教育部教學實踐研究計畫內容

Project Proposal for MOE Teaching Practice Research Program

學門分類：資訊科技類

申請年度：115 年度　☒ 一年期　☐ 多年期

計畫名稱

從「依賴」到「賦能」：自我調節學習導向之 AI 鷹架撤除模式——在非資訊背景學生程式教育之應用

計畫主持人：陳文盛助理教授

執行機構：國立東華大學通識教育中心

計畫申請日期： 114 年 12 月 1 日

一、計畫主持人

（一） 計畫主持人相關教學經歷

在 111 至 113 學年度期間，本人於國立東華大學通識教育中心擔任專任教師，主要開設程式設計與人工智慧相關通識課程。教學對象涵蓋全校各學院非資訊背景學生，累計開設 22 門課程，修課總人數達 800 人，平均教學評量 4.25 分。近三年完整教學紀錄如表 1所示。

本人教學範疇涵蓋以下五大課程類型：

1. **初級程式設計－ Python**：針對零基礎學生，從程式思維與基礎語法出發，培養運算思維與問題解決能力。
2. **中級程式設計－虛擬實境**：引導學生運用程式設計知識進行 3D 互動內容創作，結合跨域應用。
3. **人工智慧概論**：介紹 AI 基本原理與應用，培養學生對 AI 技術的理解與批判思考能力。
4. **創客入門－智慧生活裝置實作**：整合程式設計、電子電路與實作能力，進行 IoT 專題製作。
5. **科技藝術概論**：跨域結合科技與藝術，探索數位創作的可能性（藝術學院）。

（二） 教學特色與成效

綜合 111-113 學年度教學紀錄，本人共開設 22 門課程，累計修課人數達 800 人，平均教學評量 4.25 分。其中「創客入門」課程（4.35 分）與「中級程式設計－虛擬實境」課程（4.28 分）表現穩定，但「初級程式設計－ Python」課程評量（4.21 分）仍有改善空間，需透過更有效的教學設計與 AI 鷹架支持以提升學生學習成效。

本人教學特色與成效主要體現在以下五個方面：

1. **深入理解學習困難**：三年累計教授 227 位非資訊背景學生學習 Python（5 班次），對其常見迷思概念、學習障礙與動機問題有第一手觀察與因應經驗，為本計畫之教學設計提供紮實基礎。
2. **AI 融入教學經驗**：開設 6 班次「人工智慧概論」課程（累計 256 人），熟悉生成式 AI 工具的教學應用、學習影響與潛在風險，為本計畫之 AI 鷹架設計提供實務基礎。
3. **跨域整合能力**：除程式設計外，同時開設虛擬實境、創客實作與科技藝術等跨域課程，具備整合不同領域知識進行創新教學設計的能力。

Table 1: 111-113 學年度完整教學紀錄（按課程類型分類）

課程名稱	學年度	學期	班別	評量	人數	該學期平均
初級程式設計－ Python（5 班次，共 227 人）						
初級程式設計－ Python	111	1	AA	4.14	51	4.22
初級程式設計－ Python	111	2	AA	4.03	39	4.30
初級程式設計－ Python	112	1	AC	4.23	38	4.33
初級程式設計－ Python	112	2	AA	4.42	50	4.16
初級程式設計－ Python	113	1	AC	4.21	49	4.21
小計平均				**4.21**	**227**	—
中級程式設計－虛擬實境（7 班次，共 187 人）						
中級程式設計－虛擬實境	111	1	AA	4.24	29	4.22
中級程式設計－虛擬實境	111	1	AB	4.05	30	4.22
中級程式設計－虛擬實境	111	1	AC	4.10	23	4.22
中級程式設計－虛擬實境	111	2	AA	4.50	22	4.30
中級程式設計－虛擬實境	111	2	AB	4.11	29	4.30
中級程式設計－虛擬實境	112	1	AA	4.36	25	4.33
中級程式設計－虛擬實境	112	1	AB	4.61	29	4.33
小計平均				**4.28**	**187**	—
人工智慧概論（6 班次，共 256 人）						
人工智慧概論	111	1	—	4.28	40	4.22
人工智慧概論	111	2	AA	4.53	37	4.30
人工智慧概論	112	1	AB	4.25	38	4.33
人工智慧概論	112	2	AA	4.09	56	4.16
人工智慧概論	112	2	AB	3.99	35	4.16
人工智慧概論	113	1	AA	4.20	50	4.21
小計平均				**4.21**	**256**	—
創客入門－智慧生活裝置實作（3 班次，共 90 人）						
創客入門－智慧生活裝置實作	111	1	—	4.52	27	4.22
創客入門－智慧生活裝置實作	111	2	—	4.36	33	4.30
創客入門－智慧生活裝置實作	112	1	—	4.19	30	4.33
小計平均				**4.35**	**90**	—
科技藝術概論（1 班次，共 40 人）						
科技藝術概論	113	2	—	4.31	40	4.31
小計平均				**4.31**	**40**	—
總計/整體平均				**4.25**	**800**	**4.25**

4. **大班教學經驗**：曾執行 50-56 人之大班教學（112-2 人工智慧概論），對於異質性學習群體的班級經營、差異化教學與學習支持策略有實務經驗。
5. **持續改進導向**：教學評量穩定維持在 4.0 分以上，22 門課程中有 14 門達 4.2 分以上，顯示具備教學反思與持續優化的習慣，符合教學實踐研究之精神。

（三） 與本計畫之關聯性

本人之教學經歷與專業背景，與本研究計畫具有高度關聯性：

- **目標學生群體經驗**：累計 5 班次、227 人次之 Python 教學經驗，深入理解非資訊背景學生在程式學習上的困境與需求。
- **AI 教學應用基礎**：6 班次、256 人次之 AI 概論教學經驗，熟悉 AI 工具在教學中的應用模式與學習影響。
- **實踐研究能力**：持續三年穩定之教學表現（4.25 分），展現教學反思、設計優化與成效評估能力。
- **跨域教學視野**：涵蓋程式設計、AI、虛擬實境、創客實作等多元課程，具備整合不同教學策略與評量方法的經驗。

綜上所述，本人之教學經歷與專業背景，為本研究計畫之執行奠定堅實基礎。

二、計畫執行內容

（一） 研究動機與背景

1. 教學問題脈絡

隨著人工智慧技術的普及，程式設計能力已成為 21 世紀公民的基本素養，運算思維（Computational Thinking）更被視為跨領域問題解決的核心能力 [1]。本校通識教育中心開設之「初級程式設計－ Python」課程，參考哈佛大學 CS50P 課程設計 [2]，旨在培養非資訊科系學生的運算思維與程式設計能力。然而，教學實踐中面臨三大核心挑戰：

