```
import matplotlib.pyplot as plt
import numpy as np # importando numpy
import pandas as pd # importando pandas
from scipy import stats
import math
```

GENERACION DE LOS NUMEROS PSEUDOALEATORIOS

1. CUADRADOS MEDIOS

```
resultados_cuadrados = []
xo = 74731897457
digitos = 7
iteraciones = 100
xn = xo

for i in range(iteraciones):
  xnn = xn**2
  txnn = str(xnn).zfill(8)
  tam = len(txnn)
  ui = int(txnn[int(tam/2-digitos/2):int(tam/2+digitos/2)])
  rn = ui / (int('9'*digitos)+1)
  resultados_cuadrados.append(rn)
  xn = ui

resultados_cuadrados
```

```
      0.9083583,
      0.1148011,
      0.7929256,
      0.7310071,
      0.3713802,
      0.9232529,
      0.3959173,
      0.7505084,
      0.2628584,
      0.9453845,
      0.7518528,
      0.2826328,
      0.8129963,
      0.9629838,
      0.337799,
      0.1081644,
      0.9953742,
      0.769798,
      0.5889608,
      0.8748239,
      0.316856,
      0.3977247,
      0.1849369,
      0.0165698,
      0.4558272,
      0.7784362,
      0.9629174
```

```
0.9629174,
0.2099192,
0.6607052,
0.5313613,
0.3448311,
0.9084875,
0.3495376,
0.1765338,
0.6418254,
0.939844,
0.3067443,
0.9206558,
0.607102,
0.5728384,
0.1438325,
0.8778805,
0.6741722,
0.5081552,
0.2217072,
0.5408253,
0.4920051,
0.0690184,
0.6353953,
0.7271872,
0.8012238,
0.9595776,
0.7891704,
0.7899202,
0.9739223,
0.5246464,
0.253845,
0.3728402,
0.0098147]
```

2. CONGRUENCIA LINEAL

```
resultados_congruencia=[]
xo = 7
a = 74731897457
b = 37747318974
m = 19
iteraciones = 100
xn = xo
for i in range(iteraciones):
  xnn = (a*xn+b) % m
  ui = xnn/m
  resultados_congruencia.append(ui)
  xn = xnn
resultados_congruencia
```

```
0.3157894736842105,
0.0,
0.631578947368421,
0.3684210526315789,
0.8947368421052632,
0.8421052631578947,
0.9473684210526315,
0.7368421052631579,
0.1578947368421052]
```

```
        0.15789473684210525,
        0.3157894736842105,
        0.0,
        0.631578947368421,
        0.3684210526315789,
        0.8947368421052632,
        0.8421052631578947,
        0.9473684210526315,
        0.7368421052631579,
        0.15789473684210525,
        0.3157894736842105,
        0.0,
        0.631578947368421,
        0.3684210526315789,
        0.8947368421052632,
        0.8421052631578947,
        0.9473684210526315,
        0.7368421052631579,
        0.15789473684210525,
        0.3157894736842105,
        0.0,
        0.631578947368421,
        0.3684210526315789,
        0.8947368421052632,
        0.8421052631578947,
        0.9473684210526315,
        0.7368421052631579,
        0.15789473684210525,
        0.3157894736842105,
        0.0,
        0.631578947368421,
        0.3684210526315789,
        0.8947368421052632,
        0.8421052631578947,
        0.9473684210526315,
        0.7368421052631579,
        0.15789473684210525,
        0.3157894736842105,
        0.0,
        0.631578947368421,
        0.3684210526315789,
        0.8947368421052632,
        0.8421052631578947,
        0.9473684210526315,
        0.7368421052631579,
        0.15789473684210525,
        0.3157894736842105,
        0.0,
        0.631578947368421,
        0.3684210526315789,
        0.8947368421052632]
```

