



---

---

# Pygame小遊戲

講者:花花公子・嘎滋窩窩頭

---

---

環境幫幫忙

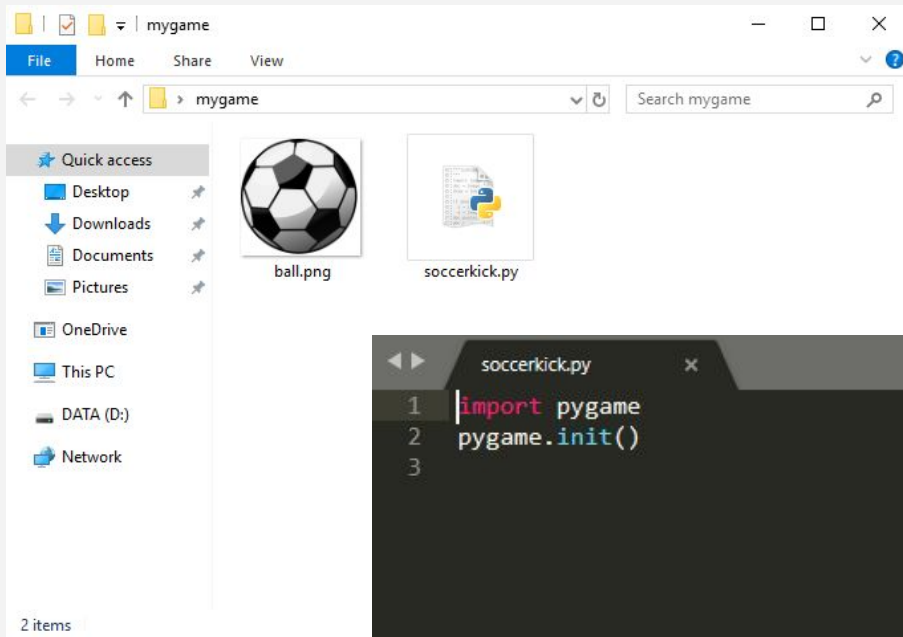
<https://reurl.cc/3qgYR>

下載課程範本

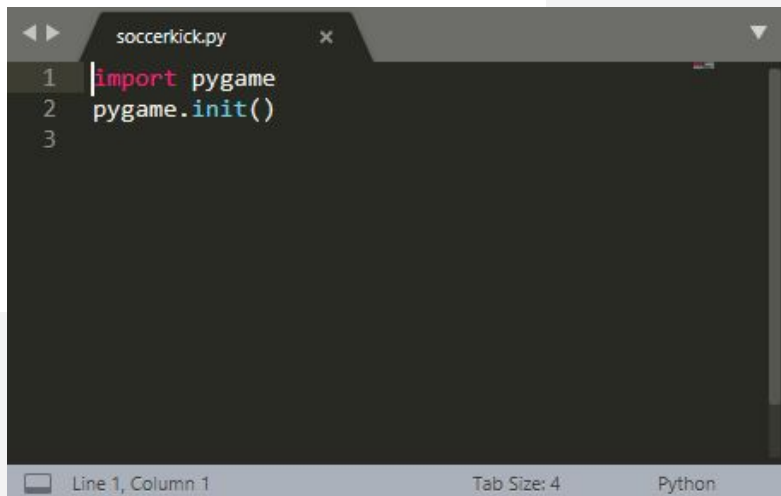
<https://reurl.cc/G050x>

零五零

# 事前準備



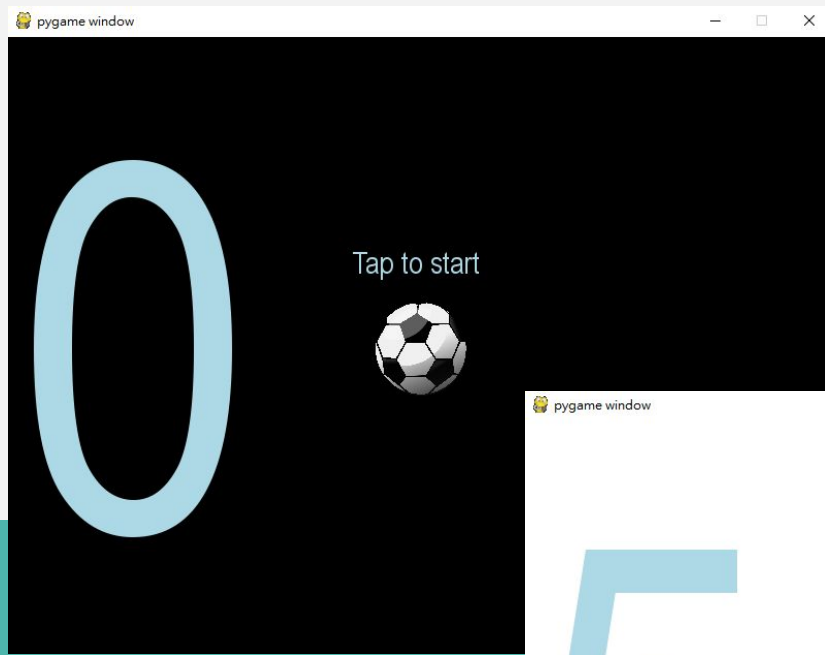
- 先把 **mygame.zip** 解壓縮
- 你會看到一顆球的圖片 **ball.png**
- 以及一個 Python 程式檔 **soccerkick.py**
- 請先打開 **soccerkick.py**



今天的目標

# 遊戲成品

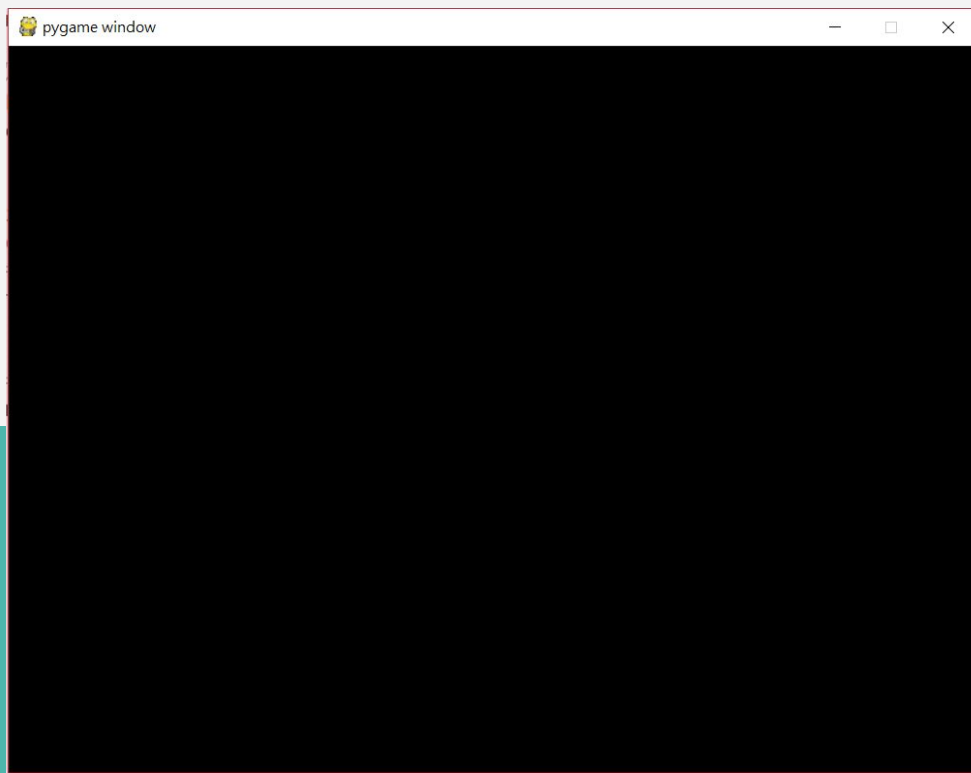
## Soccerkick



- 遊戲成品DEMO
- 顯示視窗
- 建立圖層
- 畫出圓形
- 動畫1:等速移動
- 動畫2:重力模型
- 滑鼠輸入事件:踢球
- 隨機:球會亂噴
- 動畫3:碰壁反彈
- 加入遊戲開始畫面
- 顯示文字:分數計算
- 載入圖片

那我們開始吧

# 顯示視窗





## 顯示視窗



```
1  import pygame
2  pygame.init()
3
4  # 設定視窗的長寬大小
5  SCREEN_SIZE = (800, 600)
6  screen = pygame.display.set_mode(SCREEN_SIZE)
7
8  # 是否遊戲結束
9  running = True
10 while running:
11     # -----遊戲事件偵測-----
12     for event in pygame.event.get():
13         if event.type == pygame.QUIT:
14             running = False
15
16  pygame.quit()
```

## 顯示視窗 - 初始化設定

```
1 import pygame
2 pygame.init()
3
4 # 設定視窗的長寬大小
5 SCREEN_SIZE = (800, 600)
6 screen = pygame.display.set_mode(SCREEN_SIZE)
7
```

- `import pygame`: 引入模組使用



## 顯示視窗 - 初始化設定

```
1  import pygame
2  pygame.init()
3
4  # 設定視窗的長寬大小
5  SCREEN_SIZE = (800, 600)
6  screen = pygame.display.set_mode(SCREEN_SIZE)
7
```

- `pygame.init()`: 所有pygame內的模組都做好初始化

## 顯示視窗 - 設定視窗

```
1  import pygame
2  pygame.init()
3
4  # 設定視窗的長寬大小
5  SCREEN_SIZE = (800, 600)
6  screen = pygame.display.set_mode(SCREEN_SIZE)
```

- `pygame.display.set_mode()`
  - 建立一個視窗物件, 名叫screen
  - ()內放入視窗大小

## 顯示視窗 - 設定視窗

```
1  import pygame
2  pygame.init()
3
4  # 設定視窗的長寬大小
5  SCREEN_SIZE = (800, 600)
6  screen = pygame.display.set_mode(SCREEN_SIZE)
```

- `pygame.display.set_mode()`
  - 建立一個視窗物件, 名叫screen
  - ()內放入視窗大小
- `SCREEN_SIZE`
  - 將視窗的寬度設為800
  - 高度設為600


## 顯示視窗 - 遊戲結束

```
8  # 是否遊戲結束
9  running = True
10 while running:
11     # -----遊戲事件偵測-----
12     for event in pygame.event.get():
13         if event.type == pygame.QUIT:
14             running = False
15
16 pygame.quit()
```

- 當按下右上角的  會關閉程式
- 請不要嘗試移掉這一部分，否則你將會大難臨頭



# 顯示視窗

- 寫到這裡你應該...
  - 遊戲可以正常執行 (按 `ctrl-b` 執行)
  - 800x600 黑畫面
  - 當按下右上角的  會關閉程式
- 如果沒有成功的話快快找小隊輔幫幫忙吧



```
1  import pygame
2  pygame.init()
3
4  # 設定視窗的長寬大小
5  SCREEN_SIZE = (800, 600)
6  screen = pygame.display.set_mode(SCREEN_SIZE)
7
8  # 是否遊戲結束
9  running = True
10 while running:
11     # -----遊戲事件偵測-----
12     for event in pygame.event.get():
13         if event.type == pygame.QUIT:
14             running = False
15
16 pygame.quit()
```

甲區

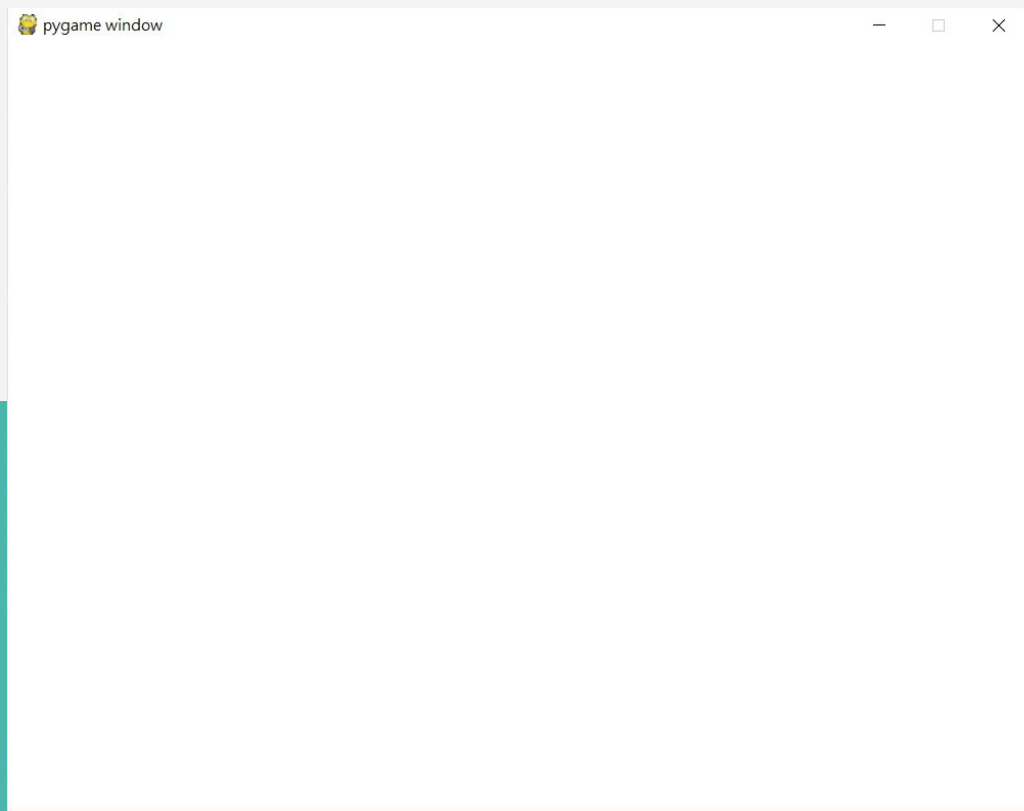
遊戲初始化

乙區

遊戲更新



# 建立圖層



## 建立圖層

視窗 screen

A diagram illustrating the relationship between a window and a canvas. A solid black rectangle represents the '視窗 screen' (Window screen). Overlapping its bottom-left corner is a white rectangle with a red dashed border, representing the '圖層 canvas' (Layer canvas). The text '視窗 screen' is written in white inside the black rectangle, and '圖層 canvas' is written in red inside the white rectangle.

