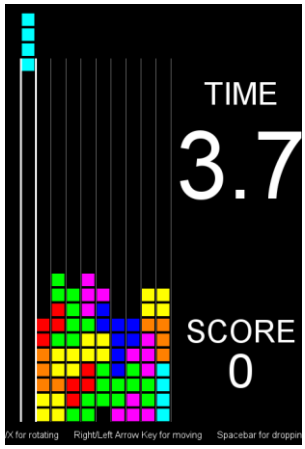
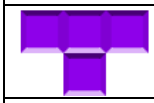
Assignment 11: Tetris

เกมเตตริส (Tetris) เป็นเกมแนวแก้ปริศนาที่โด่งดังมาก ในเกม Tetris จะมีบล็อครูปทรงต่าง ๆ ที่ประกอบด้วยก้อนสี่เหลี่ยมจัตุรัสเล็ก ๆ เรียกว่า Tetrominoes รูปทรงเหล่านี้หล่นลงมาในกระดานทีละชิ้น และเมื่อสามารถเติมเต็มทุกช่องได้ในหนึ่งแถว แถวนั้นจะหายไป โดยการหายไปนี้สามารถหายไปพร้อมกันมากที่สุดได้ 4 แถวเรียกว่า Tetris

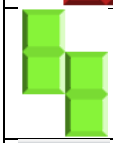


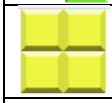
ในที่นี้เราจะมาลองเขียนเกม Tetris โดยใช้ library pygame กัน

เราแทนรูปทรงแต่ละรูปด้วย nested list ดังตัวอย่างข้างล่างนี้
(ค่าในลิสต์แทนสีของก้อน, 0 แทนไม่มีก้อน)

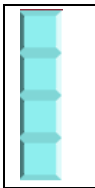
	`[ [1, 1, 1],` `  [1, 0, 0] ]`
	`[ [2, 2, 2],` `  [0, 0, 2] ]`
	`[ [3, 3, 3],` `  [0, 3, 0] ]`
	`[ [4, 4, 4, 4] ]`

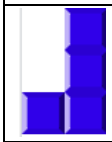
	`[ [5, 5, 0],` `  [0, 5, 5] ]`

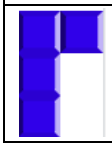
	`[ [6, 0],`
	`  [6, 6],`
	`  [0, 6] ]`

	`[ [7, 7],` `  [7, 7] ]`

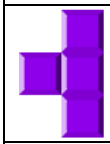
ถ้ารูปทรงถูกหมุน nested list ที่แทนรูปทรงก็เปลี่ยนไปด้วย เช่น

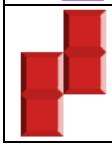
	`[ [4],`
	`  [4],`
	`  [4],`
	`  [4] ]`

	`[ [0, 2],`
	`  [0, 2],`
	`  [2, 2] ]`

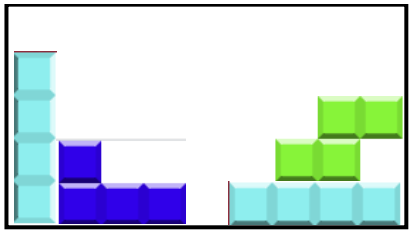
	`[ [2, 2],`
	`  [2, 0],`
	`  [2, 0] ]`

	`[ [0, 6, 6],` `  [6, 6, 0] ]`

	`[ [0, 3],`
	`  [3, 3],`
	`  [0, 3] ]`

	`[ [0, 5],`
	`  [5, 5],`
	`  [5, 0] ]`

และกระดานทั้งกระดานก็แทนด้วย nested list ในลักษณะเดียวกัน เช่น



```
[ [0, 0, 0, 0, 0, 0, 0, 0, 0],
  [4, 0, 0, 0, 0, 0, 0, 0, 0],
  [4, 0, 0, 0, 0, 0, 0, 6, 6],
  [4, 2, 0, 0, 0, 0, 6, 6, 0],
  [4, 2, 2, 2, 0, 4, 4, 4, 4] ]
```

งานของคุณ

จากโปรแกรมต้นฉบับที่มีให้ download จงเขียนชุดคำสั่งใน 4 ฟังก์ชันตามข้อกำหนดดังนี้

