

W7

```
a = " 我已經會寫簡單的 Python 程式了"  
for i in range(1,11);  
    print(i,a)
```

第一行

1. 變數 a：

- I. a 是變數名稱，用來儲存資料。
- II. 在 Python 中，變數可以儲存任何資料類型（例如字串、數字、列表等）。

2. 字串：

- I. "我已經會寫簡單的 Python 程式了" 是一個字串（String）。
- II. 字串是以雙引號 " 或單引號 ' 包裹的文字資料。

3. 作用：

- I. 將這段文字儲存在變數 a 中，之後可以透過 a 來取用或操作這段文字。

第二行

1. 關鍵字 for

- for 是用來啟動一個循環的關鍵字。
- 它會依序迭代一個可迭代對象（例如列表、字串或範圍）。

2. 變數 i

- i 是迴圈變數，每次迴圈時都會被賦予新的值。
- 在這段程式碼中，i 依次會取值 1 到 10（包含 1，但不包括 11）。

3. 函數 range(1, 11)

- I. range() 函數生成一個範圍對象。
- II. 語法：range(start, stop, step)
 - start：起始值，這裡是 1。
 - stop：結束值（**不包含**該值），這裡是 11。
 - step（可選）：每次增量（預設為 1）。
- III. 因此，range(1, 11) 生成的值為：1, 2, 3, 4, 5, 6, 7, 8, 9, 10。

4. 冒號：

- I. 冒號用於結束 for 的語法並指示接下來是該迴圈內的程式塊。

第三行

1. 在程式中，`print(i, a)` 的作用是：

- I. `print`：Python 的內建函式，用於輸出文字、數值或變數的內容。
- II. `i`：迴圈變數，代表目前迴圈執行的數值（從 1 到 10）。
- III. `,`（逗號）：在 `print` 中，逗號用於分隔多個項目，會在輸出時以空格分隔。
- IV. `a`：這是字串變數，內容是 "我已經會寫簡單的 Python 程式了"。

W9

w9_triangle_1

```
n = 5 # 總共的行數
space = ' '
for i in range(n): # 對於每一行
    # 印出空格
    print(space * (n - i - 1), end='') # 在每行前面印出空格
    # 印出星號
    print('*' * (2 * i + 1)) # 每行印出 2*i + 1 個星號
```

1. `space = ' '`

- I. 這行的意思是將字元 `' '` 賦值給變數 `space`。
- II. 這表示每次需要"空格"（或這裡使用的底線 `_`）時，我們會用變數 `space` 來代表它。這樣，程式碼更容易改動。例如，若想要改成真正的空格，只需將 `space = ' '` 即可。

`for i in range(n):`

- I. 這是一個循環，執行 `n` 次，這裡 `n = 5`，所以 `i` 會依序取值為 0, 1, 2, 3, 4。

2. `space * (n - i - 1):`

- I. 這表示用字元 `space`（即 `' '`）重複 `(n - i - 1)` 次，生成一個空格的字串。
- II. `n` 是總行數，`i` 是當前的行號（從 0 開始）。
- III. `(n - i - 1)` 是計算該行需要的空格數，保證星號輸出是居中的。

3. `end=''`:

- I. 預設情況下，`print()` 在輸出後會自動換行。
- II. 使用 `end=''`，可以讓這部分的輸出不換行，這樣星號可以緊接著空格輸出在同一行。

4. `'*'`:

- I. 這是星號字元，用來組成三角形的內容。

5. `*`（乘號運算符號）:

I. 這裡的 `*` 是字串重複運算符號，將 `'*'` 重複指定的次數。

6. $(2 * i + 1)$:

I. 每行星號的個數計算公式。

II. 當前的行數為 i (從 0 開始計算)。

III. 對於行數 i :

- 星號數量是 $2 * i + 1$ ，表示每行的星號數量隨行數增加。
- 比如：
- 第 0 行 ($i = 0$): $2 * 0 + 1 = 1$ ，印出 1 個星號。
- 第 1 行 ($i = 1$): $2 * 1 + 1 = 3$ ，印出 3 個星號。
- 第 2 行 ($i = 2$): $2 * 2 + 1 = 5$ ，印出 5 個星號。
- 依此類推。

w9_triangle_2

`n = 5` # 定義金字塔的總行數，這裡我們用 `n=5`，表示金字塔共有 5 行

`space = ' '`

第一部分：打印上半部 (包括中間行)

