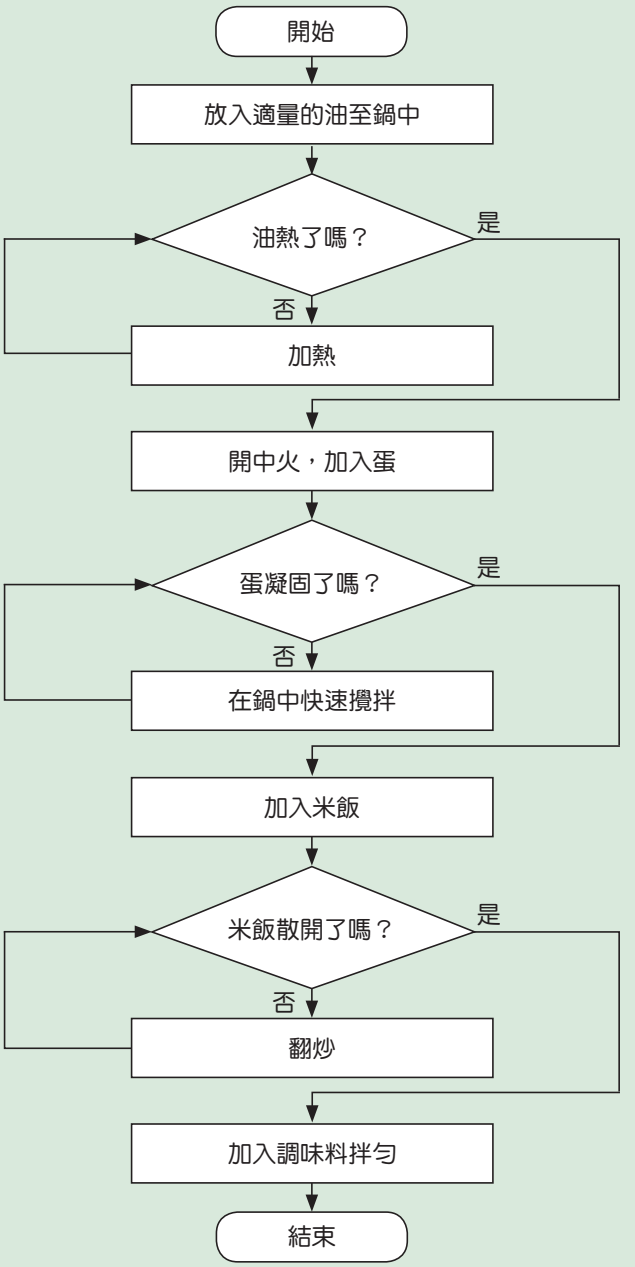
2-1 認識演算法與程式語言

在資訊科技領域，演算法是一種解決問題的方法，程式語言則是實踐演算法的工具，兩者相輔相成，在此一併討論。

在日常生活中，食譜也類似一種演算法，例如：我們可以把蛋炒飯的步驟畫成流程圖來表示（圖 2-1）。但食譜和電腦的演算法不盡相同，最主要的差別在於食譜不夠精準，會因不同人的解讀而有不同的結果，而電腦的演算法必須表示得很精確，不容許有模糊空間。



▲圖 2-1 製作蛋炒飯的流程圖

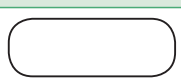
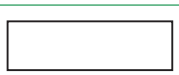
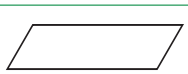
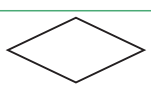
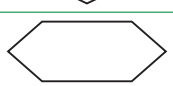
一般來說，食譜雖然把製作步驟描述得很詳細，但是不同人來解讀與實作還是有很大的差異，即使是同一個人在不同時間實作，其結果也會有不相同的品質，這個現象尤其容易出現在新手身上。以上述的蛋炒飯為例，「放入適量的油至鍋中」，而這個適量的敘述就不明確，再者「開中火」，到底是指多大的火候？至於「加入米飯」，使用剛煮好的熱米飯或隔夜冷飯，其結果又會有不同的口感。

2-1-1 演算法的基本概念

我們生活中經常遇到待解決的問題，簡單的問題，憑直覺就可以解決，例如：數字相加，對於熟悉算術的人，根本不是什麼問題；但對沒有學過加法運算的人，就會是一個問題。廣義的說，解決問題的方法就是一種演算法，我們可以把解決這些日常生活問題的方法與過程寫成**步驟**，再依照這個步驟去**執行**。

為了清晰的表示演算法，可把解決問題的步驟整理成符號來表示。美國國家標準學會（American National Standards Institute，簡稱 ANSI）於 1970 年制定標準的流程圖（Flow chart）符號，以方便判讀與相互交流解決問題的步驟，常用的流程圖符號如下表。

流程圖的符號與功能說明

符　號	意　義	說　明
	開始 / 結束	流程圖開始或結束
	處理	處理一項工作
	流程方向	流程進行的方向
	輸入 / 輸出	進行資料輸入或輸出的工作
	決策	依條件比較結果進行不同的處理
	迴圈	迴圈變數初值與終值的描述
	連接	流程的連接點

接下來就以實際的例子，將解決問題的方法與過程，以流程圖來表示。以求任意數的所有因數為問題解決的實例，如果選擇使用窮舉法來解決此問題，就是從 1 開始，針對每一個整數依序進行測試，以找出符合條件的數字。當我們面對一個新問題，又不知道如何下手時，可以考慮先採用窮舉法來求解，問題解決之後，再進一步思考如何改進解決方法。

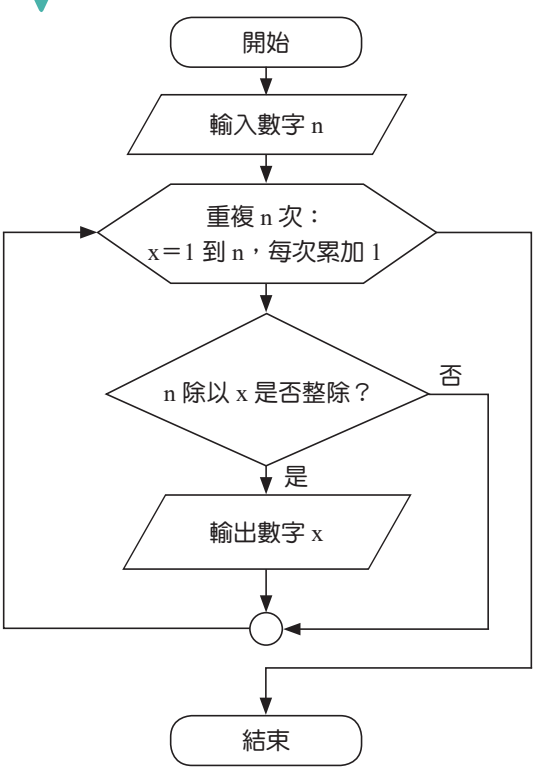
問題分析

在這個流程圖中，假設使用者輸入的數字 n＝20，接著進入計次式迴圈，迴圈內的敘述會重複執行 20 次。在這個例子中，還有一個迴圈變數 x：

執行迴圈	處理	判斷	輸出
第一次	x＝1	20 除以 1 整除	1
第二次	x＝2	20 除以 2 整除	2
第三次	x＝3	20 除以 3 沒有整除	沒有執行任何敘述
第四次	x＝4	20 除以 4 整除	4
⋮			
第二十次	x＝20	20 除以 20 整除	20

這個例子，很明確的定義了每一個步驟要執行的敘述，不會因為不同人解讀而有不同的結果。

畫流程圖

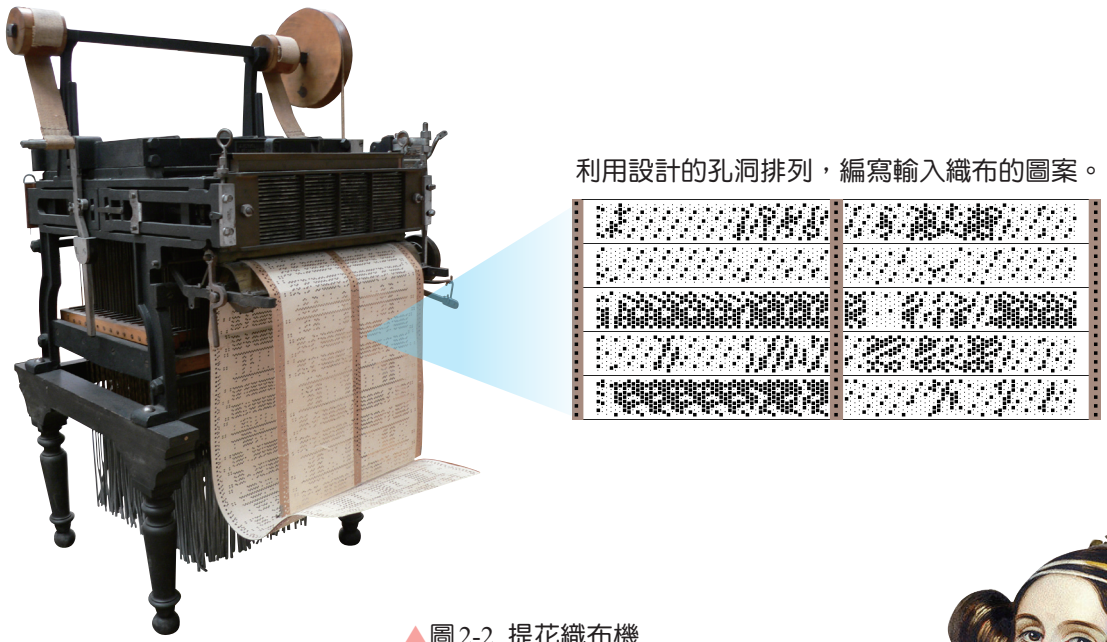


為了對演算法進行檢驗，必須將演算法轉換成電腦程式，因此最好的途徑是透過程式設計來實作演算法，要學習程式設計則需先選定一種程式語言。由於每個人的思考方式不同，解決問題的辦法也不盡相同，因此設計出來的演算法可能也會不同，但是最重要的，是要考慮演算法的正確性，也就是執行之後是否能產生正確的結果。

2-1-2 程式語言的基本概念

程式語言發展的歷史遠比電腦來得早，在電腦未發明之前，人們就開始產生編碼的概念。1801 年，法國人約瑟夫・瑪麗・雅卡爾（Joseph Marie Jacquard，1752～1834）運用木板打孔的方式，設計出可更改編織圖案的提花織布機（圖 2-2）。他是第一個以程式的概念設計機器的人，此機器展現了兩個程式設計的重要概念：

1. 複雜的設計也可以編譯成機器能了解的程式碼。
2. 依照程式碼指示，機器可不斷重複工作直到完成。



▲圖 2-2 提花織布機

而被認為是史上第一位電腦程式設計師是英國人愛達・勒芙蕾絲（Ada Lovelace，1815～1852，圖 2-3），她於 1843 年運用巴貝奇的分析機來計算伯努利數的方法，被認為是史上第一個電腦程式。直到 1940 年代，以電力驅動的現代化計算機出現後，也開啟了各種程式語言的發展。電腦只是一部機器，只要給予指令，它就會照指令執行工作，然後將結果輸出，而這些指令的組合就成為程式。

▶圖 2-3 愛達・勒芙蕾絲

2-1-3 程式語言的演變與發展

最早期的程式是使用 0 與 1 來編寫，稱為機器語言。由於電腦只看得懂 0 與 1，但是人在閱讀與編寫上都很不方便，因此電腦科學家發明了組合語言，使用一些簡單且有意義的文字來撰寫程式，例如：ADD（加）。利用組合語言寫成的程式，必須經由組譯程式的處理，才可在機器上執行。它執行的速度雖然相當快，但仍是相當複雜，必須對電腦硬體結構有相當了解的人才能撰寫，因此被歸類為低階語言（圖 2-4）。

低階語言

低階語言是一種以硬體為導向來描述指令的語言，例如：機器語言與組合語言。雖然一般人比較難看得懂，但是因為它可以直接與硬體的中央處理器溝通，因此執行效能較好。

組合語言

電腦的運算主要是靠中央處理器，而組合語言即是一種與中央處理器硬體結構密切相關的語言，若使用不同指令集的中央處理器，所對應的組合語言的語法也會不一樣。

組譯

機器語言

機器語言是由 0 與 1 兩種符號組成的代碼，它能夠直接被電腦理解並執行，是電腦硬體能夠直接識別的指令集合。但不同電腦硬體架構使用不同機器語言。

轉譯

▲圖 2-4 低階語言經由組譯程式轉換為機器語言。

由於組合語言的撰寫相當費力，而且容易出錯，在 1950 年代科學家又發明了較接近人類思維模式的高階語言（圖 2-5）。經過幾十年的演進，目前已知的程式語言種類很多，既有的程式語言仍有人在使用，但新的程式語言也不斷的出現。

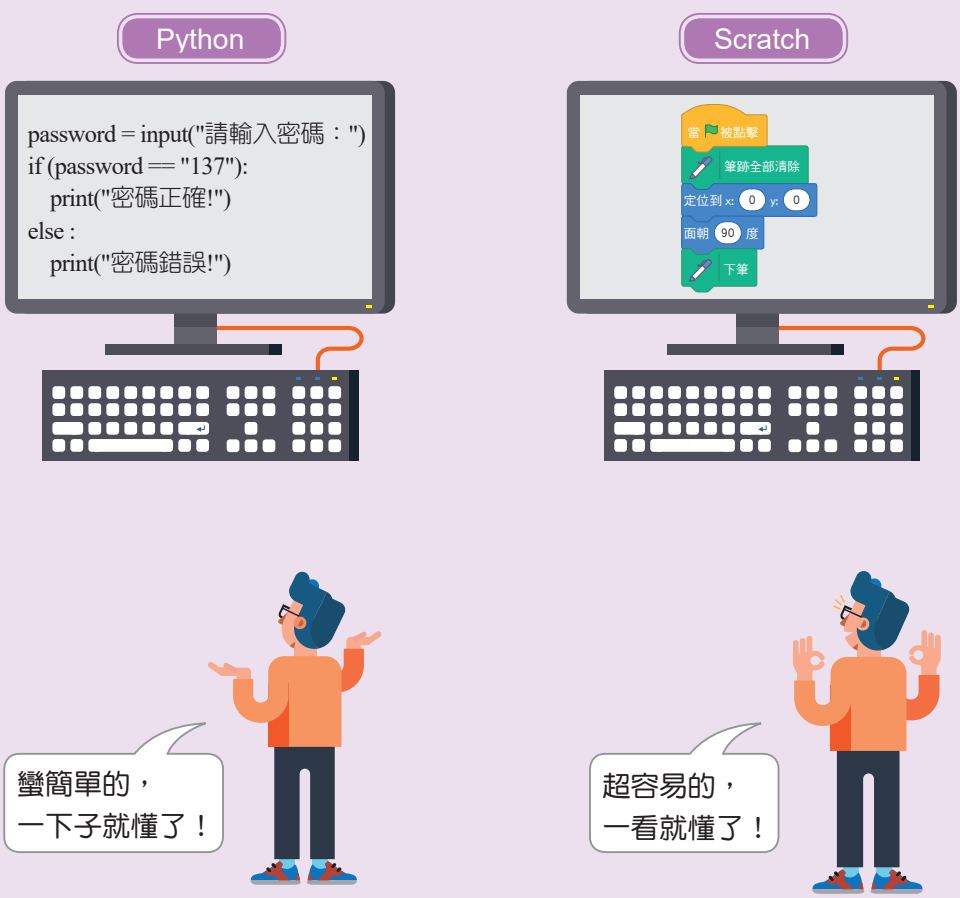
小知識

指令

在低階程式語言（如組合語言），一條要求電腦動作的指示，稱為指令（instruction），如 ADD（加）是一個指令，OR（或）也是一個指令。

高階語言

高階語言是一種語法接近人類語言的程式語言。以高階語言所設計的程式，必須先轉譯成機器語言，才能被電腦硬體執行。



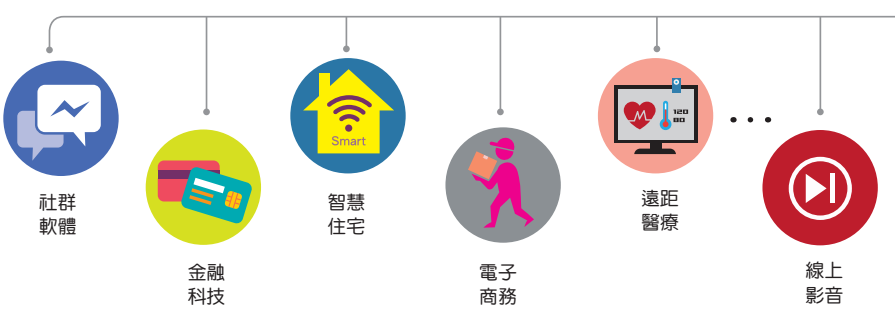
▲圖 2-5 高階語言轉譯為電腦能懂的機器語言。

2-1-4 程式語言的主要功能

綜合前面的介紹，人們為了要指揮電腦完成某項工作，就要寫許多指令，這些指令要按照一定邏輯順序排列，而電腦就依照這種順序接受指令而完成工作。總而言之，這些為完成某項工作而依其邏輯順序寫成的一連串指令，就稱為程式。程式要指揮電腦完成不同的工作，就有各種不同的功能，而主要的功能如下：

