

# 10.CuadradosMedios

January 12, 2021

**Deber Cuadrados Medios** Entornos de soporte al desarrollo de simulaciones

Universidad “Politecnica Salesiana”

Alumno: Juan Cañar. Docente: Ing. Diego Quisi.

## 1.Con un valor de 8370

```
[7]: from collections import Counter
from collections import defaultdict
import random
import psutil
import numpy as np
import pandas as pd
import math

numero = int(input("Ingrese Xo: "))
print("Semilla:", numero)

digito=int(input("Ingrese # digitos: "))
print("digito: ", digito)

iteraciones = int(input("Ingrese # de iteraciones: "))
print("iteraciones:", iteraciones)

xn=[]
ui=[]
multiplicacion=[]
rn=[]
def centros(mul):
    cortarI=int(digito/2)
    cortarD=digito-cortarI
    mitad=math.floor(len(mul)/2)
    unir=''
    for i in range(mitad-cortarI, mitad+cortarD, 1):
        unir=unir+mul[i]
    ui.append(unir)
    return unir

def cuadrado(num):
```

```

multi=(num*num)
m=str(multi)
lon=len(m)
if(len(m)%2!=0):
    if (lon < len(m)+1):
        m=str(m).zfill(len(m)+1)
multiplicacion.append(m)
return m

def dividido(n):
    ceros=[int(str(num).ljust(digito+1, "0")) for num in [1]]
    res=n/ceros[0]
    rn.append(res)
    return res

for i in range(iteraciones):
    m=str(cuadrado(int(numero)))
    if(len(m)-1>digito and int(numero)>0):
        xn.append(numero)
        dividido(int(centros(m)))
        numero=ui[-1]
    else:
        print('-Datos Erroneos')
        break

df=pd.DataFrame({"Semilla Xn":xn, "Xn x Xn":multiplicacion, "UI ":ui, "RN":rn})
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)
print(df)

```

Ingrese Xo: 8370

Semilla: 8370

Ingrese # digitos: 4

digito: 4

Ingrese # de iteraciones: 100

iteraciones: 100

	Semilla Xn	Xn x Xn	UI	RN
0	8370	70056900	0569	0.0569
1	0569	323761	2376	0.2376
2	2376	05645376	6453	0.6453
3	6453	41641209	6412	0.6412
4	6412	41113744	1137	0.1137
5	1137	01292769	2927	0.2927
6	2927	08567329	5673	0.5673
7	5673	32182929	1829	0.1829

8	1829	03345241	3452	0.3452
9	3452	11916304	9163	0.9163
10	9163	83960569	9605	0.9605
11	9605	92256025	2560	0.2560
12	2560	06553600	5536	0.5536
13	5536	30647296	6472	0.6472
14	6472	41886784	8867	0.8867
15	8867	78623689	6236	0.6236
16	6236	38887696	8876	0.8876
17	8876	78783376	7833	0.7833
18	7833	61355889	3558	0.3558
19	3558	12659364	6593	0.6593
20	6593	43467649	4676	0.4676
21	4676	21864976	8649	0.8649
22	8649	74805201	8052	0.8052
23	8052	64834704	8347	0.8347
24	8347	69672409	6724	0.6724
25	6724	45212176	2121	0.2121
26	2121	04498641	4986	0.4986
27	4986	24860196	8601	0.8601
28	8601	73977201	9772	0.9772
29	9772	95491984	4919	0.4919
30	4919	24196561	1965	0.1965
31	1965	03861225	8612	0.8612
32	8612	74166544	1665	0.1665
33	1665	02772225	7722	0.7722
34	7722	59629284	6292	0.6292
35	6292	39589264	5892	0.5892
36	5892	34715664	7156	0.7156
37	7156	51208336	2083	0.2083
38	2083	04338889	3388	0.3388
39	3388	11478544	4785	0.4785
40	4785	22896225	8962	0.8962
41	8962	80317444	3174	0.3174
42	3174	10074276	0742	0.0742
43	0742	550564	5056	0.5056
44	5056	25563136	5631	0.5631
45	5631	31708161	7081	0.7081
46	7081	50140561	1405	0.1405
47	1405	01974025	9740	0.9740
48	9740	94867600	8676	0.8676
49	8676	75272976	2729	0.2729
50	2729	07447441	4474	0.4474
51	4474	20016676	0166	0.0166
52	0166	027556	2755	0.2755
53	2755	07590025	5900	0.5900
54	5900	34810000	8100	0.8100
55	8100	65610000	6100	0.6100

