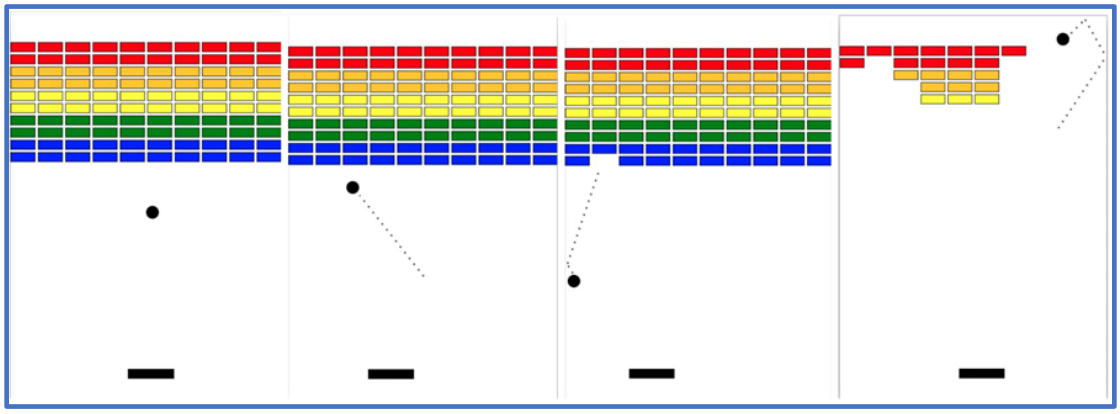
stanCode

標準程式教育機構

Assignment 6

This assignment is based on the Assignment 5 of CS106AP at Stanford University



點此下載作業檔案

==========

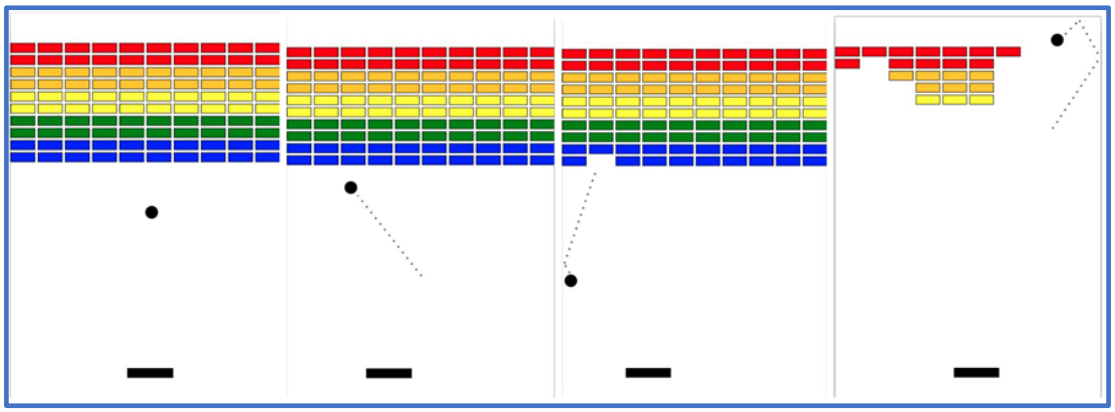
歡迎各位同學來到作業六！這份作業將使用電腦科學中最困難的概念之一
(Class & Object)，來寫出一個 Python 版的「打磚塊遊戲」。

完成這份作業之後，你可以非常驕傲地跟朋友、家人或面試官展示作品，
相信不管誰看到都會非常驚艷的！

如果過程卡關歡迎各位向助教詢問！也非常鼓勵同學們互相討論作業之概念，
但請勿直接把 code 分享給同學看，這很可能會剝奪他獨立思考的機會，
並讓他的程式碼與你的極度相似，使防抄襲軟體認定有抄襲嫌疑。