① 主要的功能就是要啟動（Booting）電腦並分配資源，指揮電腦運作。

② 其次是產生人與電腦溝通的介面，讓使用者可以透過介面來操作電腦硬體，因此人與電腦就可以產生互動。

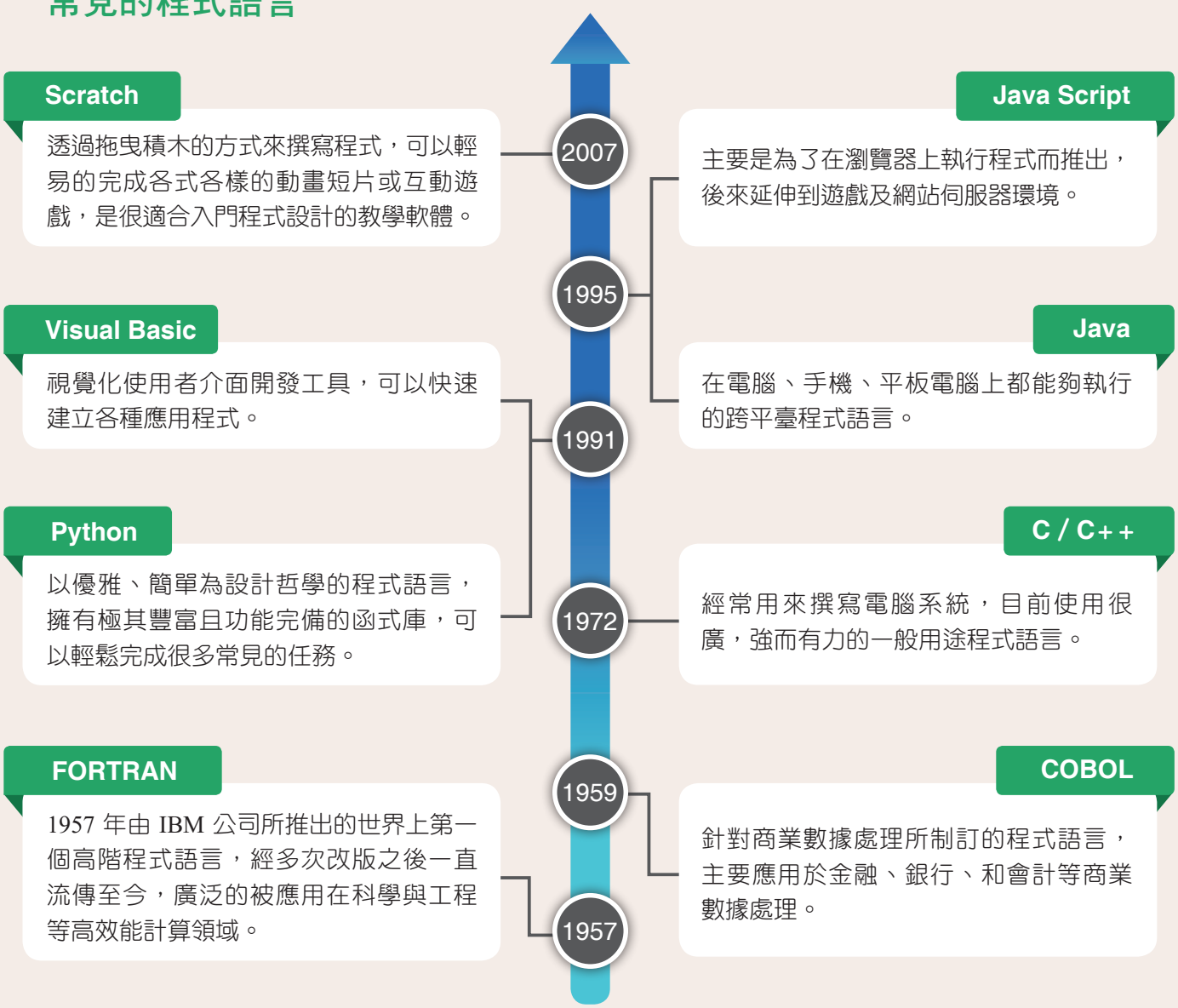


③ 把相關的電腦串連起來，特別在網路或是雲端時代，以各種硬體所建構起來的環境，需要靠各種程式發揮功能，將其串連在一起後，讓眾多使用者可以同時在線上互動與溝通。

2-1-5 程式語言的應用

從有程式語言至今，目前廣被使用的程式語言種類非常多，但因用途不同，功能也不一樣。大致上，可以分為一般用途及特殊用途。雖然不同程式語言的語法不一樣，但是基本邏輯則類似。一開始可以從一般用途的程式語言入門，建立了程式設計的基本概念後，再因應不同的專業需求，選用特殊用途的程式語言。在本課程中，我們將選用教學用途的 Scratch 語言來學程式設計。

常見的程式語言



2-2 Scratch 程式設計 – 基礎篇

搭配習作附件 7

Scratch 是由美國麻省理工學院媒體實驗室（MIT Media Lab）的終身幼稚園團隊（Lifelong Kindergarten Group）於 2007 年所發表的一套教學軟體。它提供了視覺化的圖形操作介面，學習者只要會用滑鼠拖曳積木，就能輕易的撰寫程式。

Scratch 是一套免費的自由軟體，目前已經被 150 多個國家翻譯成 40 多種語言，所以你可以使用中文來寫程式。Scratch 從 2013 年更新 2.0 版之後開始提供線上版本，只要連上網際網路，就可以用瀏覽器進行程式撰寫，不需要任何安裝程序，非常方便；2019 年，Scratch 改版 3.0，在平板電腦上也可以使用。而透過 Scratch 官網的雲端平臺，你可以將創意作品與全球分享。在本課程我們先了解 Scratch 的基本功能後，接著就要以簡單、有趣及實用的計算、繪圖、遊戲及模擬來學習 Scratch 程式設計。

2-2-1 Scratch 的官方網站

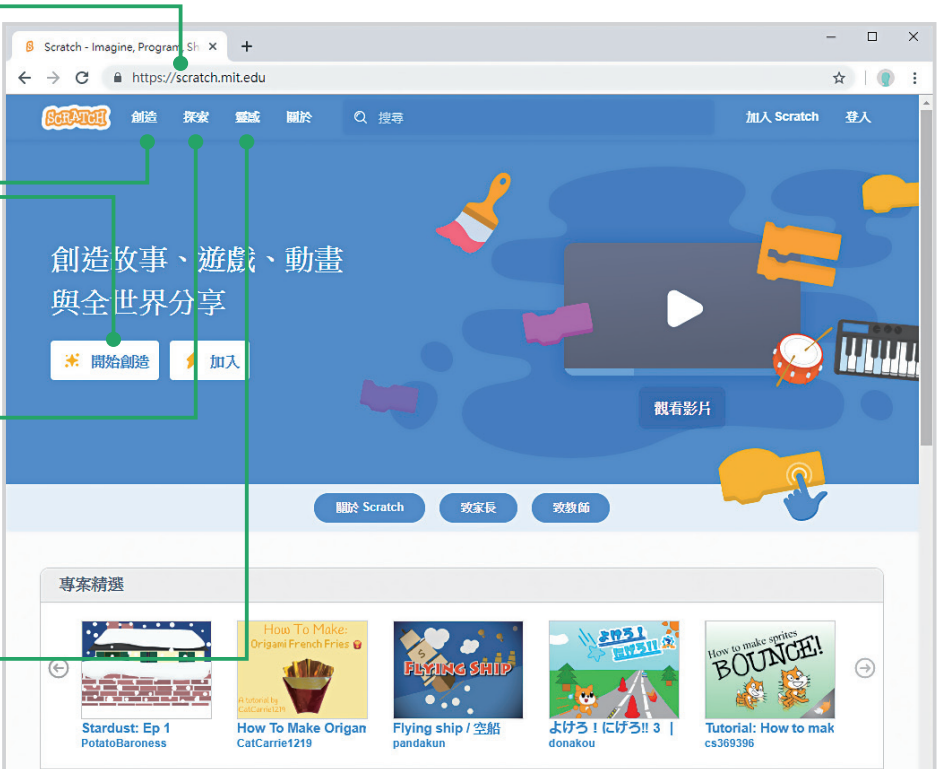
① Scratch 3.0 線上版

在瀏覽器的網址列輸入 https://scratch.mit.edu。

創造
開啟撰寫程式介面。

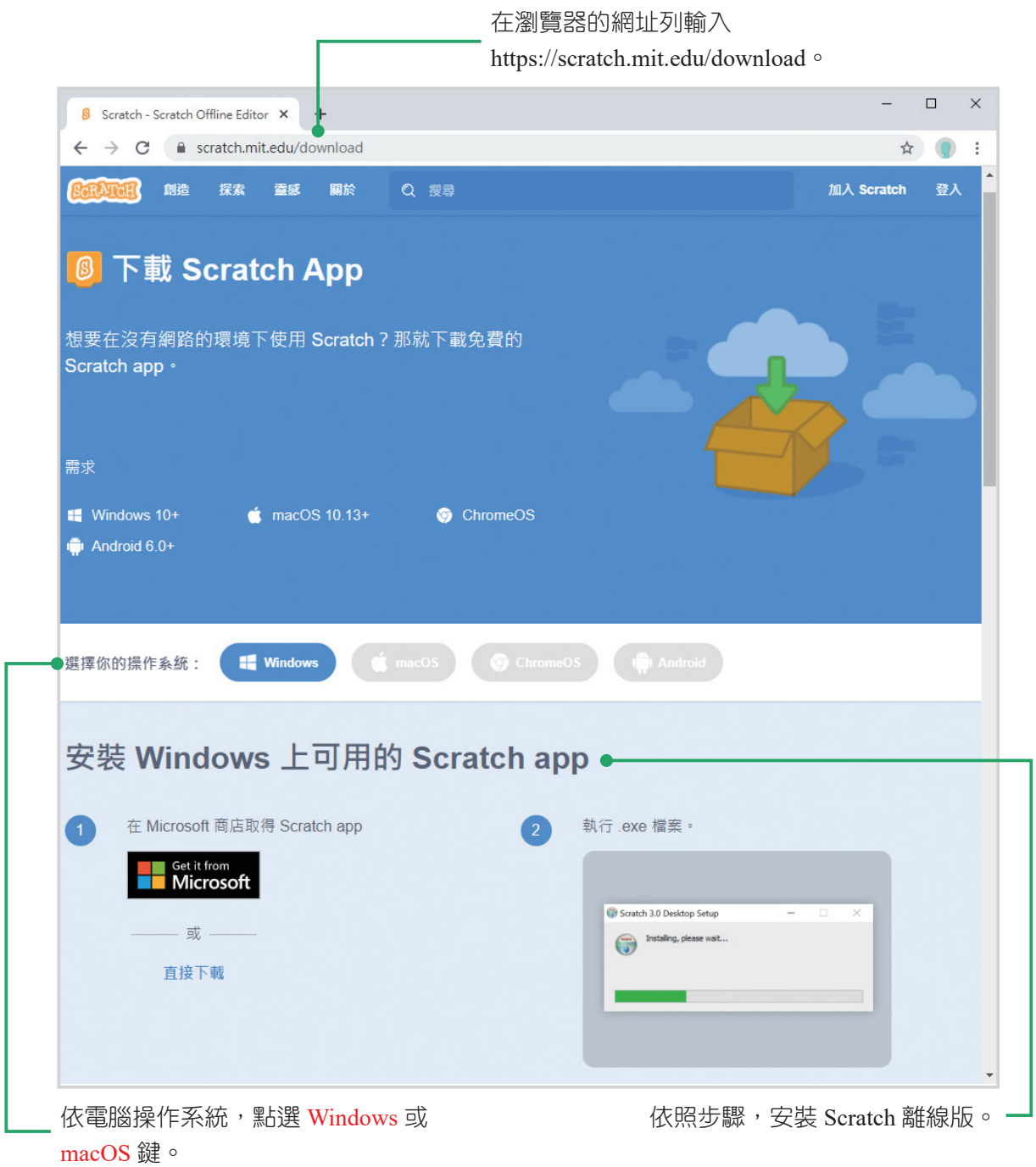
探索
專案及工作坊的開放平臺，使每個人皆能上傳、分享，並學習他人程式。

靈感
提供教程，並依不同主題分類，步驟化說明影片學習更容易。



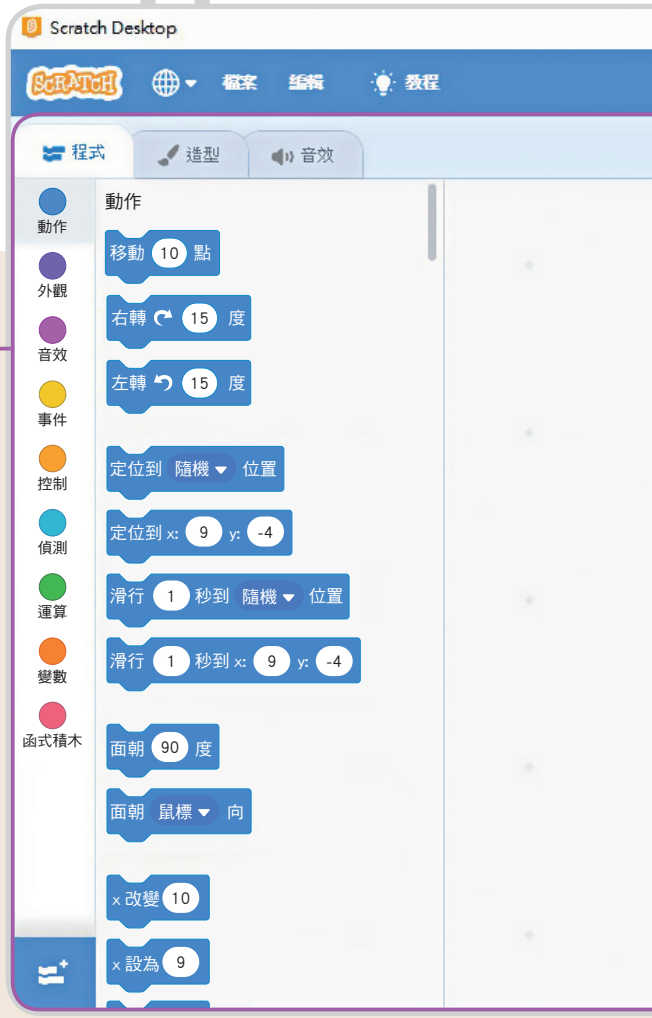
當然，我們也可以選擇從官網上直接下載並安裝離線版，完成後不必連上網路，就可以直接在自己的電腦上進行編輯。

② Scratch 3.0 離線版



2-2-2 Scratch 的操作介面介紹

啟動 Scratch 後，會看到如下圖的畫面，這個起始畫面劃分為三個區。



腳本區

此區可以定義角色的造型及聲音，並且組合各式積木以達成你想要的功能。

程式面板

按 程式 鍵顯示此面板，它提供 9 類共 100 多個積木讓你組裝，用來定義角色的行為，不同類別的積木使用不同顏色來區分。

造型面板

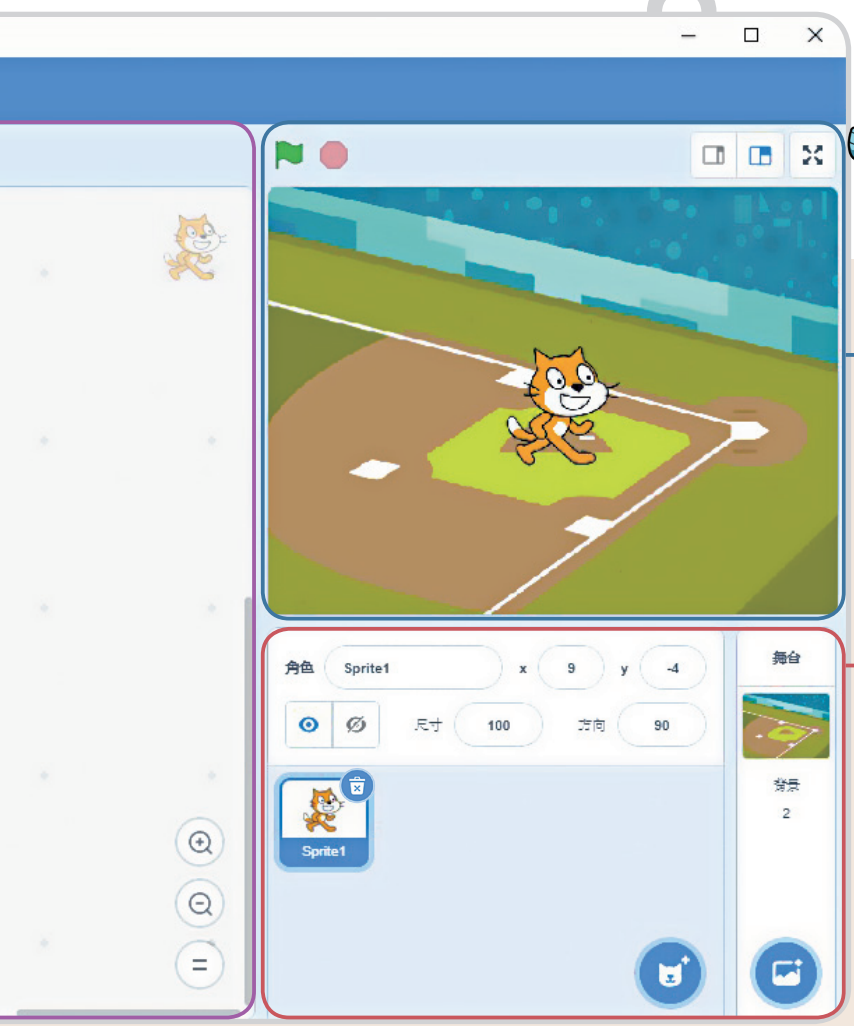
按 造型 鍵顯示此面板，可定義該角色所有會出現的造型。

音效面板

按 音效 鍵顯示此面板，可設定聲音檔，讓角色發出聲音的效果。

程式面板的功能

- **動作**：設定角色的位置、方向、旋轉及移動。
- **外觀**：設定角色的造型、特效、大小、及文字顯示。
- **音效**：設定聲音的播放、音量，也可設定樂器與彈奏音符。
- **事件**：設定當某個情況發生時需要執行的程式。
- **函式積木**：自訂積木或擴展積木。
- **控制**：設定程式的流程控制，如判斷、重複等。
- **偵測**：用於取得鍵盤或滑鼠的資訊，以判斷角色是否發生特定狀況。
- **運算**：用於算術運算、邏輯運算、字串運算或產生亂數。
- **變數**：設定變數及清單以儲存資料。