挑戰一：學習困難與高挫折率

113 學年度課程資料顯示，62% 學生反映初期學習困難，期中考僅 47% 及格，且僅 31% 學生能獨立完成期末專題。非資訊背景學生在學習程式設計時面臨多重障礙，包括抽象概念理解困難、語法錯誤頻繁、缺乏問題分解能力等 [3, 4]。

挑戰二：生成式 AI 帶來的認知卸載危機

研究顯示，OpenAI Codex 等 AI 程式碼生成工具能以約 78% 的準確率解決入門程式設計問題 [5]，而透過自然語言提示（prompt engineering）與 ChatGPT、GitHub Copilot 等工具互動，學生可快速獲得程式碼建議 [6]，為程式設計教育帶來前所未有的機會與

挑戰 [7]。然而，學生過度依賴 AI 可能導致「認知卸載」（cognitive offloading），即將思考過程外包給 AI，而未真正理解程式邏輯 [8]。研究顯示，學生傾向於直接接受 AI 生成的程式碼而不加驗證，即使程式碼包含錯誤也難以察覺 [9]。此外，AI 工具在進階程式設計問題（如 CS2 資料結構與演算法）上同樣表現優異，能超越多數學生 [10]，這使得學生更容易產生依賴心理。

挑戰三：傳統教學方法的侷限

傳統的「講述-練習」教學模式難以有效支持學生建立自主學習能力。教師難以即時診斷每位學生的學習困難，也無法提供個別化的學習支持。此外，國內針對非資訊科系學生的程式設計教學研究也指出，單向講授模式不利於培養學生的運算思維與問題解決能力 [11, 12]。

2. 傳統教學方法說明

過去本課程採用以下教學方法：

1. **講述法為主**：教師透過投影片講解程式概念與語法，學生被動接收知識。
2. **範例示範**：教師示範程式碼撰寫過程，學生模仿練習。
3. **課後作業**：指派固定題目供學生練習，但缺乏即時回饋機制。
4. **AI 工具使用方式**：允許學生自由使用 ChatGPT 等工具，但未提供使用指引或限制。

此模式的主要問題：

- 學生缺乏主動探索與問題解決的機會
- 無法培養自我調節學習能力
- AI 工具使用缺乏教學設計，導致過度依賴
- 評量方式難以區分學生真實能力與 AI 協助程度

3. 改進方向與預期效益

有鑑於此，本研究提出「AI 鷹架漸退式教學模式」(AI Scaffolding Fading Model)，結合自我調節學習理論（Self-Regulated Learning, SRL）[13, 14] 與問題導向學習（Problem-Based Learning, PBL）[15, 16]，將 AI 從「答案提供者」轉變為「學習鷹架」。此教學設計參考認知學徒制（Cognitive Apprenticeship）中的鷹架理論 [17, 18]，透過逐步減少外部支持，促進學生建立內在學習策略與自主學習能力 [19]。

研究創新性：有別於現有研究多聚焦於 AI 工具對學習成效的影響 [8, 20]，本研究首次提出系統化的 AI 鷹架漸退機制，並結合學習歷程資料分析，探討如何透過教學設計引導學生從「AI 依賴」轉向「AI 協作」，最終達到「自主學習」（如圖1所示）。

核心理念：透過三階段 AI 鷹架漸退設計：

Figure 1: 研究架構圖

1. **引導期** (1-6 週)：高度結構化 AI 提示，引導學生思考
2. **轉換期** (7-12 週)：部分鷹架移除，鼓勵學生先嘗試
3. **自主期** (13-18 週)：最小化 AI 介入，學生主導學習

預期效益：

- 提升學生程式設計學習成效與自我效能，促進內在學習動機 [21]
- 降低 AI 過度依賴，培養自我調節學習能力
- 發展有效的 AI 輔助教學模式，可推廣至其他課程
- 探索 AI 工具在教育中的負責任使用方式 [22]

具體而言，本研究以「先思考、後求助、再驗證」為核心原則，透過結構化提示模板引導學生建立有效的 AI 協作學習策略，詳細教學設計與評量機制請見第2.3節。

4. 研究問題

本研究擬探討以下四個核心問題：

1. **學習成效**：在三階段 AI 鷹架漸退模式下，學生的 Python 程式設計能力（概念理解、程式撰寫、除錯能力、問題解決）相較於傳統教學是否顯著提升？
2. **學習行為轉變**：學生對 AI 工具的依賴程度與自我調節學習能力，在引導期、轉換期、自主期三階段中如何演變？具體表現為何？
3. **學習歷程分析**：不同階段的 AI 鷹架設計（提示結構、互動頻率、回饋類型），如何影響學生的學習策略、認知負荷與問題解決歷程？
4. **學習者觀點**：學生對 AI 輔助程式設計學習的態度、使用經驗、困難挑戰與學習反思為何？如何評價此教學模式？

（二） 文獻探討

本章將文獻探討分為三個層次：第一層討論學習問題與理論基礎，對應研究背景中提出的三大挑戰；第二層討論 AI 鷹架與教學方法，提供本研究教學設計的理論基礎；第三層分析現有研究的不足與研究缺口，說明本研究的必要性與獨特貢獻。

1. 學習問題與理論基礎

非資訊科系學生的程式設計學習困難　隨著運算思維被視為 21 世紀核心素養，程式設計教育已從資訊科系專業擴展至通識教育領域。然而，非資訊背景學生在學習過程中面臨諸多挑戰，這些困難形成了本研究「挑戰一」的背景。Guzdial 在其《以學習者為中心的運算教育設計》一書中 [3] 指出，程式設計對非專業學習者而言是一項高度複雜的認知任務，需要特別設計的教學策略以降低學習障礙。Qian 與 Lehman 對程式設計學習困難進行大規模文獻分析，其 [4] 系統性回顧文獻發現，新手程式設計者面臨三大類困難：(1) 語法知識（syntactic knowledge）錯誤，如括號不配對、變數命名規則誤用；(2) 概念知識（conceptual knowledge）誤解，如對迴圈、遞迴等核心概念的錯誤理解；(3) 策略知識（strategic knowledge）缺乏，即不知如何將問題分解為可編程的子任務。

從認知心理學角度來看，認知負荷理論（Cognitive Load Theory）提供了理解這些困難的理論框架 [3]。程式設計學習需要同時處理多重認知任務：理解抽象概念（如物件、函式、資料結構）、記憶語法規則（如 Python 的縮排、關鍵字、運算子）、追蹤程式執行流程（如迴圈迭代、條件分支）、偵錯與問題解決（如閱讀錯誤訊息、使用除錯工具）。這些任務對工作記憶造成巨大負擔，特別是對缺乏先備知識的初學者。

