31 繪製同心圓：

```
from browser import document as doc

from browser import html


# 輸入圓的數量和半徑差距

cs = int(input("請輸入圓的數量: "))

rs = int(input("請輸入每個圓的半徑差: "))


# 建立畫布

canvas = html.CANVAS(width=cs * rs * 2 + 20, height=cs * rs * 2 + 20)

brython_div = doc["brython_div1"]

brython_div <= canvas


# 獲取繪圖上下文

ctx = canvas.getContext("2d")


# 設定圓心

x, y = cs * rs + 10, cs * rs + 10


# 繪製同心圓

for i in range(cs):

    radius = rs * (i + 1)    # 計算每個圓的半徑

    ctx.beginPath()

    ctx.arc(x, y, radius, 0, 2 * 3.14159)    # 繪製圓
```

```
ctx.strokeStyle = "black"    # 設定邊框顏色
ctx.lineWidth = 2    # 邊框寬度
ctx.stroke()    # 繪製邊框
```

輸入圓的數量和半徑差距：

- cs 是圓的數量，rs 是每個圓的半徑增量（每個圓與上一個圓的半徑差異）。

建立畫布並添加到頁面：

- 使用 html.CANVAS() 創建了一個畫布，並將其插入到 brython_div1 的 div 容器中。

設定圓心：

- center_x, center_y 用來設置圓心的位置，這裡選擇了畫布的中心位置。

繪製同心圓：

- 在 for 迴圈中，根據圓的數量 cs，每次計算圓的半徑，並繪製出圓。
- 每個圓的半徑依據 rs * (i + 1) 計算，這樣圓的半徑會依次增大。
- ctx.beginPath() 開始一個新的路徑。
- ctx.arc() 用來繪製圓，傳入圓心和半徑。
- ctx.strokeStyle 設定圓邊的顏色為黑色，並且設定邊框的寬度為 2 像素。
- ctx.stroke() 會將圓的邊框畫出來。

32 棋盤格圖案：

```
from browser import document as doc

from browser import html


# 棋盤格的行列數

rows, cols = 8, 8


# 每個格子的大小

square_size = 50


# 建立畫布

canvas = html.CANVAS(width=cols * square_size, height=rows * square_size)

brython_div = doc["brython_div1"]

brython_div <= canvas


# 獲取繪圖上下文

ctx = canvas.getContext("2d")


# 使用 for 迴圈來繪製棋盤格

for row in range(rows):

    for col in range(cols):

        # 判斷顏色：若行列之和為偶數則為白色，否則為黑色

        if (row + col) % 2 == 0:

            ctx.fillStyle = "white"
```

```
        else:
            ctx.fillStyle = "black"

        # 計算每個格子的左上角位置
        x = col * square_size
        y = row * square_size

        # 繪製正方形
        ctx.fillRect(x, y, square_size, square_size)
```

棋盤大小：

- rows 和 cols 分別設定為 8，表示棋盤的行數和列數。
- square_size 定義每個格子的大小，這裡設為 50 像素。

畫布大小：

- 畫布的寬度和高度分別是 cols * square_size 和 rows * square_size，這樣確保棋盤會填滿整個畫布。

顏色判斷：

- 透過 (row + col) % 2 來決定每個格子的顏色。如果行數和列數的和是偶數，就設置為白色 (fillStyle = "white")，如果是奇數，則設置為黑色 (fillStyle = "black")。

