國立臺灣師範大學科技與工程學院圖文傳播學系

碩士論文

Department of Graphic Arts and Communications

College of Technology and Engineering

National Taiwan Normal University

Master's Thesis

置入運算思維於學習鷹架中對高中程式寫作課程中之自我效能與學習成效之影響

The Influences of Inserting Computational Thinking in Learning Scaffolding on Self-Efficacy and Learning Outcome in High School Computer Programming

洪阡珈

Hong, Chien-Chia

指導教授：王健華 博士

Advisor：Wang, Chang-Hwa, Ph.D.

中華民國 111 年 9 月

September 2022

摘要

程式設計內容相互具有關聯性，然而程式概念抽象不易理解且複雜性高，因此多數學習者對於程式學習產生消極的信念，造成自我效能降低並影響學習表現。為了解決程式學習的困境，教學者多運用運算思維的概念來設計教學策略。本研究假設教學者提供學習鷹架來協助學習者與已知知識相互融會貫通，因後設認知鷹架能喚起學習者的已知知識，藉以輔助學習新知識，但其缺乏邏輯系統性的引導方式，所以本研究設計將運算思維流程置入於後設認知鷹架中來引導學習者學習程式設計。研究目的在於探討運算思維流程與傳統教學兩者鷹架引導方式對於學習者自我效能與學習成效之影響。研究方法為實驗法，實驗對象為普通高中一年級學生共 39 人，實驗組以運算思維流程之步驟來設計鷹架引導內容，對照組以傳統教學方式之程式敘述順序來設計鷹架引導內容，以學習成效測驗試題與自我效能量表作為量化研究工具，並在實驗結束後透過半結構式訪談收集質性資料。研究結果發現在進行實驗課程後，使用運算思維流程鷹架引導方式之學習成效與自我效能表現皆優於使用傳統教學鷹架引導方式，且自我效能與學習成效表現具有相關性。建議未來相關研究可深入探討影響自我效能與學習成效的其他因素。

關鍵字：程式設計學習、運算思維、學習鷹架、學習成效、自我效能

Abstract

Programming content is interconnected. However, programming concepts are mostly abstract, difficult to comprehend, and highly complicated. As a result, most learners have negative impressions about learning programming, leading to lower self-efficacy as well as learning performance. To solve this dilemma, instructors employed the idea of computational thinking when devising teaching strategies. Metacognitive scaffolds can evoke learners' known knowledge. Thus, this study hypothesized that by providing learning scaffolds, instructors, can help learners integrate new content with their existing knowledge so as to facilitate their learning of new knowledge. However, a logical and systematic scaffolding is lacking. Therefore, this study was designed to integrate computational thinking processes into metacognitive scaffolds to guide learners for studying programming. This study aimed to investigate the influences of the teaching scaffolds of computational thinking processes and traditional teaching approach on learners' self-efficacy and learning outcome. The experimental method was selected as the study methodology and 39 first-year high school students were chosen as the study subjects. For the experimental group, we adopted the steps of computational thinking processes to design the scaffolding content. For the control group, the scaffolding content was designed following the programming narrative sequence of the traditional teaching approach. We selected learning outcome related test questions and self-efficacy scale as quantitative research tools, and garnered qualitative data through semi-structured interviews at the end of the experiment. The study findings indicated that after the experimental sessions ended, learning outcome and self-efficacy driven by scaffolding of computational thinking were better than those under the traditional teaching approach. Moreover, a correlation existed between self-efficacy and learning outcome performance. It is suggested that future research works should investigate other factors affecting learners' self-efficacy and learning outcome.

Keywords: programming learning, computational thinking, learning scaffolds, learning outcome, self-efficacy

目錄

摘要 ..... **i**

Abstract ..... **ii**

目錄 ..... **iii**

表次 ..... **v**

圖次 ..... **vi**

第一章 緒論 **1**

第一節 研究背景與動機 1

第二節 研究目的與問題 3

第三節 研究範圍與限制 4

第四節 研究流程 5

第五節 名詞釋義 6

第二章 文獻探討 **8**

第一節 程式設計學習之困境 8

第二節 運算思維 9

第三節 鷹架理論 11

第四節 自我效能與程式設計學習成效 14

第五節 文獻探討小結 15

第三章 研究設計 **17**

第一節 研究架構 17

第二節 研究方法 19

第三節 研究對象 19

第四節 研究工具 20

第五節 教學內容設計 23

第六節 研究實施 29
第七節 資料處理與分析 31
第四章 研究結果與討論 33
第一節 不同鷹架引導方式之學習成效差異分析 34
第二節 不同鷹架引導方式之自我效能差異分析 35
第三節 自我效能與學習成效相關性分析 36
第四節 訪談內容分析 37
第五章 結論與建議 42
第一節 研究結論 42
第二節 後續研究建議 43
參考文獻 45
附錄一 教學課程教案 54
附錄二 教學教材 70
附錄三 課程練習題 75
附錄四 學習成效測驗試卷 81
附錄五 自我效能量表 84
附錄六 學習成效測驗評分表 90
附錄七 訪談大綱 91

表次

表 3-1 本教學課程學習內容以及對應所需先備知識 20
表 3-2 自我效能量表面向與敘述 22
表 3-3 學習目標 23
表 3-4 教學工具之功能 24
表 3-5 教學流程設計對照表 26
表 3-6 後設認知鷹架於教學課程中之教學設計對照表 27
表 3-7 兩者鷹架引導方式之差異對照表 29
表 3-8 研究問題與資料分析對照表 32
表 4-1 學習成效前後測同質性檢定表 34
表 4-2 學習成效前後測敘述性統計分析表 34
表 4-3 學習成效單因子共變數分析表 35
表 4-4 自我效能前後測同質性檢定表 35
表 4-5 自我效能前後測敘述性統計分析表 36
表 4-6 自我效能單因子共變數分析表 36
表 4-7 自我效能與學習成效相關性檢定分析表 37

圖次

圖 1-1 研究流程圖 5

圖 3-1 研究架構圖 17

圖 3-2 教學工具功能介面 24

圖 3-3 實施階段詳細流程圖 30

第一章 緒論

本章緒論旨在說明此研究的背景與動機，並確定研究目的來形成研究問題，以及研究包含的範圍。此章共分為五節來進行敘述：第一節闡述「研究背景與動機」、第二節建立「研究目的與問題」、第三節訂定「研究範圍與限制」、第四節為規劃「研究流程」、第五節為本研究的「名詞釋義」。

第一節 研究背景與動機

現今時代背景下，程式教育為多數國家的教育發展重點。我國於 108 課綱之科技領域中正式將資訊科技課程納入國、高中必修課程之一（教育部，2018）。然而，程式設計教育課程對於初次接觸程式，或是設計程式經驗較少的學習者來說是具有挑戰性的。抽象的程式概念及陌生的語法，造成學習者在學習中產生挫折感，進而影響學習者的學習表現（Abdullahi, Salleh, Nordin, & Alwan, 2018），因此，教學方式的設計在程式設計課程中扮演很重要的角色。傳統教學程式設計之方式為講述式，由教學者講解程式概念及指令用法，再針對課程內容進行測驗。然而程式設計需要學生的邏輯思維與演算法設計的思維，在傳統講述式之教學方式中，缺乏將編寫程式的思維方式融入於教學中，造成多數學習者在傳統教學中無法獲得程式寫作之思維模式（Kiss & Arki, 2017）。為解決上述問題，學者們透過架設學習鷹架來輔助學習者學習，其中後設認知鷹架（meta-cognitive scaffolding）最為廣泛運用於程式設計課程中（Prather, Pettit, Becker, Denny, Loksa, Peters, Albrecht, & Masci, 2019; Zhou, Chen, Zhou, Bao, & Li, 2021; Mohd Rum & Ismail, 2017; Denny, Prather, Becker, Albrecht, Loksa, & Pettit, 2019）。程式設計的邏輯為一種程序性思維（Papert, 1980），且涉及複雜的資料結構、語法與套件，每個概念皆環環相扣，先備知

識背景會影響新概念的學習。而後設認知鷹架能輔助學習者喚起學習過的知識（Hill & Hannafin, 2001），讓學習者能透過先備知識來對新知識進行推論理解，此功能正能輔助學習者學習程式設計概念（Bernard & Bachu, 2015）。然而後設認知鷹架的缺點為缺乏系統性之引導（Prather, Pettit, McMurry, Peters, Homer, & Cohen, 2018），由於後設認知鷹架多以提示、提問的方式來引導學習者學習，並沒有邏輯系統性的根據新知識內容來引導學習者喚起學習過的知識，因此造成學習者根據提示，自行將已知知識與新知識連結。然而每個人的理解能力不同，對於新概念的理解也會有所落差，因此造成學習表現會出現明顯的差距。而當學習者因為無法理解程式內容進而影響學習，便會產生排斥學習心理，以及對程式設計學習產生消極的信念，亦會影響學習者的信心，降低自我效能（Durak, 2018）。為了讓學習者能有邏輯性的建構程式設計概念，學者們試圖運用運算思維的概念，提供學習者一個思考的框架來協助學習，目的為讓學習者能有效理解程式概念，也希望學習者能將程式編寫的思維方式運用在解決其他問題。

運算思維為一種解決問題的思維過程，在解決問題的過程中能有效率的使用現有的資源來解決它，因此，運算思維能力的基礎是「解決問題」的能力（Kılıç, Gökoğlu, & Öztürk, 2020）。近年來，多數國家將運算思維視為國家教育重點，而運算思維與計算機科學有著密切的關聯性，可以透過程式教育來培養運算思維邏輯（Sysło & Kwiatkowska, 2013; Bocconi, Chioccariello, Dettori, Ferrari, & Engelhardt, 2016; Sun, Hu, & Zhou, 2021），此論點也已達到基本共識。教育相關組織也紛紛投入大量心力於運算思維教育中，亦有推動運算思維教育的計畫出現。美國國際社會教育科技組織（International Society for Technology in Education, ISTE）和電腦科學教師協會（Computer Science Teachers Association, CSTA）是推動運算思維教育的兩大主力，提供教育工作者、學生免費的學習資源（CSTA & ISTE, 2011）。香港賽馬會慈善信託基金成立「CoolThink 賽馬會運算思維教育」計畫，旨在發展一套系統性的運算思維教

育課程並實施於義務教育中，截至 2020 年至 2021 年間已有 168 間學校參與此計畫（香港賽馬會，2016）。紐西蘭教育部（Te Kete Ipurangi, TKI）亦於 2017 年在數位科技科的課綱中，將運算思維列入其中，目標是希望學生能應用運算思維於理解數字技術的計算機科學原理，進而將程式設計概念與計算機科學之應用整合，成為數字技術的創造者（TKI, 2017）。在我國的 108 課綱之科技領域中，資訊科技與生活科技為其中兩大學科，而運算思維為學習表現中的一大構面，目標是希望透過教育課程來培養學生運算思維之解決問題的能力（教育部，2018）。由上述可得知，運算思維與程式設計的相關性，將運算思維應用在程式設計教學上具備合理性依據，且運算思維特性能彌補傳統教學的缺點。

除了教學方式為影響程式設計學習成效的因素，研究發現學習者對於程式學習的自我效能也會影響學習成效表現（Gedeon & Valliere, 2018）。當在學習程式的過程不順利，或是遇到困難未能解決，會使學習者對於程式設計學習產生負面、消極的感知，學習的困境會使學習者產生挫折感，影響著學習者的自我效能（Toharudin, Rahmat, & Kurniawan, 2019），因此除了需提升程式設計學習成效，亦需將提升自我效能作為程式設計教育的目標。基於上述，本研究除了探討程式設計學習成效，亦將學習者的自我效能作為探討的變項之一。

綜合上述研究背景與動機，本研究欲改善程式學習的困境與程式設計教學上缺乏邏輯系統性引導的不足，研究者試圖提出一具有運算思維流程的鷹架引導方式。利用運算思維流程來引導學習者思考，協助學習者建構一個分析問題到解決問題的思維模式，並探討此教學策略對於學習者的學習成效與自我效能之影響。希冀能在程式設計教育中給予教學者另一種教學方式的選擇。

第二節 研究目的與問題

基於研究背景與動機、文獻探討之討論，本研究欲探討後設認知鷹架中，不同的鷹架引導方式，對於學習者學習程式設計之學習成效與自我效能的影響。其中運算思維流程鷹架引導方式之引導內容為依運算思維流程四步驟來設

計對應的引導內容，傳統教學鷹架引導方式之引導內容為依程式敘述的順序來設計引導內容。以下將研究目的與研究問題分列敘述。

一、研究目的

(一)在架設後設認知鷹架於程式設計課程中，不同的鷹架引導方式（運算思維流程、傳統教學鷹架引導方式），對於學習者學習成效的影響。

(二)在架設後設認知鷹架於程式設計課程中，不同的鷹架引導方式（運算思維流程、傳統教學鷹架引導方式），對於學習者自我效能的影響。

(三)探討在架設後設認知鷹架於程式設計課程中，不同的鷹架引導方式（運算思維流程、傳統教學鷹架引導方式），對於學習者學習成效、自我效能之間的關聯性。

二、研究問題

根據研究目的，形成以下研究問題：

1. 在架設後設認知鷹架於程式設計課程中，不同的鷹架引導方式（運算思維流程、傳統教學鷹架引導方式），對於學習者學習成效之差異為何？
2. 在架設後設認知鷹架於程式設計課程中，不同的鷹架引導方式（運算思維流程、傳統教學鷹架引導方式），對於學習者自我效能之差異為何？
3. 在架設後設認知鷹架於程式設計課程中，不同的鷹架引導方式（運算思維流程、傳統教學鷹架引導方式），對於學習者學習成效、自我效能間之相關性為何？

第三節 研究範圍與限制

本研究根據研究背景與動機、文獻探討與研究方法之討論內容，對於本研究的研究範圍與研究限制分列敘述如下：

一、研究範圍

本研究之教學課程為 108 課綱之科技領域中的普通高中之程式設計，以程式判斷結構 if...與 if...else...作為課程主題，故其他教學課程不在本研究範圍內。

二、研究限制

本研究的研究對象為普通型高中一年級學生，教學課程時間為一節課。故本研究最終結果與討論不宜過度推論至其他學科主題的教學內容設計、教學方法使用、其他年齡層的學習者與長時間的學習成效之研究。

第四節 研究流程

本研究之研究流程圖如圖 1-1 所示：

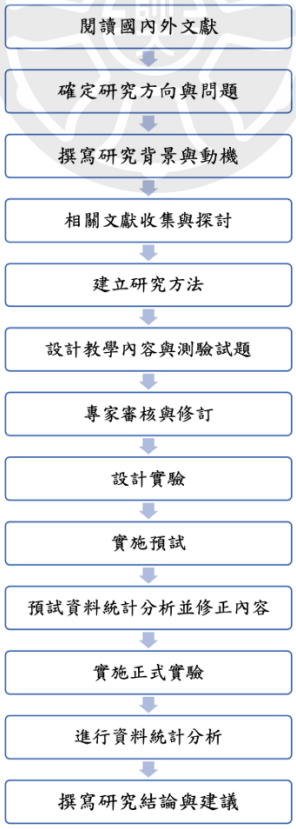


圖 1-1 研究流程圖

第五節 名詞釋義

一、運算思維（Computational Thinking）

運算思維是一種思考模式，亦是解決問題的方法之一（Wing, 2006）。運算思維有四個步驟，根據 British Broadcasting Corporation （BBC Bitesize, 2019）提出的運算思維四步驟（Decomposition, Pattern Recognition, Abstraction, Algorithm）各自的定義，其中 Abstraction 的解釋為「focusing on the important information only, ignoring irrelevant detail」，因此在本研究中，將運算思維四步驟分別命名為「問題拆解、規律辨識、摘要化、步驟化」。以下為每一步驟在本研究中所代表的意涵：