生成式 AI 帶來的機會與挑戰　2022 年以來，生成式 AI 技術的快速發展對程式設計教育帶來巨大衝擊。Finnie-Ansley 及其研究團隊針對 OpenAI Codex 進行的實證研究 [5] 顯示，生成式 AI 工具（如 GitHub Copilot、ChatGPT）能夠以約 78% 的準確率解決 CS1 等級的程式設計問題，甚至能處理複雜的編程作業。這看似為程式設計教育帶來了解決方案，卻同時引發新的教育挑戰（對應本研究「挑戰二」：認知卸載危機）：

- **過度依賴風險**：Kazemitabaar 等學者研究發現 [8]，學生在使用 AI 程式碼生成器時，常直接複製 AI 生成的程式碼而不加修改、不進行理解驗證，這種「複製貼上」（copy-paste）的行為模式可能導致學習流於表面。
- **錯誤接受率**：Prather 及其同事的質性研究 [9] 觀察到，當 AI 提供錯誤或不完整的答案時，初學者往往直接接受而不進行批判性思考，顯示出學生對 AI 的「過度信任」（overtrust）問題。
- **學習阻礙**：Finnie-Ansley 等人 [5] 警告，若學生過度依賴 AI 生成完整解答而跳過

自主思考過程，可能阻礙其建立紮實的程式設計基礎，長期而言不利於培養獨立問題解決能力。Finnie-Ansley 團隊後續研究 [10] 進一步發現，AI 工具在 CS2 進階課程考試中同樣表現優異，能超越多數學生，這使得依賴問題更加嚴重。

傳統教學法的效能與侷限　面對前述兩類挑戰（學習困難與 AI 依賴），傳統的講授式教學法顯得力不從心（對應本研究「挑戰三」：傳統教學方法的侷限）。Hmelo-Silver[16] 指出，單向講授模式難以促進學生的深度學習與自主學習能力，特別是在需要應用知識解決複雜問題的程式設計學科。臺灣本土研究也證實這個現象：林育慈與陳志洪 [11] 以及賴阿福與林志隆 [12] 的研究都發現，傳統講授模式下學生的學習焦慮高、學習動機低，需要更積極、更以學習者為中心的教學方法。

2. AI 鷹架與教學方法

AI 鷹架設計原則　為了平衡 AI 工具的協助性與學習效果，近期研究開始探索「結構化 AI 鷹架」的設計原則。Kazemitabaar 研究團隊 [20] 開發的 CodeAid 系統，採用「不直接提供完整程式碼」的設計哲學，反而透過概念解釋（解釋 programming 原理）、虛擬碼生成（提供演算法框架）、錯誤標註（標示問題所在位置）等方式提供分層級的學習支持。經過一學期的課堂實驗，研究顯示此種結構化 AI 鷹架能有效促進學生的深度學習，同時避免過度依賴，且獲得教師與學生的高度評價。

從另一個角度，Denny 等學者在 [6] 的研究指出，提示工程（prompt engineering）能力本身就是一項重要的學習目標與未來技能。學生學習如何精確描述問題（明確需求、提供上下文）、修正提示詞（根據 AI 回應調整問法）以獲得更好回應，此迭代互動的過程本身就培養了運算思維、問題分解與溝通表達能力。

然而，現有 AI 鷹架研究多採用「靜態」設計（固定程度的支持），缺乏「動態」調整機制（根據學習進展調整支持程度）。本研究提出的「三階段 AI 鷹架漸退模式」在此具有研究創新性，將鷹架支持從高（引導期）逐步降低（轉換期）直至最小化（自主期），配合學生能力發展軌跡。

自我調節學習（SRL）理論　自我調節學習（Self-Regulated Learning, SRL）理論為本研究的教學設計提供了重要的理論基礎。Zimmerman 在其影響深遠的研究 [13] 中將自我調節學習定義為學習者主動設定學習目標（目標設定）、監控學習歷程（後設認知監控）、調整學習策略（策略選擇與調適）的循環過程。這三個階段形成一個動態的回饋迴圈，促進學習者持續改進其學習表現。Azevedo 與 Gašević 針對科技輔助學習環境的研究 [14] 進一步指出，在使用進階學習技術（如 AI 工具）的環境中，SRL 能力是學習

成功的關鍵預測因子，缺乏 SRL 能力的學生容易陷入被動學習與過度依賴的困境。

Järvelä 與 Hadwin 在協作學習的脈絡下 [19] 提出社會性共享調節（socially shared regulation）的概念，強調在協作學習中，學習者如何透過互動溝通、共同計畫、互相監督等方式共同調節學習歷程，這對本研究中小組協作學習環節的設計具有啟發意義。

SRL 理論直接對應本研究的「研究問題二」（學習行為轉變），探討學生如何從低度 SRL 能力（依賴 AI）逐步發展至高度 SRL 能力（自主學習），以及 AI 鷹架漸退如何促進這個轉變過程。

問題導向學習（PBL） 問題導向學習（Problem-Based Learning, PBL）是另一個重要的教學法框架。Savery 在對跨學科 PBL 文獻的綜合回顧 [15] 中定義 PBL 為「以真實世界的複雜問題為起點，學習者透過自主探究、資料搜集與協作討論來解決問題的教學法」。此教學法強調學習者主動性與知識建構，有別於傳統的講授式教學。Hmelo-Silver 的實證研究 [16] 顯示，PBL 能有效促進深度學習（超越表面記憶的理解）、問題解決能力（分析、綜合、評估技能）與自主學習動機（內在動機與持續性），特別適合於程式設計這類需要應用知識解決實際問題的學科。

在臺灣本土的脈絡下，教學實踐研究也提供了 PBL 在程式設計教學成效的實證證據。林育慈與陳志洪的教學實踐研究 [11] 將 PBL 與翻轉教室結合應用於 Python 程式設計課程，經過一學期的實驗教學，結果顯示實驗組學生在學習成效、學習動機與學習滿意度三個面向都顯著優於對照組。賴阿福與林志隆針對非資訊科系大學生的研究 [12] 發現，問題導向教學模式能顯著降低學生的程式設計學習焦慮，提升其學習成就與自我效能，這對本研究針對通識課程非本科系學生的教學設計具有直接參考價值。

本研究將 PBL 的精神融入 AI 鷹架漸退設計：每階段都以真實的程式設計問題為導向，鼓勵學生主動探索解決方案，而 AI 鷹架的角色則從「引導者」逐步轉變為「協助者」再到「後援資源」。

動機理論與學習投入 學習動機是影響學習成效的關鍵因素。Ryan 與 Deci 的自我決定理論（Self-Determination Theory）[21] 指出，內在動機（源自興趣、好奇心與挑戰）相較於外在動機（獲得獎勵、避免懲罰）更能促進深度學習與持續投入。該理論提出三個基本心理需求：自主性（autonomy）、能力感（competence）與關係感（relatedness）。