56	6100	37210000	2100	0.2100
57	2100	04410000	4100	0.4100
58	4100	16810000	8100	0.8100
59	8100	65610000	6100	0.6100
60	6100	37210000	2100	0.2100
61	2100	04410000	4100	0.4100
62	4100	16810000	8100	0.8100
63	8100	65610000	6100	0.6100
64	6100	37210000	2100	0.2100
65	2100	04410000	4100	0.4100
66	4100	16810000	8100	0.8100
67	8100	65610000	6100	0.6100
68	6100	37210000	2100	0.2100
69	2100	04410000	4100	0.4100
70	4100	16810000	8100	0.8100
71	8100	65610000	6100	0.6100
72	6100	37210000	2100	0.2100
73	2100	04410000	4100	0.4100
74	4100	16810000	8100	0.8100
75	8100	65610000	6100	0.6100
76	6100	37210000	2100	0.2100
77	2100	04410000	4100	0.4100
78	4100	16810000	8100	0.8100
79	8100	65610000	6100	0.6100
80	6100	37210000	2100	0.2100
81	2100	04410000	4100	0.4100
82	4100	16810000	8100	0.8100
83	8100	65610000	6100	0.6100
84	6100	37210000	2100	0.2100
85	2100	04410000	4100	0.4100
86	4100	16810000	8100	0.8100
87	8100	65610000	6100	0.6100
88	6100	37210000	2100	0.2100
89	2100	04410000	4100	0.4100
90	4100	16810000	8100	0.8100
91	8100	65610000	6100	0.6100
92	6100	37210000	2100	0.2100
93	2100	04410000	4100	0.4100
94	4100	16810000	8100	0.8100
95	8100	65610000	6100	0.6100
96	6100	37210000	2100	0.2100
97	2100	04410000	4100	0.4100
98	4100	16810000	8100	0.8100
99	8100	65610000	6100	0.6100

```
[34]: counts_por_elem = Counter(rn)
```

```

indices_por_elem = defaultdict(list)
indices = []

for indice, elem in enumerate(rn):
    if counts_por_elem[elem] > 1:
        indices.append(indice)
        indices_por_elem[elem].append(indice)

print("-----")
print("RESULTADOS: ")
print(indices_por_elem)
print("-----")
print("Frecuencia de iteracciones para que se repita la semilla")
print("-----")
print("Frecuencia de iteracciones: ",indices[0])

```

-----  
RESULTADOS:

```

defaultdict(<class 'list'>, {0.81: [54, 58, 62, 66, 70, 74, 78, 82, 86, 90, 94,
98], 0.61: [55, 59, 63, 67, 71, 75, 79, 83, 87, 91, 95, 99], 0.21: [56, 60, 64,
68, 72, 76, 80, 84, 88, 92, 96], 0.41: [57, 61, 65, 69, 73, 77, 81, 85, 89, 93,
97]})

```

-----  
Frecuencia de iteracciones para que se repita la semilla

-----  
Frecuencia de iteracciones: 54

## 2. Con un valor de 890

```

[35]: numero = int(input("Ingrese Xo: "))
print("Semilla:",numero)

digito=int(input("Ingrese # digitos: "))
print("digito: ",digito)

iteraciones = int(input("Ingrese # de iteraciones: "))
print("iteraciones:",iteraciones)

xn=[]
ui=[]
multiplicacion=[]
rn=[]
def centros(mul):
    cortarI=int(digito/2)
    cortarD=digito-cortarI
    mitad=math.floor(len(mul)/2)
    unir=''
    for i in range(mitad-cortarI, mitad+cortarD, 1):
        unir=unir+mul[i]

```

```

    ui.append(unir)
    return unir

def cuadrado(num):

    multi=(num*num)
    m=str(multi)
    lon=len(m)
    if(len(m)%2!=0):
        if (lon < len(m)+1):
            m=str(m).zfill(len(m)+1)
    multiplicacion.append(m)
    return m

def dividido(n):
    ceros=[int(str(num).ljust(digito+1, "0")) for num in [1]]
    res=n/ceros[0]
    rn.append(res)
    return res

for i in range(iteraciones):
    m=str(cuadrado(int(numero)))
    if(len(m)-1>digito and int(numero)>0):
        xn.append(numero)
        dividido(int(centros(m)))
        numero=ui[-1]
    else:
        print('-Datos Erroneos')
        break

df=pd.DataFrame({"Semilla Xn":xn, "Xn x Xn":multiplicacion, "UI ":ui, "RN":rn})
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)
print(df)

```

Ingrese Xo: 890

Semilla: 890

Ingrese # digitos: 3

digito: 3

Ingrese # de iteraciones: 50

iteraciones: 50

	Semilla	Xn	Xn x Xn	UI	RN
0	890	792100	210	0.21	
1	210	044100	410	0.41	
2	410	168100	810	0.81	
3	810	656100	610	0.61	