1. 問題拆解（Decomposition）：確定程式需要判斷的目標後，再根據目標來列出需滿足的條件有哪些。
2. 規律辨識（Pattern Recognition）：從需達到的條件中來設計這些條件的成立與否，會分別產生的結果有哪些。
3. 摘要化（Abstraction）：依照判斷結構的程式敘述，將設計的條件各自對應的結果代入，來組成一個判斷結構。
4. 步驟化（Algorithm）：依照程式執行流程，將判斷條件、欲達成的判斷目標依序代入其中，再透過程式語法設計一個完整的程式來解決問題。

二、自我效能（Self-efficacy）

自我效能為個人對於自己能否達到預期結果的感知程度，並進而影響個人的表現。在本研究中，自我效能是指學習者在「自我效能量表」的得分，分數愈高，即表示對於自己能完成程式寫作的感知與信心愈佳。

三、學習成效（Learning Outcome）

學習成效指的是學習者在經過課程學習前後的能力變化。在本研究中，學習成效指的是學習者在學習課程前後，分別在「學習成效測驗試卷」的得分，分數愈高，即代表程式設計能力愈完整，學習成效愈佳。

第二章 文獻探討

本章文獻探討針對研究相關的文獻進行統整與分析，藉以建立完整的研究架構。全章分為「程式設計學習之困境」、「運算思維」、「鷹架理論」、「自我效能與程式設計學習成效」與「文獻探討小結」五節來依序進行詳述。

第一節 程式設計學習之困境

程式設計教育已成為多數國家積極發展的目標，然而對學習者來說，學習程式設計課程是具有挑戰性的。程式設計的學習是一項涉及廣泛的專業知識，並且需要一次性的掌握概念，才能順利編寫出一可執行的程式（Kadar, Mahlan, Shamsuddin, Othman, & Wahab, 2022），因此學習者對於編寫程式感到困難，因為他們難以理解程式語法、對於程式的除錯與演算法的編寫都存在問題（Islam, Sheikh, Fatima, & Alvi, 2019）。

Xinogalos（2016）提出學習者在學習程式設計時所面臨的困境有演算法的開發設計、將想法轉化為程式語法、程式結構、模組化的理解、測試與除錯，因為在缺乏解決問題的能力及對程式概念理解的並不確實，所以造成學習者學習成績不佳，放棄學習，然而這些困難會對學習者心理產生消極影響，進而影響學習成效，因此如何提升程式設計學習之成效是電腦科學教育面臨的一個巨大挑戰（Silva, Mendes, Gomes, & Cavalcanti-De-Macedo, 2021）。

由於學習者不熟悉程式語法與結構，無法理解編譯器給予的除錯提示訊息，導致學習程式設計的過程中面臨重重困難。缺乏解決問題的能力是造成學習者難以學習程式的主要原因之一（Chung, Chou, Hsu, & Li, 2016），因此在程式設計的教學除了概念的教授外，還需要讓學習者能透過實作來學習如何應用這些程式概念，且教學目標應為讓學習者能編寫出含自己的想法的程式（Alshaye,

Tasir, & Jumaat, 2019）。程式設計為一複雜性高的認知過程（Kong, Lai, & Sun, 2020），因此在教學程式設計語法的內容前，應先引導學習者該如何透過編寫程式來解決問題的思維方式，需讓學習者透過不斷的實作練習來解決問題。

綜合上述，程式設計對學習者來說一直以來都是一門具有挑戰性的科目，除了需理解複雜的知識內容，還需具備多項能力才能編寫一完整的程式。學習者在程式設計學習遇到的困難，也造成學習者對於程式設計產生消極態度。因此需透過教學策略與學習內容的設計，來改善學習者在程式設計學習上所遇到的種種困難，是程式設計教育致力發展的目標。

第二節 運算思維

本研究置入運算思維流程於學習鷹架中，故探討運算思維之意涵。此章節探討運算思維的定義，再針對運算思維教育的發展概況進行說明。以下分為「運算思維定義與特性」及「運算思維教育的發展概況」兩部份來詳述。

一、運算思維定義與特性

Papert（1980）將程序性思維（procedural thinking）一詞引入程式設計教育中，為日後運算思維一詞發展奠定了基礎。程序性思維與學習理論的「建構主義」有密切關係。周以真（Jeannette M. Wing）教授於 2006 年為運算思維（Computational Thinking）一詞定義開創性之解釋，認為運算思維即是透過設計程式來解決問題之能力，從理解問題到拆解問題，再將問題解決的能力就是運算思維(Wing, 2006)。此能力包含解決問題、系統設計、理解人類的行為等元素。透過運算思維來解決問題的過程雖然是以電腦科學的角度去思考，但 Wing（2008）認為運算思維的應用並不侷限於電腦科學領域裡，應將此「解決問題」之思考模式應用於任何問題。Wing（2011）為運算思維之內涵新增「人」的思考視角，認為運算思維是一種以人或電腦的角度去解決問題的思考

過程。Wing（2017）將運算思維之定義具體化，認為運算思維是以人、電腦或機器可有效執行的角度，去理解問題並思考解決方案的思維過程。學者們也開始探討運算思維的組成要素，美國國際社會教育科技組織（ISTE）和電腦科學教師協會（CSTA）提出運算思維的操作性定義，將運算思維分為創造力、演算法思考、批判性思考、問題解決、合作等五元素（CSTA & ISTE, 2011）；Google（2015）將運算思維分為抽象化、演算法設計、拆解、模型識別、自動化、資料分析、資料收集、資料表示、模擬、平行化、樣式一般化等十一個元素。運算思維概念源自於電腦科學中程式設計之內涵，因此學者們多以電腦科學角度來思考運算思維。拆解（Decomposition）、模型識別（Pattern Recognition）、抽象化（Abstraction）、演算法（Algorithm）等四元素為運算思維之核心架構。British Broadcasting Corporation（BBC Bitesize, 2019）將此四元素定義：

1. 拆解（Decomposition）：

將複雜的問題簡單化。遇到問題時須先將理解問題產生的來落去脈後，理解問題後再來擬定解決辦法。

2. 模型識別（Pattern Recognition）：

將複雜的問題分解出的每個小問題分別檢視，尋找每個小問題的發生與其他問題發生的類似之處，也就是每個小問題間的規律，或是固定發生之模式為何。

3. 摘要化（Abstraction）：

透過模型識別找出每個細小問題間的規律後，找出這些細小問題間的關鍵點。如問題發生的先後順序，發生的情境元素有哪些等關鍵點，最後將這些細小問題間的關鍵點歸納出並設計一解決方法。

4. 步驟化（Algorithm）：

結合上述的三元素，依擬定的解決方法來解決問題。

二、運算思維教育的發展概況

根據上節次「運算思維定義與特性」的探討，運算思維意涵源自於電腦科學領域中程序性思維，運算思維一詞的解釋由抽象至具體化之發展過程是以此概念為主軸，再從中細分出要素。因此，運算思維成為各國教育發展重點後，運算思維教育課程多與程式設計結合。在運算思維融入於程式設計的課程中，「實作」在教學過程中佔據極大部分，藉以實踐運算思維教育之目標－「解決問題的能力」。Almjally, Howland and Good（2020）研究指出在運算思維課程中，學習者在實作活動中能「實際」將學習內容輸出，過程中能增加學習者在課堂裡的參與度，進而提升學習表現。程式設計課程之實作活動使用的學習工具有 Scratch、Alice、Lego、code.org 及機器人（NAO、bee-bot），學習者在實作活動所設計的程式視為學習成效的表現。實作活動設計之目的為希望學習者透過設計程式來實際應用運算思維之概念，藉以培養學習者能將運算思維之思考模式用於解決問題（Shen, Chen, Barth-Cohen, Jiang, & Eltoukhy, 2020; Angeli, 2020; Georgiou, 2019）。因此，將運算思維融入於程式設計之課程中，教學者應在教學內容與教學活動之過程，強調運算思維之特性與應用（Zakaria & Iksan, 2020）。

第三節 鷹架理論

本研究架設學習鷹架於程式設計課程中，故須瞭解鷹架理論之意涵。本節除了探討鷹架理論的定義，也探討教學中使用的鷹架類型及其功能。接著針對學習鷹架在程式設計教學中的應用進行說明。以下分為「鷹架理論之定義與特性」、「鷹架在教學應用的類型與功能」與「學習鷹架在程式設計教學的應用」三部分進行詳述。

一、鷹架理論之定義與特性

鷹架（Scaffolding）理論源自於 Vygotsky（1978）的社會發展論，認為個體透過適當的教學輔助下，能提早具備獨立解決問題的能力」。接著 Vygotsky（1978）提出「近側發展區（zone of proximal development, ZPD）」的概念，所謂的近側發展區即是個體獨自解決問題的成效至經由旁人指導幫助下後解決問題的成效，這之間的距離即近側發展區。「鷹架理論」為近側發展區的概念延伸，鷹架一詞為 Wood, Bruner and Ross（1976）提出，認為鷹架在學習過程中扮演引導、支持學習者的角色，而學習者在鷹架的輔助下能獲得更高的學習成就。Bruner（1985）將鷹架理論融入教學策略中，將近側發展區視為教學目標所在。Bruner 提出使用鷹架的教學策略有「解釋」、「吸引學生參與」、「教師與學生相互證明與釐清」、「示範」、「合作討論」等五種策略。Azevedo and Hadwin（2005）提出鷹架應用在電腦科學教育的情境有「示範」、「概念的學習」、「科技輔助教學」、「師生論辯」、「學生自我學習能力」與「後設認知」。當學習者有能力能自行完成指定任務或解決問題時，就不需要有鷹架的支持，因此教學者應在教學過程中觀察學習者的表現，並依據學習者的學習狀況，逐漸將鷹架拆除（Doo, Bonk, & Heo, 2020）。

綜合上述，鷹架應架設於學習者的進側發展區內，能引發學習者的學習動機，進而提升學習表現，並在學習者有能力能獨立完成任務時將鷹架拆除。因此如何有效的搭建鷹架於教學過程中，教學者扮演著重要的角色（吳昀臻，2020）。

二、鷹架在教學應用的類型與功能

Hannafin, Land and Oliver（1999）將鷹架分為四種類型：

（1） 概念性鷹架（conceptual scaffolding）

協助學習者認識學習的主題與相關知識。

（2） 後設認知鷹架（meta-cognitive scaffolding）

協助學習者監控與反思學習過程。

（3） 策略性鷹架（strategic scaffolding）

提供方法幫助學習者在執行任務時能順利。

（4） 程序性鷹架（procedural scaffolding）

協助學習者運用學習資源與使用學習工具。

Kim, Cho and Kim（2018）研究發現，後設認知鷹架對於學習者在科學、技術、工程與數學學習所產生的學習成效大於策略性鷹架。他們認為後設認知鷹架比策略性鷹架所要求的門檻更高。Hill and Hannafin（2001）指出後設認知鷹架的優點在於能喚起學習者學習過的知識，幫助他們在學習過程中建構知識，因此後設認知鷹架能幫助學習者理解學習的概念、提升解決問題的能力與自我監控，適用於高度認知的學習課程（An, 2010; Chen & Chan, 2011）。

三、學習鷹架在程式設計教學的應用

一個可執行的程式，其內容涉及許多不同的知識與能力。對於程式初學者來說，運用程式的結構來設計一解決方法，並將解決方法透過程式碼來實踐的一問題解決流程，是極具挑戰性的（Cheah, 2020）。由於在程式設計課程中，讓學習者透過參考例題來學習的教學方法，獲得良好的學習成效，其次是透過練習來學習程式設計的教學方法（Brown & Wilson, 2018），因此學者們為改善學習者在學習程式設計時所遇到的困難，試圖透過架設鷹架來輔助學習者學習程式設計，設計不同類型的鷹架並應用在程式設計教學中，來探討學習者學習程式設計的表現。

Cabo（2018）將程式執行的流程圖作為程式設計學習的鷹架，認為將解決問題的方法與編寫程式碼分開來進行教學，是一種能有效幫助學習者融入程式設計課程的方法。Dawar and Murphy（2020）提出 AAAD 學習框架作為程式設計課

程學習的鷹架，研究結果發現，透過不斷的練習能提高學習者的學習成效。Jin, Sun, Ma and Wang（2021）研究發現透過架設兩種不同形式的鷹架於 PBL 教學方法，能有效的架設鷹架並將複雜的內容分解，此策略對於學習者來說會更容易的將問題順利解決。

綜合上述，鷹架應用於程式設計教學的策略為提供學習者一個思考如何解決問題的方向，且目標皆為協助學習者理解抽象程式概念，並提高學習成效。

第四節 自我效能與程式設計學習成效

Bandura（1994）認為自我效能為「一個人相信自己有能力可以達到預設目標的感知為何，且自我效能感會影響自己在事件中的表現」（Bandura, 1994）。為了要瞭解影響學習者的程式設計學習成效之因素為何，已有許多研究探討程式設計的學習成效與自我效能之間的關係。研究發現，自我效能會影響程式設計學習成效表現（Lishinski, Yadav, Good, & Enbody, 2016; Lishinski & Yadav, 2021）。簡言之，學習者對於程式設計學習的信念及感知會影響學習者學習程式設計。當學習者在學習程式設計過程中遇到困難並未順利解決，那將會使學習者抱持著消極、負面的信念及感知，進而影響學習者的學習成效（Robins, 2019; Cheah, 2020）。自我效能對於程式設計學習來說，是一個影響學習成效甚大的因素。當一個自我效能高的學習者遇到困難時，會表現出積極的學習行為，進而促進更高的學習成效表現（Psycharis & Kallia, 2017）。因此，除了有研究探討影響學習者學習程式設計的自我效能之因素，亦有學者研究如何在教學過程中提升學習者對於程式設計學習的自我效能，並試圖應用在教學策略中。研究結果發現學習者在程式設計課程中，自我效能會在編寫程式的過程裡不斷發展（Gorson & O'Rourke, 2020; Ortiz, Chiluiza, & Valcke, 2017; Kanaparan, Cullen, & Mason, 2019）。由於自我效能為影響程式設計學習成效的因素之一，因此學者們為電腦科學領域設計了評量學習者自我效能的方法，教

學者依照學習內容的難易度來循序漸進地給予學習任務，透過學習者在學習過程中的表現或是學習成果來評量學習者自我效能感的變化（Bhardwaj, 2017; Özyurt & Özyurt, 2018）。除此之外，學者們也針對程式設計的學習困境來開發自我效能量表，並依不同年齡層來設計對應的量表內容，藉以讓學習者自我評量對於學習程式設計以及在學習過程中能自行解決問題的信心感知程度（Kong, 2017; Kukul, Gokcearslan, & Gunbatar, 2017; Tsai, Wang, & Hsu, 2018; Kukul & Karatas, 2019; Steinhorst, Petersen, & Vahrenhold, 2020）。

綜合上述，對於複雜性高的程式設計科目來說，在發展教學策略與教學內容設計時皆應考量學習者的自我效能，將提升學習者的自我效能納入發展重點之一。

第五節 文獻探討小結

根據上述文獻的探討分析，了解到學習者在學習程式設計時，有不易理解概念與難以將想法透過程式語法來完整表達的困境，進而影響學習成效，也因此造成學習者心理對於程式設計產生消極的信念。許多研究致力於提升程式設計的學習成效，針對程式設計學習的困境來發展有效的教學策略，如何有邏輯性的幫助學習者建構程式概念與培養程式寫作思維，為教學者須思考的問題。為解決程式學習困境，現今程式教育多融入運算思維之概念來設計教學策略，學者們也在教學中透過架設學習鷹架，提供學習者思考的方向藉以輔助學習者學習，其中後設認知鷹架之喚起已知知識之特性與程式設計之程序性思維特性相輔相成，因此被廣泛應用於程式教育中。除了程式設計學習的困境會影響學習成效外，自我效能亦是影響學習成效的因素之一。研究發現，學習者經常根據他們對程式編寫過程的期望來評估自身的程式設計能力，進而影響學習的表現。因此教學者除了以提升學習成效為目標外，也須考量學習者的自我效能。