==========

The Breakout Game - 遊戲介紹



遊戲視窗內，共有三種元件：bricks（磚塊）、ball（球）和 paddle（板子）。

玩家要以滑鼠來操控板子（paddle）的位置，並把下落的球反彈至上方磚塊區，來消滅所有磚塊。除了遊戲視窗底部的邊界以外，球碰到任何物件及邊界都會反彈。歡迎參考上方圖示，來了解球的運動軌跡（黑色虛線是示意線，遊戲畫面裡並不需要出現呦！）。

讓遊戲終止的條件有兩個：

1. 玩家消滅視窗中的所有磚塊，成功過關！

2. 當球超出視窗底部的邊界三次，Game Over！

The Starter Files - 作業檔案介紹

本次作業的資料夾中包含了兩份檔案：

1. **breakout.py**：包含主要程式 main() ，來讓遊戲動畫順利進行。

2. **breakoutgraphics.py**：定義一個 class 叫做 BreakoutGraphics，處理所有背景的圖像、元件。兩份檔案我們都寫好了一些起始程式碼，裡面包含：

   - 所有需要 import 的 classes
   - 常數。包含遊戲各個元件的位置和大小。請您在撰寫程式時使用我們為遊戲定義的常數。

```
BRICK_SPACING = 5      # Space between bricks (in pixels). This space is used for horizontal and vertical spacing
BRICK_WIDTH = 40       # Height of a brick (in pixels)
BRICK_HEIGHT = 15      # Height of a brick (in pixels)
BRICK_ROWS = 10        # Number of rows of bricks
BRICK_COLS = 10        # Number of columns of bricks
BRICK_OFFSET = 50      # Vertical offset of the topmost brick from the window top (in pixels)
BALL_RADIUS = 10       # Radius of the ball (in pixels)
PADDLE_WIDTH = 75      # Width of the paddle (in pixels)
PADDLE_HEIGHT = 15     # Height of the paddle (in pixels)
PADDLE_OFFSET = 30     # Vertical offset of the paddle from the window bottom (in pixels)
INITIAL_Y_SPEED = 7    # Initial vertical speed for the ball
MAX_X_SPEED = 5        # Maximum initial horizontal speed for the ball
```

- BRICK_SPACING 為「磚塊與磚塊之間的距離」
- BRICK_WIDTH 為「一個磚塊的寬」
- BRICK_HEIGHT 為「一個磚塊的高」
- BRICK_ROWS 為「磚塊的總列數」
- BRICK_COLS 為「磚塊的總行數」
- BRICK_OFFSET 為「第一列磚塊頂部與視窗頂部邊界的距離」
- BALL_RADIUS 為「球的半徑」
- PADDLE_WIDTH 為「板子的寬」
- PADDLE_HEIGHT 為「板子的高」
- PADDLE_OFFSET 為「板子底部與視窗底部邊界的距離」
- INITIAL_Y_SPEED 為「球在 y 方向移動的初始速度」
- MAX_X_SPEED 為「球在 x 方向移動的最大速度」

Milestone 1 - BreakoutGraphics Constructor (breakoutgraphics.py)

這是一份相對複雜的作業，因此我們將過程分成幾個里程碑，請同學根據 milestone 的順序完成本次作業。

首先，請打開 **breakoutgraphics.py**，並完成「**BreakoutGraphics**」這個 class 的「constructor」（請不要改變已經寫好的 constructor keyword arguments）。**Constructor** 的目標就是製作出所有遊戲基本元件，並放在遊戲視窗（window）上。動畫的部分則會在之後的 milestone 完成，請先無須理會喔～

