封面 Cover Page

教育部教學實踐研究計畫成果報告

Project Report for MOE Teaching Practice Research Program

計畫編號/Project Number：

學門專案分類/Division：

計畫年度：■111 年度一年期　□110 年度多年期

執行期間/Funding Period：2022.08.01 – 2023.07.31

運用思考可視化於程式設計課堂以發展學生的程式設計思考能力

Python 程式設計

計畫主持人(Principal Investigator)：莊永裕

協同主持人(Co-Principal Investigator)：

執行機構及系所(Institution/Department/Program)：國立中央大學/資訊工程學系

成果報告公開日期：□立即公開 ■延後公開（統一於 2025 年 7 月 31 日公開）

繳交報告日期(Report Submission Date)：2023 年 9 月 19 日

本文與附件 Content & Appendix

運用思考可視化於程式設計課堂以發展學生的程式設計思考能力

Introducing making-thinking-visible in a programming course to help students develop their programming thinking

一. 本文 Content

1. 研究動機與目的 Research Motive and Purpose

申請人於以往開設程式設計、程式語言課程的教學現場，發現有如下三個問題仍待解決，並認為這些問題均圍繞於學生們是否妥善地發展程式設計的思考過程，以深耕程式設計能力。申請人認為這些問題間有極大的相關性，會決定學生們未來是否能應對軟體的各種設計與變化，以及學習全新程式語言的快慢與否，影響未來任職軟體研發工程師的能力：

問題 (1)

程式設計非常需要自主思考能力，但學生卻容易流於制式化的套用

程式設計已被認知為一個複雜形式的人類行為 (Weinberg, 1971)，因此可能需要花上十年的時間才能脫離新手的程度，跨越門檻成為高手 (Winslow, 1996)。雖然近年許多程式語言採用直譯器的形式，甚至採用視覺化的方塊表現程式語言結構，但這都只降低了入門時某方面的門檻而已，未能真正協助初學者理解程式設計的方法。因為程式設計的本質是運用知識的策略，而不只是知識本身 (Davies, 1993)。像 Python 這類程式語言採用了直譯器的方式，雖然可以讓初學者省去學習了解編譯的過程，且能只撰寫一部分程式碼隨即執行並看到結果，但要如何運用所學的知識來撰寫這一點並沒有任何改變；而 Scratch 這類視覺化程式語言，能讓初學者輕鬆地用拖曳（Drag-and-Drop）的方式進行程式設計，不僅減少了輸入陳述（Statement）時可能產生的語法錯誤，也提高了學習者的興趣，但也有不少學習者只學到重混（Remix）來產生創意，沒能深入理解程式運作的原理。

在申請人的教學現場裡，經常發現很多學生只想問「怎麼做到這件事」、「這樣寫為何不會那麼動」之類的問題，只看重題目與答案的對應關係。有些學生甚至在自己的答案（亦即程式碼）無法符合題目要求時，會直接刪除所有答案，剪貼（Copy-and-Paste）所謂的正確解答來執行，以求「答對」。但程式設計其實是一個思考的過程，也是一個創作行為，比起是非或是填空，更像是應用問題甚至作文，不見得只有單一個解法，而一個解法對於不同狀況也有不同的變化。對有責任培養學生成為專業軟體人才的資工系所來說，申請人認為這是一個很大的問題：當未來在職場上需要運用程式設計能力以實作出產品時，不僅無法有創意地設計出令人驚艷的特色，即便在給定的規格下，如果不是自己曾看過的實作手法，也會寫不出來或錯誤百出；甚至在程式設計的過程裡，有意無意地「借用」他人的程式碼，衍生授權與抄襲的法律問題。

問題 (2)

即便能找到良好的程式碼範例，也難以從中學到思考程式設計的技巧

現在不像以往程式範例不易取得，無論是作業的經典題型，或是常見的程式設計模式應用例子 (Gamma et al., 1993)，在網路上隨意搜尋均能取得公開的程式碼，

然而許多學生卻無法很好地藉此吸收、學習到當中的程式設計技巧。像是資料結構與演算法裡經典的各種排序法，很容易就能找到以各種程式語言實作的範例程式碼，有時也會在網頁裡附上詳盡的解說，或是程式碼當中有親切的註解。然而這些說明都是針對該程式碼，亦即「答案」的解釋，而非對於問題本身的求解過程。由於對於給定的問題，每個人思考的方式多少都有些差異，過程中產生的疑問自然也不同，看解答時不見得能獲得自己想知道的細節。比如說對於一個氣泡排序法的範例程式碼，初學者未必能理解為何需要二重迴圈，就申請者的教學經驗裡，學生們容易混淆與產生疑問的地方也不盡相同，像是當中的兩層索引之間的關係，或是陣列的邊界條件等。

