第二章 文獻探討

本研究是以問題解決的觀點，探討教學策略與學習工具對程式語言學習之影響。以下分別就「問題解決」、「程式語言教學」及「程式語言學習工具」進行探討。

第一節 問題解決

人類每天會遭遇許多問題，且透過解決問題適應環境，因此問題解決能力是現今社會生存的重要條件之一。因為問題解決能力的重要性，其技巧的教與學是長久以來大家研究及爭論的主題，許多學者均宣稱問題解決技巧是學校教育最重要的目標(Dennis & Kansky, 1984; Holloway, 1975)，而程式語言的教學目標也在於讓學生運用程式解決問題。因此為了促進學生的問題解決能力，在設計程式語言教學活動時，應深入瞭解問題解決的相關理論，運用適當的問題解決策略在程式語言教學中讓學生進行學習。本節將就「問題解決的意義」、「問題解決歷程」、「問題解決的認知需求」及「問題解決的教學策略」作深入探討。

壹、問題解決的意義

「問題」是學習者面臨目標但不知如何馬上達成這個目標。「問題解決」則是能整合長期記憶體中相關的宣告性知識、規則及認知策略，並應用來解決之前未遇過的問題，是心智運作的過程。Polya(1945)認為問題解決是一種思考的類型，是學習者遇到困難及障礙時，能找到克服的方法並達到目標的過程。因此所有的認知活動在本質上可以說是問題解決，因為人類的認知是有目的性的，為達成目標，人類會去除過程中所遭遇之障礙(Anderson, 1983)。

在問題解決的過程中，「問題」不屬於是、否或簡單一句話即可回答的問題，而是屬於必須自行找到方法以達到目的之「難題」(張春興，民 90)。像：請問先生您貴姓？您是從鄉下來的嗎？您住在什麼地方？這類問題不包括在研究範圍內。但例如：人在五樓而四樓起火怎麼逃生？汽車鑰匙丟了，下班後怎樣回家？這類問題則是屬於問題解決的研究範圍。Richard(1992)曾提出構成問題的三項共通要件為：(1)已知的事實：包括問題的情境、狀態及資訊，即問題的初始狀態，(2)目標：問題最終的狀態，(3)障礙：從已知的事實到到最終目標的過程中所會遇到的困難。Newell 及 Simon (1972)在定義「問題」時也認為，「問題」是由「問題空間」(problem space) 所構成，包括三種狀態：(1)目標狀態、(2)初始狀態及(3)中間狀態。而問題解決便是透過中間狀態將問題從初始狀態轉換到目標狀態的過程。

從皮亞傑的觀點來看，問題解決是運用已有的基模及策略來解決所面臨的問題。Smith 及 Ragan(1999)則認為問題解決是能整合學習過的規則、宣告性知識和認知策略來解決目前所遭遇問題的能力，並在問題解決的過程中能產生新的學習。問題解決不同於一般規則的學習，為解決問題，學習者要選擇及整合許多新舊知識及規則來得到問題的答案，是屬於高層次的認知活動，而不是單一的規則或知識的學習，因此問題解決的過程包含了許多心智歷程。許多學者在探討是否有系統化的解題步驟讓學習者順利解決問題，這些解題步驟即「問題解決歷程」。下一節將整理眾家學者所提出的問題解決歷程，作為設計程式語言的問題解決步驟之參考。

貳、問題解決歷程

Dewey(1910)以科學的觀點將問題解決歷程分為以下五個步驟：(1)發現問題：問題解決者意識到問題存在，(2)定義問題：問題解決者確認問題的關鍵所在及可能的限制，(3)提出假設：提出可能的解決方案，(4)驗證假設：逐一檢驗解決方案，確認其可行性，並選出最佳的方案，(5)實際行動：以實際行動驗證假設，並隨時修改，成立結論。Wallas(1926)則提出不同的方法，他以創意的觀點將問題解決分為四個步驟：(1)準備期(preparation)：問題解決者搜集問題背景相關資料，(2)醞釀期(incubation)：若無法立即解決，轉移注意力，(3)豁朗期(illumination)：問題解決者突然得到靈感、獲得問題的解答，(4)驗證期(verification)：驗證解答。後續所提出的問題解決歷程大多整合上述兩個歷程而來，其中以 Polya(1945)所提出的問題解決步驟最常被採用及參考。他所提出的一般性問題解決歷程可應用來解決許多不同類型的問題，包括四個步驟：「瞭解問題、設計計劃、實行計劃、回頭看」。SolSo(1991)提出一個較詳細的問題解決歷程，包含六個步驟：「確認問題、問題表徵、計劃解決的行動、執行計劃、評估計劃、評估解決的行動」。Rubinstein(1975)則提出「保留」的觀念，認為在問題解決的過程中不急著下判斷，要心存懷疑，並隨時修正策略，其問題解決歷程為：「綜觀全圖、先不下判斷、運用模型、修正策略、問正確問題、質疑結果」。綜合上述之歷程發現，問題解決不外乎以下四個步驟：

(1)瞭解問題：問題描述是問題解決的第一步驟，也是所有問題解決的起點，因此要釐清問題所包含的相關資訊。

(2)擬定解題計畫：利用與待解決問題相關的已有知識及環境中可獲得的資源來計劃相關的解題方法。

(3)將計劃付諸實現：將思考所得的方法，付諸實行。

(4)驗證解題結果：驗證解答的正確性及檢視解題所得到的啟發與心得。

上述所提之問題解決步驟為一般性問題解決技巧，可廣泛應用來解決各種問題，但一般性問題解決技巧不一定能有效運用於特定領域的問題解決歷程(Anderson, 1985)，因此許多學者針對不同領域的知識特性提出特定問題解決技巧，以期更有效地解決問題。Etter(1995)修正 Polay 的問題解決步驟，提出應用於工程及科學的問題解決歷程，其步驟為：「定義問題、搜集資訊、產生及評估可能的解答、修正及實施解答、證實及測試解答」。Meier, Hovde 及 Meier(1996)亦針對數學及科學問題的解決歷程，將問題解決分為：「定義問題、評估情境、計劃解題策略、實行計劃、確認結果」等五個步驟。Rayner-Canham(1990)曾將問題解決的歷程運用於電腦輔助教學教材的設計上，提出：「確認問題、計劃解決策略、搜集相關資訊、執行計劃」四個步驟。Deek, Turoff 及 McHugh(1999)則整合學者所提出的問題解決步驟，認為促進程式語言學習的問題解決步驟分別為：「闡述問題、計劃解答、設計解答、將計劃轉換成程式碼、測試及撰寫文件」。