- **`rotateR(shape)`**
  - รับ `shape` เป็น nested list (`shape` ก็คือรูปทรงต่าง ๆ ในเกม)
  - คืน nested list ที่เกิดจากการหมุนรูป `shape` ไปทางขวา `90` องศา

Input ของ function	Output ของ function
`shape = [[1,1,1], [0,1,0]]`	`[[0,1], [1,1], [0,1]]`
`shape = [[1,2,3,4]]`	`[[1], [2], [3], [4]]`

- **`rotateL(shape)`**
  - รับ `shape` เป็น nested list (`shape` ก็คือรูปทรางต่าง ๆ ในเกม)
  - คืน nested list ที่เกิดจากการหมุนรูป `shape` ไปทางซ้าย `90` องศา

Input ของ function	Output ของ function
`shape = [[1,1,1], [0,1,0]]`	`[[1,0], [1,1], [1,0]]`
`shape = [[1,2,3,4]]`	`[[4], [3], [2], [1]]`

- **`animate_drop(shape, board, c)`**
  - รับ `shape` และ `board` เป็น nested list, `c` เป็นจำนวนเต็ม
  - คืนลิสต์ของบอร์ด โดยบอร์ดแรกสุดเป็นบอร์ดเมื่อ `shape` ถูกวางไว้ที่บนสุดของ `board` และช่องซ้ายสุดของ `shape` อยู่ที่ช่องที่ `c` ของบอร์ด ส่วนบอร์ดถัด ๆ ไปจะเป็นบอร์ดเมื่อ `shape` ขยับลงมาทีละช่องจนไม่สามารถลงต่อได้ *แต่ถ้าไม่สามารถวาง `shape` ใน `board` ตอนเริ่มต้นได้ (เพราะมีอย่างอื่นขวางอยู่) ก็ต้องคืนลิสต์ว่าง*

Input ของ function	Output ของ function

`shape = [[2,2,2],[0,0,2]]`
 	
`board = [[4,0,0,0],`
`[1,0,0,0],`
`[1,0,0,0],`
`[1,1,0,0]]`
 	
`c = 1`
	`[ [[4,2,2,2],`
	`[1,0,0,2],`
	`[1,0,0,0],`
	`[1,1,0,0]],`
	
	`[[4,0,0,0],`
	`[1,2,2,2],`
	`[1,0,0,2],`
	`[1,1,0,0]],`
	
	`[[4,0,0,0],`
	`[1,0,0,0],`
	`[1,2,2,2],`
	`[1,1,0,2]]`
	`]`

`shape` และ `board` เหมือนตัวอย่างบน แต่ `c = 0`	ลิสต์ว่าง `[]`

- **`animate_clear(board)`**
  - รับ `board` เป็น nested list
  - คืนลิสต์ของบอร์ด โดยบอร์ดแรกสุดเป็นบอร์ดที่เปลี่ยนทุกแถวที่ทุกค่าในแถวไม่เป็นศูนย์ ให้เป็นเป็นศูนย์ก่อน จากนั้น บอร์ดถัด ๆ ไปจะเป็นบอร์ดเมื่อขยับทุกแถวลงมาทีละช่องจนไม่สามารถขยับต่อได้ (แต่ถ้าไม่มีแถวที่ต้องเปลี่ยนเป็น `0` หมด ก็คืน ลิสต์ว่าง)

Input ของ function	Output ของ function

`board = [[3,3,0,0],`
`[2,2,2,6],`
`[0,5,5,4],`
`[7,7,7,6],`
`[0,0,0,0],`
`[4,0,1,1]]`
	`[ [[3,3,0,0],`
	`[0,0,0,0],`
	`[0,5,5,4],`
	`[0,0,0,0],`
	`[0,0,0,0],`
	`[4,0,1,1]],`
	
	`[[0,0,0,0],`
	`[3,3,0,0],`
	`[0,0,0,0],`
	`[0,5,5,4],`
	`[0,0,0,0],`
	`[4,0,1,1]],`
	
	`[[0,0,0,0],`
	`[0,0,0,0],`
	`[3,3,0,0],`
	`[0,0,0,0],`
	`[0,5,5,4],`
	`[4,0,1,1]],`
	
	`[[0,0,0,0],`
	`[0,0,0,0],`
	`[0,0,0,0],`
	`[3,3,0,0],`
	`[0,5,5,4],`
	`[4,0,1,1]]`
	`]`