如下圖（一）所示，我們已經先將空白的視窗（GWindow）製作出來了。同學在撰寫程式時可以使用「self.window」來呼叫視窗這個物件（object）。你的第一個任務就是在這個視窗上，加上板子（paddle）和球（ball）。

```
class BreakoutGraphics:

    def __init__(self, ball_radius=BALL_RADIUS, paddle_width=PADDLE_WIDTH,
                 paddle_height=PADDLE_HEIGHT, paddle_offset=PADDLE_OFFSET,
                 brick_rows=BRICK_ROWS, brick_cols=BRICK_COLS,
                 brick_width=BRICK_WIDTH, brick_height=BRICK_HEIGHT,
                 brick_offset=BRICK_OFFSET, brick_spacing=BRICK_SPACING,
                 title='Breakout'):

        # Create a graphical window, with some extra space.
        window_width = brick_cols * (brick_width + brick_spacing) - brick_spacing
        window_height = brick_offset + 3 * (brick_rows * (brick_height + brick_spacing) - brick_spacing)
        self.window = GWindow(width=window_width, height=window_height, title=title)

        # Create a paddle.
        # Center a filled ball in the graphical window.
        # Default initial velocity for the ball.
        # Initialize our mouse listeners.
        # Draw bricks.
```

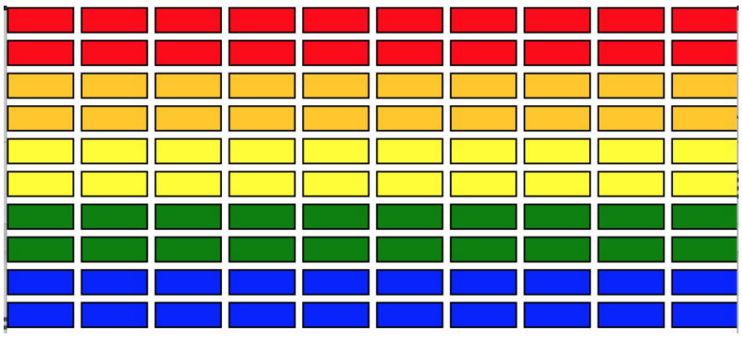
圖（一）breakoutgraphics.py 檔案內容

撰寫的步驟歡迎參考以下的順序：

- 製造板子（paddle），並讓它顯示在視窗中正確位置（板子的中心對齊視窗垂直的中心線；與視窗底部邊界的距離為 PADDLE_OFFSET）
- 製造球（ball），並將它加在視窗的正中間
- 請先忽略單行註解提到的「**Default initial velocity for the ball.**」球的起始速度數值，我們會在下一個 milestone 詳細與大家說明
- 撰寫兩個 **mouse listener (onmouseclicked** 以及 **onmousemoved)**，但先不需要在括弧放入 function 名稱

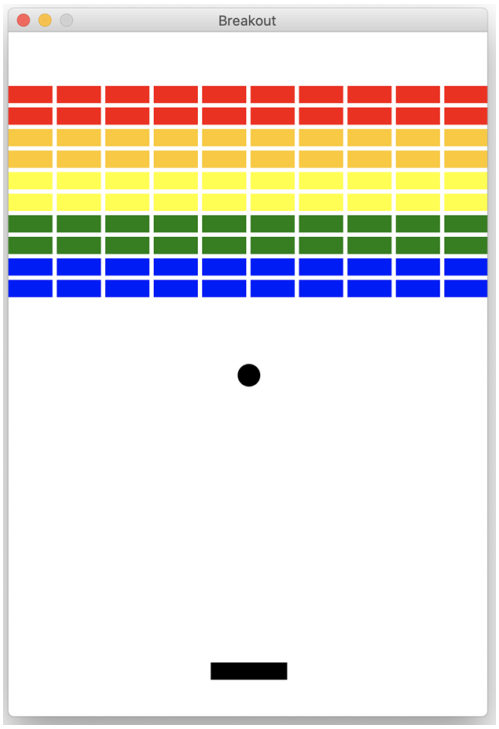
完成板子跟球之後，還需要畫上所有的磚塊（bricks）。如下圖（二）所示，每一列的最左邊磚塊的 x 座標都是 0；最右邊磚塊的右側會緊貼右側視窗邊界。請同學使用我們定義的常數（BRICK_SPACING、BRICK_WIDTH、BRICK_HEIGHT、BRICK_ROWS、BRICK_COLS 及 BRICK_OFFSET，來完成與圖（二）一模一樣的磚塊配置。

