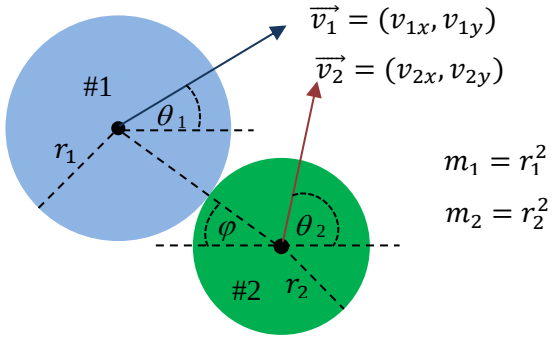
Assignment 2: Elastic Collision

รูปข้างล่างนี้แสดงลูกบอล #1 (สีฟ้า) กับ #2 (สีเขียว) ชนกันในระนาบสองมิติ โดยลูกบอล #1 และ #2 เคลื่อนในทิศและความเร็วตามเวกเตอร์ $\overrightarrow{v_1}$ และ $\overrightarrow{v_2}$ ตามลำดับ คำถามที่น่าสนใจคือ หลังจากชนแล้ว ลูกบอลทั้งสองจะเคลื่อนในทิศใดด้วยความเร็วเท่าใด



ขอยกผลมาเลยข้างล่างนี้ (อ่านคำอธิบายฉบับเต็มได้ที่ https://en.wikipedia.org/wiki/Elastic_collision)

Two-dimensional collision with two moving objects [edit]

The final x and y velocities components of the first ball can be calculated as:[2]

$$v'_{1x} = \frac{v_1 \cos(\theta_1 - \varphi)(m_1 - m_2) + 2m_2 v_2 \cos(\theta_2 - \varphi)}{m_1 + m_2} \cos(\varphi) + v_1 \sin(\theta_1 - \varphi) \cos(\varphi + \frac{\pi}{2})$$

$$v'_{1y} = \frac{v_1 \cos(\theta_1 - \varphi)(m_1 - m_2) + 2m_2 v_2 \cos(\theta_2 - \varphi)}{m_1 + m_2} \sin(\varphi) + v_1 \sin(\theta_1 - \varphi) \sin(\varphi + \frac{\pi}{2})$$

where v_1 and v_2 are the scalar sizes of the two original speeds of the objects, m_1 and m_2 are their masses, θ_1 and θ_2 are their movement angles, that is, $v_{1x} = v_1 \cos\theta_1$, $v_{1y} = v_1 \sin\theta_1$ (meaning moving directly down to the right is either a -45° angle, or a 315°angle), and lowercase phi (φ) is the contact angle. (To get the x and y velocities of the second ball, one needs to swap all the '1' subscripts with '2' subscripts.)

$m_1 = r_1^2$

$m_2 = r_2^2$

This equation is derived from the fact that the interaction between the two bodies is easily calculated along the contact angle, meaning the velocities of the objects can be calculated in one dimension by rotating the x and y axis to be parallel with the contact angle of the objects, and then rotated back to the original orientation to get the true x and y components of the velocities[3][4][5][6][7][8]

สิ่งที่ต้องทำ

โปรแกรมในหน้าถัดไป แสดงลูกบอลหลายลูกเคลื่อนที่ในระนาบ 2 มิติ โปรแกรมนี้เขียนไว้ให้ลูกบอลเปลี่ยนทิศเมื่อกระทบผนัง แต่ยังไม่ได้เขียนให้เปลี่ยนทิศและความเร็วเมื่อลูกบอลกระทบกันเอง (ลอง download code แล้วสั่งทำงานดู) สิ่งที่ต้องทำคือ

- เขียนคำสั่งคำนวณความเร็วใหม่ในแนวแกน x และ y ของลูกบอลทั้งสอง ในบริเวณ**สีเขียว**ใน code แล้วสั่งทำงาน
- ตัวแปรที่กำหนดลักษณะของลูกบอลที่ใช้ได้ในบริเวณ**สีเขียว** มีดังนี้