繪製格子：

- ctx.fillRect(x, y, square_size, square_size) 繪製每個格子，其中 x 和 y 是格子左上角的座標。

% 2 是取模運算，表示對 2 取餘數。這樣會得到兩種結果：

- 如果 (row + col) 是偶數，則 (row + col) % 2 的結果是 0

33 螺旋線條：

```
from browser import document as doc

from browser import html

import math


# 螺旋的參數設定

length_increase = 5    # 每次增加的長度

angle_increase = 10    # 每次增加的角度（度數）

num_steps = 100    # 迴圈次數（畫的線段數量）


# 建立畫布

canvas = html.CANVAS(width=400, height=400)

brython_div = doc["brython_div1"]

brython_div <= canvas


# 獲取繪圖上下文

ctx = canvas.getContext("2d")


# 設定起始點為畫布的中心

center_x, center_y = 200, 200


# 設定初始角度和長度
```

```
current_angle = 0

current_length = 5


# 使用 for 迴圈來畫螺旋線條

for i in range(num_steps):

    # 計算新的終點位置

    end_x = center_x + current_length * math.cos(math.radians(current_angle))

    end_y = center_y + current_length * math.sin(math.radians(current_angle))


    # 畫線條

    ctx.beginPath()

    ctx.moveTo(center_x, center_y)  # 起點

    ctx.lineTo(end_x, end_y)  # 終點

    ctx.strokeStyle = "black"  # 設定顏色為黑色

    ctx.lineWidth = 1  # 設定線條寬度

    ctx.stroke()  # 畫線


    # 更新起點為當前終點，並增加長度和角度

    center_x, center_y = end_x, end_y

    current_length += length_increase  # 長度增加

    current_angle += angle_increase  # 角度增加
```

- length_increase = 5：每次迴圈中，線條的長度增加 5 像素。
- angle_increase = 10：每次迴圈中，角度增加 10 度。

- num_steps = 100：這是迴圈的次數，表示總共畫多少段線條，這裡設定為 100。

畫布設定：

- 畫布的尺寸是 400x400 像素，這樣可以提供足夠的空間來繪製螺旋。

起始點和角度設定：

- 起始點是畫布的中心，設定為 center_x = 200, center_y = 200。
- 初始角度 current_angle 設為 0 度，這是從水平線開始。
- 初始線條長度 current_length 設為 5 像素。

螺旋繪製：

- 使用 math.cos 和 math.sin 計算每段線條的終點座標，這些座標是基於當前角度和長度來計算的。
- 每次繪製一段線條，將畫布的起點更新為當前終點，並增加長度和角度。
- math.radians 將角度轉換為弧度，因為 cos 和 sin 函數需要弧度作為輸入

34 多邊形繪製：

```
from browser import document as doc

from browser import html

import math


# 設定畫布大小

canvas_width = 820

canvas_height = 200


# 設定每個多邊形的半徑

radius = 50


# 設定起始位置

center_x, center_y = 50, 100


# 設定畫布

canvas = html.CANVAS(width=canvas_width, height=canvas_height)

brython_div = doc["brython_div1"]

brython_div <= canvas


# 獲取繪圖上下文

ctx = canvas.getContext("2d")


# 使用 for 迴圈繪製從 3 到 10 邊的正多邊形
```

```
for sides in range(3, 11):  # 從 3 邊到 10 邊
    # 計算每個頂點的角度
    angle = 2 * math.pi / sides

    # 開始繪製多邊形
    ctx.beginPath()

    for i in range(sides):
        # 計算每個頂點的座標
        x = center_x + radius * math.cos(i * angle)
        y = center_y + radius * math.sin(i * angle)

        if i == 0:
            # 設置起始點
            ctx.moveTo(x, y)
        else:
            # 繪製線條到下一個頂點
            ctx.lineTo(x, y)

    # 關閉路徑，完成多邊形
    ctx.closePath()

    # 設定顏色並繪製多邊形
    ctx.strokeStyle = "black"  # 設定邊框顏色
```

ctx.lineWidth = 2 # 設定邊框寬度

ctx.stroke() # 繪製邊框

移動中心位置，以避免多邊形重疊

center_x += 100 # 偏移一點以繪製下一個多邊形

多邊形邊數：

- num_sides = [3, 4, 5, 6, 7] 表示我們要繪製的多邊形的邊數，包括三角形（3 邊）、正方形（4 邊）、五邊形（5 邊）等。

半徑設定：

- radius = 50 定義了每個多邊形的半徑，即從中心到每個頂點的距離。

頂點計算：

- 每個多邊形的頂點都是根據邊數和半徑來計算的。我們使用三角函數來計算每個頂點的 (x, y) 座標：
    - x = center_x + radius * math.cos(i * angle)：根據角度和半徑計算頂點的 x 座標。
    - y = center_y + radius * math.sin(i * angle)：根據角度和半徑計算頂點的 y 座標。
- 角度 angle 是每個角的角度，計算公式是 2 * math.pi / sides，即 360 度除以邊數。