圖層 canvas

# 建立圖層



```
1 import pygame
2 pygame.init()
3
4 SCREEN_SIZE = (800, 600)
5 screen = pygame.display.set_mode(SCREEN_SIZE)
6
7 # 畫布 -----
8 canvas = pygame.Surface(SCREEN_SIZE)
9 # -----
10
11 running = True
12 while running:
13     # 遊戲事件偵測 -----
14     for event in pygame.event.get():
15         if event.type == pygame.QUIT:
16             running = False
17     # 更新畫面 -----
18     canvas.fill(pygame.Color('WHITE'))
19     screen.blit(canvas, (0, 0))
20     pygame.display.update()
21     # -----
22 pygame.quit()
23
```

甲區

乙區

## 建立圖層 - 生出一個圖層(畫布)

```
1 import pygame
2 pygame.init()
3
4 SCREEN_SIZE = (800, 600)
5 screen = pygame.display.set_mode(SCREEN_SIZE)
6
7 # 畫布 -----
8 canvas = pygame.Surface(SCREEN_SIZE)
9 # -----
```

- **pygame.Surface**(圖層的尺寸)
  - 新增一個圖層物件, 指定尺寸, 叫 **canvas**
  - 將圖層大小設成與視窗相同大小

**注意：Surface的 S 要大寫!!**

## 建立圖層 - 圖層填滿

```
11  running = True
12  while running:
13      # 遊戲事件偵測 -----
14      for event in pygame.event.get():
15          if event.type == pygame.QUIT:
16              running = False
17      # 更新畫面 -----
18      canvas.fill(pygame.Color('WHITE'))
19      screen.blit(canvas, (0, 0))
20      pygame.display.update()
21      # -----
22  pygame.quit()
```

- `canvas.fill(顏色)`: 將圖層塗滿指定的顏色

**pygame裡的顏色怎麼表達?**

# 建立圖層 - pygame.Color

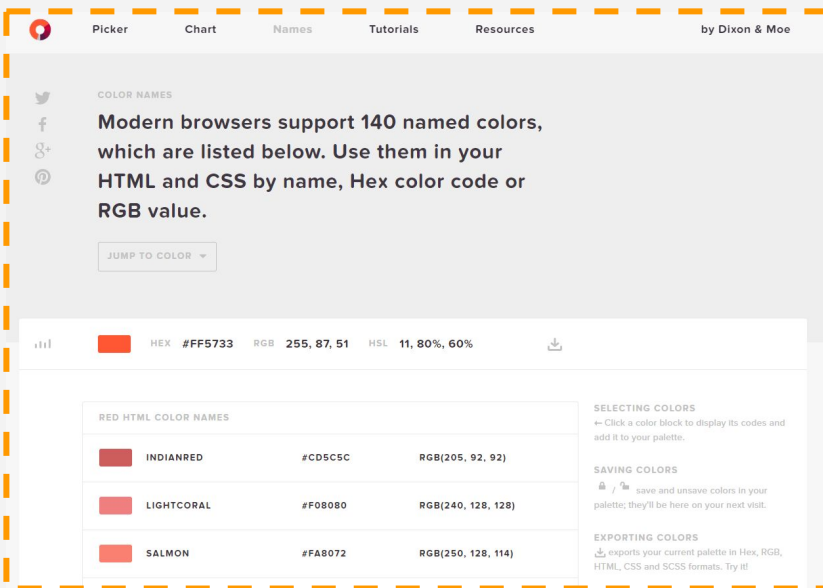
```
canvus.fill(pygame.Color('WHITE'))
```

- `pygame.Color('顏色')`
  - 這個模組提供了更便利的方法來表達顏色
  - 很多顏色已經被定義好可以直接使用
- `pygame.Color`使用的顏色表:

<https://htmlcolorcodes.com/color-names/>



	SALMON
	LIGHTBLUE
	PALEGREEN
	BLACK



The screenshot shows the HTML Color Codes website. The main content area lists 140 named colors. A search bar is present. Below the search bar, there is a table of color names and their corresponding Hex, RGB, and HSL values. The table is titled "RED HTML COLOR NAMES".

Color Name	Hex	RGB	HSL
INDIANRED	#CD5C5C	RGB(205, 92, 92)	
LIGHTCORAL	#F08080	RGB(240, 128, 128)	
SALMON	#FA8072	RGB(250, 128, 114)	

Additional information from the screenshot includes a "SELECTING COLORS" section with instructions on how to click a color block to display its codes and add it to the palette, and a "SAVING COLORS" section with instructions on how to save and unsave colors in the palette. There is also an "EXPORTING COLORS" section with instructions on how to export the current palette in Hex, RGB, HTML, CSS and SCSS formats.

# 建立圖層 - pygame.Color

```
canvus.fill(pygame.Color('WHITE'))
```

**注意：Color的 C 要大寫!!**

- pygame.Color使用的顏色表:

<https://htmlcolorcodes.com/color-names/>

	SALMON
	LIGHTBLUE
	PALEGREEN
	BLACK

Picker Chart Names Tutorials Resources by Dixon & Moe

JUMP TO COLOR ▾

HEX #FF5733 RGB 255, 87, 51 HSL 11, 80%, 60%

RED HTML COLOR NAMES

INDIANRED	#CD5C5C	RGB(205, 92, 92)
LIGHTCORAL	#F08080	RGB(240, 128, 128)
SALMON	#FA8072	RGB(250, 128, 114)

SELECTING COLORS  
← Click a color block to display its codes and add it to your palette.

SAVING COLORS  
🔒 / 🗑️ save and unsave colors in your palette; they'll be here on your next visit.

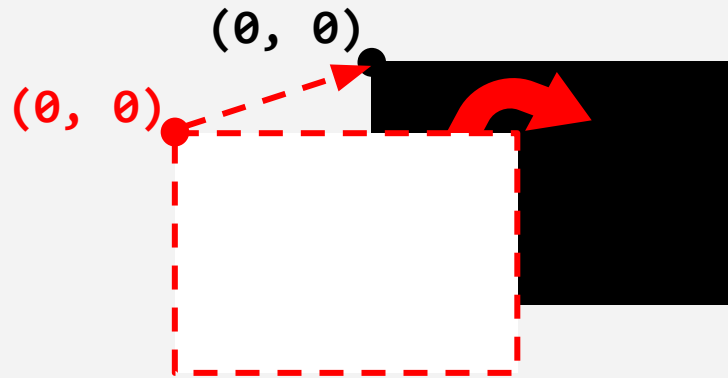
EXPORTING COLORS  
📄 exports your current palette in Hex, RGB, HTML, CSS and SCSS formats. Try it!



## 建立圖層 - 視窗貼貼

```
17      # 更新畫面 -----
18      canvas.fill(pygame.Color('WHITE'))
19      screen.blit(canvas, (0, 0))
20      pygame.display.update()
21      # -----
```

- `screen.blit(圖層物件, 左上角座標位置)`
  - 將圖層(canvas)貼到視窗(screen)上面
  - 左上角座標位置 = (0, 0): 圖層原點會對齊視窗(0, 0)的位置
- 注意: 原點 (0, 0) 在視窗左上角!



建立圖層 - `screen.blit(canvas, (0, 0))`

`(0, 0)`

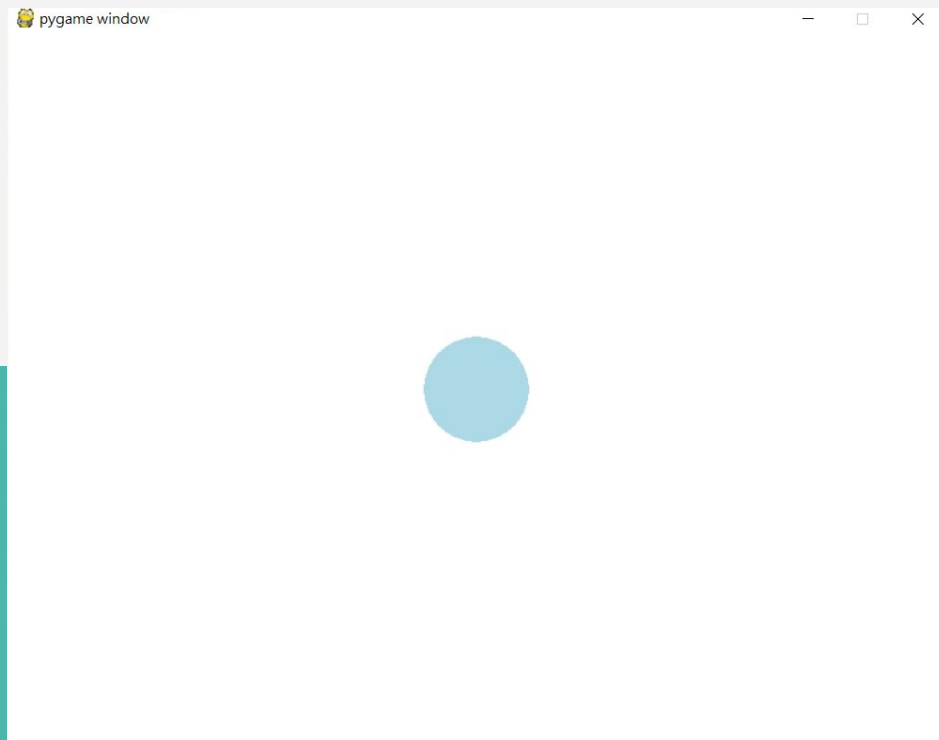


## 建立圖層 - 更新畫面

```
21 # 更新畫面 -----
22 canvas.fill(pygame.Color('WHITE'))
23 pygame.draw.circle(canvas, pygame.Color('LIGHTBLUE'), (x, y), r)
24 screen.blit(canvas, (0, 0))
25 pygame.display.update()
26 # -----
27 pygame.quit()
```

- `pygame.display.update()`
  - 若沒有寫上這行，前面寫再多也不會顯示出來
  - 當程式執行到這行時，pygame才會為我們更新顯示一次

# 畫出圓形



# 畫出圓形



```
1 import pygame
2 pygame.init()
3
4 SCREEN_SIZE = (800, 600)
5 screen = pygame.display.set_mode(SCREEN_SIZE)
6
7 # 球的位置和尺寸 -----
8 x = 400 # 橫軸位置
9 y = 300 # 縱軸位置
10 r = 45 # 半徑
11 # -----
12
13 canvas = pygame.Surface(SCREEN_SIZE)
14
15 running = True
16 while running:
17     # 遊戲事件偵測 -----
18     for event in pygame.event.get():
19         if event.type == pygame.QUIT:
20             running = False
21     # 更新畫面 -----
22     canvas.fill(pygame.Color('WHITE'))
23     pygame.draw.circle(canvas, pygame.Color('LIGHTBLUE'), (x, y), r) # 根據座標和半徑，在畫布上畫出圓形(球)
24     screen.blit(canvas, (0, 0))
25     pygame.display.update()
26     # -----
27 pygame.quit()
```

## 畫出圓形 - 設定圓形

```
4  SCREEN_SIZE = (800, 600)
5  screen = pygame.display.set_mode(SCREEN_SIZE)
6
7  # 球的位置和尺寸 -----
8  x = 400 # 橫軸位置
9  y = 300 # 縱軸位置
10 r = 45  # 半徑
11 # -----
12
13 canvas = pygame.Surface(SCREEN_SIZE)
```

- 設定球的大小與位置
  - 半徑:45
  - 位置座標為(400, 300)