綜合上述問題解決歷程，程式語言的問題解決可整理為以下四個步驟：

(1)瞭解問題需求：程式設計過程中，瞭解問題需求是第一步驟，要知道程式的輸入及輸出要求。

(2)擬定解題計劃：將問題分解成小問題，並以模組化的方式規劃出解決問題的步驟。

(3)撰寫程式碼：撰寫程式碼以實行所設計的解題計劃。

(4)測試與除錯：進行測試，確認解題的正確性及找出程式錯誤之處。

參、問題解決的認知需求

問題解決是將問題從目標狀態轉換到目的狀態的過程，在過程中要運用已有的基模及策略來解決所面臨的問題，因此問題解決是一複雜的心智歷程，需要各種不同的認知需求。專家與初學者在本身所具備的知識及問題解決技巧上有所不同，由專家的特性可以瞭解問題解決的認知需求。專家的問題解決有兩個特性：(1)專家對於與待解決問題相關的事實、宣告性知識有完整的掌握；(2)專家有許多有效的策略可以用來解決問題(Palumbo & Reed, 1991； Hayes, 1989)。專家通常除了具備與知識領域相關的事實及常規外，還會將知識有效地整合為主要的概念並引導自己思考及應用。如在物理學科方面，當專家及初學者被要求口頭描述用來解決物理問題的方法時，初學者較少提及物理上的原理，而通常描述自己打算使用的公式及如何運用這些公式，相反地，專家通常提及適用於問題的主要原理、定律、為什麼要應用這些定律及如何應用等問題，例如他們會應用基本的牛頓第二運動定律來解決問題(Larkin, 1981/83)。因此完善的宣告性知識及認知策略是問題解決不可或缺的認知需求。

Cannon-Bowers 及 Bell(1997)曾定義問題解決的認知過程及技巧包括：(1)靈活性(flexibility)、(2)敏捷性(quickness)、(3)彈性(resilience)、(4)適應性(adaptability)、(5)勇於冒險(risk taking)及(6)正確性(accuracy)。Oser, Gualtieri 及 Cannon-Bowers(1999)則認為上述的問題解決認知過程及技巧太過廣泛，因此加以修改並提出訓練問題解決的理論構架，其中四項更明確的能力為：「資訊處理技巧、情境評估技巧、推理技巧及監控」。Smith 和 Ragan(1999)則提出問題解決的認知需求為：(1)回憶及應用相關規則的能力，(2)回憶宣告性知識的能力，(3)回憶及應用認知策略的能力。綜合以上專家問題解決的特性及問題解決認知過程的

需求與技巧，可以歸納在問題解決的過程中所需具備的認知需求為：(1)資訊處理技巧、(2)情境評估技巧、(3)認知策略及(4)後設認知，以下分別就此四項需求進一步說明：

一、資訊處理技巧

問題解決的過程中，學習者首先要能意識到問題的存在，而資訊處理技巧指的便是學習者對於環境刺激的處理技巧。根據資訊處理理論，人類處理資訊的過程與電腦的過程類似：「接受資訊、儲存在記憶體中、當需要時擷取相關資訊」。但人類資訊處理的過程中還包含了許多認知活動，例如：理解、複述、思考、問題解決、記憶、遺忘、想像等。因此以下由資訊處理理論探討資訊處理技巧對問題解決的影響。

資訊處理理論起源於對行為理論的反省，行為主義心理學家在研究學習時，只重視外在的刺激與個體反應間的連結，卻忽略學習者內在的心理運作或思維活動。而資訊處理理論則著重刺激及反應在學習者心智上的過程，學習者被認為是主動找尋及處理資料，而非只是被動地接受刺激。學習者選擇及注意環境中的特徵，轉換資訊，將新的資訊與舊知識作連結，並整合成有意義的訊息進行其它處理或將訊息儲存於記憶體中，其模式如圖 2-1 所示：

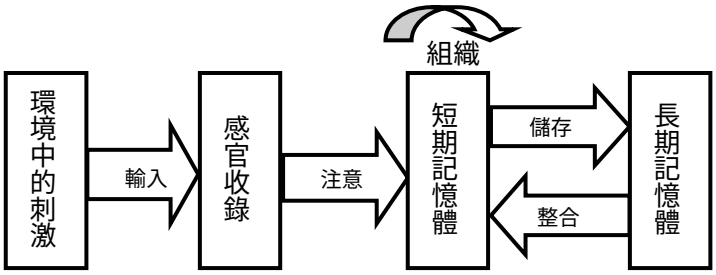

圖 2-1 訊息處理之模式(整理自:張春興，民 85。教育心理學-三化取向的理論與實踐。台北：臺灣東華書局，P.225)

環境中的刺激經過感官收錄後，如果決定進一步處理就加以注意，並予以

編碼轉換為另一種形式的訊息進入短期記憶體，否則即放棄此刺激(張春興，民85)。研究指出，短期記憶體有兩個特性，一為有限的記憶容量，一為有限的停留時間。一般來說，短期記憶體一次所能保存的容量為 7 個左右的項目(items，例如有意義的文字、數字)，但可以透過將資訊整合成有意義的訊息之方式來增加容量。而資訊在短期記憶體所能保留的時間很短，如果沒有經過處理，很快便會消失，例如當我們聽到一個電話號碼時，短時間之內可以記得，但是如果沒有特別注意，幾分鐘之後便可能無法再想起這個號碼，但透過複述，可以使資訊在短期記憶體的時間增加，例如將電話號碼默唸幾次，可以延長此串數字保留的時間。而專家與初學者很重要的差別在於專家能將大量資訊組合成相似的型態，並將資訊做正確的編碼及分類，以進入長期記憶體(Glaser, 1986)。Druckman 及 Bjork(1991)也曾經提到，專家不只比初學者擁有較多的知識，更能將這些知識有效地整合。因此專家在遇到問題時，能夠快速且正確地將問題重新整理、分類，並找到核心所在。而初學者缺乏對環境的敏感度，在接受環境的刺激後，往往沒有立即進一步予以處理，形成感官收錄過的遺忘，加上初學者對工作領域的知識整合不完整或不適當(Glaser, 1986)，即使對環境中的刺激有反應，也無法快速且正確地重新整理及瞭解所遇到的問題並加以解決。因此資訊處理技巧是問題解決的重要認知需求，學習者擁有適當的資訊處理技巧才能保留環境中的刺激到工作記憶體中進行處理，並回憶長期記憶中的資訊整合成新的知識來解決問題。

二、情境評估技巧

情境評估技巧指的是能瞭解問題需求的技巧，學習者要能從已知的事實中辨別問題的情境。學習者所具備的宣告性知識會影響解決問題的成效，因此以下以人類的記憶模式來探討環境評估對問題解決的影響。