本研究的 AI 鷹架漸退設計旨在滿足這三個需求：透過逐步減少外部支持提升自主性，透過適當的鷹架支持保證能力感（避免過高挫敗），透過小組協作建立關係感。這對應本研究的「預期效益」中提到的「促進內在學習動機」。

認知學徒制（Cognitive Apprenticeship） 認知學徒制（Cognitive Apprenticeship）理論則為本研究的「AI 鷹架漸退」機制提供了直接的理論基礎。Collins，Brown 與 Newman 在其開創性的論文 [17] 中提出的認知學徒制模式，強調透過「示範（modeling）-指導（coaching）-鷹架（scaffolding）-逐步退出（fading）」四個階段的有系統過程，將專家的隱性認知歷程（如思考策略、問題解決步驟、決策機制）外顯化給學習者。其中「鷹架漸退」（scaffolding fading）的核心精神是：初期提供大量支持，隨著學習者能力增長逐步減少外部協助，最終促使學習者內化知識與技能、達到獨立運作。此理論框架直接啟發了本研究的 AI 鷹架三階段漸退設計。

3. 教學設計與研究缺口

如何設計有效的 AI 鷹架以促進學習 綜合上述文獻，可以看出現有研究主要聚焦於生成式 AI 工具的能力評估與影響分析。例如 Finnie-Ansley 及其同事 [5] 與 Becker 研究團隊 [7] 的研究多探討「AI 能做什麼」、「學生如何使用 AI」、「使用 AI 對成績的影響」等議題，但較少深入探討如何在教學中有效、系統化地整合 AI 工具以促進真正的深度學習。Kazemitabaar 團隊的 CodeAid 研究 [20] 是重要的突破，其研究顯示精心設計的 AI 鷹架（如分層提示、漸進式提示）能在提供協助與促進學習之間取得平衡。

然而，即使是 CodeAid 研究也主要採用固定結構的鷹架設計，缺乏長期（全學期或跨學期）、系統性的縱貫性教學設計研究，探討如何透過階段性、有計畫地調整 AI 鷹架的支持程度（從高到低），並配合教學策略與學習活動，培養學生從 AI 依賴逐步轉變為 AI 協作、最終達到自主學習的能力。

研究缺口與本研究定位 綜觀前述文獻，本研究識別出以下五個研究缺口：

1. **動態鷹架機制的缺乏**：現有 AI 輔助學習研究多採用固定程度的支持設計 [20]，鮮少探討如何隨學習進展動態調整鷹架強度。本研究提出「三階段 AI 鷹架漸退模式」，系統性地從高度引導過渡至自主學習。
2. **SRL 發展軌跡的模糊**：既有文獻 [13, 14] 雖確立 SRL 於科技輔助學習的重要性，卻未深入探究 AI 環境中學生 SRL 能力的演變歷程。本研究將追蹤三階段中學生從依賴轉向自主的具體變化。
3. **過程資料的忽視**：多數研究側重學習成果的量化分析 [5, 7]，較少關注學習歷程的質性探究。本研究蒐集 AI 互動記錄與編程軌跡，深入分析問題解決過程。
4. **本土脈絡的欠缺**：國內程式設計教學研究 [11, 12] 聚焦於 PBL 等教學法，AI 輔助教學的實證探究仍屬起步階段。本研究以通識課程為場域，填補此一研究空缺。
5. **實務轉化的不足**：現有成果多停留於理論層次或實驗情境，缺乏可供移轉的教學

模式。本研究完整記錄 18 週實踐歷程，提供可操作的教學指引。

本研究的獨特定位在於：整合動態鷹架設計、SRL 能力追蹤、學習歷程分析、本土實證研究與長期教學實踐五個面向，此系統性的整合取徑為當前研究領域所欠缺。

（三） 教學設計與規劃

1. 教學理念與設計原則

本研究之教學設計以「AI 鷹架漸退模式」為核心，融合三大理論框架：自我調節學習理論（Self-Regulated Learning，SRL）[13]、認知學徒制（Cognitive Apprenticeship）[17]與問題導向學習（Problem-Based Learning，PBL）[15]。設計原則如下：

原則一：結構化 AI 支持——相關研究顯示 [20]，採用「不直接提供完整程式碼」的 AI 鷹架設計，透過概念解釋、虛擬碼生成、錯誤標註等方式提供分層支持，能有效促進學生學習並避免過度依賴。本課程據此設計 AI 工具為學習引導者而非答案提供者。

原則二：漸進式鷹架撤除——依據認知學徒制相關文獻 [17] 的「示範-指導-鷹架-漸退」模式，本課程將 AI 支持程度從高（引導期）逐步降至低（自主期）。鷹架的核心精神在於初期提供支持、隨能力增長逐步撤除，促進學習者內化知識與技能 [18]。

原則三：問題導向情境學習——PBL 能有效促進深度學習、問題解決能力與自主學習動機 [16]；在臺灣脈絡下的教學實踐亦證實 [11]，結合 PBL 與翻轉教室於程式設計課程能顯著提升成效與動機。本課程每週以真實程式設計問題為導向，增進學習動機與知識遷移。

原則四：自我調節學習培養——SRL 定義為學習者主動設定目標、監控歷程、調整策略的循環過程 [13]；在科技輔助學習環境中，SRL 能力是學習成功的關鍵預測因子 [14]。本課程透過學習反思報告、AI 使用紀錄回顧等活動，引導學生發展後設認知能力。

原則五：差異化教學支持——考量學生程式設計先備知識與學習步調之差異，本課程採用多層次差異化策略：(1) 提供基礎、進階與挑戰三種難度的練習題；(2) AI 鷹架支持程度可依個別學習需求彈性調整；(3) 設置「學習夥伴」制度，促進同儕互助學習。

2. 課程架構概述

本課程為期 **18 週**，每週 **3 小時**，共計 **54 小時**。課程設計分為兩大部分：

第一部分：Python 基礎知識建構（第 1–9 週）——課程架構參考哈佛大學 CS50P（CS50's Introduction to Programming with Python）的設計 [2]，涵蓋 Python 程式設計核心知識：函式與變數、條件判斷、迴圈、例外處理、套件、檔案處理等基礎概念。此部

分著重建立學生的程式設計基礎能力。

第二部分：問題解決與專題應用（第 10–18 週）——運用前 9 週所學的 Python 基礎知識，透過真實問題情境與專題實作，培養學生的問題解決能力、知識整合能力與自主學習能力。此部分著重將所學知識應用於實際問題，並逐步撤除 AI 鷹架，達成研究目標。

