Prueba Chi Cuadrada

Nombre: David Egas

Realice un programa que permita calcular el valor de Chi-Cuadrada y genere la gráfica de distribución de los 100 primeros números pseudo-aleatorios generados por los métodos de cuadrados medios y productos medios.

Emplee el siguiente nivel de significancia $\alpha=0.05$

Para primer método de numeros PseudoAleatorios

In [35]:
```
import collections
def divisionValores(long,numeroOriginal,digitos):
    numero=int((long/2)-(digitos/2)+1)
    sNumero=str(numeroOriginal)
    numUI=sNumero[numero-1:numero+3]
    return int(numUI)
```

In [36]:
```
def buscarSemilla(valor,lista):
    for x in range(0,len(lista)):
        if (lista[x]==valor):
               return True
```

In [38]:
```
listaFinal = []
Rn=0
def generarNumeroPseudoaleatorios(iteraciones,semilla):
    iteraciones=iteraciones
    Xo=semilla
    digitos=7
    Xn=0
    XnPo=0
    longuitud=0
    Ui=0

    arregloSemillas=[]
    semillasRepetidas=[]

    if len(str(Xo))>3:
        print('PROCESO NORMAL')
        for i in range(iteraciones):
            if (i==0):
                Xn=Xo
                XnPo=Xn**2
                longuitud=len(str(XnPo))
                Ui=divisionValores(longuitud,XnPo,digitos)
                arregloSemillas.append(Xn)
                Rn=Ui/10000
                print(Rn)

            else:
                Xn=Ui
                XnPo=Xn**2
                longuitud=len(str(XnPo))
```

```
                Ui=divisionValores(longuitud,XnPo,digitos)
                Rn=Ui/10000
                print(Rn)
                listaFinal.append(Rn)

                busquedaSemilla=buscarSemilla(Xn,arregloSemillas)
                if(busquedaSemilla==True):
                    semillasRepetidas.append(Xn)
                    print('LA SEMILLA SE VUELE A REPETIR EN LA ITERACION:',i,'VALOR:',Xn)
                arregloSemillas.append(Xn)

    else:
        print('LA SEMILLA ES MENOR A 3')

    print('LA FRECUENCIA DE REPETICION ES DE:')
    #Vemos la frecuencia de repeticion
    counter=collections.Counter(semillasRepetidas)
    print(counter)
```

In [39]:
```
iteraciones=100
semilla=74731897457
generarNumeroPseudoaleatorios(iteraciones,semilla)
```

```
PROCESO NORMAL
0.4975
0.2475
0.6125
0.3751
0.1407
0.1979
0.3916
0.1533
0.235
0.5522
0.3049
0.9296
0.8641
0.7466
0.5574
0.3106
0.9647
0.9306
0.866
0.7499
0.5623
0.3161
0.9991
0.9982
0.9964
0.9928
0.9856
0.9714
0.9436
0.8903
0.7926
0.6282
0.3946
0.1557
0.2424
0.5875
0.3451
0.119
0.1416
```

```
0.2005
0.402
0.1616
0.2611
0.6817
0.4647
0.2159
0.4661
0.2172
0.4717
0.2225
0.495
0.245
0.6002
0.3602
0.1297
0.1682
0.2829
0.8003
0.6404
0.4101
0.1681
0.2825
0.798
0.6368
0.4055
0.1644
0.2702
0.73
0.5329
0.2839
0.8059
0.6494
0.4217
0.1778
0.3161
0.9991
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 75 VALOR: 3161
0.9982
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 76 VALOR: 9991
0.9964
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 77 VALOR: 9982
0.9928
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 78 VALOR: 9964
0.9856
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 79 VALOR: 9928
0.9714
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 80 VALOR: 9856
0.9436
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 81 VALOR: 9714
0.8903
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 82 VALOR: 9436
0.7926
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 83 VALOR: 8903
0.6282
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 84 VALOR: 7926
0.3946
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 85 VALOR: 6282
0.1557
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 86 VALOR: 3946
0.2424
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 87 VALOR: 1557
0.5875
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 88 VALOR: 2424
0.3451
```

```
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 89 VALOR: 5875
0.119
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 90 VALOR: 3451
0.1416
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 91 VALOR: 1190
0.2005
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 92 VALOR: 1416
0.402
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 93 VALOR: 2005
0.1616
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 94 VALOR: 4020
0.2611
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 95 VALOR: 1616
0.6817
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 96 VALOR: 2611
0.4647
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 97 VALOR: 6817
0.2159
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 98 VALOR: 4647
0.4661
LA SEMILLA SE VUELE A REPETIR EN LA ITERACION: 99 VALOR: 2159
LA FRECUENCIA DE REPETICION ES DE:
Counter({3161: 1, 9991: 1, 9982: 1, 9964: 1, 9928: 1, 9856: 1, 9714: 1, 9436: 1, 8903: 1,
7926: 1, 6282: 1, 3946: 1, 1557: 1, 2424: 1, 5875: 1, 3451: 1, 1190: 1, 1416: 1, 2005: 1,
4020: 1, 1616: 1, 2611: 1, 6817: 1, 4647: 1, 2159: 1})
```

In [40]:
```
chisquare(lista, ddof=0.05)
```

Out[40]:
```
Power_divergenceResult(statistic=1.162162162162162, pvalue=0.9968424171841995)
```

El valor estadístico es: 1.162162162162162 y el valor de prueba es 0.9968424171841995 por lo que los números siguen una distribución uniforme.