Anderson 及 Bower(1973)曾提出一個「人類連結記憶」(Human Associate Memory，簡稱 HAM)模式，認為人類接收到的訊息經過處理後，進入到長期記憶體中以命題網絡(proposition network)的模式存放。命題(proposition)是資訊中可以被評斷對或錯的最小單位，例如：「小明看書」，「看」連接「小明」和「書」兩個概念，是一個命題。命題通常可以分析成包括四種型式的連結：(1)情境-事實、(2)地點-時間、(3)主詞-述詞及(4)關係詞-受詞。情境-事實是連結發生什麼事和在什麼情境中發生的訊息，地點-時間的連結說明時間和地點，而事實又由主詞和述詞的連結所說明，主詞說明誰，述詞則說明主詞發生了什麼事，關係詞-受詞的連結則說明述詞的性質。如圖 2-2，箭頭代表連結，圓圈為節點，是連結的接合點，樹狀圖的最底部節點是命題所在，即真正的記憶位置(鄭麗玉，民 82)。

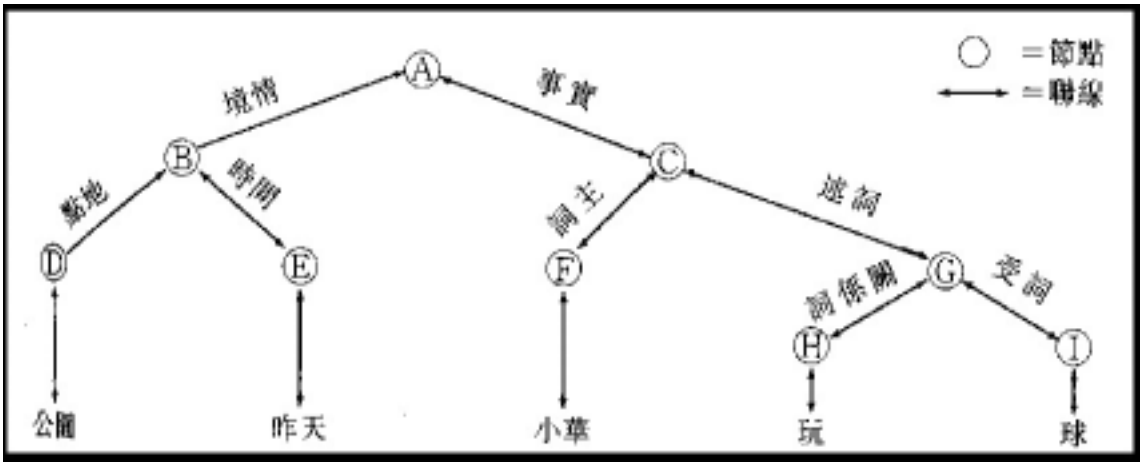

圖 2-2 人類連結記憶模式中，「小華昨天在公園玩球」句子的記憶表徵。

(摘自鄭麗玉，民 82。認知心理學-理論與應用。台北：五南圖書，P.71)

Anderson(1973)根據人類連結記憶模式發展一個更詳盡的「思想調整控制」(Adaptive Control of Thought，簡稱 ACT)模式，擴充命題網絡的概念，並區分「宣告性性記憶」(declarative memory)和「產出性記憶」(production memory)的差別。

宣告性記憶即是對宣告性知識的記憶，產出性記憶即對程序性知識的記憶。宣告性知識以命題為基礎，並以命題網絡的方式儲存在記憶體中，能幫助學習者瞭解問題及問題空間的限制(如：起始狀態、目標狀態)。Glaser(1986)曾提出專家所擁有的能力便是能從問題空間中辨識出有意義的型態，當遇到問題時，能從情境中找到相關的概念、規則，並且正確地應用來解決問題(Smith & Ragan, 1999)，這是因為專家能有效地回憶宣告性知識，亦即專家擁有較好的情境評估技巧。情境評估的兩個不可或缺技巧是線索辨識(cue recognition)及型態辨識(pattern recognition)，專家有比初學者更好的線索辨識技巧，可以很快也從環境中察覺重要的特徵(Cannon-Bowers & Bell, 1997)，且試著去瞭解問題更深層的結構，並從長期記憶中找到適合的宣告性知識來解決問題。而初學者則著重在問題表層結構(Chi, Feltonvitch & Glaser, 1981)，因此往往無法找到問題所在。

宣告性知識以命題為基礎，是對事實的描述。但單一的規則或線索不足以解決問題，要能整合及應用許多規則間的關係所組成的型態才能達到目標(Klein, 1995)。因此要整合宣告性知識成為程序性知識，才能應用來解決問題。專家有較多問題相關領域的知識及經驗，且其經驗及既有的規則會轉換成產出的形式存放在記憶體中，以 if-then 的結構存在(Smith & Ragan, 1999)，因此可以較容易整合從環境中所辨識出的規則及線索，並由記憶中找到相對應的樣本，規劃可能的解答。

初學者的情境評估技巧不如專家，遇到問題時不容易辨識出可應用的規則。且初學者的經驗不足，缺乏相關的樣本用以解決問題，所以讓初學者作問題解決學習時，不能假設他們能自動發現環境中的規則，而是要提供相關的先備知識，幫助初學者辨識情境以瞭解問題需求。

三、認知策略

日常生活中我們經常遇到待解決的問題，且這些問題複雜多變，所以學習者需要發展相關的技巧或策略來應付，如領域相關問題解決技巧或計劃，Cannon-Bowers 及 Bell(1997)定義這些技巧為「推理技巧」，也就是「認知策略」，因此認知策略即問題解決技巧，是問題解決的認知需求之一。

從問題初始狀態到目標的過程中是透過所謂的問題解決路徑(problem solving path)來達成，而建立問題解決路徑的策略可以分為一般性問題解決、特定問題解決及創造性問題解決。根據許多學者所提出的問題解決歷程，一般性問題解決的步驟可以歸納成「瞭解問題、擬定解題計劃、將計劃付諸實現、驗證解題結果」四個步驟。一般性問題解決技巧可以普遍應用在各個領域的問題解決，但針對不同領域知識的特性，學者提出不同的問題解決策略，如應用於地球科學的問題解決(張俊彥，民 88)，Rayner-Canham(1990)提出運用於電腦輔助教學教材設計上的問題解決模型，許多電腦學者也提出程式設計的解題過程可分為四個步驟：(1)瞭解問題需求，(2)擬定解題計劃、設計演算法，(3)撰寫程式碼，(4)測試與除錯。特定問題解決因為考量特定領域知識特性，因此在成效上比一般性問題解決更好。創造性問題解決較其他問題解決歷程更強調在解決過程中的兩種基本思考技能-發散思考(divergent thinking)及收斂思考(convergent thinking)的交替應用，也就是強調問題解決者在選擇或執行解決方法前需儘可能的想出多樣化解決方法，再從中進行選擇或實施。

