、解釋和優化程式碼，所得結果類似。Colab提供優越的Python使用及學習的環境平臺，然而有執行時間限制，免費版用戶的Notebook在閒置一段時間後，或大量的下載資料，會導致自動斷開連接。有時候Colab資源分配網路的問題，連線可能比較慢。由Google Scholar搜尋到的資料，無法提供文獻的卷數及頁數，需要自己再加入，再者本設計搜尋的結果與Google Scholar有時稍有出入，因為本設計是開源模型，有興趣者可進一步的再做增修。

結論

在程式語言編碼方面，發現AI可作為自學或初學者的強力助手，本研究使用的都是免費且不用下載安裝軟體，對於初學者較方便，只要按圖索驥即可，稍有基礎後，再嘗試更複雜的設計及做編修。總之，資訊科技的進步，給予AI提示詞，即可自動生成要設定的程式碼，對於不了解的某個功能說明也可再與之對話，因此可達到自我學習的功能，初學者亦能快速上手，大大地降低進入學習程式語言的門檻。本文展示了如何利用AI工具和Google Colaboratory平台輔助自學程式編碼，並以整理Google Scholar資料為實際案例。透過Colab的雲端運算能力和AI工具的自動化功能，初學者能夠快速掌握Python程式設計技巧，並應用於實際問題的解決。未來研究可以進一步探討AI工具在其他程式語言學習中的應用，及如何優化自學流程，再者本設計是開源模型，有興趣者可進一步的在做修改。

誌謝

感謝澄清醫院中港分院於電腦軟硬體方面的協助，及醫學教育研究部經費（CH11300280）補助。本研究通過澄清醫院人體試驗委員會（IRB）的審查（HP230014）。

利益衝突聲明

不同工具各有其優缺點，不同公司的AI皆會不斷地自我學習、繼續精進，且各有所長，本文僅就自學過程中提出部分使用上的經驗，與本研究提及的公司或軟體無任何利益衝突。

考文獻

1. Lin M: Large Language Model (LLM). 2023. Retrieved from https://www.markreadfintech.com/p/llm
2. Altman SH: OpenAI第一屆DevDay主題演講。2023。Retrieved from https://www.youtube.com/watch?v=lydmOkYSLyE

3. Wikipedia: ChatGPT. 2024. Retrieved from https://zh.wikipedia.org/wiki/ChatGPT
4. ZDNET: ChatGPT can write code. Now researchers say it's good at fixing bugs, too. 2025. Retrieved from https://pse.is/7ht72s
5. MSP Corporation: ChatGPT vs. Microsoft 365 Copilot: What' s the difference? 2023. Retrieved from https://pse.is/7ht942
6. UNWIRE.HK：中國AI公司新發表DeepSeek V3：一文看清2大技術及6個原因讓AI巨頭們感意外。2024。Refrieved from https://pse.is/7hta3v
7. 騰訊網：成本是GPT的1/20，轟動矽谷的中國大模型DeepSeek是怎麼做到的？2025。Retrueved from https://pse.is/7htape
8. 蔡崇煌、顏啟華、林正介 等：資訊科技輔以人工智慧（AI）優化家庭檔案數位化。台灣家庭醫學雜誌 2024；34（1）：24-39。
9. Baidu: DeepSeek. 2025. Retrieved from https://pse.is/7htbqg
10. Xu W, Ouyang F: The application of AI technologies in STEM education: a systematic review from 2011 to 2021. International Journal of STEM Education 2022; 9(1): 1-20.
11. 數位時代：新｜全球43大AI工具排行榜，Perplexity、Gamma…有哪些必收好用AI App？2025。Retrieved from https://pse.is/7hu2pa
12. Wong MF, Guo S, Hang CN, et al.: Natural language generation and understanding of big code for AI-assisted programming: a review. Entropy2023; 25(6): 888.

Artificial Intelligence-assisted Self-learning of Coding: An Example of Using Google Colaboratory for Organization of data from Google Scholar

Ming-Jung Tsai[1], Wei-Geng William[4], Chung-Huang Tsai[2,3,4], Ho-Yuan Chen[1]

Graduate Institution of Education[1], Bachelor of Science in Senior Wellness and Sports Science[2], Center for General Education[3], Tunghai University; Department of Family Medicine, Chung Kang Branch, Cheng Ching Hospital[4]

Abstract

Purposes

Rapid advances in artificial intelligence (AI) techniques have made self-learning coding highly efficient and convenient. This study investigated the use of AI tools and the Google Colaboratory (Colab) platform to assist self-learning of Python coding, with the application of Colab to the organization of data retrieved from Google Scholar as an illustrative example.

Methods

AI models such as Microsoft Copilot, DeepSeek, ChatGPT, and Google Gemini were used sequentially to assist with Python coding, and codes were executed in Google Drive/Colab. All processes were performed on an ASUS laptop (TSDMU3MP) with an Intel Core i7-1165G7 2.80 GHz processor and the Windows 11 operating system.

Results

We investigated the feasibility of using AI to assist with coding and discovered that AI could serve as a powerful aid for developers. By leveraging the cloud computing capabilities of Colab, even beginners with no prior experience can quickly learn Python programming techniques. During the application of self-learned Python coding to the organization of academic data from Google Scholar, the provision of prompts to AI models led to the automatic generation of Python codes. Certain models provided additional information such as user instructions, function explanations, code annotations, example executions, and precautions. AI could also debug and optimize code. This enabled beginners with no prior experience to effortlessly gain familiarity with Python programming on the Colab platform. Using different input conditions in different AI models produced different results, which could be compared to achieving self-learning of Python programming.

Conclusions

With AI assistance, the barrier to learning coding can be significantly reduced, and familiarity with Python code execution can be easily gained on the Colab platform. The annotations and explanations provided by AI can also serve as references for self-learning by programming beginners. (Cheng Ching Medical Journal 2025; 21(3): 13-22)

Keywords : Artificial intelligence, Google Colaboratory, Google Scholar, Python, Self-learning

Received: February 2025; Accepted: March 2025