



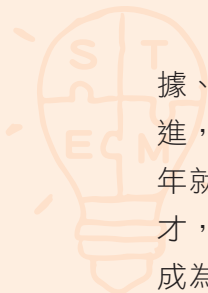
Scratch 讓學生學習想像 · 程式 · 分享

以Scratch開啟孩子的 運算思維能力

文／呂聰賢

與程式設計初相遇，適合淺顯易懂的圖像化程式設計語言。Scratch 是為協助兒童學習創意思考、協同合作及系統性思考所發展出來的一個程式設計工具。被廣泛的與各種感應板、感測器結合，如 Arduino、mBot 機器人、Tello 空拍機等，非常容易激發學生高度的學習興趣。

看見未來趨勢



資訊、網路蓬勃發展的世代，大數據、雲端服務、IoT 物聯網、App 等不斷躍進，美國境內跟程式有關的工作，每隔數年就以倍數成長，殷切需求的大量軟體人才，和經濟發展層面數位技能供需落差，成為各國皆無法忽視的課題。誠然不是每

一個人都要成為工程師，但身處一個資訊科學與物聯網全力運轉的世界，了解這些改變我們生活方式的工具和科技是一件重要的事。包括前美國總統歐巴馬、前英國首相卡麥隆、新加坡總理李顯龍等各國政要，以及微軟的比爾蓋茲、Apple 的賈伯斯和臉書創辦人祖克伯等科技巨擘，無不大聲疾呼程式教育的重要。近年來各國也紛

紛調整教育政策，投入更多的資金與人力資源。英國在 2014 年將程式教育列為 5 ~ 16 歲學童的必修課程，而美國在 2016 年初宣布將投入 40 億美元資金，讓電腦科學深入美國的基礎教育，更將程式教育列入通識課程。

此刻，臺灣也正在進行教育的重要變革，12 年國教課綱中資訊科技課程以運算思維為主軸，透過電腦科學相關知能的學習，培養邏輯思考、系統化思考等運算思維，並藉由資訊科技之設計與實作，增進運算思維的應用能力、解決問題能力、團隊合作以及創新思考。教育部也在 2017 年辦理了高中和國中小的運算思維種子教師培訓，並進行 100 多場的教育推廣，希望教師們都能了解並具備推動運算思維的能力。

認識運算思維

那什麼是運算思維 (computational thinking) 呢？第一個提出運算思維的人是卡內基美隆大學的 Jeannette M. Wing (華裔，周以真)，她對運算思維的定義是：「運算思維是利用電腦科學的基本概念進行問題解決、系統設計與人類行為理解的思維模式」，她認為在基礎語言能力中應該加入電腦運算的因素，在讀、寫和算數之外，還需要加上電腦運算的概念：「電腦運算思維的技巧，並不只是電腦科學家的專利，而是每個人都應該具備的能力及素養。」(見圖 1)

