

# Escalona\_Joaquín\_7

September 2018

## Ejercicio 1

Jueguen con el código de clase “bounce.py”. Averiguen qué hacen cada parte del código. Describan en palabras qué hacen los siguientes elementos del código:

```
##--coding:utf-8

import numpy as np
import matplotlib.pyplot as plt

plt.figure(1)
plt.clf()
#Creacion de ejes
plt.axis([-10,10,-10,10,])

#Define properties of the 'bouncing balls'
#n bolas (10 bolas en este caso)
n      = 10
#crear arreglo con n*2 datos entre 0 y 1 aleatorio y
#reordenar en una matriz de n filas y 2 columnas
#pos = posicion, varía entre -10 y 10
pos    = (20 * np.random.sample(n*2)-10).reshape(n,2)
#crear arreglo con n*2 datos aleatorios distribución gaussiana
#y reordenar en matriz de n filas y 2 columnas
#además vel = velocidad varía entre -1 y 1
vel    = (0.3 * np.random.normal(size=n*2)).reshape(n,2)
#arreglo de n datos
#varia entre 100 y 200
#tamaño del radio de las bolas
sizes  = 100 * np.random.sample(n)+100

#Colors where each row is (Red,Green,Blue,Alpha), Each can go
#from 0 to 1. Alpha is transparency,
#crea 10 bloques de datos de 1 fila y 4 columnas cada uno
colors = np.random.sample([n,4])

#Draw all the circles and return an object 'circles' that allows
#manipulation of the plotted circles.

circles = plt.scatter(pos[:,0],pos[:,1],marker = 'o',s = sizes,c=colors)

#se realizarán 100 iteraciones
for i in range(100):
    pos = pos + vel
```

```

bounce = abs(pos) > 10    #Find balls that are outside walls
vel[bounce] = -vel[bounce] #Bounce if outside the walls
circles.set_offsets(pos)  #Change the positions
plt.draw()
plt.show()
#tiempo de duración
plt.pause(0.05)

```

### (a) `np.random.sample`

Retorna números flotantes mediante "distribución continua uniforme" en el intervalo [0.0,1.0) (Ver más info en este [link](#)). Fue utilizado para seleccionar aleatoriamente distintas características de las bolas como su color, forma, tamaño, etc.

### (b) `np.random.normal`

Entrega muestras aleatorias de una distribución normal gaussiana (Ver [aquí](#)). Se utilizó para definir la velocidad (aleatoria) de las bolas.

### (c) que hace `reshape`?

Cambia la forma de un array a otra que tenga el mismo número de elementos (en filas \* columnas). En este caso se quiso tener un arreglo de datos en forma matricial con n filas y 2 columnas.

### (d) por que le quitamos un valor de "10" a los valores de `np.random.sample` en la definición de `pos`?

Simplemente para que la posición oscile entre -10 y 10 y así no salga de los ejes.

### (e) como de grande es la caja donde existen las pelotas?

Esto queda definido en `plt.axis()` el cual es de 20x20

### (f) cuantas pelotas tiene su código?

Definido por la variable `n`, hay 10 bolas en este caso.

### (g) como puede hacer que se muevan mas lentamente las pelotas?

En la sentencia `vel = (0.3 * np.random.normal(size=n*2)).reshape(n,2)`, debemos cambiar el 0.3 por algún número  $x$  que satisfaga  $0 < x < 0.3$  para reducir la velocidad. (Con  $x = 0$  las bolas no se mueven)

### (h) como podemos modificar el programa para que corra por mas tiempo?

Variando las iteraciones en la sentencia `for` por un numero mayor que 100

### (i) `pos` y `vel` son arreglos de tamaño `n x j`. Que esta haciendo exactamente la linea `pos = pos + vel`?

Esto es para que exista movimiento de las bolas. Al sustituir por la sentencia `pos = pos` vemos que las bolas no se mueven. Y al sustituir por `pos = pos + vel` vemos que las bolas quedan en el origen de coordenadas (0,0) sin moverse.