另一個不容易從範例程式碼學習到程式設計思維的可能性，申請人認為是在於無中生有的過程，亦即為何對於給定的問題給出這樣的解答，這是不容易從所謂答案的程式碼逆推回去的。關於程式設計的書籍或網頁，或許礙於篇幅的緣故，也很少著墨於思考撰寫時的中間過程，這很可能是學習者在試圖透過模仿來學習時，途中出錯卻不知為何發生也不知如何解決的原因。因為這樣一個思考的過程是看不見的 (Perkins & Ritchhart, 2003)，沒有透過適當的引導與問答無法找出問題點；即使是抽象化為運算思維的學習，也需要知道學生的個別學習狀況並據此提供不同的協助 (Hsu et al., 2018)；而程式設計更是需要具體落實到程式碼撰寫，這段思考過程有相當的一段「距離」，該過程會遇到的阻礙也因人而異。

問題 (3)

程式設計專題題材不一定有創意，且分工效果不佳

申請人在教學課程裡，都會在期中講授完基礎知識並完成個別技術的練習之後，讓學生們準備期末的程式設計專題。過程中也會採取分組的方式來激發討論，並期望組員能各自負責部分技術實作來完成產品，也強調程式開發動機的說明。儘管也常有優秀的組別能有出人意表的點子並能很好地實作出來，但是不少學生還是侷限於重製他們接觸過的遊戲或是網路上找到的範例，有時甚至未能與前面所學的技術連結。就申請人的觀察，有些學生也未必從這段專題實作的過程獲得對程式設計的反饋與省思；在這樣一個產品的設計與開發過程中，如果沒有感受到程式設計的困難可說是有些不可思議的。因為前面學到的各個實作技術都是一種知識，儘管講解了運用這些知識的策略，實際上要拿這些知識來套用問題求解並不容易，需要有理解問題、思考計畫、執行計畫、回顧反饋這四個階段 (Polya, 1945)，而求解策略的忽視正是程式初學者常遇到困難的原因之一 (Perkins & Martin, 1986)。因此，申請人認為在課堂裡加入檢視學生們的思考過程、協助他們擬定計畫並自我反饋的手法，才能確實達到程式設計教學的效果。因為藉由帶領學生們思考他們答案裡錯誤的成因，能獲得更好的學習效果，而不是只解釋最後成功的程式碼。

另一個不容易消除的常見狀況則是分組專題裡的分工效果不佳及參與態度消極，這往往發生在學生們在專題開始時實作能力出現落差時。原本專題的意義除了在討論創意與思考過程之外，組員之間也能產生教學相長的效果，互相分享技術與學習歷程，並增加對程式設計的自信與成就感；然而就申請人在課堂的觀察，分工效果不佳其實表示一開始就將各部分職責分開，將預計程式設計較有難度的部分給予實作能力較強的學生進行，而之後的專題進行時對於技術的討論變得較少，因此分享與交流的效果不如預期。其實申請人認為分工不均本身並非問題，而是學生們喪失了教學相長與切磋琢磨的機會。近年來程式設計的學習相當普遍，尤其是資工系學生們，在入學前或多或少接觸過，彼此間的程式設計能力也有相當差距，如果能藉由專題的討論進行，既能讓程式設計能力不佳的學生有所成長，原本就很擅長的學

生也能藉此檢視自己的思考過程，透過對同儕的解惑來反思自己的解答，獲得進一步提升能力的機會。

總結對於上述在教學現場觀察到的問題，申請人認為直譯器與視覺化只能降低入門時某方面的門檻，而未能協助理解程式設計的運用策略。亦即 Python 採用直譯器可只撰寫部分程式碼隨即執行，可讓初學者容易開始，省去學習編譯的過程，但與運用策略的學習無關；另一方面，Scratch 採用方塊結構可減少輸入時可能的語法錯誤，讓初學者可輕鬆地拖曳方塊來撰寫程式，提高學習興趣，但也同樣無法促進對於運用策略的理解。程式設計是個不可見的思考過程，卻常以解題測驗來觀察學習狀況；儘管檢驗答題可有效得知學習成效，通常僅能由程式的輸出入進行黑箱檢驗，仍然無法得知初學者卡在哪裡？如果老師能得知學生的思考過程，便能適時給予協助來學好運用策略，並培養自主思考與檢視過程的良好習慣。

2. 研究問題 Research Question

前述的實際教學現場問題，促使申請人思考如下的研究問題：是否能引導並檢視學生程式設計思考過程，以協助學生找出問題？本研究運用哈佛大學 Project Zero 的思考可視化研究 (Perkins & Ritchhart, 2003)，導入異質分組的小組合作學習來促進「可視化」的實踐。

也就是說，儘管學生對課程內容很感興趣，學習上卻有前述的問題，對此本研究分析其原因為思考的不可視，亦即程式設計的思考難以被呈現。這導致學生自己既無法釐清自己的困難點，也無法向老師提問，更難以與同儕討論；同時，老師也無法得知學生卡在哪個癥結點，沒有辦法給予適切的協助。

對於這個狀況，本研究遵循思考可視化的概念，對於程式設計的範例與練習題設計一個相對應的思考模組，均使用逐步問答的學習單呈現，用以輔助學生思考並記錄自己的狀況，得以一步步釐清思考的正確性與困難點。我們將這樣的學習單連同題目本身、以及解答的程式碼視為一個學習模組。學生在學習某個程式設計的概念時，可以像數學的應用問題一般思考解法，且有多個小題來引導，在思考這個應用問題的解法時，可以釐清自己對於這個解法的逐步思考過程，而不是直接嘗試直接寫出結果的程式碼；畢竟程式碼當中的執行錯誤，往往只是結果錯誤，而即便能獲得直譯器的錯誤警示，也未必能正確反映自己思考上的盲點。因為程式碼是整個思考完成之後撰寫出的結果，每一行程式碼並非思考的過程。本研究以設計出符合思考可視化手法的學習模組，輔以異質分組的小組合作學習，讓學生透過同儕學習與教學相長的效果，落實自己對於可視化的練習。