Smith 和 Ragan(1999)認為，認知策略可以促進學習者(1)瞭解及陳述問題，(2)將問題分成子目標，(3)尋找、選擇和整合相關知識，(4)將知識整理成一系列可應用的解答，(5)監控這些解答是否可以成功達成目標。因此認知策略是問題解決的

重要的認知需求之一，有完備的認知策略可以讓學習者順利解決問題。

四、後設認知

除了認知策略外，問題解決還需要後設認知的技巧。後設認知是指個人對自己認知歷程的認知，也就是說當個人經由認知思維從事求知活動時，自己能明確瞭解所學知識的性質與內容，且能瞭解如何進一步支配知識以解決問題。從學習心理學的觀點來看，後設認知包括兩個層面，後設認知知識及後設認知技能 (張春興，民 85)。後設認知知識因其對象的不同可分為三類：(1)對「人物」的認知：是指對自己與別人認知能力的瞭解，如瞭解到自己的語文能力比數學能力好、我覺得某某人比我能幹等，(2)對「作業」的認知：瞭解作業的困難度或熟悉度，如兒童會逐漸地瞭解到記憶一篇文章的「大意」比「逐字逐句」的記憶要容易得多，(3)對「略策」的認知：瞭解應使用何種方法才能達到某種解題的目的，如兒童將會瞭解到讀書最重要的方法是注意課文主要的論點，並且以自己的語言說出這些主要的論點(鄭昭明，民 82)。由此可知，後設認知知識是指個人對自己所學知識的明確瞭解，個人不但瞭解自己所學知識的性質與內涵，同時也知道知識中所蘊含的意義及原理原則(張春興，民 85)。

後設認知技能是指在求知活動中個人對自己行動適當監控的心理歷程(張春興，民 85)。Gitomer 和 Glaser (1987) 認為，根據資訊處理理論，「自我調節」和後設認知的察覺 (metacognitive awareness) 在意義上大略是相同的，因為「自我調節」是一種讓學習者能自我激發、活化認知活動及學習行為，並因此能有系統的達到既定目標的過程。意即，「自我調節」能使學習者在面臨新的問題時，能感覺自己更有達成目標的能力 (Schunk, 1996)。Cohen, Freeman, Wolf 及 Militello(1995)也提出後設認知可以幫助學習者更有彈性地作出決策。因此學習者

必須能熟練地運用認知技能，才能在順利解決問題。

綜合以上所述，問題解決是一個複雜的認知過程，其所包含的認知需求遠超過一般知識、規則的學習，對初學者而言，問題解決的學習是極大的認知負載，如果沒有好的教學策略引導學習，學習者在面對問題時容易迷失，不知從何下手來解決問題，造成學習成效不佳。因此好的教學策略會促進初學者對問題解決的學習，以下提出常應用於問題解決的教學策略。

肆、問題解決的教學策略

問題解決在教學中扮演極重要的角色，但也因為其過程為高層次的認知活動，學習者不容易在其中自己習得問題解決的技巧。而教學者往往在教學的過程中會以無效的策略來教問題解決，例如只教授問題的相關規則，卻忽略了學習者無法自行選擇及結合這些規則來解決問題的事實，造成學習成效不佳。Pea(1983)認為問題解決或思考技巧應該直接被教導，而非預設學習者能在問題解決過程中自己習得這些技巧。因此透過教學設計來教導問題解決，可以讓學生更容易習得問題解決的技巧。Swan 及 Black(1988)的研究也發現，整合一般性問題解決策略於 Logo 程式語言教學，並直接教導問題解決技巧的教學方式，較能促進學生問題解決能力的學習。

許多策略可應用來教導問題解決並促進學生問題解決學習成效，例如模擬、個案研究、演練範例等，這些教學策略大多是遵從「引導發現」的方法，教學過程中會引導學習者去發覺及學習問題相關的規則、知識，並應用來解決問題，但在練習的過程中應逐步減少引導的程度，讓學習者有更多的機會自己面對問題及處理相關的資訊。以下分別說明常應用於問題解決的教學策略之進行方式及優缺

點：

一、模擬

模擬(simulations)通常是描寫一個問題情境讓學習者與問題互動，學習者面對問題時可以想像在實際環境中作決策，但不用承受作決策所造成的結果。模擬通常是電腦化的，因為電腦化的教材可以讓學習者在作決策的過程中得到立即的反應。

透過模擬的方式來教導問題解決可以將問題設計得更接近實際情境，問題的解答也可以更複雜，讓學習者在學習的過程中可以應用更多的資訊及策略來解決問題。

二、個案研究與個案問題

個案研究(case studies)是呈現一個實際情境讓學習者來扮演要解決問題的人，與模擬其實很類似，需要學習者選擇及運用各種的規則及原理解決問題。應用個案研究來教問題解決時，在個案的設計方面可以讓學生檢視所給的資訊、辨識出問題的目的狀態及選擇適當的原理來解決問題。

個案的設計應以實際生活中的問題為主，但現實生活的問題往往太過複雜而不易融入問題解決教學之中。因此較有效率的方式是將現實生活的問題簡化，忽略一些細節，讓學習者在不失真實的問題解決情境中學習。

三、演練範例

由問題解決的認知需求可以瞭解，具備足夠的宣告性知識才能應用相關的規則來解決問題，因此獲得基模(schema acquire)及規則自動化(rule automation)被認為是熟練問題解決的基本條件。基模可以讓問題解決的專家辨認出問題的狀態並選擇適當的基模來解決問題，自動化的規則可以讓問題解決者在應用這些規則時

不用再花心思考慮如何使用，而可以全心思索問題需求。專家擁有較完備的基模及自動化的規則，因此在遇到問題時可以較快地找到解決的方法，但初學者不同，因為缺乏完備的宣告性知識，當面臨問題時，一方面評估問題的情境，一方面找出可行的方法來解決，一方面學習新的知識，造成沉重的認知負載而無法達到學習的目標，而演練範例可用來解決這個問題，Cooper 及 Sweller(1987)在代數變換的問題上證實，透過演練範例可以讓學習者將注意力集中在問題本身、減少認知負載，讓學習者全心在解決問題的過程，學習效果尤佳。但演練範例本身也可能造成認知負載，因為演練範例的成效有賴於範例本身的原型(prototype)及與問題的相關性。如果演練範例需要學習者在各種資訊來源間分散注意力，則不會有好的學習成效，甚至可能比直接讓學習者解決問題成效更差。Ward 及 Sweller(1990)以物理上幾何光學及動力學為教材的實驗中也指出，結構化的演練範例可減低學生在心智上需整合各種資訊來源的認知負載，因此在測驗結果上優於傳統問題解決或未結構化範例。因此，演練範例的教學設計要能有效地降低學習者的認知負載才能促進學生問題解決的學習。