## Ejercicio 2

Lean la documentación sobre `plt.scatter`. Como se definen los colores? El tipo de símbolo? El tamaño de los símbolos?

### Solución

Al leer la [documentación](#), para definir los colores, el parámetro correcto es **c**. Los posibles valores que puede tomar son:

- A single color format string
- A sequence of color specifications of length n.
- A sequence of n numbers to be mapped to colors using cmap and norm.
- A 2-D array in which the rows are RGB or RGBA

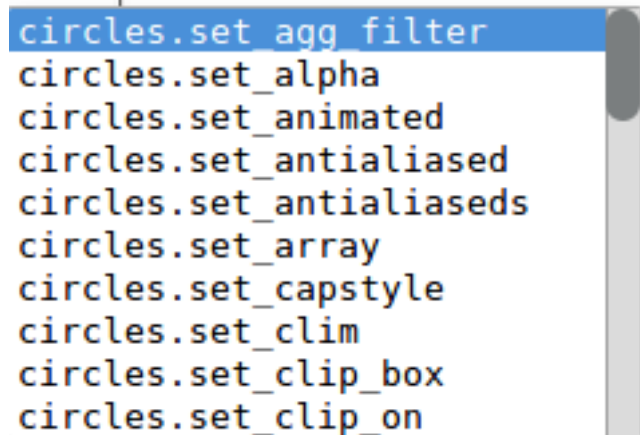
El tipo de símbolo se crea con el parámetro **marker** que por defecto es 'o' (círculos). Para el tamaño de los símbolos el parámetro es **size**

## Ejercicio 3

Examine the circles object with help and ?. Look for “set\_\*” methods (circles.set\_`TAB`) that will let you set something and then get help on those as well to learn how to use them. There are corresponding “get\_\*” methods that let you examine the existing values.

### Solución

Al examinar circles con `help(circles)` se muestra mucha información acerca de métodos que se le puede atribuir a circles, que es un tipo *PathCollection*. Al hacer `TAB` en `circles.set_` se muestran mas funciones para atribuirle a circles, tal como se muestra en la imagen a continuación



```
circles.set_agg_filter
circles.set_alpha
circles.set_animated
circles.set_antialiased
circles.set_antialiaseds
circles.set_array
circles.set_capstyle
circles.set_clim
circles.set_clip_box
circles.set_clip_on
```

## Ejercicio 4

Modifiquen el programa para que en vez de rebotar, las pelotas sigan viajando en la misma dirección, pero sigan dentro del marco del plot.

### Solución

En el ciclo `for` se ha introducido la sentencia `pos[bounce]=-pos[bounce]` para que las líneas sigan la misma trayectoria.

```
#-*-coding:utf-8

import numpy as np
import matplotlib.pyplot as plt

plt.figure(1)
plt.clf()
plt.axis([-10,10,-10,10,])

n      = 10
pos    = (20 * np.random.sample(n*2)-10).reshape(n,2)
vel    = (0.3 * np.random.normal(size=n*2)).reshape(n,2)
sizes  = 100 * np.random.sample(n)+100

colors = np.random.sample([n,4])

circles = plt.scatter(pos[:,0],pos[:,1],marker = 'o',s = sizes,c=colors)

for i in range(1000):
    pos = pos + vel
    bounce = abs(pos) > 10
    pos[bounce] = -pos[bounce]
    circles.set_offsets(pos)
    plt.pause(0.05)
```

## Ejercicio 5

Si tienen N iteraciones en su código, modifíquelo para que en la iteración N - 20 escriba un archivo en formato ASCII con las siguientes columnas:

pos(x), vel(x), pos(y), vel(y)

El archivo debe tener n líneas (donde n = numero de pelotas), y 4 columnas. No se olviden de indicar en el archivo (en la cabecera) lo que significa cada columna.