綜合上述，本研究選擇探討後設認知鷹架中其不同的鷹架引導方式，對於學習者的學習成效與自我效能的影響。

第三章 研究設計

本章為針對本研究的研究設計，根據研究動機、目的與結合文獻探討之內容後設計出學習內容、檢定學習成效之方法。本章節分為「研究架構」、「研究方法」、「研究對象」、「研究工具」、「教學設計」、「研究實施」、「資料處理與分析」七節，來完整此研究進行。

第一節 研究架構

本研究綜合文獻探討之內容，及根據研究目的與問題來設計本研究之研究架構。下圖 3-1 為本研究之架構圖，並詳述各項目。

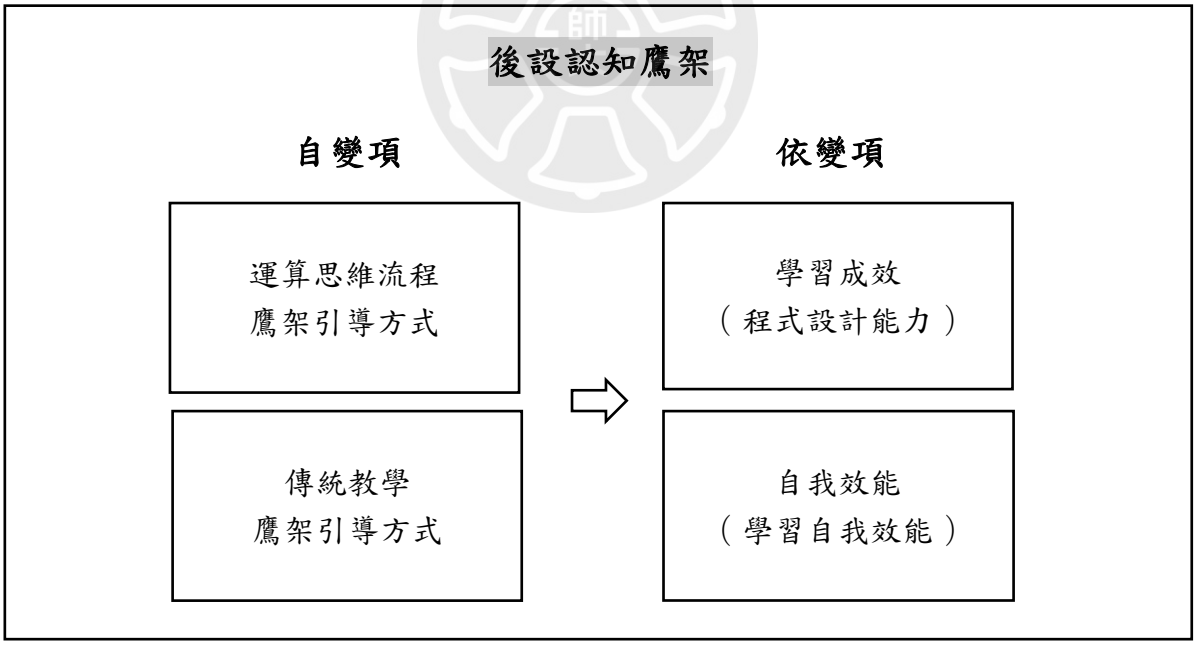


圖 3-1 研究架構圖

一、自變項

本研究之自變項為後設認知鷹架中之引導方式，分為「運算思維流程鷹架引導方式」與「傳統教學鷹架引導方式」等兩個層級。實驗組使用運算思維流程鷹架引導方式，對照組使用傳統教學鷹架引導方式，教學設計之詳細內容於第五節教學內容設計。

二、依變項

本研究之依變項為「學習成效」與「自我效能」。學習成效為瞭解學習者對於教學內容的理解程度，自我效能為瞭解學習者對於程式設計學習的信心感知程度，因此分別在實驗課程前後進行學習成效測驗與自我效能量表填寫，再針對學習者填答內容分別進行評分與統計分析，檢驗兩者不同的鷹架引導方式對於學習成效與自我效能的影響。

三、學習內容

本研究的教學主題為「判斷結構：if…與 if…else…程式設計」，實驗組與對照組的學習內容無差異。學習內容由「python.org」對此主題之說明及收集 python 教學書籍後，進行歸納統整與調整並由專家進行審查與修訂。

四、實驗時間

實驗組與對照組所安排之課程與測驗的時間皆無差異，課程時間皆為一節課五十分鐘，不包含測驗時間。課程前後進行的學習成效測驗與填寫自我效能量表的時間約為五十分鐘。

五、教學工具

實驗組與對照組所使用的教學工具無差異，教學工具皆為 Python(3.8.8)、Python IDE_Spyder(4.2.5)與課程練習題目。

第二節 研究方法

本研究使用的研究方法為實驗設計法的「準實驗」方法，並搭配「半結構式訪談」法來訪談參與本教學實驗之專家與學習者，在經過本教學實驗後的想法與感受，以及對於本課程之想法與建議。透過收集質性資料來進一步了解專家與學習者的想法與感受，藉以輔助分析量化研究結果之原因。

在準實驗方法中，透過實驗組、對照組的設計來探討架設後設認知鷹架於程式寫作課程中，使用不同的鷹架引導方式之教學策略，對於學習者學習程式寫作的自我效能與學習成效之影響。在實施教學實驗前，給予全體學習者學習成效測驗試題，來進行前測。教學實驗結束後，進行學習成效測驗之後測，藉以檢測學習者在教學實驗前後的學習成效之變化與差異，並徵求自願受訪的學習者來進行訪談。訪談前會先擬定好訪談大綱，包含訪談範圍與題目，而在訪談過程中，會根據受訪者的回應來適時調整訪談的內容。

第三節 研究對象

本研究對象的選擇為 39 位補習班之普通高中一年級學生，雖然全體學生不是皆來自同所學校，但在資訊科技課程規劃方面，上、下兩學期皆需修習資訊科技之必修課程。課程授課方式為傳統教學方式，教學者講授學習內容，測驗目的為檢測對於程式概念的理解程度，因此本研究之研究對象皆具有基礎程式概念的先備知識與學習經驗。

因考量全體學習者在學習本教學課程前的先備知識程度不均，進而影響學習成效，故課程的先備知識內容置入於教學講義中，並於實作示範前進行複習，詳細的教學講義內容於附錄二教學教材。下表 3-1 為本教學課程的學習內容與所需先備知識之對照表。

表 3-1

本教學課程學習內容以及對應所需先備知識

學習主題	學習內容	先備知識
1. 認識判斷結構	1. 判斷結構的基本概念 2. 認識日常生活中的判斷結構應用	1. 語法規則－變數命名 2. 資料型態（int） 3. 輸出指令（print） 4. input( )函式 5. 算術運算子（加、減、乘、除、餘數運算） 6. 邏輯運算子（和、或） 7. 關係運算子（大於、小於、大於等於、小於等於、不等於）
2. 判斷結構程式設計	1. 判斷結構的程式敘述	

第四節 研究工具

根據研究問題與目的以及研究設計，本研究的研究工具分別為「學習成效測驗試卷」、「自我效能量表」與「訪談大綱」三項，以下將分列詳述。

一、學習成效測驗試卷

學習成效為分析學生在課程前後測之成績表現。本研究學習成效之前後測試卷為研究者針對課程內容進行出題，並請三位專家進行試題內容審查與修訂，以建立本試卷之專家效度並進行預試，詳細學習成效測驗試卷內容於附錄四。試卷共計五題判斷結構程式的實作題，學習者需根據題意，使用 Spyder 來編寫出能執行題目要求的判斷結構程式。

五題實作題的程式內容為評量學習成效的依據，故研究者邀請 2 位電腦科學領域專家來共同成立評分小組，透過預試的全體學習者之作答內容，共同討論應將哪些項目列為本測驗之評分項目，並訂定評分的標準為何。評分標準訂定完成後，將給予評分小組每人五份程式內容錯誤皆不同的試卷進行批改，確定每位專家的評分標準皆一致，以建立本評分表之專家效度。評分項目為依程式敘述的輸入、輸出以及判斷結構來分出十項項目，每個項目皆為一分，只要該項目有錯誤即為零分。詳細學習成效測驗評分表內容於附錄六。

二、自我效能量表

自我效能為評量學習者在課程前後對於學習程式設計的信心感知程度。本研究使用的自我效能量表改編自 Steinhorst, Petersen and Wahrenhold（2020）提出的程式導論課程自我效能量表，針對本研究不適用的題目與面向進行調整與修改後，請專家進行評估與修正，建立量表的專家效度。此量表在正式實驗前進行預試，問卷之 cronbach’s α 值為 0.938，表示有良好可信度。 本量表共計 25 題，使用李克特量表（Likert Scale）之七點計分的填答模式，分別為「非常不同意」、「不同意」、「有點不同意」、「普通」、「有點同意」、「同意」與「非常同意」，來探討學習者對程式設計學習的自我效能。 下表 3-2 為量表的面向與題型敘述內容，完整量表內容於附錄五。

表 3-2

自我效能量表面向與敘述

面向	敘述	題數
理解程式執行流程	對於程式執行的流程是否了解？是否有信心能依程式執行流程來編寫程式？	6
使用結構和模組來解決問題	對於程式結構是否了解？能使用程式結構或是模組概念來解決問題嗎？	5
堅持、除錯、解決問題的能力	即使編寫程式過程中困難重重，也有信心編寫程式嗎？	6
控制程式執行流程	在編寫程式的過程中，是否能應用程式執行流程於程式編寫的過程裡？	3
其他	在學習過程中看到陌生的程式術語時，能勇於學習嗎？	5

三、訪談大綱

為更進一步了解不同鷹架引導方式對於學習者學習程式寫作之自我效能與學習成效的影響，輔以訪談參與本教學實驗之專家與學習者，收集質性資料來輔助說明研究結果。本研究使用半結構式訪談法來設計訪談內容，訪談大綱之內容經由指導教授來進行審核與修訂，詳細的訪談大綱內容於附錄七。

第五節 教學內容設計

本研究以「判斷結構：if...與 if...else...程式設計」作為教學主題。課程學習內容與學習成效測驗試題，皆給予三位資訊教育科技領域專家來進行審查與修訂。此課程的學習內容包含教學工具使用方式、程式判斷結構概念理解與實作。教學工具為 Python(3.8.8)、Python IDE_Spyder(4.2.5)與課程練習題，透過 Spyder IDE 來實作所學之程式設計概念。以下將整個學習內容分成「學習目標」、「教學工具」與「學習流程」三項進行詳述。

一、學習目標

課程內容為「判斷結構：if...與 if...else...程式設計」，依據程式設計之意涵來訂定學習目標，並依據學習目標來規劃學習內容，如下表 3-3 所示。

表 3-3

學習目標

學習者在本研究之教學課程，有以下三個學習目標：

1.	能了解「if...與 if...else...」2 種判斷結構的概念。
2.	能正確應用「if...與 if...else...」2 種判斷結構的程式敘述。
3.	能設計 1 個具有判斷結構的程式。

二、教學工具

課程之教學方式為口頭講授與搭配教學教材，教學者進行例題示範與學習者實作練習題。課程練習題共有三題程式判斷結構的實作題，並在題目卷中給予不同鷹架引導方式所設計的內容來協助學習者解題。完整課程練習題內容於附錄三。

由於本研究欲了解在架設後設認知鷹架中，不同的鷹架引導方式對於學習者學習程式設計之學習成效與自我效能影響，故程式設計之內容為重要學習表現依據，因此採用 Spyder 做為教學工具。Spyder 之程式編輯區域簡單明瞭，不僅能清楚看到完整之程式碼，也能立即顯示程式執行結果。學習者在實作過程中使用 Spyder 的區塊依序為「程式編輯區域」、「執行程式之按鈕」、「程式執行結果區域」。以下表 3-4 為本教學課程會使用的功能與方式，圖 3-2 為各功能介面示意圖。

表 3-4

教學工具之功能

功能名稱	使用方式
1. 程式編輯區域	輸入指令的區域
2. 執行程式按鈕	按下此按鈕，即會執行
3. 程式執行結果區域	承 2，指令執行之結果會顯示於此區域

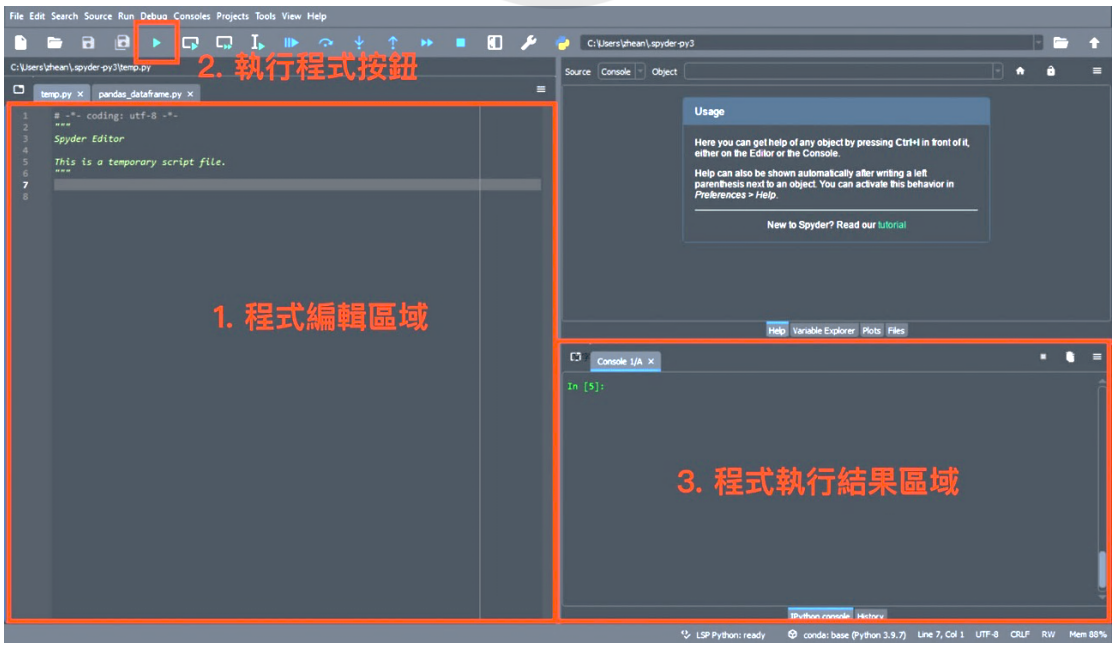


圖 3-2 教學工具功能介面

三、學習流程

本教學課程之目標為讓學習者能更有邏輯系統性的學習程式寫作，教學者講述課程內容並搭配教材，與進行實作示範後讓學習者實作練習之教學方式，讓學習者理解程式概念與指令用法後，能透過實際操作來應用所學的概念。教學主題為「判斷結構：if...與 if...else...程式設計」，研究者將教學活動內容與步驟寫為課程教案，課程教案分為課程概述與教學活動步驟兩部份，詳細內容於附錄一。

本研究欲了解架設後設認知鷹架於程式設計課程中，給予不同的鷹架引導方式對於學習者的自我效能與學習成效之影響，故在教學過程中會設計不同的鷹架引導方式之教學策略，來協助學習者學習程式設計，並在課程的前後進行學習成效測驗與自我效能量表的填寫。實驗組、對照組的學習份量相同，皆學習相同的教材，使用相同的教學工具，兩組的差異在於使用不同的教學策略。教學課程與學習成效測驗皆在電腦教室進行，以下分為「學習流程設計」與「不同鷹架引導方式內容設計」兩部分進行詳述。