繪製多邊形：

- 使用 ctx.moveTo(x, y) 設定起點，並使用 ctx.lineTo(x, y) 繪製每個邊。
- ctx.closePath() 用來關閉路徑，完成多邊形的繪製。

顏色設定和偏移：

- 使用 ctx.strokeStyle = "black" 設定邊框顏色，並使用 ctx.lineWidth = 2 設定邊框寬度。
- center_x += 100 用來移動畫布的中心點位置，這樣繪製的多邊形就不會重疊。

邊數範圍：

- 使用 for sides in range(3, 11)，這樣迴圈會從 3 邊（即三角形）開始，一直到 10 邊（即十邊形）。

頂點計算：

- 每個多邊形的頂點是根據其邊數和半徑來計算的，使用三角函數計算每個頂點的 (x, y) 座標。
- 角度 angle 是每個角的角度，計算公式是 2 * math.pi / sides。

繪製多邊形：

- 使用 ctx.moveTo(x, y) 設定起點，並使用 ctx.lineTo(x, y) 繪製每個邊。
- ctx.closePath() 用來關閉路徑，完成多邊形的繪製。

偏移：

- center_x += 120 用來移動每個多邊形的中心，這樣每個多邊形就不會重疊。

35 隨機顏色的圓形：

```
from browser import document as doc

from browser import html

import random


# 設定畫布大小

canvas_width = 500

canvas_height = 500


# 設定圓形的半徑範圍

min_radius = 10

max_radius = 50


# 設定要繪製的圓形數量

num_circles =    int(input("請輸入圓的數量: "))


# 設定畫布

canvas = html.CANVAS(width=canvas_width, height=canvas_height)

brython_div = doc["brython_div1"]

brython_div <= canvas


# 獲取繪圖上下文

ctx = canvas.getContext("2d")
```

```
# 隨機顏色生成函數
def random_color():
    return f"rgb({random.randint(0, 255)}, {random.randint(0, 255)}, {random.randint(0, 255)})"

# 使用 for 迴圈繪製隨機圓形
for _ in range(num_circles):
    # 隨機生成圓心位置
    center_x = random.randint(50, 450)
    center_y = random.randint(50, 450)

    # 隨機生成圓形的半徑
    radius = random.randint(min_radius, max_radius)

    # 隨機生成顏色
    color = random_color()

    # 設定圓形顏色並繪製
    ctx.beginPath()
    ctx.arc(center_x, center_y, radius, 0, 2 * 3.14159)  # 畫圓
    ctx.fillStyle = color  # 設定圓形顏色
    ctx.fill()  # 填充顏色
    ctx.stroke()  # 畫邊框
```

---

1. **隨機顏色生成**：
    - random_color() 函數用來隨機生成 RGB 顏色，每個顏色分量（紅、綠、藍）都是隨機選擇的，範圍是從 0 到 255。
2. **隨機位置與大小**：
    - center_x 和 center_y 使用 random.randint(0, canvas_width) 和 random.randint(0, canvas_height) 隨機生成，這樣圓形的位置會隨機分佈在畫布內。
    - radius 使用 random.randint(min_radius, max_radius) 隨機生成圓形的半徑，範圍設為 10 到 50 像素。
3. **繪製圓形**：
    - ctx.arc(center_x, center_y, radius, 0, 2 * 3.14159) 用來畫圓，這個方法會根據隨機生成的位置和半徑來畫出圓形。
    - ctx.fillStyle = color 設定填充顏色，ctx.fill() 填充圓形，ctx.stroke() 畫圓形邊框。
4. **繪製多個圓形**：
    - 使用 for _ in range(num_circles) 來繪製多個圓形，每次循環生成一個新的隨機圓形。