3 LIBRERIAS

```
np.random.seed(12082021) # para poder replicar el random
resultados_libreria= np.random.uniform(0, 1, size=(2, 100))
```

```
resultados_libreria=resultados_libreria[0]
resultados_libreria
```

```
array([0.0213125 , 0.1925147 , 0.65022317, 0.59901792, 0.96318574,
       0.94105618, 0.72789031, 0.0947742 , 0.19688377, 0.77112984,
       0.72658558, 0.93959203, 0.92571122, 0.42733289, 0.07023847,
       0.34217477, 0.52154158, 0.60041226, 0.90333886, 0.76506077,
       0.86892796, 0.21763692, 0.53873561, 0.67958618, 0.84539166,
       0.61548304, 0.38370923, 0.63574691, 0.23378763, 0.41329067,
       0.07071273, 0.31224262, 0.20953384, 0.53652589, 0.57905369,
       0.39814381, 0.59827327, 0.20438962, 0.02074584, 0.25938279,
       0.86849638, 0.39662733, 0.3421929 , 0.14355372, 0.19711259,
       0.5059967 , 0.03546392, 0.29331827, 0.90547185, 0.80765768,
       0.50734945, 0.28777342, 0.61602177, 0.90067799, 0.22150541,
       0.91321866, 0.07154236, 0.92843868, 0.71569544, 0.93907898,
       0.19859609, 0.48317233, 0.96721161, 0.70322813, 0.22804718,
       0.36413519, 0.93205421, 0.98742882, 0.24431537, 0.8584519 ,
       0.44855912, 0.82888944, 0.35179667, 0.75280866, 0.30556722,
       0.38674451, 0.19351149, 0.23589404, 0.63790189, 0.57603676,
       0.48283931, 0.06693581, 0.08359963, 0.9909249 , 0.72531107,
       0.09937889, 0.04179326, 0.87970611, 0.19575825, 0.93669143,
       0.9545418 , 0.09976707, 0.07734505, 0.72275603, 0.56639226,
```

Dividir las series en grados de libertad

```
n = math.sqrt(100)
intervalos = []
rango=1/10
for i in range(1,11):
  intervalos.append(round(rango*i,2))
intervalos
```

```
[0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
```

Obtener las frecuencias de cada lista pseudoaleatoria

Metodo de conteo

```
def obtener_frecuencia(lista_numeros):
  frecuencias_general=[]
  contador=0
  for indice in range(len(intervalos)):

    for valor in lista_numeros:
      if(indice==0):
        if(valor<=intervalos[indice]):
          contador+=1

      else:
        if(valor<=intervalos[indice] and valor>intervalos[indice-1]):
          contador+=1
    frecuencias_general.append(contador)
    contador=0
  return frecuencias_general
```

```
return frecuencias_general
```

1. Frecuencia de cuadrados medios

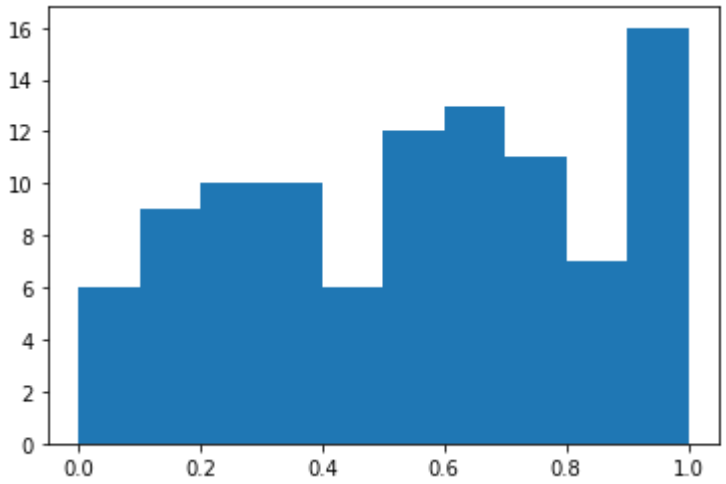
+ Código + Texto

```
frecuencias_cuadrados = []
frecuencias_cuadrados = obtener_frecuencia(resultados_cuadrados)
frecuencias_cuadrados
```

```
[6, 9, 10, 10, 6, 12, 13, 11, 7, 16]
```

Graficos de la frecuencia encontrada.

```
plt.hist(x=resultados_cuadrados, bins=10,range=(0,1))
plt.show()
```



2. Frecuencia de congruencia lineal

```
frecuencias_congruencia = obtener_frecuencia(resultados_congruencia)
frecuencias_congruencia
```

```
[11, 11, 0, 22, 0, 0, 11, 11, 23, 11]
```

Grafico de frecuencias

```
plt.hist(x=resultados_congruencia, bins=10,range=(0,1))
plt.show()
```

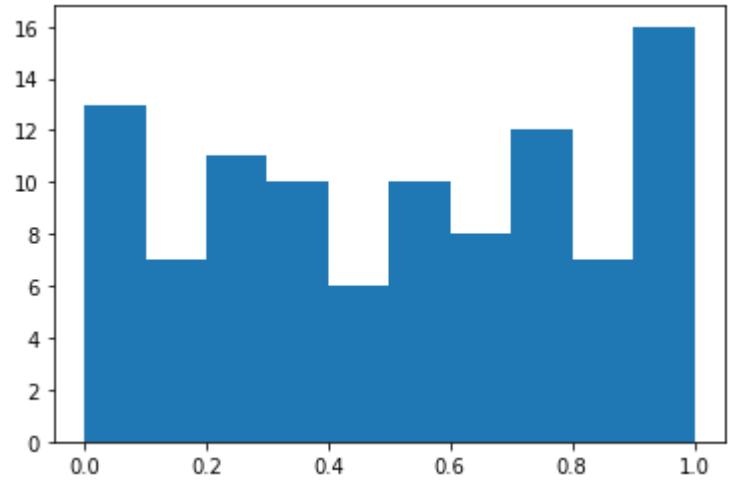
3. Frencuencias de libreria


```
frecuencias_libreria = obtener_frecuencia(resultados_libreria)
frecuencias_libreria
```

```
[13, 7, 11, 10, 6, 10, 8, 12, 7, 16]
```

Grafico

```
plt.hist(x=resultados_libreria, bins=10,range=(0,1))
plt.show()
```



Obtencion del Chi-Cuadrado

valor de chi por tabla

```
chi_cuadrado=18.307
```

definicion del metodo

```
def obtener_chi(frecuencias):
  resultados_chi_frecuencias=[]
  sumatoria=0
  for i in frecuencias:
    valor=((i-10)*(i-10))/10
    print(valor)
```

```
    resultados_chi_frecuencias.append(valor)
    sumatoria=sumatoria+valor


  return sumatoria
```

Cuadrados medios

```
valor_cuadrados= obtener_chi(frecuencias_cuadrados)
print(valor_cuadrados)
if(valor_cuadrados<chi_cuadrado):
  print('Es valido')
else:
  print('NO es valido')
```

```
    1.6
    0.1
    0.0
    0.0
    1.6
    0.4
    0.9
    0.1
    0.9
    3.6
    9.200000000000001
    Es valido
```

Conguencia lineal

```
valor_cuadrados= obtener_chi(frecuencias_congruencia)
print(valor_cuadrados)
if(valor_cuadrados<chi_cuadrado):
  print('Es valido')
else:
  print('NO es valido')
```

```
    0.1
    0.1
    10.0
    14.4
    10.0
    10.0
    0.1
    0.1
    16.9
    0.1
    61.800000000000004
    NO es valido
```

Libreria

```
valor_cuadrados= obtener_chi(frecuencias_libreria)
print(valor_cuadrados)
if(valor_cuadrados<chi_cuadrado):
  print('Es valido')
else:
  print('NO es valido')
```

```
0.9
0.9
0.1
0.0
1.6
0.0
0.4
0.4
0.9
3.6
8.8
Es valido
```

✓ 0 s completado a las 13:34