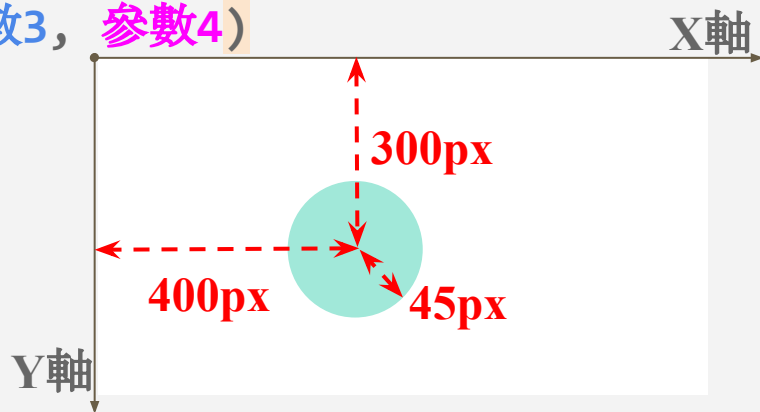
## 畫出圓形 - 設定圓形

```
21 # 更新畫面 -----
22 canvas.fill(pygame.Color('WHITE'))
23 pygame.draw.circle(canvas, pygame.Color('LIGHTBLUE'), (x, y), r)
24 screen.blit(canvas, (0, 0))
25 pygame.display.update()
26 # -----
27 pygame.quit()
```

- `pygame.draw.circle(參數1, 參數2, 參數3, 參數4)`

- 參數1: 圖層物件 → canvas
- 參數2: 圓形的顏色 → LIGHTBLUE
- 參數3: 圓心座標(x, y) → (400, 300)
- 參數4: 圓的半徑 → 45

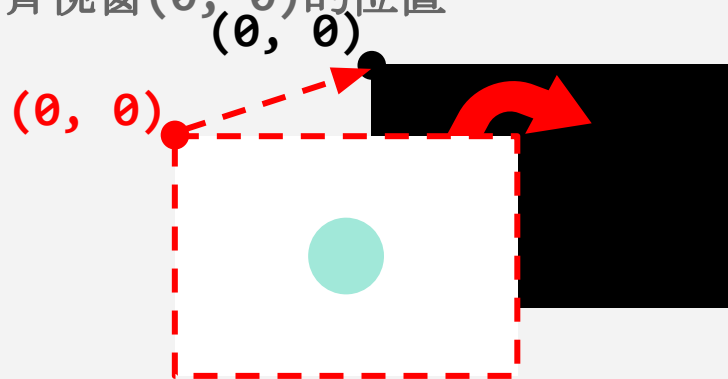
- 注意: 原點 (0, 0) 在圖層左上角!



## 畫出圓形 - 顯示

```
21 # 更新畫面 -----
22 canvas.fill(pygame.Color('WHITE'))
23 pygame.draw.circle(canvas, pygame.Color('LIGHTBLUE'), (x, y), r)
24 screen.blit(canvas, (0, 0))
25 pygame.display.update()
26 # -----
27 pygame.quit()
```

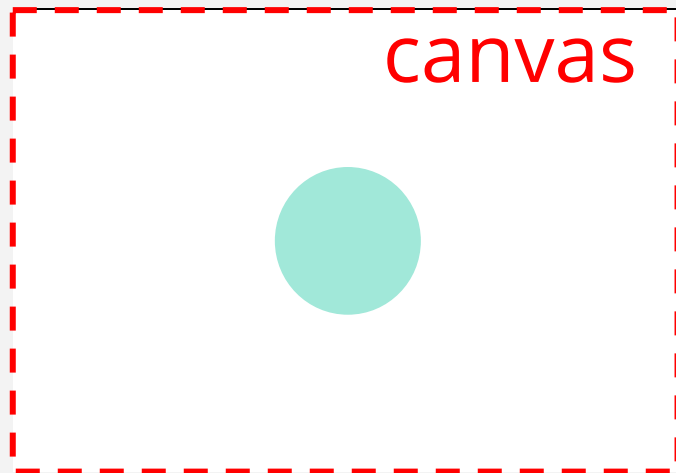
- `screen.blit`(圖層物件, 左上角座標位置)
  - 將圖層(canvas)貼到視窗(screen)上面
  - 左上角座標位置 =  $(0, 0)$ : 圖層原點會對齊視窗 $(0, 0)$ 的位置
- 注意: 原點  $(0, 0)$  在視窗左上角!





畫出圓形 - `screen.blit(canvas, (0, 0))`

`(0, 0)`



## 畫出圓形 - 顯示

```
21 # 更新畫面 -----
22 canvas.fill(pygame.Color('WHITE'))
23 pygame.draw.circle(canvas, pygame.Color('LIGHTBLUE'), (x, y), r)
24 screen.blit(canvas, (0, 0))
25 pygame.display.update()
26 # -----
27 pygame.quit()
```

- `pygame.display.update()`
  - 若沒有寫上這行，前面寫再多也不會顯示出來
  - 當程式執行到這行時，pygame才會為我們更新顯示一次

## 其他幾何圖形(補充)

形狀	模組名稱	參數1	參數2	參數3	參數4	參數5
圓形	<code>pygame.draw.circle</code>	圖層	顏色	圓心(x, y)	半徑	-
長方形	<code>pygame.draw.rect</code>	圖層	顏色	[x, y, 寬, 高]	-	-
線段	<code>pygame.draw.line</code>	圖層	顏色	起點(x, y)	終點(x, y)	線段粗度

更多pygame.draw工具: <https://www.pygame.org/docs/ref/draw.html>

# 讓球動起來！

## 動畫1:等速移動



- 動畫是利用視覺暫留的原理，畫面快速的變幻可以讓我們有連貫的感覺
- 因此我們只要在一定的時間間隔內更新畫面，就可以達到動畫的效果

# 動畫1:等速移動



```
1  import pygame
2  pygame.init()
3
4  SCREEN_SIZE = (800, 600)
5  screen = pygame.display.set_mode(SCREEN_SIZE)
6
7  x = 400
8  y = 300
9  r = 45
10
11 canvas = pygame.Surface(SCREEN_SIZE)
12
13 # 建立時鐘
14 clock = pygame.time.Clock()
15
16 running = True
17 while running:
18     # 遊戲事件偵測 -----
19     for event in pygame.event.get():
20         if event.type == pygame.QUIT:
21             running = False
22     # 動畫控制 -----
23     clock.tick(60) # 每秒執行 60 次
24     y = y + 1 # 球往下等速移動
25     # 更新畫面 -----
26     canvas.fill(pygame.Color('WHITE'))
27     pygame.draw.circle(canvas, pygame.Color('LIGHTBLUE'), (x, y), r)
28     screen.blit(canvas, (0, 0))
29     pygame.display.update()
30     # -----
31
32 pygame.quit()
```

## 動畫1:等速移動 - 計時器

```
13  # 建立時鐘
14  clock = pygame.time.Clock()
15
16  running = True
17  while running:
18      # 遊戲事件偵測 -----
19      for event in pygame.event.get():
20          if event.type == pygame.QUIT:
21              running = False
22      # 動畫控制 -----
23      clock.tick(60) # 每秒執行 60 次
24      y = y + 1 # 球往下等速移動
```

- 利用 `pygame.time` 模組, 建立一個時鐘物件
  - `clock = pygame.time.Clock()`

## 動畫1:等速移動 - 計時器

```
13  # 建立時鐘
14  clock = pygame.time.Clock()
15
16  running = True
17  while running:
18      # 遊戲事件偵測 -----
19      for event in pygame.event.get():
20          if event.type == pygame.QUIT:
21              running = False
```

**注意：Clock的 C 要大寫!!**

- 利用 `pygame.time` 模組, 建立一個時鐘物件
  - `clock = pygame.time.Clock()`



## 動畫1:等速移動 - 計時器

```
13  # 建立時鐘
14  clock = pygame.time.Clock()
15
16  running = True
17  while running:
18      # 遊戲事件偵測 -----
19      for event in pygame.event.get():
20          if event.type == pygame.QUIT:
21              running = False
22      # 動畫控制 -----
23      clock.tick(60) # 每秒執行 60 次
24      y = y + 1 # 球往下等速移動
```

- `clock.tick(60)`: 保證這個迴圈一秒執行60次(也就是平均 $1/60$ 秒執行一次)

## 動畫1:等速移動 - 計時器

```
13  # 建立時鐘
14  clock = pygame.time.Clock()
15
16  running = True
17  while running:
18      # 遊戲事件偵測 -----
19      for event in pygame.event.get():
20          if event.type == pygame.QUIT:
21              running = False
22      # 動畫控制 -----
23      clock.tick(60) # 每秒執行 60 次
24      y = y + 1 # 球往下等速移動
```

- 球每1/60秒, 往下移動一個像素

## 動畫2：重力模型

```

1 import pygame
2 pygame.init()
3
4 SCREEN_SIZE = (800, 600)
5 screen = pygame.display.set_mode(SCREEN_SIZE)
6
7 x = 400
8 y = 300
9 r = 45
10 vx = 0.0 # x速度
11 vy = 0.0 # y速度
12 ax = 0.0 # x加速度
13 ay = 0.68 # y加速度
14
15 canvas = pygame.Surface(SCREEN_SIZE)
16 clock = pygame.time.Clock()
17
18 # 每一幀球的移動
19 def ball_animation():
20     global x, y, vx, vy, ax, ay
21     # 運動公式
22     x = int(x + vx)
23     y = int(y + vy)
24     vx = vx + ax
25     vy = vy + ay
26
27 running = True
28 while running:
29     # 遊戲事件偵測 -----
30     for event in pygame.event.get():
31         if event.type == pygame.QUIT:
32             running = False
33     # 動畫控制 -----
34     clock.tick(60)
35     ball_animation()
36     # 更新畫面 -----
37     canvas.fill(pygame.Color('WHITE'))
38     pygame.draw.circle(canvas, pygame.Color('LIGHTBLUE'), (x, y), r)
39     screen.blit(canvas, (0, 0))
40     pygame.display.update()
41     # -----
42
43 pygame.quit()

```

## 動畫2:重力模型



## 動畫2:重力模型 - 變數初始化

```
4  SCREEN_SIZE = (800, 600)
5  screen = pygame.display.set_mode(SCREEN_SIZE)
6
7  x = 400
8  y = 300
9  r = 45
10 vx = 0.0 # x速度
11 vy = 0.0 # y速度
12 ax = 0.0 # x加速度
13 ay = 0.88 # y加速度
14
15 canvas = pygame.Surface(SCREEN_SIZE)
16 clock = pygame.time.Clock()
```

- 先初始化重力模型所需要的變數

## 動畫2:重力模型 - ball\_animation()

```
19 def ball_animation():
20     global x, y, vx, vy
21     # 運動公式
22     x = int(x + vx)
23     y = int(y + vy)
24     vx = vx + ax
25     vy = vy + ay
```

- 寫一個新的函式, 叫作 `ball_animation()`
- 計算出每一幀(1/60秒)的球的位置、速度

## 動畫2:重力模型 - ball\_animation()

```
19 def ball_animation():
20     global x, y, vx, vy
21     # 運動公式
22     x = int(x + vx)
23     y = int(y + vy)
24     vx = vx + ax
25     vy = vy + ay
```

- int(): 把浮點數(小數)轉換成整數
- 因為x, y表示球的像素位置, 所以必須是整數

(回憶前面畫出圓形的部分 ↓)

```
pygame.draw.circle(canvas, pygame.Color('LIGHTBLUE'), (x, y), r)
```



那, global是什麼?

```
global x, y, vx, vy
```



# global

- 外層定義好的變數

```
7  x = 400
8  y = 300
9  r = 45
10 vx = 0.0 # x速度
11 vy = 0.0 # y速度
12 ax = 0.0 # x加速度
13 ay = 0.88 # y加速度
```

```
19 def ball_animation():
20     global x, y, vx, vy
21     # 運動公式
22     x = int(x + vx)
23     y = int(y + vy)
24     vx = vx + ax
25     vy = vy + ay
```

- 在函式裡面，如果需要改動到外層變數的時候，就必須用到 `global` 這個工具，宣告哪些變數是代表外層變數，否則將會出現預期外的錯誤。

## 動畫2:重力模型 - ball\_animation()

```
27     running = True
28     while running:
29         # 遊戲事件偵測 -----
30         for event in pygame.event.get():
31             if event.type == pygame.QUIT:
32                 running = False
33         # 動畫控制 -----
34         clock.tick(60)
35         ball_animation()
36         # 更新畫面 -----
```

- 每一幀(1/60秒)執行一次ball\_animation()

# 滑鼠輸入事件：踢球

```

1 import pygame
2 pygame.init()
3
4 SCREEN_SIZE = (800, 600)
5 screen = pygame.display.set_mode(SCREEN_SIZE)
6
7 x = 400
8 y = 300
9 r = 45
10 vx = 0.0
11 vy = 0.0
12 ax = 0.0
13 # ay = 0.88
14 ay = 0.3 # 為了測試把重力加速度先調小
15
16 canvas = pygame.Surface(SCREEN_SIZE)
17 clock = pygame.time.Clock()
18
19 def ball_animation():
20     global x, y, vx, vy, ax, ay
21     # 牛頓運動定律
22     x = int(x + vx)
23     y = int(y + vy)
24     vx = vx + ax
25     vy = vy + ay
26
27 # 取得兩個點的距離平方
28 def distance_square(pos1, pos2):
29     x1 = pos1[0]
30     y1 = pos1[1]
31     x2 = pos2[0]
32     y2 = pos2[1]
33     return (x1-x2)*(x1-x2) + (y1-y2)*(y1-y2)