`x1, y1`	คือพิกัดของจุดศูนย์กลางลูกบอล #1	`x2, y2`	คือพิกัดของจุดศูนย์กลางลูกบอล #2
`r1`	คือรัศมีของลูกบอล #1	`r2`	คือรัศมีของลูกบอล #2
`v1x, v1y`	คือความเร็วในแนวนอนและแนวตั้งของลูกบอล #1	`v2x, v2y`	คือความเร็วในแนวนอนและแนวตั้งของลูกบอล #2

เมื่อฟังก์ชัน `update_v` เริ่มทำงาน ตัวแปร `v1x, v1y, v2x, v2y` เก็บความเร็วในแนวแกน x และ y ของลูกบอลสองลูกแล้ว คงต้องเขียนคำสั่งคำนวณค่าของ θ_1 , θ_2 , φ , m_1 , m_2 , v_1 และ v_2 ในสูตร แล้วมีคำสั่งอีกอย่างน้อย 4 คำสั่งข้างล่างนี้ เพื่อหาค่าความเร็วใหม่หลังการชนของลูกบอลทั้งสอง (เก็บใส่ตัวแปร `v1_x, v1_y, v2_x, v2_y`)

```
v1_x = _____________________________
v1_y = _____________________________
v2_x = _____________________________
v2_y = _____________________________
```

ก่อนจะทำคำสั่ง **`return (v1_x, v1_y), (v2_x, v2_y)`** ที่เขียนให้แล้วในโปรแกรม (คำสั่งนี้คืนความเร็วใหม่ กลับไปใช้งาน)

โปรแกรมต้นฉบับ

ห้ามเปลี่ยนโค้ดส่วนที่เป็นสีแดงโดยเด็ดขาด เปลี่ยนได้เฉพาะส่วนที่มีพื้นหลังเป็นสีเขียวเท่านั้น

```
# Prog-02: Ellastic Collision
# Your ID & name

import math
import random
from matplotlib import pyplot as plt
from matplotlib import animation, rc

WIDTH, HEIGHT = 300, 300

class Ball(plt.Circle) :
  def __init__(self, r=None, xy=None,
               v=None, color=None):
    if r == None:
      r = random.randint(4, 10)
    if xy == None:
      x = random.randint(r, WIDTH-r)
      y = random.randint(r, HEIGHT-r)
      xy = (x,y)
    if v == None:
      dx = -5 + random.random()*10
      dy = -5 + random.random()*10
      v = (dx,dy)
    self.v = v
    if color == None:
      color = [random.random() for i in range(3)]
    super().__init__(xy, radius=r,
                     facecolor=color)
#-----------------------------------------------
def update_v_if_collide(b1, b2):
  x1, y1 = b1.center
  x2, y2 = b2.center
  r1 = b1.radius
  r2 = b2.radius
  v1x, v1y = b1.v
  v2x, v2y = b2.v
  dx = x1 - x2
  dy = y1 - y2
  t = dx**2 + dy**2 - (r1+r2)**2
  if t <= 0:
    if t < 0:
      x1 -= v1x; y1 -= v1y
      x2 -= v2x; y2 -= v1y
    b1.v, b2.v = update_v(x1, y1, r1, v1x, v1y,
                          x2, y2, r2, v2x, v2y)

def update_v(x1, y1, r1, v1x, v1y,
             x2, y2, r2, v2x, v2y):

  # insert your code here
  v1_x = v1x
  v1_y = v1y
  v2_x = v2x
  v2_y = v2y

  return (v1_x, v1_y), (v2_x, v2_y)
```