第二節 程式語言教學

壹、程式語言的教學目標

根據高中電腦課程標準，程式語言的教學目標在讓學生「瞭解程式語言的基本結構，以及運用程式語言解決問題」(吳正己、何榮桂，民 87)。為了達到此目標，在教學設計上應該「以問題解決為導向，再導入需要使用到的相關指令敘述」(吳正己、林凱胤，民 86)。邱貴發(民 79)認為程式語言教學包含「語言教學」及「設計教學」，從字面看，語言教學是指教導語言指令的概念及用法，設計教學則強調在設計方法來解決問題。但從實際教學觀點來看，兩者可視為同義詞，尤其在教初學者時，不可能也不應該把「程式語言」及「程式設計」分開。因此高中程式語言教學內容應包含「程式設計知識」及「程式設計策略」。Davies(1993)曾區別程式設計知識及程式設計策略兩者的不同，程式設計知識指的是宣告性知識，例如學習者能知道「for」迴圈如何運作，而程式設計策略則是能適當地使用程式設計的知識，例如學習者能正確地將「for」迴圈應用在程式中。Linn 及 Dalbey(1989)曾提出理想的電腦程式設計教學過程所產生的一連串認知成就，這些認知階段包括：(1)程式語言特徵，(2)設計技巧及計劃、測試、撰寫程式碼等程序性技巧，(3)問題解決技巧、知識及策略(包括使用程序性技巧)，且能整合特定語言中所學的知識及策略，應用到新的語言及情境中。由這一連串的認知成就可以發現，學生要先對程式語言特徵有確實瞭解才能運用程式設計策略來設計程式以解決問題。

綜合以上所述，程式語言教學的最終目標是培養問題解決的能力，透過設計程式來解決問題，也就是希望學生擁有程式設計策略的能力。而這些能力建構在程式設計知識上，因此在教學中同時要著重學生對程式語法及語義的瞭解。

貳、程式語言的知識架構

程式語言的教學目標在於讓學生瞭解程式語法、語義及促進問題解決能力，因此學生應習得這三項能力。許多研究在探討學習者發展這些能力的過程中所學習的相關知識，發現程式語言知識是以階層式的架構存在(Gagne, 1985; Pea & Kurland, 1984; Shneiderman & Mayer, 1979; Mayer & Fay, 1987)，透過這個知識架構，學生逐漸發展程式設計的能力。

以認知心理學的觀點來看，人類的「領域相關知識(domain-specific knowledge)」及「策略性知識(strategic knowledge)」可應用在任何形式的學習(Alexander & Judy, 1988)。其中領域相關知識由三種知識組成：(1)宣告性知識(declarative knowledge)、(2)程序性知識 (procedural knowledge)及(3)條件性知識(conditional knowledge)。宣告性知識是「知其然」(knowing what)的知識，用來陳述事實、觀念等一般性概念的知識。程序性知識是整合宣告性知識成為「知道如何做」(knowing how)的知識，例如：騎腳踏車、織毛線、打球等技能性的知識均屬於程序性知識。條件性知識是「知道何時做什麼事」的知識，當遇到一個新的情境時，能根據情境選擇適當的程序性知識來應用。

在程式設計的領域中，其知識分類與上述的知識分類大致相同，Bayman 及 Mayer(1988)將程式設計的知識分為兩大主要類別：概念性知識(conceptual knowledge)及策略性知識(strategic knowledge)。概念性知識指的是瞭解程式的架構、語法及執行流程，策略性知識指的是對問題的描述、分析、設計演算法及除錯的能力。當學生面臨問題時，先運用策略性知識分析問題、瞭解需求並設計方法解決問題，因此設計演算法求得解答的過程便是展現其策略性知識的表現。當形成解題計劃後，學生需要擁有概念性知識才能將設計的演算法付諸實

現，因此學生描述程式執行過程及撰寫程式碼的能力便是其概念性知識的呈現。其中概念性知識所包含的範圍相當廣泛，包含宣告性知識、程序性知識及條件性知識，此三類知識即與上述組成「領域相關知識」的三類知識相同。Deek(1998)根據程式設計的特性，將程式語言的概念性知識分為三個部分：(1)語法的知識(syntactical knowledge)，(2)語意的知識(semantic knowledge)及(3)實務的知識(pragmatic knowledge)。語法的知識即宣告性知識，學生要能知道程式語言的語法結構，例如在 BASIC 程式語言中，利用 For 迴圈的語法列印 1 到 5 的整數，其語法為：

```
For i=1 To 5
    Print  i
Next  i
```

在語法中要指定控制變數 i 來設定起始值及終止值，學生如果沒有指定控制變數而寫成：

```
For 1 To 5
    Print  i
Next  i
```

則會產生錯誤。語意的知識是程序性知識，學生要能瞭解語法所代表的意義，例如上述利用 For 迴圈的語法列印 1 到 5 的整數之程式中，學生要能瞭解第一次進入 For 迴圈內的敘述時，i 所代表的值是 1，第二次進入 For 迴圈內的敘述時，i 所代表的值是 2，依序進入至第五次以後會離開迴圈敘述。實務的知識則是知道何時要使用何種程序性知識，例如要求學生計算 1 到 100 的整數和，學生要能正確選擇迴圈的語法來達到題目的要求。

綜合上述學者研究結果(表 2-1)，程式語言的知識架構可分為概念性知識及策

略性知識，其中概念性知識是學生對語言的語法、語義之瞭解及使用，包含宣告性知識、程序性知識及條件性知識。策略性知識則是設計演算法得到問題解答的能力，即問題解決能力。而程式語言教學目標即在於促進學習者習得這些知識。

表 2-1

一般學習領域與程式設計領域之知識架構

<table>
<tr><td colspan="2">一般學習領域</td><td colspan="3">程式設計領域</td></tr>
<tr><td colspan="2">DeCort(1990)</td><td>教學目標</td><td>Deek(1998)</td><td colspan="2">Bayman & Mayer(1988)</td></tr>
<tr><td rowspan="3">領域相關知識</td><td>宣告性知識</td><td>瞭解語法</td><td>語法的知識</td><td rowspan="3">概念性知識</td><td>宣告性知識</td></tr>
<tr><td>程序性知識</td><td>瞭解語意</td><td>語意的知識</td><td>程序性知識</td></tr>
<tr><td>條件性知識</td><td rowspan="2">促進問題解決能力</td><td rowspan="2">實務的知識</td><td>條件性知識</td></tr>
<tr><td colspan="2">策略性知識</td><td colspan="2">策略性知識</td></tr>
</table>

