結合編程的數學課：線上 Python 工作坊報告

戴怡嘉、潘維凱
聖保羅書院舊生及老師

溯源

十多年前，作者之一受另一作者的鼓勵，透過 JUPAS 選取了香港大學的理學士課程，並主修數學；後來又在研究院師從王必弘博士進行過若干年數學研究，當中一些結果就有使用到計算機的輔助。回顧大學課程，結合編程的數學課堂並不鮮見，當中有強調程式語法及練習編程技巧的，也有使用虛擬代碼（pseudocode）並著重分析算法效能的。其實這是很自然的，畢竟計算機本來就是為了幫助人們進行運算而誕生。把運算部分交給計算機，我們就可以更集中地思考數學問題。

今年初，新冠變種病毒 Omicron 大流行，而學校須於三月至四月放特別假期。值此時機，作者二人皆希望以線上工作坊作一些常規課堂以外的創新嘗試。一番交談後，我們確立了一個線上 Python 工作坊的具體合作執行方案，主題為結合編程的數學課，對象為聖保羅書院的初中學生，工作坊分為四節各九十分鐘的 Zoom 課堂，內容涵蓋基本運算、編程及繪圖。分工方面，一位負責編寫教材，而另一位則主持工作坊，過程中的交流令雙方皆有所得著，教材製作與課堂教學相得益彰。工作坊最後吸引逾半百學生參與，我們謹此表示謝意。

工作坊大部分教材以 MIT 授權條款供同工使用，源代碼已發佈於 https://github.com/HanlunAI/NextGenCurriculumDemonstration，拋磚引玉，還望不吝指教。

起始

首先，我們需要建構一個可以運行 Python 的執行環境。傳統建構環境的方法需時甚長，當時學生分別使用自己在家的電腦亦會帶來很多不確定因素。於是，我們引進 Google Colaboratory（簡稱 Colab）到課堂，執行雲端預設的 Linux 環境，同學只要用合適的帳戶登入瀏覽器就可以開始了。

Colab 支持交互式 Python 筆記本的編程及執行（運行檔案類型為 .ipynb，意指 interactive Python notebook）。筆記本包括程式碼儲存格和文字儲存格，可以預先準備一些內容和代碼，然後透過 Zoom 的聊天室把筆記本的連結傳送給同學，就簡單地完成了分發。同學們只需要按圖 1 所示的按鈕就可以打開：

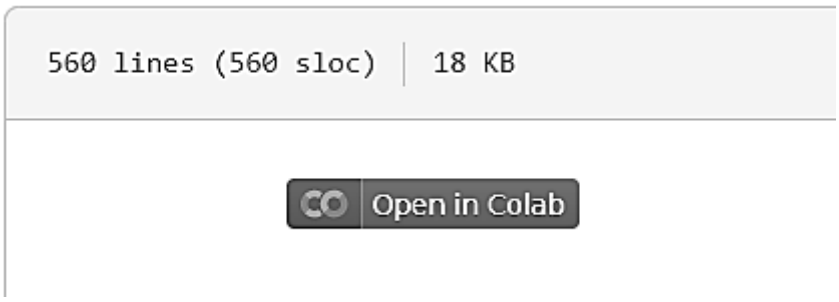

圖 1

同學們進入筆記本後，先要學習基本操作。程式碼儲存格的執行方法有幾種：或於鍵盤按快捷鍵「Shift+Enter」、或以鼠標點擊運行鍵、或在欄目表選取執行，以上皆可。百尺高台，起於累土，我們執行的第一行代碼就是輸出字串「Hello World!」，可見圖 2：

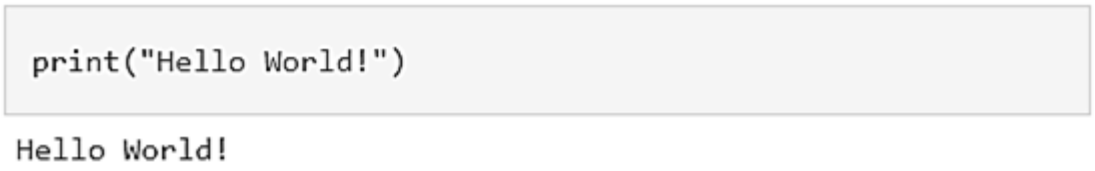

圖 2

然後，透過基礎計數原理的加法法則和乘法法則的練習，我們讓同學們嘗試了不同的數據類型，由字符串（string）到整數（integer）及浮點數（floating point）；由列表（list）到字典（dictionary）等。