課程分為三個教學階段，配合 AI 鷹架漸退設計（如圖2所示）：

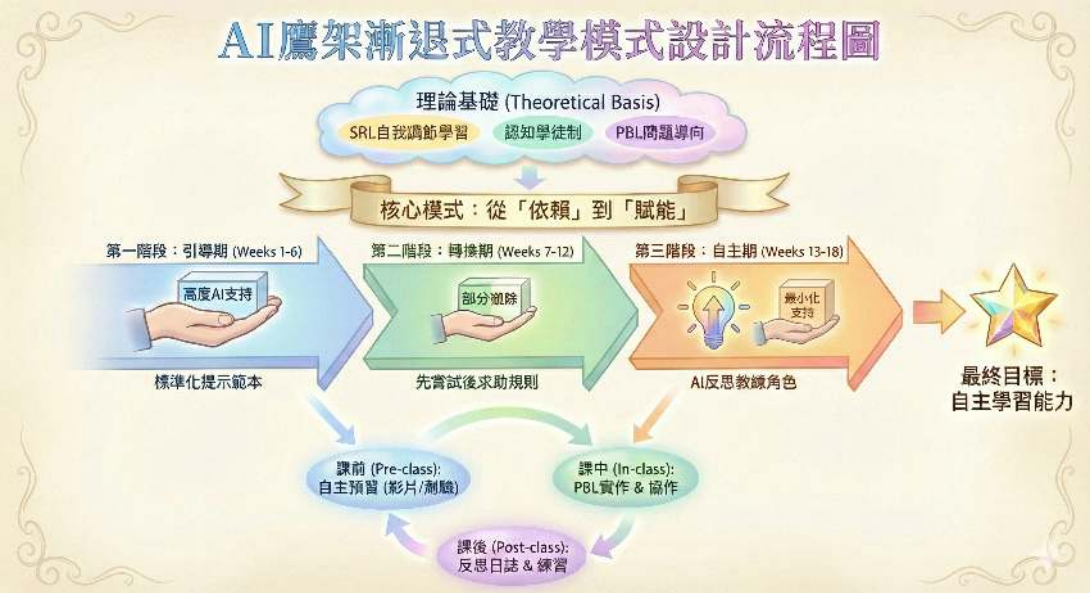

Figure 2: AI 鷹架漸退教學模式流程圖

Table 2: 三階段課程架構與研究目標對應

階段	週次	課程重點	對應研究目標
引導期	1–6 週	Python 基礎知識（CS50P 核心內容）	建立基礎能力、學習 AI 使用策略
轉換期	7–12 週	進階知識與問題解決應用	減少 AI 依賴、培養獨立思考
自主期	13–18 週	專題實作與知識整合	建立自主學習能力、完成專題

3. 18 週課程設計

第一階段：引導期（1–6 週） 教學目標：

- 建立 Python 程式設計基礎概念與語法
- 學習有效使用 AI 工具的策略與提示工程技巧
- 培養程式設計的基本問題解決思維

AI 鷹架設計：高度結構化支持

- 提供標準化提示範本（prompt templates），引導學生正確提問方式
- AI 回應包含三層次：概念解說 → 虛擬碼引導 → 思考提問
- AI 不直接輸出完整程式碼，要求學生根據提示自行實作
- 錯誤診斷模式：AI 指出錯誤類型與可能原因，不直接提供修正版本

學習成效指標：

- 能正確使用 Python 基本語法撰寫簡單程式（正確率達 70% 以上）
- 能運用提示範本與 AI 進行有效對話（提示品質評分達 3 分/5 分以上）
- 能識別並描述程式錯誤類型（錯誤辨識正確率達 60% 以上）

週次規劃：

Table 3: 引導期週次規劃

週次	主題	CS50P 對應內容	課堂活動
1	課程介紹與環境設定	Week 0: Functions (1/2)	認識 Python、安裝環境、第一支程式
2	變數與資料型別	Week 0: Variables (2/2)	變數宣告、資料型別、輸入輸出、前測施測
3	條件判斷	Week 1: Conditionals	if-elif-else、布林運算、邏輯判斷
4	迴圈結構	Week 2: Loops	while 迴圈、for 迴圈、range 函式
5	函式設計	Week 0: Functions (進階)	函式定義、參數傳遞、回傳值
6	資料結構基礎	Week 2: Lists	串列、字典、基本操作、第一次量表施測

第二階段：轉換期（7–12 週） 教學目標：

- 學習進階 Python 概念與技術
- 培養獨立思考與問題解決能力
- 減少對 AI 工具的依賴程度

AI 鷹架設計：部分撤除支持

- 移除標準化提示範本，鼓勵學生自行設計提示詞
- 建立「先嘗試後求助」規則：學生需先獨立嘗試 15 分鐘再使用 AI
- AI 回應轉為「問題診斷」模式，提供方向性建議而非具體解法
- 引入同儕協作機制：參考社會性共享調節概念之相關研究 [19]
- 實施「AI 使用額度制」：每週限定 AI 諮詢次數，培養策略性求助行為

學習成效指標：

- 能獨立設計有效的 AI 提示詞（提示自主設計比例達 80% 以上）
- 能在無 AI 協助下完成基礎程式任務（獨立完成率達 50% 以上）
- 能與同儕協作解決程式問題（同儕互評滿意度達 4 分/5 分以上）

週次規劃：

Table 4: 轉換期週次規劃

週次	主題	CS50P 對應內容	課堂活動
7	例外處理	Week 3: Exceptions	try-except、錯誤類型、防禦性程式設計
8	套件應用	Week 4: Libraries	第三方套件、pip 安裝、API 串接
9	檔案處理	Week 6: File I/O	檔案讀寫、CSV 處理、資料存取
10	問題解決實作 (一)	綜合應用	整合 Week 0–6 知識解決實際問題
11	問題解決實作 (二)	綜合應用	小組協作解決複雜問題、期中專題規劃
12	期中專題發表	綜合應用	專題報告、同儕互評、第二次量表施測

第三階段：自主期（13–18 週） 教學目標：

- 整合所學知識完成期末專題
- 建立自主學習能力與後設認知
- 培養程式設計的持續學習態度

AI 鷹架設計：最小化支持

- AI 角色轉變為「反思教練」(reflection coach)
- 不主動提供程式建議，僅回應學生明確且具體的提問
- 重點功能：引導學生回顧學習歷程、檢視學習策略、自我評估
- 實施「AI 使用日誌」：學生需記錄每次 AI 使用的目的、過程與反思

學習成效指標：

- 能獨立完成中等複雜度的程式專題（專題評分達 70 分以上）
- 展現自我調節學習行為（SRL 量表得分較前測提升 10% 以上）
- 能批判性評估 AI 建議並做出合理判斷（反思報告品質達 4 分/5 分以上）

