Ingeniería en Sistemas Computacionales
PROGRAMACION ORIENTADA A OBJETOS
Oscar David Galvan Alvarez
TUE0096

EXAMEN MOVIMIENTO BROWNIANO

Explicación del inciso faltante del examen de movimiento browniano, así como también el resultado final que se obtuvo del mismo.

CODIGO:

undefined - EXAMEN MOV BROWNIANO_DAVID ALVAREZ_INCISO D).py

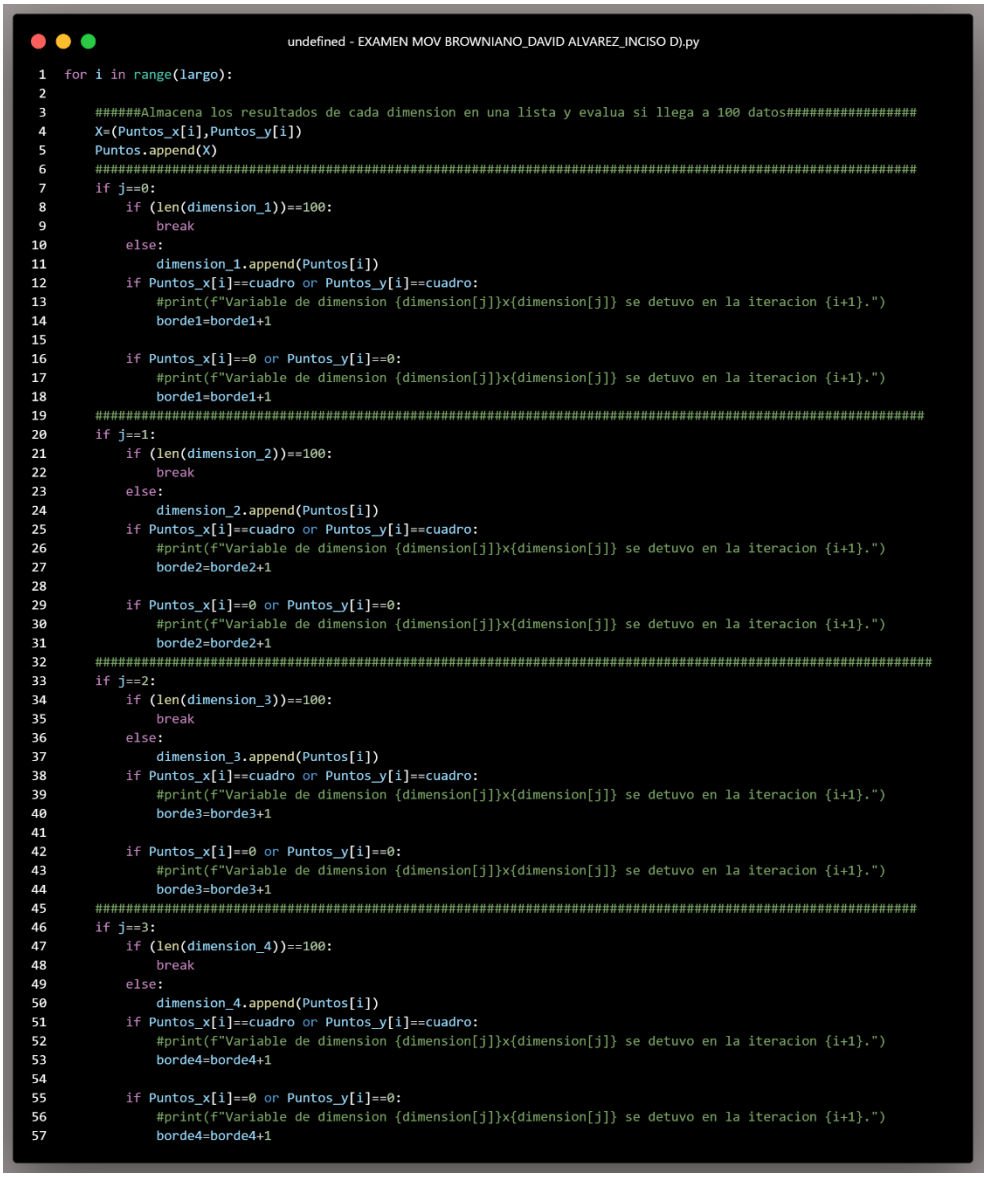
```python
import matplotlib.pyplot as plt
import numpy as np
import random

#####VARIABLES#######
dimension=[50,100,200,500]
######LISTA DE COLORES#####
#Color=["green","red","purple","orange","yellow"]
#################Listas de dimensiones###################################
dimension_1=[]
dimension_2=[]
dimension_3=[]
dimension_4=[]
borde1=0
borde2=0
borde3=0
borde4=0
```

En esta primer parte primero se importaron las librerías necesarias, así como también se inicializaron las variables con listas vacías y las variables numéricas para contadores, en el caso de la lista de colores solo se puso como comentario ya que no era necesario utilizarla en el código.