工作坊又在簡單的題目引入一些變化，讓同學們嘗試不同的編程基礎。例如：透過輸入驗證（input validation）示範條件語句（if-else statement）的用法；透過列出乘法法則下的全部選項介紹 for 迴圈（for loop）；透過打包重複執行的程序介紹 Python 函數的概念。把一些重複進行的程序打包成 Python 功能函數，打包後的 Python 函數可以在以後引用，有需時也可以選

擇使用前人打包在模塊（module）或套件（package）中的函數。整個體驗旨在讓同學感受到 Colab 其實就是一台強大的計算機。

筆記本詳見上述 GitHub 資料庫中的 DemonstrationAlpha.ipynb。編程基礎難以盡錄，筆記本有一定留白，而更著重於鼓勵他們學會學習，懂得自行透過試驗或互聯網搜尋來查找答案。動手試錯是編程實踐過程中的重要一環，良好的習慣是使用草稿儲存格（scratch code cell）進行試驗以保持筆記本潔淨。

承接

緊接下來的筆記本是 DemonstrationBeta.ipynb，內容主線為排列的概念和記法。在定義階乘的時候，我們讓同學體會到完成一件事有很多不同做法，使用 for 迴圈固然可以，而我們亦示範了使用遞歸函數（recursive function）的方法，可見圖 3：

```
def factorial(n):
  if n == 0:
    return 1
  return n*factorial(n-1)
```

圖 3

這種定義階乘的方法涉及到初始化（initialization）及層遞關係（recursion），也算是向皮亞諾公理（Peano axioms）中的歸納法原理（induction principle）致敬。類似地，排列 P_r^n 也可以選擇使用 for 迴圈或遞歸函數來構作。不管那種做法，構作 Python 函數的驗證輸入過程，都有利於加強同學們對函數定義域的注意。

下一環節即展示出結合編程於數學課的巨大優勢。我們先讓同學各師各法，動手寫自己的代碼去列出 A、B、C 三個物件全部 3! = 6 種全排列。然後，再以遞歸函數介紹有效地列舉全部 $n!$ 的一般辦法：$n = 0$ 的初始化狀態很容易理解，後面的層遞關係其實只是在先前的 $(n - 1)!$ 種全排列各自製作 n 個副本，然後輾轉加入新的字母即可。

物件以英文字母作代號，姑且使用 A 至 H，使 n 不大於 8，代碼可見圖 4：

```
#@title We meant, checkout the result first.

#You need to make sure the list of tokens are distinct
token = ["A","B","C","D","E","F", "G", "H"]

#@markdown You may hide the code and simply use the slider to input n

#Allow n = 0,1,2,3,4,5,6,7,8
n = 6 #@param {type:"slider", min:0, max:8, step:1}
def switchingFullPermutation(n):
  global token
  if n == 0:
    return [""]
  previousList = switchingFullPermutation(n-1)
  #print(previousList)
  newChar = token[n-1]
  newList = []
  sign = -1
  for i in range(len(previousList)):
    insertNew = [previousList[i][0:j]+newChar+previousList[i][j:n] for j in range(n)]
    if sign == -1:
      insertNew.reverse()
    newList =  newList + insertNew
    sign *= -1
  return newList

def switchingFullPermutationPrint(n):
  print("The following is a list of {}! = {} distinct full permutations given by SJT algorithm
  count = 0
  longList = switchingFullPermutation(n)
  for item in longList:
    count += 1
    print(str(count)+".\t"+item, end = '\n')
  return

switchingFullPermutationPrint(n)
```

圖 4

工作坊上，我們使用程式切換幾個不太大的 n，然後印出結果，可喜的是參與的同學很快就看出層遞關係的規律了。（程式也容許同學在執行中打印更多過程，例如上面的代碼有一行以＃標記的注釋單行，只要去掉＃就可以打印一些過程。）若非使用程式，對於 $n > 5$，要把 $n!$ 項不同排法全部寫出就顯得費時失事且不切實際了。再者，使用上述程式生成全部排列法時，會發現更多有趣的規律：相鄰的兩種全排列所差的只是對調了某兩個字母。其實這就是 Steinhaus–Johnson–Trotter（SJT）算法能給出的排法。（註：給定一項，SJT 算法可以運算出這種排法的下一項，將來也許值得再擇一文詳細介紹。）