（一）學習流程設計：

教學課程開始前，給予全體學習者課程講義、練習題目及練習題之電子檔案與實作練習之執行檔案，並進行課程說明，說明內容包含學習目標、學習流程及教學工具 Spyder 之使用方式。教學活動分為：(1) 概念講解：程式判斷結構的基本概念與程式敘述指令教學；(2) 實作示範：依運算思維流程四步驟來依序進行程式判斷結構的編寫示範；(3) 實作練習：在練習題目卷中，對照組使用傳統教學鷹架引導方式，實驗組使用運算思維流程鷹架引導方式來設計對應的引導內容，協助學習者實作。鷹架引導內容位於練習題目之前，以直接螢幕呈現的方式輔助學習者依題目順序編寫程式於執行檔案中。以下表 3-5 為實驗組與對照組之教學流程對照表，詳細的學習內容於附錄一教學課程教案。

表 3-5

教學流程設計對照表

流程項目	實驗組 運算思維流程 鷹架引導方式	對照組 傳統教學 鷹架引導方式
概念講解	1. 講解程式判斷結構的基本概念，包含單項結構與多項結構概念，並以日常生活中判斷結構應用作為教學範例。 2. 講解判斷結構 if...與 if...else...程式敘述與指令用法。	
實作示範	依運算思維流程四步驟所設計的判斷結構程式寫作思維之教學策略來進行實作示範，詳細的實作示範之教學內容於附錄一教學課程教案，以下說明運算思維流程四步驟各自對應的教學內容。 1. 問題拆解 解題前，教學生思考程式要判斷的目標為何，根據目標來列出所需要的條件有哪些。 2. 規律辨識 列出判斷目標所需的條件後，教學生思考每一個條件的成立與否，各自對應的結果為何。 3. 摘要化 教學生依每個條件各自對應的結果，來組成一個判斷結構。 4. 步驟化 教學生依照程式執行流程，將欲達到的結果、判斷條件依序代入其中，再使用程式語法來設計程式，完成解題。	
實作練習	在程式判斷結構實作題目卷中給予依運算思維流程四步	在程式判斷結構實作題目卷中給予依程式敘述順序設計

	驟設計的鷹架引導內容，引導內容於「（二）不同鷹架引導方式之引導內容設計」進行詳述。	的鷹架引導內容，引導內容於「（二）不同鷹架引導方式之引導內容設計」進行詳述。

（二）不同鷹架引導方式之引導內容設計：

後設認知鷹架在本教學課程之目的為協助學習者學習程式設計，並在實作時能更容易完成程式寫作。因此本研究架設後設認知鷹架於「實作練習」教學活動，在此教學活動會給予學習者四題判斷結構程式實作題來進行程式寫作練習，學習者需透過課堂所學的 if...與 if...else...判斷結構的程式設計內容，依題目要求來設計程式，詳細練習題內容於附錄三。

在練習題目卷中使用「運算思維流程鷹架引導」與「傳統教學鷹架引導」兩種鷹架引導方式來設計對應的鷹架引導內容，並透過設計「自由設計判斷結構程式」之題目，讓學習者能在鷹架拆除後，獨立進行程式寫作。下表 3-6 為後設認知鷹架在本教學課程中之實驗組與對照組的教學設計對照表。

表 3-6

後設認知鷹架於教學課程中之教學設計對照表

組別 / 項目	實驗組	對照組
架設鷹架目的	引導學習者進行程式寫作	
鷹架架設時機	教學活動：判斷結構程式設計－實作練習	
鷹架引導方式	運算思維流程鷹架引導	傳統教學鷹架引導

鷹架拆除時機	在課程練習題中，設計一題「自由設計判斷結構程式」之題目，並提供參考答案供學習者參考。

本研究之自變項為「傳統教學鷹架引導方式」與「運算思維流程鷹架引導方式」兩種鷹架引導方式，以下分為「傳統教學鷹架引導方式」與「運算思維流程鷹架引導方式」來分列詳述。

（一）傳統教學鷹架引導方式:

本教學課程的傳統教學鷹架引導方式指的是依照程式敘述之順序來設計引導內容。因此傳統教學鷹架引導方式為將課程內容重點依判斷結構程式敘述順序來設計引導內容，協助學習者回顧課程所學，能更容易完成學習。對照組在課程「實作練習」教學活動中，以傳統教學鷹架引導方式來協助學習者實作程式寫作。

（二）運算思維流程鷹架引導方式:

運算思維為一解決問題流程，透過此流程（問題拆解→規律辨識→摘要化→步驟化）來思考問題進而有效地解決問題。因此在運算思維流程鷹架引導方式，將根據運算思維流程的四步驟來設計對應的引導內容，給予學習者具有邏輯順序性的引導，協助學習者完成程式設計學習。實驗組在課程「實作練習」教學活動中，以運算思維流程鷹架引導方式來協助學習者實作程式寫作。表 3-7 為兩者鷹架引導方式之差異對照表。

表 3-7

兩者鷹架引導方式之差異對照表

<table>
<tr><td rowspan="2">鷹架引導內容</td><td>實驗組
運算思維流程
鷹架引導方式</td><td>對照組
傳統教學
鷹架引導方式</td></tr>
<tr><td>作答前你可以想想下列問題：
題目要求程式需判斷什麼？
課程重點回顧：
1. 確定使用者所輸入的數值代表的意義為何，以及電腦讀取數值後，會顯示哪些不同的結果。 → 問題拆解
2. 依據不同的顯示結果，列出判斷此結果的條件各自為何。 → 規律辨識
3. 將判斷的條件代入判斷結構的程式敘述。在程式敘述中，需先寫判斷的條件式，接著寫條件的成立、不成立，顯示的結果各自為何。 → 摘要化
if(條件式):
條件成立，執行的程式區塊
if(條件式):
條件成立，執行的程式區塊
if(條件式):
程式區塊 1，條件成立，執行的程式區塊
else:
程式區塊 2，條件不成立，執行的程式區塊
4. 依據程式執行流程來設計程式，並使用程式語法來完成解題。
程式執行流程： → 步驟化
使用者輸入一數值→電腦讀取輸入的數值後，進入條件判斷的區塊→判斷條件成立、不成立，顯示對應的結果
-----確定好上述內容，使用程式語法來將你的想法輸出，完成解題吧！-----</td><td>作答前你可以想想下列問題：
題目要求程式需判斷什麼？
課程重點回顧：
1. 判斷結構 if…與 if…else…的程式敘述分別為：
if(條件式):
條件成立，執行的程式區塊
if(條件式):
條件成立，執行的程式區塊
→ 判斷結構程式敘述順序
if(條件式):
程式區塊 1，條件成立，執行的程式區塊
else:
程式區塊 2，條件不成立，執行的程式區塊
2. 程式執行流程為先讀取使用者輸入的值，再進行條件的判斷，來決定要執行哪一個程式區塊，並顯示對應的結果。
3. 在判斷結構的程式敘述中，需先寫判斷的條件式，再寫條件成立時，顯示的結果為何；條件不成立時，顯示的結果為何。
-----確定好上述內容，使用程式語法來將你的想法輸出，完成解題吧！-----</td></tr>
<tr><td>兩者差異說明</td><td colspan="2">兩者鷹架引導內容之差異在於設計策略不同。運算思維流程鷹架引導內容是依照運算思維流程四步驟來依序設計對應的引導內容；傳統教學鷹架引導內容為根據判斷結構程式敘述的順序來設計對應的引導內容。</td></tr>
</table>

第六節 研究實施

本研究以架設後設認知鷹架於程式設計課程中，透過不同鷹架引導方式在練習題中之作答提示，來幫助學習者理解程式設計概念，並能將其實際應用於

實作程式寫作。「學習成效前後測試卷」與「自我效能量表」等研究工具皆完成後，於實驗實施前與課堂老師討論教學時程的安排，確定實施時間後，開始進行教學實驗課程。實施階段詳細流程圖如下圖 3-3 所示。以下分為實驗前、實驗中與實驗後三部分來進行詳述。

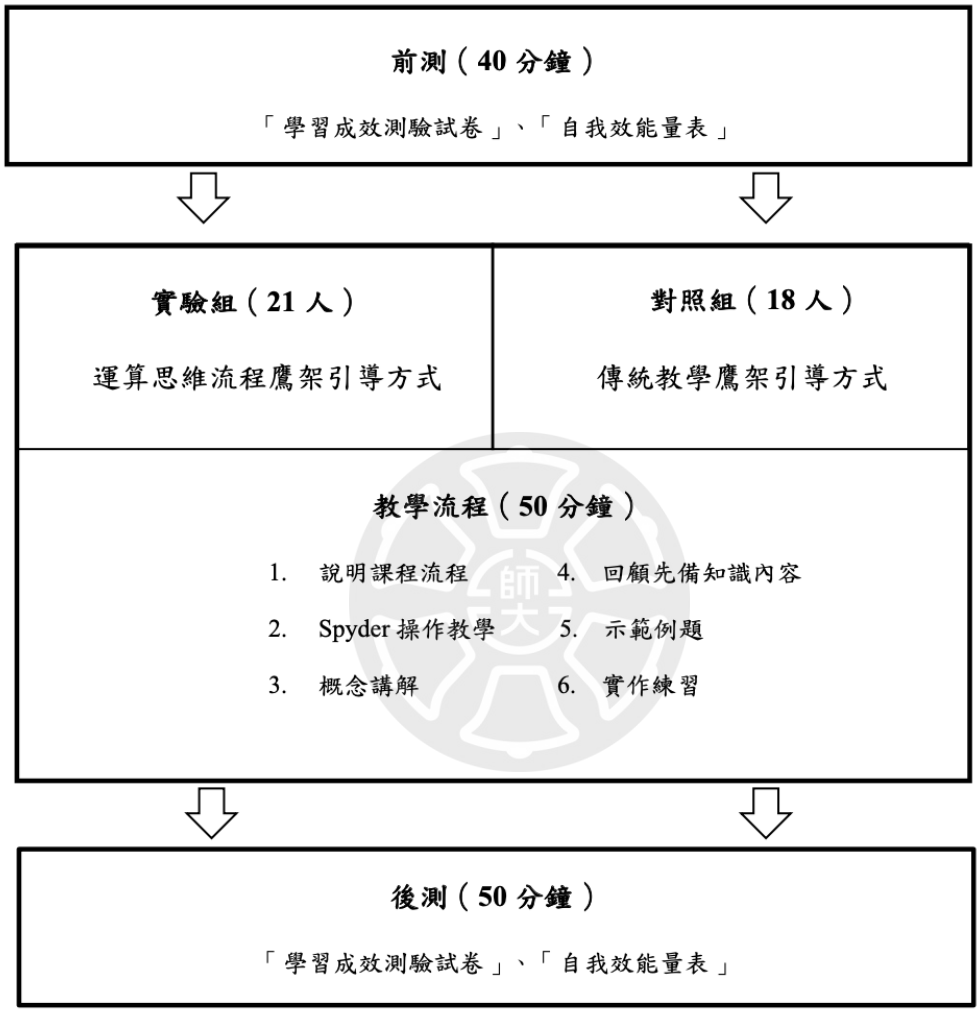


圖 3-3 實施階段詳細流程圖

一、實驗前

在實驗前，先與研究對象的老師聯絡，並說明教學課程內容及實施流程。協調實驗實施時間後，於實驗日前三天對研究對象說明教學課程內容，並發放「青少年參與意願書暨家長知情同意書」。在課程前一天，進行學習成效測驗

與自我效能量表之前測，時間約為五十分鐘。測驗結束後，為每位研究對象教學教學工具 Spyder 操作方式，並上機練習操作，時間約為三十分鐘。

二、實驗中

課程時間為一節課約五十分鐘，在實驗開始後，教學者依教案的規劃來授課並搭配講義，完整的教學講義內容於附錄二教學教材。學習者在實作練習題的過程中有任何問題皆可以詢問教學者，但不能與其他同學討論。

三、實驗後

教學課程結束會休息十分鐘後，再進行學習成效測驗與自我效能量表之後測，測驗時間約五十分鐘。學習成效測驗可參考教學講義與練習題內容，但不可與其他同學討論。測驗結束後，徵求願意受訪的學習者來進行訪談。

第七節 資料處理與分析

本研究使用實驗法來設計實驗，主要以量化資料分析為主。使用的研究工具包含：「學習成效前後測試卷」用以檢測學習者在教學課程前後的學習成效之差異；「自我效能量表」用以了解學習者在教學課程前後的自我效能感受差異；「訪談大綱」用以了解參與本教學實驗之專家與學習者的感受、想法與建議。以下表 3-8 為研究問題與資料分析對照表。

表 3-8

研究問題與資料分析對照表

研究問題	分析方式
1. 在架設後設認知鷹架於程式設計課程中，不同的鷹架引導方式，對於學習者學習成效之差異為何？	單因子共變數分析
2. 在架設後設認知鷹架於程式設計課程中，不同的鷹架引導方式，對於學習者自我效能之差異為何？	單因子共變數分析
3. 在架設後設認知鷹架於程式設計課程中，不同的鷹架引導方式，對學習者學習成效、自我效能間的相關性為何？	皮爾森積差相關分析

1. 單因子共變數分析

使用此分析方式是為了避免前測間接影響後測的結果，因此控制前測的影響，檢測兩組別之間在學習成效與自我效能分數的差異。

2. 皮爾森積差相關分析

檢測在架設後設認知鷹架於程式設計課程，使用不同的鷹架引導方式，對於學習者學習成效、自我效能間的相關程度為何。

3. 質性資料分析

依訪談大綱來訪談參與本教學實驗的專家與學習者，訪談內容全程進行錄音，並將內容逐字列稿。訪談內容以編碼的方式來進行分類，專家為 E，實驗組為 EG，對照組為 CG，第二碼為訪談順序，第三碼為訪談內容逐字稿中的段落數，最末碼為訪談內容逐字稿中的行數。

第四章 研究結果與討論

本研究旨在探討架設後設認知鷹架於程式設計課程中，不同鷹架引導方式對於學習者的學習成效與自我效能的影響。在本研究的教學課程中使用學習成效前後測試卷來檢測學習者在教學課程前後的學習成效表現，學習成效前後測試卷的評量標準採用學習成效測驗評分表，此評分表具有專家效度。而自我效能量表目的在於瞭解架設後設認知鷹架於程式設計課程中，不同鷹架引導方式對於學習者的自我效能影響。另外將學習成效表現與自我效能之數據結果進行相關性分析，探討這兩個變項間的關聯性為何。另外，本研究使用半結構式訪談之目的在於更深入了解專家與學習者在參與本研究之教學課程後的感受與建議，因此以質性資料來輔助，使研究結果更具體。

研究對象為普通型高中一年級學生共 39 人，使用傳統教學的鷹架引導方式之對照組為 18 人，使用運算思維流程的鷹架引導方式之實驗組為 21 人，兩組皆進行一堂課的教學課程。將數據資料進行統整後，共 39 筆有效數據，將有效的樣本之數據以統計軟體 IBM SPSS Statistics 26.0 進行數據的統計分析，並根據分析結果依序來回答每一個研究問題。在質性訪談資料中，共有 8 筆資料，實驗組與對照組各 3 位學習者，以及參與本教學課程的 2 位專家。

本章節研究結果與討論為根據量化與質性資料之分析結果，來詳細說明本研究結果。以下分為「不同鷹架引導方式之學習成效差異分析」、「不同鷹架引導方式之自我效能差異分析」、「自我效能與學習成效相關性分析」以及「訪談內容分析」四個節次來依序進行詳述。

第一節 不同鷹架引導方式之學習成效差異分析

本節為了解在架設後設認知鷹架於程式設計課程中，使用不同的鷹架引導方式對於學習者的學習成效差異。將實驗組與對照組的學習成效前後測分數進行單因子共變數分析，來比較兩組學習成效差異，並回答研究問題一：「在架設後設認知鷹架於程式設計課程中，不同的鷹架引導方式，對於學習者學習成效之差異為何？」以下表 4-1 為學習成效前後測同質性檢定表，表 4-2 為學習成效前後測敘述性統計分析表，表 4-3 為兩組學習成效單因子共變數分析表。