參、程式語言與問題解決

程式語言知識架構中的策略性知識是指對問題的描述、分析、設計演算法及除錯的能力，即問題解決能力。當學生面臨問題時，先運用策略性知識分析問題、瞭解需求並設計方法解決問題。過去許多學者在探討學習程式語言是否能促進問題解決能力，雖然研究結果不一，但程式設計在本質上是屬於另一種形式的問題解決(Schwartz, 1988)，因此程式語言與問題解決仍有其密不可分的關係。以下針對過去研究者對程式語言是否能培養問題解決能力不同的觀點及看法進行探討。

Palumbo(1990)指出，學習程式語言就是在學習問題解決技巧。Papert(1980)也曾提出，小孩子學習 Logo 程式語言，可以建立其智慧模型(intellectual models)，並將此智慧模型應用到其它的情境以幫助思考。但 Pea(1983)研究 8 至 12 歲的學

童學習 Logo 語言發現，學童雖然很容易學習 Logo 的語法及語意，但並無法整合程式敘述以完成特定目標，因此孩童並沒有在 Logo 的學習過程中得到問題解決的方法。Pea 認為這是因為問題解決是複雜的認知過程，孩童並沒有能力自行從 Logo 的學習過程中領悟到問題解決的方法並應用到其它方面的問題，而是要靠老師在教學過程之中直接教導問題解決的技巧。然而 Logo 是適合小孩子學習的程式語言，許多研究也針對高中生或大學生探討程式語言學習對問題解決能力的影響。Palumbo 及 Reed(1991)的研究選擇以 BASIC 為高中生學習程式語言的工具，經過 15 週的課程，結果發現 11 位學習 BASIC 的學生在問題解決能力上優於另外 11 位控制組的學生。Choi 及 Repman(1993)則以大學生為研究對象，在經過 15 週的 Pascal 與 FORTRAN 程式語言學習後，結果在問題解決能力上均優於沒有學習程式語言的對照組學生。Norris 及 Jackson(1992)針對 72 位大學生進行一學期的 BASIC 程式設計課程，結果發現學生在課程中所學習的問題解決及關鍵性思考(critical thinking)能力確實有遷移到其它情境。由上述的研究結果來看，不同年齡層的學生有其適合學習的程式語言，且在經過一定時間的的學習後(如經過約一學期)，在問題解決能力上均有進步。

雖然上述實驗證實程式語言學習對問題解決能力有正面影響，但在過去的研究中也有不少結果並不支持此論點，認為學習程式語言對學生問題解決技巧並無影響。Pea 及 Kurland(1984)將 32 個學生分 Logo 組和非 Logo 組進行教學，結果發現在問題解決的學習成效上並無顯著差別。Gaffney(1984)的研究發現，學習 Logo 或 BASIC 語言並沒有增進小學生對非電腦問題的解決能力。Ennis(1994)以 20 個高中生為研究對象，將問題解決技巧整合到 BASIC 課程進行教學，結果在問題解決能力上與控制組並無顯著差異。Rucinski(1991)以一群實習教師分別教授

BASIC 程式語言及幾何學課程，結果發現程式語言課程對於學生的問題解決能力並無顯著影響。

深入探討過去許多實驗結果不一致的可能因素發現，實驗設計、問題解決本身的特性及學習遷移是主要的原因(Burton & Magliaro, 1987/88; Palumbo & Reed, 1991; Sloane & Linn, 1988)：

一、實驗設計

實驗設計直接影響到實驗結果，因為實驗進行的方式、評量的工具及評量方式、實驗時間長短、所選用的程式語言、教學方法等，都可能造成不同的結果。

不同年齡層的實驗對象應選用不同的程式語言來教學，各種語言有其特性，隨著年齡的成長，心智成熟度也不盡相同，選用相同的語言，不見得能達到真正的學習效果。以過去的研究設計來看，對於低年級的學生可選用 Logo 來教學，但對高中以上的學生，Logo 便略嫌簡單，應該選用 BASIC 或 Pascal 等語言。

在實驗時間方面，因程式語言與問題解決技巧的教學並非一兩個小時便能教授其觀念，至少要以一學期為單位，完整教授相關內容，並讓學生有充分時間練習，真正學習瞭解後，才能確實瞭解學生對問題解決能力方面是否有所影響。

教學的方法會影響學生學習成效，Lieberman(1985)指出，Logo 教學的效果與老師所扮演的角色及使用的教學方法有直接的關係，是否有以問題解決的理論為基礎發展其教學，往往會影響學生在問題解決的學習。

二、問題解決的特性

過去的研究為人質疑的便是問題解決能力是否可以評量、如何評量及是否公平等問題。因為問題解決為心智運作的過程，如何「讀心」是許多人在實驗時的一大挑戰，雖然有研究發展出相關的評量工具，如 Watson-Glaser Critical

Thinking Appraisal(Watson & Glaser, 1980)及 Thurstone Test of Mental Alertness (Thurstone & Thurstonw, 1986)，但是否還會有其它因素影響其評量結果，如學生本身對題目的閱讀能力及理解能力等，是實驗時較難控制的因素。

因為問題解決能力與其心智成長有極大的相關，而每個人心智發展的速度及程度亦不相同，外在環境及其它學習也可能影響問題解決能力。因此實驗的時間往往需要拉長，在實驗過程中，是否有其它因素干擾受測的學生，都可能造成不同的實驗結果。

三、學習遷移

除了上述所提兩大影響實驗結果的因素，另外還有一項因素是實驗設計者對實驗結果的設定不同，Salomon 及 Perkins(1987)指出，Clements(1984)與 Pea(1984)的實驗會有不同的結果是因為兩者所設定的學習是不同類型的學習遷移。因為學習遷移通常分為兩種類型—近遷移及遠遷移。近遷移(near transfer)又稱為垂直遷移(vertical transfer)，根據 Burton 及 Magliaro(1087/88)指出，近遷移指的是遷移到另一個問題解決領域，但與原本已經很熟悉的技巧相似，因此遷移的程度較小。遠遷移(far transfer)又稱為水平遷移(horizontal transfer)，指的是遷移到另一個問題解決領域，但與原本遷移的技巧明顯不同，遷移的程度較大。因為近遷移比遠遷移所需要遷移的程度較小，一般說來也比遠遷移容易達成。Daley 及 Linn(1986)提到，學過程式語言的學生較易將所學的概念(如：loop)遷移到另一種程式語言，但不見得容易將程式語言所學的演算法遷移到較不相似的環境(如：解河內塔問題)，因為這兩種遷移是屬於不同類型的學習遷移。

以這個觀點來看 Mayer, Dyck 和 Vilberg(1986)的研究，結果指出學習程式語言可以增進思考或問題解決技巧，但限於與程式語言中所強調之概念相關的部