把 $n!$ 項全部排出其實有 $(n!)!$ 種排法，當中當然也有字典序排法（lexicographic order）。要打印出這種排法，由先前的結果出發也可，重寫代碼也可，直接使用 Python 套件也可。三種寫法皆可事先準備，然後在場老師就可以拿捏同學們當時興致所在，再作取捨了。

再接下來的筆記本是 DemonstrationGamma.ipynb。既然把編程融入於排列的教學取得不俗的效果，組合亦可如法炮製。由於我們主旨是數學教育，當編程技巧過重時，工作坊就會輕輕帶過，讓一般同學運行預先寫好的代碼，有興趣的同學將來也可自行探索。

我們先是學習計算組合 C_r^n，然後動手列出全部組合，繼而享受到觀察規律的便利。舉一例，圖 5 是輸出 $C_3^8 = 56$ 項不同組合後，再動手修整的結果：

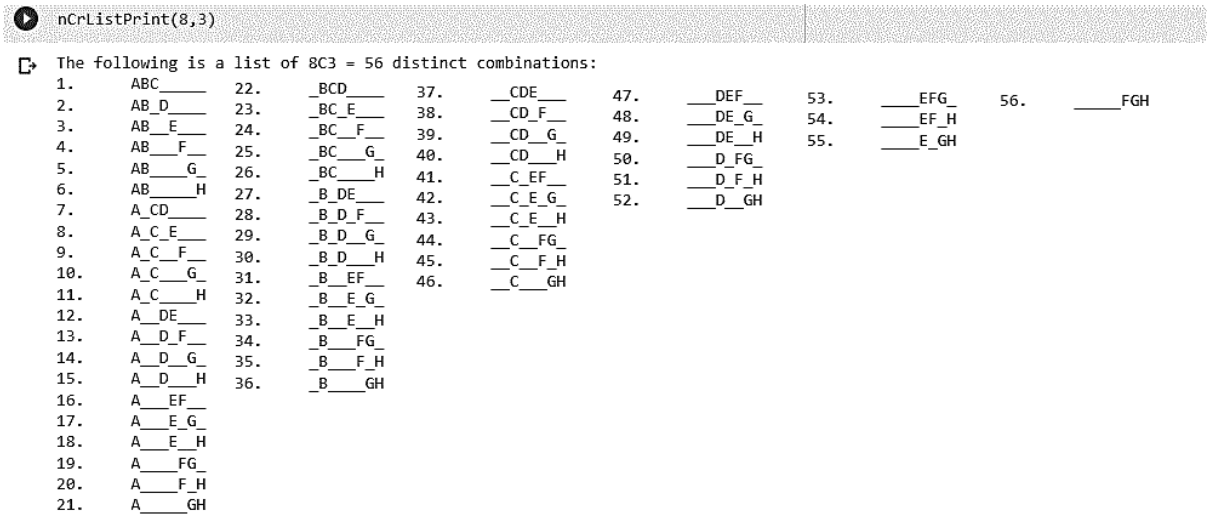

圖 5

很容易就可以觀察到一些規律，這些規律也給了好奇的同學一些方向去證明恆等式：對於 $1 \le r \le n$，$C_r^n = C_{r-1}^{n-1} + C_{r-1}^{n-2} + \cdots + C_{r-1}^{r-1}$。

筆記本有更多例子及應用。在此，我們僅再舉一例說明運算思維對學習數學有很大好處：眾所周知，楊輝三角形（Pascal's triangle）可以幫助我們找到二項展式中的係數，使其中第 i 行對應 $n = i - 1$ 時，展開 $(x+y)^n$ 所得的全部 $n+1$ 個係數，Python 代碼及結果可見圖 6；以運算思維推廣，可得類似的三角錐，以幫助我們找到三項展式中的係數，使其中第 i 行對應