4	610	372100	210	0.21
5	210	044100	410	0.41
6	410	168100	810	0.81
7	810	656100	610	0.61
8	610	372100	210	0.21
9	210	044100	410	0.41
10	410	168100	810	0.81
11	810	656100	610	0.61
12	610	372100	210	0.21
13	210	044100	410	0.41
14	410	168100	810	0.81
15	810	656100	610	0.61
16	610	372100	210	0.21
17	210	044100	410	0.41
18	410	168100	810	0.81
19	810	656100	610	0.61
20	610	372100	210	0.21
21	210	044100	410	0.41
22	410	168100	810	0.81
23	810	656100	610	0.61
24	610	372100	210	0.21
25	210	044100	410	0.41
26	410	168100	810	0.81
27	810	656100	610	0.61
28	610	372100	210	0.21
29	210	044100	410	0.41
30	410	168100	810	0.81
31	810	656100	610	0.61
32	610	372100	210	0.21
33	210	044100	410	0.41
34	410	168100	810	0.81
35	810	656100	610	0.61
36	610	372100	210	0.21
37	210	044100	410	0.41
38	410	168100	810	0.81
39	810	656100	610	0.61
40	610	372100	210	0.21
41	210	044100	410	0.41
42	410	168100	810	0.81
43	810	656100	610	0.61
44	610	372100	210	0.21
45	210	044100	410	0.41
46	410	168100	810	0.81
47	810	656100	610	0.61
48	610	372100	210	0.21
49	210	044100	410	0.41

```
[38]: counts_por_elem = Counter(rn)

indices_por_elem = defaultdict(list)
indices = []

for indice, elem in enumerate(rn):
    indices.append(indice)
    indices_por_elem[elem].append(indice)
print("-----")
print("RESULTADOS: ")
print(indices_por_elem)
print("-----")
print("Frecuencia de iteracciones para que se repita la semilla")
print("-----")
print("Frecuencia de iteracciones: ",indices[4])
```

-----

RESULTADOS:

```
defaultdict(<class 'list'>, {0.41: [0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44,
48], 0.81: [1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49], 0.61: [2, 6, 10,
14, 18, 22, 26, 30, 34, 38, 42, 46], 0.21: [3, 7, 11, 15, 19, 23, 27, 31, 35,
39, 43, 47]})
```

-----

Frecuencia de iteracciones para que se repita la semilla

-----

Frecuencia de iteracciones: 4

### 3. Con un valor de 1205

```
[2]: from collections import Counter
from collections import defaultdict
import random
import numpy as np
import pandas as pd
import math

numero = int(input("Ingreso Xo: "))
print("Semilla:",numero)

digito=int(input("Ingreso # digitos: "))
print("digito: ",digito)

iteracciones = int(input("Ingreso # de iteracciones: "))
print("iteracciones:",iteracciones)

xn=[]
ui=[]
multiplicacion=[]
```



```

rn=[]
def centros(mul):
    cortarI=int(digito/2)
    cortarD=digito-cortarI
    mitad=math.floor(len(mul)/2)
    unir=''
    for i in range(mitad-cortarI, mitad+cortarD, 1):
        unir=unir+mul[i]
    ui.append(unir)
    return unir

def cuadrado(num):

    multi=(num*num)
    m=str(multi)
    lon=len(m)
    if(len(m)%2!=0):
        if (lon < len(m)+1):
            m=str(m).zfill(len(m)+1)
    multiplicacion.append(m)
    return m

def dividido(n):
    ceros=[int(str(num).ljust(digito+1, "0")) for num in [1]]
    res=n/ceros[0]
    rn.append(res)
    return res

for i in range(iteraciones):
    m=str(cuadrado(int(numero)))
    if(len(m)-1>digito and int(numero)>0):
        xn.append(numero)
        dividido(int(centros(m)))
        numero=ui[-1]
    else:
        print('-Datos Erroneos')
        break

df=pd.DataFrame({"Semilla Xn":xn, "Xn x Xn":multiplicacion, "UI ":ui, "RN":rn})
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)
print(df)

```

Ingrese Xo: 1205

Semilla: 1205

Ingrese # digitos: 4

digito: 4

Ingrese # de iteraciones: 50

iteraciones: 50

	Semilla	Xn	Xn x Xn	UI	RN
0		1205	01452025	4520	0.4520
1		4520	20430400	4304	0.4304
2		4304	18524416	5244	0.5244
3		5244	27499536	4995	0.4995
4		4995	24950025	9500	0.9500
5		9500	90250000	2500	0.2500
6		2500	06250000	2500	0.2500
7		2500	06250000	2500	0.2500
8		2500	06250000	2500	0.2500
9		2500	06250000	2500	0.2500
10		2500	06250000	2500	0.2500
11		2500	06250000	2500	0.2500
12		2500	06250000	2500	0.2500
13		2500	06250000	2500	0.2500
14		2500	06250000	2500	0.2500
15		2500	06250000	2500	0.2500
16		2500	06250000	2500	0.2500
17		2500	06250000	2500	0.2500
18		2500	06250000	2500	0.2500
19		2500	06250000	2500	0.2500
20		2500	06250000	2500	0.2500
21		2500	06250000	2500	0.2500
22		2500	06250000	2500	0.2500
23		2500	06250000	2500	0.2500
24		2500	06250000	2500	0.2500
25		2500	06250000	2500	0.2500
26		2500	06250000	2500	0.2500
27		2500	06250000	2500	0.2500
28		2500	06250000	2500	0.2500
29		2500	06250000	2500	0.2500
30		2500	06250000	2500	0.2500
31		2500	06250000	2500	0.2500
32		2500	06250000	2500	0.2500
33		2500	06250000	2500	0.2500
34		2500	06250000	2500	0.2500
35		2500	06250000	2500	0.2500
36		2500	06250000	2500	0.2500
37		2500	06250000	2500	0.2500
38		2500	06250000	2500	0.2500
39		2500	06250000	2500	0.2500
40		2500	06250000	2500	0.2500
41		2500	06250000	2500	0.2500
42		2500	06250000	2500	0.2500
43		2500	06250000	2500	0.2500

```

44      2500  06250000  2500  0.2500
45      2500  06250000  2500  0.2500
46      2500  06250000  2500  0.2500
47      2500  06250000  2500  0.2500
48      2500  06250000  2500  0.2500
49      2500  06250000  2500  0.2500