請注意：若調整常數 BRICK_ROWS, BRICK_COLS 的數值，磚塊數目應該也會隨之變動喔！



圖（二）所有磚塊完成示意圖

若完成上述所有項目，恭喜，打磚塊的雛型完成了！
若您現在執行 **breakout.py**，彈出的視窗會與下圖（三）相同



圖（三）打磚塊起始畫面。此時畫面都是靜止的

Milestone 2 - Event Driven Programming & Animation

現在要來開始製作動畫了！請同學在 coder 端（**breakoutgraphics.py**）編輯板子（paddle）跟球（ball）最基本的移動動畫。

板子（paddle）動畫：

- 讓板子隨著滑鼠位置「水平」移動
- 板子「正中間」的 x 座標會與滑鼠的 x 座標相同
- 板子底部與視窗底部邊界保持 PADDLE_OFFSET 的距離
- 請勿讓板子超出遊戲視窗的左右兩側；也就是說，就算滑鼠移到視窗外，我們還是要在視窗中看見完整的 paddle（請思考看看靠近邊界的動畫要怎麼設定才會看起來順暢）

球（ball）動畫：

- 製作 instance variables（dx & dy）來定義球的速度，初始值先給 0 即可

  因為球的速度是我們認為打磚塊遊戲中最重要的「靈魂」，所以我們要把「dx & dy 改變為 private variables (__dx&__dy)」，讓在 user 端編輯的使用者無法直接改變它們的數值

- 為了讓遊戲更有趣，我們希望每次遊戲在點擊開始的時候，球可以朝隨機的方向落下，但避免垂直落下（__dx = 0）。請定義並編輯 onmouseclicked(...) 內的 method，並使其可以完成下方要求：

  - 將 __dy 的初始數值設為 INITIAL_Y_SPEED
  - 將 __dx 的初始數值，設定為一個「1 到 MAX_X_SPEED 間的隨機整數」，並利用下方程式碼隨機改變球在 x 軸的方向：

```
if ( random.random( ) > 0.5 ):
    __dx = -__dx
```

- 定義好速度後，接下來便是讓球可以在碰觸到視窗的上、下、左、右邊界時反彈！

  請同學切換至 **breakout.py**（user 端），讓球按照在 coder 端定義好的球速移動，並在碰到視窗邊界時反彈（可以忽略磚塊與板子，讓球穿過他們即可）。
  然而，這邊您可能會發現，我們無法在 user 端直接得到球速值（代表 private variable 發揮作用了!）。解決的辦法很簡單，請在 **breakoutgraphics.py** 定義兩個 **getter functions** 來將球速的值送至 user 端使用。

- 要讓動畫看起來很連貫，最重要的莫過於加上 **pause(...)** 了。請在 def main() 中操控動畫的 while loop 裡，加上 「pause(FRAME_RATE)」來讓動畫可以正確運作（請確保每一次迴圈都能執行一次 pause(...) ，否則會因電腦運算的速度遠大於螢幕更新的頻率而當機！）

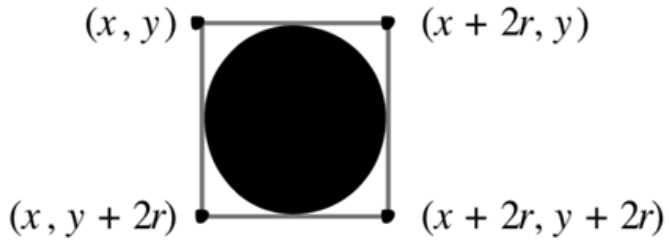
- 執行程式後，球會停留在初始位置，等到使用者點按滑鼠後，才開始往下掉（開始遊戲）。並且，當球在移動的時候，球不會受到滑鼠「重複點按」的影響。這是 milestone 2 最困難的部分，留給您思考！