週次規劃：

4. 每週課堂活動設計

每週 3 小時課程採用「課前-課中-課後」三段式設計：

Table 5: 自主期週次規劃

週次	主題	學習重點	課堂活動
13	進階主題（一）	Week 7: Regular Expressions	正規表示式、文字處理應用
14	進階主題（二）	Week 8: OOP 基礎概念	類別與物件基本概念（簡化版）
15	期末專題規劃	專題設計	專題選題、需求分析、架構設計
16	期末專題實作(一)	專題開發	核心功能開發、問題解決
17	期末專題實作(二)	專題完善	功能完善、測試除錯、後測施測
18	期末專題發表	成果展示	專題報告、同儕互評、第三次量表施測

課前活動（自主學習約 30 分鐘）：

- 觀看教師錄製的概念講解影片（10-15 分鐘）
- 完成線上預習測驗
- 記錄預習疑問點

課中活動（面對面教學 3 小時）：

- 概念澄清與重點講解（30 分鐘）：針對預習問題進行解答
- 範例示範與練習（60 分鐘）：教師示範、學生跟做練習
- PBL 任務實作（60 分鐘）：小組協作解決本週程式設計任務
- 成果分享與反思（30 分鐘）：小組報告、教師回饋、學習反思

課後活動（自主學習約 60 分鐘）：

- 完成延伸練習題
- 撰寫 AI 使用紀錄與學習反思（每 4 週繳交一次正式報告）

5. 形成性評量機制

本課程採用多元形成性評量，即時掌握學生學習狀況並提供回饋：

6. PBL 任務設計

本課程 PBL 任務設計強調三大核心能力：「資料處理」、「資料視覺化」與「真實問題解決」。基礎練習題（如變數運算、迴圈練習、函式撰寫等）已融入每週課堂練習中，PBL 任務則聚焦於具有挑戰性的應用問題。任務設計結合「Vibe Coding」理念——鼓勵學生以自然語言描述需求，透過與 AI 協作逐步實現程式功能，培養人機協作的程式開發能力。

Table 6: 形成性評量機制設計

評量類型	評量內容	實施頻率	回饋方式
課堂即時測驗	概念理解檢核、程式追蹤題	每週	即時回饋與講解
程式實作評量	PBL 任務完成度、程式碼品質	每週	教師評語與同儕回饋
AI 互動紀錄分析	提示品質、求助策略、依賴程度	每 2 週	個別化學習建議
學習反思報告	學習歷程回顧、策略調整規劃	每 4 週	書面回饋與面談

PBL 任務評量規準　為確保評量一致性與透明度，本課程制定 PBL 任務評量規準如表7所示：

Table 7: PBL 任務評量規準表

評量向度	優秀 (5 分)	良好 (4 分)	待加強 (3 分)	配分
程式正確性	完全正確執行，處理各種邊界情況	大致正確，少數情況有誤	部分正確，需修正多處錯誤	30%
程式碼品質	結構清晰、命名恰當、註解完整	結構尚可、命名合理、有基本註解	結構混亂、命名不當、缺乏註解	20%
問題解決歷程	展現系統性思考與創意解法	能有條理地解決問題	解題過程較為零散	25%
AI 工具運用	策略性使用 AI，展現批判思考	適當使用 AI，能評估建議	過度依賴或未能有效運用 AI	15%
反思與改進	深入反思學習歷程，提出具體改進	能反思學習，有改進意識	反思較為表面	10%

核心技術能力培養　本課程 PBL 任務涵蓋以下技術能力：

- **資料處理**：CSV/JSON 檔案讀取、資料清理、資料轉換、資料篩選與彙整
- **資料視覺化**：使用 Matplotlib、Pandas 繪製統計圖表（長條圖、折線圖、圓餅圖、散佈圖）
- **API 串接**：呼叫政府開放資料 API、處理 JSON 回應、錯誤處理
- **問題解決**：需求分析、演算法設計、程式實作、測試除錯

臺灣開放資料應用　本課程大量採用「政府資料開放平臺」（data.gov.tw）之公開資料集，讓學生處理真實、在地化的資料，提升學習動機與實用性。使用之資料集包括：

- 中央氣象署開放資料：天氣預報、地震資訊、雨量觀測

- 交通部開放資料：YouBike 即時資訊、公車動態、臺鐵時刻
- 環境部開放資料：空氣品質 AQI、紫外線指數
- 衛生福利部開放資料：醫療院所、食品營養成分
- 內政部開放資料：人口統計、行政區域資訊
- 花蓮縣政府開放資料：在地觀光景點、活動資訊

Vibe Coding 教學策略　「Vibe Coding」強調以直覺、自然的方式與 AI 協作開發程式，本課程將此理念融入 PBL 任務：

- **引導期**：教師示範如何用自然語言描述需求，AI 協助產生程式框架，學生理解並修改
- **轉換期**：學生自行撰寫需求描述，批判性評估 AI 建議，選擇性採納並改進
- **自主期**：學生主導開發流程，AI 僅作為諮詢對象，培養獨立解決問題能力

7. AI 工具使用規範

為確保 AI 工具的教育價值，本課程制定明確的使用規範：

鼓勵的 AI 使用方式：

- 請求概念解釋：「請解釋 Python 中 for 迴圈與 while 迴圈的差異」
- 尋求除錯提示：「我的程式出現 IndexError，可能是什麼原因？」
- 請求虛擬碼引導：「如何設計一個計算平均值的演算法步驟？」
- 程式碼審查：「請檢視我的程式碼，指出可以改進的地方」

避免的 AI 使用方式：

- 直接要求完整程式碼：「幫我寫一個計算成績的程式」
- 複製貼上作業題目：「解這道程式設計題目」
- 未經思考的求助：在未嘗試理解問題前直接詢問 AI

PBL 任務一覽　各階段 PBL 任務設計如表8所示：

期末專題建議方向　學生可從以下方向選擇期末專題主題，皆須包含資料處理與視覺化分析：

1. **環境分析類**：空氣品質趨勢分析、氣溫變化視覺化、紫外線指數統計
2. **交通分析類**：YouBike 使用熱點分析、公車路線效率評估、交通流量視覺化
3. **人口社會類**：縣市人口趨勢分析、年齡結構變化圖、人口密度地圖
4. **健康飲食類**：食品營養成分比較、醫療資源分布分析、健康指標統計
5. **觀光旅遊類**：花蓮景點分析、觀光人次趨勢、旅遊資源視覺化儀表板