36 雪花圖案：

```
from browser import document as doc

from browser import html

import math


# 設定畫布大小

canvas_width = 500

canvas_height = 500

num = int(input("請輸入枝條數量: "))


# 設定畫布

canvas = html.CANVAS(width=canvas_width, height=canvas_height)

brython_div = doc["brython_div1"]

brython_div <= canvas


# 獲取繪圖上下文

ctx = canvas.getContext("2d")


# 設定畫布的中心點

center_x, center_y = canvas_width / 2, canvas_height / 2


# 設定直線的長度

line_length = 100
```

```
# 使用 for 迴圈繪製六條對稱的直線，形成雪花圖案
for i in range(num):
    # 計算每條線的角度，角度間隔 60 度
    angle = i * 2 * math.pi / num    # 360°除以 num，等於 60°

    # 計算直線的結束點 (x, y)
    x = center_x + line_length * math.cos(angle)
    y = center_y + line_length * math.sin(angle)

    # 開始繪製直線
    ctx.beginPath()
    ctx.moveTo(center_x, center_y)    # 起點為中心
    ctx.lineTo(x, y)    # 終點為計算得到的 (x, y)
    ctx.strokeStyle = "black"    # 設定顏色
    ctx.lineWidth = 2    # 設定線條寬度
    ctx.stroke()    # 繪製直線
```

畫布的設定：

- 畫布大小設為 500x500 像素，並且中心點位置 (center_x, center_y) 設為畫布的中間。

直線的長度：

- line_length = 100 設定每條直線的長度。這樣每條直線將從畫布的中心延伸出 100 像素。

計算角度：

- 使用 i * 2 * math.pi / 6 來計算每條直線的角度。由於雪花是六邊對稱

的，每條直線之間的角度為 60 度，因此每次迴圈的角度增量為 2 * math.pi / 6，即 60 度。

繪製直線：

- ctx.moveTo(center_x, center_y) 用來設置每條直線的起點為畫布的中心點。
- ctx.lineTo(x, y) 用來設置每條直線的終點，這個終點是通過三角函數計算得到的。

繪製結果：

- 每次循環都會繪製一條從中心點向外延伸的直線，最後形成六條對稱的直線，從而構成簡單的雪花圖案。

37 旋轉方形：

```
from browser import document as doc

from browser import html

import math


# 設定畫布大小

canvas_width = 500

canvas_height = 500


an = int(input("請輸入角度"))


# 設定畫布

canvas = html.CANVAS(width=canvas_width, height=canvas_height)

brython_div = doc["brython_div1"]

brython_div <= canvas


# 獲取繪圖上下文

ctx = canvas.getContext("2d")


# 設定方形的邊長

side_length = 100


# 設定畫布的中心點

center_x, center_y = canvas_width / 2, canvas_height / 2
```

```
# 設定繪製方形的次數
num_squares = 12

# 使用 for 迴圈繪製旋轉的方形
for i in range(num_squares):
    # 計算每個方形的旋轉角度
    angle = i * an * math.pi / 180  # 每次增加 angle 度 (轉換為弧度)

    # 保存當前繪圖狀態
    ctx.save()

    # 移動到方形的中心
    ctx.translate(center_x, center_y)

    # 旋轉畫布
    ctx.rotate(angle)

    # 設定方形的起始位置
    ctx.beginPath()
    ctx.rect(-side_length / 2, -side_length / 2, side_length, side_length)  # 繪製方形
    ctx.strokeStyle = "black"  # 設定邊框顏色
    ctx.lineWidth = 2  # 設定邊框寬度
```

```
ctx.stroke()  # 繪製邊框


# 恢復繪圖狀態
ctx.restore()
```

畫布設定：

- 畫布大小設為 500x500 像素，並將中心點設為畫布的中間 (center_x, center_y)。

方形邊長：

- 每個方形的邊長設定為 100 像素，並且我們將會繪製多個方形，每個方形的中心都位於畫布中心。

旋轉角度：

- 每次迴圈，我們都將角度增加 15 度，因此在 for 迴圈中，角度會依次是 0、15、30、45、...，直到 180 度。這個角度會轉換為弧度，因為畫布的 rotate() 方法需要的是弧度。