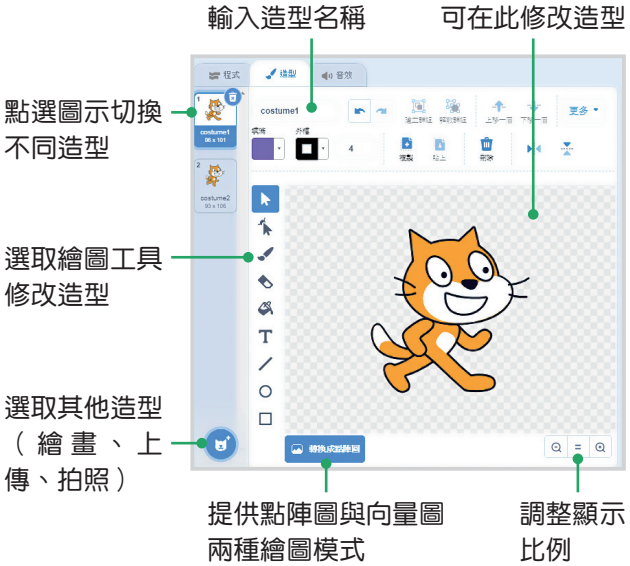
舞臺區

Scratch 提供的繪圖環境，大小是寬 480 點，高 360 點，此區會呈現程式執行結果。

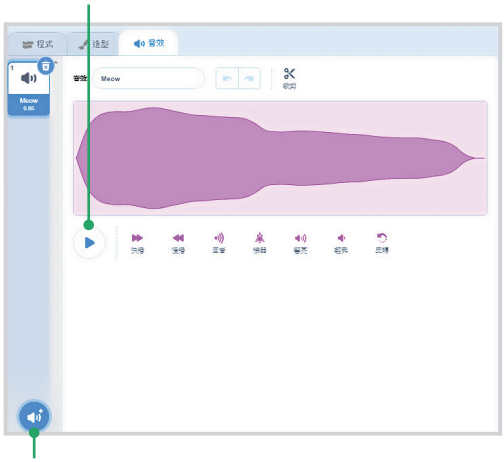
角色區

此區會列出所有用到的角色縮圖。按下角色縮圖，即可在角色資訊區，重新給角色取一個有意義的名稱。在 Scratch 中，雖然只有一個舞臺，但你可以為這個舞臺設定不同的背景。

造型面板的功能



音效面板的功能



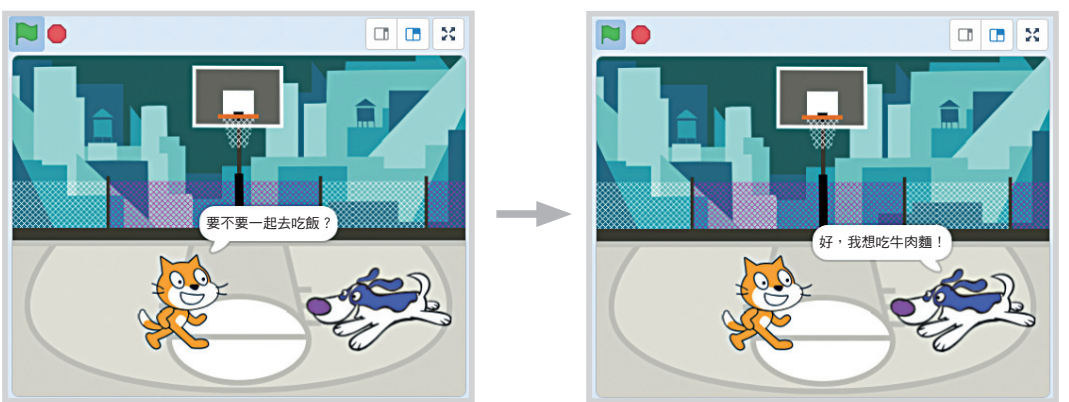
2-2-3 簡易的 Scratch 動畫實作

認識了 Scratch 的操作介面後，接著來實作一個簡易的動畫，以了解這些介面交錯運用所能達到的功能。

範例

對應習作第 43 頁

設定場景在籃球場，小貓和小狗碰面，進行簡單的對話後再相約去吃飯。以下為範例完成時的舞臺區畫面。



準備工作

回到 Scratch 的操作介面，開始設定舞臺與背景。

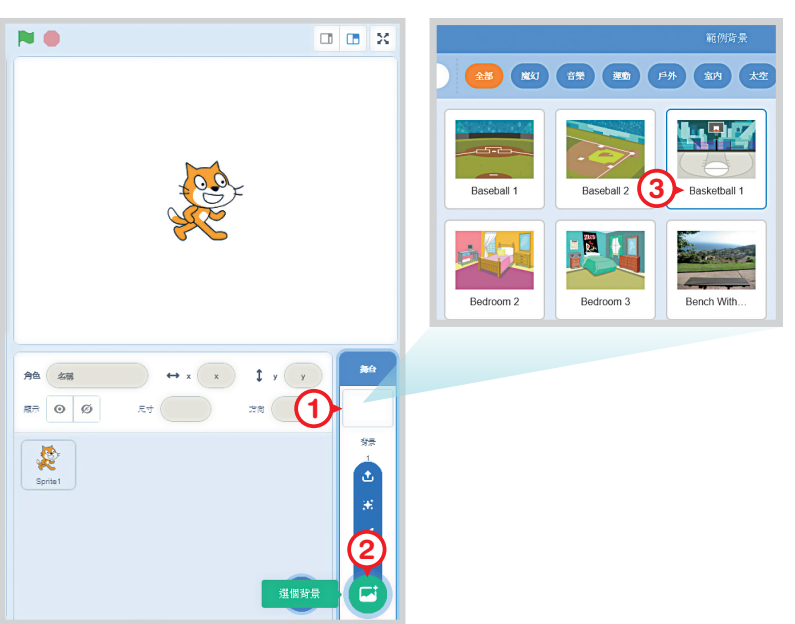
舞臺設計

步驟 1 開新檔案，匯入舞臺背景。

① 在角色區中用滑鼠點選右下方的**舞臺**。

② 點選下方**選個背景**按鍵列的 。

③ 從範例背景中，選擇 Basketball 1 圖片。

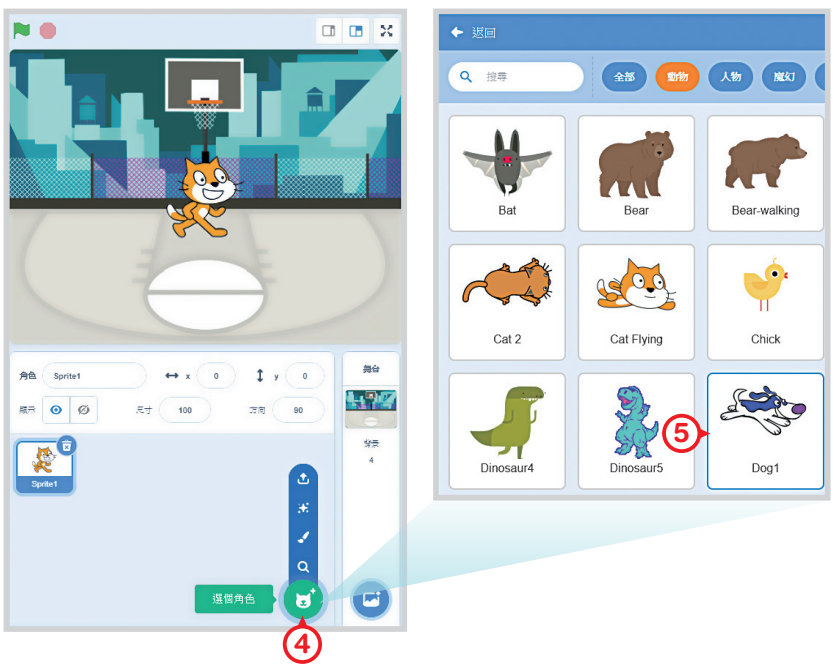


實作動畫

在開啟一個新的專案時，Scratch 會預設一隻小貓角色，而且有兩種造型、一種聲音。我們的動畫剛好有小貓角色，因此可以直接用這隻預設的小貓。除了小貓之外，還需要一隻小狗角色，可以從 Scratch 內建的範例庫中做挑選。