3. 文獻探討 Literature Review

程式設計的學習難度其實是在於它的一個思考的過程，需要在心裡模擬電腦程式的執行，亦即假想程式碼會如何於電腦裡被解讀與求值運作（Evaluation）。換句話說，它是一個將執行期間的動態變化，嘗試以靜態的程式碼來投射表現的思考行為。如同現在電腦架構裡的內除程式概念（Stored Program），程式必須要能被儲存並再次載入執行(Knuth, 1970; Neumann, 1993; Turing, 1937)，進行程式設計時必須要能想像並解讀這個過程。初學者與專家的巨大差異正式在於這個能力的強弱，初學

者需要從專家學習這樣的知識理解以及知識運用策略。因此申請人認為在教學現場裡正視這個思考過程非常重要，與其快速地寫出程式碼來執行試誤，不如將時間重點放在釐清過程想法與精煉程式碼，謀定而後動，減少後續迷失的時間成本。

至於如何將這段思考過程釐清，讓它變得清晰可見能與同儕討論，則可參考思考可視化（Make Thinking Visible）的研究 (Perkins & Ritchhart, 2003; Ritchhart et al., 2011; Ritchhart & Perkins, 2008)。藉由為思考過程準備思考模組（Thinking Module），並訂定相對應的思考常式（Thinking Routine），可讓學生們根據這些模組寫下他們的想法，讓這些想法變得具體看得見，因而得以進行比較與討論。以程式設計的場合來說，許多學生容易有邏輯的跳躍，或是始終誤以為某個環節是絕對正確的，導致陷入錯誤迴圈當中無法跳脫，但旁人看來卻無法了解其問題所在。我們可以對程式設計的命題設計相對應思考模組，對於一段範例程式碼或練習題，以一個表單呈現對各個程式設計思考步驟的問與答，讓學生們依此表單填寫自己的程式設計想法，藉此和同學們討論以釐清自己的程式設計思考狀況。比如說對於氣泡排序法，其相對應的思考模組表單裡列出了像是「對於排序這件事預期的結果是什麼」「排序時每一個步驟的操作是什麼」「排序時打算使用哪個運算子進行比較」「排序時預計如何進行交換」等等。開發出配合給定問題的思考模組表單這件事，即為本計畫在課程前的執行重點。當藉由這樣的思考模組來讓整個程式設計的思考脈絡顯得完整之後，便可落實至程式碼撰寫。

和同學們討論的部分，我們則導入小組合作學習（Cooperative Learning）的方式 (Hwang et al., 2012; Slavin, 1980)。為了產生教學相長的效果，我們也將使用異質分組（Heterogeneous Grouping），讓一個小組裡有程式設計能力不同的同學，可以互相分享討論學習經驗與互相獲得建議。藉由這個方式，可以期待讓程式設計能力較強的同學練習解釋自己的思考出發點，使得自己更加釐清並學會表達，進一步增進自己的程式設計能力；而程式設計能力相對較弱的同學，便能模仿並學習這些程式設計的思考方法，學到如何思考建構一個程式，增強程式設計的能力。當小組的成員一起釐清對一個命題的程式設計思考之後，可以一起進行程式設計將這個思考落實為程式碼。此時可以讓學生們自由地合作撰寫程式碼，或是嘗試軟體工程實務的群體程式設計（Mob Programming）。群體程式設計是一種軟體開發的手法，讓整個團隊在同個時間、同個地點、以同一個電腦來一起工作於某個項目 (Shiraishi et al., 2019; Zuill & Meadows, 2016)。其中輸入程式碼之角色由一人負責，稱為駕駛員（Driver），其他的人扮演領航員（Navigator）的角色，負責討論點子並引導駕駛員產生程式碼。駕駛員的工作有團隊成員輪流擔任，一般說來會以 15 分鐘輪替。這個程式設計的進行方式，乍聽之下會認為因為將多人拉進同一份工作裡而導致效率變差，但在許多實務現場裡卻發現程式碼品質獲得提升，修改協調的時間減少了，最終整體的開發時間得以加速。當小組成員一起撰寫程式碼時不一定要嚴格遵循這樣的方式，但可以了解這種開發方式的存在。

4. 教學設計與規劃 Teaching Planning

申請人於自己開設的「Python 程式設計」課程裡進行本教學實踐研究的規劃，該課程為資工系選修。前 4 週為基礎概念的說明，採用以往的教學方法，第 5 週之後採用本計畫的教學實踐手法。這是考量前半課程內容相對單純，且集中於個別學習各種基礎知識，像是邏輯運算思維、程式語言結構、開發環境建置等內容，較無