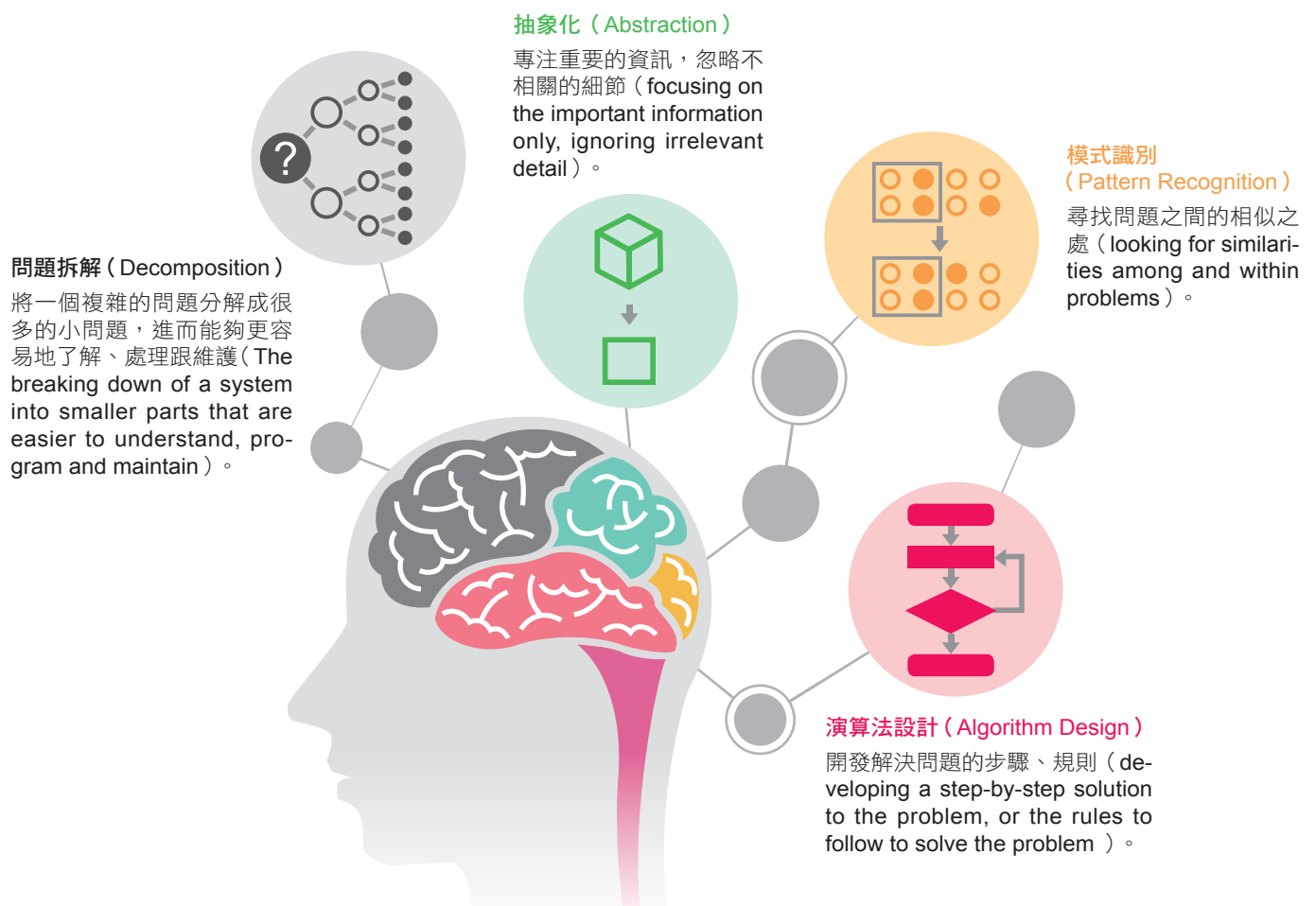


圖 1. Google 官方網站上對於運算思維的四個主要面向 (圖片來源：Google 官網)



簡單的說，運算思維是「觀察分解、歸納辨識、重點摘要、解決方案」；目的在培養學生解決問題的能力；教學重點在「想、觀察」。

資訊科學是培養運算思維能力的重要途徑，但不是只有程式設計或資訊科技課程才能做到，其他領域也可以透過不同方式來進行。在教育部的運算思維推動計畫網站中（<http://comphinking.csie.ntnu.edu.tw/>）有各領域教學示例，包括語文、數

學、自然、社會、藝文等，提供大家參考。

運算思維與程式教育

學習程式設計的工具軟體與活動很多，淺顯易懂的視覺化程式設計語言適合與程式的初相遇，像是 Code.org、Kodu、Scratch、Blockly 等，簡單介紹如表 1。