  小提示：onmouseclicked(...)括弧裡面的程式設計必須偵測「球是否已經處於運動狀態了」

Milestone 3 - Check for Collisions

這份作業最有趣、也最具挑戰性的部分莫過於設定反彈條件。

球在碰到不同遊戲元件時，會有不同的相應動作。因此，我們需要一個工具來判斷「球碰到了什麼物件」？在碰撞時，我們模擬球是一個正方形，並在下方所示，球的四個頂點偵測碰撞：



我們可以使用「**window.get_object_at(x座標, y座標)**」，來偵測每次球在 .move() 之後，是否有碰觸到物件，並決定是否要把它移除和改變球的移動方向。也就是說，球的四個頂點要輪流做到以下五點：

1. 使用 .get_object_at(x座標, y座標) 探測該點是否有物件
2. 如果得到的物件「**非None**」，就代表有碰撞發生，需要進一步判斷碰撞的物件是板子（paddle）還是磚塊（brick），並作出相應的動作：
   - ❖ 板子 → 球反彈
   - ❖ 磚塊 → 球反彈＆移除磚塊（**.remove(...)**）
3. 如果其中一個頂點有發生碰撞，則尚未檢查的剩餘頂點不需檢查（每次偵測只會有一個頂點產生動作）
4. 如果該頂點探測結果是 **None**，請往下檢查另一個頂點
5. 如果四個頂點都得到 **None**，那麼我們就可以確定沒有碰撞發生

Milestone 4 - Finishing up

當您寫到這裡，代表作業大致上已經完成（耶!），下面幾點讓我們做個收尾：

- 當球超出視窗底部邊界時，球會回到起始座標，等待玩家下一次點擊開始（可以從 milestone 2 的 code 做一些調整）
- 遊戲終止條件為「消滅所有磚塊」或是「球掉出視窗下方 NUM_LIVES 次」
- 最常見的 bug 是「當球從板子側邊撞擊時，球卡在板子裡上下震盪」。您會不會發生這種問題呢？
  Hint：建議從每個迴圈按照順序發生的事情，一步步思考看看：）

完成上述基本要求之後，同學可以另外開一個檔案加入有趣的 extentions 功能！（請勿直接加在已完成基本功能的作業檔案上，讓我們可以測試您的作業是否符合基本的要求。）