角色安排

步驟 2　新增小狗角色，並轉換小狗方向，讓小貓與小狗互看。

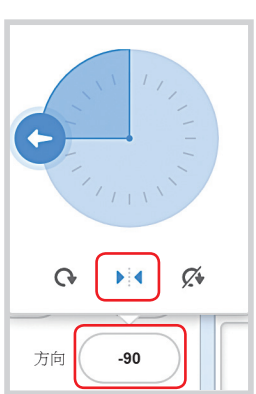


④ 在角色區中，下方**選個角色**按鍵列中，按下 ，準備開啟小狗角色。

⑤ 從範例角色中，選擇 Dog1。

⑥ 用滑鼠將小貓與小狗拖曳至舞臺區的左下方與右下方。

⑦ 點選小狗角色縮圖，在角色資訊區調整角色的方向，將預設的方向 90 改為 –90，旋轉方式修改為 。



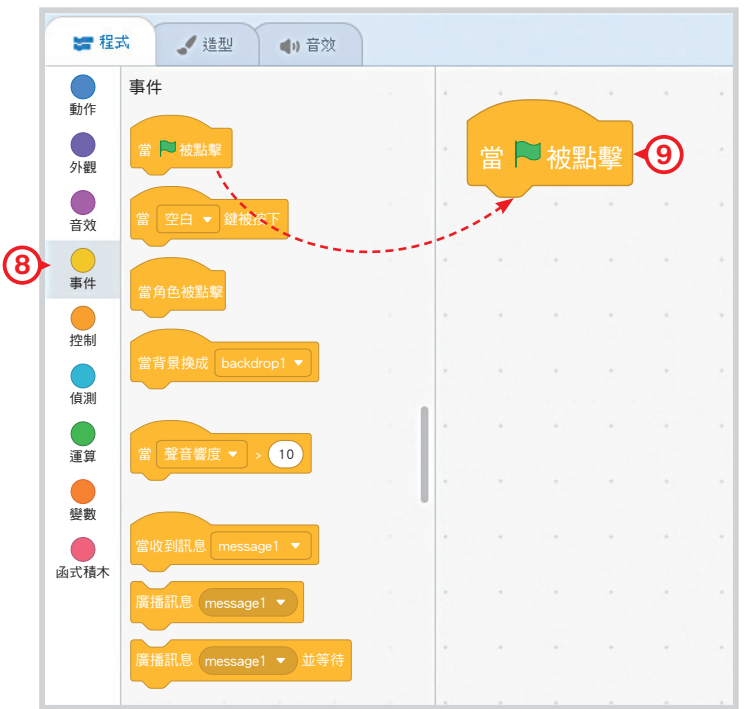
撰寫程式

步驟 3 小貓準備開始走路。

⑧ 點選事件類別。

⑨ 將 

積木拖曳至腳本區中。



小提示

如何使用此積木？

通常放在程式的起頭，用來啟動程式。

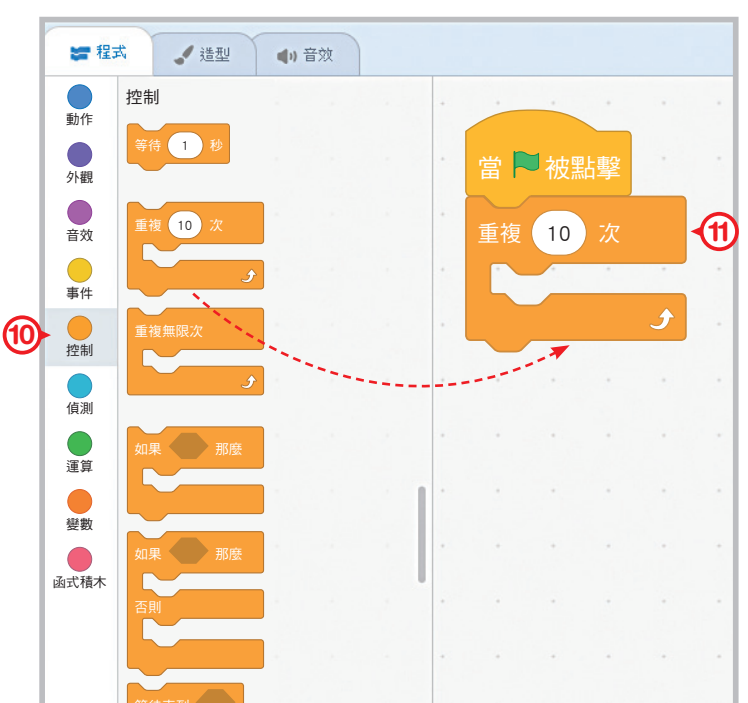
下方有突起，可以拼接上方有凹槽的積木。

步驟 4 讓小貓的腳走動 10 次。

⑩ 點選控制類別。

⑪ 拖曳 

積木，拼接在前一個積木下方。



小提示

如何使用此積木？

在此輸入重複執行次數，預設是 10 次。

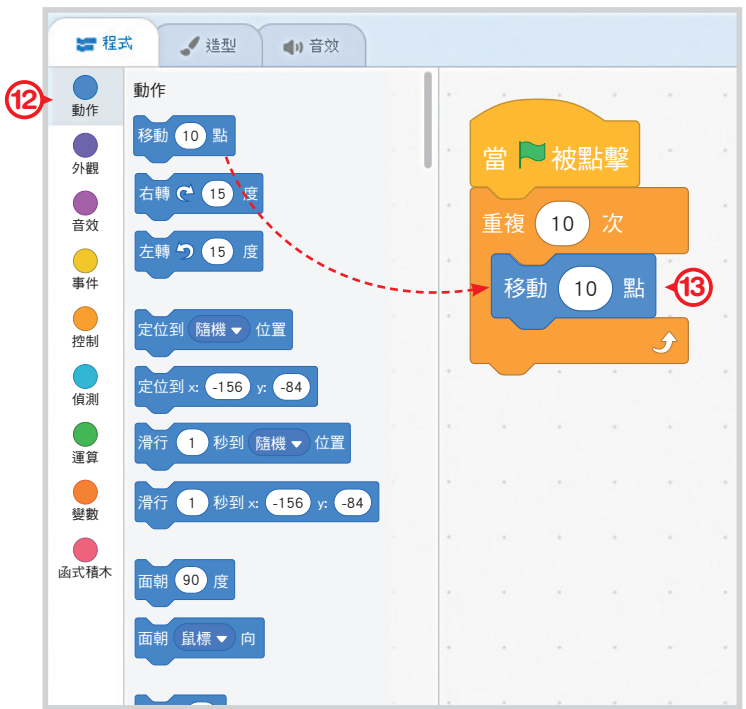
可以嵌入需要重複執行的積木。

步驟 5 設定讓小貓向前移動 10 點。

⑫ 點選**動作**類別。

⑬ 拖曳 

積木，嵌入前一個積木下方。



小提示

如何使用此積木？

設定角色往前移動距離，移動幾點代表移動幾個像素。

輸入移動點數

步驟 6 設定叫出小貓左右腳交替造型。

⑭ 點選**外觀**類別。

⑮ 拖曳 

積木，嵌入前一個積木下方。

小提示

如何使用此積木？

設定角色換至下一種造型，或由最後的造型換至第一種造型。

小貓有兩種造型，重複執行讓造型不斷切換，做出走路動作。

因電腦的程式執行速度很快，重複 10 次一下子就結束，所以要想辦法讓過程慢下來。

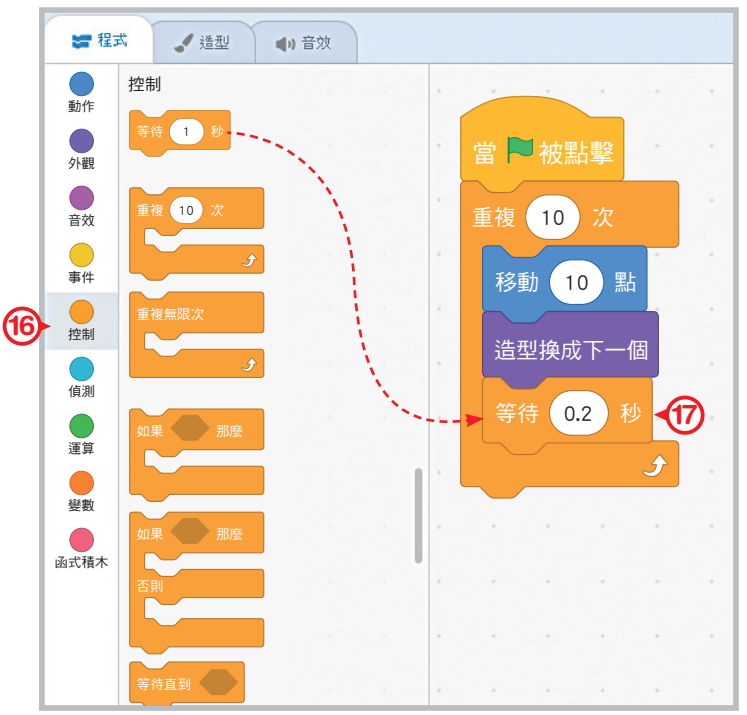
步驟 7 設定小貓每走一步就停頓一下。

⑯ 點選**控制**類別。

⑰ 拖曳 

積木，嵌入前一個積木下方，並將秒數改為 

。



小提示

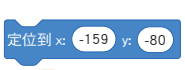
如何使用此積木？

設定角色移動速度。等待時間愈長，移動速度愈慢。

按下舞臺區的綠旗時，小貓就會往前走。如果我們希望下一次再按下綠旗時，小貓會先回到原來的位置再往前走，就要設定此角色的起始位置。

步驟 8 設定小貓從何處開始走路。

⑱ 點選**動作**類別。

⑲ 拖曳 

積木，嵌入 

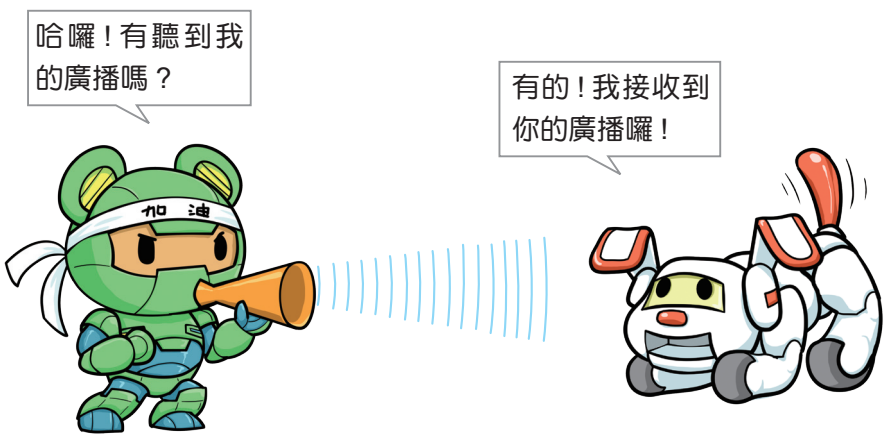
積木下方。

小提示

設定起始位置

用此積木，輸入角色一開始出現的位置坐標。

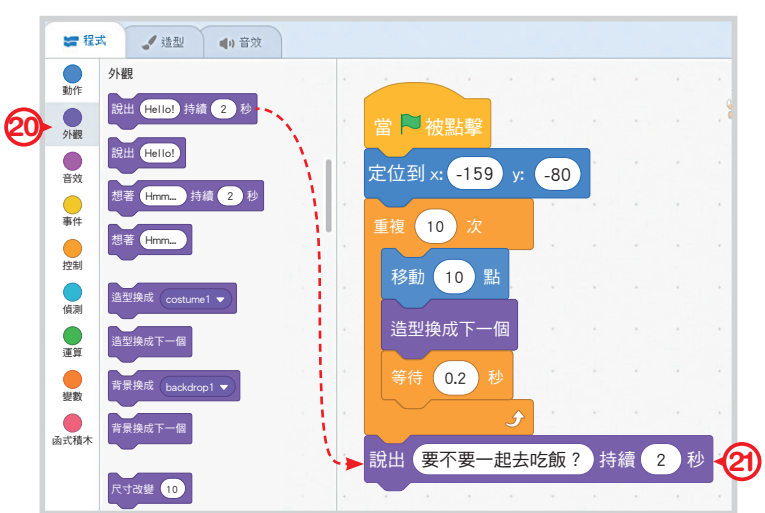
在這個範例中，當小貓走到小狗面前時，就要開始簡單的對話，小貓先問：「要不要一起去吃飯？」，小狗回答：「好，我想吃牛肉麵！」由於小狗的回答是接續著小貓說完才開始講的，需要使用廣播與接收的觀念，讓不同角色間的積木可以串接起來。



步驟 9 **開始撰寫貓狗對話程式，設定讓小貓先說話。**

⑳ 點選外觀類別。

㉑ 將 說出 Hello! 持續 2 秒 積木，拼接在之前完成的程式積木下方，改為 說出 要不要一起去吃飯？ 持續 2 秒 。



小提示

如何使用此積木？

用來顯示角色表現在畫面的訊息，例如：角色要說的話。

步驟 10 設定小貓發訊息，啟動小狗程式。

22 點選**事件**類別。

23 將 廣播訊息 message1 積木，拼接在前一個積木下方，改為 廣播訊息 小狗說話 。

小提示

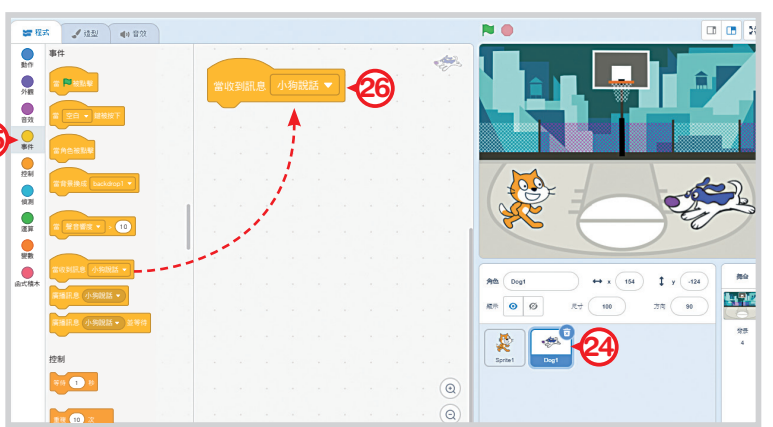
如何使用此積木？

主要使用在此角色對其他角色發送訊號，開始啟動或控制其他角色的程式。

先用滑鼠按下 ▼ 廣播訊息 message1 → 出現選單後，點選「新的訊息」。 廣播訊息 message1 → 新的訊息 message1

將跳出「新的訊息」視窗，並填入「小狗說話」，表示用來啟動在小狗角色上的說話程式。 廣播訊息 小狗說話 完成！

步驟 11 小狗接收到訊息後，啟動程式。



24 回到角色區點選小狗。

25 點選**事件**類別。

26 將 當收到訊息 小狗說話 積木拖曳至腳本區中。

步驟 12 讓小狗說話。

27 點選**外觀**類別。

28 將 說出 Hello! 持續 2 秒 積木，拼接在之前完成的程式積木下方，改為 說出 好，我想吃牛肉麵！ 持續 2 秒

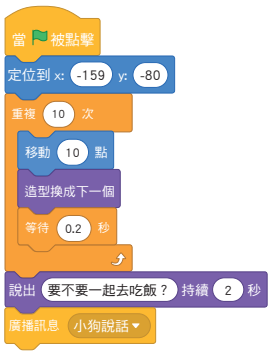
小提示

廣播積木與接收積木的互動方式

❶ 當程式執行到小貓的廣播積木。

❷ 小貓的廣播積木發出小狗說話訊息。

❸ 小狗的接收積木收到訊息後，啟動執行。



小貓程式

小狗程式

按下 測試你的程式。完成了這個 Scratch 動畫創作，你是不是覺得寫程式一點都不難，而且很有趣呢！關於積木排列方式，都會影響程式執行的結果。不妨試試習作附件 7 的小遊戲，練習排列的先後順序。

在此單元中，我學到的有：

1. 明白了完成動畫所需的一系列步驟。
2. 為了讓角色走路，我使用了迴圈執行同一序列動作許多次。
3. 為了讓角色對話，我使用了廣播功能來形成另一事件發生的某事件。
4. 確認程式碼可以運作，包含發現並解決問題。
5. 了解並連結整部動畫與部分程式碼間的關係。
6. 了解運算也是一種創造工具，而我可以進行創造！

以上介紹了 Scratch 的操作介面及範例的實作，你務必要了解且熟練腳本區、舞臺區和角色區所扮演的角色及功能。熟練之後，就可以開始用 Scratch 計算與繪圖了！

2-3 Scratch 程式設計 – 計算篇

▲圖 2-6 結帳時應用的資訊科技。

電腦剛發明時，被稱為電子計算機。它的功能，主要用於幫助人們處理大量的計算問題（圖 2-6），本節將運用流程圖與 Scratch 程式設計來解決一些數學上的計算問題，讓你體驗一下電腦程式的魅力。

在進行算術運算時，必須指定適當的運算符號來表達運算式，如下表所示。

算術運算的類型、符號、 Scratch 運算積木對照表

運算類型	運算符號	Scratch 運算積木
加	＋	(+)
減	－	(-)
乘	×	(*)
除	÷	(/)

接下來，我們將從簡單的範例開始，學習如何分析問題，之後形成流程圖，接著再依照流程圖寫出 Scratch 程式碼解決問題。

而在解決問題的過程中，可以使用後面介紹的循序結構、選擇結構和重複結構三種基本流程結構，來安排解題步驟的順序與過程。

準備工作

設定積木

由於接下來需要頻繁使用變數積木，因此先熟悉一下基礎的操作過程。

步驟 1 開始設定新變數。

① 到程式面板，點選**變數**類別。

② 按下**建立一個變數**鍵，來新增變數。

步驟 2 選擇名稱與適用範圍。

③ 跳出**新的變數**視窗。

④ 這裡輸入變數的名稱 A。

⑤ 接著點選**適用於所有角色**。

⑥ 按下**確定**鍵。

步驟 3 產生變數積木群組。

⑦ 產生了 5 種變數積木。

小知識

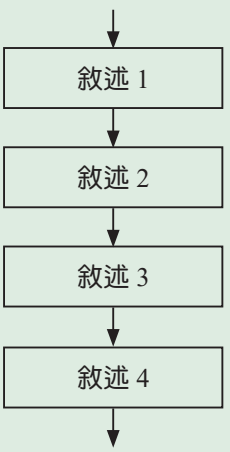
變數

在程式設計過程中，若需要記錄某些文字或數字資料，可以用一個名稱記錄起來，且暫存在電腦中。因資料內容是可變動的，因此，這個名稱就是變數。

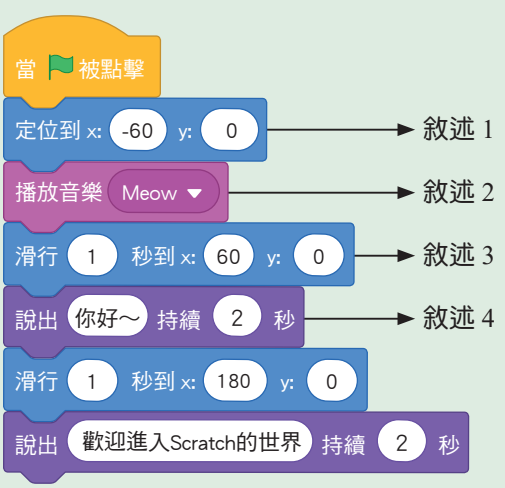
2-3-1 循序結構

程式若是由上而下依序執行，就稱為循序結構。

循序結構－流程圖



例如：Scratch 程式碼由上而下依序執行



範例

對應習作第 45 頁

求 A 與 B 兩數的平均數。

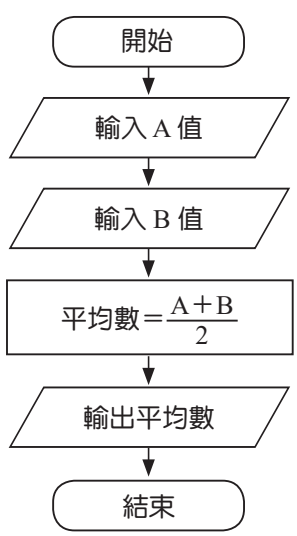
問題分析

閱讀並分析題目，將問題拆解為輸入、處理、輸出三個階段，請寫出各階段相對應的內容。

階段	內容
輸入	A 與 B 的值
處理	平均數 $=\frac{A+B}{2}$
輸出	平均數

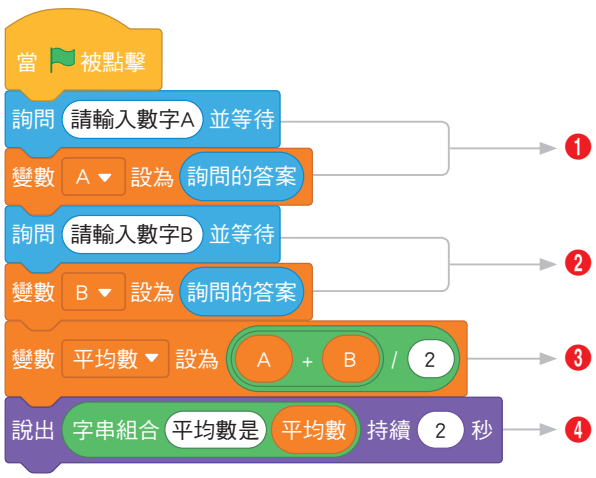
畫流程圖

將問題分析的內容，依序畫成流程圖。



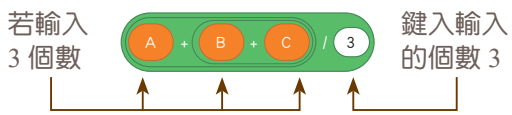
撰寫程式

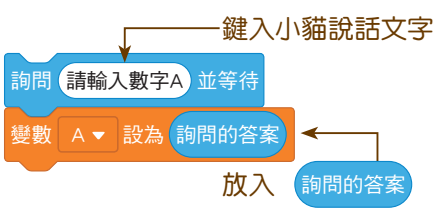
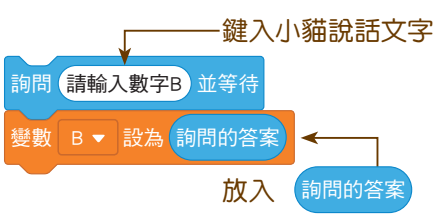
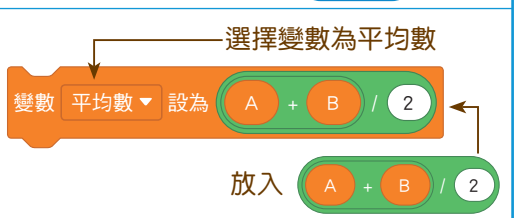
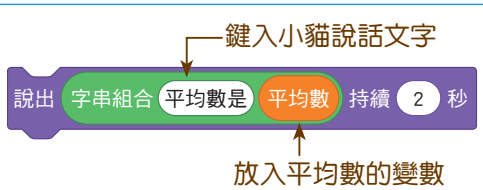
依照流程圖的各步驟，寫出 Scratch 程式碼。



分析

如何撰寫計算兩個數以上平均數的積木？



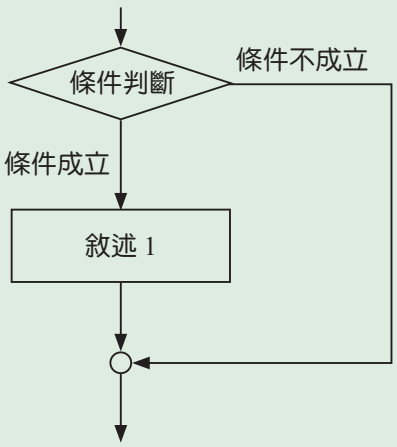
問題解析	問題實作
ⒶA如何設定輸入 A 值？	❶用此積木，設定變數 A 值為 詢問的答案 。 
ⒷB如何設定輸入 B 值？	❷用此積木，設定變數 B 值為 詢問的答案 。 
ⒸC如何計算 A 與 B 的平均數？	❸用此積木，設定變數平均數，數值是 $\frac{A+B}{2}$ 。 
ⒹD如何輸出平均數？	❹設定小貓說出平均數的數值，持續 2 秒。 

2-3-2 選擇結構

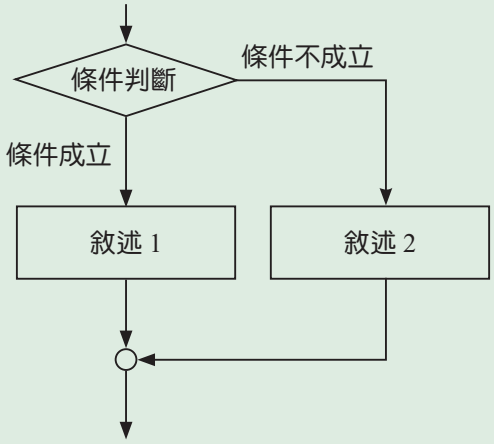
依條件判斷的結果，決定要執行哪一段程式碼敘述。條件判斷的結果，有條件成立（true）及條件不成立（false）兩種。