ใส่เลขประจำตัว ชื่อ นามสกุล

ดู VDO แนะนำเรื่องฟังก์ชัน

```
def move(ball):
  x, y = ball.center
  vx, vy = ball.v
  r = ball.radius
  x += vx
  y += vy
  if not (r <= x <= WIDTH-r):
    vx *= -1
    x += vx
    if x-r < 0: x = r
    if x+r > WIDTH: x = WIDTH-r
  if not (r <= y <= HEIGHT-r):
    vy *= -1
    y += vy
    if y-r < 0: y = r
    if y+r > HEIGHT: y = HEIGHT-r
  ball.center = (x,y)
  ball.v = (vx,vy)

def total_Ek():
  sum_Ek = sum(0.5*(balls[i].radius**2 *
       balls[i].v[0]**2 + balls[i].v[1]**2))
                for i in range(len(balls)))
  return sum_Ek

def animate(i):
  for i in range(len(balls)):
    move(balls[i])
  #print(total_Ek())
  for i in range(len(balls)):
    for j in range(i+1, len(balls)):
      update_v_if_collide(balls[i], balls[j])
  return balls

#-----------------------------------------------
fig, ax = plt.subplots()
fig.set_size_inches(4,4)
ax.set_xlim((0, WIDTH))
ax.set_ylim((0, HEIGHT))

balls = []
n = 9
for i in range(n):
  b = Ball(xy = ((i+1)*WIDTH/(n+1), HEIGHT-70),
           v=(0,-1))
  balls.append(b)
  ax.add_patch(b)
b = Ball(xy=(WIDTH/2, 40), r=30,
         v=(1,5), color='red')
balls.append(b)
ax.add_patch(b)
anim = animation.FuncAnimation(fig, animate,
                               frames=None,
                               interval=30,
                               blit=True)
plt.show()
```

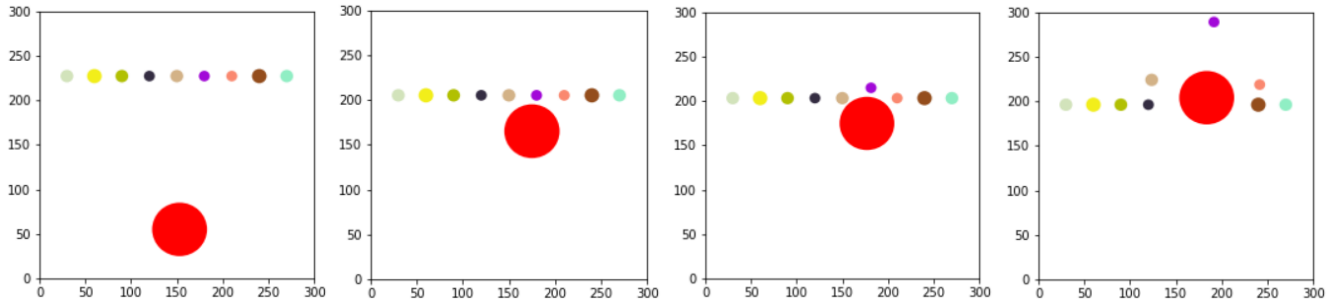
download code นี้ได้

รู้ได้อย่างไรว่าเขียนคำสั่งถูกต้อง

- สั่ง run โปรแกรมแล้วดูด้วยสายตาว่า วงกลมทั้งหลายหลังกระทบกันแล้วเปลี่ยนทิศทางและความเร็วได้อย่างที่คาดไหม
- เนื่องจากพลังงานจลน์รวมในระบบต้องเท่าเดิมตลอดเวลา (อาจต่างกันได้เล็กน้อยจากความแม่นยำในการคำนวณของ float) ถ้าต้องการให้แสดงพลังงานจลน์รวมระหว่างการทำงาน ให้ลบเครื่องหมาย # ตรงนี้ออก (อย่าลืมใส่กลับดังเดิมก่อนส่งการบ้าน)