```

## 滑鼠輸入事件: 踢球

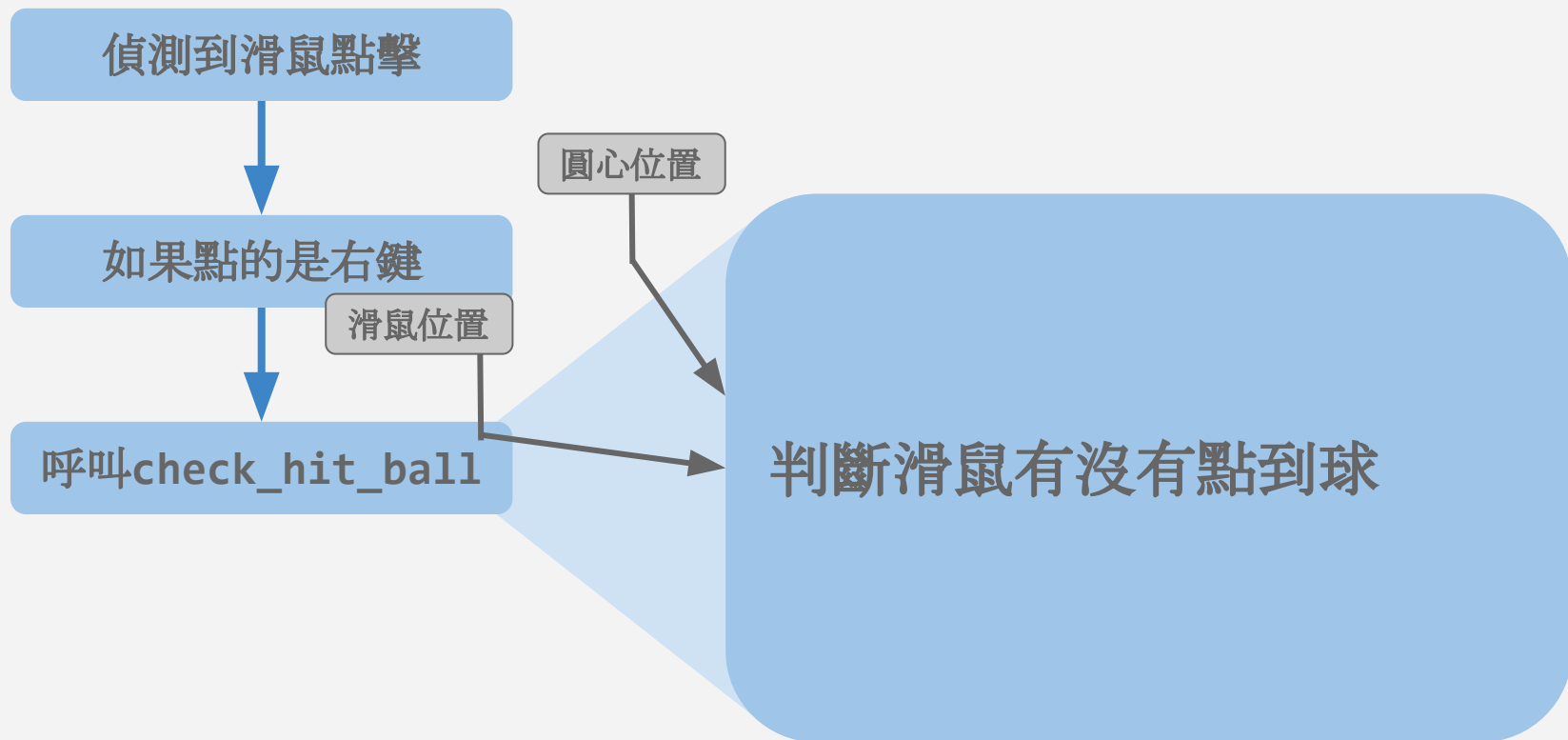
```

35 # 球彈上去的效果
36 def ball_bounce():
37     global vy
38     vy = -20.0 # 球往上飛
39
40 # 確認滑鼠有沒有按到球
41 def check_hit_ball(mouse_pos):
42     ball_pos = [x, y]
43     # 當滑鼠座標離球心的距離小於半徑時, 代表滑鼠座標在球的面積範圍內
44     if r * r > distance_square(mouse_pos, ball_pos):
45         print('HIT!')
46         print(mouse_pos)
47         ball_bounce()
48     else:
49         print('NO!')
50
51 running = True
52 while running:
53     # 遊戲事件偵測 -----
54     for event in pygame.event.get():
55         if event.type == pygame.QUIT:
56             running = False
57         # 當滑鼠事件發生
58         if event.type == pygame.MOUSEBUTTONDOWN:
59             if event.button == 1: # 按下左鍵時
60                 check_hit_ball(event.pos) # 確認滑鼠有沒有按到球
61
62     # 動畫控制 -----
63     clock.tick(60)
64     ball_animation()
65
66     # 更新畫面 -----
67     canvas.fill(pygame.Color('WHITE'))
68     pygame.draw.circle(canvas, pygame.Color('LIGHTBLUE'), (x, y), r)
69     screen.blit(canvas, (0, 0))
70     pygame.display.update()
71     # -----
72
73 pygame.quit()

```



## 滑鼠輸入事件: 踢球



## 1. 調整加速度

```
6  SCREEN_SIZE = (800, 600)
7  screen = pygame.display.set_mode(SCREEN_SIZE)
8
9  x = 400
10 y = 300
11 r = 45
12 vx = 0.0
13 vy = 0.0
14 ax = 0.0
15 # ay = 0.88
16 ay = 0.3 # 為了測試把重力加速度先調小
```

- 為了測試方便, **重力加速度 ay** 先調小(大約在 **0.3** 以下)

## 2. 函式:兩點距離的平方、讓球彈起來

```
distance_square(pos1, pos2)
```

- `distance_square`(座標位置1, 座標位置2)
  - 座標位置(x, y)
- `distance_square`會回傳這兩個位置的距離平方
  - 距離的平方 =  $(x1-x2)^2 + (y1-y2)^2$

```
ball_bounce()
```

- 直接改動vy, 讓球彈起

## 2. 函式:兩點距離的平方、讓球彈起來

```
def distance_square(pos1, pos2):  
    x1 = pos1[0]  
    y1 = pos1[1]  
    x2 = pos2[0]  
    y2 = pos2[1]  
    return (x1-x2)*(x1-x2) + (y1-y2)*(y1-y2)
```

```
def ball_bounce():  
    global vy  
    vy = -20.0 # 球往上飛
```



### 3. 函式:檢查滑鼠有沒有點到球

- `check_hit_ball(滑鼠位置)`:  
依照滑鼠位置決定球要不要彈起來

呼叫 `distance_square(滑鼠位置, 圓心位置)`

得到

滑鼠與圓心的  
距離<sup>2</sup>

$\geq \text{半徑}^2$

輸出: 'NO!'

$< \text{半徑}^2$

輸出: 'HIT!'、滑鼠座標

呼叫 `ball_bounce()`

## 4. 判斷滑鼠右鍵有沒有按下去

### 顯示視窗 - 遊戲結束

```
# 是否遊戲結束
running = True
while running:
    # -----遊戲事件偵測-----
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

pygame.quit()
```

- 當按下右上角的  會關閉程式
- 請不要嘗試移掉這一部分，否則你將會大難臨頭



4.

## 顯示視窗 - 遊戲結束

```
# 是否遊戲結束
running = True
while running:
    # -----遊戲事件偵測-----
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

pygame.quit()
```

把判斷滑鼠的程式碼放在這



4.

## 顯示視窗 - 遊戲結束

```
# 是否遊戲結束
running = True
while running:
    # -----遊戲事件偵測-----
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
```

- 當按下
- 請不要 `pygame.quit()`



4.

## 顯示視窗 - 遊戲結束

```
# 是否遊戲結束
running = True
while running:
    # -----遊戲事件偵測-----
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
    # 當滑鼠事件發生
    if event.type == pygame.MOUSEBUTTONDOWN:
        if event.button == 1:
            check_hit_ball(event.pos)
```

- 當按下
- 請不要 `pygame.quit()`





## 4. 判斷滑鼠右鍵有沒有按下去

```
57 # 當滑鼠事件發生
58 if event.type == pygame.MOUSEBUTTONDOWN:
59     if event.button == 1: # 按下左鍵時
60         check_hit_ball(event.pos) # 確認滑鼠有沒有按到球
```

- **if event.type == pygame.MOUSEBUTTONDOWN**

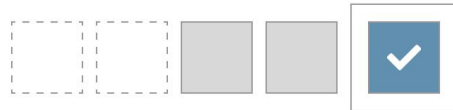
- 翻譯:如果偵測到滑鼠點擊

- 事件

- 事件種類:滑鼠移動、滑鼠點擊、鍵盤按下、關掉視窗
- 每種事件各自會有不同的資料:

滑鼠點擊 → 按了哪個鍵、滑鼠點擊位置

鍵盤按下 → 按了哪個鍵



## 4. 判斷滑鼠右鍵有沒有按下去

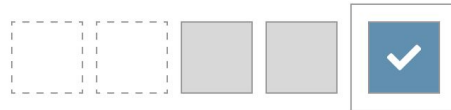
```
57 # 當滑鼠事件發生
58 if event.type == pygame.MOUSEBUTTONDOWN:
59     if event.button == 1: # 按下左鍵時
60         check_hit_ball(event.pos) # 確認滑鼠有沒有按到球
```

- `if event.type == pygame.MOUSEBUTTONDOWN`

- 翻譯:如果 事件種類是 滑鼠點擊

- 事件

- 事件種類:滑鼠移動、滑鼠點擊、鍵盤按下、關掉視窗
- 每種事件各自會附帶不同的資料:  
滑鼠點擊 → 按了哪個鍵、滑鼠點擊位置  
鍵盤按下 → 按了哪個鍵





## 4. 判斷滑鼠右鍵有沒有按下去

```
57         # 當滑鼠事件發生
58         if event.type == pygame.MOUSEBUTTONDOWN:
59             if event.button == 1: # 按下左鍵時
60                 check_hit_ball(event.pos) # 確認滑鼠有沒有按到球
```

- `if event.type == pygame.MOUSEBUTTONDOWN:` 如果滑鼠點擊
- `if event.button == 1`
  - 翻譯: 如果 滑鼠點擊時的按鍵是 左鍵
  - 對應關係:

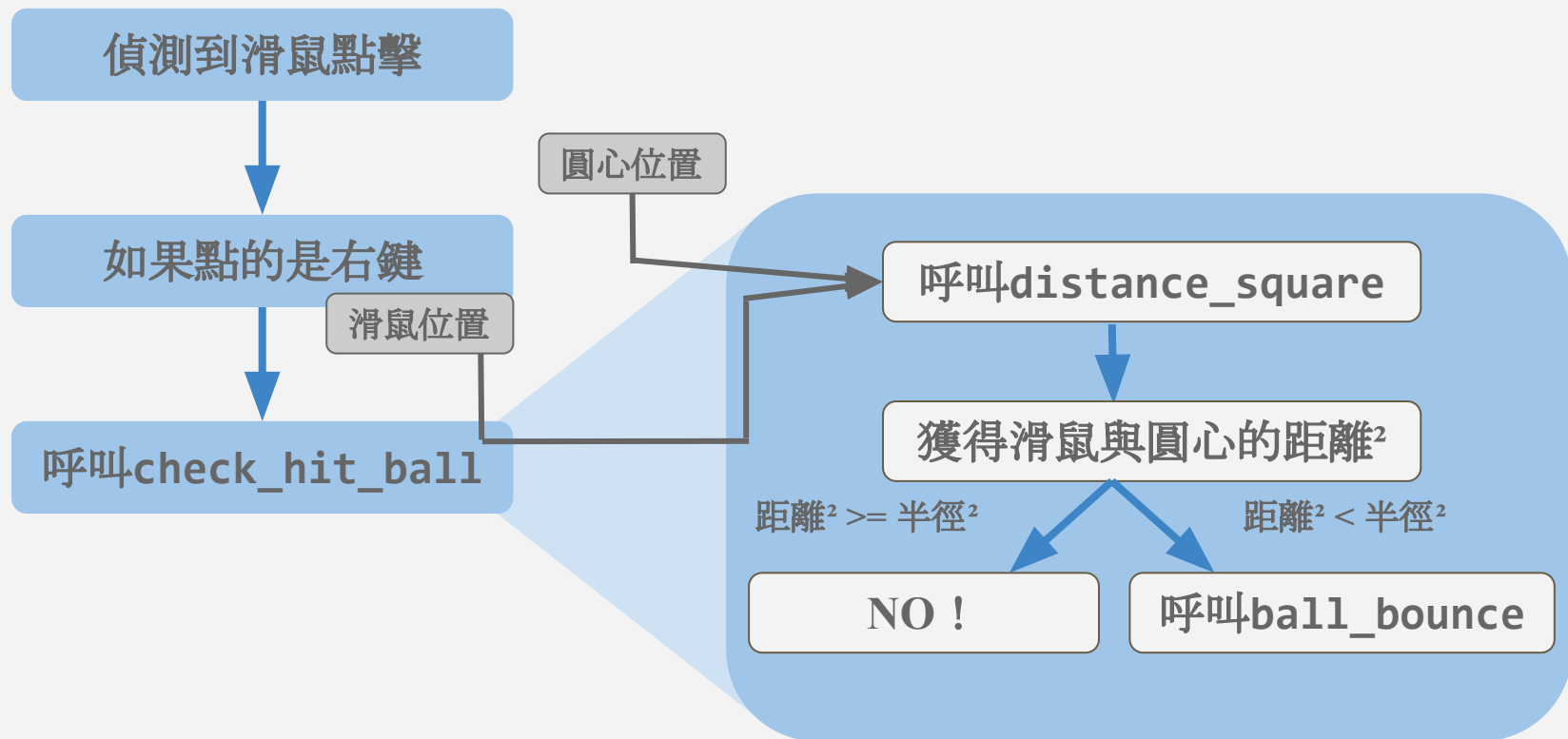
1	2	3	4	5
左鍵	滾輪(按下)	右鍵	滾輪(上滑)	滾輪(下滑)