`board = [[3,0,0,0,2],`
`[3,0,4,0,2],`
`[4,4,4,0,2]]`
	`[]`

`board = [[0,0,0,0,0],`
`[2,0,0,0,0],`
`[2,3,3,4,4]]`
	`[ [[0,0,0,0,0],`
	`[2,0,0,0,0],`
	`[0,0,0,0,0]],`
	
	`[[0,0,0,0,0],`
	`[0,0,0,0,0],`
	`[2,0,0,0,0]]`
	`]`

อย่าลืม

- ก่อนจะทำโจทย์ได้ ต้องติดตั้ง `package` ที่ชื่อ `pygame`
  (วิธีติดตั้ง `pygame` ใน `Thonny` เหมือนการบ้านที่ต้องติดตั้ง `pandas` แต่ในการบ้านนี้ชื่อ `pygame`)
- ห้ามเขียนคำสั่งในฟังก์ชันที่ไปใช้ตัวแปรที่อยู่นอกฟังก์ชัน (หรือที่เรียกว่าตัวแปร `global`)
- อนุญาตให้เพิ่มคำสั่งในบริเวณพื้นที่สีเขียวที่กำหนดเท่านั้น (สามารถเขียนฟังก์ชันเพิ่มเติมในบริเวณนี้ได้)
- ตั้งชื่อแฟ้ม ให้ถูกต้องตามที่เขียนใน `CourseVille`
- เพิ่มเติมข้อมูลใน `comment` ต้นโปรแกรมให้ตรงตามความจริง
- ก่อนส่งโปรแกรม ควรสั่งทำงานโปรแกรมที่ส่งอีกครั้ง ว่าทำงานได้ตามที่ต้องการ

ข้อแนะนำ

- ถ้าต้องการ `copy` ลิสต์ซ้อนลิสต์ `x` ให้ใช้คำสั่ง `copy.deepcopy(x)` (ต้อง `import copy` ก่อน)

```
# Prog-11: Tetris
# Fill in your ID & Name
# ...
# Declare that you do this by yourself

import pygame
import copy
import random

def make_shape():
    # เขียนให้แล้ว

def top(board):
    # เขียนให้แล้ว

def scoring(board):
    # เขียนให้แล้ว

def show_text(x,y,size,text,screen):
    # เขียนให้แล้ว

def pgame():
    # เขียนให้แล้ว
```

ไม่เพิ่ม ลบ หรือ เปลี่ยนแปลง บริเวณคำสั่ง สีแดง เด็ดขาด

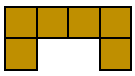
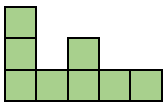
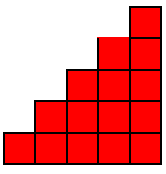
```
def rotateR(shape):

    return ???

def rotateL(shape):

    return ???

def animate_drop(shape, board, c):

    return ???

def animate_clear(board):

    return ???
#-----------------------------------

pgame()
```

ไม่ใช้ตัวแปรใด ๆ ที่อยู่นอกฟังก์ชัน

download code ฉบับเต็ม

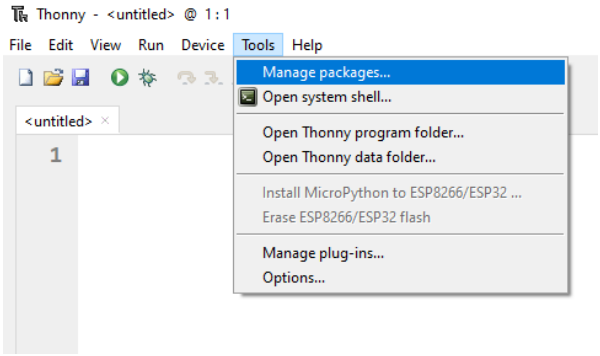
คำเตือน

- ข้อมูลของ `shape` ที่ใช้ทดสอบ ไม่จำเป็นต้องเหมือนกับ `shape` ในเกมที่ใช้กันทั่วไป เช่น