ผลลัพธ์

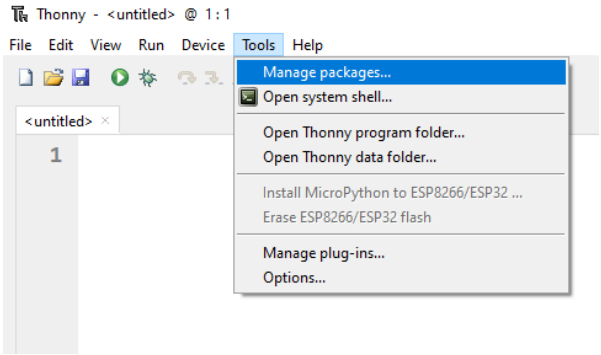
วินโดว์แสดงภาพเคลื่อนไหวของลูกบอลในระนาบสองมิติเคลื่อนไหวไปมาที่มีการเปลี่ยนทิศและความเร็วของการเคลื่อนไหวเมื่อกระทบผนังและลูกบอลกันเอง (ดูตัวอย่างผลก่อนและหลังการเปลี่ยนแปลงได้ที่ https://youtu.be/pY-QhHT5BYU)



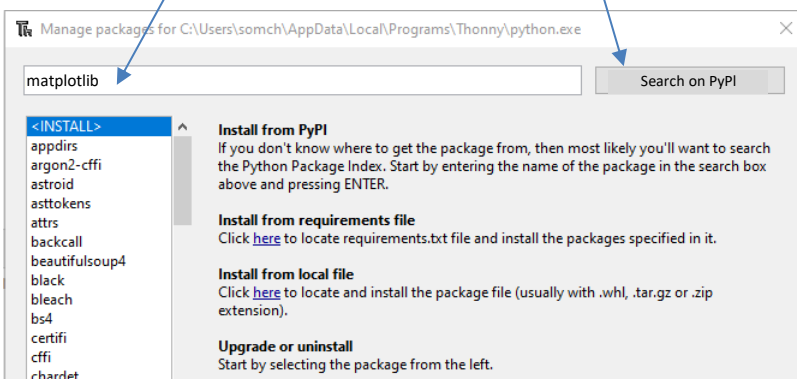
วิธีติดตั้ง matplotlib

โปรแกรมในการบ้านนี้ ต้องการคลังคำสั่งที่ชื่อว่า `matplotlib` ซึ่งไม่มีให้มา ต้องถูกติดตั้งเพิ่มเติม ทำได้ดังนี้

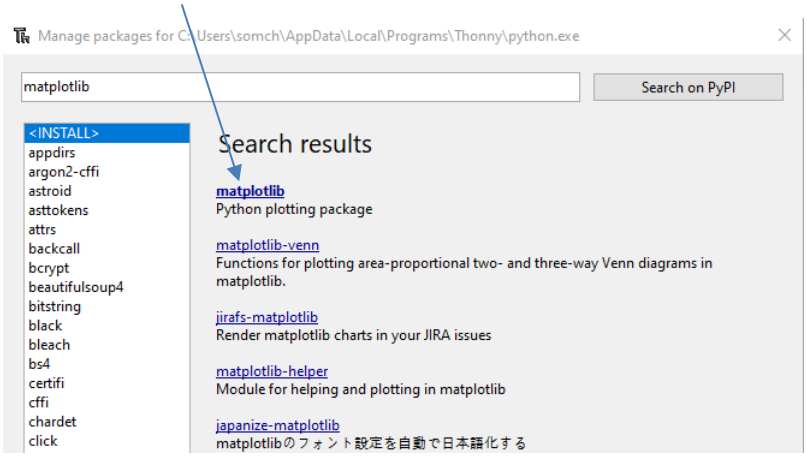
- ใน `Thonny` เลือกเมนู `Tools -> Manage packages`



- ใส่คำว่า **`matplotlib`** และกดปุ่ม **`Search on PyPI`**



- จากนั้นคลิกเลือก



- แล้วก็กดปุ่ม `Install` รอจนเสร็จ แล้วก็กดปุ่ม `Close`
- เปิดโปรแกรม `ball_collision.py` ที่ `download` มา แล้วสั่ง `run` ดู