表 4-1

學習成效前後測同質性檢定表

F	*df1*	*df2*	*p*
.028	1	37	.867

表 4-2

學習成效前後測敘述性統計分析表

		N	*Mean*	*SD*
前測	實驗組	21	4.6476	0.31562
	對照組	18	4.3667	0.28491
後測	實驗組	21	8.1048	0.40801
	對照組	18	6.2111	0.44707

表 4-3

學習成效單因子共變數分析表

變異來源	*SS*	*df*	*MS*	*F*	*p*
分組組別	28.374	1	28.374	151.842	.000***
誤差	6.727	36	0.187		
總計	2080.56	39			

***p<.001

依據 4-1 的同質性檢定結果為不顯著（p=0.867>0.05），代表樣本具同質性，可以做進一步的比較分析。控制前測影響後進行單因子共變數分析的結果從表 4-2 與表 4-3 可看出，兩組的學習成效具有顯著差異（p=.000），代表不同鷹架引導方式對於程式設計學習之學習成效有顯著影響。

第二節 不同鷹架引導方式之自我效能差異分析

本節為了解在架設後設認知鷹架於程式設計課程中，使用不同的鷹架引導方式對於學習者的自我效能差異。將實驗組與對照組的自我效能前後測分數進行單因子共變數分析，來比較兩組自我效能差異，並回答研究問題二：「在架設後設認知鷹架於程式設計課程中，不同的鷹架引導方式，對於學習者自我效能之差異為何？」以下表 4-4 為自我效能前後測同質性檢定表，表 4-5 為自我效能前後測敘述性統計分析表，表 4-6 為兩組自我效能單因子共變數分析表。

表 4-4

自我效能前後測同質性檢定表

F	*df1*	*df2*	*p*
.098	1	37	.755

表 4-5

自我效能前後測敘述性統計分析表

		N	*Mean*	*SD*
前測	實驗組	21	4.0738	0.16017
	對照組	18	4.0917	0.16738
後測	實驗組	21	5.3881	0.18500
	對照組	18	4.7361	0.19838

表 4-6

自我效能單因子共變數分析表

變異來源	*SS*	*df*	*MS*	*F*	*p*
分組組別	4.103	1	4.103	109.154	.000***
誤差	1.353	36	0.038		
總計	1014.770	39			

***p<.001

依據表 4-4 的同質性檢定結果為不顯著（p=0.755>0.05），代表樣本具同質性，可以做進一步的比較分析。控制前測影響後進行單因子共變數分析的結果從表 4-5 與表 4-6 可看出，兩組的學習成效具有顯著差異（p=.000），代表不同引導方式對於程式設計學習之自我效能有顯著影響。

第三節 自我效能與學習成效相關性分析

本節為了解進行完本研究的教學課程後，學習者的自我效能與學習成效表現的相關性。將實驗組與對照組的自我效能與學習成效分數，進行相關性分析來檢測兩者之間的相關性，並回答研究問題三：「在架設後設認知鷹架於程式

設計課程中，不同的鷹架引導方式，對於學習者學習成效、自我效能之間的相關性為何？」以下表 4-7 為自我效能與學習成效相關性檢定分析表。

表 4-7

自我效能與學習成效相關性檢定分析表

		學習成效
自我效能	皮爾森(Pearson)相關係數	.847**
	p	.000
	N	39

依據表 4-7 的相關性分析結果顯示，自我效能後測與學習成效後測有顯著的相關性，且相關係數達到 0.847，表示進行本研究的教學課程後，學習者自我效能越高，對於學習成效表現也越佳。

第四節 訪談內容分析

本節為瞭解專家對於本課程之想法與建議，以及學習者在課程結束後的實際感受與想法。教學課程結束後，研究者邀請參與本課程的專家，以及徵求學習者來進行訪談，訪談內容分為「學習內容與教學策略設計」以及「個人感受與想法」兩部份來進行分析與詳述。

（一）學習內容與教學策略設計

在學習內容與教學策略設計的部分，受訪的專家皆表示使用鷹架引導方式的教學策略來教學程式設計課程，對於學習是有幫助的，也認為運算思維流程鷹架引導內容可以讓學習者在實作時獲得有邏輯性的指引。針對兩者不同鷹架

引導方式的差異，專家認為運算思維流程鷹架引導內容相較於傳統教學鷹架引導內容，更能夠引導學生如何思考一個問題的解決方法。傳統教學鷹架內容能讓學習者掌握課程內容重點後，從重點內容中梳理出一個解決問題的方法，而運算思維流程鷹架引導內容將課程內容重點依運算思維流程之步驟來列出，提供一個解決問題的方法之思考模式。

使用鷹架引導方式來教程式肯定是會有幫助的，你給學生作答的提示，給他們一個思考的方向，他們在實作的時候，會比較知道該如何下手。

學生在上機實作寫程式的時候，很常會卡住，不知道現在要先寫什麼指令。但其實寫程式就是一個解決問題的過程，只是需要以程式碼來呈現的。（E-1-2-3）

運算思維的鷹架引導內容對於學習程式是會有幫助的，就像我剛剛說的，寫程式就是一個解決問題的過程，而運算思維就是一個有邏輯性的去分析問題，深入了解問題後，再來解決問題。所以把運算思維置入在鷹架中，去引導學生是可以幫助學生學習程式的。我覺得這個教學策略也值得去嘗試應用在其他課程中。（E-1-3-3）

兩者引導內容在內容設計上是不同的。傳統教學的引導內容就像教科書的重點整理，將每個單元的重點條列出，讓學習者能快速抓到學習的重點。（E-2-2-3）

運算思維的引導內容是提供學習者一個思考的模式，學生在依序看完引導內容後，能有一個完整的思路，而傳統教學的引導內容需要學生自己去理出一套程式寫作的邏輯。（E-2-3-4）

受訪的學習者表示，在過往學習程式寫作的經驗中，以老師講授為主，對於實作程式寫作相對較陌生，因此在實作程式寫作時所給予的引導內容是有幫

助的，也認為在課程中教學者先實作示範例題，再讓進行練習的教學流程設計有助於學習，能在程式寫作練習時能根據引導內容來回想教學者示範的過程。

我覺得練習題上的引導內容是有幫助的，我在做練習題前先看過引導內容能讓我比較知道該怎麼去寫練習題的程式。（CG-2-3-6）

對於寫程式我其實是滿害怕的，因為我覺得指令很難理解，這次在課堂上老師有先帶著我們示範做例題，我有試著去寫看看，所以在寫練習題的時候比較有進入狀況。（EG-1-2-3）

我在寫練習題的時候，還是有突然卡住的地方，一直出現錯誤，我有再回去看引導的內容，再來試著找出錯誤的地方，真的不行再詢問老師，所以我覺得練習上的引導內容對我來說是有幫助的。（EG-2-2-4）

我覺得練習題的引導內容，有讓我再複習一次課程內容的感覺，這樣在寫練習題的時候，就不會完全不知道該怎麼寫。（CG-3-3-4）

練習題的提示內容讓我在寫練習寫程式時，有一個參考的依據滿好的，這樣我在寫的過程中如果遇到問題時，就不會不知道該怎麼辦，因為我自己對於寫程式是滿沒有信心的。（EG-3-2-4）

課程中老師先示範例題給我們看後，再讓我們練習，且在練習題中有給我們課程重點內容，對我來說是有幫助的，我覺得我在練習寫程式的時候沒有自己預先設想的那麼困難。（CG-1-3-5）

（二）個人學習感受與想法

在個人學習感受與想法的部分，參與本教學實驗的專家認為使用鷹架引導方式來教學程式設計課程能提升學習者的信心，也提及到多數學習者對於程式設計較缺乏自信與興趣，因此除了學習內容的設計外，教學者使用的教學策略對於程式設計的學習也很重要。

學生在學習過程中所遇到的問題都能獲得解答，或是有資源可以讓他們去利用，那他們在學習上是會獲得信心的。在程式課程中，上機寫程式對大部分的學生來說都具有挑戰性的，加上他們對程式碼會覺得很難，不好懂，在學習上就會沒信心。（E-1-1-4）

這個教學策略對於學生來說是有幫助的，在實作的時候有引導內容的輔助，會讓學生在實作過程中遇到問題時，有找尋解答的方向，進而有成功寫出正確的機會，那自然而然的就能獲得信心。（E-2-1-3）

受訪的學習者皆表示在完成本教學課程的學習後，對於學習程式寫作能較有信心，未來在學習其他程式設計課程時也會試著去挑戰，也提及在課程前對於自己的表現較無自信，而在實際經過學習後，發現自己在編寫程式的表現比預期的好。

我覺得我自己是在學習的時候，概念都能理解，可是在應用部分就會不知道該怎麼實作出來，所以在課程中有老師的示範加上練習題有提示的內容，我在寫程式的時候，就沒有一開始就完全卡住。（EG-2-3-3）

這次在學習程式的過程中，比我想像的還順利，整體來說還不錯。我原本以為我會完全寫不出來，之後在學習其他的程式時，我會試著挑戰看看。（CG-3-2-5）

在寫程式的過程中，看到顯示錯誤都會很緊張，這次課程練習的過程中，出現錯誤無法解決時有獲得老師的幫助並順利完成。在測驗的時候也有遇到問題，自己有成功解決，整體來說有帶給我寫程式的信心。（EG-3-3-4）

我覺得在學習過程中的感覺滿順的，也對於自己最後在測驗的時候能自行寫出程式覺得有成就感。（CG-2-2-4）

課程結束後有覺得比較有信心，因為一開始對於要寫程式就有點怕，但在經過老師的示範加上練習題的提示內容，讓我在寫程式的過程中沒有覺得很慌亂。（CG-1-2-3）

在課程中我所遇到問題最後都有解決，有把程式寫出來覺得滿好的，雖然在測驗的時候，還是有卡住的地方，但在學習的過程中滿有成就感的。」（EG-1-3-3）

第五章 結論與建議

本研究為探討在架設後設認知鷹架於程式設計課程中，不同鷹架引導方式對於學習者的學習成效與自我效能的影響。透過文獻探討、教學實驗設計與實施及數據的分析結果，了解在架設後設認知鷹架於程式設計課程中，不同鷹架引導學習者回顧課程內容重點的方式，對於學習者學習成效與自我效能的影響差異。將研究結果整合為本研究之結論，本章節分為「研究結論」與「後續研究建議」兩個節次，將依序進行詳細論述。

第一節 研究結論

本研究欲了解架設後設認知鷹架於程式設計課程中，使用不同鷹架引導方式對於學習者的學習成效與自我效能的影響。以下將依據研究問題來逐一做出本研究之結論。

一、在架設後設認知鷹架於程式設計課程中，不同鷹架引導方式對於程式設計學習之學習成效有顯著影響

實驗組與對照組在進行教學課程後，後測的平均分數皆有顯著差異。實驗組在教學課程中，教學者會以運算思維流程來實作示範例題，及引導學習者在進行實作練習編寫程式時，能有邏輯系統性的回顧課程內容；對照組在教學課程中，教學者會以傳統教學方式來實作示範例題，與引導學習者回顧課程內容。因此可以得出：在架設後設認知於程式設計課程中，使用運算思維流程的鷹架引導回顧課程內容之方式，對於學習者學習成效表現優於使用傳統教學的鷹架引導方式之結論。

二、在架設後設認知鷹架於程式設計課程中，不同鷹架引導方式對於程式設計學習之自我效能有顯著影響

自我效能指的是學習者對於程式設計學習的信念及感知程度，分數越高即表示對於程式設計學習的信念及感知越佳，反之則越差。研究結果表示實驗組與對照組間的自我效能有顯著差異，因此可以得出：在架設後設認知於程式設計課程中，使用運算思維流程的鷹架引導回顧課程內容之方式，對於學習者自我效能的提升優於使用傳統教學的鷹架引導方式之結論。

三、在架設後設認知鷹架於程式設計課程中，不同的鷹架引導方式，對於學習者的自我效能感變化會影響學習成效的表現

自我效能為學習者對於程式設計學習的信念及感知程度。學習者在經過教學課程後，自我效能的變化是否會影響學習成效的表現是本研究欲探討的問題。研究結果表示，學習者的自我效能與學習成效有顯著相關性，因此可以得出：在架設後設認知於程式設計課程中，使用不同鷹架引導學習者回顧課程內容的方式，對於學習者的自我效能感變化會影響學習成效的表現之結論。

第二節 後續研究建議

本研究以進行實驗的方式來探討架設後設認知於程式設計課程中，使用不同鷹架引導學習者回顧課程內容的方式，對於學習者的學習成效與自我效能之影響。在進行本研究的過程中，研究者發現關於程式設計學習還有許多變項能做更進一步的探討，在程式設計的教學方式亦有能深入發展的空間。本節次將分為「延長教學時間」、「深入探討影響自我效能之因素」、「探討學習遷移對程式設計學習的影響」三個部分來依序進行詳述。

一、延長教學課程時間

本研究的教學課程時間不包含前後測的時間，完整的教學時間為一節課的時間。雖本研究依據課程的時間來設計適宜的學習份量，但若是要建立一個完整的運算思維之思考模式，可以安排以學期或學年為時間單位的課程規劃，來長期觀察學習者在學習過程中的表現，藉以有更全面性的探討。

二、探討影響自我效能之因素

「在架設後設認知鷹架於程式設計課程中，使用運算思維流程的鷹架引導回顧課程內容之方式，對於學習者自我效能的提升優於使用傳統教學的鷹架引導方式」為本研究結論之一。因此對於未來相關研究可以深入探討影響自我效能的因素為何，亦可增加質性資料，訪談學習者在程式設計學習的過程中，對於程式設計學習的想法感受，來更進一步的探討影響程式設計學習的自我效能之因素，為程式設計教學提出更多的教學策略與內容設計的參考方向。

三、探討學習遷移對程式設計學習的影響

本研究為解決程式設計學習上的困境，提出不同的教學策略，探討學習者的學習成效與自我效能。運算思維融入程式設計的教育目標在於學習者能將其解決問題的思維模式，應用至解決任何問題上。因此未來研究可將學習遷移作為程式設計學習的研究變項，探討學習者將所學之知識應用至解決課程外的問題之能力為何。

參考文獻

一、 中文部分

香港賽馬會（2016）。賽馬會運算思維教育。取自 https://www.coolthink.hk

教育部（2018）。**十二年國民基本教育課程綱要國民中學暨普通型高級中等學校－科技領域**。臺北：國家教育研究院。

吳昀臻、鄭雅婷（2020）。淺談鷹架理論與課程的效益。**臺灣教育評論月刊，9**(2)，69-73。

二、 外文部分

Azevedo, R., & Hadwin, A. (2005). Scaffolding self-regulated learning and metacognition: Implications for the design of computer-based scaffolds. *Instructional Science, 33*, 367-379.

An, Y. J. (2010). Scaffolding wiki-based, ill-structured problem solving in an online environment. *MERLOT Journal of Online Learning and Teaching, 6*(4), 723-734.

Abdullahi, M. S. I., Salleh, N., Nordin, A., & Alwan, A. A. (2018). Cloud-based learning system for improving students' programming skills and self-efficacy. *Journal of Information and Communication Technology, 17*(4), 629-651.

Alshaye, I., Tasir, Z., Jumaat, N.F. (2019). *The conceptual framework of online problem-based learning towards problem-solving ability and programming skills*. Proceedings of the 2019 IEEE Conference e-Learning, e-Management and e-Services (pp. 12-14), Penang, Malaysia.

Almjally, A., Howland, K., & Good, J. (2020). *Investigating children's spontaneous gestures when programming using TUIs and GUIs.* Proceedings of the 2020 ACM Interaction Design and Children Conference (pp. 36-48), London, United Kingdom.

Angeli, C., & Giannakos, M. (2020). Computational thinking education: Issues and challenges. *Computers in Human Behavior, 105*, 106185.

Bruner, J. S. (1985). *Child's talk: Learning to use language.* New York: Norton.