在單向選擇結構中，只定義了條件成立時要執行的敘述；在雙向選擇結構中，分別定義了條件成立與條件不成立時要執行的敘述。

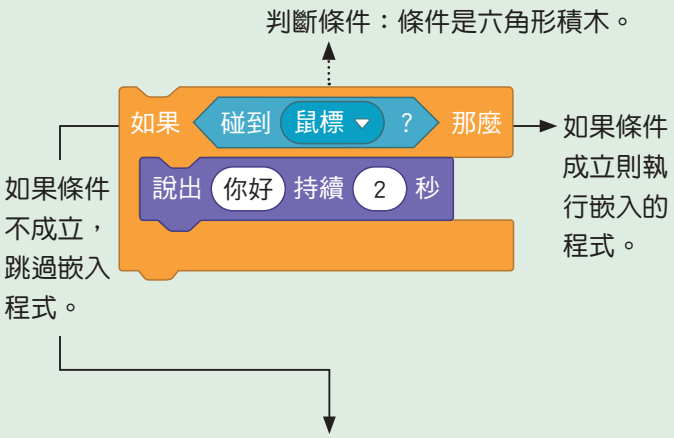
單向 選擇結構－流程圖



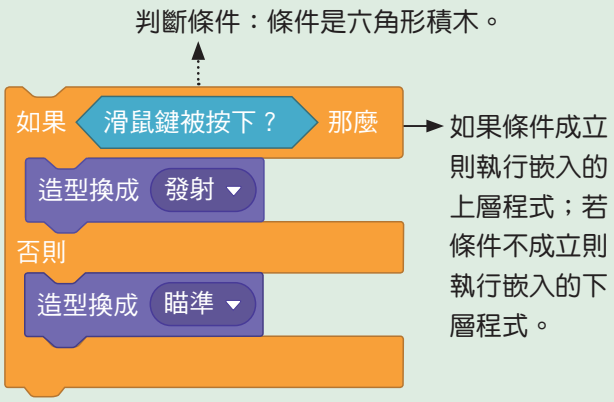
雙向 選擇結構－流程圖



例如：使用單向選擇結構的 Scratch 程式碼



例如：使用雙向選擇結構的 Scratch 程式碼



對應習作第 47 頁

範例

將輸入的各項成績換算為學期成績，並判斷學期成績是否及格？
（作業成績占 40%、測驗成績占 40%、平時表現占 20%，學期成績 60 分為及格分數。）

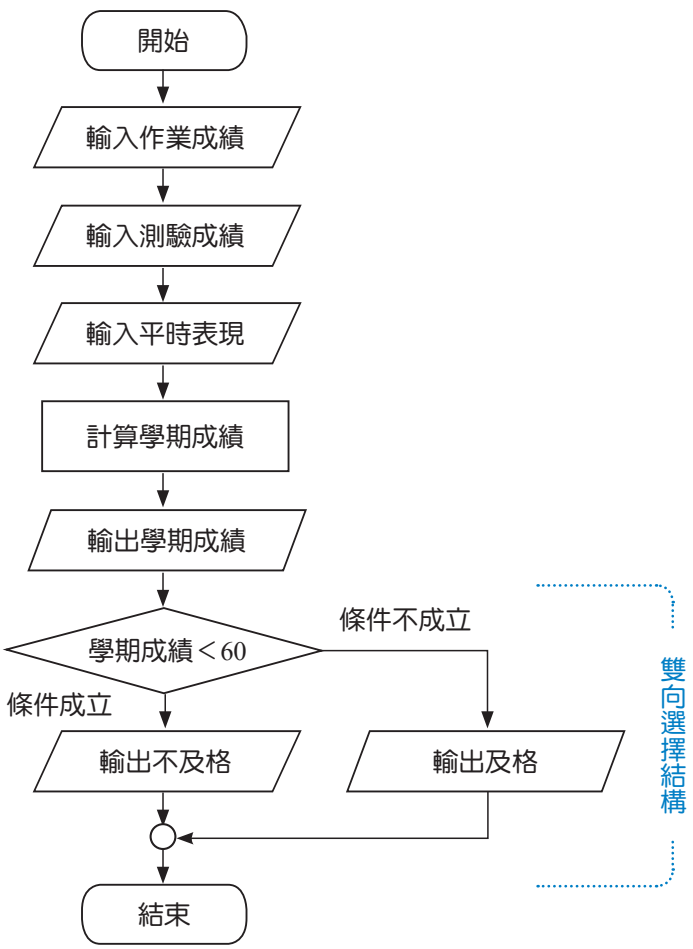
問題分析

閱讀並分析題目，將問題拆解為輸入、處理、輸出、判斷、輸出五個階段，請寫出各階段相對應的內容。

階段	內容
輸入	
處理	
輸出	
判斷	
輸出	

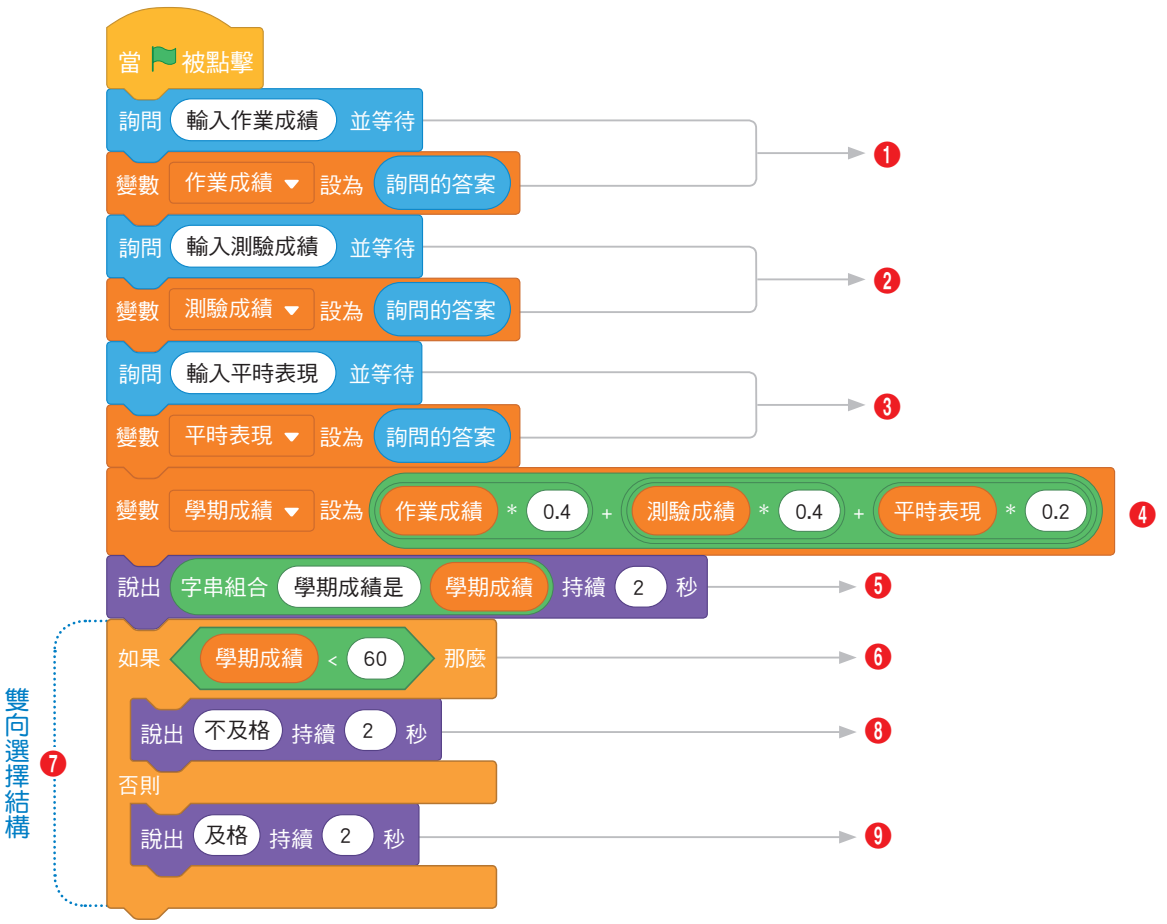
畫流程圖

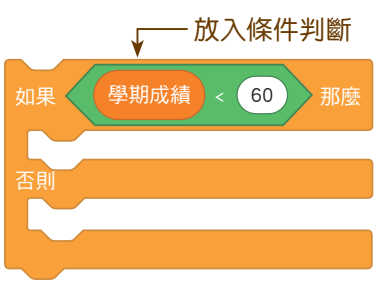
將問題分析的內容，依序畫成流程圖。



撰寫程式

依照流程圖的各步驟，寫出 Scratch 程式碼。



問題解析	問題實作
㈠如何設定輸入作業成績、測驗成績、平時表現？	❶設定**輸入作業成績**。 
	❷接著設定**輸入測驗成績**。 
	❸最後設定**輸入平時表現**。 
㈡如何計算學期成績？	❹先寫出算式，再設定**學期成績**。 
㈢如何輸出學期成績？	❺用此積木，說出**學期成績是**。 
㈣判斷學期成績是否不及格？	❻設定此程式為條件，判斷**學期成績**是否低於 60 分？ 
㈤如何依照條件判斷的結果，控制輸出及格或不及格？	❼若條件成立（**學期成績** ＜ 60），執行嵌入積木上層程式，也就是❽。 若條件不成立（**學期成績** ≥ 60），執行嵌入積木下層程式，也就是❾。 
㈥如何設定輸出學期成績是否及格？	❽用此積木，說出**不及格**。 
	❾用此積木，說出**及格**。 

2-3-3 重複結構

需要重複被執行的程式碼可以使用重複結構，根據被重複執行的情況，可以分為計次式迴圈與條件式迴圈兩種。

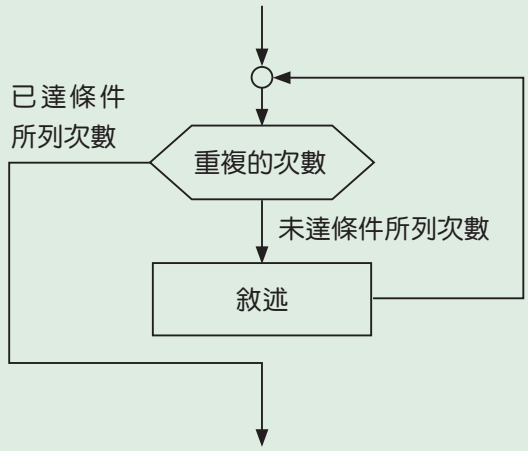
當我們明確知道某段敘述要被重複執行的次數時，就會使用計次式迴圈；當我們不知道某段敘述要被重複執行的次數，而是知道終止情況時，就會使用條件式迴圈。

小知識

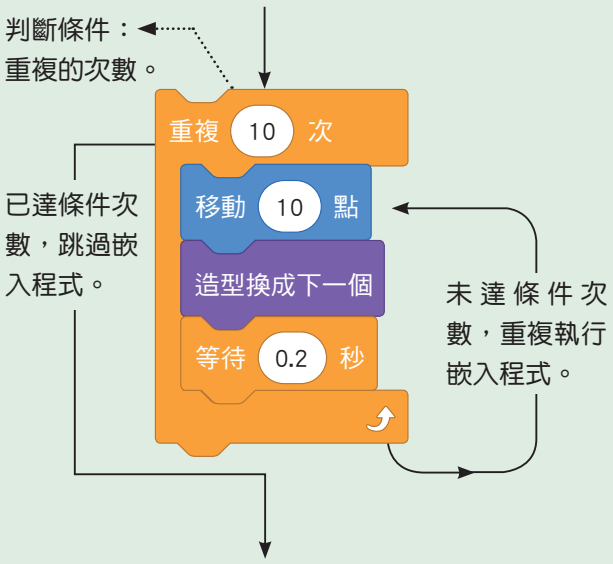
迴圈

在一個程式語言內，可以讓程式碼反覆執行的敘述。

計次式迴圈－流程圖



例如：使用重複結構的 Scratch 程式碼



範例

試計算 1 ＋ 2 ＋ 3 ＋ 4 的值。

問題分析

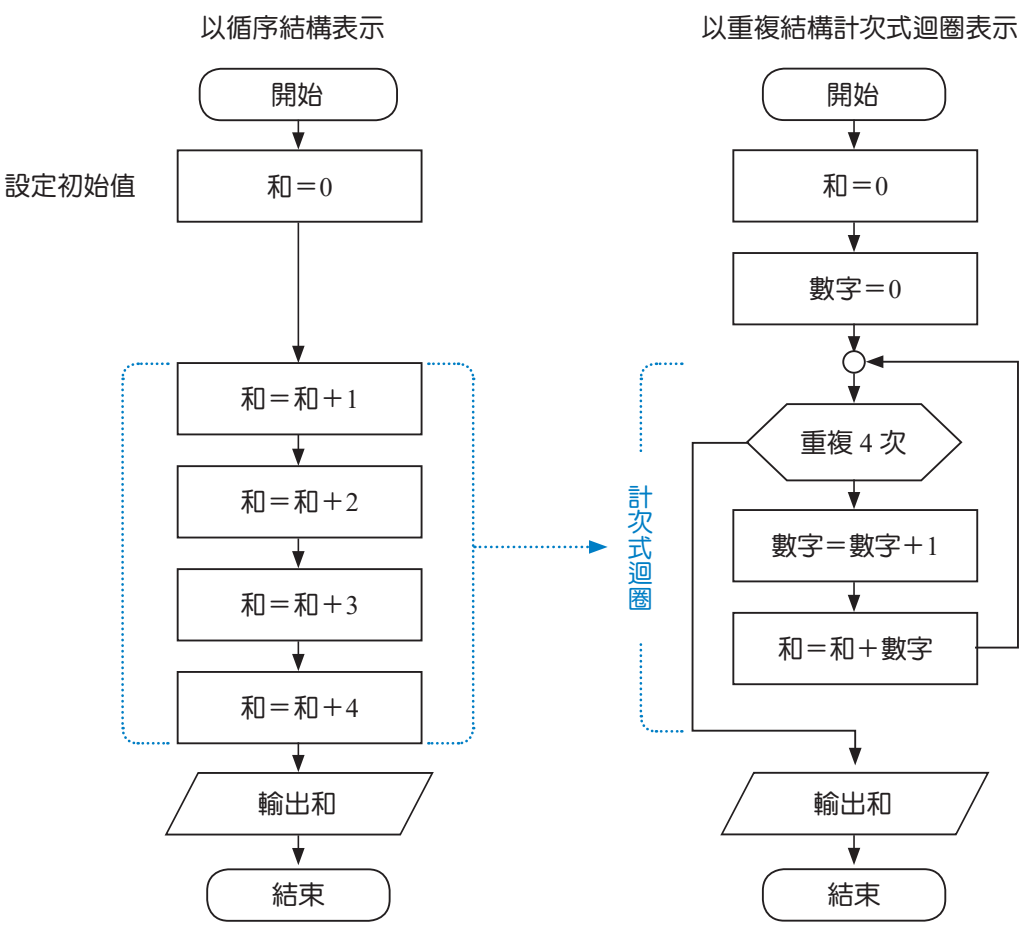
閱讀並分析題目，將問題拆解為處理、輸出兩個階段，請寫出各階段相對應的內容。

階段	內容
處理	
輸出	

設定初始值：和設為 0 。

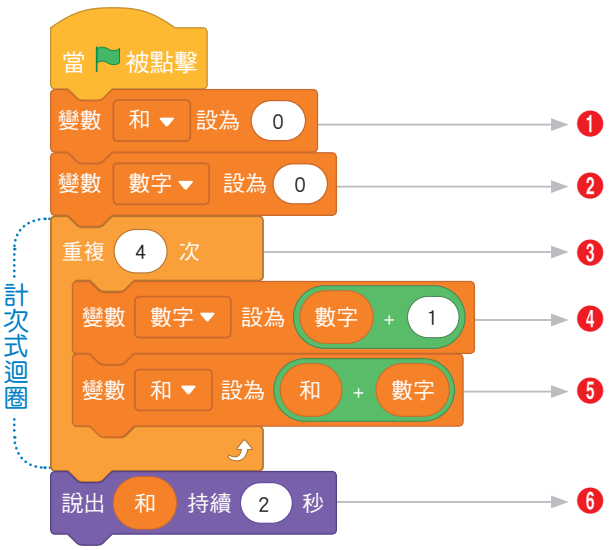
畫流程圖

將問題分析的內容，依序畫成流程圖。



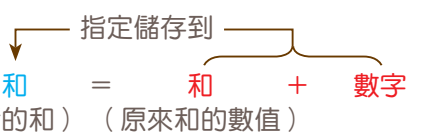
撰寫程式

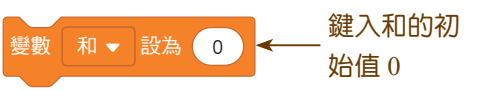
依照流程圖的各步驟，寫出 Scratch 程式碼。



分析

1. 程式的起頭，通常要先設定每個變數一開始的數值，這種程序稱為設定初始值。
2. 程式設計，除了條件判斷之外，等號的意義並不是左邊 ＝ 右邊，而是指定將等號右邊的數值儲存到等號左邊的變數，這種運算方式稱為指定運算（assignment operation）。
3. 流程圖中的運算式：



問題解析	問題實作
(A)如何將開始時的和設為 0 ？	❶用此積木，設定和的初始值為 0 。 
(B)如何將開始時的數字設為 0 ？	❷用此積木，設定數字的初始值為 0 。 
(C)如何重複計算加法 4 次？	❸用此積木，讓嵌入的程式，也就是❹、❺，重複執行 4 次。 
(D)每次重複計算加法時，如何讓數字增加 1？	❹執行時，讓數字的數值每次增加 1。 
(E)每次重複計算加法時，如何讓和加上數字？	❺執行時，讓和每次都加上數字的數值。 
(F)如何輸出和的數值？	❻用此積木，設定說出和的數值。 