繪製方形：

- 使用 ctx.translate(center_x, center_y) 來將畫布的繪圖原點移到畫布的中心。
- 使用 ctx.rotate(angle) 來旋轉畫布，旋轉角度隨著每次迴圈的增加而增大。
- 使用 ctx.rect() 繪製方形，並且將方形的中心設置在畫布中心。

保存與恢復繪圖狀態：

- 每次繪製方形之前，我們使用 ctx.save() 保存當前的繪圖狀態，這樣旋轉和移動的變化不會影響到後續的繪圖。
- 繪製完成後，使用 ctx.restore() 恢復繪圖狀態，這樣每次迴圈都從相同的原始狀態開始。

38 漸變色矩形：

```
from browser import document as doc

from browser import html


# 設定畫布大小

canvas_width = 500

canvas_height = 500


# 設定畫布

canvas = html.CANVAS(width=canvas_width, height=canvas_height)

brython_div = doc["brython_div1"]

brython_div <= canvas


# 獲取繪圖上下文

ctx = canvas.getContext("2d")


# 設定矩形數量

num_rectangles = 20


# 設定每個矩形的寬高

rect_width = 20

rect_height = canvas_height / num_rectangles


# 使用 for 迴圈繪製漸變色矩形
```

```
for i in range(num_rectangles):
    # 計算漸變顏色
    r = 255 - int((255 / num_rectangles) * i)  # 紅色分量逐漸減少
    g = 0  # 綠色保持為 0
    b = int((255 / num_rectangles) * i)  # 藍色分量逐漸增加

    color = f"rgb({r}, {g}, {b})"  # 組合成 RGB 顏色

    # 設定矩形位置
    x = i * rect_width  # 每個矩形沿 x 軸排列
    y = 0  # 矩形從畫布的頂部開始

    # 繪製矩形
    ctx.beginPath()
    ctx.rect(x, y, rect_width, canvas_height)  # 繪製矩形
    ctx.fillStyle = color  # 設定填充顏色
    ctx.fill()  # 填充矩形
```

畫布設定：

- 畫布大小設為 500x500 像素，並將矩形數量設為 20。

漸變顏色計算：

- 每個矩形的顏色是基於紅色 (r) 和藍色 (b) 的變化來計算的：
    - 紅色從 255 開始逐漸減少到 0。
    - 藍色從 0 開始逐漸增加到 255。

    - 綠色 (g) 始終為 0。
- 這樣可以實現從紅色到藍色的顏色漸變。

矩形的位置與大小：

- 每個矩形的寬度是 20 像素，並且沿著 x 軸排列，x = i * rect_width 確保矩形不重疊。
- 矩形高度設為畫布的高度，這樣矩形會占滿畫布的垂直空間。

繪製矩形：

- 使用 ctx.rect() 繪製矩形，並用 ctx.fillStyle 設置顏色，ctx.fill() 填充矩形。

39 星星圖案：

```
from browser import document as doc

from browser import html

import random

import math


# 設定畫布大小

canvas_width = 500

canvas_height = 500


# 設定畫布

canvas = html.CANVAS(width=canvas_width, height=canvas_height)

brython_div = doc["brython_div1"]

brython_div <= canvas


# 獲取繪圖上下文

ctx = canvas.getContext("2d")


# 繪製五角星函數

def draw_star(ctx, x, y, radius, color, rotation_angle):

    ctx.beginPath()

    points = 5    # 五角星

    outer_radius = radius

    inner_radius = radius / 2    # 內部頂點半徑
```

```
    # 計算五角星的每個點，並應用隨機旋轉角度
    for i in range(points * 2):
        angle = (math.pi / points) * i + rotation_angle  # 加入旋轉角度
        r = outer_radius if i % 2 == 0 else inner_radius
        px = x + r * math.cos(angle)
        py = y - r * math.sin(angle)  # Y 軸向下為正，需取負數
        if i == 0:
            ctx.moveTo(px, py)
        else:
            ctx.lineTo(px, py)

    ctx.closePath()
    ctx.fillStyle = color
    ctx.fill()

# 繪製星星數量
num_stars = 30

# 使用 for 迴圈繪製星星
for _ in range(num_stars):
    # 隨機生成星星的位置和大小
    x = random.randint(50, canvas_width - 50)  # 避免星星超出畫布邊界
    y = random.randint(50, canvas_height - 50)
```

radius = random.randint(10, 30) # 星星大小

color = f"rgb({random.randint(150, 255)}, {random.randint(150, 255)}, {random.randint(150, 255)})" # 隨機亮色