Ahora en esta siguiente parte se inicializa un ciclo for el cual abarca todos los ciclos para que así vaya avanzando en las dimensiones a utilizar almacenadas en una lista previamente, así como también se crean las listas que almacenaran los puntos y el ciclo que delimita el rango de veces que crea puntos y los almacena en dichas listas.

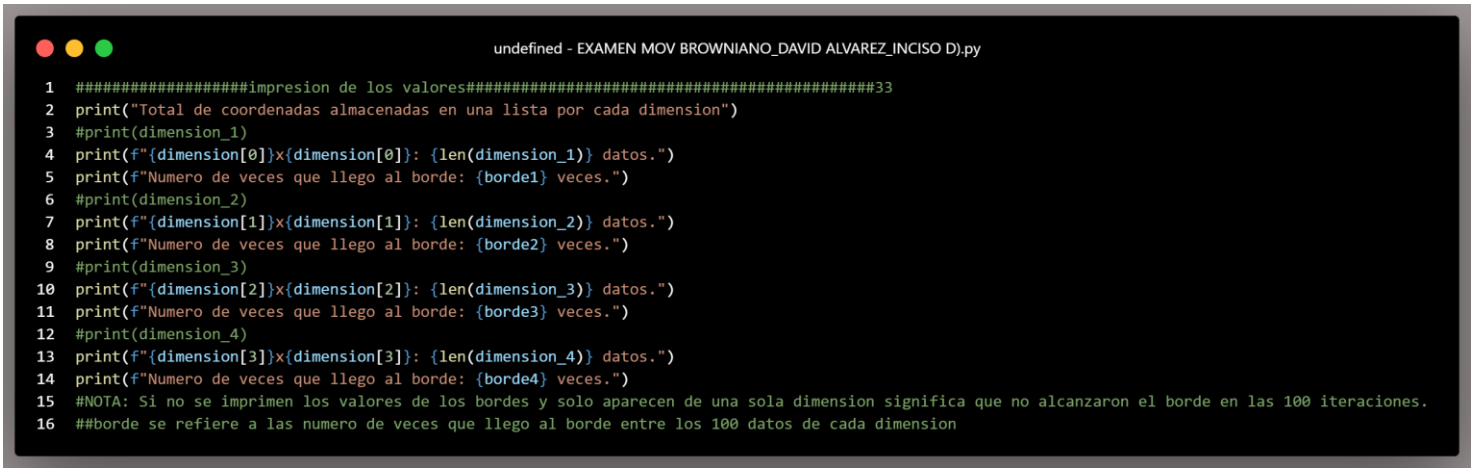
undefined - EXAMEN MOV BROWNIANO_DAVID ALVAREZ_INCISO D).py

```python
for j in range(0,4):
    ###################Listas donde se almacenan las coordenadas de los puntos#############################
    Puntos=[]
    Puntos_x=[]
    Puntos_y=[]
    cuadro=dimension[j]

    ######################Almacena la primer coordenada que es el punto central#############################
    Primera=cuadro/2
    Puntos_x.append(Primera)
    Puntos_y.append(Primera)

    ####################Ciclo for que limita el rango de iteraciones##################
    for i in range(0,1000):
        P_x=random.randint(0,cuadro)
        P_y=random.randint(0,cuadro)
        Puntos_x.append(P_x)
        Puntos_y.append(P_y)
    largo=len(Puntos_x)
```




```python
for i in range(largo):

    ######Almacena los resultados de cada dimension en una lista y evalua si llega a 100 datos#################
    X=(Puntos_x[i],Puntos_y[i])
    Puntos.append(X)
    ##############################################################################################################
    if j==0:
        if (len(dimension_1))==100:
            break
        else:
            dimension_1.append(Puntos[i])
        if Puntos_x[i]==cuadro or Puntos_y[i]==cuadro:
            #print(f"Variable de dimension {dimension[j]}x{dimension[j]} se detuvo en la iteracion {i+1}.")
            borde1=borde1+1

        if Puntos_x[i]==0 or Puntos_y[i]==0:
            #print(f"Variable de dimension {dimension[j]}x{dimension[j]} se detuvo en la iteracion {i+1}.")
            borde1=borde1+1
    ##############################################################################################################
    if j==1:
        if (len(dimension_2))==100:
            break
        else:
            dimension_2.append(Puntos[i])
        if Puntos_x[i]==cuadro or Puntos_y[i]==cuadro:
            #print(f"Variable de dimension {dimension[j]}x{dimension[j]} se detuvo en la iteracion {i+1}.")
            borde2=borde2+1

        if Puntos_x[i]==0 or Puntos_y[i]==0:
            #print(f"Variable de dimension {dimension[j]}x{dimension[j]} se detuvo en la iteracion {i+1}.")
            borde2=borde2+1
    ##############################################################################################################
    if j==2:
        if (len(dimension_3))==100:
            break
        else:
            dimension_3.append(Puntos[i])
        if Puntos_x[i]==cuadro or Puntos_y[i]==cuadro:
            #print(f"Variable de dimension {dimension[j]}x{dimension[j]} se detuvo en la iteracion {i+1}.")
            borde3=borde3+1

        if Puntos_x[i]==0 or Puntos_y[i]==0:
            #print(f"Variable de dimension {dimension[j]}x{dimension[j]} se detuvo en la iteracion {i+1}.")
            borde3=borde3+1
    ##############################################################################################################
    if j==3:
        if (len(dimension_4))==100:
            break
        else:
            dimension_4.append(Puntos[i])
        if Puntos_x[i]==cuadro or Puntos_y[i]==cuadro:
            #print(f"Variable de dimension {dimension[j]}x{dimension[j]} se detuvo en la iteracion {i+1}.")
            borde4=borde4+1

        if Puntos_x[i]==0 or Puntos_y[i]==0:
            #print(f"Variable de dimension {dimension[j]}x{dimension[j]} se detuvo en la iteracion {i+1}.")
            borde4=borde4+1
```