## 4. 判斷滑鼠右鍵有沒有按下去

```
57     # 當滑鼠事件發生
58     if event.type == pygame.MOUSEBUTTONDOWN:
59         if event.button == 1: # 按下左鍵時
60             check_hit_ball(event.pos) # 確認滑鼠有沒有按到球
```

- `if event.type == pygame.MOUSEBUTTONDOWN:` 如果滑鼠點擊
- `if event.button == 1:` 如果點擊的是左鍵
- `check_hit_ball(event.pos)`
  - 呼叫 `check_hit_ball`
  - 將滑鼠點擊時的位置 (`event.pos`) 傳給 `check_hit_ball` 判斷

## 滑鼠輸入事件: 踢球

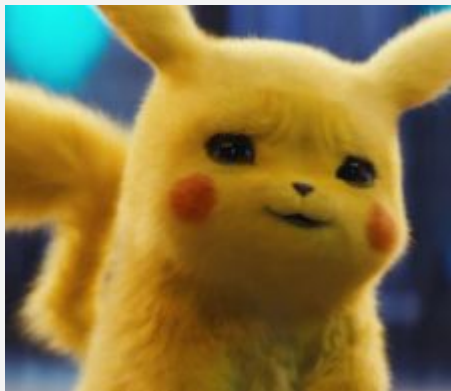


## 其他事件種類 & 附帶的資料

事件種類 event.type	滑鼠按下 MOUSEBUTTONDOWN	滑鼠放開 MOUSEBUTTONUP	滑鼠移動 MOUSEMOTION	鍵盤按下 KEYDOWN	鍵盤放開 KEYUP	視窗關閉 QUIT
按下/放開的按鍵	event.button		event.buttons	event.key		
滑鼠位置	event.pos					
滑鼠移動速度			event.rel			

隨機：讓球會亂噴

才不告訴逆勒



如何產生一定範圍內隨機的數？

## 如何產生一定範圍內隨機的數？

```
1  import pygame
2  import random # 引入隨機模組
3  pygame.init()
```

- 先引入 random 模組！



## 如何產生一定範圍內隨機的數？

```
random.uniform(-5.5, 5.5)
```

- 上述的範例，可以產生  $-5.5 \sim 5.5$  之間隨機的數
  - 例： $-5.43$ 、 $4.26$ 、 $3.14$ 、 $0.3\dots$  都是有可能出現的數

## 小挑戰:讓球不只會往上彈

- 要求1:
  - 當球被按到時, 讓球產生一個隨機的**水平速度(vx)**值
  - 這個隨機的速度值介在 **-5.0 ~ 5.0** 之間
- 要求2:
  - 當球被按到時, 讓球產生一個隨機的**垂直速度(vy)**值
  - 這個隨機的速度值介在 **-21.0 ~ -20.0** 之間
- 只能修改 **ball\_bounce()** 函式

## 小挑戰:讓球不只會往上彈 (解答)

- global 要記得加!!

```
# 球彈上去的效果
def ball_bounce():
    global vx, vy
    vx = random.uniform(-5.5, 5.5)    # 球的 x 速度是介在 -5.0 ~ 5.0 之間的隨機數字
    vy = random.uniform(-21.0, -20.0) # 球的 y 速度是介在 -21.0 ~ -20.0 之間的隨機數字
```



## 動畫3：碰壁反彈

## 動畫3: 碰壁反彈 - 事前準備

```
6  SCREEN_SIZE = (800, 600)
7  screen = pygame.display.set_mode(SCREEN_SIZE)
8
9  x = 400
10 y = 300
11 r = 45
12 vx = 0.0
13 vy = 0.0
14 ax = 0.0
15 # ay = 0.88
16 ay = 0.3 # 為了測試把重力加速度先調小
```

- 為了測試方便, **重力加速度 ay** 一樣先維持在 **0.3**

## 動畫3: 碰壁反彈 - ball\_animation()修正

code在這裡

<http://codepad.org/eifVPj9Z>

- 修正ball\_animation()的內容，讓球碰到左右的視窗邊界會有反彈的效果。
- 這部分程式比較複雜，因此直接提供程式碼給大家複製
- 直接覆蓋原本的ball\_animation()即可。

```
def ball_animation():  
    global x, y, vx, vy  
    x_max = SCREEN_SIZE[0] - r # 球剛好碰到右壁時  
    x_min = r # 球剛好碰到左壁時  
  
    # 牛頓運動定律 -----  
  
    x = int(x + vx)  
    y = int(y + vy)  
    vx = vx + ax  
    vy = vy + ay  
  
    # 彈性碰撞 -----  
  
    if x > x_max: # 碰到右邊的牆  
        ex = x - x_max  
        t = ex / vx  
        vx = -vx  
        x = int(x_max - vx*(1-t))  
        if abs(vx) < 1:  
            vx = 0  
            x = x_max  
  
    if x < x_min: # 碰到左邊的牆  
        ex = x - x_min  
        t = ex / vx  
        vx = -vx  
        x = int(x_min - vx*(1-t))  
        if abs(vx) < 1:  
            vx = 0  
            x = x_min
```

## 動畫3: 碰壁反彈

- 改完`ball_animation()`就可以試著執行遊戲囉

加入遊戲啟始畫面

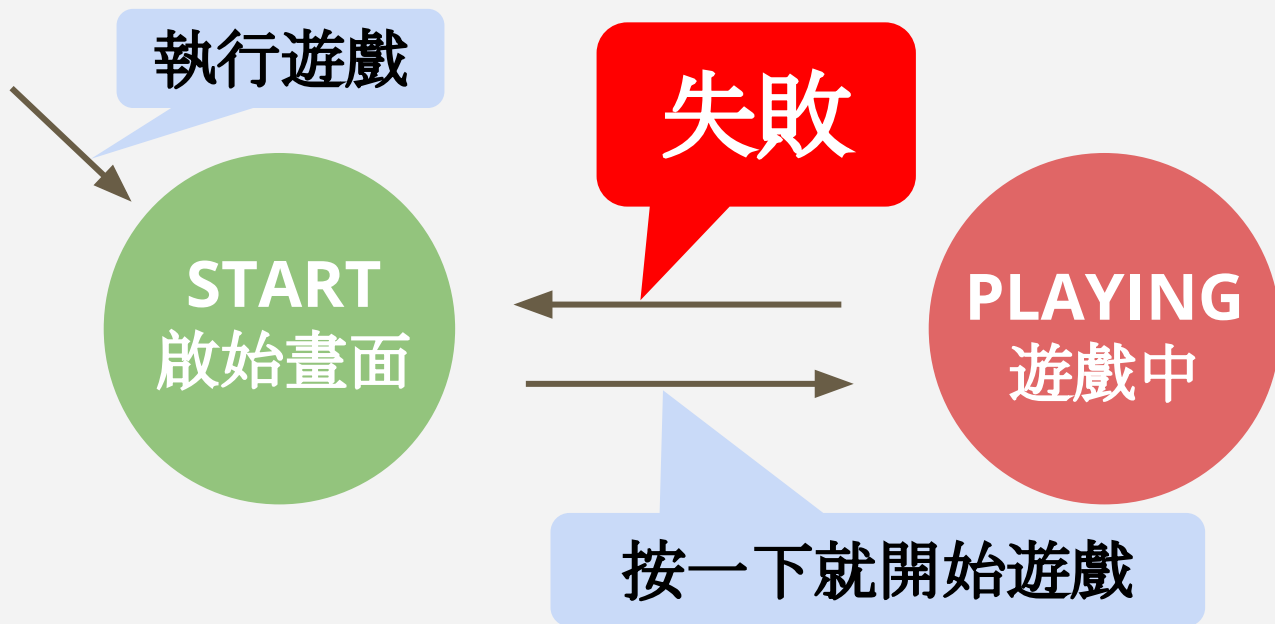


電腦要怎麼決定  
當下要呈現『啟始畫面』還是『遊戲中』的狀態？

電腦要怎麼決定  
當下要呈現『啟始畫面』還是『遊戲中』的狀態？

⇒ 遊戲狀態 State

## 加入遊戲啟始畫面 - state 變數



## 加入遊戲啟始畫面 - state 變數

執行遊戲

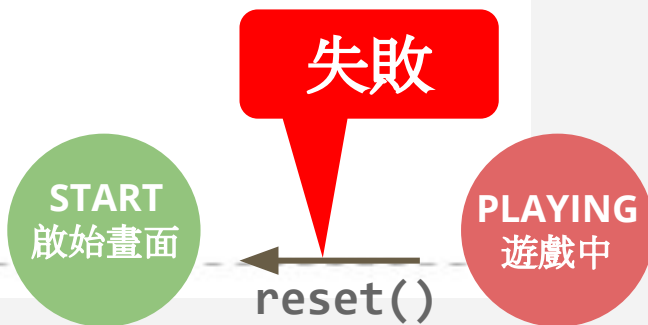
START  
啟始畫面

```
9   x = 400
10  y = 300
11  r = 45
12  vx = 0.0
13  vy = 0.0
14  ax = 0.0
15  ay = 0.88
16
17  # state 變數表示著當下的遊戲狀態：[START=起始畫面]，[PLAYING=遊戲中]
18  state = 'START'
19
20  canvas = pygame.Surface(SCREEN_SIZE)
21  clock = pygame.time.Clock()
```

- 先初始化一個新的變數叫作 **state**
- **state** 的可能值為 '**START**' 以及 '**PLAYING**'
- 用來表示當下的**遊戲狀態**

## 加入遊戲啟始畫面 - reset 函式

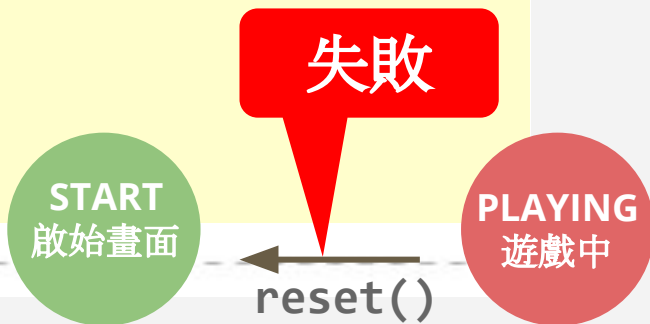
```
23  # 重置遊戲狀態
24  def reset():
25      global x, y, vx, vy, state
26      state = 'START' # 跳回[起始畫面]
27      # 球的位置、速度重置 -----
28      x = 400
29      y = 300
30      vx = 0.0
31      vy = 0.0
32      # -----
```



- 寫一個新的函式叫作 **reset()**, 內容為：
  - 把遊戲狀態 **state** 改成 **START** (啟始狀態)

## 加入遊戲啟始畫面 - reset 函式

```
23  # 重置遊戲狀態
24  def reset():
25      global x, y, vx, vy, state
26      state = 'START' # 跳回[起始畫面]
27      # 球的位置、速度重置 -----
28      x = 400
29      y = 300
30      vx = 0.0
31      vy = 0.0
32  # -----
```

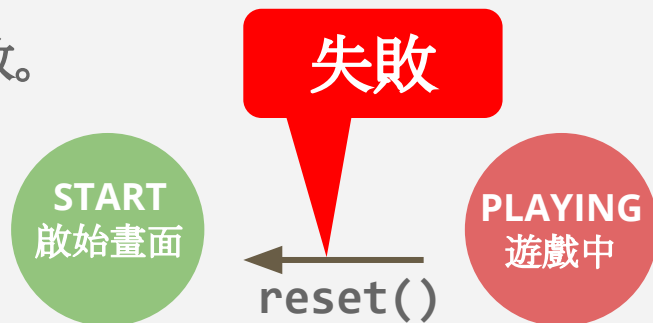


- 寫一個新的函式叫作 **reset()**, 內容為：
  - 把球的**位置**、**速度**做一個重置的動作

## 加入遊戲啟始畫面 - check\_game\_over 函式

```
# 檢查球有沒有掉下去，掉下去就 game over
def check_game_over():
    y_max = SCREEN_SIZE[1] + r # 當球完全掉到視窗範圍的下界之下時，此時的 y座標
    if y > y_max: # 當球掉下去，就 game over 重置遊戲
        reset()
```