Table 8: PBL 任務設計一覽表

週次	PBL 任務名稱	任務說明	技術重點
引導期：結構化 AI 協作任務			
6	花蓮空氣品質監測儀表板	讀取環境部 AQI 資料，根據空氣品質等級提供健康建議，繪製 AQI 趨勢圖	資料處理、條件判斷、折線圖
轉換期：半自主問題解決任務			
9	花蓮 YouBike 使用分析系統	串接 YouBike API，分析各站點使用率，繪製站點分布圖與使用統計	API 串接、資料清理、長條圖與圓餅圖
10	臺灣人口結構視覺化分析	讀取內政部人口統計 CSV，分析各縣市人口結構、年齡分布	CSV 處理、Pandas 分析、多圖表繪製
11–12	期中專題：在地生活資料應用	整合至少兩種開放資料，進行資料處理與視覺化分析	多元資料整合、視覺化、小組協作
自主期：獨立專題開發任務			
13	新聞/社群文字資料探勘	使用正規表示式擷取網路文字資料，進行詞頻統計與視覺化	正規表示式、文字處理、詞頻統計圖
14	地震資料分析與視覺化系統	串接氣象署地震開放資料，分析地震分布規律	API 串接、OOP 概念、散佈圖與統計
15–18	期末專題：自選主題	自選臺灣開放資料主題，完成資料蒐集、處理、分析與視覺化	完整資料分析流程、知識整合

8. 學習支持與補救機制

為協助學習落後或遭遇困難的學生，本課程建立完善的學習支持機制：

- **預警系統**：透過每週形成性評量，及早識別學習困難學生
- **課後輔導**：每週提供 1 小時課後諮詢時段，提供個別化指導
- **學習夥伴制度**：配對程度較佳與需要協助的學生，促進同儕學習
- **補充教材**：提供基礎概念補充影片與額外練習題
- **彈性 AI 支持**：對學習落後學生，適度延長高支持階段時間

9. 教學設計特色總結

本課程教學設計之核心特色如下：

1. **理論基礎紮實**：融合 SRL、認知學徒制與 PBL 三大理論框架
2. **AI 鷹架創新**：採用漸退式 AI 支持，兼顧學習效能與自主能力培養
3. **在地化情境**：大量運用臺灣開放資料，提升學習動機與實用性
4. **多元評量設計**：結合形成性與總結性評量，全面掌握學習成效
5. **差異化教學**：提供彈性支持機制，照顧不同程度學生需求

（四） 研究設計與執行規劃

1. 研究設計

本研究採用**前後測對比設計**，整合量化與質性資料 [23]，探討 AI 鷹架漸退教學模式對學生學習成效之影響。

2. 研究問題

1. **學習成效**：學生的 Python 程式設計能力是否顯著提升？
2. **行為轉變**：學生對 AI 的依賴程度與自我調節學習能力如何轉變？
3. **歷程分析**：AI 鷹架設計如何影響學生的學習策略？
4. **學習者觀點**：學生對 AI 輔助學習的態度與反思為何？

3. 研究架構

- **自變項**：三階段 AI 鷹架教學（引導期 → 轉換期 → 自主期）
- **依變項**：程式設計能力、AI 依賴程度、自我調節學習能力
- **控制變項**：教學內容、授課教師、上課時數

4. 研究對象

國立東華大學 115 學年度第二學期「初級程式設計－ Python」通識課程修課學生，預估 50–60 人。研究對象為非資訊科系學生，無程式設計經驗或僅有基礎經驗。

5. 研究工具

本研究使用以下六項工具蒐集資料：

Table 9: 研究工具一覽

工具	內容說明	施測時間
Python 能力測驗	選擇題與程式撰寫題	第 2 週（前測）、第 17 週（後測）
SRL 與 AI 使用量表	短版量表 16–20 題	第 6、12、17 週
PBL 專題評量	程式正確性、問題分析等五向度	第 12 週、第 18 週
AI 互動日誌	問題類型、嘗試步驟、採納理由	每週 3–5 筆
學習反思報告	150–200 字，「一錯一改」架構	每 4 週 1 次
焦點團體訪談	每場 6–8 人，約 60 分鐘	第 12 週（2 場）

6. 資料蒐集時程

Table 10: 資料蒸集時程

週次	資料類型	對應研究問題
第 2 週	Python 前測	學習成效
第 6 週	第一次量表、反思報告	行為轉變
第 12 週	第二次量表、訪談、期中專題	行為、歷程、觀點
第 17 週	Python 後測、第三次量表	學習成效、行為轉變
第 18 週	期末專題、反思報告	學習成效、觀點
全學期	AI 互動日誌	歷程分析

7. 資料分析方法

量化分析：

- 前後測比較：比較學期初與學期末成績差異
- 三時點量表：分析第 6、12、17 週量表分數變化趨勢
- AI 互動行為：統計學生「先嘗試後求助」的比例變化

質性分析：採用主題分析法 [24]，聚焦自我調節、AI 依賴、學習策略三主題，雙人編碼確保信度。

8. 研究倫理

- **知情同意**：開學第一週說明研究目的，學生簽署同意書
- **資料保護**：所有資料去識別化，僅供學術研究
- **AI 使用揭露**：明確告知使用規範與學術誠信要求 [7]

（五） 預期完成工作項目與成果

1. 研究執行時程

Table 11: 研究執行時程

期程	工作項目	預期產出
準備階段（115 年 8 月–116 年 1 月）		
115 年 8–9 月	開發 AI 學習平台、製作 18 週課程教材	AI 平台 beta 版、教材初稿
115 年 10–12 月	研究工具預試與修訂、AI 平台測試、教材專家審查	驗證後研究工具、AI 平台正式版
116 年 1 月	課程網站建置、施測準備	課程網站上線
實施階段（116 年 2 月–6 月）		
116 年 2–3 月	執行引導期教學（1–6 週）、前測與第一次量表施測	前測資料、第一次量表資料
116 年 4 月	執行轉換期教學（7–12 週）、第二次量表施測與訪談	第二次量表資料、訪談逐字稿
116 年 5–6 月	執行自主期教學（13–18 週）、後測與第三次量表施測	後測資料、期末專題成果
分析撰寫階段（116 年 6–7 月）		
116 年 6–7 月	資料分析、撰寫研究報告	成果報告、論文初稿

2. 預期成果與 KPI

教學成效指標：

Table 12: 教學成效 KPI 指標

KPI 項目	衡量指標	預期目標
程式設計能力提升	後測成績高於前測	達統計顯著差異
AI 依賴程度下降	第三次量表分數低於第一次	呈現下降趨勢
SRL 能力提升	第三次量表分數高於第一次	呈現上升趨勢
期末專題完成率	獨立完成期末專題比例	$\geq 70\%$
課程及格率	期末成績及格比例	$\geq 75\%$

研究產出：

- **教學模組**：18 週完整課程教材、PBL 任務與評量規準、三階段 AI 提示範本庫
- **研究工具**：Python 程式設計能力測驗、AI 依賴與 SRL 量表中文版
- **研究發表**：投稿《教學實踐與研究》期刊 1 篇、校內教學成果發表 1 場
- **開放資源**：GitHub 開源專案（課程教材、AI 提示範本）

3. 預期貢獻

學術貢獻：

- 發展「AI 鷹架漸退教學模式」，整合認知學徒制 [17] 與 SRL 理論 [13]
- 提供臺灣高教脈絡下 AI 輔助程式設計教學的實證資料 [12]

實務貢獻：

- 發展可操作的 AI 鷹架設計原則 [20]
- 透過 AI 鷹架漸退提升學生自主性 [21]
- 開放教學資源供其他教師參考 [7]

參考文獻

[1] Jeannette M. Wing. “Computational Thinking”. In: *Communications of the ACM* 49.3 (Mar. 2006), pp. 33–35. DOI: 10.1145/1118178.1118215.