用重複結構計次式迴圈來計算 1 ＋ 2 ＋ 3 ＋ 4 編寫程式的速度，好像沒有比使用循序結構快，但是它給了我們很大的擴充彈性，使用計次式迴圈很容易推展到任意數字。畫流程圖時，我們可以設計讓使用者輸入一個數字 N ，再計算從 1 累加到 N 的值。

範例

試計算 1 ＋ 2 ＋ ……＋ N 的值。

問題分析

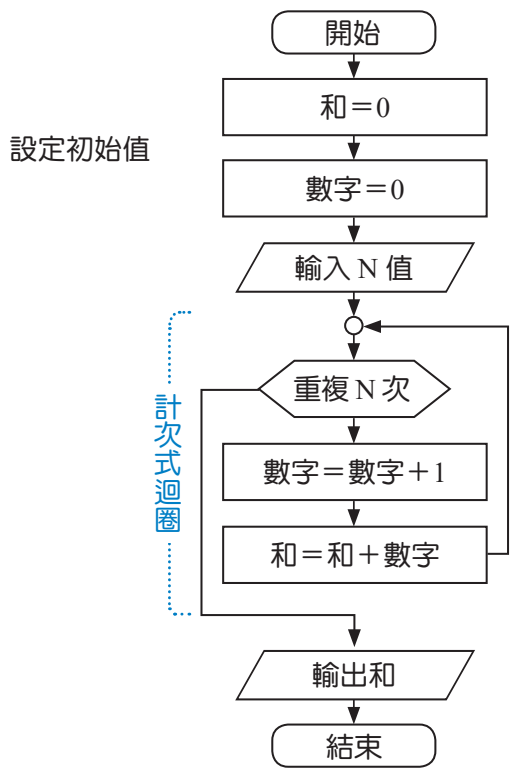
閱讀並分析題目，將問題拆解為輸入、處理、輸出三個階段，請寫出各階段相對應的內容。

設定初始值：和設為 0，數字設為 0。

階段	內容
輸入	
處理	
輸出	

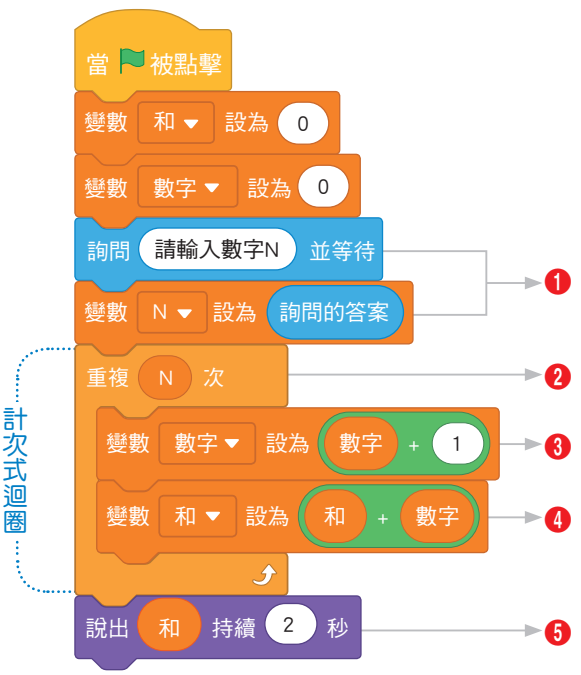
畫流程圖

將問題分析的內容，依序畫成流程圖。



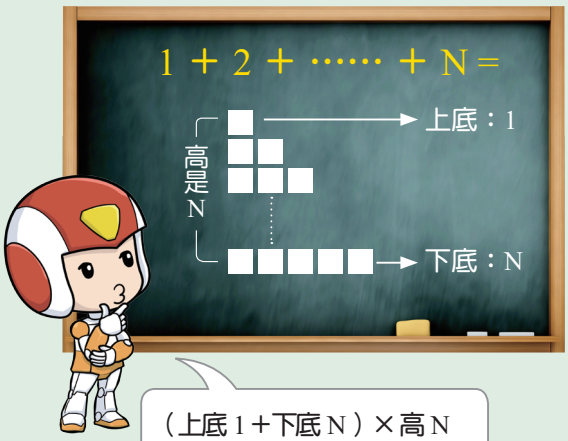
撰寫程式

依照流程圖的各步驟，寫出 Scratch 程式碼。



問題解析	問題實作
ⒶⒶ如何設定輸入 N 的值？	❶用此積木，設定輸入 N 的值。 
ⒷⒷ如何重複計算加法 N 次？	❷用此積木，讓嵌入的程式，也就是❸、❹，重複執行 N 次。 
ⒸⒸ每次重複計算加法時，如何讓數字增加 1？	❸執行時，讓數字的數值每次增加 1 。 
ⒹⒹ每次重複計算加法時，如何讓和加上數字？	❹執行時，讓和每次加上數字的數值。 
ⒺⒺ如何輸出和的數值？	❺用此積木，設定說出和的數值。 

你可能會覺得，要計算 1 ＋ 2 ＋ …… ＋ N 的總和，不是用公式一下子就算出來了嗎？為什麼要那麼麻煩寫一堆程式碼來計算呢？



其實，雖然累加的確有公式可以速算。但是如果需要計算連乘時怎麼辦呢？試著藉由如右範例，模仿累加的程式碼，用計次式迴圈寫看看連乘吧！

範例

試計算 1 × 2 × …… × N 的值。

問題分析

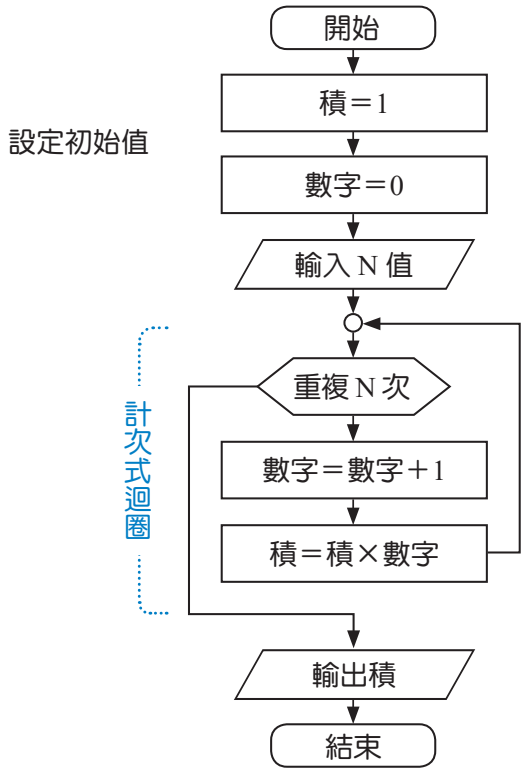
閱讀並分析題目，將問題拆解為輸入、處理、輸出三個階段，請寫出各階段相對應的內容。

設定初始值：積設為 1，數字設為 0 。

階段	內容
輸入	
處理	
輸出	

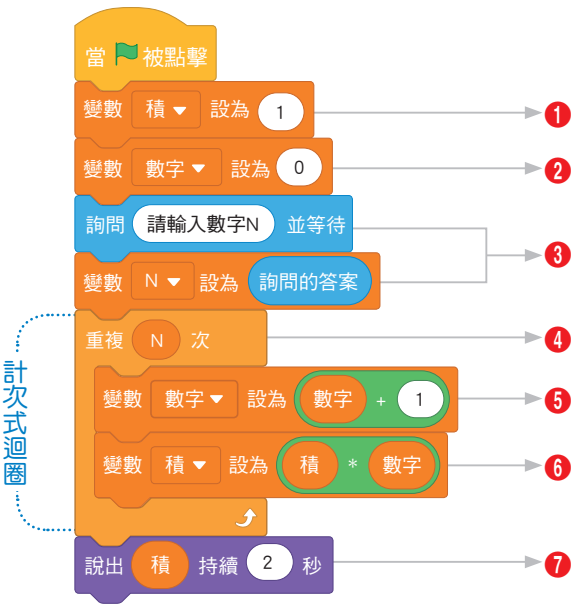
畫流程圖

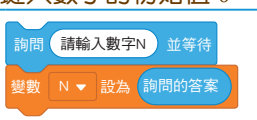
將問題分析的內容，依序畫成流程圖。



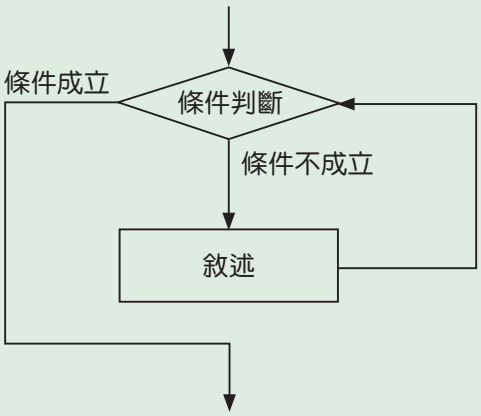
撰寫程式

依照流程圖的各步驟，寫出 Scratch 程式碼。

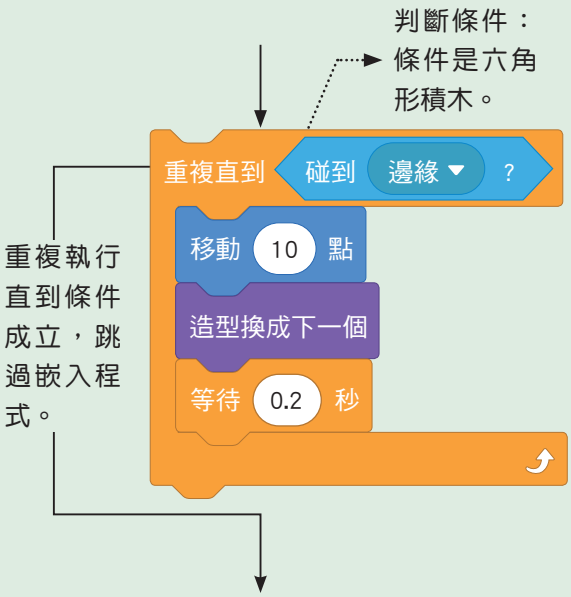


問題解析	問題實作
(A)如何將開始時的積設為 1 ？	❶用此積木，設定積的初始值為 1 。 
(B)如何將開始時的數字設為 0 ？	❷用此積木，設定數字的初始值為 0 。 
(C)如何設定輸入 N 的值？	❸用此積木，設定輸入 N 的值。 
(D)如何重複計算乘法 N 次？	❹用此積木，讓嵌入的程式，也就是❺、❻，重複執行 N 次。 
(E)每次重複計算乘法時，如何讓數字增加 1？	❺執行時，讓數字的數值每次增加 1 。 
(F)每次重複計算乘法時，如何讓積乘以數字？	❻執行時，讓積每次都乘以數字的數值。 
(G)如何輸出積的數值？	❼用此積木，設定說出積的數值。 

條件式迴圈－流程圖



例如：使用條件式迴圈的 Scratch 程式碼



小知識

且 左右兩邊條件式要同時成立，運算結果才會成立。

或 左右兩邊條件式有一個成立，運算結果就會成立。

不成立 反向運算，當條件成立，運算結果為不成立；當條件不成立，運算結果為成立。

範例

設計一個電腦系統的密碼驗證機制，條件如下。

(1)若第一次輸入密碼錯誤後，可再重複嘗試輸入兩次。

(2)若輸入三次密碼都錯誤，跳出使用者帳號被鎖定的訊息。

問題分析

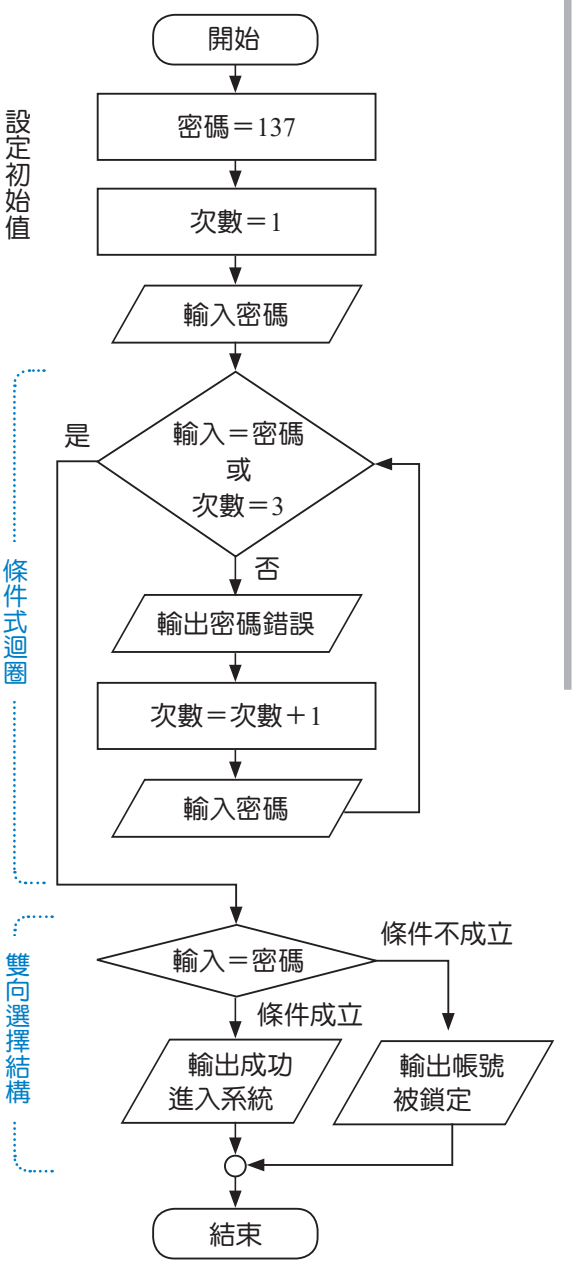
閱讀並分析題目，將問題拆解為輸入、判斷、處理、判斷、輸出五個階段，請寫出各階段相對應的內容。

設定初始值：密碼設為 137 ，次數設為 1。

階段	內容
輸入	
判斷	
處理	
判斷	
輸出	

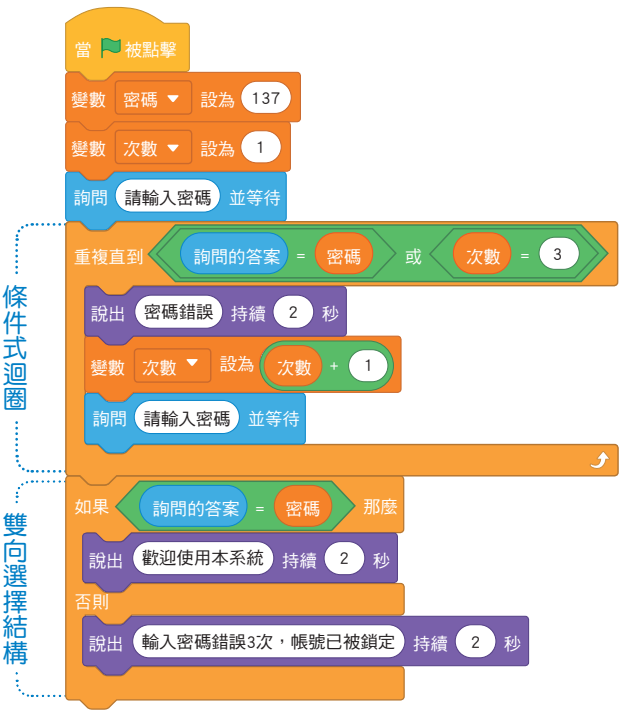
畫流程圖

將問題分析的內容，依序畫成流程圖。



撰寫程式

依照流程圖的各步驟，寫出 Scratch 程式碼。



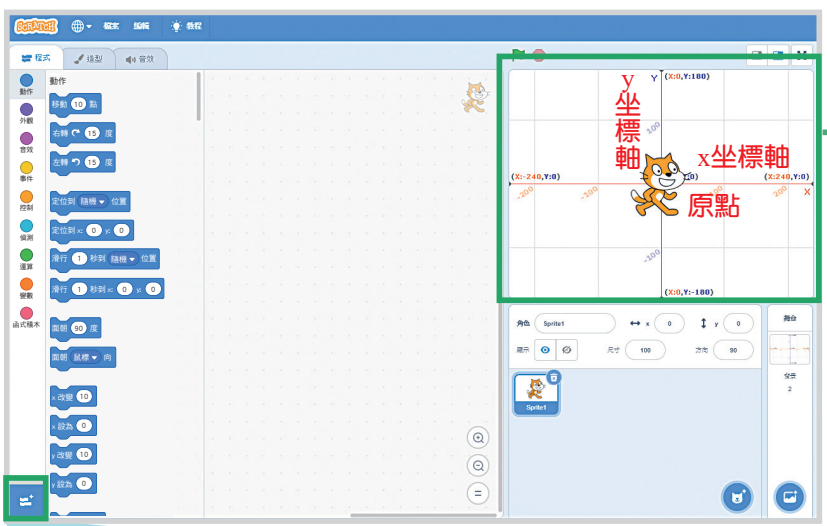
在此單元中，我學到的有：

1. 明白程式碼是由上而下循序執行的序列。
2. 明白程式碼有計次式與條件式兩種不同的迴圈。
3. 明白可以依據條件做決定而執行不同的程式碼。
4. 明白用於數學與邏輯的運算式。
5. 可以使用變數來儲存與更新的資料。