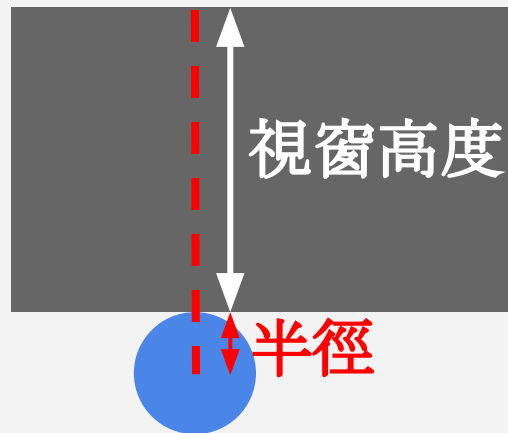
- 寫一個新的函式 `check_game_over()`
- 用來檢查球是不是掉下去了，掉下去就失敗。



## 加入遊戲啟始畫面 - check\_game\_over 函式

```
# 檢查球有沒有掉下去，掉下去就 game over
def check_game_over():
    y_max = SCREEN_SIZE[1] + r # 當球完全掉到視窗範圍的下界之下時，此時的 y座標
    if y > y_max: # 當球掉下去，就 game over 重置遊戲
        reset()
```

- SCREEN\_SIZE[0]: 視窗**寬度**
- SCREEN\_SIZE[1]: 視窗**高度**
- 當 **y 位置**比視窗的最下界再**更下面一個半徑**的距離，則球就完全消失在視窗之外，失敗。

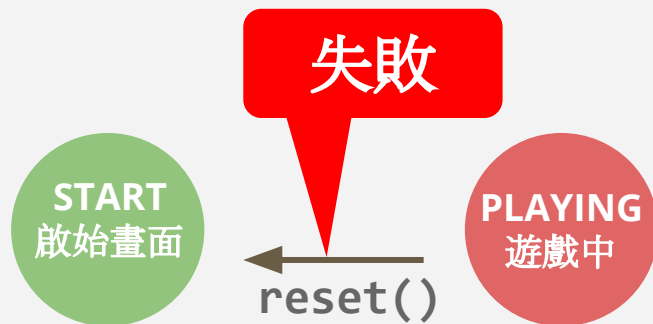




## 加入遊戲啟始畫面 - check\_game\_over 函式

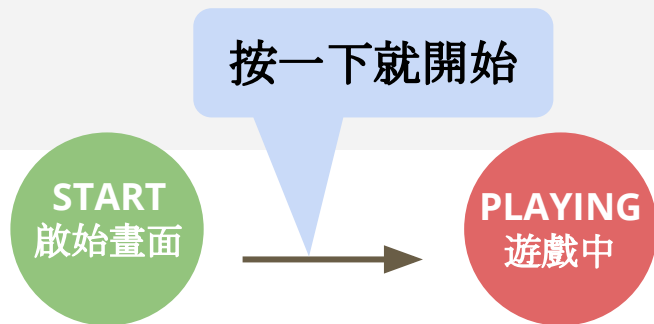
```
# 檢查球有沒有掉下去，掉下去就 game over
def check_game_over():
    y_max = SCREEN_SIZE[1] + r # 當球完全掉到視窗範圍的下界之下時，此時的 y座標
    if y > y_max: # 當球掉下去，就 game over 重置遊戲
        reset()
```

- 呼叫剛剛寫好的 reset() 重置遊戲



## 加入遊戲啟始畫面 - 按一下就開始

```
94 while running:
95     # 遊戲事件偵測 -----
96     for event in pygame.event.get():
97         if event.type == pygame.QUIT:
98             running = False
99
100         if event.type == pygame.MOUSEBUTTONDOWN:
101             if event.button == 1: # 按下左鍵時
102                 check_hit_ball(event.pos) # 無論在什麼遊戲狀態，只要按到球，球就會彈上去
103                 if state == 'START': # 如果遊戲在[啟始畫面],
104                     state = 'PLAYING' # 則進入[遊戲中]的狀態
105
106     # 動畫控制 -----
```



- 當滑鼠按下左鍵時(任何地方), 如果此時在啟始畫面 'START', 則進入遊戲中 'PLAYING' 的狀態

## 加入遊戲啟始畫面 - 根據不同 state 做不同的動畫控制

```
106     # 動畫控制 -----
107     clock.tick(60)
108     # 不同的遊戲狀態，畫面的顯示也不同 -----
109     if state == 'START': # [起始畫面] -----
110         canvas.fill(pygame.Color('BLACK')) # 畫布背景為黑色
111     elif state == 'PLAYING': # [遊戲中] -----
112         ball_animation() # 只有在[遊戲中]，球才會動
113         check_game_over() # 只有在[遊戲中]，會去檢查有沒有球掉下去 game over
114         canvas.fill(pygame.Color('WHITE')) # 畫布背景為白色
```

- 啟始畫面 'START' 時:將畫布背景設為黑色

## 加入遊戲啟始畫面 - 根據不同 state 做不同的動畫控制

```
106 # 動畫控制 -----
107 clock.tick(60)
108 # 不同的遊戲狀態，畫面的顯示也不同 -----
109 if state == 'START': # [起始畫面] -----
110     canvas.fill(pygame.Color('BLACK')) # 畫布背景為黑色
111 elif state == 'PLAYING': # [遊戲中] -----
112     ball_animation() # 只有在[遊戲中]，球才會動
113     check_game_over() # 只有在[遊戲中]，會去檢查有沒有球掉下去 game over
114     canvas.fill(pygame.Color('WHITE')) # 畫布背景為白色
```

- 遊戲中 'PLAYING' 時：
  - `ball_animation()`：只有在遊戲中的時候，需要計算球的移動

## 加入遊戲啟始畫面 - 根據不同 state 做不同的動畫控制

```
106 # 動畫控制 -----
107 clock.tick(60)
108 # 不同的遊戲狀態，畫面的顯示也不同 -----
109 if state == 'START': # [起始畫面] -----
110     canvas.fill(pygame.Color('BLACK')) # 畫布背景為黑色
111 elif state == 'PLAYING': # [遊戲中] -----
112     ball_animation() # 只有在[遊戲中]，球才會動
113     check_game_over() # 只有在[遊戲中]，會去檢查有沒有球掉下去 game over
114     canvas.fill(pygame.Color('WHITE')) # 畫布背景為白色
```

- 遊戲中 'PLAYING' 時：
  - `check_game_over()`：只有在遊戲中的時候，需要判斷是否失敗

## 加入遊戲啟始畫面 - 根據不同 state 做不同的動畫控制

```
106 # 動畫控制 -----
107 clock.tick(60)
108 # 不同的遊戲狀態，畫面的顯示也不同 -----
109 if state == 'START': # [起始畫面] -----
110     canvas.fill(pygame.Color('BLACK')) # 畫布背景為黑色
111 elif state == 'PLAYING': # [遊戲中] -----
112     ball_animation() # 只有在[遊戲中]，球才會動
113     check_game_over() # 只有在[遊戲中]，會去檢查有沒有球掉下去 game over
114     canvas.fill(pygame.Color('WHITE')) # 畫布背景為白色
```

- 遊戲中 'PLAYING' 時：
  - 畫布的背景設為白色

- 寫到這裡，簡單的踢足球小遊戲陽春版已經大致完成了啦！

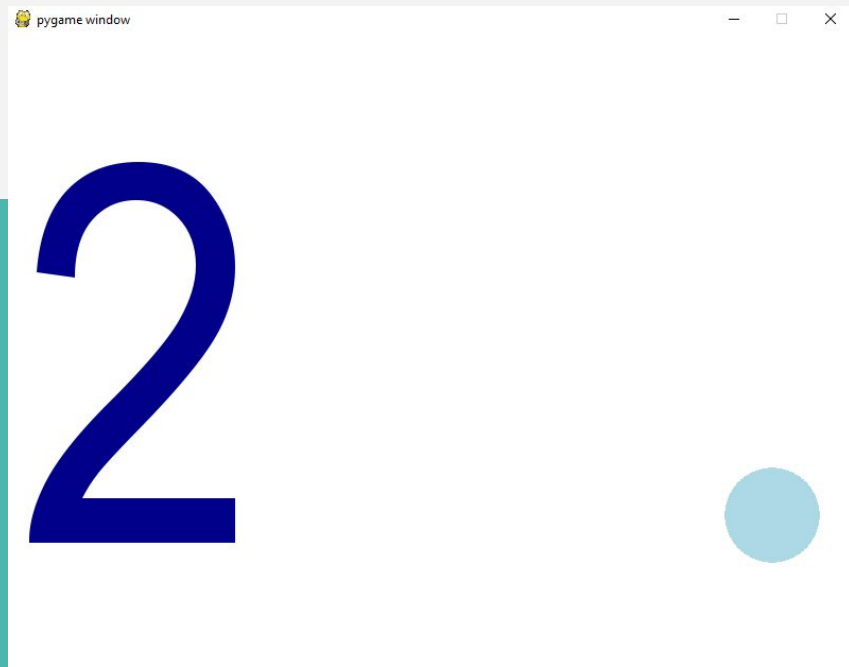
- 寫到這裡，簡單的踢足球小遊戲陽春版已經大致完成了啦！
- 但，總覺得還可以再加一些元素...



- 寫到這裡, 簡單的踢足球小遊戲陽春版已經大致完成了啦 !
- 但, 總覺得還可以再加一些元素...
  - 算分數?

- 寫到這裡, 簡單的踢足球小遊戲陽春版已經大致完成了啦 !
- 但, 總覺得還可以再加一些元素...
  - 算分數?
  - 載入圖片?

# 進階：顯示文字-計分功能



## 計分功能

```
11  vx = 0.0
12  vy = 0.0
13  ax = 0.0
14  ay = 0.88
15
16  state = 'START'
17  score = 0      # score 遊戲分數
```

- 先建立一個新的變數 **score**
- 並將它設為 0

**pygame 要怎麼顯示文字?**

# pygame 要怎麼顯示文字?

`pygame.font.SysFont()`

建立  
字型樣本

`render()`

用字型樣本  
建立文字物件

`blit()`

將文字物件  
貼到畫布上

## 計分功能 - 建立字型樣本

pygame.font.SysFont()

建立  
字型樣本

```
16 state = 'START'
17 score = 0      # score 遊戲分數
18
19 canvas = pygame.Surface(SCREEN_SIZE)
20 clock = pygame.time.Clock()
21 myfont = pygame.font.SysFont('Arial', 500)    # 字型設定
```

- 設定字型樣本：pygame.font.SysFont( 字型\*, 大小 )
  - myfont 就是我們建立的字型樣本

# 計分功能 - 建立文字物件

render()

用字型樣本  
建立文字物件

```
93  running = True
94  while running:
...      .....
107      # 動畫控制 -----
108      clock.tick(60)
109      if state == 'START':
110          canvas.fill(pygame.Color('BLACK'))
111      elif state == 'PLAYING':
112          ball_animation()
113          check_game_over()
114          canvas.fill(pygame.Color('WHITE'))
115
116      # 更新畫面 -----
117
118      score_text = myfont.render(str(score), True, pygame.Color('DARKBLUE')) # 新增分數的文字物件
119      canvas.blit(score_text, (10, 10)) # 把文字物件貼到畫布上
120
121      pygame.draw.circle(canvas, pygame.Color('LIGHTBLUE'), (x, y), r)
```

- 執行時每一幀(1/60秒)都建立一個新的文字物件
- 設定文字物件：字型樣本.render(內容, 是否消除鋸齒, 顏色)
  - text 就是我們創造的文字物件



# 計分功能 - 建立文字物件

render()

用字型樣本  
建立文字物件

```
93  running = True
94  while running:
...      .....
107      # 動畫控制 -----
108      clock.tick(60)
109      if state == 'START':
110          canvas.fill(pygame.Color('BLACK'))
111      elif state == 'PLAYING':
112          ball_animation()
113          check_game_over()
114          canvas.fill(pygame.Color('WHITE'))
115
116      # 更新畫面 -----
117
118      score_text = myfont.render(str(score), True, pygame.Color('DARKBLUE')) # 新增分數的文字物件
119      canvas.blit(score_text, (10, 10)) # 把文字物件貼到畫布上
120
121      pygame.draw.circle(canvas, pygame.Color('LIGHTBLUE'), (x, y), r)
```

- 執行時每一幀(1/60秒)都建立一個新的文字物件
- 設定文字物件：字型樣本.render(內容, 是否消除鋸齒, 顏色)
  - text 就是我們創造的文字物件