```

```

[6]: counts_por_elem = Counter(rn)

indices_por_elem = defaultdict(list)
indices = []

for indice, elem in enumerate(rn):
    indices.append(indice)
    indices_por_elem[elem].append(indice)
print("-----")
print("RESULTADOS: ")
print(indices_por_elem)
print("-----")
print("Frecuencia de iteracciones para que se repita la semilla")
print("-----")
print("Frecuencia de iteracciones: ",indices[5])

```

-----  
RESULTADOS:

```

defaultdict(<class 'list'>, {0.452: [0], 0.4304: [1], 0.5244: [2], 0.4995: [3],
0.95: [4], 0.25: [5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
42, 43, 44, 45, 46, 47, 48, 49]})

```

-----  
Frecuencia de iteracciones para que se repita la semilla

-----  
Frecuencia de iteracciones: 5

## 0.1 CON LA LIBRERIA psutil OBTENEMOS ALGUNAS PROPIEDADES DE LA PC

### 4. MEMORIA

```

[5]: mem = psutil.virtual_memory()
memoria=mem.total
memoria

```

```

[5]: 17015255040

```

```

[6]: from collections import Counter
from collections import defaultdict
import random
import numpy as np

```

```

import pandas as pd
import math

numero = memoria
print("Semilla:", numero)

digito=int(input("Ingrese # digitos: "))
print("digito: ", digito)

iteraciones = int(input("Ingrese # de iteraciones: "))
print("iteraciones:", iteraciones)

xn=[]
ui=[]
multiplicacion=[]
rn=[]
def centros(mul):
    cortarI=int(digito/2)
    cortarD=digito-cortarI
    mitad=math.floor(len(mul)/2)
    unir=''
    for i in range(mitad-cortarI, mitad+cortarD, 1):
        unir=unir+mul[i]
    ui.append(unir)
    return unir

def cuadrado(num):

    multi=(num*num)
    m=str(multi)
    lon=len(m)
    if(len(m)%2!=0):
        if (lon < len(m)+1):
            m=str(m).zfill(len(m)+1)
    multiplicacion.append(m)
    return m

def dividido(n):
    ceros=[int(str(num).ljust(digito+1, "0")) for num in [1]]
    res=n/ceros[0]
    rn.append(res)
    return res

for i in range(iteraciones):
    m=str(cuadrado(int(numero)))
    if(len(m)-1>digito and int(numero)>0):
        xn.append(numero)

```

```

        dividido(int(centros(m)))
        numero=ui[-1]
    else:
        print('-Datos Erroneos')
        break

df=pd.DataFrame({"Semilla Xn":xn, "Xn x Xn":multiplicacion , "UI ":ui, "RN":rn})
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)
print(df)

```

Semilla: 17015255040

Ingrese # digitos: 4

digito: 4

Ingrese # de iteraciones: 56

iteraciones: 56

	Semilla Xn	Xn x Xn	UI	RN
0	17015255040	0289518904076245401600	4076	0.4076
1	4076	16613776	6137	0.6137
2	6137	37662769	6627	0.6627
3	6627	43917129	9171	0.9171
4	9171	84107241	1072	0.1072
5	1072	01149184	1491	0.1491
6	1491	02223081	2230	0.2230
7	2230	04972900	9729	0.9729
8	9729	94653441	6534	0.6534
9	6534	42693156	6931	0.6931
10	6931	48038761	0387	0.0387
11	0387	149769	4976	0.4976
12	4976	24760576	7605	0.7605
13	7605	57836025	8360	0.8360
14	8360	69889600	8896	0.8896
15	8896	79138816	1388	0.1388
16	1388	01926544	9265	0.9265
17	9265	85840225	8402	0.8402
18	8402	70593604	5936	0.5936
19	5936	35236096	2360	0.2360
20	2360	05569600	5696	0.5696
21	5696	32444416	4444	0.4444
22	4444	19749136	7491	0.7491
23	7491	56115081	1150	0.1150
24	1150	01322500	3225	0.3225
25	3225	10400625	4006	0.4006
26	4006	16048036	0480	0.0480
27	0480	230400	3040	0.3040
28	3040	09241600	2416	0.2416