教育部的運算思維推動計畫中，對各學習階段資訊科技課程的規畫建議如表 2。

表 1 淺顯易懂的視覺化程式設計語言舉例

Code.org	Code.org 是一個數億人力投入的公益網站，它在 2013 年發起的「Hour of Code」一小時玩程式活動，已有超過 180 個國家、數以百萬的學生加入。與知名遊戲的合作，「憤怒鳥」、「植物大戰殭屍」、「星際大戰」，乃至於最新的「Minecraft（當個創世神）」等知名遊戲角色元素，讓小朋友在遊戲中快樂、循序地學習程式設計。
Scratch	Scratch 是 2007 年美國麻省理工學院媒體實驗室終身幼兒園團隊（Lifelong Kindergarten group at the MIT Media Lab）的一個計畫，為協助兒童學習創意思考、協同合作及系統性思考所發展出來的一個程式設計工具。
Kodu	2013 年微軟公司發起「WeSpeakCode 一起玩程式」活動，開發了 Kodu 這套免費的視覺化物件導向程式設計語言。即使沒有程式設計的基礎，利用直覺的圖示，小朋友也能很容易地把創意發揮出來，建構屬於自己的 3D 虛擬世界，設計出自己的 3D 遊戲，體會程式設計的趣味和成就感。
Blockly	由 Google 開發的「Blockly Games」，2014 年底上線，是一個給兒童或電腦初學者的玩遊戲學程式線上課程。藉由圖形化的介面及一個個具有學習主題的關卡，讓闖關者了解程式語言的邏輯。
App Inventor	Google 開發的應用軟體，讓不懂程式語言的人也能創作 Android 手機的程式 APP，堆積木組程式，並能控制手機硬體功能及整合 Google 各種服務，設計專屬自己的手機 APP。

表 2 教育部的運算思維推動計畫對各學習階段資訊科技課程的規畫建議

國小階段	<ul style="list-style-type: none"> • 資訊科技之體驗與應用 • 目標導向運算思維研習 • Code.org、Scratch、Kodu、Blockly 等
國中階段	<ul style="list-style-type: none"> • 利用運算思維與資訊科技解決問題 • 問題導向運算思維學習 • S4A、App Inventor 等
高中階段	<ul style="list-style-type: none"> • 了解運算思維原理，並能進一步整合應用 • 創作導向運算思維實踐 • Java/C/C++/Python、Arduino、App Inventor 等

Scratch的優點

與程式設計初相遇，適合淺顯易懂的圖像化程式設計語言。**Scratch** 將程式設計指令轉化為簡單的積木方塊，不需要一行一行寫程式，只要拖曳組合這些指令積木，就能完成程式設計，創作互動式故事、遊戲、動畫和音樂等作品。**Scratch** 不只簡單、有趣，而且程式概念完整。更棒的是，**Scratch** 被廣泛的與各種感應板、感測器結合，如 **Arduino**、**mBot** 機器人、**Tello** 空拍機等，非常容易激發學生高度的學習興趣。學習的動機與興趣，絕對是影響學習成效極重要的關鍵。



圖 2. Scratch 能與各種感應板或感測器結合

如何以Scratch開啟孩子的運算思維能力

美國哈佛大學在其《運用 Scratch 的運算思維》(Computational Thinking With Scratch) 研究中指出：學生在建立運算思維概念的過程包含以下「**運算概念**」(Concepts)、「**運算實作**」(Practices) 和「**建立觀點**」(Perspectives) 三個主要面向。

一、運算概念 (Concepts)

學習程式設計的七個基本概念是：序列 (sequence)、迴圈 (loops)、平行 (parallelism)、事件 (events)、條件 (conditionals)、運算子 (operators)、資料 (data) (見表 3)，這些概念不僅在 **Scratch** 的學習非常有用，在學習其他程式語言時也一樣很有幫助。我們可以透過明確的主題範例，讓學生循序學習這些概念。

表 3 學習程式設計的七個基本概念

項次	概念	說明與範例	程式碼圖示
1	序列 (sequence)	說明：確定任務的一系列步驟 範例：先向右走，再向左走，然後說「大家好」。	

(續下頁)



