metodo-cuadrados-medios

July 13, 2020

0.1 Realizar el metodo de cuadrados medios para generar numeros randomicos

[90]:
```
import math
import pandas as pd

cantidad = 50

x0 = 43242 ##1000 - 9999
digitos = len(str(x0))
ui = ''
xn = x0
df = pd.DataFrame(columns = ['Xn','digitos','Xn2','longitud','UI','Rn'], index
 ↪= range(cantidad));

def divisor(numero):
    resultado = '1'
    for i in range(len(str(numero))):
        resultado+='0'
    return int(resultado)

for i in range(cantidad):
    xn2 = str(int(math.pow(xn,2)))
    longitud = (len(xn2))
    if(longitud % 2 != 0):
        xn2="0"+xn2
        xn ="0"+str(xn)
    longitud = (len(xn2))
    digitos = len(str(xn))
    ui = xn2[int((longitud/2)-digitos/2):int((longitud/2)+digitos/2)]
    rn = int(ui)/divisor(xn)
    df.iloc[i] = [xn,digitos,xn2,longitud,ui,rn]
    xn=int(ui)
print(df)
print("\nAutor: Hernan Leon")
```

```
          Xn digitos          Xn2 longitud     UI      Rn
0      43242       5   1869870564       10  69870  0.6987
1      69870       5   4881816900       10  81816 0.81816
```

2	81816	5	6693857856	10	93857	0.93857
3	93857	5	8809136449	10	09136	0.09136
4	9136	4	83466496	8	4664	0.4664
5	4664	4	21752896	8	7528	0.7528
6	7528	4	56670784	8	6707	0.6707
7	6707	4	44983849	8	9838	0.9838
8	9838	4	96786244	8	7862	0.7862
9	7862	4	61811044	8	8110	0.811
10	8110	4	65772100	8	7721	0.7721
11	7721	4	59613841	8	6138	0.6138
12	6138	4	37675044	8	6750	0.675
13	6750	4	45562500	8	5625	0.5625
14	5625	4	31640625	8	6406	0.6406
15	6406	4	41036836	8	0368	0.0368
16	368	3	135424	6	354	0.354
17	354	3	125316	6	253	0.253
18	0253	4	064009	6	6400	0.64
19	6400	4	40960000	8	9600	0.96
20	9600	4	92160000	8	1600	0.16
21	01600	5	02560000	8	25600	0.256
22	025600	6	0655360000	10	553600	0.5536
23	553600	6	306472960000	12	472960	0.47296
24	472960	6	223691161600	12	691161	0.691161
25	691161	6	477703527921	12	703527	0.703527
26	703527	6	494950239729	12	950239	0.950239
27	950239	6	902954157121	12	954157	0.954157
28	954157	6	910415580649	12	415580	0.41558
29	415580	6	172706736400	12	706736	0.706736
30	706736	6	499475773696	12	475773	0.475773
31	475773	6	226359947529	12	359947	0.359947
32	359947	6	129561842809	12	561842	0.561842
33	561842	6	315666432964	12	666432	0.666432
34	666432	6	444131610624	12	131610	0.13161
35	0131610	7	017321192100	12	7321192	0.732119
36	7321192	7	53599852300864	14	9985230	0.998523
37	9985230	7	99704818152900	14	0481815	0.0481815
38	481815	6	232145694225	12	145694	0.145694
39	0145694	7	021226741636	12	1226741	0.122674
40	01226741	8	01504893481081	14	04893481	0.0489348
41	4893481	7	23946156297361	14	4615629	0.461563
42	4615629	7	21304031065641	14	0403106	0.0403106
43	403106	6	162494447236	12	494447	0.494447
44	494447	6	244477835809	12	477835	0.477835
45	477835	6	228326287225	12	326287	0.326287
46	326287	6	106463206369	12	463206	0.463206
47	463206	6	214559798436	12	559798	0.559798
48	559798	6	313373800804	12	373800	0.3738
49	373800	6	139726440000	12	726440	0.72644

Autor: Hernan Leon

0.2 Conclusiones

Tiene una fuerte tendencia a degenerar a cero rapidamente

Los números generados pueden repetirse cíclicamente después de una secuencia corta

La utilización de números primos puede generar ciclos más largos en la generación de números pseudoaleatorios