# 計分功能 - 建立文字物件

```
93  running = True
94  while running:
...      .....
107      # 動畫控制 -----
108      clock.tick(60)
109      if state == 'START':
110          canvas.fill(pygame.Color('BLACK'))
111      elif state == 'PLAYING':
112          ball_animation()
```

render()

用字型樣本  
建立文字物件

str(score)

```
121  pygame.draw.circle(canvas, pygame.Color('LIGHTBLUE'), (x, y), r)
```

- 設定文字物件：字型樣本.render(內容, 是否消除鋸齒, 顏色)
  - text 就是我們創造的文字物件
- 建立文字物件時，顯示的內容一定要是字串，因此必須用str()工具來將score轉換成字串

# 計分功能 - 建立文字物件

blit()

將文字物件  
貼到畫布上

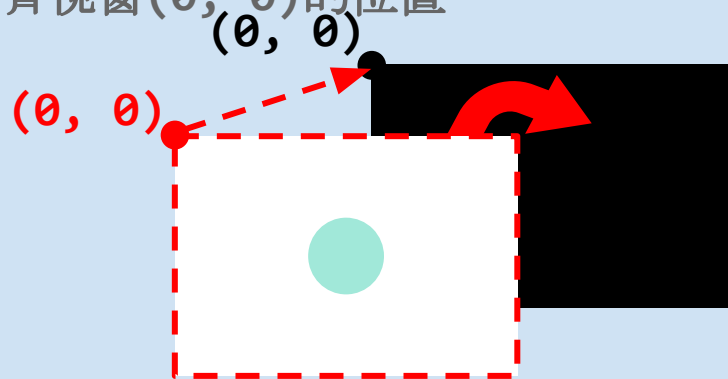
```
93  running = True
94  while running:
...      .....
107      # 動畫控制 -----
108      clock.tick(60)
109      if state == 'START':
110          canvas.fill(pygame.Color('BLACK'))
111      elif state == 'PLAYING':
112          ball_animation()
113          check_game_over()
114          canvas.fill(pygame.Color('WHITE'))
115
116      # 更新畫面 -----
117
118      score_text = myfont.render(str(score), True, pygame.Color('DARKBLUE')) # 新增分數的文字物件
119      canvas.blit(score_text, (10, 10)) # 把文字物件貼到畫布上
120
121      pygame.draw.circle(canvas, pygame.Color('LIGHTBLUE'), (x, y), r)
```

- 執行時每一幀(1/60秒)都建立一個新的文字物件
- 設定文字物件：字型樣本.render(內容, 是否消除鋸齒, 顏色)
  - text 就是我們創造的文字物件

# 回憶 - 視窗貼貼

```
21 # 更新畫面 -----
22 canvas.fill(pygame.Color('WHITE'))
23 pygame.draw.circle(canvas, pygame.Color('LIGHTBLUE'), (x, y), r)
24 screen.blit(canvas, (0, 0))
25 pygame.display.update()
26 # -----
27 pygame.quit()
```

- `screen.blit`(圖層物件, 左上角座標位置)
  - 將圖層(canvas)貼到視窗(screen)上面
  - 左上角座標位置 =  $(0, 0)$ : 圖層原點會對齊視窗 $(0, 0)$ 的位置
- 注意: 原點  $(0, 0)$  在視窗左上角!



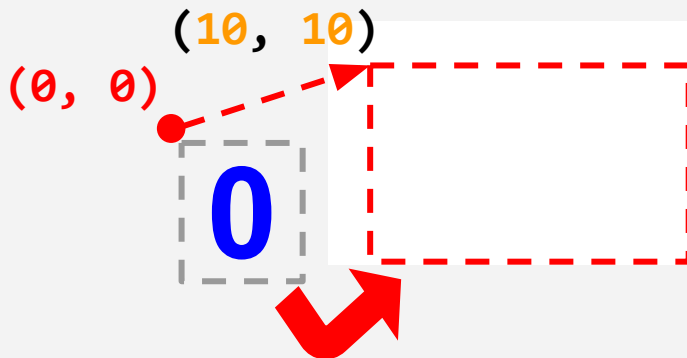
blit()

## 計分功能 - 畫布貼貼

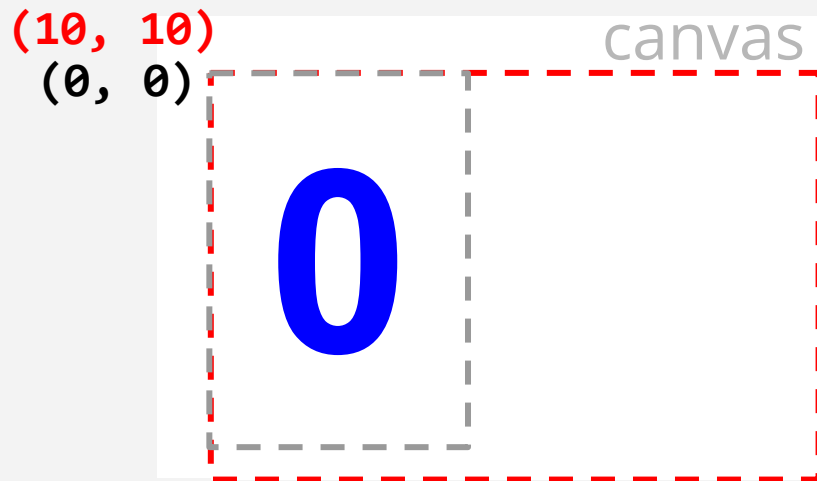
將文字物件  
貼到畫布上

```
canvas.blit(score_text, (10, 10))
```

- `canvas.blit`(圖層物件, 左上角座標位置)
  - 將文字物件(`score_text`)貼到畫布(`canvas`)上面
  - 文字物件原點會對齊畫布(10, 10)的位置
- 注意: 原點 (0, 0) 在畫布左上角!



畫布貼貼 - `canvas.blit(score_text, (10, 10))`



如何計分？

如何計分？

⇒找到『判斷是否按到球』的函式！



```
def check_hit_ball(mouse_pos):
```

## 計分功能 - 踢一球加一分

```
def check_hit_ball(mouse_pos):  
    global score  
    ball_pos = [x, y]  
    if r * r > distance_square(mouse_pos, ball_pos):  
        print('HIT!')  
        print(mouse_pos)  
        ball_bounce()  
    else:  
        print('NO!')
```

## 計分功能 - 踢一球加一分

```
def check_hit_ball(mouse_pos):  
    global score  
    ball_pos = [x, y]  
    if r * r > distance_square(mouse_pos, ball_pos):  
        print('HIT!')  
        print(mouse_pos)  
        ball_bounce()  
    else:  
        print('NO!')
```

A ←

B ←

C ←

- 小小挑戰:要在哪裡加一行,才能讓每踢一下球,分數就加一分?
- A? B? 還是C?

## 計分功能 - 踢一球加一分

```
def check_hit_ball(mouse_pos):  
    global score  
    ball_pos = [x, y]  
    if r * r > distance_square(mouse_pos, ball_pos):  
        print('HIT!')  
        print(mouse_pos)  
        ball_bounce()  
    else:  
        print('NO!')
```

- 小小挑戰:要在哪裡加一行,才能讓每踢一下球,分數就加一分?
- A? B? 還是C?
- 答案:B

## 計分功能 - 踢一球加一分

```
def check_hit_ball(mouse_pos):  
    global score  
    ball_pos = [x, y]  
    if r * r > distance_square(mouse_pos, ball_pos):  
        print('HIT!')  
        print(mouse_pos)  
        ball_bounce()  
        score = score + 1 # 踢一球, 加一分  
    else:  
        print('NO!')
```

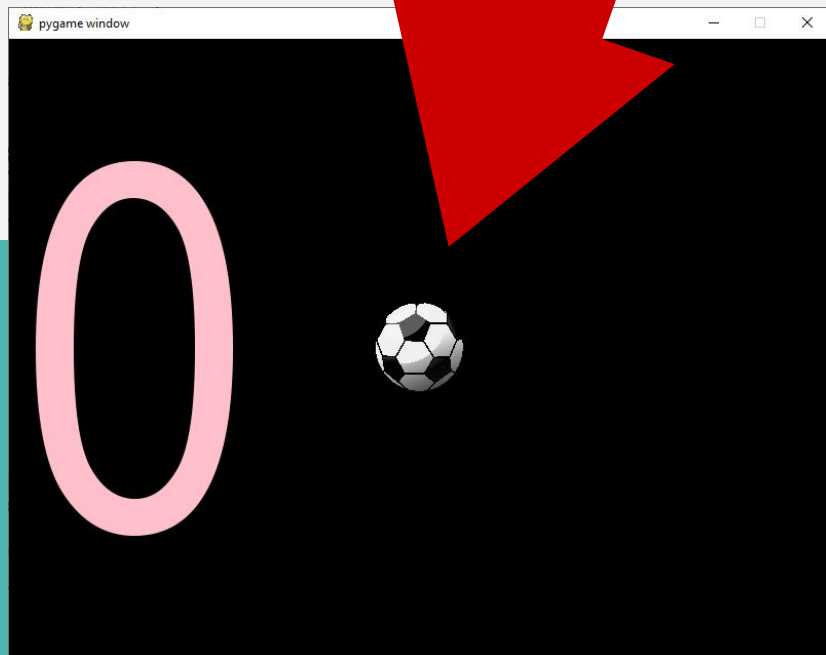
- 小小挑戰:要在哪裡加一行,才能讓每踢一下球,分數就加一分?
- A? B? 還是C?
- 答案:B

## 計分功能 - 遊戲開始, 分數歸零

```
if event.type == pygame.MOUSEBUTTONDOWN:
    if event.button == 1: # 按下左鍵時
        check_hit_ball(event.pos)
        if state == 'START':          # 如果遊戲在[啟始畫面]
            state = 'PLAYING'
            score = 0                  # -- 分數歸零
```

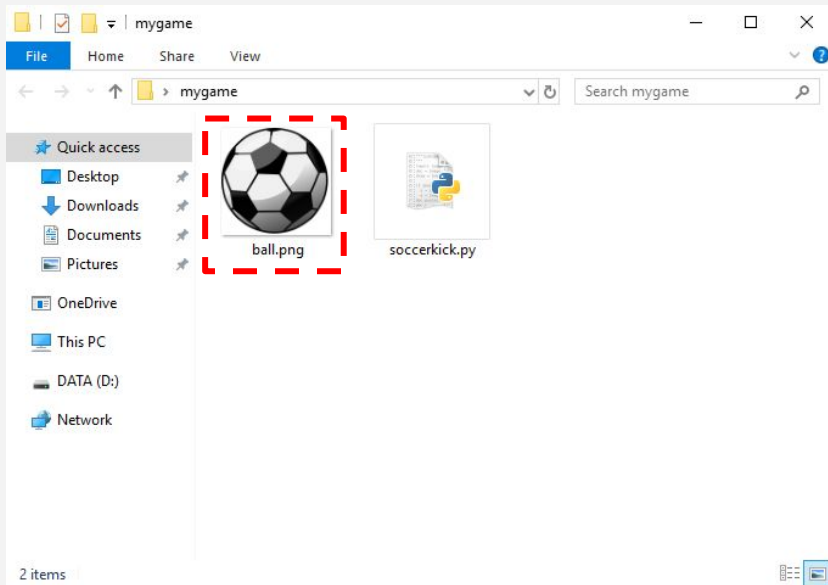
- 當遊戲狀態 `state` 從 啟始畫面 'START' 進入遊戲中 'PLAYING' 時:
  - 讓分數 `score` 歸零

# 進階：載入圖片



## 進階:載入圖片

- 原本提供的資料夾裡, 已經有一個足球的圖片檔在裡面, 並且與 `soccerkick.py` 在同一個資料夾層底下





## 載入圖片

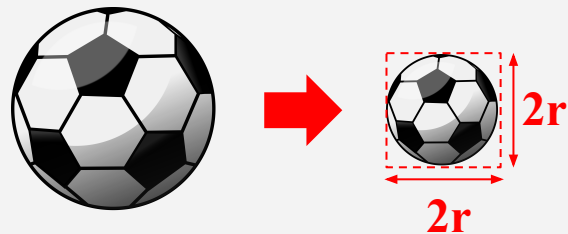
```
canvas = pygame.Surface(SCREEN_SIZE)
clock = pygame.time.Clock()
myfont = pygame.font.SysFont('Arial', 500)

picture = pygame.image.load('ball.png')
picture = pygame.transform.scale(picture, (r*2, r*2))

def reset(): ...
```

- `pygame.image.load(圖片檔案路徑)`
  - 把圖片載入
  - `picture` 現在是一個圖片物件

## 載入圖片



```
canvas = pygame.Surface(SCREEN_SIZE)
clock = pygame.time.Clock()
myfont = pygame.font.SysFont('Arial', 500)

picture = pygame.image.load('ball.png')
picture = pygame.transform.scale(picture, (r*2, r*2))

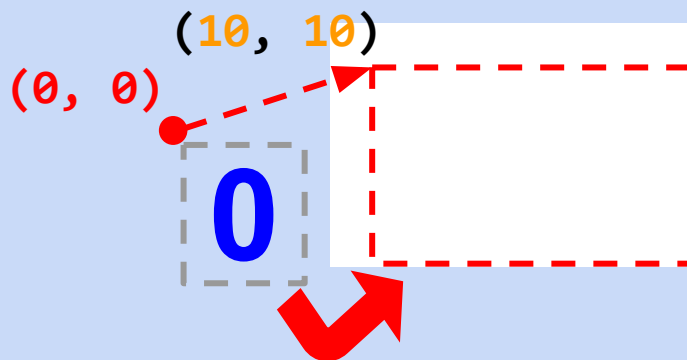
def reset(): ...
```

- `pygame.transform.scale`(圖片物件, 目標尺寸)
  - 把足球圖片的長寬調成: 球的直徑 $2r \times$ 直徑 $2r$

# 回憶 - 畫布貼貼

```
canvas.blit(score_text, (10, 10))
```

- `canvas.blit`(圖層物件, 左上角座標位置)
  - 將文字物件(`score_text`)貼到畫布(`canvas`)上面
  - 文字物件原點會對齊畫布(10, 10)的位置
- 注意: 原點 (0, 0) 在畫布左上角!