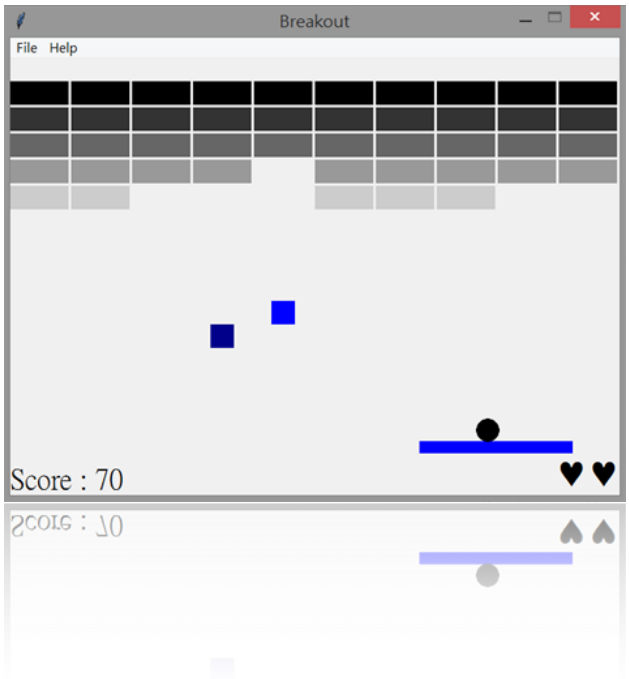
以下提供一些 extentions 的想法給各位參考：

• 加入計分板 GLabel！然而要注意的是，當球碰到計分板是否會反彈？

• 球的速度隨著分數變高而變快，破關的難度瞬間提升！

• 每一個磚塊的分數可以不一樣（例如紅色最高分）

• 發揮你的想像力，期待看到有趣的 extension！



評分標準

Functionality - 程式是否有通過我們的基本要求？程式必須沒有 bug 、能順利完成指定的任務、並確保程式並沒有卡在任何的無限迴圈（infinite loop）之中。

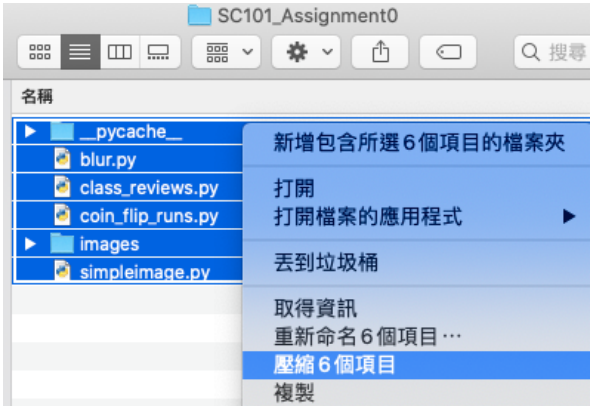
Style - 如同在課堂上所說，好的程式要有好的使用說明 (comment)，也要讓人一目瞭然，這樣全世界的人才能使用各位的 code 去建造更多更巨大更有趣的程式，因此請大家寫出精簡扼要的main( )程式概要、function comments和單行註解。

作業繳交

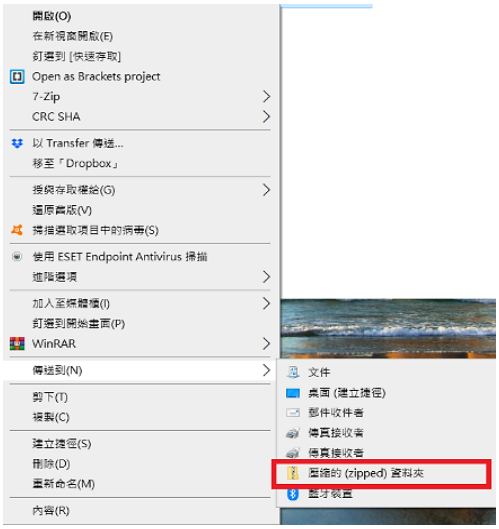
恭喜您完成 Assignment6！請同學於作業繳交期限前，依照下圖將您完成的作業的下載連結上傳至社團提供的作業繳交表單。

1. 以滑鼠「全選」作業資料夾內的所有檔案，並壓縮檔案。請見下圖說明。

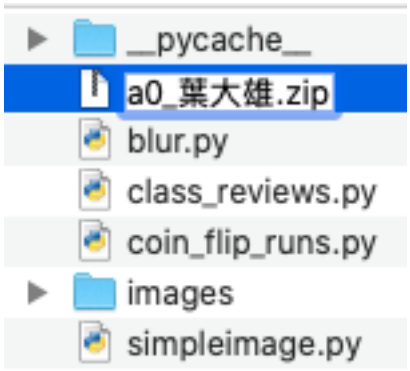
macOS：按右鍵選擇「壓縮n個項目」



Windows：按右鍵選擇「傳送到」→「壓縮的(zipped)資料夾」



2. 將壓縮檔(.zip)重新命名為「a(n)_中文姓名」。如：
assignment 0命名為a0_中文姓名;
assignment 1命名為a1_中文姓名; …



3. 將命名好的壓縮檔(.zip)上傳至new E3 作業繳交區