Para el segundo método de números PseudoAleatorios

In [42]:
```
import numpy as np
import time


rango = int((input("Ingrese la cantidad de numeros pseudoaleatorios:  ")))

a = 74731897457
b = 37747318974
m = 19

cont = 0
lista = []
contInterac = []

for n in range(rango):
    semilla = int(time.time())

    print("semilla",semilla,"iteracion",n)
    cont = 0
    while(True):

        xn = (a * semilla + b) % m
        semilla = xn
        div = semilla / m

        #print(cont,'\t\t',semilla,'\t',"{0:.4f}".format(div),(semilla in lista) is True)
```

```
        if((div in lista) is True):

            contInterac.append(cont)

            break

        cont = cont+1
        lista.append(div)
```

```
Ingrese la cantidad de numeros pseudoaleatorios:  100
semilla 1611767638 iteracion 0
semilla 1611767638 iteracion 1
semilla 1611767638 iteracion 2
semilla 1611767638 iteracion 3
semilla 1611767638 iteracion 4
semilla 1611767638 iteracion 5
semilla 1611767638 iteracion 6
semilla 1611767638 iteracion 7
semilla 1611767638 iteracion 8
semilla 1611767638 iteracion 9
semilla 1611767638 iteracion 10
semilla 1611767638 iteracion 11
semilla 1611767638 iteracion 12
semilla 1611767638 iteracion 13
semilla 1611767638 iteracion 14
semilla 1611767638 iteracion 15
semilla 1611767638 iteracion 16
semilla 1611767638 iteracion 17
semilla 1611767638 iteracion 18
semilla 1611767638 iteracion 19
semilla 1611767638 iteracion 20
semilla 1611767638 iteracion 21
semilla 1611767638 iteracion 22
semilla 1611767638 iteracion 23
semilla 1611767638 iteracion 24
semilla 1611767638 iteracion 25
semilla 1611767638 iteracion 26
semilla 1611767638 iteracion 27
semilla 1611767638 iteracion 28
semilla 1611767638 iteracion 29
semilla 1611767638 iteracion 30
semilla 1611767638 iteracion 31
semilla 1611767638 iteracion 32
semilla 1611767638 iteracion 33
semilla 1611767638 iteracion 34
semilla 1611767638 iteracion 35
semilla 1611767638 iteracion 36
semilla 1611767638 iteracion 37
semilla 1611767638 iteracion 38
semilla 1611767638 iteracion 39
semilla 1611767638 iteracion 40
semilla 1611767638 iteracion 41
semilla 1611767638 iteracion 42
semilla 1611767638 iteracion 43
semilla 1611767638 iteracion 44
semilla 1611767638 iteracion 45
semilla 1611767638 iteracion 46
semilla 1611767638 iteracion 47
semilla 1611767638 iteracion 48
semilla 1611767638 iteracion 49
semilla 1611767638 iteracion 50
```

```
semilla 1611767638 iteracion 51
semilla 1611767638 iteracion 52
semilla 1611767638 iteracion 53
semilla 1611767638 iteracion 54
semilla 1611767638 iteracion 55
semilla 1611767638 iteracion 56
semilla 1611767638 iteracion 57
semilla 1611767638 iteracion 58
semilla 1611767638 iteracion 59
semilla 1611767638 iteracion 60
semilla 1611767638 iteracion 61
semilla 1611767638 iteracion 62
semilla 1611767638 iteracion 63
semilla 1611767638 iteracion 64
semilla 1611767638 iteracion 65
semilla 1611767638 iteracion 66
semilla 1611767638 iteracion 67
semilla 1611767638 iteracion 68
semilla 1611767638 iteracion 69
semilla 1611767638 iteracion 70
semilla 1611767638 iteracion 71
semilla 1611767638 iteracion 72
semilla 1611767638 iteracion 73
semilla 1611767638 iteracion 74
semilla 1611767638 iteracion 75
semilla 1611767638 iteracion 76
semilla 1611767638 iteracion 77
semilla 1611767638 iteracion 78
semilla 1611767638 iteracion 79
semilla 1611767638 iteracion 80
semilla 1611767638 iteracion 81
semilla 1611767638 iteracion 82
semilla 1611767638 iteracion 83
semilla 1611767638 iteracion 84
semilla 1611767638 iteracion 85
semilla 1611767638 iteracion 86
semilla 1611767638 iteracion 87
semilla 1611767638 iteracion 88
semilla 1611767638 iteracion 89
semilla 1611767638 iteracion 90
semilla 1611767638 iteracion 91
semilla 1611767638 iteracion 92
semilla 1611767638 iteracion 93
semilla 1611767638 iteracion 94
semilla 1611767638 iteracion 95
semilla 1611767638 iteracion 96
semilla 1611767638 iteracion 97
semilla 1611767638 iteracion 98
semilla 1611767638 iteracion 99
```

In [43]:
```
from scipy.stats import chisquare
print(lista)
chisquare(lista, ddof=0.05)
```

```
[0.2631578947368421, 0.10526315789473684, 0.42105263157894735, 0.7894736842105263, 0.0526
3157894736842, 0.5263157894736842, 0.5789473684210527, 0.47368421052631576, 0.68421052631
57895]
```

Out[43]:
```
Power_divergenceResult(statistic=1.1621621621621623, pvalue=0.9968424171841995)
```

Resultados:

El valor estadístico es: 1.1621621621621623 y el valor de prueba es 0.9968424171841995 por lo que los números siguen una distribución uniforme.

In [ ]: