Objetivo:

```
Consolidar los conocimientos adquiridos generación de números pseudoaleatorios.
```

Introducción: Es fundamental verificar la calidad de los números pseudoaleatorios. Ademas es importante no olvidar las 2 propiedades más importantes que deben tener los números pseudoaleatorios: uniformidad e independencia.

La uniformidad se puede verificar usando las pruebas de bondad de ajuste test Chi Cuadrada

Chi-Cuadrada

Esta prueba verifica la desviación del valor esperado y se usa cuando se trabaja con variables nominales (categorías o grupos) Debemos responder a la pregunta: ¿Difieren las frecuencias observadas de la frecuencia esperada?

Pasos para aplicar la prueba:

1. Tomar la serie de N números pseudo-aleatorios.
2. Dividir la serie en n intervalos (grados libertad)
3. Calcular la esperanza E_i=N/n
4. Calcular la cantidad de números observados por intervalo

O_i

1. Calcular Chi – Cuadrado: $\chi 0^2=(\sum(i=1)^k \ [(O_i-E_i)] \ ^2)/E_i$
2. Si $\chi 0^2 \leqslant \chi(k-1)^2$ se acepta H0 (los números están distribuidos uniformemente) Las semillas para generar los números son las siguientes:

   Cuadrados medios: Xo=74731897457, D=7 Congruencia lineal: a=74731897457, b=37747318974, Xo=7, M=19

La gráfica deberá analizar las distribuciones generadas por ambos métodos y compararlas con la ideal. Cuaderno de Python y generar el PDF.

In [23]:
```
%matplotlib inline
import numpy as np
import matplotlib.pyplot as plt
```

In [24]:
```
from scipy.stats import chisquare

semilla =74731897457
tam=7

tam1 = int(tam)
numero1 = int(semilla)
valores =[]
for i in range(100):
    numero2 = (numero1**2)
    snumero2 = str(numero2)
    tam2 = len(snumero2)
    numero=int((tam2/2)-(tam1/2)+1)
    sNumero=str(numero2)
    numUI=sNumero[numero-1:numero+tam1]
    Ui=int(numUI)
    numero1 = int(Ui)
    Rn=Ui/10000
    print(Rn)
    valores.append(Rn/1000)
```

```
4975.2356
2969.2755
5969.949
291.0626
1743.7118
5308.4145
9264.5038
1030.6602
2604.4786
3087.7785
3760.6506
2492.9352
7259.1139
4734.6131
6561.2066
9432.0478
3525.7014
570.3619
3126.9697
9395.0471
6910.0112
8254.7841
1460.5376
1700.8101
7549.9626
1935.2613
2362.9927
7345.0025
9061.725
4859.9756
9362.8325
2632.423
6508.5092
692.0064
8728.5764
8045.9706
7642.896
3859.2668
3940.2335
5440.0345
3975.3611
3495.8753
1144.1131
9947.8559
9837.0071
6708.6854
6459.7961
8965.6535
2942.682
3773.5312
9537.7173
8051.2947
3346.3462
8032.8902
7324.9652
5115.1812
5078.7088
```

```
3283.0751
8582.1122
2649.8133
5105.2485
3562.2467
9601.5516
9793.1274
5344.2726
1249.6231
5578.9205
4353.9453
6839.6753
1158.2094
4490.1424
1378.7722
127.7949
3153.6466
4868.7769
4988.5019
5151.2063
4926.3451
8876.0442
4160.6403
927.7059
6382.3689
4632.7756
2609.7599
8467.3564
6124.4046
8331.7045
7299.8753
8179.3955
2510.7454
8424.6362
4495.1023
5944.6874
9308.2837
4145.4396
4669.4772
4017.3213
8870.4274
4482.2586
642.1572
```

In [25]:
```
chisquare(valores, ddof=0.05)
```

Out[25]:
```
Power_divergenceResult(statistic=144.7327699014051, pvalue=0.0018572813746324
72)
```

In [26]:
```
from scipy.stats import norm
def plot_function(size = 100, bins = 10, loc=0, scale=1, color='red'):
    data = norm.rvs(size=size, loc=loc, scale=scale)
    binwidth = (max(data) - min(data))/ bins
    plt.hist(data,
             bins=np.arange(min(data), max(data) + binwidth, binwidth),
             color=color)
    plt.show()
```

Congruencia lineal: a=74731897457, b=37747318974, Xo=7, M=19

In [27]:
```
import time
xn = int(time.time())
a=0.9999999999997491
b=37747318974.00
x0=1234567
valores =[]
for i in range(100):
    xn1 = ((a * xn + x0) % b) / b
    valores.append(xn1)
    print(xn1)
    xn = xn1

print(len(valores))
```

```
0.042731722009465636
3.270608552575828e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
```

```
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
3.270608439457818e-05
100
```

In [28]:
```
chisquare(valores, ddof=19)
```

Out[28]:
```
Power_divergenceResult(statistic=3.9264490982161515, pvalue=1.0)
```

resultados se mantiene la uniformidad e independecia ya que los valores de prueba estan en los rangos esperados con un valor estadistico de 3.9264490982161515

Referencias

[1] : https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6291769/ (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6291769/)