項次	概念	說明與範例	程式碼圖示
2	迴圈 (loops)	<p>說明：重複執行相同的動作</p> <p>範例：運用迴圈畫出三個藍色的正方形。</p>	
3	平行 (parallelism)	<p>說明：在同一時間讓多個事件一起發生</p> <p>範例：同時執行三件事</p> <ol style="list-style-type: none"> 1. 旋轉一圈。 2. 每隔 1 秒換造型。 3. 播放 meow 音效 2 次。 	
4	事件 (events)	<p>說明：一個事件引發另一個事件的發生</p> <p>範例：四種觸發事件</p> <ul style="list-style-type: none"> • 點選綠旗時，重複播放音樂 • 按鍵盤右鍵時，移動 10 點 • 收到訊息 change 時，開始改變顏色 • 當角色被點擊時，說出「好哦」2 秒 	
5	條件 (conditionals)	<p>說明：根據條件做出決定</p> <p>範例：如果碰到紅色時就播放 meow 音樂，否則音效就停止。</p>	
6	運算子 (operators)	<p>說明：數學運算與邏輯表達</p> <p>範例：數學基本計算如加、減、乘、除，邏輯判斷大於、小於、等於。例如貓咪每 1 秒會隨機在左右變換位置。</p>	

(續下頁)

項次	概念	說明與範例	程式碼圖示
7	資料 (data)	說明：儲存、讀取和更新資料 範例：貓咪預設生命值為 10，如果碰到蝙蝠時就減少生命值 1，當生命值小於 1 時遊戲就結束。	

二、運算實作 (Practices)

透過實作可以讓學生更好地理解運算思維以及程式的概念，通過漸進式、任務式的情境問題來引導，可以提高學生的學習興趣，也更容易理解及激發創造力。下列的四個做法，有助於專案的發展和完成。

- 1. 實驗與迭代 (experimenting and iterating) :** “Developing a little bit, then trying it out, then developing more.” 我們在面對問題時，往往很難一次解決所有的需求，所以開發的過程通常會包含多個開發週期。這是一個漸進的過程，先開發一部分，實驗測試看看，然後進入下一個週期做更進一步的開發。由少而多，由簡單到複雜，有系統地一步步去完成。
- 2. 測試與除錯 (testing and debugging) :** “Making sure things work - and finding and solving problems when they arise.” 我們測試程式，確認產品或功能如同設計者所預期，當出現問題時，我們就依循以下的步驟來除錯。

- ★ 進行測試
- ★ 尋找問題所在
- ★ 檢查程式指令是否正確
- ★ 檢查程式邏輯是否正確

★ 修正指令

★ 再次測試

- 3. 重複使用與混合 (reusing and remixing) :** “Making something by building on existing projects or ideas.” 專案建立的過程可以利用別人已經建構好的功能或專案，在此基礎上加強或是加入其他想法，改造成新的專案。觀摩學習他人作品是躍進的好方法（如圖 3，頁 10），站在「巨人」的肩膀上可以讓你看得更遠。在 Scratch 的網頁上有許多專案，可以幫助學生發想。
- 4. 抽象簡化和模組化 (abstracting and modularizing) :** “Exploring connections between the whole and the parts.” 分析問題時，學習將任務抽象簡化和模組化，亦即分解為多個小任務並去探索以及理解每個部分之間的關聯。抽象簡化是將許多小問題分析歸納成一個整體或是更高層次的問題，將複雜的問題簡化成簡單的問題，讓人可更好地去思考各個層次的問題。模組化可以讓程式方便閱讀與理解，在測試除錯上也更容易。畫流程圖可以協助學生更理解各個部分的連結關係。