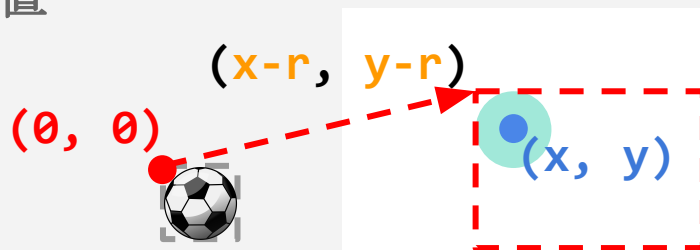


## 載入圖片 - 畫布貼貼

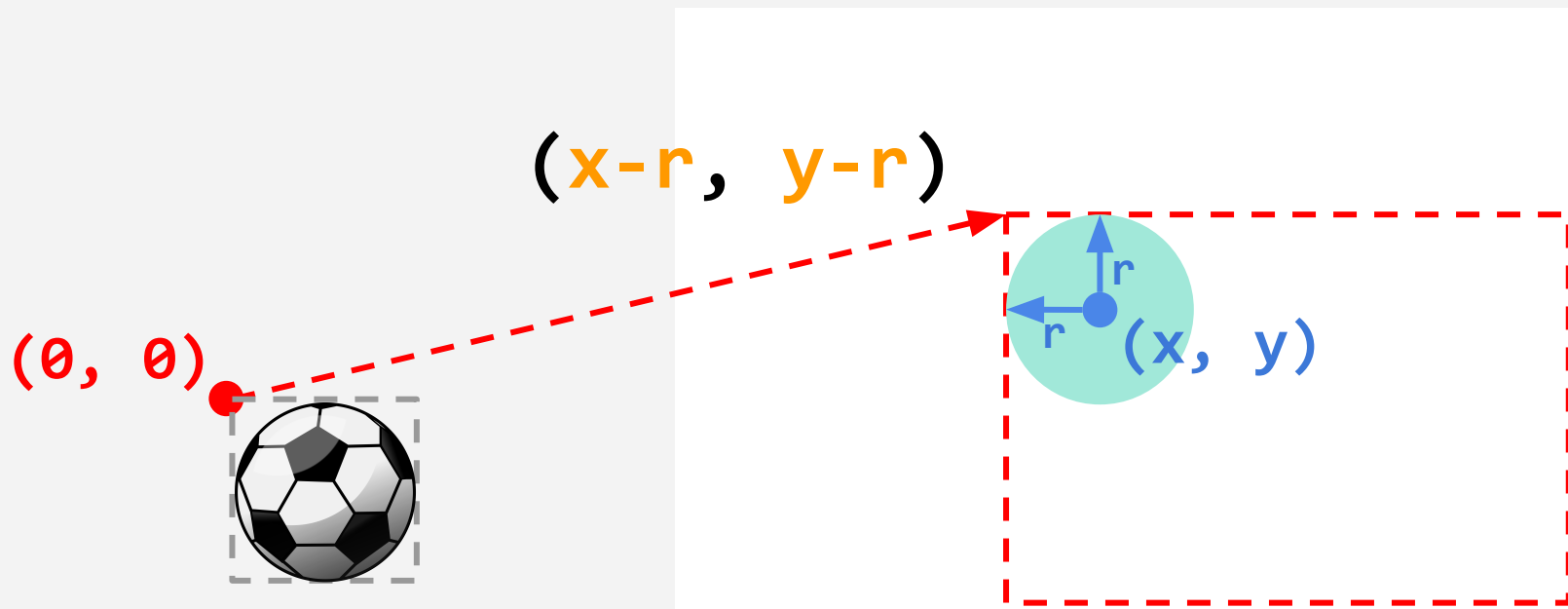
```
# 更新畫面 -----
score_text = myfont.render(str(score), True, pygame.Color('PINK'))
canvas.blit(score_text, (10, 10))
# pygame.draw.circle(canvas, pygame.Color('LIGHTBLUE'), (x, y), r)
canvas.blit(picture, (x-r, y-r)) # 把球的圖片根據球的位置座標貼在畫布上面
screen.blit(canvas, (0, 0))
pygame.display.update()
# -----
```

- **canvas.blit(圖層物件, 左上角座標位置)**
  - 將**圖片物件(picture)**貼到**畫布(canvas)**上面
  - **picture** 原點會對齊畫布(**x-r**, **y-r**)的位置

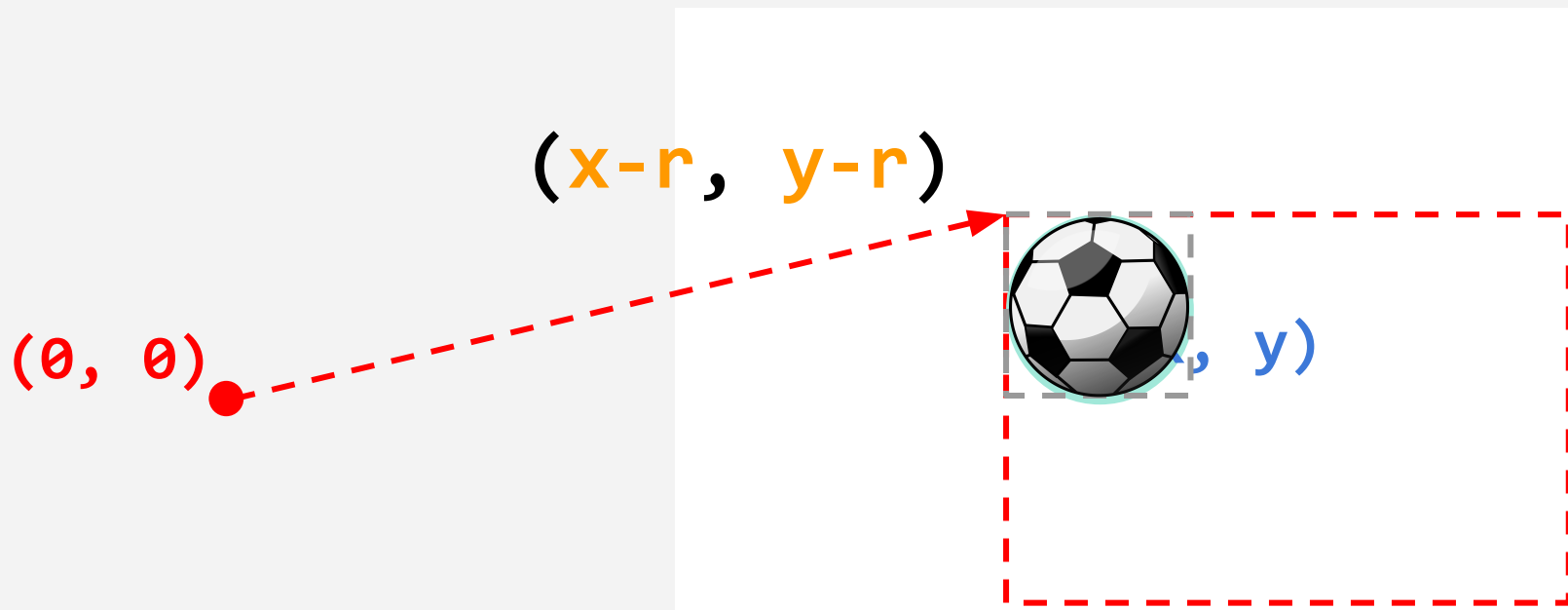
圖片的**左上角**  
對齊 **球**的**左上角**



## 載入圖片 - 畫布貼貼



## 載入圖片 - 畫布貼貼



```

1 import pygame
2 import random
3 pygame.init()
4
5 SCREEN_SIZE = (800, 600)
6 screen = pygame.display.set_mode(SCREEN_SIZE)
7
8 x = 400
9 y = 300
10 r = 45
11 vx = 0.0
12 vy = 0.0
13 ax = 0.0
14 ay = 0.88
15
16 state = 'START'
17 score = 0
18
19 canvas = pygame.Surface(SCREEN_SIZE)
20 clock = pygame.time.Clock()
21 myfont = pygame.font.SysFont('Arial', 500)
22 myfont_xs = pygame.font.SysFont('Arial', 30)
23 picture = pygame.image.load('ball.png')
24 picture = pygame.transform.scale(picture, (r*2, r*2))
25
26 def reset():
27     global x, y, vx, vy, state
28     state = 'START'
29     # 球的位置、速度重置 -----
30     x = 400
31     y = 300
32     vx = 0.0
33     vy = 0.0
34     # -----
35
36 def check_game_over():
37     y_max = SCREEN_SIZE[1] + r
38     if y > y_max:
39         reset()

```

```

40
41 def ball_animation():
42     global x, y, vx, vy
43     x_max = SCREEN_SIZE[0] - r # 球剛好碰到右壁時, 此時的球心 x 座標
44     x_min = r # 球剛好碰到左壁時, 此時的球心 x 座標
45
46     # 運動公式 -----
47
48     x = int(x + vx)
49     y = int(y + vy)
50     vx = vx + ax
51     vy = vy + ay
52
53     # 彈性碰撞 -----
54
55     if x > x_max: # 碰到右邊的牆
56         ex = x - x_max
57         t = ex / vx
58         vx = -vx
59         x = int(x_max - vx*(1-t))
60         if abs(vx) < 1:
61             vx = 0
62             x = x_max
63
64     if x < x_min: # 碰到左邊的牆
65         ex = x - x_min
66         t = ex / vx
67         vx = -vx
68         x = int(x_min - vx*(1-t))
69         if abs(vx) < 1:
70             vx = 0
71             x = x_min
72
73 def distance_square(pos1, pos2):
74     x1 = pos1[0]
75     y1 = pos1[1]
76     x2 = pos2[0]
77     y2 = pos2[1]
78     return (x1-x2)*(x1-x2) + (y1-y2)*(y1-y2)
79

```

```

80 def ball_bounce():
81     global vx, vy
82     vx = random.uniform(-5.0, 5.0)    # 球的 x 速度是介在 -5.0 ~ 5.0 之間的隨機數字
83     vy = random.uniform(-21.0, -20.0) # 球的 y 速度是介在 -21.0 ~ -20.0 之間的隨機數字
84
85 def check_hit_ball(mouse_pos):
86     global score
87     ball_pos = [x, y]
88     if r * r > distance_square(mouse_pos, ball_pos):
89         ball_bounce()
90         score = score + 1
91
92 running = True
93 while running:
94     # 遊戲事件偵測 -----
95     for event in pygame.event.get():
96         if event.type == pygame.QUIT:
97             running = False
98
99         if event.type == pygame.MOUSEBUTTONDOWN:
100             if event.button == 1:
101                 check_hit_ball(event.pos)
102                 if state == 'START':
103                     state = 'PLAYING'
104                 score = 0
105

```

```

92 running = True
93 while running:
94     # 遊戲事件偵測 -----
95     for event in pygame.event.get():
96         if event.type == pygame.QUIT:
97             running = False
98
99         if event.type == pygame.MOUSEBUTTONDOWN:
100             if event.button == 1:
101                 check_hit_ball(event.pos)
102                 if state == 'START':
103                     state = 'PLAYING'
104                 score = 0
105
106     # 動畫控制 -----
107     clock.tick(60)
108     if state == 'START':
109         canvas.fill(pygame.Color('BLACK'))
110         hint_text = myfont_xs.render('Tap to start', True, pygame.Color('SALMON'))
111         canvas.blit(hint_text, (325, 210))
112     elif state == 'PLAYING':
113         ball_animation()
114         check_game_over()
115         canvas.fill(pygame.Color('WHITE'))
116
117     # 更新畫面 -----
118     score_text = myfont.render(str(score), True, pygame.Color('SALMON'))
119     canvas.blit(score_text, (10, 10))
120     canvas.blit(picture, (x-r, y-r))
121     screen.blit(canvas, (0, 0))
122     pygame.display.update()
123     # -----
124
125 pygame.quit()

```





**FREE TIME**