Ahora se inicializa el ciclo for que sirve para que se vayan recorriendo los datos de las listas tanto con los puntos como con las dimensiones utilizadas, se almacenan los puntos en las listas dimensiones con las cuales se alamcena una primera coordenada la cual es la del centro de donde inicia para después con ciclos if evaluar si j el cual sirve para recorrer las dimensiones 50,100,200,500 es igual a la iteración 0,1,2 o 3 que corresponden al índice de su respectiva lista, ejecutando otras evaluaciones indicando que si se cumplen los 100 datos en esa caminata se detenga y sino seguir almacenando datos en las listas dimensiones así como también evaluando si alguno de los puntos toco el borde incrementando un contador que previamente se inicializo para indicar el numero de veces que esto paso.




```python
####################impresion de los valores#############################################33
print("Total de coordenadas almacenadas en una lista por cada dimension")
#print(dimension_1)
print(f"{dimension[0]}x{dimension[0]}: {len(dimension_1)} datos.")
print(f"Numero de veces que llego al borde: {borde1} veces.")
#print(dimension_2)
print(f"{dimension[1]}x{dimension[1]}: {len(dimension_2)} datos.")
print(f"Numero de veces que llego al borde: {borde2} veces.")
#print(dimension_3)
print(f"{dimension[2]}x{dimension[2]}: {len(dimension_3)} datos.")
print(f"Numero de veces que llego al borde: {borde3} veces.")
#print(dimension_4)
print(f"{dimension[3]}x{dimension[3]}: {len(dimension_4)} datos.")
print(f"Numero de veces que llego al borde: {borde4} veces.")
#NOTA: Si no se imprimen los valores de los bordes y solo aparecen de una sola dimension significa que no alcanzaron el borde en las 100 iteraciones.
##borde se refiere a las numero de veces que llego al borde entre los 100 datos de cada dimension
```

Justo después de que se terminen de almacenar los datos se ejecutaran los respectivos prints que imprimen los valores almacenados en las listas y en las variables quedando tal que así.

RESULTADO:

```
PS C:\Users\David> & C:/Users/David/AppData/Local/Programs/Python/
 D)/EXAMEN MOV BROWNIANO_DAVID ALVAREZ_INCISO D).py"
Total de coordenadas almacenadas en una lista por cada dimension
50x50: 100 datos.
Numero de veces que llego al borde: 7 veces.
100x100: 100 datos.
Numero de veces que llego al borde: 2 veces.
200x200: 100 datos.
Numero de veces que llego al borde: 1 veces.
500x500: 100 datos.
Numero de veces que llego al borde: 0 veces.
```

Indicando la dimensión y la cantidad de datos almacenados, así como también las veces que se toco el borde en esos 100 datos siendo que entre mas grande es la dimensión es menor la probabilidad de que se toque el borde.