隨機生成旋轉角度

rotation_angle = random.uniform(0, 2 * math.pi) # 旋轉角度，範圍 0 到 2 π

繪製星星

draw_star(ctx, x, y, radius, color, rotation_angle)

五角星繪製邏輯：

- 使用幾何計算來定位五角星的每個頂點：
    - 角度公式：angle = (math.pi / points) * i，其中 points = 5 表示五角星有 5 個頂點，每個頂點間隔 72 度。
    - 半徑切換：外頂點使用 outer_radius，內頂點使用 inner_radius，交替形成星形。

隨機化設計：

- 位置 (x, y)：隨機選擇，範圍控制在畫布內，避免超出邊界。
- 大小 (radius)：在 10 到 30 像素之間隨機變化。
- 顏色：隨機生成明亮的 RGB 顏色，確保星星看起來鮮艷。

防止邊界問題：

- 在 x 和 y 的隨機範圍中，留出星星的半徑空間 (50 像素)，避免星星超出畫布邊界。

執行步驟：

- 使用 for 迴圈生成 30 顆星星，每顆星星的位置、大小、顏色均隨機，並通過 draw_star() 函數繪製。

旋轉角度的隨機化：

- 在 draw_star 函數中，加入 rotation_angle 參數，這個角度是每顆星星的旋轉角度。
- 使用 random.uniform(0, 2 * math.pi) 生成一個隨機的浮動角度，範圍是 0 到 2π（360 度），使每顆星星的方向隨機。

角度應用：

- 每次計算星星頂點的位置時，我們將 rotation_angle 加入到計算公式 angle = (math.pi / points) * i + rotation_angle 中，使得星星根據隨機的旋轉角度進行旋轉。

40 曼陀羅圖案：

```
from browser import document as doc

from browser import html

import math


# 設定畫布大小

canvas_width = 500

canvas_height = 500


# 設定畫布

canvas = html.CANVAS(width=canvas_width, height=canvas_height)

brython_div = doc["brython_div1"]

brython_div <= canvas


# 獲取繪圖上下文

ctx = canvas.getContext("2d")


# 設定中心點

center_x = canvas_width / 2

center_y = canvas_height / 2


# 設定圓的數量和半徑增長

num = int(input("請輸入圓的數量"))

radius_increment = 40
```

```
# 設定線條的數量（圓周上的線條數量）
num_lines = int(input("請輸入線的數量"))

# 繪製曼陀羅圖案
for i in range(num):
    radius = (i + 1) * radius_increment    # 每個圓的半徑

    # 繪製同心圓
    ctx.beginPath()
    ctx.arc(center_x, center_y, radius, 0, 2 * math.pi)    # 圓形
    ctx.strokeStyle = "black"    # 設定邊框顏色
    ctx.lineWidth = 2    # 邊框寬度
    ctx.stroke()

    # 繪製圓周上的線條
    for j in range(num_lines):
        angle = (2 * math.pi / num_lines) * j    # 計算每條線的角度
        line_x = center_x + radius * math.cos(angle)    # 線的終點 X 坐標
        line_y = center_y + radius * math.sin(angle)    # 線的終點 Y 坐標

        ctx.beginPath()
        ctx.moveTo(center_x, center_y)    # 從中心點開始
        ctx.lineTo(line_x, line_y)    # 畫線到圓周上的點
```

```
ctx.strokeStyle = "black"    # 設定線條顏色
ctx.lineWidth = 1    # 線條寬度
ctx.stroke()
```

同心圓：

- 我們使用 ctx.arc(center_x, center_y, radius, 0, 2 * math.pi) 來繪製每個同心圓，並且讓圓的半徑隨著迴圈的次數增長 ((i + 1) * radius_increment)。
- 每個圓的邊框顏色為黑色，邊框寬度設置為 2。