牽涉程式設計的策略運用學習。而在課程後半進入各種程式設計應用，並開始進行分組專題實作，便會針對給定的範例學習具體的程式設計概念，每個範例也都有搭配的練習題實作讓小組進行，培養思考過程與團隊精神。

在第 5 週之後的實際運作上，我們在第 5 週進行程式設計能力測驗，並據此將全班 60 位學生平均分為 12 組，這裡的平均是以程式設計能力測驗的結果由高至低以 S 型分組，亦即讓程式設計能力較高的學生分散至各組，也讓能力沒那麼好的學生可以平均分配至各組，讓每組裡有能力不同的學生可以互相討論，使用給定的思考模組學習單逐步討論程式解題。這樣的規劃是基於不同程度的學生可能對於題目的理解不一樣，擅長的地方與感到困難的地方可能也不同，如果能在思考模組學習單的輔助之下逐一檢驗確認同儕的思考過程，應該能有助於增進理解，也能讓能力較佳的學生試著將解法說出來，讓能力較弱的學生試著表達感到困難之處，以求教學相長的效果。

5. 研究設計與執行方法 Research Methodology

本研究的目標為協助學生找出思考過程的問題並練習說明，以期發展自我的程式設計思考能力，解決習慣以樣板套用來學習的問題。在授課對象的部分，原本該課程設定為大三以上，期望讓資工系到了大三程式設計能力仍然不夠的學生，能在這個機會自己加強；不過依審查委員建議，本研究的授課對象應該更明確鎖定沒有太多基礎的學生，因此開課前修改授課對象為大二以上。

在研究的設計與執行上，申請人帶領助教在課程前針對每一個範例程式碼、練習題，設計出相對應的思考模組。該思考模組是以學習單的方式呈現，引導學生進行群體程式設計與討論。比如說，對於氣泡排序法，我們列出了像是：

「對於排序這件事預期的結果是什麼」

「排序時每一個步驟的操作是什麼」

「排序時打算使用哪個運算子進行比較」

「排序時預計如何進行交換」

而非只是給予輸入資料來檢查輸出資料。不同於以往教學裡讓學生自行思考解法、寫出程式碼直接執行、並當程式執行有問題時再回過頭去檢視程式碼，我們採取的是讓學生們練習寫下自己對這個問題的每一個思考步驟，亦即讓思考的過程可視化，而不是單看程式碼執行狀況或結果。

以底下的圖 1 為例，該程式練習是在學習深度學習函式庫的進度時使用。該段進度先講授深度學習的基礎概念，並了解相關函式庫運用之後，學習如何利用事先訓練好的 VGG16 模型，來辨識給定的照片是否為貓或狗。更正確地說，是判斷給定照片裡的動物被認為是各種貓或狗的可能性為多少。這裡還沒有讓學生動手訓練模型，而是先學會如何寫出程式來運用已經訓練好的模型。在底下的函式說明部分，先讓學生們看一次範例程式碼，了解其運作與思考，接著給予如圖 2 的實作題目，分為多個小題逐步以問答的方式確認學生思考該解法的狀況。這裡的圖 1 與圖 2 是配套的範例程式碼與練習題，均讓學生們以分組的方式討論並填寫思考模組學習單。在圖 2 的實作題目裡，我們重點放在確認學生思考的過程而非程式碼撰寫結果，同時為了避免讓學生認為這非常簡單而不需寫出，採取分組討論並要求寫下紙本，這讓學生們可以落實隨堂練習。實際上我們在課堂裡也觀察到這樣進行的效果，許多時候學生之間會發現彼此的思考過程意外地不同，而有熱烈討論。

圖片辨識程式練習

姓名:_____

學號:_____

範例

利用預訓練的VGG16模型，預測圖片中的內容，顯示該圖片，並列出對該圖片預測的前十個結果。

示意結果如下:


```
('n02105855', 'Shetland_sheepdog', 0.16731805)
('n02110185', 'Siberian_husky', 0.1174201)
('n02106166', 'Border_collie', 0.03971494)
('n02086910', 'papillon', 0.037853315)
('n02971356', 'carton', 0.037699178)
('n02110063', 'malamute', 0.03647313)
('n02109961', 'Eskimo_dog', 0.0348393)
('n02445715', 'skunk', 0.033449266)
('n03085013', 'computer_keyboard', 0.029057832)
('n02106030', 'collie', 0.026934102)
```

參考下方的範例程式碼和下方的說明。

範例程式碼

```
import matplotlib.pyplot as plt
import keras.utils as image
from keras.applications.vgg16 import (VGG16,
decode_predictions, preprocess_input)
from numpy import asarray, expand_dims
from PIL import Image

def predict(path: str, feature_size: int):
    x = image.img_to_array(image.load_img(path,
target_size=(224, 224)))
    x = expand_dims(x, axis=0)
    return
decode_predictions(model.predict(preprocess_input(x))
, top=feature_size)[0]
```

```
def showimg(path: str, imgname: str):
    plt.figure(figsize=(15, 15))
    plt.subplot(2, 5, 1)
    plt.title(imgname)
    plt.axis('off')
    plt.imshow(asarray(Image.open(path)))
    plt.show()
# 載入完成學習的模型VGG16
model = VGG16(weights='imagenet')
# 判斷影像
path = "test/50.jpg"
plt.figure(figsize=(20, 10))
showimg(path, "picture")
results = predict(path, 10)
for result in results:
    print(result)
```