[2] Harvard University. *CS50's Introduction to Programming with Python*. Accessed: 2024-11-01. 2024. URL: https://cs50.harvard.edu/python/.

[3] Mark Guzdial. *Learner-Centered Design of Computing Education: Research on Computing for Everyone*. San Rafael, CA: Morgan & Claypool Publishers, 2015. DOI: 10.2200/S00684ED1V01Y201511HCI033.

[4] Yizhou Qian and James Lehman. “Students’ Misconceptions and Other Difficulties in Introductory Programming: A Literature Review”. In: *ACM Transactions on Computing Education* 18.1 (Oct. 2017). DOI: 10.1145/3077618.

[5] James Finnie-Ansley et al. “The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming”. In: *Proceedings of the 24th Australasian Computing Education Conference*. ACE ’22. Virtual Event, Australia: ACM, Feb. 2022, pp. 10–19. DOI: 10.1145/3511861.3511863.

[6] Paul Denny, Viraj Kumar, and Nasser Giacaman. “Conversing with Copilot: Exploring Prompt Engineering for Solving CS1 Problems Using Natural Language”. In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE ’23. Toronto, ON, Canada: ACM, Mar. 2023, pp. 1136–1142. DOI: 10.1145/3545945.3569823.

[7] Brett A. Becker et al. “Programming Is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation”. In: *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. SIGCSE ’23. Toronto, ON, Canada: ACM, Mar. 2023, pp. 500–506. DOI: 10.1145/3545945.3569759.

[8] Majeed Kazemitabaar et al. “Studying the Effect of AI Code Generators on Supporting Novice Learners in Introductory Programming”. In: *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. CHI ’23. Hamburg, Germany: ACM, Apr. 2023. DOI: 10.1145/3544548.3580919.

[9] James Prather et al. “”It’s Weird That it Knows What I Want”: Usability and Interactions with Copilot for Novice Programmers”. In: *ACM Transactions on Computer-Human Interaction* 31.1 (Feb. 2024). DOI: 10.1145/3617367.

[10] James Finnie-Ansley et al. “My AI Wants to Know if This Will Be on the Exam: Testing OpenAI’s Codex on CS2 Programming Exercises”. In: *Proceedings of the 25th Australasian Computing Education Conference*. ACE ’23. Melbourne, Australia: ACM, Jan. 2023, pp. 97–104. DOI: 10.1145/3576123.3576134.

[11] 林育慈 and 陳志洪. “結合問題導向學習與翻轉教室於程式設計課程之研究”. In: *教學實踐與研究* 4.1 (2021). Lin, Y.-T., & Chen, C.-H. (2021). Integrating Problem-Based Learning and Flipped Classroom in Programming Courses, pp. 1–30.

[12] 賴阿福 and 林志隆. “非資訊科系學生程式設計學習成效分析”. In: *教育科學研究期刊* 65.3 (2020). Lai, A.-F., & Lin, C.-L. (2020). Analysis of Programming Learning Effectiveness for Non-CS Majors, pp. 85–112.

[13] Barry J. Zimmerman. “Becoming a Self-Regulated Learner: An Overview”. In: *Theory into Practice* 41.2 (2002), pp. 64–70. DOI: 10.1207/s15430421tip4102_2.

[14] Roger Azevedo and Dragan Gašević. “Analyzing Multimodal Multichannel Data about Self-Regulated Learning with Advanced Learning Technologies: Issues and Challenges”.

In: *Computers in Human Behavior* 96 (2019), pp. 207–210. DOI: 10.1016/j.chb.2019.03.025.

[15] John R. Savery. "Overview of Problem-Based Learning: Definitions and Distinctions". In: *Interdisciplinary Journal of Problem-Based Learning* 1.1 (2006), pp. 9–20. DOI: 10.7771/1541-5015.1002.

[16] Cindy E. Hmelo-Silver. "Problem-Based Learning: What and How Do Students Learn?" In: *Educational Psychology Review* 16.3 (2004), pp. 235–266. DOI: 10.1023/B:EDPR.0000034022.16470.f3.

[17] Allan Collins, John Seely Brown, and Susan E. Newman. "Cognitive Apprenticeship: Teaching the Crafts of Reading, Writing, and Mathematics". In: *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser*. Ed. by Lauren B. Resnick. Hillsdale, NJ: Lawrence Erlbaum Associates, 1989, pp. 453–494.

[18] David Wood, Jerome S. Bruner, and Gail Ross. "The Role of Tutoring in Problem Solving". In: *Journal of Child Psychology and Psychiatry* 17.2 (1976), pp. 89–100. DOI: 10.1111/j.1469-7610.1976.tb00381.x.

[19] Sanna Järvelä and Allyson F. Hadwin. "Promoting and Researching Monitoring in Collaborative Learning". In: *International Handbook of Collaborative Learning*. Ed. by Cindy E. Hmelo-Silver et al. New York, NY: Routledge, 2015, pp. 200–218.

[20] Majeed Kazemitabaar et al. "CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs". In: *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*. CHI '24. New York, NY, USA: ACM, May 2024. DOI: 10.1145/3613904.3642773.

[21] Richard M. Ryan and Edward L. Deci. "Self-Determination Theory and the Facilitation of Intrinsic Motivation, Social Development, and Well-Being". In: *American Psychologist* 55.1 (2000), pp. 68–78. DOI: 10.1037/0003-066X.55.1.68.

[22] Virginia Dignum. "Responsible Artificial Intelligence: How to Develop and Use AI in a Responsible Way". In: *Artificial Intelligence: Foundations, Theory, and Algorithms* (2019). DOI: 10.1007/978-3-030-30371-6.

[23] John W. Creswell and J. David Creswell. *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. 5th. Thousand Oaks, CA: SAGE Publications, 2018.

[24] Virginia Braun and Victoria Clarke. “Using Thematic Analysis in Psychology”. In: *Qualitative Research in Psychology* 3.2 (2006), pp. 77–101. DOI: 10.1191/1478088706qp063oa.