w10_ex

-from browser import document, window： 引入 Brython 提供的模組，用於操作 DOM 和取得瀏覽器環境的資訊。

-window.location.href： 獲取當前瀏覽器頁面的完整 URL。 urlparse 和

-parse_qs： 用來解析 URL 並提取查詢參數（如 ?page_title=... 中的值）。

-page_title： 從查詢參數中獲取 page_title 值，預設為 'w10'（如果 URL 中沒有 page_title 參數）。

-url = "https://mde.tw/list/1a.txt"：指定一個遠端的.txt 檔案作為數據來源。

-open(url).read()： 嘗試開啟遠端檔案並讀取其內容。

-line_data = data.split("\n")[1:]： 將檔案內容按行拆分為一個列表，並跳過第一行（通常第一行是表頭或說明文字）。

-div = document.getElementById("brython_div1")： 找到頁面上 ID 為 -brython_div1 的 容器，用於放置後續生成的連結。

-for line in line_data[:-1]： 遍歷檔案中的每一行數據，跳過最後一行（通常是空行）。

-line.split("\t")： 根據制表符（\t）拆分每行數據，獲取學生資料。

-stud_num 和 github： 從拆分後的數據中提取學生學號和 GitHub 帳號。

-page_url 和 main_url： 構建學生作業的具體頁面和主頁 URL。

try 和 except： 嘗試訪問 page_url（作業頁面），如果失敗（如該頁面不存在），則進入例外處理邏輯。

-open(page_url).read()： 模擬訪問 URL，如果能讀取到內容，說明目標頁面存在。

-document.createElement("a")： 動態創建一個 <a> 超連結元素。

-a.href 和 a.text： 設定連結的目標 URL 和顯示文字。 如果目標頁面存在，顯示學生學號與頁面名稱；否則僅顯示學號並指向主頁。

-a.style.marginRight = "10px"： 設定超連結之間的間距以美化版面。

-link_div = document.createElement("div")： 為每個超連結創建一個單獨的 容器。

-link_div.appendChild(a)： 將剛創建的超連結元素添加到該容器中。

-div.appendChild(link_div)： 將每個包含超連結的容器添加到主容器（brython_div1）。

w11_hw

-For：迴圈語法，用來遍歷序列中的每個值。

-in range(n)：生成從 0 到 n-1 的整數序列。

-n：變數。

-center_x = n // 2 和 center_y = n // 2：定義圓心。

-radius = n：定義圓的大小(半徑)。

-distance = (n-x)：計算 n 與 x 之距離。

-if distance <= radius:：判斷 distance 是否小於或等於 radius。

-print()：是用來輸出字串或字元的函式。(不帶參數的 print()表示輸出換行符。)

w12_hw

-for i in range(1, 6)：生成從 1 到 5 的整數序列（不包括 6）。

-num = i：將迴圈變數 i 的值賦給變數 num。

-print(num)：輸出變數 num 的值。

w13_hw

-from browser import html：從 Brython 的 browser 模組中匯入 html，用來操作 HTML 元素。

-from browser import document as doc：將 HTML 文件（DOM 結構）作為 doc，方便操作頁面中的元素。

-import math：匯入 Python 的數學模組，用於數學運算，例如平方根和圓周率計算。

-html.CANVAS：建立一個 HTML <canvas> 元素，寬高為 500x500 像素。

-doc["brython_div1"]：選取 HTML 中 ID 為 brython_div1 的元素。

-brython_div <= canvas：將 canvas 元素新增到 brython_div1 內。

-canvas.getContext("2d")：取得 Canvas 的 2D 繪圖上下文，用於繪製圖形。

-ctx.lineWidth = 2：設定繪製的線條寬度為 2 像素。

-square_size：定義正方形的邊長為 200 像素。

-offset：控制正方形的垂直位置偏移量。

-x_offset：控制正方形的水平位置偏移量。

-square_center：計算第一個正方形的中心點座標。

-circle_radius：計算圓的半徑

-fill_shape：通用函數，用來填充特定區域的顏色

-ctx.save()：保存當前的畫布狀態。

-ctx.beginPath()：開始繪製新路徑。

-exclusion_callback：定義排除區域的路徑。

-path_callback：定義要填充的主要區域。

-ctx.clip()：將繪圖限制在排除區域外。

-ctx.fillStyle = color：設定填充顏色。

-ctx.fill()：填充指定路徑的區域。

-ctx.restore()：恢復先前保存的畫布狀態。

-ctx.rect(x, y, width, height)：繪製一個矩形，指定左上角座標 (x, y) 及寬高。

-ctx.arc(x, y, radius, startAngle, endAngle)：繪製一個圓弧或圓。

-ctx.strokeStyle = 'red'：設定描邊顏色為紅色

-ctx.stroke()：描繪路徑的邊框

-ctx.moveTo(x, y)：設定起始點座標。

-ctx.lineTo(x, y)：繪製一條線到指定座標。

w14_ex

-這是頁面中的一個 元素，id="brython_div1"。後續會在這個容器中顯示畫布。

-<script src="/static/brython.js"></script> 和 <script src="/static/brython_stdlib.js"></script>：這兩行引入了 Brython 的核心庫和標準庫，讓瀏覽器可以執行 Python 程式碼。

-window.onload：這行程式碼確保在網頁完全加載後執行函數。

-brython()：這個函數啟動 Brython 執行環境。debug:1 開啟除錯模式，pythonpath 設定 Python 模組的路徑。

-random_color_generator()：這個函數生成一個隨機的顏色。

-is_point_in_circle(px, py, cx, cy, r)：此函數用來判斷一個點 (px, py) 是否位於圓心 (cx, cy) 和半徑 r 的圓內。

-ensure_no_isolated_circles(circles)：此函數檢查給定的圓是否互相重疊。如果兩個圓之間的距離小於或等於它們的半徑和，則視為重疊。

-draw_circles()：此函數繪製隨機生成的圓形並使用掃描線填充顏色。

w15_ex

-sum：儲存累加的結果。

-init：定義累加的起始值。

-upto：定義累加的結束值。

-def name(start, end)：def 為定義一個函式(函式名稱 name)，並且接受 2 個參數 start 和 end。

-total：變數，用於儲存數字累加後的結果。

-for：為 python 的迴圈結構，用於遍歷序列或範圍

-range(start, end+1)：函式生成從 start 到 end 的整數序列，end+1 是因為 range 的結束值不包含 end。

-If：條件判斷，用於判斷條件是否成立，通常會搭配 else(否則)做使用。

str(i)：將數字轉化為字串型別，方便檢查是否包含特定字元。

-return：返回值，用於將函式執行結果返回給呼叫函式的地方，返回的值可以被儲存到變數中或直接用於其他運算。

- f-string：字串格式化，用於將變數或表達式嵌到字串中，基本語法為

f”文字{變數或表達式}文字”{}用於插入變數或表達式，例:{1}為直接插入數字 1、{return}為插入變數 return 得值。

w16_exam1

-ctx.font = “大小 字體”：為設定字體的種類以及大小。

-ctx.fillText(“word”, x, y)：在(x, y)處繪製文字”word”。

-ctx.stroke()：描繪路徑的邊框

-ctx.moveTo(x, y)：設定起始點座標。

-ctx.lineTo(x, y)：繪製一條線到指定座標。

-ctx.fill()：填充指定路徑的區域。

-ctx.arc(x, y, radius, startAngle, endAngle)：繪製一個圓弧或圓。

w16_exam2

-ctx = canvas.getContext("2d") 獲取 canvas 的 2D 繪圖上下文 (context)。 所有後續繪圖操作都會基於這個上下文對象進行。

-line_color = "blue"：設定線條顏色為藍色。

-line_width = 2：設定線條寬度為 2 像素。

-ctx.strokeStyle = line_color：將藍色應用為線條的顏色樣式。

-ctx.lineWidth = line_width：將線條寬度設置為 2。 你傳送了 lines 是一個列表，每個項目是一條線的座標，由起點 (x1, y1) 和終點 (x2, y2) 組成。 你傳送了

for line in lines:：逐一遍歷 lines 中的每條線。

(x1, y1), (x2, y2) = line：解構每條線的起點和終點座標。

-ctx.beginPath()：開始繪製新路徑。

-ctx.moveTo(x1, y1)：將畫筆移動到起點 (x1, y1)。

-ctx.lineTo(x2, y2)：從起點畫到終點 (x2, y2)。

-ctx.stroke()：使用藍色、2 像素的樣式繪製該線段。 你傳送了

-ctx.fillStyle = "red"：將填充文字的顏色設置為紅色。

-ctx.font = "10px Arial"：設定字體大小為 10px，字型為 Arial。

-ctx.fillText("原點", 31, 10)：在座標 (31, 10) 處繪製文字 "原點"，用來標示迷宮的原始起點位置。

w16_exam3

-line_color = "blue"：設定線條顏色為藍色。

-line_width = 2：設定線條寬度為 2 像素。

-ctx.strokeStyle = line_color：將藍色應用為線條的顏色樣式。

-ctx.lineWidth = line_width：將線條寬度設置為 2。

-line_color = "red" 作用：定義一個變數

-line_color，並將其值設為 "red"，表示線條的顏色為紅色

-ctx.beginPath()： 開啟一個新的繪圖路徑，確保不影響之前的繪圖結果。

-ctx.moveTo(131, 10)： 將畫筆移動到新紅色線條的起點 (131, 10)。

-ctx.lineTo(131, 30)： 繪製一條從 (131, 10) 到 (131, 30) 的直線。

-ctx.stroke()： 使用設定好的紅色線條樣式將這條新線條顯示在畫布上。 你傳送了

-ctx.strokeStyle = line_color：將 canvas 繪圖上下文的 描邊顏色（strokeStyle）設置為剛才定義的紅色。這將影響後續所有描邊的顏色。 你傳送了

-ctx.lineWidth = 2：設定線條的寬度為 2 像素。這將影響後續繪製的所有線條。 你傳送了

-ctx.beginPath() 作用：啟動一條新的繪圖路徑。這是為了保證不將新繪製的線條與之前的繪圖操作混在一起。

-ctx.moveTo(131, 10)：將畫筆移動到座標 (131, 10)，這是紅色線條的起點。 具體效果：這一步不會在畫布上繪製任何東西，只是設定起始點。 你傳送了

-ctx.lineTo(131, 30) 作用：從剛才的起點 (131, 10) 繪製一條直線到 (131, 30)，這是紅色線條的終點。 具體效果：這一步只設定路徑，實際線條的繪製由後面的 ctx.stroke() 完成