此處的範例程式碼是使用深度學習模型VGG16預測圖片的程式碼。
這裡包含了 predict和showimg兩個函式。

函式說明

(1) predict 這個函式的輸入為圖片的路徑和希望回傳的預測結果數。函式中會將圖片重新調整大小和格式成符合VGG16模型要求的格式，再以模型進行預測，並以tuple形式回傳模型預測的結果，包含"編號"、"名稱"和"相似度"。

(2) showimg這個函式的輸入為圖片路徑和圖片名稱。可以根據設定的格式顯示出圖片，函式中的可選設定包含"圖片大小"、"子圖設定"、"圖片標題"、"刻度顯示"等。

圖 1 圖片辨識程式練習的思考模組學習單

實作題目

利用預訓練的VGG16模型，預測圖片中的內容，顯示該圖片，並列出對該圖片預測的前十五個結果。

1. 一共有200張圖片(檔名為1~200.jpg)、使用根據自己的學號 % 200 + 1的jpg圖片進行預測
2. 圖片網址: https://reurl.cc/lvrN2j
3. 請建立VGG16 imagenet model、將feature_size設定成15
4. 把圖片標題 改成預測結果中相似度最高的名稱

示意結果如下:


```
('n02106550', 'Rottweiler', 0.8525288)
('n02101006', 'Gordon_setter', 0.035721004)
('n02108551', 'Tibetan_mastiff', 0.014799018)
('n02096294', 'Australian_terrier', 0.014474452)
('n02105412', 'kelpie', 0.010754345)
('n02100877', 'Irish_setter', 0.009715852)
('n02096051', 'Airedale', 0.008588539)
('n02107908', 'Appenzeller', 0.005901348)
('n04254680', 'soccer_ball', 0.0040339218)
('n02097474', 'Tibetan_terrier', 0.0036161565)
('n02107142', 'Doberman', 0.0033236071)
('n02111277', 'Newfoundland', 0.0030919744)
('n02107683', 'Bernese_mountain_dog', 0.0028606358)
('n02099601', 'golden_retriever', 0.0027757664)
('n02099267', 'flat-coated_retriever', 0.002587885)
```

參考下方的範例程式碼，一步步完成表格中的問題。

*** 不需要填寫程式碼空缺**

範例程式碼

```
import matplotlib.pyplot as plt
import keras.utils as image
from keras.applications.vgg16 import (VGG16,
decode_predictions, preprocess_input)
from numpy import asarray, expand_dims
from PIL import Image

def predict(path: str, feature_size: int):
    x = image.img_to_array(image.load_img(path,
target_size=(224, 224)))
    x = expand_dims(x, axis=0)
    return
decode_predictions(model.predict(preprocess_input(x)
), top=feature_size)[0]

def showimg(path: str, imgname: str):
    plt.figure(figsize=(15, 15))
    plt.subplot(2, 5, 1)
    plt.title(imgname)
    plt.axis('off')
    plt.imshow(asarray(Image.open(path)))
```

```
    plt.show()
```

與前題相同，此題是使用深度學習模型VGG16預測圖片的程式碼。不同處在於預測用的圖片、輸出的預測結果數、圖片標題。

函式與參數

(1) 預測用的圖片是根據輸入函式predict的參數path決定的。因此可以更改path來決定要使用的圖片檔名。

(2) 第二個要求是更改feature_size，也就是更改獲得的預測結果數。這也同樣可以更改輸入函式predict的參數來實現。

(3) 第三個要求為更改圖片的標題，要求更改成預測結果中相似度最高那項的名稱。這一步需要的是獲取第一名的預測結果名稱，並加入showimg的imgname中。若像是範例題一樣以results接受predict的回傳值，那麼應該如何表示第一名的預測結果的名稱?

圖 2 對應於圖片辨識程式練習的實作題目之思考模組學習單

為了瞭解學習進展，我們從程式設計能力的前後測、學期總成績來進行觀察，並和前一年進行比較。雖然前一年同課程由於未執行教學實踐研究計畫而沒有全面要求學生填寫前後測，有效樣本數較少，但仍有些數據可以參考比較。

此外，我們安排助教觀察每週學習單的討論填寫、期末專題來了解學生們的學習狀況。本校也支援聘用一位助教觀課，撰寫週誌並拍攝照片。最後對於期末專題的部分，也給予學習單，促進主題的討論與實作；這部分的學習單比較不是引導逐步的思考過程，而是以分項詢問的方式。如圖 3，顯示了兩組學生填寫期末專題進度紀錄的學習單成果，可以看出有熱烈的討論與豐富的構思過程，甚至對於分工也可以有更近一步的討論（這裡將寫有學生姓名的部分遮蔽）。圖 4 則為某組學生填寫期末專題的實作中/後討論之學習單成果（版面排列緣故將一張學習單的上下部分至於圖 4 的左右排放），雖然填寫的量還能再改進，但可看出有助於學生反思。