29	2416	05837056	8370	0.8370
30	8370	70056900	0569	0.0569
31	0569	323761	2376	0.2376
32	2376	05645376	6453	0.6453
33	6453	41641209	6412	0.6412
34	6412	41113744	1137	0.1137
35	1137	01292769	2927	0.2927
36	2927	08567329	5673	0.5673
37	5673	32182929	1829	0.1829
38	1829	03345241	3452	0.3452
39	3452	11916304	9163	0.9163
40	9163	83960569	9605	0.9605
41	9605	92256025	2560	0.2560
42	2560	06553600	5536	0.5536
43	5536	30647296	6472	0.6472
44	6472	41886784	8867	0.8867
45	8867	78623689	6236	0.6236
46	6236	38887696	8876	0.8876
47	8876	78783376	7833	0.7833
48	7833	61355889	3558	0.3558
49	3558	12659364	6593	0.6593
50	6593	43467649	4676	0.4676
51	4676	21864976	8649	0.8649
52	8649	74805201	8052	0.8052
53	8052	64834704	8347	0.8347
54	8347	69672409	6724	0.6724
55	6724	45212176	2121	0.2121

## 5. FRECUENCIA

```
[17]: frecuencia = psutil.cpu_freq()
frecuencia = int(frecuencia.current)
frecuencia
```

[17]: 2900

```
[18]: numero = frecuencia
print("Semilla:", numero)

digito=int(input("Ingrese # digitos: "))
print("digito: ", digito)

iteraciones = int(input("Ingrese # de iteraciones: "))
print("iteraciones:", iteraciones)

xn=[]
ui=[]
multiplicacion=[]
```

```

rn=[]
def centros(mul):
    cortarI=int(digito/2)
    cortarD=digito-cortarI
    mitad=math.floor(len(mul)/2)
    unir=''
    for i in range(mitad-cortarI, mitad+cortarD, 1):
        unir=unir+mul[i]
    ui.append(unir)
    return unir

def cuadrado(num):

    multi=(num*num)
    m=str(multi)
    lon=len(m)
    if(len(m)%2!=0):
        if (lon < len(m)+1):
            m=str(m).zfill(len(m)+1)
    multiplicacion.append(m)
    return m

def dividido(n):
    ceros=[int(str(num).ljust(digito+1, "0")) for num in [1]]
    res=n/ceros[0]
    rn.append(res)
    return res

for i in range(iteraciones):
    m=str(cuadrado(int(numero)))
    if(len(m)-1>digito and int(numero)>0):
        xn.append(numero)
        dividido(int(centros(m)))
        numero=ui[-1]
    else:
        print('-Datos Erroneos')
        break

df=pd.DataFrame({"Semilla Xn":xn, "Xn x Xn":multiplicacion, "UI ":ui, "RN":rn})
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)
print(df)

```

Semilla: 2900  
Ingrese # digitos: 4  
digito: 4

Ingrese # de iteraciones: 50  
iteraciones: 50

	Semilla	Xn	Xn x Xn	UI	RN
0		2900	08410000	4100	0.41
1		4100	16810000	8100	0.81
2		8100	65610000	6100	0.61
3		6100	37210000	2100	0.21
4		2100	04410000	4100	0.41
5		4100	16810000	8100	0.81
6		8100	65610000	6100	0.61
7		6100	37210000	2100	0.21
8		2100	04410000	4100	0.41
9		4100	16810000	8100	0.81
10		8100	65610000	6100	0.61
11		6100	37210000	2100	0.21
12		2100	04410000	4100	0.41
13		4100	16810000	8100	0.81
14		8100	65610000	6100	0.61
15		6100	37210000	2100	0.21
16		2100	04410000	4100	0.41
17		4100	16810000	8100	0.81
18		8100	65610000	6100	0.61
19		6100	37210000	2100	0.21
20		2100	04410000	4100	0.41
21		4100	16810000	8100	0.81
22		8100	65610000	6100	0.61
23		6100	37210000	2100	0.21
24		2100	04410000	4100	0.41
25		4100	16810000	8100	0.81
26		8100	65610000	6100	0.61
27		6100	37210000	2100	0.21
28		2100	04410000	4100	0.41
29		4100	16810000	8100	0.81
30		8100	65610000	6100	0.61
31		6100	37210000	2100	0.21
32		2100	04410000	4100	0.41
33		4100	16810000	8100	0.81
34		8100	65610000	6100	0.61
35		6100	37210000	2100	0.21
36		2100	04410000	4100	0.41
37		4100	16810000	8100	0.81
38		8100	65610000	6100	0.61
39		6100	37210000	2100	0.21
40		2100	04410000	4100	0.41
41		4100	16810000	8100	0.81
42		8100	65610000	6100	0.61
43		6100	37210000	2100	0.21
44		2100	04410000	4100	0.41



```

45      4100  16810000  8100  0.81
46      8100  65610000  6100  0.61
47      6100  37210000  2100  0.21
48      2100  04410000  4100  0.41
49      4100  16810000  8100  0.81

```

```

[30]: counts_por_elem = Counter(rn)

indices_por_elem = defaultdict(list)
indices = []

for indice, elem in enumerate(rn):
    if counts_por_elem[elem] > 1:
        indices.append(indice)
        indices_por_elem[elem].append(indice)

print("-----")
print("RESULTADOS: ")
print(indices_por_elem)
print("-----")
print("Frecuencia de iteracciones para que se repita la semilla")
print("-----")
print("Frecuencia de iteracciones: ",indices[4])

```

-----  
RESULTADOS:

```

defaultdict(<class 'list'>, {0.41: [0, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44,
48], 0.81: [1, 5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45, 49], 0.61: [2, 6, 10,
14, 18, 22, 26, 30, 34, 38, 42, 46], 0.21: [3, 7, 11, 15, 19, 23, 27, 31, 35,
39, 43, 47]})

```

-----  
Frecuencia de iteracciones para que se repita la semilla

-----  
Frecuencia de iteracciones: 4

## 6. CAPACIDAD DE DISCO LOCAL D

```

[3]: discod = psutil.disk_usage('/')
discod = int(discod.total/10000000)
discod

```

[3]: 29602

```

[4]: from collections import Counter
from collections import defaultdict
import random
import psutil
import numpy as np
import pandas as pd
import math

```

```

numero = discod
print("Semilla:", numero)

digito=int(input("Ingrese # digitos: "))
print("digito: ", digito)

iteraciones = int(input("Ingrese # de iteraciones: "))
print("iteraciones:", iteraciones)

xn=[]
ui=[]
multiplicacion=[]
rn=[]
def centros(mul):
    cortarI=int(digito/2)
    cortarD=digito-cortarI
    mitad=math.floor(len(mul)/2)
    unir=''
    for i in range(mitad-cortarI, mitad+cortarD, 1):
        unir=unir+mul[i]
    ui.append(unir)
    return unir

def cuadrado(num):

    multi=(num*num)
    m=str(multi)
    lon=len(m)
    if(len(m)%2!=0):
        if (lon < len(m)+1):
            m=str(m).zfill(len(m)+1)
    multiplicacion.append(m)
    return m

def dividido(n):
    ceros=[int(str(num).ljust(digito+1, "0")) for num in [1]]
    res=n/ceros[0]
    rn.append(res)
    return res

for i in range(iteraciones):
    m=str(cuadrado(int(numero)))
    if(len(m)-1>digito and int(numero)>0):
        xn.append(numero)
        dividido(int(centros(m)))
        numero=ui[-1]

```

```

else:
    print('-Datos Erroneos')
    break

df=pd.DataFrame({"Semilla Xn":xn, "Xn x Xn":multiplicacion ,"UI ":ui, "RN":rn})
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)
print(df)

```

Semilla: 29602

Ingrese # digitos: 4

digito: 4

Ingrese # de iteraciones: 40

iteraciones: 40

	Semilla Xn	Xn x Xn	UI	RN
0	29602	0876278404	6278	0.6278
1	6278	39413284	4132	0.4132
2	4132	17073424	0734	0.0734
3	0734	538756	3875	0.3875
4	3875	15015625	0156	0.0156
5	0156	024336	2433	0.2433
6	2433	05919489	9194	0.9194
7	9194	84529636	5296	0.5296
8	5296	28047616	0476	0.0476
9	0476	226576	2657	0.2657
10	2657	07059649	0596	0.0596
11	0596	355216	5521	0.5521
12	5521	30481441	4814	0.4814
13	4814	23174596	1745	0.1745
14	1745	03045025	0450	0.0450
15	0450	202500	0250	0.0250
16	0250	062500	6250	0.6250
17	6250	39062500	0625	0.0625
18	0625	390625	9062	0.9062
19	9062	82119844	1198	0.1198
20	1198	01435204	4352	0.4352
21	4352	18939904	9399	0.9399
22	9399	88341201	3412	0.3412
23	3412	11641744	6417	0.6417
24	6417	41177889	1778	0.1778
25	1778	03161284	1612	0.1612
26	1612	02598544	5985	0.5985
27	5985	35820225	8202	0.8202
28	8202	67272804	2728	0.2728
29	2728	07441984	4419	0.4419
30	4419	19527561	5275	0.5275