`for i in range(1, n + 1):` # 從第 1 行到第 n 行， i 代表行數

這裡我們打印空格，讓星號可以靠右對齊

空格數量是 $(n - i)$ ，因為第一行需要 4 個空格，第二行需要 3 個空格，以此類推

`print(space * (n - i), end='')` # `end=''` 防止換行，讓星號和空格在同一行輸出

打印星號。每一行的星號數量是 i ，從 1 開始逐行增加

第一行 1 顆星，第二行 2 顆星，第三行 3 顆星... 這樣增加

`print('*' * i)` # 每行輸出 i 個星號

第二部分：打印下半部 (去除中間行)

`for i in range(n - 1, 0, -1):` # 這裡我們從 $n-1$ 行開始，逐行減少星號數量

空格的數量依然是 $(n - i)$ ，這是因為下半部的對稱性

比如當 $i=n-1$ 時，空格數量是 1，當 $i=n-2$ 時，空格數量是 2

`print(space * (n - i), end='')` # 同樣使用 `end=''` 來避免換行

打印星號數量為 i ，從 $n-1$ 開始逐行減少

所以當 $i = n-1$ 時，打印 $n-1$ 顆星，當 $i = n-2$ 時，打印 $n-2$ 顆星，依此類推

`print('*' * i)` # 每行輸出 i 顆星號

11x11 的字元區域中，以 "*" 字元列印出圓型區域。

定義區域大小

width = 11

height = 11

圓心位置

center_x = width // 2 # 圓心 x 坐標，位於區域中間

center_y = height // 2 # 圓心 y 坐標，位於區域中間

圓的半徑

radius = 5

縱向縮放比例

vertical_scale = 1 # 適當調整比例以接近圓形

遍歷字元區域

for y in range(height): # 每一行

for x in range(width): # 每一列

計算每個點與圓心的距離（考慮縮放比例）

dx = x - center_x

dy = (y - center_y) * vertical_scale # 縱向乘以縮放比例

判斷是否在圓內

if dx ** 2 + dy ** 2 <= radius ** 2:

print("*", end="") # 圓內點用 "*" 表示

else:

print(" ", end="") # 圓外點用空格表示

print() # 換行

1. 定義區域大小

I. 作用：定義文字區域的寬度和高度。

- width：區域的總列數（水平寬度）。
- height：區域的總行數（垂直高度）。

II. 使用範例：

- 當 width = 11 和 height = 11 時，文字區域是一個 11x11 的矩形。

2. 計算圓心位置

I. 作用：計算圓心在文字區域中的位置。

- center_x：圓心的水平位置。
- center_y：圓心的垂直位置。

II. 計算方法：

- 使用整數除法 `//`，將寬度或高度的一半作為圓心位置。
- 例如：
 - 如果 `width = 11`，則 `center_x = 11 // 2 = 5`。
 - 如果 `height = 11`，則 `center_y = 11 // 2 = 5`。
- 圓心位於區域的中心點 `(5, 5)`。

3. 定義圓的半徑、縮放比例

作用：

I. `radius`：定義圓的半徑。

- 半徑表示從圓心到圓周的距離。
- 此處半徑為 `5`。

II. `vertical_scale`：調整縱向縮放比例。

- 值為 `1` 表示縱向比例與水平方向一致。
- 可通過調整此值來壓縮或拉伸圓形。

4. 繪製圓形

I. 外層迴圈：遍歷每一行

- `for y in range(height):`

- 作用：
 - 逐行遍歷文字區域中的每一行。
 - `y` 代表當前行的索引（從 `0` 開始）。
 - 範圍為 `0 ~ height-1`（共 `11` 行）。

II. 內層迴圈：遍歷每一列

- `for x in range(width):`

- 作用：
 - 在每行中，逐列遍歷文字區域中的每個點。
 - `x` 代表當前列的索引（從 `0` 開始）。
 - 範圍為 `0 ~ width-1`（共 `11` 列）。