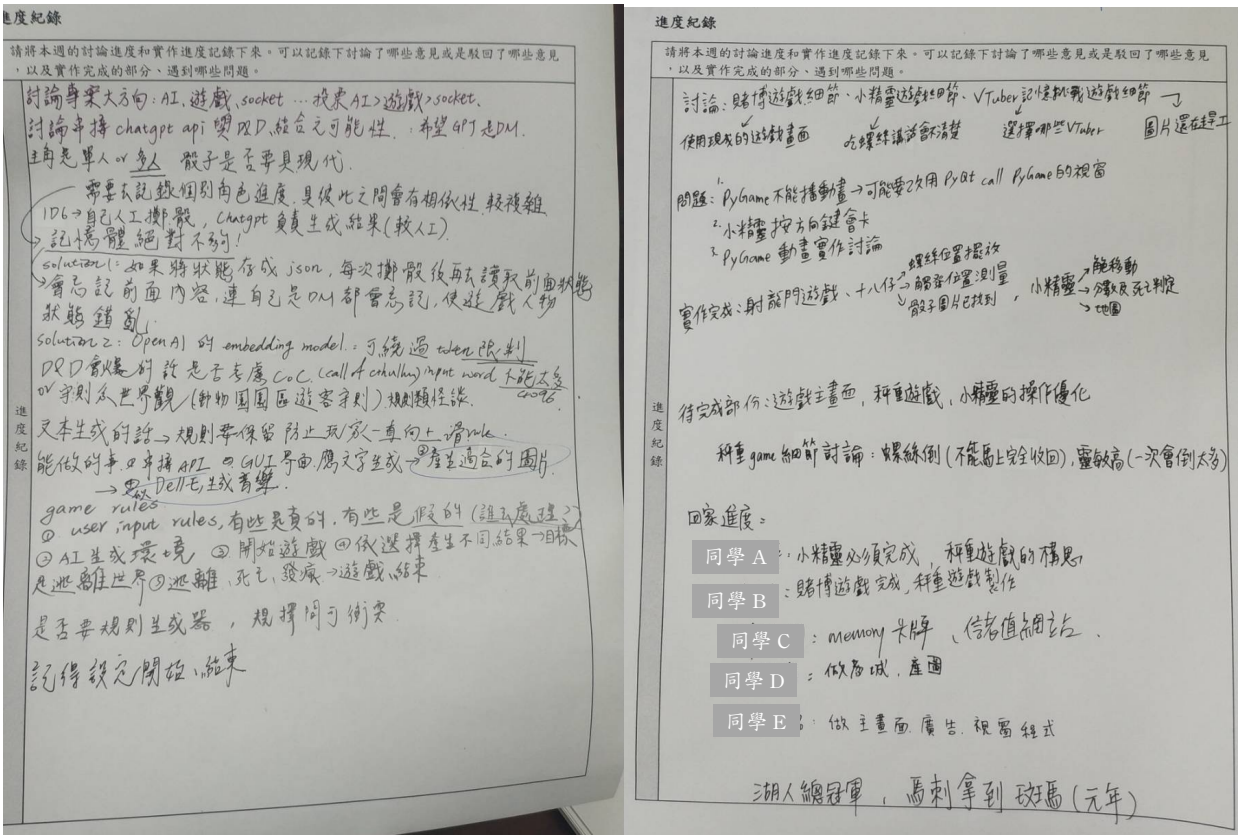


圖 3 兩組學生填寫期末專題進度紀錄的學習單成果

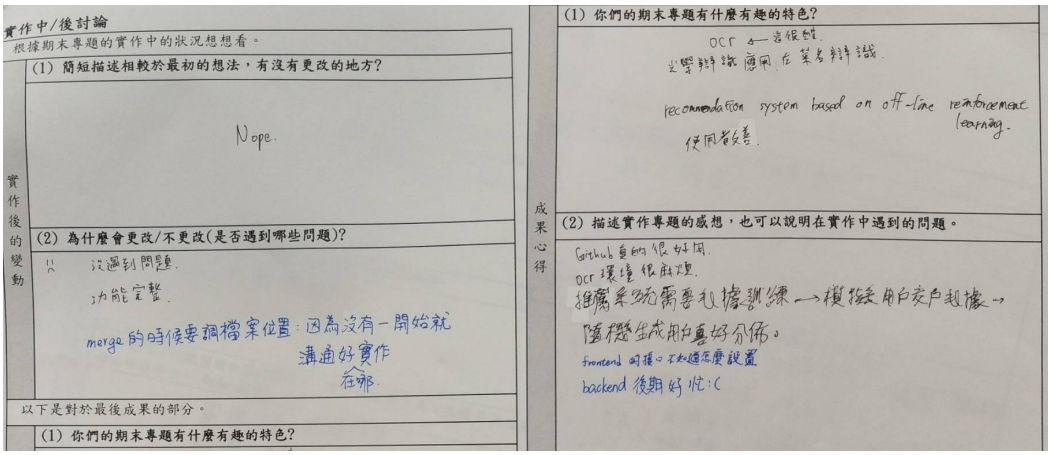


圖 4 某組學生填寫期末專題的實作中/後討論之學習單成果

6. 教學暨研究成果 Teaching and Research Outcomes

(1) 教學過程與成果

在教學的過程當中，原本擔憂資工系的大學生是否會對於填寫學習單的形式有所排斥，但可能由於多位助教事先討論並設計出來的學習單效果不錯，加上現場有耐心的助教帶領之下，學生們大都能有所討論並完整填寫思考模組學習單。我們觀察到在分組後的第一週時，學生的討論還不是很熱烈，但由於助教要求群體程式設計與填寫學習單的情況下，不會各自做其他的事，而是有目標的進行簡短的討論；在第二週第三週之後的隨堂練習裡，則可以明顯地看出學生的課堂討論逐漸變得熱烈了，圖 5 為學生們分組討論狀況的照片。這裡也感謝學校對於觀課助教的支援，可以在原本的課程助教之外又多安排一位助教協助觀察學生狀況並拍照紀錄。