## Solución

La cantidad de bolas es 1000 para tener muchos datos en el ejercicio 6. La idea mostrada aquí se encontró en [Stack-Overflow](#)

```
#-*-coding:utf-8
```

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

```
plt.figure(1)
plt.clf()
plt.axis([-10,10,-10,10,])
```

```
n      = 1000
pos     = (20 * np.random.sample(n*2)-10).reshape(n,2)
vel     = (0.3 * np.random.normal(size=n*2)).reshape(n,2)
sizes   = 100 * np.random.sample(n)+100
```

```
colors = np.random.sample([n,4])
```

```
circles = plt.scatter(pos[:,0],pos[:,1],marker = 'o',s = sizes,c=colors)
```

```
#N iteraciones (100)
```

```
for i in range(100):
```

```
    pos = pos + vel
```

```
    bounce = abs(pos) > 10
```

```
    vel[bounce] = -vel[bounce]
```

```
    #datos inicial
```

```
    if (i == 0):
```

```
        np.savetxt('datos_0.csv',np.column_stack((pos[:,0],vel[:,0],pos[:,1],vel[:,1])))
```

```
    #datos con N-20
```

```
    elif (i == (100-20)):
```

```
        np.savetxt('datos.csv',np.column_stack((pos[:,0],vel[:,0],pos[:,1],vel[:,1])))
```

```
    circles.set_offsets(pos)
```

```
    plt.pause(0.05)
```

## Ejercicio 6

Escriban un programa que lea el archivo del ejercicio 5, y haga dos histogramas: uno de las distancias relativas al centro (0,0) para todas las pelotas y otra de la energía kinetica especifica de las pelotas. Comparen los histogramas con los valores con los que inicializaron sus “pelotas” al principio del programa (van a tener que modificar el código para escribir mas de un archivo). (deben de tener 4 histogramas en total). Como se comparan las distribuciones iniciales y las extraídas después de un tiempo (N-20)?

### Solución

Pude hacer este código gracias a la fundamental ayuda de **Franco Sepulveda**

```
# -*- coding: utf-8 -*-

import numpy as np
import matplotlib.pyplot as plt

# leer datos iniciales y asignarlos a las variables
pos_x, vel_x, pos_y, vel_y = np.loadtxt('datos_0.csv', delimiter='\t', unpack=True)

# Distancia relativa al (0.0)
dist0 = np.sqrt(pos_x**2 + pos_y**2)
# subplot para tener 4 graficos en uno
# idea de Franco Sepulveda
with plt.xkcd():
    plt.subplot(2,2,1)
    plt.title('Histograma en 0')
    plt.xlabel('Distribucion relativa')
    plt.ylabel('Numero de bolas')
    plt.hist(dist0)

# Energía cinética
Ek_0 = 0.5*(vel_x**2 + vel_y**2)
with plt.xkcd():
    plt.subplot(2,2,2)
    plt.title('Energia cinetica en 0')
    plt.xlabel('Energia cinetica')
    plt.ylabel('Numero de bolas')
    plt.hist(Ek_0)

# Leer datos en N-20
pos_x1, vel_x1, pos_y1, vel_y1 = np.loadtxt('datos.csv', delimiter='\t', unpack=True)
# Distancia relativa al (0.0) en N-20 iteraciones
dist1 = np.sqrt(pos_x1**2 + pos_y1**2)
with plt.xkcd():
    plt.subplot(2,2,3)
    plt.xlabel('Distribucion relativa en N-20')
    plt.ylabel('Numero de bolas')
    plt.hist(dist1)

# Energía cinética
Ek_1 = 0.5*(vel_x1**2 + vel_y1**2)
with plt.xkcd():
    plt.subplot(2,2,4)
    plt.xlabel('Energia cinetica en N-20')
    plt.ylabel('Numero de bolas')
```

```
plt.hist(Ek_1)
#tight_layout para mejor estética
plt.tight_layout()
plt.savefig('hist.pdf')
```