5. 計算點到圓心的距離

I. 計算水平方向距離：

- `dx = x - center_x` 表示該點與圓心在水平方向的距離。

II. 計算垂直方向距離（考慮縮放）：

- `dy = (y - center_y) * vertical_scale` 表示該點與圓心在垂直方向的距離。

- 如果 `vertical_scale = 1`，則無縮放影響。

6. 判斷是否在圓內

I. 距離公式：`dx ** 2 + dy ** 2` 計算該點到圓心的平方距離。

- 如果平方距離小於或等於半徑的平方，則點位於圓內。
- 否則，點在圓外。

II. 輸出字符：

- 圓內的點輸出 `*`。
- 圓外的點輸出空格。

III. `end=""` 的作用：

- 防止自動換行，確保每行的字符緊密排列。

7. 每行結束後換行

I. `print()`，作用：在每行的最後執行換行，使輸出結果呈現矩形區域。

像素繪製圓形

```
from browser import html
from browser import document as doc

# 利用 html 建立 canvas 超文件物件
canvas = html.CANVAS(width=400, height=400)
brython_div = doc["brython_div1"]
brython_div <= canvas

# 每一格的 pixel 數
gs = 20

# canvas 的上下文
ctx = canvas.getContext("2d")

def dRect(lux, luy, w, h, s=1, c="lightgrey"):
    """繪製網格框"""
    ctx.lineWidth = s
    ctx.strokeStyle = c
    ctx.beginPath()
    ctx.rect(lux, luy, w, h)
    ctx.stroke()
```

```

def grid(width, height, grid_pix):
    """繪製網格"""
    for i in range(width):
        for j in range(height):
            dRect(i * grid_pix, j * grid_pix, grid_pix, grid_pix, 1, "lightgrey")

def fill(x, y, color):
    """填充顏色"""
    ctx.fillStyle = color
    ctx.fillRect(x * gs, y * gs, gs, gs)

# 繪製網格
grid(11, 11, gs)

# 定義圓的參數
width, height = 11, 11
center_x, center_y = 5, 5 # 圓心座標（以網格單位表示）
radius = 5                # 圓的半徑（以網格單位表示）

# 填充圓形內部區域
for y in range(height):
    for x in range(width):
        # 計算點到圓心的距離
        distance = ((x - center_x) ** 2 + (y - center_y) ** 2) ** 0.5
        if distance <= radius:
            fill(x, y, "black")

```

1. 前置設定

I. canvas 和 brython_div :

- **Canvas**: 是一個 HTML5 Canvas 元素，設定寬高為 400 像素。
- **brython_div**: 是網頁上的一個 <div> 元素 (ID 為 brython_div1)。
- 用 <= 把 **canvas** 加入到 **brython_div** 中，使它顯示在網頁上。

II. gs :

- gs = 20
- 定義每個網格的大小為 20 像素。
- 例如，11x11 網格中，每個網格為 20x20 像素。

III. ctx :

- ctx = canvas.getContext("2d")
- ctx 是 Canvas 的繪圖上下文，允許我們使用 2D 繪圖工具繪製圖形。

2. 繪製函數

dRect 函數：繪製單個網格框

I. `def dRect(lux, luy, w, h, s=1, c="lightgrey"):`

- `lux` 和 `luy` :
 - 表示矩形左上角的 `x` 和 `y` 座標（`lux` 是橫向，`luy` 是縱向）。
 - 例如，`lux=0, luy=0` 則表示矩形從 Canvas 的左上角開始畫。
- `w` 和 `h` :
 - `w` 是矩形的寬度，`h` 是矩形的高度。
 - 例如，`w=50, h=30` 則表示畫出一個寬 50 像素、高 30 像素的矩形。
- `s`（可選，預設值為 1）：
 - 表示矩形邊框的線條寬度。
 - 預設值為 1，若要更粗的邊框，可以設為更大的值，例如 `s=2`。
- `c`（可選，預設值為 "lightgrey"）：
 - 表示矩形邊框的顏色。
 - 預設為淺灰色（"lightgrey"），但可以改為其他顏色，例如 "black"、"red"。

II. `ctx.lineWidth = s`

- 設定邊框的線條寬度為 `s`。
- 例如，若 `s=2`，邊框的粗細會加倍。

III. `ctx.strokeStyle = c`

- 設定邊框的顏色為 `c`。
- 例如，若 `c="red"`，矩形的邊框顏色會變成紅色。

IV. `ctx.beginPath()`

- 開始一個新的繪圖路徑，確保繪製的圖形不會影響其他圖形。

V. `ctx.rect(lux, luy, w, h)`

- 在 Canvas 上定義一個矩形的路徑。
- （`lux, luy`）為矩形的左上角，`w` 為寬度，`h` 為高度。

VI. `ctx.stroke()`

- 使用之前設定的線條樣式和顏色，繪製出矩形的邊框。

grid 函數：繪製整體網格

```
def grid(width, height, grid_pix):
```


I. width :

- 表示網格的列數（橫向）。
- 例如，width=10 表示網格有 10 列。

II. height :

- 表示網格的行數（縱向）。
- 例如，height=10 表示網格有 10 行。

III. grid_pix :

- 表示每個網格的邊長（像素大小）。
- 例如，grid_pix=20 表示每個網格是 20x20 像素的正方形。

外層迴圈：處理網格的列數（橫向）

- for i in range(width):
- 作用：遍歷每一列（橫向位置）。
- 變數 i 為列的索引（從 0 開始）。

內層迴圈：處理網格的行數（縱向）

- for j in range(height):
- 作用：遍歷每一行（縱向位置）。
- 變數 j 為行的索引（從 0 開始）。

呼叫 dRect 繪製單個網格框

- dRect(i * grid_pix, j * grid_pix, grid_pix, grid_pix, 1, "lightgrey")

I. i * grid_pix 和 j * grid_pix :

- 計算每個網格左上角的 x 和 y 座標：
 - x 座標：i * grid_pix（列索引乘以網格大小）。
 - y 座標：j * grid_pix（行索引乘以網格大小）。

II. grid_pix, grid_pix :

- 設定矩形的寬度與高度（每個網格都是正方形）。

III.1 :

- 邊框的線條寬度設為 1 像素。

IV. "lightgrey" :

- 邊框顏色設為淺灰色。

fill 函數：填充指定網格

fill 是一個用於填充指定網格單元的函數。該函數將在 Canvas 上的 (x, y) 格位置，繪製一個填滿顏色的矩形。

x 和 y：

- 這是目標網格的行列座標。
- 例如，x = 3, y = 4 將會填充第 4 行、第 3 列的網格（以 0 為起始索引）。

color：

- 矩形填充的顏色，可以是任何有效的 CSS 顏色值（例如 "red", "#00FF00", "rgba(0, 0, 255, 0.5)" 等）。

ctx.fillStyle = color

- 使用 Canvas API 的 fillStyle 設置要繪製矩形的顏色。

ctx.fillRect(x * gs, y * gs, gs, gs)

- 使用 fillRect 方法在 Canvas 上繪製一個矩形：
 - x * gs 和 y * gs 是矩形的左上角座標。
 - gs 是每格的大小（20 像素），定義矩形的寬與高。

3. 主程式邏輯

I. 繪製 11x11 網格

- grid(11, 11, gs)
- 呼叫 grid 函數繪製 11x11 的網格，每格大小為 20x20 像素。

II. 圓形參數設定

- center_x, center_y = 5, 5 # 圓心座標
- radius = 5 # 圓的半徑
- 圓心位置設為 (5, 5)（對應網格中心）。
- 半徑設定為 5（以網格為單位）。

III. 填充圓形內部區域

```
for y in range(height):
```

```
    for x in range(width):
```

```
        distance = ((x - center_x) ** 2 + (y - center_y) ** 2) ** 0.5
```

```
        if distance <= radius:
```

```
            fill(x, y, "black")
```

- 遍歷網格的每個點 (x, y)。
- 計算到圓心的距離：

$$\text{distance} = \sqrt{(x - \text{center}_x)^2 + (y - \text{center}_y)^2}$$

- 使用公式：
- 判斷是否在圓內：
 - 若 distance <= radius，填充該格為黑色。