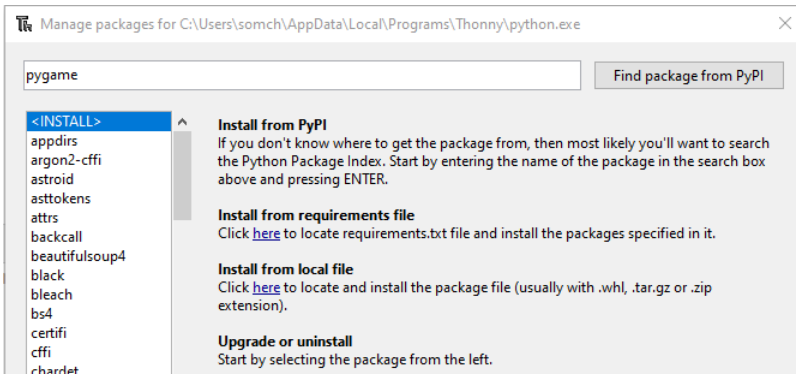




วิธีติดตั้ง pygame

- ใน `Thonny` เลือกเมนู `Tools -> Manage packages`

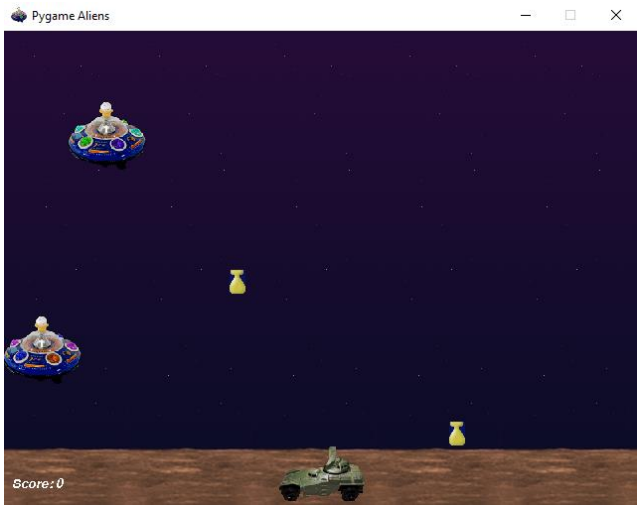


- ใส่คำว่า `pygame` และกดปุ่ม `Find package from PyPI`



- แล้วก็กดปุ่ม `install` รอจนเสร็จ แล้วก็กดปุ่ม `close`
- ที่ `Thonny` เลือกเมนู `Tools -> Open system shell...` จะแสดงวินโดว์ดำ ๆ
    - ถ้าใช้ `Windows` ก็ใส่คำสั่ง `python -m pygame.examples.aliens` แล้วกด `enter`
    - ถ้าใช้ `Mac OS` ก็ใส่คำสั่ง `python3 -m pygame.examples.aliens` แล้วกด `enter`

ถ้ามีเกมโผล่ขึ้นมาให้เล่น ก็แสดงว่าได้ติดตั้งสำเร็จ



เฉลย

```
def rotateR(shape):
    shape_row, shape_column = len(shape), len(shape[0])
    new_shape = []
    for i in range(shape_column):
        new_shape.append([shape[j][i] for j in range(shape_row-1,-1,-1)])
    return new_shape

def rotateL(shape):
    shape_row, shape_column = len(shape), len(shape[0])
    new_shape = []
    for i in range(shape_column-1,-1,-1):
        new_shape.append([shape[j][i] for j in range(shape_row)])
    return new_shape

def animate_drop(shape,board,c):
    board_row, board_column = len(board), len(board[0])
    shape_row, shape_column = len(shape), len(shape[0])
    frame = []
    for t in range(board_row-shape_row+1):
        new_board = copy.deepcopy(board)
        for i in range(shape_row):
            for j in range(shape_column):
                if board[t+i][c+j] !=0 and shape[i][j] !=0:
                    return frame
                new_board[t+i][c+j] += shape[i][j]
        frame.append(new_board)
    return frame
```

```
def animate_clear(board):
    board_row, board_column = len(board), len(board[0])
    frame =[]
    check = 0
    new_board = copy.deepcopy(board)
    for i in range(board_row):
        if 0 not in new_board[i]:
            new_board[i] = [0]*len(board[0])
            check = 1
    if check == 0:
        return frame
    frame.append(new_board)
    while True:
        b = copy.deepcopy(new_board)
        for i in range(board_row-1,0,-1):
            if b[i] == [0]*board_column:
                b[i] = b[i-1][:]
                b[i-1] = [0]*board_column
        if new_board == b:
            return frame
        frame.append(b)
        new_board = b
    return frame
```