$n = i - 1$ 時，展開 $(x + y + z)^n$ 所得的全部 $\frac{(n+1)(n+2)}{2}$ 個係數，Python 代碼及結果可見圖 7。

```
n = 3 #@param{type:"slider", min:0, max:30, step:1}

#an experiment on yield statement

def triangle():
  global n
  layer = [1]
  yield layer
  while len(layer) <= n:
    first = layer + [0]
    second = [0] + layer
    layer = [sum(binomial) for binomial in zip(first, second)]
    yield layer

for layer in triangle():
  print(layer)
print("The above is the Pascal triangle of order {}.\n\n".format(n))
```

```
[1]
[1, 1]
[1, 2, 1]
[1, 3, 3, 1]
The above is the Pascal triangle of order 3.
```

圖 6

```
n = 3 #@param{type:"slider", min:0, max:30, step:1}

#an experiment on yield statement

def pyramid():
  global n
  layer = [[1]]
  yield layer
  while len(layer) <= n:
    size = len(layer)+1
    first = layer + [[0 for i in range(size)]]
    second = [[0]] + [[0] + line for line in layer]
    third = [[0]] + [line + [0] for line in layer]
    layer = [[sum(trinomial) for trinomial in zip(first[i], second[i], third[i])] for i in range(size)]
    yield layer

for layer in pyramid():
  print(layer)
  #comment the previous line and uncomment the next line to see inverted triangle
  #print(layer[::-1])
print("The above is an analogous pyramid of order {}.".format(n))
```

```
[[1]]
[[1], [1, 1]]
[[1], [2, 2], [1, 2, 1]]
[[1], [3, 3], [3, 6, 3], [1, 3, 3, 1]]
The above is an analogous pyramid of order 3.
```

圖 7

轉換

下一個筆記本是 DemonstrationDelta.ipynb，透過圖像轉換的課題，討論 Python 繪圖。筆記本刻意選取了一種 S 型函數 $F(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{t^2}{2}}dt$ 。這是一種累積分佈函數（cumulative distribution function），滿足以下三項特性：

（I） 這是一個單調遞增函數（monotonic increasing function）；

（II） 當 x 趨向無窮大時，$F(x)$ 趨向 1；

（III）當 x 趨向無窮小時，$F(x)$ 趨向 0。

另外，累積分佈函數還滿足右連續性，考慮到受眾為初中生，工作坊就沒有細表了。重點在於介紹三個常用 Python 套件來進行繪圖：NumPy 支援列陣運算，便於準備繪圖所需的大量數據點；SciPy 提供特殊函數；Matplotlib 就為這些數據點提供可視化工具。

中學課程會學習四種圖像變換：$f(x)+k$、$kf(x)$、$f(x-k)$ 及 $f\left(\frac{x}{k}\right)$。有了便利的繪圖工具，我們設計了活動讓學生透過編輯一些代碼，以變換不同的參數，並觀看函數圖像變換後就會變成另一的圖像。敏銳的學生更有望指出四種圖像變換中，後兩種會保留特性（I），（II），（III）。

其實 $F(x)$ 就是標準正態分佈對應的累積分佈函數， $F(x-k)$ 就是改變平均值後正態分佈所對應的累積分佈函數，而 $F\left(\frac{x}{k}\right)$ 就是改變方差後正態分佈所對應的累積分佈函數。

如圖 8 所示，我們亦在筆記本製作了互動素材，讓同學可以拉動滑杆觀看正態分佈的累積分佈函數及其概率密度函數（probability density function）的變化。

至於累積分佈函數及概率密度函數兩者之間的關係，在適當條件下，可以用微積分基本定理（fundamental theorem of calculus）準確描述。一般而言，這是在較後期才會涉獵的課題，現在透過圖像示意，同學也就較容易得到直觀感受了。