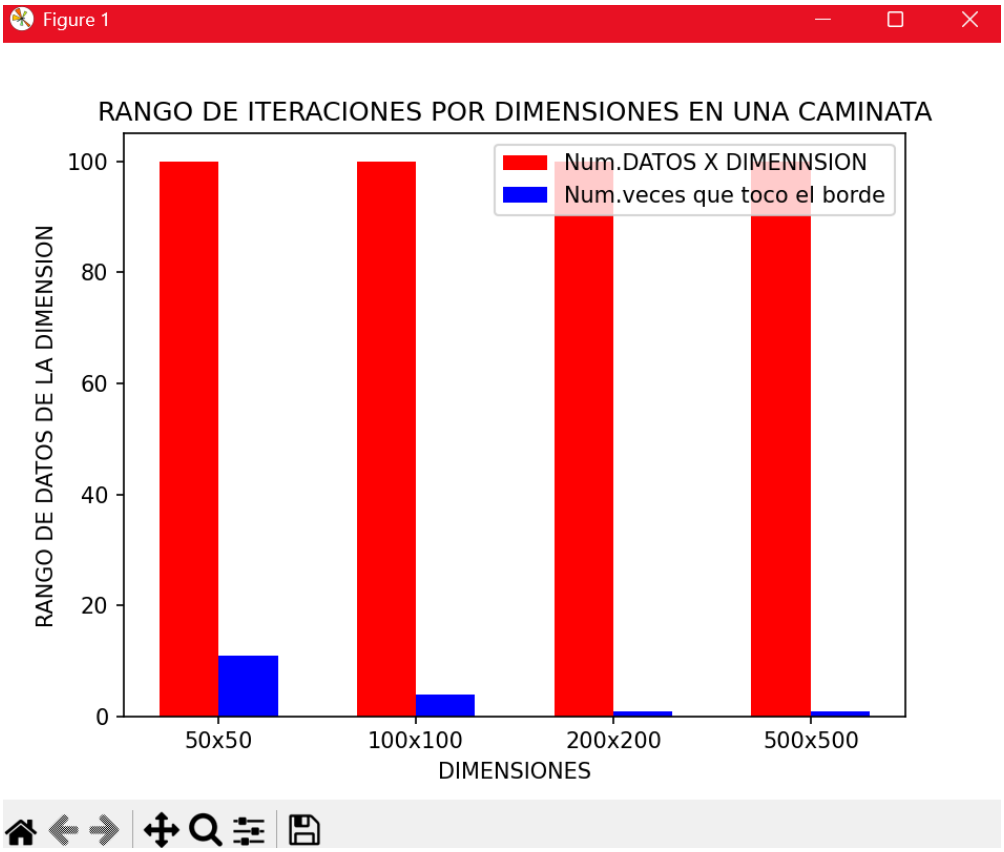
GRAFICO:

undefined - EXAMEN MOV BROWNIANO_DAVID ALVAREZ_INCISO D).py

```python
##########vectores que se utilizan en al grafica de barras#############
Vector_d=[f"{dimension[0]}x{dimension[0]}",f"{dimension[1]}x{dimension[1]}",f"{dimension[2]}x{dimension[2]}",f"{dimension[3]}x{dimension[3]}"]
Vector_L=[(len(dimension_1)),(len(dimension_2)),(len(dimension_3)),(len(dimension_4))]
Vector_b=[(borde1),(borde2),(borde3),(borde4)]
Vector_d_largo=np.arange(len(Vector_d))
Ancho=0.3

############GRAFICO DE BARRAS####################
plt.bar(Vector_d_largo-Ancho/2,Vector_L,color="red",label="Num.DATOS X DIMENNSION",width=Ancho)
plt.bar(Vector_d_largo+Ancho/2,Vector_b,color="blue",label="Num.veces que toco el borde",width=Ancho)
plt.xticks(Vector_d_largo,Vector_d)
plt.title("RANGO DE ITERACIONES POR DIMENSIONES EN UNA CAMINATA")
plt.xlabel("DIMENSIONES")
plt.ylabel("RANGO DE DATOS DE LA DIMENSION")
plt.legend()
plt.show()
```

Ahora bien, para generar el grafico de barras que se selecciono primer tenemos que tomar los datos que necesitamos los cuales almacenamos en 3 listas, las cuales almacenan las dimensiones, total de datos en las listas dimensiones y numero de veces que se toco el borde respectivamente. Utilizando también numpy para con np.arange tomar el valor del largo de la lista del vector para poder graficar 2 barras para un solo elemento en la gráfica, con plt. bar indicamos el desplazamiento para que no se empalmen uno con otro así como también la leyenda, el color y el ancho, así como también le ponemos un titulo y sus respectivos datos para el eje x y el eje y para saber de lo que trata la grafica y con plt.show mostramos el resultado el cual es el siguiente.

RESULTADO:



En este caso la grafica representa los 100 datos que se tienen en cada dimensión y el cómo a comparación con los datos de las veces que se alcanzó el borde se va reduciendo la frecuencia conforme la dimensión se va haciendo mas grande.

GITHUB: https://github.com/D4V1D216/PROGRAMACION-ORIENTADA-A-OBJETOS.git