Bandura, A. (1994). Self-efficacy. *Encyclopedia of human behavior, 4*, 71-81.

Bernard, M., and Bachu, E. (2015). Enhancing the metacognitive skill of novice programmers through collaborative learning. *Metacognition: Fundaments, Applications, and Trends, 76,* 277-298.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). Developing computational thinking in compulsory education-implications for policy and practice. *Joint Research Centre (Seville site),* 3-68.

Bhardwaj, J. (2017). In search of self-efficacy: Development of a new instrument for first year Computer Science students. *Computer Science Education, 27*(2), 79-99.

Brown, N.C. & Wilson, G. (2018). Ten quick tips for teaching programming. *PLoS computational biology, 14*(4), 1-8.

BBC Bitesize. (2019). Computational thinking-KS3 Computing. Retrieved from https://www.bbc.co.uk/bitesize/topics/z7tp34j

CSTA & ISTE (2011). Operational definition of computational thinking for K-12 education. Retrieved from https://www.iste.org

Chen, K., Chan, A. H. S. (2011). A review of technology acceptance by older adults. *Gerontechnology, 10*(1), 1-12.

Chung, I. L., Chou, C. M., Hsu, C. P., & Li, D. K. (2016). *A programming learning diagnostic system using case-based reasoning method.* Proceedings of the IEEE International Conference on System Science and Engineering (ICSSE) (pp. 1-4), Puli, Taiwan.

Cabo, C. (2018). *Effectiveness of Flowcharting as a Scaffolding Tool to Learn Python.* Proceedings of the 2018 IEEE Frontiers in Education Conference (FIE) (pp. 1-7), San Jose, USA.

Cheah, C. S. (2020). Factors Contributing to the Difficulties in Teaching and Learning of Computer Programming: A Literature Review. *Contemporary Educational Technology, 12*(2), 272.

Denny, P., Prather, B., Becker, A., Albrecht, Z., Loksa, D., & Pettit, R. (2019). *A Closer Look at Metacognitive Scaffolding: Solving Test Cases Before Programming*. Proceedings of the 19th Koli Calling International Conference on Computing Education Research (pp. 1-10), Finland, Koli.

Doo, M. Y., Bonk, C., & Heo, H. (2020). A meta-analysis of scaffolding effects in online learning in higher education. *The International Review of Research in Open and Distributed Learning, 21*(3), 60-80.

Dawar, D., Murphy, M. (2020). An assignment a day: Scaffolded learning approach for teaching introductory computer programming. *Information Systems Education Journal, 18*(4), 59-73.

Google (2015). Exploring computational thinking. Retrieved from https://edu.google.com/resources/programs/exploring-computational-thinking/

Gedeon, S.A. and Valliere, D. (2018), "Closing the loop: measuring entrepreneurial self-efficacy to assess student learning outcomes". *Entrepreneurship Education & Pedagogy,1*(4), 272-303.

Georgiou, K., Angeli, C. (2019). *Developing Preschool Children's Computational Thinking with Educational Robotics: The Role of Cognitive Differences and scaffolding*. Proceedings of the 16th International Conference on Cognition and Exploratory Learning in Digital Age (PP. 101-108), Cagliari, Italy: IADIS Press.

Gorson, J., O'Rourke, E. (2020). *Why do CS1 Students Think They're Bad at Programming? Investigating Self-efficacy and Self-assessments at Three Universities*. Proceedings of the 2020 ACM Conference on International Computing Education Research (pp. 170-181), Virtual Event, New Zealand.

Hill, J., & Hannafin, M. (1997). Cognitive strategies and learning from the World Wide Web. *Educational Technology Research and Development, 45*(4), 37-64.

Hannafin, M., Land, S., & Oliver, K. (1999). Open learning environments: Foundations, methods, and models. *Instructional-design theories and models: A new paradigm of instructional theory, 2,* 115-140. Mahwah, NJ: Lawrence Erlbaum Associates.

Islam, N., Sheikh, G. S., Fatima, R., & Alvi, F. (2019). A Study of Difficulties of Students in Learning Programming. *Journal of Education & Social Sciences, 7*(2), 38-46.

Jin, Y., Sun, J., Ma, H., & Wang, X. (2021). *The impact of different types of scaffolding in project-based learning on girls' computational thinking skills and self-efficacy*. Proceedings of the 2021 Tenth International Conference of Educational Innovation through Technology (EITT) (pp. 1-6), Chongqing, China.

Kong, S. C. (2017). Development and validation of a programming self-efficacy scale for senior primary school learners. In S. C. Kong, J. Sheldon, & K. Y. Li (Eds.), Proceedings of the International Conference on Computational Thinking Education (pp. 97-102), The Education University of Hong Kong.

Kim, K., Cho, Y. K., & Kim, K. (2018). “BIM-Based Decision-Making Framework for Scaffolding Planning.” *Journal of Management in Engineering, 34*(6), 04018046-1-04018046-17.

Kanaparan, G., Cullen, R., & Mason, D. (2019). Effect of self-efficacy and emotional engagement on introductory programming students. *Australasian Journal of Information Systems,* 23, 1-24.

Kukul, V., Gokcearslan, S., & Gunbatar, M. S. (2017). Computer programming self-efficacy scale (CPSES) for secondary school students: Development, validation and reliability. *Educational Technology Theory and Practice, 7*(1), 158-179.

Kukul, V., & Karatas, S. (2019). Computational thinking self-efficacy scale: Development, validity and reliability. *Informatics in Education, 18*(1), 151-164.

Kılıç, S., Gökoğlu, S., & Öztürk, M. (2020). A Valid and Reliable Scale for Developing Programming-Oriented Computational Thinking. *Journal of Educational Computing Research, 59*(2), 257-286.

Kong, S. C., Lai, M., & Sun, D. (2020). Teacher development in computational thinking: Design and learning outcomes of programming concepts, practices and pedagogy. *Computers y Education*, *151*, 103872.

Kadar, R., Mahlan, S. B., Shamsuddin, M., Othman, j., & Wahab, N. A. (2022). *Analysis of Factors Contributing to the Difficulties in Learning Computer Programming among Non-Computer Science Students*. Proceedings of the 2022 IEEE 12th Symposium on Computer Applications & Industrial Electronics (ISCAIE)(pp. 89-94), Penang Island, Malaysia.

Lishinski, A., Yadav, A., Good, J., & Enbody, R. (2016). *Learning to program: Gender differences and interactive effects of students' motivation, goals, and self-efficacy on performance*. Proceedings of the 2016 ACM Conference on

International Computing Education Research (pp. 211-220), Melbourne, VIC, Australia: ACM.

Lishinski, A., Yadav, A. (2021). Self-Evaluation Interventions: Impact on Self-Efficacy and Performance in Introductory Programming. *ACM Transactions on Computing Education, 21*(3), 23:1-23:28.

Mohd Rum, S. N., & Ismail, M. A. (2017). Metocognitive Support Accelerates Computer Assisted Learning for Novice Programmers. *Educational Technology & Society, 20*(3), 170-181.

Ortiz, M., Chiluiza, K., & Valcke, M. (2017). *Gamification in computer programming: Effects on learning, engagement, self-efficacy and intrinsic motivation*. Proceedings of the 11th European Conference on Games Based Learning (ECGBL 2017) (pp.507-514), Graz, Austria.

Özyurt, H., & Özyurt, Ö. (2018). Analyzing the effects of adapted flipped classroom approach on computer programming success, attitude toward programming, and programming self-efficacy. *Computer Applications in Engineering Education, 26*(6), 2036-2046.

Papert, S. (1980). *Children, computers, and powerful ideas.* New York, NY: Basic Books.

Psycharis, S., & Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional Science, 45*(5), 583-602.

Prather, J., Pettit, R., McMurry, K., Peters, A., Homer, J., & Cohen, M. (2018). *Metacognitive Difficulties Faced by Novice Programmers in Automated Assessment Tools.* Proceedings of the 2018 ACM Conference on International Computing Education Research (pp. 41-50), New York, USA.

Prather, J., Pettit, R., Becker, B. A., Denny, P., Loksa, D., Peters, A., Albrecht, Z., & Masci, K. (2019). *First Things First: Providing Metacognitive Scaffolding for Interpreting Problem Prompts*. Proceedings of the 50th ACM Technical Symposium on Computer Science Education (pp. 531-537), New York, USA.

Robins, A. (2019). Novice programmers and introductory programming. *The Cambridge Handbook of Computing Education Research, Cambridge Handbooks in Psychology*, 327-376.

Sysło, M. M., & Kwiatkowska, A. B. (2013). *Informatics for all high school students.* Proceedings of the International conference on Informatics in Schools: Situation, Evolution, and Perspectives (pp. 43-56), Berlin, Heidelberg.

Shen, J., Chen, G., Barth-Cohen,L., Jiang,S., & Eltoukhy, M. (2020). Connecting computational thinking in everyday reasoning and programming for elementary school students. *Journal of Reasearch on Technology in Education*, 1-21.

Steinhorst, P., Petersen, A., & Vahrenhold, J. (2020). *Revisiting self-efficacy in introductory programming*. Proceedings of the 2020 ACM Conference on International Computing Education Research (ICER'20) (pp.158-169), Virtual Event, New Zealand.

Silva, L., Mendes, A., Gomes, A., & Cavalcanti-De-Macedo, G. (2021). *Regulation of Learning Interventions in Programming Education: A Systematic Literature Review and Guideline Proposition*. Proceedings of the 52nd ACM Technical Symposium on Computer Science Education (pp. 13-20), Virtual Event, USA.

Sun, L., Hu, Linlin., & Zhou, D. (2021). Improving 7th-graders'computational thinking skills through unplugged programing activities: A study on the influence of multiple factors. *Thinking skills and creativity*, *42*, 2-15.

Te Kete Ipurangi. (2017). Digital Technologies and the national curriculum. Retrieved from https://www.tki.org.nz

Tsai, M. J., Wang, C. Y., & Hsu, P. F. (2018). Developing the computer programming self-efficacy scale for computer literacy education. *Journal of Educational Computing Research, 56*(8), 1345-1360.

Toharudin, U., Rahmat, A., & Kurniawan, I. S. (2019). The important of self-efficacy and self-regulation in learning: How should a student be? *Journal of Physics: Conference Series, 1157*(2), 1-6.

Vygotsky, L. S. (1978). Mind in society : The development of higher psychological processes. *Cambridge, MA: Harvard University Press*.

Wood, D. J., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry, 17*, 89-100.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.

Wing, J. M. (2008). Computational thinking and thinking about computing. Philosophical Transactions on the Royal Society A: Mathematical, *Physical and Engineering Sciences, 366*(1881), 3717-3725.

Wing, J. M. (2011). Research Notebook: Computational Thinking--What and Why? The Link Magazine. The magazine of the Carnegie Mellon University School of Computer Science. Retrieved from http://people.cs.vt.edu/~kafura/CS6604/Papers/CT-What-And-Why.pdf

Wing, J. M. (2017). Computational thinking's influence on research and education for all. *Italian Journal of Educational Technology, 25*(2), 7-14.

Xinogalos, S. (2016). Designing and deploying programming courses: Strategies, tools, difficulties and pedagogy. *Education and Information Technologies, 21*(3), 559-588.

Yildiz Durak, H. (2018). Digital story design activities used for teaching programming effect on learning of programming concepts, programming self-

efficacy, and participation and analysis of student experiences. *Journal of Computer Assisted learning, 34*, 740-752.

Zakaria, I. N., Iksan, Z.H. (2020). Computational Thinking among High School Students. *Universal Journal of Educational Research, 8*(11A), 9-16.

Zhou, P., Li, J., Chen, F., Zhou, H., Bao, S., & Li, M. (2021). *Design of metacognitive scaffolding for k-12 programming education and its effects on students' problem solving ability and metacognition*. Proceedings of the 2021 Tenth International Conference of Educational Innovation through Technology (EITT) (pp. 182-186), Chongqing, China.

附錄一 教學課程教案

壹、教案概述

科目/領域別	資訊科技/科技領域	課程名稱	判斷結構：if...與 if...else...程式設計
教學對象	普通型高中一年級學生	教學時數	共 2 節課
教學設備	個人電腦、網路		
教學摘要	本課程的內容為程式判斷結構 if...與 if...else...的程式設計，在教學流程中會以運算思維四步驟（問題拆解、規律辨識、摘要化、步驟化）來進行，並在課程練習題中以不同方式來引導學習者回顧課程內容重點，完成程式編寫。期許學生能將本課程所學的程式設計流程應用於解決任何問題上。		
教學目標	1. 了解判斷結構的概念及應用 2. 了解判斷結構的程式敘述 3. 能設計一具有判斷結構的程式		
先備知識	1. 語法規則－變數命名 2. 資料型態（int） 3. 輸出指令（print） 4. input( )函式 5. 算術運算子（加、減、乘、除、餘數運算） 6. 邏輯運算子（和、或） 7. 關係運算子（大於、小於、大於等於、小於等於、不等於）		

<table>
<tr><td>運算思維</td><td colspan="2">在本教學內容中，運算思維四步驟各自代表的意義為：
1. 問題拆解（Decomposition）：確定程式的判斷目標（期望達到的結果為何），再根據判斷的目的來列出對應的條件式有哪些。
2. 規律辨識（Pattern Recognition）：從條件式中來設計條件式的成立與否，各自對應的結果有哪些。
3. 摘要化（Abstraction）：依照設計的條件式各自對應的結果，來組成一個判斷結構。
4. 步驟化（Algorithm）：依照程式執行流程，將判斷條件、欲達到的結果依序代入其中，來設計程式。</td></tr>
<tr><td rowspan="2">與課綱綱要對應</td><td>學習內容</td><td>1. 資 A-V-2 重要演算法的概念與應用：

演算法結構中的「判斷結構（if...與 if...else...）」概念與應用。

2. 資 P-V-3 重要演算法的程式設計實作：

演算法結構中的「判斷結構（if...與 if...else...）」程式設計實作。</td></tr>
<tr><td>學習表現</td><td>1. 運 t-IV-4 能應用運算思維解析問題</td></tr>
</table>

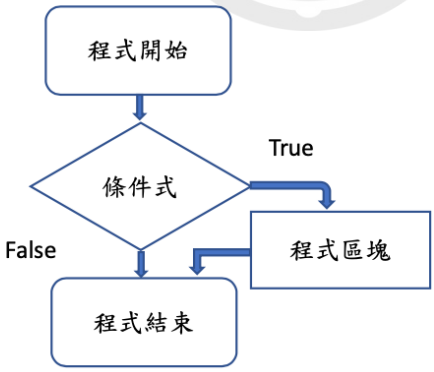
		2. 運 t-V-2 能使用程式設計實現運算思維的解題方法

貳、教學活動步驟

活動一		
一、 認識判斷結構 在設計程式時，必須將問題細分為多個小步驟，來讓電腦依序的執行。判斷結構在程式設計中會依據指定條件的成立與否，來判斷是否讀取程式來執行下一步。使用判斷結構能設計一具有判斷能力的程式。		
教學活動	**活動內容**	**教材**
判斷結構的基本概念	1. 判斷結構的敘述中，需有條件式的設計，條件式可以不只為一項，可以是多項條件，電腦依判斷結構的條件式，進行單向、雙向及多向的判斷。 2. 多項條件範例：在購物的過程中，決定要不要購買某項商品時，需要考慮的條件很多（外觀特徵、價錢、耐用度等），消費者會依照這些條件有沒有符合自身需求，來判斷是否購買此商品。 3. 單項條件範例：（1）設定鬧鐘時，鬧鐘會在使用者設定的時間準時響起，其餘時間並不會有任	簡報、講義