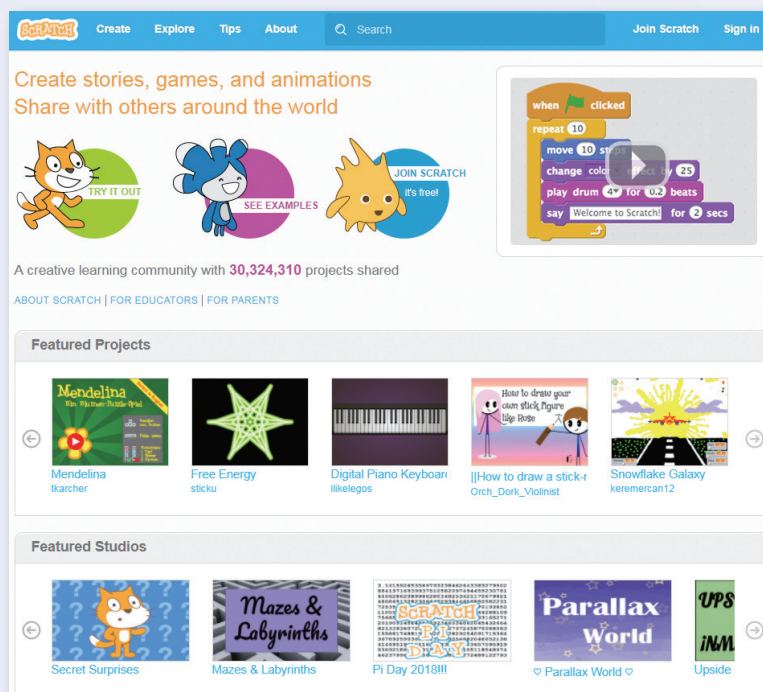


圖 3. Scratch 官方網站的範例作品（圖片來源：Scratch 官方網站）

三、建立觀點（Perspectives）

老師可以規劃小組任務，讓學生以團隊合作的方式完成專案，並安排創作發表的機會和舞台，以持續激發學生創作的動力和熱情。

1. **表達（expressing）**：“realizing that computation is a medium of creation, ‘I can create.’” 讓學生理解電腦運算也是一種創作的媒介，我們可以透過電腦運算來表達自己的想法和創意。
2. **連結（connecting）**：“recognizing the power of creating with and for others, ‘I can do different things when I have access to others.’” 與他人合作，連結眾人的力量，在溝通、討論的互相學習中，可以激發更多的想法和創意，展現比一個人更多也更大的力量。

3. **提問及探索（questioning）**：“feeling empowered to ask questions about the world, I can (use computation to) ask questions to make sense of (computational things in) the world.” 鼓勵學生勇敢地去對任何事物提出問題，「用電腦運算的模式」去思考、提出、理解並分析問題，然後「透過電腦運算」去建立自己對電腦和世界的理解。

結語

我們的生活中，程式無所不在。「軟體吃掉全世界」是世界趨勢，也是現在進行式，不管未來你是否想要成為一位工程師，在不久的未來，程式設計將會被視為基本素養。



學習程式設計可以教你如何思考，培養運算思維、發展解決問題的能力，幫助我們面對因為新科技而不斷改變的世界。Fun Coding，大家一起來！

後記

在即將施行的 12 年國教課綱中，國小階段是沒有資訊課教學時數的。換句話說，國小不上資訊課，到了國中直接學程式設計，這真是一件非常令人難以想像而且擔憂的事。

筆者在國小任教，年復一年看著一群群滿懷興奮走進電腦教室的孩子們，即使在 3C 當道的今日，學校的電腦課依然是許多小朋友真正開始學習資訊科技的起點，期待所有學子能自動具備這些基礎能力是不切實際的想法。因此筆者認為，國小應該要有資訊科技課程的時數，資訊科技的基本素養，包括作業系統、打字、網路運用、電子郵件、文書處理、簡報製作等，也應該在國小資訊課程裡被教導，否則巨大的數位落差將無可避免地再次重擊我們的教育現場。

參考網址

1. 教育部運算思維推動計畫：
<http://compthinking.csie.ntnu.edu.tw/>
2. Computational Thinking With Scratch:
<http://scratched.gse.harvard.edu/ct/defining.html>
3. Google Computational Thinking for Educators:
<https://computationalthinkingcourse.withgoogle.com>
4. Google for Education:
<https://edu.google.com/resources/programs/exploring-computational-thinking>

呂聰賢
新北市昌福國民小學資訊組長