分，且學生在問題轉換及程式理解上有進步，但並沒有證據顯示在一般能力(邏輯推理、視覺、語言能力)有改變。因為程式語言中強調的概念透過學習遷移到相關的思考或問題解決技巧是屬於近遷移，較遷移到一般能力的遠遷移容易達成。Clements 和 Gullo(1984)也有類似的實驗結果，在 l18 個 6 歲的小孩中，分別以 Logo 及 CAI 教問題解決，結果發現實驗組在反思能力、分散式思考及後設認知等特定思考技巧方面表現較好，但在一般的認知發展上並沒有差別，此結果同樣也可歸因於這兩種能力對於 Logo 的學習是分屬於不同類型的遷移。

不同程度的學習遷移設定會影響實驗結果，因此想要瞭解學習程式語言對問題解決能力是否有影響，要考慮學習遷移的程度對學生而言是屬於何種類型，不同的類型應該有其相對應的教法及施行的時機，而非一開始學生還在適應階段便設定希望學生能達到遠遷移的效果。另外還要考慮學生的背景能力，對不同背景能力的學生而言，遷移的程度亦不相同，因為遠遷移與近遷移是相對的，並非有絕對的劃分界線，所以程式語言對不同程度的學生在問題解決能力可能會有不同的影響。

綜合以上所述，程式設計確實能促進問題解決能力，許多研究結果也建議，以明確的問題解決方法教授程式語言可以引起問題解決的學習遷移到其它相關領域(Salomon & Perkins, 1989; Seidman, 1988; Shaw, 1986; Sloane & Linn, 1988)，且即使問題解決技巧沒有被學習或遷移，學生在學習程式語言的過程中，仍會對電腦的控制過程有相當程度的瞭解(Wiburg, 1989)。但程式語言教學者對於教學內容的設計、教學的方式及內容的設定會影響學習者透過程式語言學習問題解決的成效，因此在設計教材時應該深入瞭解問題解決的理論基礎，運用適當的教學策

略，才能達到促進學生的學習成效及問題解決的目標。

第三節 程式語言學習工具

壹、程式語言學習工具的選擇

不同程式語言的特性會影響學生建立一般性問題解決的技巧(Dalbey & Linn, 1985)，也是影響學生學習經驗的重要因素，所選用的程式語言要能符合教學目標(Cox & Clark, 1994)，因此當今電腦教育的重要議題是要慎選程式語言以達到學生的學習成效。

目前程式語言的種類繁多，但功能及用途各異，並非所有語言均適合初學者學習。以程式發展環境來看，視覺化的環境提供使用者開發圖形化介面的方法，在建立輸出入介面時，不必撰寫程式碼來描述介面元件的外觀與配置，只要使用系統所提供的工具便可達成。相對於視覺化環境，以文字為基礎的程式語言環境只提供基本的撰寫程式、編譯、除錯的程式寫作環境，使用者要自行撰寫程式碼建立輸出入介面。不少程式語言初學者對於要花許多時間學習語法才能寫出一個簡單的程式感到挫折，且單純文字編輯環境也容易讓學生感到無趣(Bishop-Clark, 1998)。而視覺化環境則改善了這個問題，初學者對於不用具備許多程式語言概念便可開發視窗化程式的學習方式感到興趣。

以設計的基本觀念來看，程序式(procedural)程式語言與物件導向(object-oriented)程式語言是目前兩大主要類別(Robins, Rountree & Rountree, 2003)。程序式程式語言是眾多程式語言中最基本的，也是最常見的一種，程式本身是透過一連串指定敘述組合而成。指定敘述是將一些子運算式加以組合而成的表示式，只要依序執行這些敘述就可以得到結果。程序式程式語言通常提供幾種基本的控制流程結構，例如：循序流程結構、選擇流程結構、迴圈流程結構，在寫作的過程中，設計者要根據所要解決的問題，找出解決問題之演算法寫成程

式。相對於程序式程式語言，物件導向程式語言的概念主要架構在物件上，藉由物件與物件之間的互動完成問題的解答，比較符合真實世界的環境。一個物件包含屬性(property)與此物件如何處理與執行的方法(method)，設計者的工作便是將所有的物件連結起來以解決問題。Funkhouser(1993)認為物件導向的方法具有「自然、容易使用及功能強大」等特性，因為物件在程式領域中的本質與問題領域中本質相符合，學習者可以很容易地連結兩個領域並促進程式設計的學習成效。因此物件導向程式設計反應真實世界的問題解決，有助於問題解決學習。但部分研究並不支持此論點，Detienne(1990)認為物件的概念並不容易理解，問題領域與程式領域間的連結並不直接，因此在問題領域中對物件的概念，不一定對程式領域的物件觀念有所幫助。

高中電腦課程標準在程式語言這部分並沒有特別指定所使用的語言為何，而一般的教科書是以 BASIC 為教學工具。BASIC 語言是在 1964 年，由 Dartmounth 學院的 John Kemeny 與 Thomas 共同發展的一套電腦語言，是專門設計來供初學者學習的程式語言。由於 BASIC 語法接近人類所使用的自然語言(英文)與數學算式，因此廣受電腦初學者的喜愛，1970 年代，BASIC 已成為教學上主要的程式語言，許多學校將它列為學習電腦語言的入門課程。Quick Basic(QB)是 BASIC 語言其中一個版本，屬於程序式的程式語言，提供整合發展環境，雖然採用文字模式，但提供設計者一個撰寫程式、編譯、除錯的程式寫作環境，成為早期教授高中程式語言的教學工具。然而近年來視窗環境盛行，多數教科書也改用視覺化發展環境的 Visual Basic(VB)作為教授程式語言的工具。VB 架構在 BASIC 這種程序式的語言上，但因為其視覺化的設計方法，在設計時都是由介面物件開始設計，然後再應用事件驅動的視窗程式運作模型逐一填入各個介面所引發的程式動作。雖

然 VB 並不屬於物件導向的程式語言 (VB 發展到 6.0，包含 6.0 以前的版本並沒有具備完整物件導向程式語言的特性) ，但 VB 執行的過程是以物件為主，可以說是基於物件(object-based)的程式設計方式，與 QB 採用程序導向設計方式不盡相同。

QB 與 VB 兩種程式語言其基本語法相同，但發展環境及設計概念上有所不同，應用在教學上對初學者的學習成效也可能有所差異。下一部分將探討 QB 與 VB 應用於程式設計教學的情況。

貳、QB 與 VB 應用在程式語言教學

QB 與 VB 雖然都是以 BASIC 的語法為主, 但 VB 基於物件的特性及視覺化的開發環境，與 QB 程序導向及文字介面的撰寫程式方式有極大的不同，對於初學者學習程式語言的成效也可能不盡相同。過去許多教學者對於兩個語言在教學上的影響有許多相關的討論(Martin, 1999；Oberman, 1999；White, 1997；Bishop-Clark, 1998)，但並沒有定論何種語言最適合初學者學習。Matrin(1999)認為 VB 雖然讓學生覺得在使用上較為有趣，但由學生所寫的程式顯示，學生的學習著重在 VB 表面的知識，而非在瞭解程式語言的概念，因為學生太依賴 VB 內建的物件, 將學習重心擺在物件的屬性及使用上。Holt(1995)也建議物件的概念應該在學生已學完程式的基本概念(如迴圈、陣列、副程式)後再介紹較為適合。Matrin 並指出六項高中生使用 VB 學習程式語言所引起的困擾：