31	5275	27825625	8256	0.8256
32	8256	68161536	1615	0.1615
33	1615	02608225	6082	0.6082
34	6082	36990724	9907	0.9907
35	9907	98148649	1486	0.1486
36	1486	02208196	2081	0.2081
37	2081	04330561	3305	0.3305
38	3305	10923025	9230	0.9230
39	9230	85192900	1929	0.1929

```
[9]: counts_por_elem = Counter(rn)

indices_por_elem = defaultdict(list)
indices = []

for indice, elem in enumerate(rn):
    if counts_por_elem[elem] > 1:
        indices.append(indice)
        indices_por_elem[elem].append(indice)

print("-----")
print("RESULTADOS: ")
print(indices_por_elem)
print("-----")
print("Frecuencia de iteracciones para que se repita la semilla")
print("-----")
print("Frecuencia de iteraciones: ", indices[0])
```

```
-----
RESULTADOS:
defaultdict(<class 'list'>, {0.776: [28, 32, 36, 40, 44, 48], 0.176: [29, 33,
37, 41, 45, 49], 0.976: [30, 34, 38, 42, 46], 0.576: [31, 35, 39, 43, 47]})

-----
Frecuencia de iteracciones para que se repita la semilla

-----
Frecuencia de iteraciones:  28
```

## 7. Número de lecturas del disco

```
[7]: numero_write=psutil.disk_io_counters()
numero_write=numero_write.write_count
numero_write
```

```
[7]: 2597710
```

```
[12]: from collections import Counter
from collections import defaultdict
import random
import psutil
import numpy as np
```

```

import pandas as pd
import math

numero = numero_write
print("Semilla:", numero)

digito=int(input("Ingrese # digitos: "))
print("digito: ", digito)

iteraciones = int(input("Ingrese # de iteraciones: "))
print("iteraciones:", iteraciones)

xn=[]
ui=[]
multiplicacion=[]
rn=[]
def centros(mul):
    cortarI=int(digito/2)
    cortarD=digito-cortarI
    mitad=math.floor(len(mul)/2)
    unir=''
    for i in range(mitad-cortarI, mitad+cortarD, 1):
        unir=unir+mul[i]
    ui.append(unir)
    return unir

def cuadrado(num):

    multi=(num*num)
    m=str(multi)
    lon=len(m)
    if(len(m)%2!=0):
        if (lon < len(m)+1):
            m=str(m).zfill(len(m)+1)
    multiplicacion.append(m)
    return m

def dividido(n):
    ceros=[int(str(num).ljust(digito+1, "0")) for num in [1]]
    res=n/ceros[0]
    rn.append(res)
    return res

for i in range(iteraciones):
    m=str(cuadrado(int(numero)))
    if(len(m)-1>digito and int(numero)>0):
        xn.append(numero)

```

```

        dividido(int(centros(m)))
        numero=ui[-1]
    else:
        print('-Datos Erroneos')
        break

df=pd.DataFrame({"Semilla Xn":xn, "Xn x Xn":multiplicacion , "UI ":ui, "RN":rn})
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)
print(df)

```

Semilla: 2597710

Ingrese # digitos: 3

digito: 3

Ingrese # de iteraciones: 45

iteraciones: 45

	Semilla Xn	Xn x Xn	UI	RN
0	2597710	06748097244100	972	0.972
1	972	944784	478	0.478
2	478	228484	848	0.848
3	848	719104	910	0.910
4	910	828100	810	0.810
5	810	656100	610	0.610
6	610	372100	210	0.210
7	210	044100	410	0.410
8	410	168100	810	0.810
9	810	656100	610	0.610
10	610	372100	210	0.210
11	210	044100	410	0.410
12	410	168100	810	0.810
13	810	656100	610	0.610
14	610	372100	210	0.210
15	210	044100	410	0.410
16	410	168100	810	0.810
17	810	656100	610	0.610
18	610	372100	210	0.210
19	210	044100	410	0.410
20	410	168100	810	0.810
21	810	656100	610	0.610
22	610	372100	210	0.210
23	210	044100	410	0.410
24	410	168100	810	0.810
25	810	656100	610	0.610
26	610	372100	210	0.210
27	210	044100	410	0.410
28	410	168100	810	0.810

29	810	656100	610	0.610
30	610	372100	210	0.210
31	210	044100	410	0.410
32	410	168100	810	0.810
33	810	656100	610	0.610
34	610	372100	210	0.210
35	210	044100	410	0.410
36	410	168100	810	0.810
37	810	656100	610	0.610
38	610	372100	210	0.210
39	210	044100	410	0.410
40	410	168100	810	0.810
41	810	656100	610	0.610
42	610	372100	210	0.210
43	210	044100	410	0.410
44	410	168100	810	0.810

```
[14]: counts_por_elem = Counter(rn)

indices_por_elem = defaultdict(list)
indices = []

for indice, elem in enumerate(rn):
    if counts_por_elem[elem] > 1:
        indices.append(indice)
        indices_por_elem[elem].append(indice)

print("-----")
print("RESULTADOS: ")
print(indices_por_elem)
print("-----")
print("Frecuencia de iteracciones para que se repita la semilla")
print("-----")
print("Frecuencia de iteraciones: ", indices[0])
```

```
-----
RESULTADOS:
defaultdict(<class 'list'>, {0.81: [4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44],
0.61: [5, 9, 13, 17, 21, 25, 29, 33, 37, 41], 0.21: [6, 10, 14, 18, 22, 26, 30,
34, 38, 42], 0.41: [7, 11, 15, 19, 23, 27, 31, 35, 39, 43]})

-----
Frecuencia de iteracciones para que se repita la semilla

-----
Frecuencia de iteraciones: 4
```