圓周上的線條：

- num_lines 設置為 12，這表示每個圓周上會有 12 條線條。每條線的角度通過 (2 * math.pi / num_lines) * j 計算，確保每條線均勻分佈。
- 每條線從中心點 (center_x, center_y) 畫到圓周上的一個點 (line_x, line_y)，這些點是根據圓的半徑和角度計算出來的。

圖案對稱性：

- 由於所有的線條都以固定的角度分佈（(2 * math.pi / num_lines)），並且每條線都從中心點發出，所以整個圖案會展現出對稱的效果。

41 繪製直線：

```
from browser import document as doc

from browser import html

import math


# 設定畫布大小

canvas_width = 500

canvas_height = 500


# 設定畫布

canvas = html.CANVAS(width=canvas_width, height=canvas_height)

brython_div = doc["brython_div1"]

brython_div <= canvas


# 獲取繪圖上下文

ctx = canvas.getContext("2d")


# 設定間距和線條數量

line_spacing = int(input("請輸入每條直線之間的間距"))  # 每條直線之間的間距

num_lines = int(input("請輸入直線數量"))  # 繪製的直線數量

angle_deg = int(input("請輸入傾斜角度"))  # 設定所有直線的傾斜角度（度數）


# 將角度轉換為弧度

angle_rad = math.radians(angle_deg)
```

```
# 固定的 X 偏移量（根據角度計算）
x_offset = math.tan(angle_rad) * canvas_height

# 使用 for 迴圈繪製平行的傾斜直線
for i in range(num_lines):
    y_position = i * line_spacing  # 計算每條直線的 Y 坐標

    # 計算起點和終點位置
    x_start = 0  # 直線起點 X 坐標
    y_start = y_position  # 直線起點 Y 坐標
    x_end = canvas_width  # 直線終點 X 坐標
    y_end = y_position + x_offset  # 直線終點 Y 坐標，根據角度偏移

    # 繪製直線
    ctx.beginPath()
    ctx.moveTo(x_start, y_start)
    ctx.lineTo(x_end, y_end)
    ctx.strokeStyle = "black"  # 設定線條顏色
    ctx.lineWidth = 2  # 設定線條寬度
    ctx.stroke()
```

畫布設置：

- 畫布的寬度設為 500 像素，高度設為 500 像素，這樣可以容納多條直線。

直線間距與數量：

- line_spacing 設置為 20，表示每條直線之間的垂直間距。
- num_lines 設置為 25，表示總共繪製 25 條直線。

繪製直線：

- 在 for 迴圈中，i 表示直線的編號，每次迴圈增加一條直線。
- 計算每條直線的 Y 坐標 y_position = i * line_spacing，這樣每條直線的 Y 坐標都會隨著 i 逐漸增加，從而產生間隔。
- 使用 ctx.moveTo(0, y_position) 設置每條直線的起點為畫布的左邊 (x = 0)，y = y_position，然後使用 ctx.lineTo(canvas_width, y_position) 設置終點為畫布的右邊，形成水平的平行直線。

42 繪製矩形:

```
from browser import document as doc

from browser import html
```

```
# 設定畫布大小
canvas_width = 500
canvas_height = 500

# 設定畫布
canvas = html.CANVAS(width=canvas_width, height=canvas_height)
brython_div = doc["brython_div1"]
brython_div <= canvas

# 獲取繪圖上下文
ctx = canvas.getContext("2d")

# 設定起始矩形的尺寸和間距增量
initial_width = int(input("請輸入初始矩形寬度"))  # 初始矩形寬度
initial_height = int(input("請輸入初始矩形高度"))  # 初始矩形高度
width_increment = int(input("請輸入矩形寬度增加的大小"))  # 每次矩形寬度增加的大小
height_increment = int(input("請輸矩形高度增加的大小"))  # 每次矩形高度增加的大小
spacing = int(input("請輸入矩形的間距"))  # 每個矩形的間距

# 設定矩形的起始位置
x_position = 50  # 起始矩形的 X 坐標
y_position = 50  # 起始矩形的 Y 坐標
```

```
# 繪製 10 個矩形，並且每個矩形的尺寸和間距增加
for i in range(10):
    # 計算矩形的寬度和高度
    width = initial_width + (width_increment * i)  # 寬度隨著迴圈增加
    height = initial_height + (height_increment * i)  # 高度隨著迴圈增加

    # 繪製矩形
    ctx.beginPath()
    ctx.rect(x_position, y_position, width, height)  # 每次繪製在 Y 軸上
    ctx.strokeStyle = "black"  # 設定矩形邊框顏色
    ctx.lineWidth = 2  # 設定矩形邊框寬度
    ctx.stroke()

    # 更新 y_position，確保矩形不會重疊
    y_position += height + spacing  # 確保每個矩形之間有間距
```