(1) 程序：學生常出現的迷失概念在於認為物件就是程序，這是因為 VB 程式碼可以和物件連結在一起，因此造成學生的錯誤概念。

(2) 可讀性：學生不能一次看到所有的程式碼，只能機動地透過物件看到與其相

關的程式碼。但這是因為 Matrin 的課程所使用的版本為 3.0，這個問題在 5.0 之後的版本便已解決。

(3) 變數：VB 的物件包含名稱(names)及類型(types)，可能與變數造成混淆，例如將標籤物件(label)的標題(caption)屬性誤當成是變數。

(4) 資料型態：VB 允許使用者設定自由型態的變數，造成學生重覆宣告變數的錯誤。

(5) 陣列：VB 的物件也有內建類似陣列的用法，但並非真正的陣列，因此在教學過程中還要多花時間解釋二者之間的差異，且通常只會造成學生負面的學習經驗。

(6) 線上輔助：線上輔助的內容太過專業，對於初學者並沒有幫助。

因此 Martin 認為 VB 並不是適合初學者學習。

Oberman(1999)則針對 Martin 的論點提出相反的看法，他認為 VB 是一個非常適合初學者學習的程式語言，並提到採用 VB 來教學的兩個理由是想要讓程式作業更吸引人及讓學生所寫的程式更符合日常生活的問題解決。Oberman 除了針對 Martin 所指出學生的困擾提出反駁外，並提出 VB 對初學者的優點：

(1) 直覺的本質：VB 提供快速建立視窗環境的方式，與學生平常習慣的視窗操作方式相同，且 VB 的指令按鈕(Command Button)之設計方式與口述命令的情境相同，學生可以用更直覺的方式來設計程式。

(2) 模組化：VB 的模組化設計方式可以簡化教授副程式及重覆使用程式碼的觀念。

(3) 繪圖：相較於文字介面程式語言，VB 視覺化的特性讓設計者在繪圖功能上更節省時間。

(4) 與使用者相關：程序式的程式語言能很容易地按照順序安排程式的發展，但使用者的需求通常是希望依照不同的需求而有不同的回應，而 VB 基於物件的特性正好可以符合這樣的設計方式，因此 VB 較容易與實際生活的需求設計相結合。

因此 Oberman 認為 VB 是非常適合初學者學習的程式語言。

White(1997)比較程式語言初學者使用 QB 與 VB 為學習工具在程式作品與學習成效上的影響，雖然從統計數字看來並無顯著差異，但由學生的作品及反應可發現，初學者使用 VB 比使用 QB 能發展出更好的程式，因此他仍相信導入物件的設計觀念能提供學習者更容易的方式來瞭解資料及控制結構。Bioshop-Clark(1998)以相同的教學目標對兩組不同的學生分別以 QB 及 VB 進行教學，結果兩組的學生在循序、選擇、重覆概念的理解表現沒有顯著差異，但在某些情況下，VB 組的表現更好，因此 Bioshop-Clark 認為 VB 適合程式設計初學者學習。

第四節 歸納與結論

雖然過去的研究結果對學習程式語言是否能促進問題解決能力並無定論，但程式設計的過程就是問題解決的過程，因此程式語言的教學活動中包含了問題解決教學。促進學生對程式語言的瞭解及問題解決能力是程式語言教學的主要目標，但學生的先備知識，教學活動的策略及學習工具都會影響學生的學習成效，因此應該適當地設計程式語言教學活動以增進學生的學習成效。以下歸納五點促進學生學習成效之教學設計：

(1) 提供學生相關的概念模型

學生學習程式語言最普遍的困難便是缺乏電腦完整的概念模型，無法瞭解程式執行時在電腦內部的運作過程，例如學生不易瞭解變數在記憶體中運作的觀念。過去的研究發現，概念模型可以是升學生的程式語言學習成效(Bayman & Mayer,1988; Wu, Lin & Hsu,1997; 吳文鴻，民 91)，因此提供學生適當的概念模型，如將變數在記憶體中的運作過程視覺化，以幫助學生瞭解。

(2) 選擇適合學生的程式語言為教學工具

不同年齡層的學生其心智發展的程度不同，適合學習的程式語言亦不盡相同，以過去研究來看，因為 Logo 的使用介面及方式較為親切，較適合用來教授給學齡前或低年級的初學者，而 BASIC、Pascal 等程式語言其編輯畫面雖不如 Logo 方便及人性化，但對高中生及大學生而言，其心智發展已有一定程度，應有能力在老師的教導下學習這類程式語言。

除了配合學習者的心智年齡來選擇適合的程式語言進行學習，程式語言本身的特性也會影響學習成效，因此深入瞭解語言本身所具備的特性，配合教學目標，選擇適合的程式語言學習工具，才能促進學生的學習成效。

(3) 以問題導向的教學方式將問題解決策略整合到教學活動之中

不論選用何種程式語言，應該將之視為一種工具，語法及語言的特性應適當教授即可，而非過度強調。目前程式語言的教學仍是以語法為主的教學方式，練習的範例也以語法為主，學習成效自然不佳。教師在設計教學活動時，應以問題導向的方式進行教學，讓學生瞭解學習程式語言的目標在於問題解決，而非只是語法的練習。且在教學活動中整合問題解決的技巧及步驟，讓學生從設計程式的過程中練習問題解決，並習慣解題步驟及技巧，以促進學生問題解決的學習。

(4) 提供適當的範例供學生練習

學習遷移有近遷移及遠遷移兩種不同的類型，一般說來，近遷移比遠遷移容易達成。適當的範例會幫助學生學習，因此在教學設計上應配合學生學習的情況，給與適合的範例，從近遷移的相關例子循序漸進到遠遷移的題目，學生才能從範例練習中獲益。另外學生往往因為缺乏解題策略及方法，無法順利設計程式，教師應適時提供專家的解題策略供學生參考，建立其解題的信心。

(5) 進行後設認知技能的訓練

後設認知技能讓學習者在學習活動中能適當地監控自己的行動，因此能幫助學習者在問題解決時更有彈性地作出決策。過去研究發現，學習者透過自我解釋、自我監控等後設認知技能的訓練，能幫助學習者進行新舊知識的連結，有助於增進學習者程式語言的問題解決學習成效(林育聖，民 91； 羅漢村，民 92)。