圖 5 學生們分組討論狀況的照片

在具體的程式設計能力測驗表現上，相較於去年有小幅度的進步。如表 1，可以看出今年學生的平均前測分數和去年比分數相當接近（滿分為 100，去年 62.0，今年 62.3），而後測平均分數則有小幅的進度（去年 71.9，今年 75.7）。這裡的前測有效樣本數 59 人，授課前進行且不計入學期成績，後測有效樣本數 51 人，用於異質分組；去年有效樣本數則為 25 人（去年未全面要求學生填寫）。

表 1 去年的程式設計能力測驗平均分數有小幅的進步

	前測	後測
111學年度	62.3	75.7
110學年度	62.0	71.9

另一方面，我們也從學期成績分布來看學生們的學習成效，藉此觀察本研究是否有助於學生在程式設計上的學習，結果如圖 6，看得出略有提升。雖然本課程的定位原本就不是在於艱深的程式設計技巧，而是在於基礎程式設計能力的培養，也因此去年學生的成績也偏高，但今年可以看出 90 分以上的學生更多了，分布上是將 80-89 分的學生們提升至 90-99 分的級距。關於這些教學研究成果，我們正在撰寫論文準備發表。

學期: 1102 , 課號: CE3065

◎修課人數: 58

◎全班總平均: 90.1724137931

◎標準差: 5.00574382728

◎最高分: 99

◎最低分: 71

分數級距	人數
100分	0
90-99分	42
80-89分	14
70-79分	2
60-69分	0
50-59分	0
40-49分	0
30-39分	0
20-29分	0
10-19分	0
0-9分	0

學期: 1112 , 課號: CE3065

◎修課人數: 60

◎全班總平均: 90.8333333333

◎標準差: 9.41899751164

◎最高分: 97

◎最低分: 28

分數級距	人數
100分	0
90-99分	52
80-89分	5
70-79分	2
60-69分	0
50-59分	0
40-49分	0
30-39分	0
20-29分	1
10-19分	0
0-9分	0

圖 6 學期成績分布略有提升

(2) 教師教學反思

在教學上申請人發現有幾個狀況和預想的小有差異。第一個是原本猜想學生們一開始填寫我們設計的思考模組學習單會有意見，或是不願意進行群體程式設計來實作，但其實接受度都算高，並不會有原先猜想需要多花時間導入。

第二個是在課程前的準備上，由於設計思考模組學習單既需要有程式設計的底子，又要求具備剖析自己思考過程能力的助教，因此儘管申請人原本就有去年的範例程式碼、練習題、解答程式碼，但一開始也須要花不少時間培訓助教對此設計出思考模組學習單，並反覆審視設計出來的思考模組學習單是否有效，這部分應該是值得未來更進一步思考如何帶助教努力的地方。

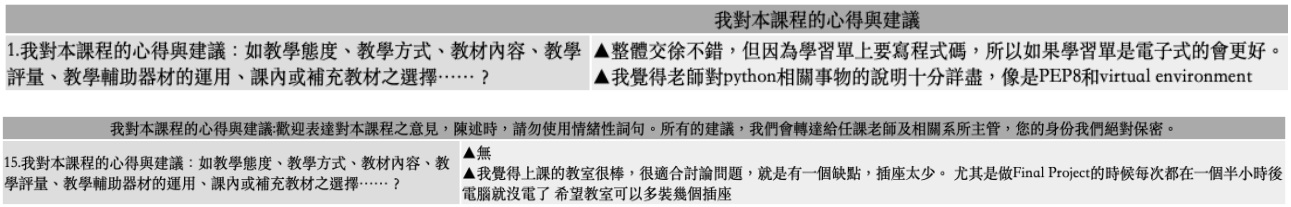
第三個其實是分組的狀況，原本預期是不是有很多學生不願意異質分組，但可能是同系的緣故，加上期初加選時申請人就再三強調是異質性分組，因此在進行上沒有什麼困難，討論也都相當踴躍。不過全部 12 組當中，仍然有 2 組的學生私下反映問題，其中 1 組的問題較大，其原因都是有組員擺明了對課程不願意花心思，不配合其他組員的專題實作；而當這樣的學生在一組裡多於兩位時，就會出現較大的困難。