	何聲響。（2）某家商店的會員制度為當年消費滿額即可成為 vip 會員，未達指定金額即是一般會員。	
認識日常生活中的判斷結構應用	1. 給予一生活情境，讓學生討論回答這些情境為單向還是雙向判斷。 2. 範例：（1）天氣預報發布當週的下雨機率，你認為當天下雨機率只要為 50%以上（含）就需帶傘出門。（2）當手機電量低於 20%時，螢幕上的電池會呈現紅色，高於 20%則呈現綠色。（3）考試成績，未達 60 分為不及格，必須參加補考。	

活動二
二、 判斷結構的程式設計 本課程講授 if...與 if...else...兩種判斷結構，判斷結構程式碼在 python 語言中，條件式後須加冒號「：」，且執行程式區塊需有縮排（四個空白字元），以此來區分為同一個執行程式區塊。講解完判斷結構的程式敘述後，將會使用 spyder 來示範兩種判斷結構的程式設計，最後會提供練習題讓學生亦使用 spyder 進行實作練習。

教學活動	活動內容	教材
判斷結構的程式敘述講解	1. 判斷結構 if...為單向的判斷，當條件式成立時，電腦會執行程式區塊；條件不成立時，電腦會執行判斷結構 if...後的程式區塊。 2. 判斷結構 if...的程式語法為： if(條件式): □條件成立，執行的程式區塊 if(條件式): □條件成立，執行的程式區塊 （1） 判斷結構 if...的條件式可以是單一個或多個。 （2） 條件式後需加冒號「：」。 （3） □為執行的程式區塊需要縮排四個空白字元的部分。 3. 判斷結構 if...的流程圖為：  4. 判斷結構 if...else...為雙向的判斷，此判斷結構一定有兩個執行的程式區塊。當條件成立時，電腦會執行程式區塊內的程式；條件不成立時，電腦會執行另一個程式區塊內的程式。	簡報、講義

5. 判斷結構 if...else...的程式語法為：

if(條件式):

□程式區塊 1，條件成立，執行的程式區塊

else:

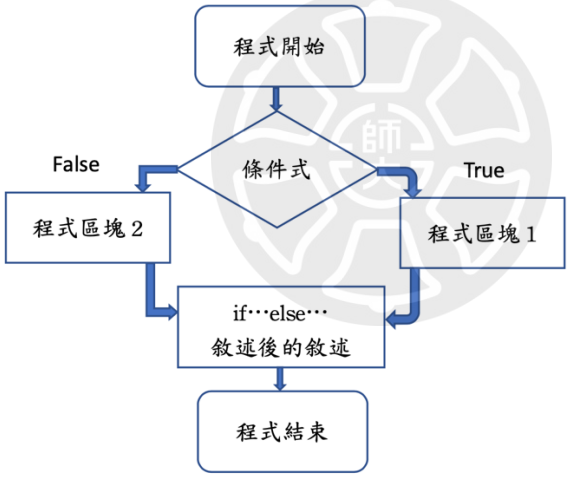
□程式區塊 2，條件不成立，執行的程式區塊

（1） 判斷結構 if...else...一定會執行一個程式區塊 1 或 2，電腦會判斷條件式是否成立，來執行一個程式區塊。

（2） 條件式後需加冒號「：」。

（3）□ 為執行的程式區塊需要縮排四個空白字元的部分。

6. 判斷結構 if...else...的流程圖為：



判斷結構程式設計－實作示範

1. 在實作示範的教學活動中皆會示範兩種判斷結構的程式設計方式。教學者會依運算思維流程的四步驟來進行解題示範，表(一)為運算思維流程四步驟各自的教學方式。

運算思維四步驟	實作示範－教學方式
1. 問題拆解	解題前，教學生思考程式要判斷的目標為何，根據目標

		來列出所需要的條件有哪些。
2. 規律辨識	列出判斷目標所需的條件後，教學生思考每一個條件的成立與否，各自對應的結果為何。
3. 摘要化	教學生依據每一個條件各自對應的結果，來組成一個判斷結構。
4. 步驟化	教學生依照程式執行流程，將欲達到的結果、判斷條件依序代入其中，再使用程式語法來設計程式，完成解題。

▲表(一) 實作示範－教學方式

2. 示範例題 1：今有一程式可以判斷使用者輸入的數字為偶數或奇數。請根據圖(一)程式執行結果，設計一具有判斷結構的程式。

圖(一)為使用者輸入數字「28」後的執行結果，電腦回應輸入的數字「28 是偶數」。

```
請輸入一個你想要判斷的數字（整數）:28
數字 28 是偶數

In [22]:
```

▲圖(一)程式執行結果

(1) 使用 if...判斷結構來設計的程式碼：

```
num=int(input('請輸入一個你想要判斷的數字(整數):'))
if (num%2 ==0):
    print('數字 %d 是偶數' %num)
if (num%2 != 0) :
    print('數字 %d 是奇數' %num)
```

(2) 使用 if...else...判斷結構來設計的程式碼：

```
num=int(input("請輸入一個你想要判斷的數字(整數):"))

if (num%2==0):

    print("%d 是偶數" %num)

else:

    print("%d 是奇數" %num)
```

(3) 表(二)為例題 1 依運算思維四步驟進行的教學內容

運算思維四步驟	示範例題 1－教學內容
1. 問題拆解	1. 判斷的目標：

			判斷使用者輸入的數字為奇數還是偶數。如果是奇數，將顯示「數字 xx 是奇數」；如是偶數則會顯示「數字 xx 是偶數」。 2. 條件： （1）偶數為 2 的倍數 （2）奇數為不能被 2 整除的數		
		2. 規律辨識	列出條件的成立與否，各自顯示的結果為何。 判斷偶數的條件：2 的倍數，如果此條件成立，會顯示「數字 xx 為偶數」；此條件不成立，即代表不是 2 的倍數，則會顯示「數字 xx 為奇數」。		
		3. 摘要化	根據判斷條件與顯示的結果來組成一判斷結構。 兩種判斷結構的程式敘述皆需先寫判斷的條件式，接著寫條件的成立與否，各自顯示的結果。		

			（1）判斷結構 if...程式敘述：
			
			if(條件式):
			　　條件成立，執行的程式區塊
			if(條件式):
			　　條件成立，執行的程式區塊
			
			（2）判斷結構 if...else...程式敘述：
			
			if(條件式):
			　　程式區塊 1，條件成立，執行的程式區塊
			else:
			
			　　程式區塊 2，條件不成立，執行的程式		

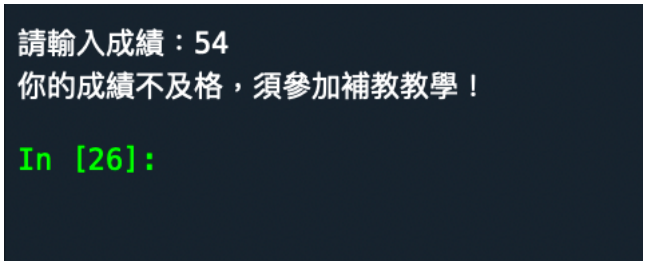
		4. 步驟化	1. 程式執行流程： （1）使用者輸入一數值 （2）電腦讀取輸入的數值後，進入條件判斷的區塊 （3）判斷的條件有兩個，當判斷偶數的條件成立，會		

	顯示該數為偶數；條件不成立，則會顯示該數為奇數。 2. 依據程式執行流程來設計程式，並使用程式語法來完成解題。

▲表（二）示範例題 1－教學內容

3. 示範例題 2：學校規定學期成績低於 60 分的學生必須參加補救教學課程。今有一程式可以判斷使用者輸入的成績是否及格。如判斷為及格，會顯示「恭喜你的成績及格！」；判斷為不及格，則會顯示「你的成績不及格，須參加補教教學！」。請根據圖（二)程式執行結果，設計一具有判斷結構的程式。

圖（二)為使用者輸入成績「54」後的執行結果，電腦回應輸入的成績 54 為「你的成績不及格，須參加補救教學！」。



▲圖（二)程式執行結果

（1） 使用 if…判斷結構來設計的程式碼：

```
score=int(input("請輸入成績："))
if (score >= 60):
```

```
    print('恭喜你的成績及格!')
print('你的成績不及格，須參加補教教學！')
```

（2） 使用 if…else…判斷結構來設計的程式碼：

```
score=int(input("請輸入成績："))
if (score >= 60):
    print('恭喜你的成績及格!')
else:
    print(('你的成績不及格，須參加補教教學！'))
```

（3）表（三）為例題 2 依運算思維四步驟進行的教學內容

運算思維四步驟	示範例題 2－教學內容
1. 問題拆解	1. 判斷的目標： 判斷使用者輸入的成績是否及格。如果成績及格，將顯示「恭喜你的成績及格！」；如果成績不及格，將顯示「你的成績不及格，須參加補救教學！」。 2. 條件： （1）成績高於 60 分（含）代表及格 （2）成績低於 60 分代表不及格

		2. 規律辨識	列出條件的成立與否，各自顯示的結果為何。 判斷成績及格的條件：成績須高於 60 分（含），如果此條件成立，會顯示「恭喜你的成績及格！」。 條件二： 不及格成績為低於 60 分。如果此條件不成立，會顯示「恭喜你的成績及格！」；此條件不成立，即代表成績低於 60 分，則會顯示「你的成績不及格，須參加補救教學！」		
		3. 摘要化	兩種判斷結構的程式敘述皆需先寫判斷的條件式，接著寫條件的成立與否，各自顯示的結果。 （1）判斷結構 if…程式敘述： if（條件式）: 　　條件成立，執行的程式區塊		

			if（條件式）:
			　　條件成立，執行的程式區塊
			
			（2）判斷結構 if...else...程式敘述：
			
			if（條件式）:
			　　程式區塊 1，條件成立，執行的程式區塊
			else:
			　　程式區塊 2，條件不成立，執行的程式區塊		

		4. 步驟化	1. 程式執行流程： （1）使用者輸入一數值 （2）電腦讀取輸入的數值後，進入條件判斷的區塊 （3）判斷的條件有兩個，當判斷及格的條件成立，會顯示「恭喜你的成績及格！」；條件不成立，則會顯示「你的成績不及格，須參加補救教學！」		

	2. 依據程式執行流程來設計程式，並使用程式語法來完成解題。	

▲表（三）示範例題 2－教學內容

判斷結構程式設計－實作練習

1. 實作示範後，在實作練習的教學活動中，將給予學生三題練習題進行實作，完整的練習題題目於附錄三。實驗組、對照組練習題內容皆相同，相異處為鷹架引導方式的不同。

在實驗組的練習題中，給予學生具有運算思維流程的鷹架引導方式來回顧課程內容重點；對照組的練習題中，則給予傳統教學鷹架引導方式來回顧課程內容重點。

2. 實驗組－運算思維流程的鷹架引導方式：

作答前你可以想想下列問題：

題目要求程式需判斷什麼？

課程重點回顧：

1. 確定使用者所輸入的數值代表的意義為何，以及電腦讀取數值後，會顯示哪些不同的結果。
2. 依據不同的顯示結果，列出判斷此結果的條件各自為何。
3. 將判斷的條件代入判斷結構的程式敘述。在程式敘述中，需先寫判斷的條件式，接著寫條件的成立、不成立，顯示的結果各自為何。

```
if(條件式):
    條件成立，執行的程式區塊
if(條件式):
    條件成立，執行的程式區塊
```

```
if(條件式):
    程式區塊1，條件成立，執行的程式區塊
else:
    程式區塊2，條件不成立，執行的程式區塊
```

4. 依據程式執行流程來設計程式，並使用程式語法來完成解題。

程式執行流程：

使用者輸入一數值→電腦讀取輸入的數值後，進入條件判斷的區塊→判斷條件成立、不成立，顯示對應的結果

-----確定好上述內容，使用程式語法來將你的想法輸出，完成解題吧！-----

給予實驗組「運算思維流程」的課程重點回顧的編號排序，為依照運算思維流程四步驟的順序來排序。

3. 對照組－傳統教學的鷹架引導方式：

作答前你可以想想下列問題：

題目要求程式需判斷什麼？

課程重點回顧：

1. 判斷結構 if…與 if…else…的程式敘述分別為：

```
if(條件式):
    條件成立，執行的程式區塊
if(條件式):
    條件成立，執行的程式區塊
```

```
if(條件式):
    程式區塊 1，條件成立，執行的程式區塊
else:
    程式區塊 2，條件不成立，執行的程式區塊
```

2. 程式執行流程為先讀取使用者輸入的值，再進行條件的判斷，來決定要執行哪一個程式區塊，並顯示對應的結果。
3. 在判斷結構的程式敘述中，需先寫判斷的條件式，再寫條件成立時，顯示的結果為何；條件不成立時，顯示的結果為何。

-----確定好上述內容，使用程式語法來將你的想法輸出，完成解題吧！-----

對照組的「傳統教學鷹架引導方式」為依判斷結構的程式順序來回顧課程重點。

附錄二 教學教材

判斷結構－if…與 if…else…

一、當你需要設計一個具有判斷能力的程式時，需使用判斷結構式。電腦會根據條件成立的與否來決定是否執行程式區域。

二、條件式可為單一項或多項，電腦會依據條件式來進行單向、雙向及多向的判斷。

三、1. 單項條件範例：

（1）設定鬧鐘時，鬧鐘會在使用者設定的時間準時響起，其餘時間並不會有任何聲響。

（2）某家商店的會員制度為當年消費滿額即可成為 vip 會員，未達指定金額即是一般會員。

2. 多項條件範例：

在購物的過程中，決定要不要購買某項商品時，需要考慮的條件很多（外觀特徵、價錢、耐用度等），消費者會依照這些條件有沒有符合自身需求，來判斷是否購買此商品。

四、日常生活中的判斷結構應用：

（1） 天氣預報發布當週的下雨機率，你認為當天下雨機率只要為 50%以上（含）就需帶傘出門。

（2） 當手機電量低於 20%時，螢幕上的電池會呈現紅色，高於 20%則呈現綠色。

（3） 考試成績，未達 60 分為不及格，必須參加補考。

五、判斷結構 if…為單向的判斷，當條件式成立時，電腦會執行程式區塊；條件不成立時，電腦會執行判斷結構 if…後的程式區塊。

1. 程式語法：

if(條件式):

□條件成立，執行的程式區塊

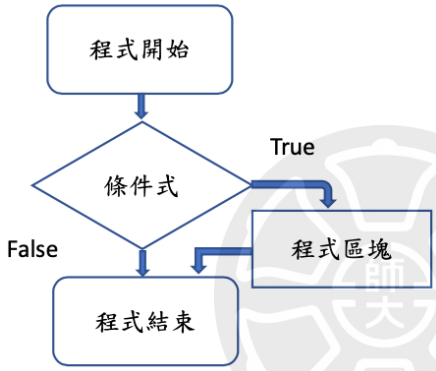
if(條件式):

□條件成立，執行的程式區塊

（1）條件式後需加冒號「：」。

（2）□ 為執行的程式區塊需要縮排四個空白字元的部分。

2. 流程圖：



六、判斷結構 if…else…為雙向的判斷，此判斷結構一定有兩個執行的程式區塊。當條件成立時，電腦會執行程式區塊內的程式；條件不成立時，電腦會執行另一個程式區塊內的程式。

1. 程式語法：

if(條件式):

□程式區塊 1，條件成立，執行的程式區塊

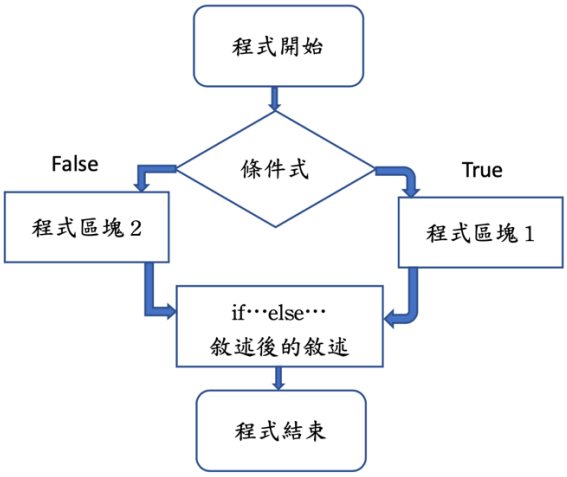
else:

□程式區塊 2，條件不成立，執行的程式區塊

（1）條件式後需加冒號「：」。

（2）▭為執行的程式區塊需要縮排四個空白字元的部分。

2. 流程圖：



在設計具有判斷結構的程式前，你必須要先知道以下各項內容：

一、語法規則－變數名稱

（1）第一個字母需是「英文字母（大、小寫皆可）」、「底線符號 __ 」、「中文」，不能以數字來作為起始字元。

（2）變數名稱是由「大、小寫英文字母」、「數字」、「底線符號 __ 」、「中文」。

二、資料型態

（1）整數：int

三、print 輸出函式

print( )指令用於列印指定的項目內容於輸出的裝置上。輸出時可一次輸出多個項目，每個項目之間以「,」逗號來分隔。

程式語法：

（1） 輸出單一項目

print( )

（2） 一次輸出多個項目

print(項目 1, 項目 2, 項目 3, …, …, …)

四、input 輸入函式

input( )指令是用於讓使用者透過輸入設備（鍵盤），來輸入資料。讓電腦讀取使用者所輸入的資料，必須將輸入值儲存至一個變數（變數名稱自訂）。

例：假設有一程式是讓使用者輸入年紀，我們把輸入的年紀設定為變數 a。

程式語法：

a=input('請輸入你的年紀：')

五、關係運算子

（1）>= 大於等於

（2）!= 不等於

（3）= 等於

六、算數運算子：

（1）＋：加

（2）－：減

（3）*：乘

（4）/：除

（5）%：餘數運算

七、邏輯運算子

（1）and：和

例：(a>b)and(b>c)，表示(a>b)、(b>c)兩個條件皆成立。

（2）or：或

例：(a>b)or(b>c)，表示(a>b)、(b>c)其中一個條件需成立。

附錄三 課程練習題－對照組

練習題

姓名： **座號：**

作答前你可以想想下列問題：

題目要求程式需判斷什麼？

課程重點回顧：

1. 判斷結構 if…與 if…else…的程式敘述分別為：

```
if(條件式):
    條件成立，執行的程式區塊
if(條件式):
    條件成立，執行的程式區塊
```

```
if(條件式):
    程式區塊 1，條件成立，執行的程式區塊
else:
    程式區塊 2，條件不成立，執行的程式區塊
```

2. 程式執行流程為先讀取使用者輸入的值，再進行條件的判斷，來決定要執行哪一個程式區塊，並顯示對應的結果。
3. 在判斷結構的程式敘述中，需先寫判斷的條件式，再寫條件成立時，顯示的結果為何；條件不成立時，顯示的結果為何。

-----確定好上述內容，使用程式語法來將你的想法輸出，完成解題吧！-----

一、

三角形的三邊長中，任意兩邊長的總和會大於第三邊長。

請設計一程式能讓使用者分別輸入三邊各自的長度，電腦會依此來判斷是否能構成一三角形。

下圖為使用者分別輸入三邊長為 12、7、9 後，顯示「可以構成三角形」的程式執行結果。

```
請輸入三角形邊長1：12

請輸入三角形邊長2：7

請輸入三角形邊長3：9
可以構成三角形
```

二、

小靜規定自己一天三餐吃的花費須控制在 150 元內（包含 150 元）。

請設計一程式能讓小靜分別輸入早餐、午餐、晚餐的金額，電腦會判斷今日的花費是否超過預算。

下圖為使用者分別輸入早餐、午餐、晚餐的金額後，顯示「今日花費超過預算囉！」的程式執行結果。

```
請輸入早餐的金額：50

請輸入午餐的金額：55

請輸入晚餐的金額：50
今日花費超過預算囉！
```

三、

貨運公司規定國內一般包裹紙箱的體積（長 x 寬 x 高）不能大於 450 cm^3。

請設計一程式能讓使用者分別輸入紙箱的長、寬、高長度，電腦會根據貨運公司的尺寸規定，來判斷紙箱是否符合尺寸，可以寄送。

下圖為使用者分別輸入長、寬、高的長度後，顯示「尺寸符合規定，可以寄送！」的程式執行結果。

```
請輸入長的長度：10

請輸入寬的長度：5

請輸入高的長度：4
尺寸符合規定，可以寄送！
```

＊自由設計程式題：

想想日常生活中有哪些是判斷結構的應用，設計一個判斷結構的程式吧！（例如：判斷數字的大小、判斷時間是否準時、判斷地震震度度等級……等）

附錄三 課程練習題－實驗組

練習題

姓名： **座號：**

作答前你可以想想下列問題：

題目要求程式需判斷什麼？

課程重點回顧：

1. 確定使用者所輸入的數值代表的意義為何，以及電腦讀取數值後，會顯示哪些不同的結果。
2. 依據不同的顯示結果，列出判斷此結果的條件各自為何。
3. 將判斷的條件代入判斷結構的程式敘述。在程式敘述中，需先寫判斷的條件式，接著寫條件的成立、不成立，顯示的結果各自為何。

```
if(條件式):
    條件成立，執行的程式區塊
if(條件式):
    條件成立，執行的程式區塊
```

```
if(條件式):
    程式區塊 1，條件成立，執行的程式區塊
else:
    程式區塊 2，條件不成立，執行的程式區塊
```

4. 依據程式執行流程來設計程式，並使用程式語法來完成解題。

   程式執行流程：

   使用者輸入一數值→電腦讀取輸入的數值後，進入條件判斷的區塊→判斷條件成立、不成立，顯示對應的結果

-----確定好上述內容，使用程式語法來將你的想法輸出，完成解題吧！-----

一、

三角形的三邊長中，任意兩邊長的總和會大於第三邊長，請設計一程式能讓使用者分別輸入三邊各自的長度，電腦會依此來判斷是否能構成一三角形。

下圖為使用者分別輸入三邊長為 12、7、9 後，顯示「可以構成三角形」的程式執行結果。

```
請輸入三角形邊長1：12

請輸入三角形邊長2：7

請輸入三角形邊長3：9
可以構成三角形
```

二、

小靜規定自己一天三餐吃的花費須控制在 150 元內（包含 150 元）。請設計一程式能讓小靜分別輸入早餐、午餐、晚餐的金額，電腦會判斷今日的花費是否超過預算。

下圖為使用者分別輸入早餐、午餐、晚餐的金額後，顯示「今日花費超過預算囉！」的程式執行結果。

```
請輸入早餐的金額：50

請輸入午餐的金額：55

請輸入晚餐的金額：50
今日花費超過預算囉！
```

三、

貨運公司規定國內一般包裹的紙箱體積（長*寬*高）不能超過 500 立方公分，且長、寬、高邊長的總和不能大於 150 公分。請設計一程式能讓使用者分別輸入長、寬、高的長度，電腦會根據貨運公司的尺寸規定，來判斷是否符合尺寸，可以寄送。

下圖為下圖為使用者分別輸入長、寬、高的長度後，顯示「尺寸符合規定，可以寄送！」的程式執行結果。

```
請輸入長的長度：10

請輸入寬的長度：5

請輸入高的長度：4
尺寸符合規定，可以寄送！
```

＊自由設計程式題：

想想日常生活中有哪些是判斷結構的應用，設計一個判斷結構的程式吧！（例如：判斷數字的大小、判斷時間是否準時、判斷地震震度度等級……等）

附錄四 學習成效測驗試卷

測驗題

姓名： **座號：**

作答說明：

請開啟 test.py 檔，按照題號依序作答。 作答完成後，請另存新檔(File >> Save)，並將檔名命名為「日期__座號」，如：「0401_01.py」，再上傳至雲端資料夾即可！

1. 三角形的內角和為 180 度。請設計一程式能讓使用者分別輸入三個角度，輸入後電腦會顯示是否能構成一個三角形，如果能構成三角形會顯示「可以構成三角形」；不能構成則會顯示「無法構成三角形」。

   下圖為分別輸入「45、30、60」三個角度後，程式的執行結果為「不能構成三角形」，請依此設計程式。

```
請輸入三角形角度1：45

請輸入三角形角度2：30

請輸入三角形角度3：60
不能構成三角形
```

2. 補習班提出一獎勵方案：數學、英文兩科段考成績的平均為 80 分以上（含），即可獲得 100 元禮卷。請設計一程式能讓使用者分別輸入數學、英文的段考成績，電腦會根據輸入的成績來計算平均，並顯示「可獲得 100 元

禮卷」或「不可獲得 100 元禮卷」。

下圖為分別輸入數學、英文兩科的成績「67、88」後，程式的執行結果為「不可獲得 100 元禮卷」，請依此來設計程式。

```
請輸入數學成績：60

請輸入英文成績：90
不可獲得100元禮卷
```

3. 家中餐桌的長度為 25 公分，寬度為 20 公分。媽媽想為餐桌添購桌巾，請設計一程式能讓媽媽輸入長度、寬度後，電腦能判斷面積是否符合尺寸，並顯示「尺寸符合」或「尺寸不符合」。

   下圖為分別輸入長度「20」、寬度「10」後，程式的執行結果為「尺寸不符合」，請依此來設計程式。

```
請輸入長度：20

請輸入寬度：10
尺寸不符合
```

4. 某一遊樂設施只有三個座位，且限重 175 公斤。請設計一程式能讓使用者輸入三個公斤數後，會顯示「已超重」或「未超重」。

   下圖為分別輸入「55、67、40」三個公斤數後，程式的執行結果為「未超重」，請依此來設計程式。

```
請輸入體重1：55

請輸入體重2：67

請輸入體重3：40
未超重
```

5. 新商場開幕舉辦優惠回動：開幕前三天消費金額平均達 200 元以上（含），憑三張發票即可兌換一張 100 元禮卷。請設計一程式能讓使用者輸入三天各自消費的金額後，顯示「恭喜獲得 100 元禮卷」或「未符合兌換 100 元禮卷資格」。

下圖為分別輸入「270、98、100」三天各自的消費金額後，程式的執行結果為「未符合兌換 100 元禮卷資格」，請依此來設計程式。

```
請輸入第一天消費金額：270

請輸入第二天消費金額：98

請輸入第三天消費金額：100
未符合兌換100元禮卷資格
```

附錄五 自我效能量表

親愛的同學您好：

這份問卷是為了瞭解您在學習完「判斷結構：if…與 if…else…程式設計」課程後的感受，以作為學術研究使用。**此份問卷並無標準答案，請您依照自己的真實感受來作答即可。**

本問卷共有 25 題，每一題都有 7 個選項，由左至右依序為「非常不同意」、「不同意」、「有點不同意」、「普通」、「有點同意」、「同意」、「非常同意」，請根據題目敘述，在符合您感受的選項空格內打勾。

本問卷所得的資料僅作為研究之用，採絕對保密不會公開，您的填答對本研究有莫大的幫助，在此致上最誠摯的感謝！

姓名：　　座號：

題目	非常不同意	不同意	有點不同意	普通	有點同意	同意	非常同意
1. 我能理解判斷結構程式中，條件式的程式敘述。							

2. 我能理解判斷結構的程式執行流程。							
3. 我能理解程式執行的流程。							
4. 我能了解程式輸入與輸出的執行流程。							
5. 我能了解程式輸入與輸出的執行流程。							
6. 我能確定問題中的判斷模式，並撰寫具有判斷結構的程式。							
7. 我能設計一個結構正確的程式。							

8. 我能在適當的問題設計一具有判斷結構的程式。							
9. 我能解釋不同判斷結構使用方法上的差異。							
10. 我能解釋 if…和 if…else…兩者之間的差異。							
11. 給你一判斷結構的程式設計題目，我能分別使用不同的判斷結構來設計程式。							
12. 即使我已經花了幾個小時修復錯誤，並且剛剛發現了一個新的錯誤，我還是能撰寫一個正確執行的程式。							

13. 即使我之前嘗試撰寫類似的程式時沒有成功，我還是能撰寫出一個能正確執行的程式。							
14. 給你一個解決問題的方法和一個不正確的程式，我能找出不正確的程式錯誤之處。							
15. 我能透過程式執行的流程來分析問題，再設計解決方法，並使用程式碼來解決問題。							
16. 我能設計一個執行流程正確的程式。							
17. 我能將一個問題拆解出一組簡單的小問題，再依序來思考解決方法。							

18. 我能描述程式執行的過程中，遇到判斷結構時，所使用的程式語法為何。							
19. 我能建構一個具有判斷結構的程式，並簡單描述其中的條件語句。							
20. 我能撰寫一個具有判斷結構的程式。							
21. 給你一個具有判斷結構的程式，我能簡要的解釋它的功能。							
22. 給你一個相似的程式問題的解決方法，我能參考後，撰寫一個程式來解決新的問題。							
23. 我能解釋不同判斷結構之間的差異。							

24. 透過實作練習程式編寫的經驗，能讓我在遇到不同問題時，撰寫出一個新的程式。							
25. 給你一個判斷結構的流程圖，我能描述電腦在執行具有判斷結構程式的過程中，執行的流程為何。							

附錄六 學習成效測驗評分表

1. 程式敘述－輸入（3 分）	
評分項目	**得分**
1.1　縮排格式	
1.2　符號	
1.3　變數命名	
2. 程式敘述－輸出（3 分）	
評分項目	**得分**
2.1　縮排格式	
2.2　符號	
2.3　拼字	
3. 判斷結構（4 分）	
評分項目	**得分**
3.1　if…或 if…else 程式結構	
3.2　符號	
3.3　縮排格式	
3.4　條件式	

附錄七 訪談大綱

訪談大綱－專家

訪談範圍
1. 對於鷹架引導方式在程式寫作教學上的想法與建議。 2. 對於兩者鷹架引導內容差異的感受與想法。
第一部分 鷹架引導方式的教學應用
1. 你覺得使用鷹架引導方式來教學程式寫作，對於學習是否有幫助的？並說明原因。 2. 使用運算思維流程的鷹架引導方式，對於學習程式寫作是否有幫助？並說明原因。 3. 你覺得使用鷹架引導方式來教學程式寫作，是否能提升學習者在學習程式寫作上的信心？並說明原因。 4. 你覺得學習者在參與完本教學課程後，對於未來在學習程式寫作上是否能更有信心的來嘗試學習？並說明原因。
第二部分 鷹架引導內容
1. 覺得兩者鷹架引導內容的差異為何？ 2. 覺得運算思維流程的鷹架引導內容有沒有需要改善的地方？並說明原因。 3. 覺得傳統教學的鷹架引導內容有沒有需要改善的地方？並說明

原因。

訪談大綱－學習者

訪談範圍
1. 參與本研究教學課程後的感受與想法。 2. 程式實作練習活動遇到的困難。 3. 課程練習題引導內容的想法與建議。
第一部分 教學課程參與
1. 參與完本課程後你認為對於你的程式寫作有沒有幫助？ 2. 課程內容有哪個地方是你最（有興趣／無興趣）的？並說明各自的原因。 3. 參與完本課程後你覺得有讓你對程式寫作的信心提升嗎？並說明原因。 4. 如果別的課程也使用本課程的教學流程，你覺得對於學習會有幫助嗎？
第二部分 程式寫作
1. 在實作練習判斷結構程式寫作時，是否有遇到困難？如果有，是什麼困難？ 2. 如果遇到 Spyder IDE 故障，或是程式碼出現錯誤時，你會如

何解決？ 3. 課程中安排老師先進行判斷結構程式寫作的示範，接著讓你們練習，你覺得這樣的安排對於學習程式寫作有幫助嗎？並說明原因。
第三部分 鷹架引導內容
1. 你認為練習題中所給予的引導內容有沒有幫助？如果有，說明你覺得有幫助的地方為何，以及原因。 2. 如果不同的程式寫作單元也會給予引導內容來幫助學習，你會願意學習嗎？ 3. 覺得引導內容有沒有需要改進的地方？並說明原因。