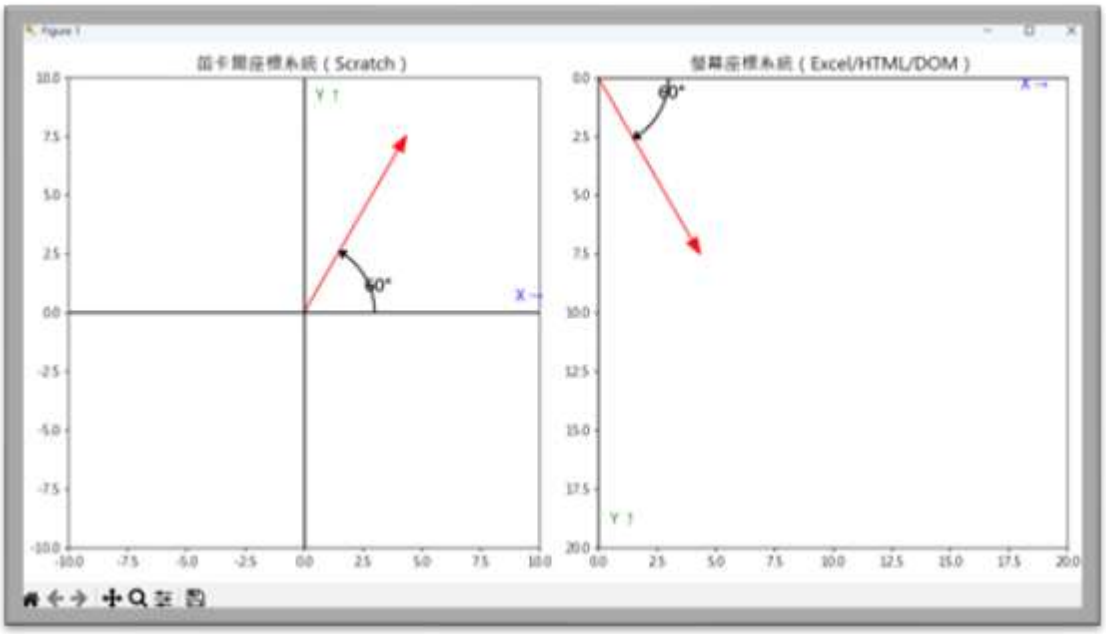
🧭 程式世界中的座標系統：從笛卡爾到螢幕座標的轉換之旅

在程式設計與圖形繪製中，「座標系統」是不可或缺的概念。不同語言與工具所採用的座標系統各有差異，理解這些差異不僅能幫助我們更準確地定位元素，也能避免方向錯誤與角度混淆的問題。



- 📌 座標系統的兩大類型

1. 數學型座標系統（笛卡爾座標）

這類座標系統以數學中的笛卡爾座標為基礎，原點通常在畫布中心或左下角，X 軸向右，Y 軸向上。角度以逆時針方向為正。

常見工具：

- Python（Matplotlib）
- Java（Graphics2D）
- JavaScript（Canvas）
- HTML（SVG）

2. 螢幕型座標系統（像素座標）

這類座標系統以螢幕像素為基礎，原點在左上角，X 軸向右，Y 軸向下。角度以順時針方向為正。

常見工具：

- Excel（儲存格定位）
- HTML/CSS（DOM 元素定位）
- JavaScript（DOM 操作）

⚠️ 特殊例外：Scratch

Scratch 雖然是視覺化程式工具，但其座標系統採用中心為原點，Y 軸向上，與笛卡爾座標相似。

🎯 為何座標系統差異重要？

在繪製圖形、動畫或定位元素時，若未理解座標系統的方向與原點位置，可能會導致：

- 元素位置錯誤
- 角度方向相反
- 動畫移動方向不如預期

Python 實作：比較兩種座標系統

以下 Python 程式碼使用 `matplotlib` 同時繪製兩種座標系統，並標示 45° 的角度方向：

- 左圖：笛卡爾座標系統（Y 軸向上，角度逆時針）
- 右圖：螢幕座標系統（Y 軸向下，角度順時針）

```
python
import matplotlib.pyplot as plt
import numpy as np

fig, axs = plt.subplots(1, 2, figsize=(12, 6))
```

```
# 笛卡爾座標系統
ax = axs[0]
ax.set_xlim(-10, 10)
ax.set_ylim(-10, 10)
ax.set_aspect('equal')
ax.set_title('笛卡爾座標系統', fontsize=14)
ax.axhline(0, color='black')
ax.axvline(0, color='black')
ax.text(9, 0.5, 'X →', fontsize=12, color='blue')
ax.text(0.5, 9, 'Y ↑', fontsize=12, color='green')
angle_rad = np.deg2rad(45)
x = np.cos(angle_rad) * 8
y = np.sin(angle_rad) * 8
ax.arrow(0, 0, x, y, head_width=0.5, head_length=0.7, fc='red', ec='red')
ax.text(x/2, y/2, '45°', fontsize=12, color='red', rotation=45)

# 螢幕座標系統
ax = axs[1]
ax.set_xlim(0, 20)
ax.set_ylim(20, 0)  # Y 軸向下
ax.set_aspect('equal')
ax.set_title('螢幕座標系統(Excel/HTML/DOM)', fontsize=14)
ax.axhline(0, color='black')
ax.axvline(0, color='black')
ax.text(18, 1, 'X →', fontsize=12, color='blue')
ax.text(1, 2, 'Y ↓', fontsize=12, color='green')
x = np.cos(angle_rad) * 8
y = np.sin(angle_rad) * 8
ax.arrow(0, 0, x, -y, head_width=0.5, head_length=0.7, fc='red', ec='red')
ax.text(x/2, -y/2, '45°', fontsize=12, color='red', rotation=-45)

plt.tight_layout()
plt.show()
```

- 🔍 結語：選擇正確的座標系統，讓圖形更精準

無論你是設計網頁、製作動畫、撰寫資料視覺化程式，理解座標系統的差異都是基本功。透過本文與 Python 圖示，你可以更清楚地掌握各種座標系統的特性，避免方向錯誤，提升圖形準確度。