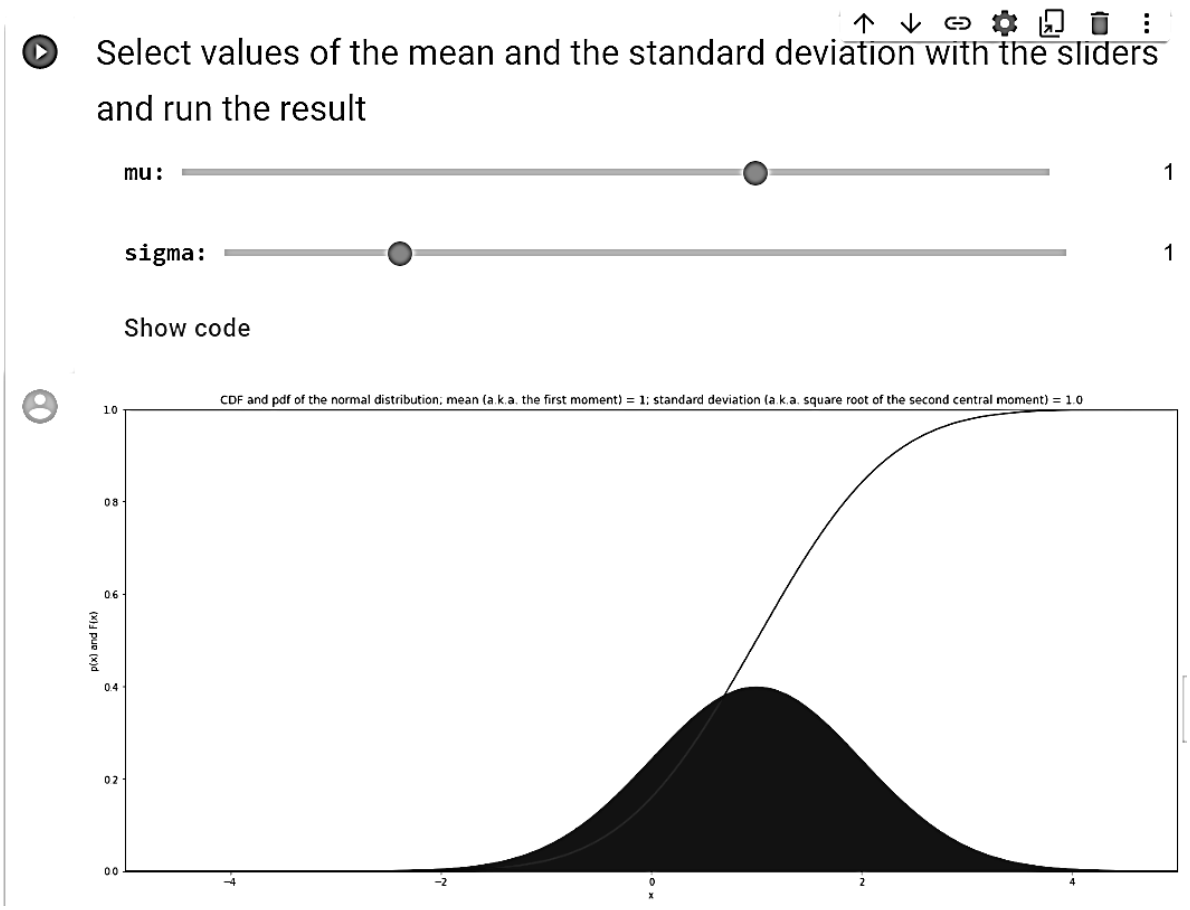

圖 8

適逢文憑試十週年，略作小議。十年前，作者之一在香港大學擔任助教，親身見證到修讀數學延伸部分單元一／二的同學明顯較未修讀的同學更能適應大學課堂。香港諸多大學亦早已在收生之時將其成績當作一科，有些學系甚至再增加其權重。時至今日，文憑試尚未為此兩科正名，實在可惜。所謂名不正則言不順，分配給此兩科的資源也就不落大方，學生選科時有所猶豫亦在所難免了。又有所謂名正則言順，結合上句可以推論名之正與言之順存在等價關係。由此想像，有關方面若能為其正名，自當有一番新氣象。

合奏

我們希望同學可以持續發展這種創新學習模式，所以刻意把此系列中最後的筆記本 DemonstrationEpsilon.ipynb 交給同學在工作坊完結後自行探索，當中內容會介紹二項分佈，然後再透過中央極限定理（central limit theorem）把將其與正態分佈連繫起來。

而工作坊最後一節課可謂百花齊放。談笑間，師生以代碼會友，各自就有興趣的課題發揮創意，融入編程於數學。當中有互動遊戲、繪製五角星及極坐標系中的常見曲線、以至數論中的一些難題等，在取得創作者的同意後，將來亦當逐步公諸同好，發佈於 GitHub 資料庫。

謹此示範報告，我們亦誠邀同工，一起推動前瞻性的試驗，讓下一代創造不一樣的可能。

首作者電郵：yktai@hanlunai.com