起始矩形的尺寸：

- 初始矩形的寬度 (initial_width) 和高度 (initial_height) 分別設定為 30 和 20。這是第一個矩形的尺寸。

尺寸遞增：

- width_increment 和 height_increment 控制每次繪製的矩形寬度和高度增長的量。在每次迴圈中，這些值將會根據迴圈的次數進行增加。

矩形位置：

- 每個矩形的 x_position 都是固定的，而 y_position 則根據每次迴圈的 i

值逐漸向下移動。這樣，每個矩形之間會有固定的間距。

- y_position 更新：
  - 每次繪製矩形後，y_position 都會增加一個值，這個值是當前矩形的高度 (height) 加上額外的間距 (spacing)。這樣可以確保每個矩形的下邊不會與下一個矩形重疊。
  - y_position += height + spacing 這一行代碼確保了矩形之間的垂直間距。
- 矩形尺寸遞增：
  - 每次繪製的矩形寬度和高度都會根據 width_increment 和 height_increment 遞增。
-

繪製矩形：

- ctx.rect(x_position, y_position + i * spacing, width, height) 用來繪製每個矩形。i * spacing 確保矩形之間的間距。

43 繪製圓形

```
from browser import document as doc

from browser import html
```

```
# 設定畫布大小
canvas_width = 500
canvas_height = 500

# 設定畫布
canvas = html.CANVAS(width=canvas_width, height=canvas_height)
brython_div = doc["brython_div1"]
brython_div <= canvas

# 獲取繪圖上下文
ctx = canvas.getContext("2d")

# 設定圓形的起始半徑和位置增量
initial_radius = int(input("請輸入初始圓形半徑"))  # 初始圓形半徑
radius_increment = int(input("請輸入圓形半徑增加的大小"))  # 每次圓形半徑增加的大小
position_increment = int(input("請輸入圓心位置的增量"))  # 每次圓心位置的增量（水平和垂直）
num = int(input("請輸入圓型數量"))  # 圓型數量

# 繪製 10 個圓形，每次增加半徑和位置
for i in range(num):
    radius = initial_radius + (radius_increment * i)  # 計算每個圓形的半徑
    x_position = (i + 1) * position_increment  # 計算每個圓形的 X 坐標
    y_position = (i + 1) * position_increment  # 計算每個圓形的 Y 坐標
```

```
# 繪製圓形
ctx.beginPath()
ctx.arc(x_position, y_position, radius, 0, 2 * 3.14159)  # 圓心 (x_position, y_position)，半徑 radius
ctx.strokeStyle = "black"  # 設定圓形邊框顏色
ctx.lineWidth = 2  # 設定圓形邊框寬度
ctx.stroke()
```

---

radius 遞增：

- 每個圓形的半徑隨著循環次數 (i) 增加。第一個圓形的半徑是 20，每次增加 10。

position_increment：

- 每次圓形的位置也會隨著循環次數增加，使得每個圓形的中心位置會在畫布上沿著對角線（x_position 和 y_position）遞增。這樣可以確保圓形不會重疊。

ctx.arc(x_position, y_position, radius, 0, 2 * 3.14159)：

- 使用 arc() 方法來繪製圓形，這裡的 (x_position, y_position) 是圓心，radius 是圓的半徑，0 和 2 * 3.14159 表示圓形的起始角度和終止角度，這樣就可以繪製完整的圓。