2-4 Scratch 程式設計 – 繪圖篇

Scratch 提供一個很方便且易學的繪圖環境，如下圖，它的舞臺畫面寬 480 點，高 360 點。在畫面的正中央是坐標系統的原點（0 , 0）。原點往右的 x 軸坐標是正數，往左是負數；原點往上的 y 軸坐標是正數，往下是負數。

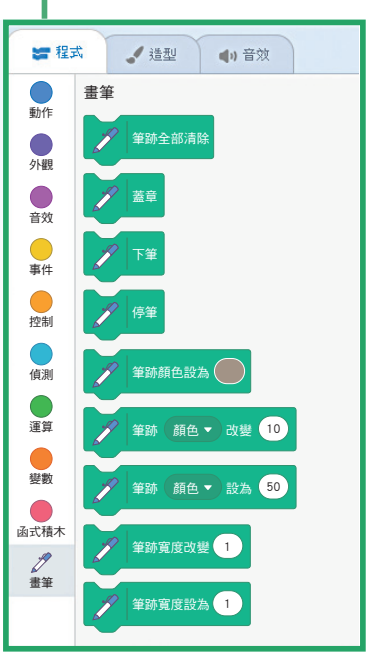
透過 Scratch 的畫筆，可以繪製出很多種有趣的幾何圖案。所以，接下來就要實際體驗，如何運用 Scratch 程式設計來繪圖。在使用畫筆前，需要先從擴展功能中新增相關積木。



舞臺區標示出坐標軸與原點

程式面板中展示所有新增的畫筆積木。

腳本區下方可以添加擴展許多功能，本節使用畫筆功能。



準備工作

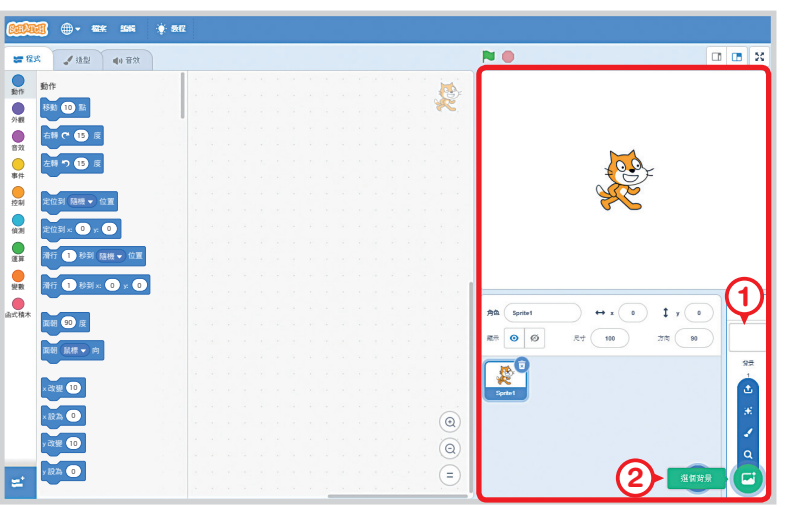
從下一頁起的三個範例中，需要觀察小貓移動的坐標位置變化，因此先設計舞臺，選擇坐標圖形 xy-grid 作為背景。

舞臺設計

步驟 1 開新檔案，匯入舞臺背景。

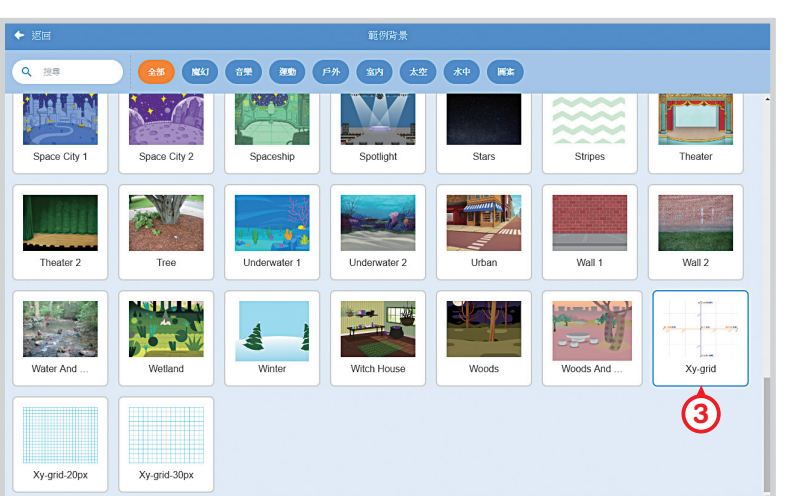
① 在角色區中用滑鼠點選右下方的**舞臺**。

② 點選下方**選個背景**按鍵列的 後，從範例庫中選擇背景。



步驟 2 選擇舞臺樣式。

③ 從範例背景庫中，選擇 xy-grid 圖片。



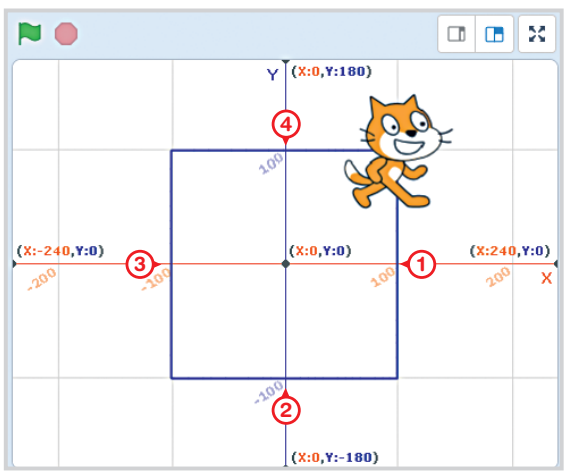
2-4-1 畫正方形

接下來，讓我們利用不同的方法來畫正方形。你可以說出這三種方法的優缺點嗎？

範例

對應習作第 49 頁

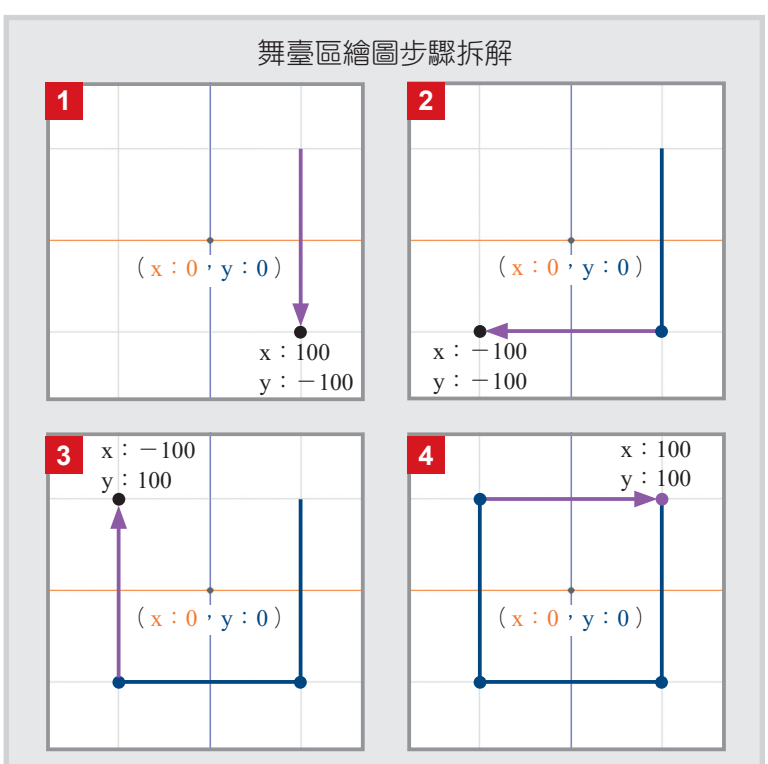
利用坐標積木讓小貓畫出一個正方形。



撰寫程式

步驟 1 設定繪圖位置的坐標。

想一想！如果將圖中的點與坐標刪除，利用方向是否也可畫圖呢？



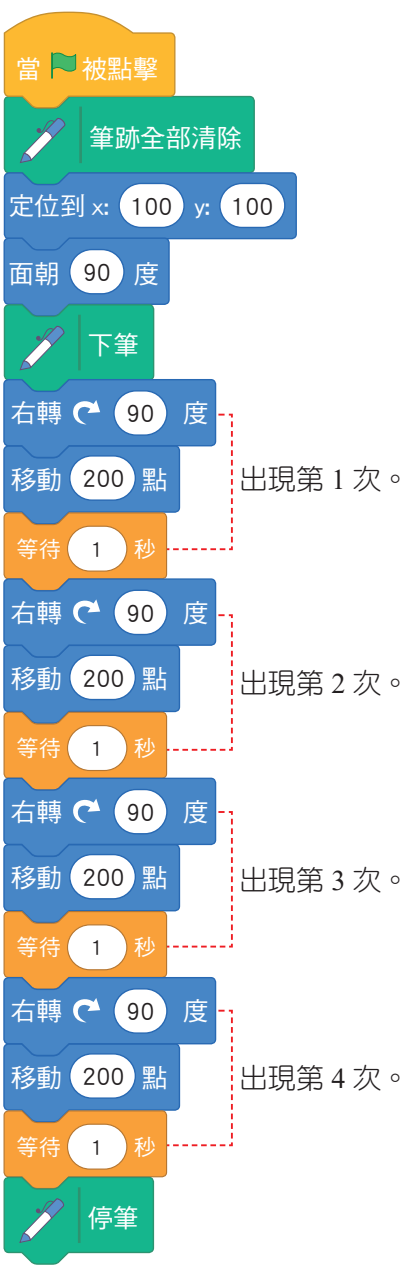
問題解析	問題實作
(A)如何設定角色的初始位置？	用此積木，輸入角色一開始出現的位置坐標。 定位到 x: 100 y: 100
(B)如何控制角色滑行至指定位置？	用此積木，輸入滑行時間與終點位置的坐標。 滑行 1 秒到 x: 100 y: -100

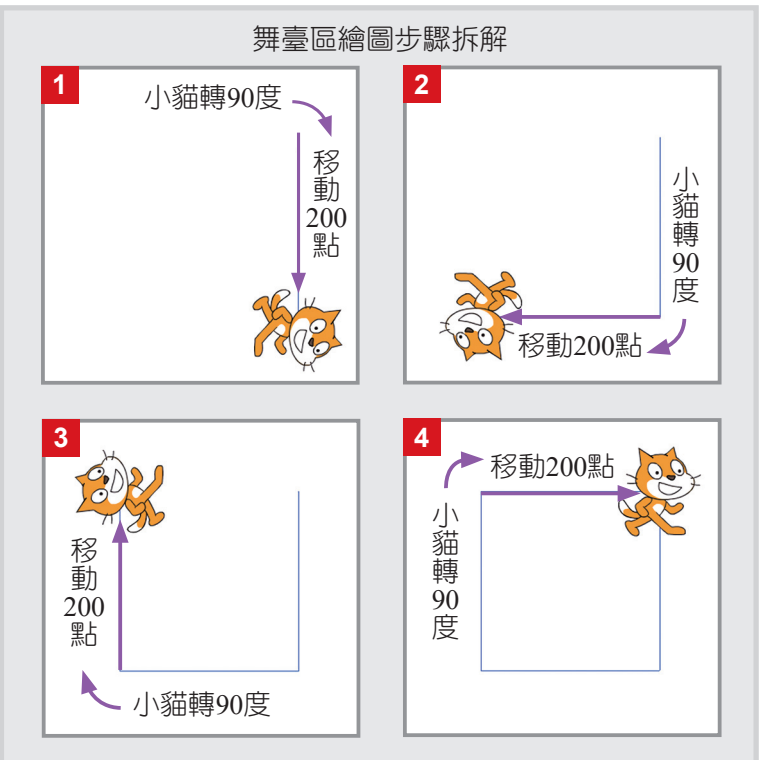
範例

利用方向積木讓小貓畫出一個正方形。

撰寫程式

步驟 1 設定前進方向與移動距離。





想一想！這個程式和上一個直接指定坐標的程式有什麼差別呢？你有沒有發現，當我們使用方向來寫程式時，程式碼會有重複出現的部分呢？對於重複出現的程式碼，我們可以用程式設計的**迴圈**概念來處理，這樣就可以精簡程式碼。

問題解析	問題實作
(A)如何設定角色的初始方位？	用此積木，輸入角色一開始所朝向的方位。 面朝 90 度
(B)如何控制角色的轉向？	用此積木，輸入旋轉角度，向不同方向旋轉。 左轉 90 度 右轉 90 度
(C)如何控制角色移動的距離？	用此積木，輸入要往前移動幾個點，代表移動幾個像素。 移動 200 點

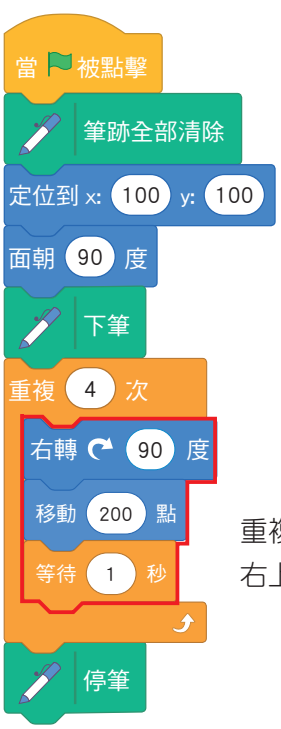
範例

對應習作第 50 頁

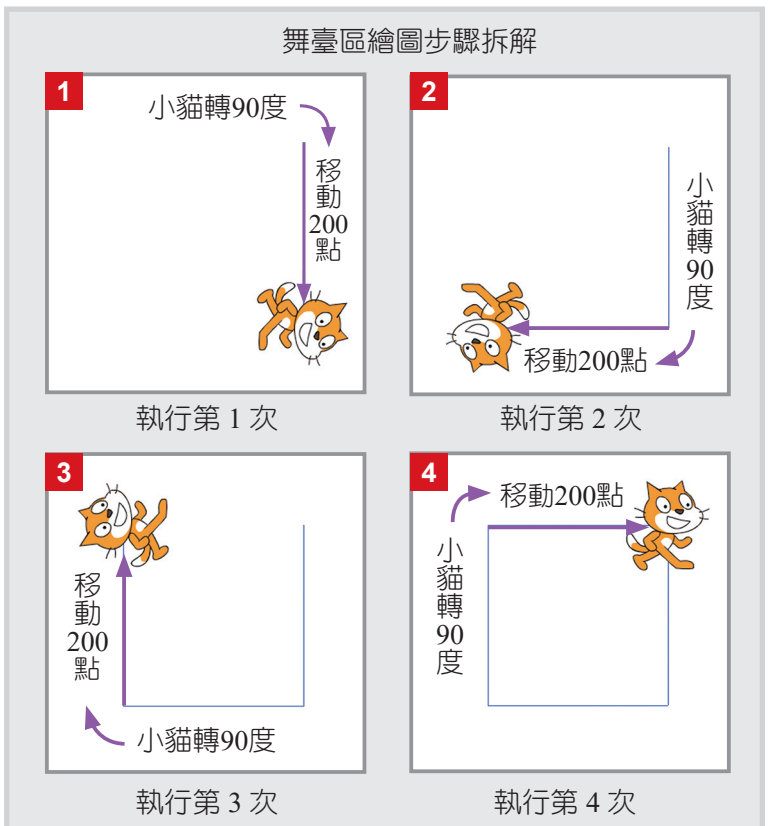
利用計次式迴圈讓小貓畫出一個正方形。

撰寫程式

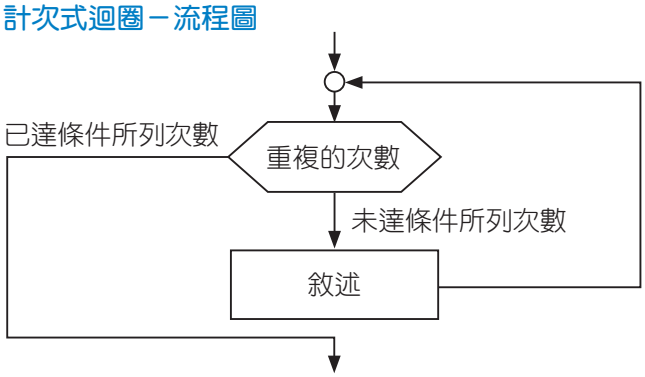
步驟 1 設定計次式迴圈，取代前一頁範例中重複的部分。

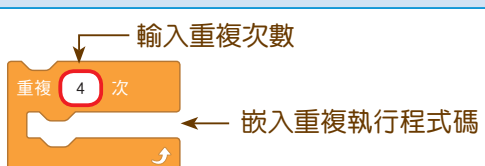


重複執行 4 次，對照右上圖的 1 ～ 4。



計次式迴圈－流程圖



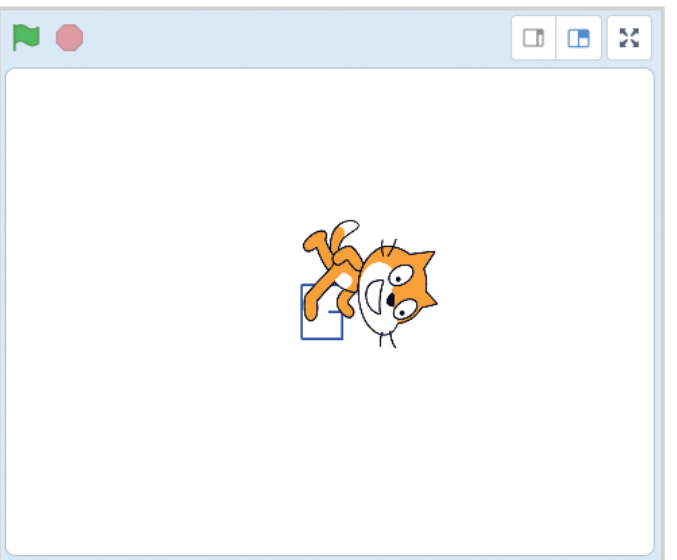
問題解析	問題實作
(A)如何設定計次式迴圈？	用此積木取代重複執行程式碼。 
(B)如何控制角色的轉向？	用此積木，輸入旋轉角度，向不同方向旋轉。 
(C)如何控制角色移動的距離？	用此積木，輸入要往前移動幾個點，代表移動幾個像素。 

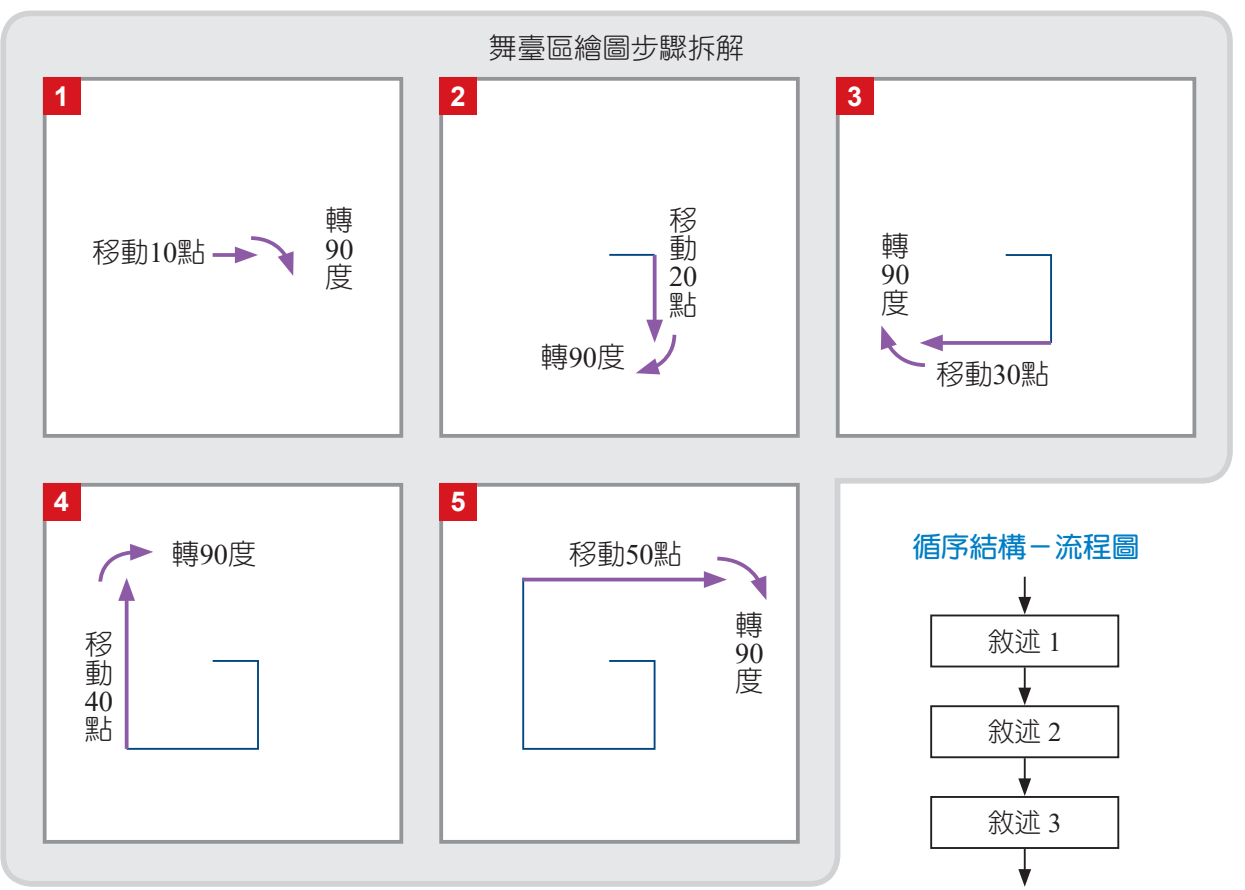
2-4-2 畫擴散方形

現在我們要畫一個擴散的方形，每次移動的距離都增加，並轉 90 度。想一想，有沒有比較快的方法？

範例

利用循序結構畫出一個擴散的方形。



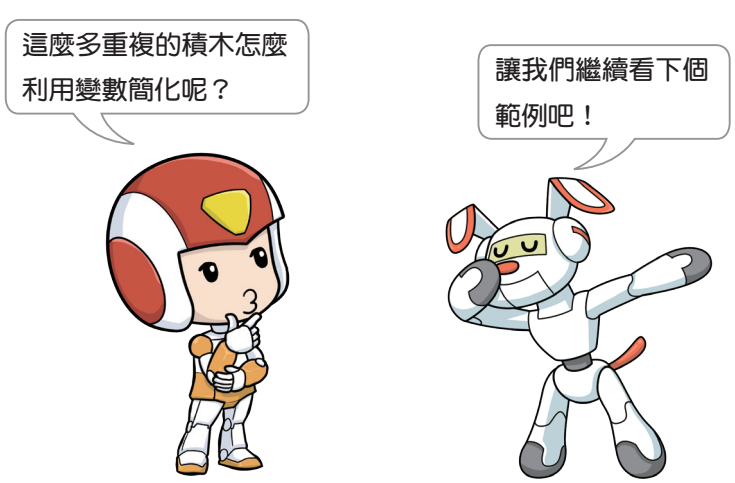


撰寫程式

步驟 1 設定方向，但逐漸增加每次移動的距離。

問題解析	問題實作
(A)如何控制角色移動的距離？	用此積木，輸入要往前移動幾個點，並逐漸增加移動的距離。 移動 10 點
(B)如何控制角色的轉向？	用此積木，輸入旋轉角度，向不同方向旋轉。 右轉 90 度

觀察左方程式碼，有沒有重複出現的地方呢？小貓每次移動的點數是有規則的數列：**10**、**20**、**30**、**40**、**50**，所以我們可以配合**變數**的概念來產生這組數列，以達到精簡程式碼的目的。



範例

利用計次式迴圈與變數畫出一個擴散的方形。

撰寫程式

步驟 1 開始設定新變數。

① 到程式面板，點選變數類別。

② 按下建立一個變數鍵，來新增變數。

步驟 2 選擇名稱與適用範圍。

③ 跳出新的變數視窗。

④ 在這裡輸入變數的名稱長度。

⑤ 接著點選適用於所有角色。

⑥ 按下確定鍵。

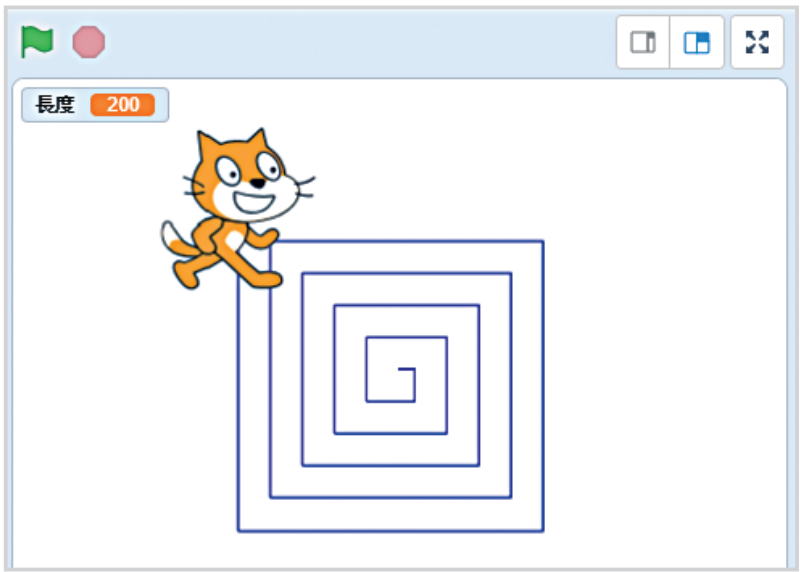
步驟 3 產生變數積木群組。

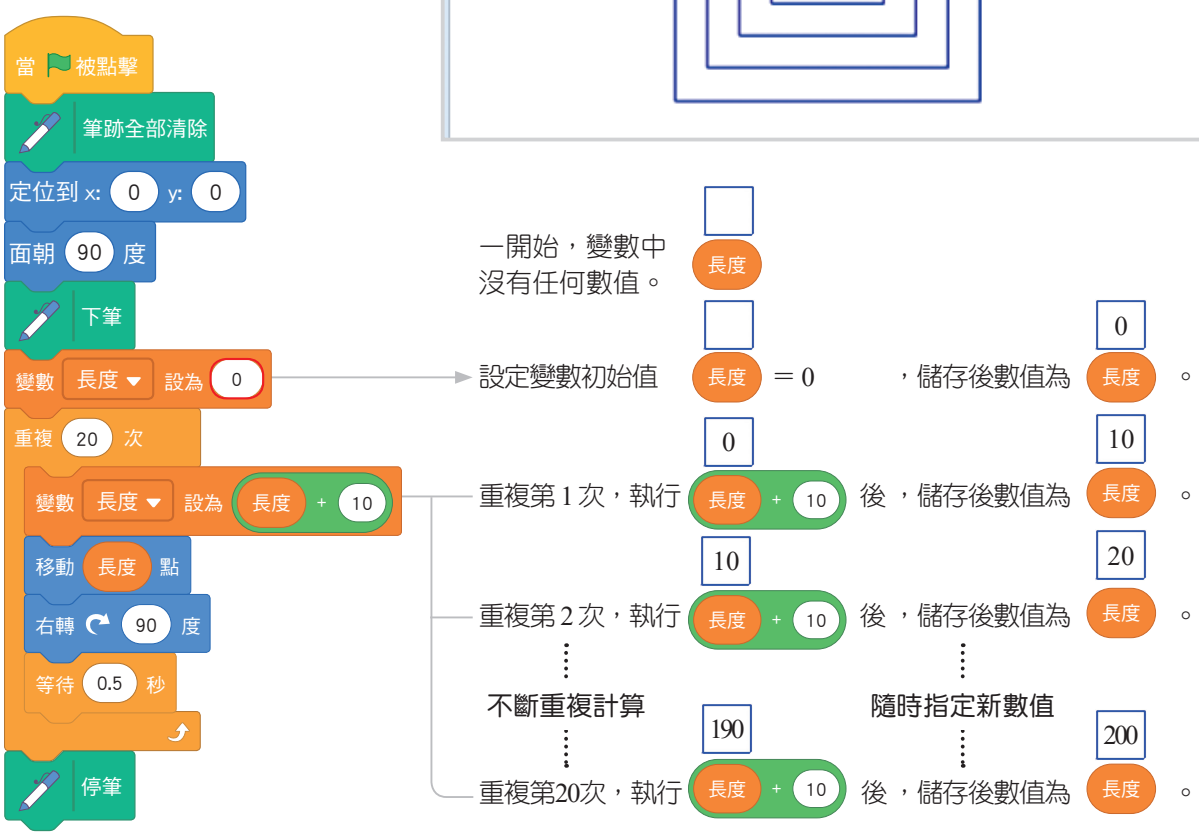
⑦ 產生了 5 種變數積木。

步驟 4

利用迴圈積木撰寫程式

接著開始撰寫程式，加入變數積木，再利用迴圈積木取代上一個範例中重複的部分後，按綠旗執行。





問題解析	問題實作
(A)如何設定變數的初始值？	用此積木，輸入變數一開始的數值。 變數 長度 設為 0 ←鍵入初始值
(B)如何改變變數的數值？	根據指定運算的原則，輸入每次執行到此積木時，要重新儲存到變數的數值。 變數 長度 設為 長度 + 10 放入指定變數數值的運算式
(C)如何改變每次移動的距離？	用變數取代固定數值，改變每次移動的距離。 移動 長度 點 放入長度的變數

2-4-3 畫旋轉正方形

如果每次都先畫一個正方形後，再旋轉 30 度，要怎麼使用迴圈進行繪圖呢？

範例

對應習作第 51、52 頁

利用巢狀結構畫出 12 個旋轉的正方形。

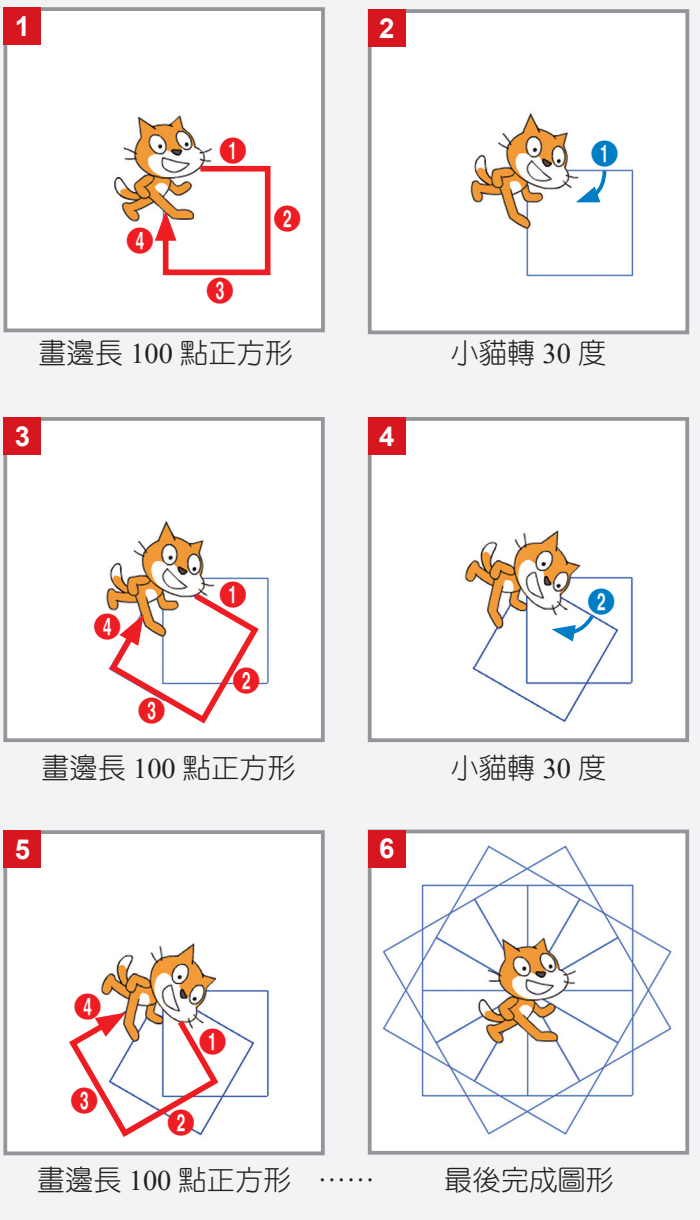
撰寫程式

重複執行 4 次，內層迴圈的步驟對照右圖❶～❹。

重複執行 12 次，外層迴圈每重複 1 次，內層迴圈就會重複 4 次，對照右圖的❶、❷……。

巢狀結構指的是結構裡面還有結構。在範例的程式碼中，重複 12 次的迴圈結構中，又包含了一個重複 4 次的迴圈結構，所以是一個巢狀結構的程式。

舞臺區繪圖步驟拆解



在此單元中，我學到的有：

1. 能夠觀察事物特徵，將重複出現的部分，使用迴圈來簡化程式碼。
2. 能夠使用巢狀迴圈來簡化程式碼。
3. 能夠使用不同的方法達到相同的目標，並且了解其差別。

重點回顧

演算法是一種解決問題的方法，程式語言則是實踐演算法的工具。解決問題的方法與過程可以寫成具體的步驟，再依照步驟去執行。為了表示演算法，可把具體的步驟整理成流程圖，以方便判讀與相互交流。而為了對演算法進行檢驗，必須將演算法轉換成電腦程式。因此，透過程式設計來實作演算法，要學習程式設計則需先選定一種程式語言。由於每人的思考模式不同，解決問題的辦法也不同，設計出來的演算法也會不同，但最重要的就是執行能產生正確的結果。

電腦只是一部機器，為了要指揮電腦完成某項工作，就要配合演算法，編寫許多指令或敘述，這些為完成某項工作而依其邏輯順序寫成的一連串指令，就是程式。只要給予指令，電腦就照指令執行，然後輸出結果。

程式語言發展的歷史遠比電腦來得早。從有程式語言至今，廣被使用的程式語言種類非常多，但因用途不同，功能也不一樣。雖然不同程式語言的語法不一樣，但基本邏輯則類似。建議初學者可以從一般用途的程式語言入門，建立了程式設計的基本概念後，再因應不同的專業需求，選用特殊用途的程式語言。

本課程選用易學有趣的 Scratch 3.0 來學程式設計。Scratch 是免費的自由軟體，它提供了視覺化的圖形操作介面，只要會用滑鼠拖曳積木，就能輕易的撰寫程式。在基礎篇以簡易的動畫實作，介紹操作介面及程式碼。接著在計算篇中，學習程式設計的基本邏輯（循序、選擇及重複）結構，在繪圖篇中，則加強迴圈概念來精簡程式碼。