(3) 學生學習回饋

在期中教學回饋與期末教學評量裡，觀察到學生的反應都相當正面，期末教學評量的分數為 4.48（5 點量表）。不過今年由於學校的期末教學評量截止日

較早，許多學生忙於期末專題實作，助教也忘了提醒學生填寫，因此填答率僅有 26.67%（修課人數 60 人，實際填卡數 16 份）。

可以改進的部分，也有學生提出具體建議，像是希望學習單電子化的建議，以及教室裡插座不夠用的狀況，如圖 7。教室的部分是硬體設備，由於我們選擇的教室是可以多人圍繞在一張桌子，方便學生們討論，未來可以多準備電源延長線，相對容易解決。而學習單的電子化比較需要思考，因為固然電子化方便學生填寫，尤其是程式碼部分，但相反地如果直接在電腦或手機上輸入，可能不如紙本形式方便多人討論？這點需要再多加考慮，而為了避免學生手寫很多的程式碼，可能在學習單的設計上改進問答方式會有所改善。



我對本課程的心得與建議	
1.我對本課程的心得與建議：如教學態度、教學方式、教材內容、教學評量、教學輔助器材的運用、課內或補充教材之選擇……？	▲整體交徐不錯，但因為學習單上要寫程式碼，所以如果學習單是電子式的會更好。 ▲我覺得老師對python相關事物的說明十分詳盡，像是PEP8和virtual environment

我對本課程的心得與建議:歡迎表達對本課程之意見，陳述時，請勿使用情緒性詞句。所有的建議，我們會轉達給任課老師及相關系所主管，您的身份我們絕對保密。	
15.我對本課程的心得與建議：如教學態度、教學方式、教材內容、教學評量、教學輔助器材的運用、課內或補充教材之選擇……？	▲無 ▲我覺得上課的教室很棒，很適合討論問題，就是有一個缺點，插座太少。 尤其是做Final Project的時候每次都在一個半小時後電腦就沒電了 希望教室可以多裝幾個插座

圖 7 學生們在期中教學回饋與期末教學評量的心得建議

7. 建議與省思 Recommendations and Reflections

在本研究計畫的執行中，我們觀察到思考模組表單確實能協助學生思考與討論。具體來說，

1) 討論的狀況比以往來得有目標、而且更加熱烈
2) 學生們確實能就由提問與解釋來釐清彼此的思考過程
3) 在計畫的執行上，未來需努力有更多可量化的研究評估指標

由於本研究計畫目標為藉由可視化研究方法來釐清學生的程式設計思考過程，只能藉由申請人與助教的觀察來確認成效，在學生們做練習題時下去看討論狀況，以及從收回的學習單來了解，但這不容易量化。如果要以數字來看，目前僅能從程式設計能力測驗與學期成績來看學習成效，這點是申請人認為未來也許可以再多思考並設計其他方法的省思部分。

二. 參考文獻 References

Davies, S. P. (1993). Models and theories of programming strategy. *International Journal of Man-Machine Studies*, *39*(2), 237-267.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1993). Design patterns: Abstraction and reuse of object-oriented design. European Conference on Object-Oriented Programming,

Hsu, T.-C., Chang, S.-C., & Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, *126*, 296-310.

Hwang, W.-Y., Shadiev, R., Wang, C.-Y., & Huang, Z.-H. (2012). A pilot study of cooperative programming learning behavior and its relationship with students' learning performance. *Computers & Education*, *58*(4), 1267-1281. https://doi.org/https://doi.org/10.1016/j.compedu.2011.12.009

Knuth, D. E. (1970). Von Neumann's first computer program. *Acm computing surveys (csur)*, *2*(4), 247-260.
Neumann, J. v. (1993). First draft of a report on the EDVAC. *IEEE Annals of the History of Computing*, *15*(4), 27-75. https://doi.org/10.1109/85.238389
Perkins, D., & Martin, F. (1986). Fragile knowledge and neglected strategies in novice programmers. at Empirical Studies of Programmers, 1st Workshop, Washington, DC,
Perkins, D., & Ritchhart, R. (2003). Making thinking visible. *New horizons for learning*, *8*.
Polya, G. (1945). *How to solve it; a new aspect of mathematical method*. Princeton University Press.
Ritchhart, R., Church, M., & Morrison, K. (2011). *Making thinking visible: How to promote engagement, understanding, and independence for all learners*. John Wiley & Sons.
Ritchhart, R., & Perkins, D. (2008). Making thinking visible. *Educational Leadership*, *65*, 57-61.
Shiraishi, M., Washizaki, H., Fukazawa, Y., & Yoder, J. (2019, 15-19 Jul 2019). Mob Programming: A Systematic Literature Review. 2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC),
Slavin, R. E. (1980). Cooperative Learning. *Review of Educational Research*, *50*(2), 315-342. https://doi.org/10.3102/00346543050002315
Turing, A. M. (1937). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London mathematical society*, *s2-42*(1), 230-265. https://doi.org/https://doi.org/10.1112/plms/s2-42.1.230
Weinberg, G. M. (1971). *The psychology of computer programming*. Van Nostrand Reinhold New York.
Winslow, L. E. (1996). Programming pedagogy—a psychological overview. *Acm sigcse bulletin*, *28*(3), 17-22.
Zuill, W., & Meadows, K. (2016). Mob programming: A whole team approach. Agile 2014 Conference, Orlando, Florida,