## 8. Tiempo dedicado a escribir en el disco

```
[2]: time_write=psutil.disk_io_counters()
time_write=time_write.write_time
time_write
```

[2]: 1225

```
[10]: from collections import Counter
from collections import defaultdict
import random
import psutil
import numpy as np
import pandas as pd
import math

numero = time_write
print("Semilla:", numero)

digito=int(input("Ingrese # digitos: "))
print("digito: ", digito)

iteraciones = int(input("Ingrese # de iteraciones: "))
print("iteraciones:", iteraciones)

xn=[]
ui=[]
multiplicacion=[]
rn=[]
def centros(mul):
    cortarI=int(digito/2)
    cortarD=digito-cortarI
    mitad=math.floor(len(mul)/2)
    unir=''
    for i in range(mitad-cortarI, mitad+cortarD, 1):
        unir=unir+mul[i]
    ui.append(unir)
    return unir

def cuadrado(num):

    multi=(num*num)
    m=str(multi)
    lon=len(m)
    if(len(m)%2!=0):
        if (lon < len(m)+1):
            m=str(m).zfill(len(m)+1)
    multiplicacion.append(m)
    return m

def dividido(n):
    ceros=[int(str(num).ljust(digito+1, "0")) for num in [1]]
    res=n/ceros[0]
```



```

    rn.append(res)
    return res

for i in range(iteraciones):
    m=str(cuadrado(int(numero)))
    if(len(m)-1>digito and int(numero)>0):
        xn.append(numero)
        dividido(int(centros(m)))
        numero=ui[-1]
    else:
        print('-Datos Erroneos')
        break

df=pd.DataFrame({"Semilla Xn":xn, "Xn x Xn":multiplicacion ,"UI ":ui, "RN":rn})
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)
print(df)

```

```

Semilla: 1225
Ingrese # digitos: 4
digito: 4
Ingrese # de iteraciones: 4
iteraciones: 4

```

	Semilla Xn	Xn x Xn	UI	RN
0	1225	01500625	5006	0.5006
1	5006	25060036	0600	0.0600
2	0600	360000	6000	0.6000
3	6000	36000000	0000	0.0000

```

[11]: counts_por_elem = Counter(rn)

indices_por_elem = defaultdict(list)
indices = []

for indice, elem in enumerate(rn):
    if counts_por_elem[elem] > 1:
        indices.append(indice)
        indices_por_elem[elem].append(indice)

print("-----")
print("RESULTADOS: ")
print(indices_por_elem)
print("-----")
print("Frecuencia de iteraciones para que se repita la semilla")
print("-----")
print("Frecuencia de iteraciones: ",indices)

```

-----  
RESULTADOS:

defaultdict(<class 'list'>, {})

-----  
Frecuencia de iteracciones para que se repita la semilla

-----  
Frecuencia de iteracciones: []

### 9.Numero de bytes recibidos

```
[13]: bytes_rec = psutil.net_io_counters()  
      bytes_rec= bytes_rec.bytes_recv  
      bytes_rec
```

[13]: 1190209435

```
[14]: from collections import Counter  
      from collections import defaultdict  
      import random  
      import psutil  
      import numpy as np  
      import pandas as pd  
      import math  
  
      numero = bytes_rec  
      print("Semilla:", numero)  
  
      digito=int(input("Ingrese # digitos: "))  
      print("digito: ", digito)  
  
      iteracciones = int(input("Ingrese # de iteracciones: "))  
      print("iteracciones:", iteracciones)  
  
      xn=[]  
      ui=[]  
      multiplicacion=[]  
      rn=[]  
      def centros(mul):  
          cortarI=int(digito/2)  
          cortarD=digito-cortarI  
          mitad=math.floor(len(mul)/2)  
          unir=''  
          for i in range(mitad-cortarI, mitad+cortarD, 1):  
              unir=unir+mul[i]  
          ui.append(unir)  
          return unir  
  
      def cuadrado(num):
```

```

multi=(num*num)
m=str(multi)
lon=len(m)
if(len(m)%2!=0):
    if (lon < len(m)+1):
        m=str(m).zfill(len(m)+1)
multiplicacion.append(m)
return m

def dividido(n):
    ceros=[int(str(num).ljust(digito+1, "0")) for num in [1]]
    res=n/ceros[0]
    rn.append(res)
    return res

for i in range(iteraciones):
    m=str(cuadrado(int(numero)))
    if(len(m)-1>digito and int(numero)>0):
        xn.append(numero)
        dividido(int(centros(m)))
        numero=ui[-1]
    else:
        print('-Datos Erroneos')
        break

df=pd.DataFrame({"Semilla Xn":xn, "Xn x Xn":multiplicacion, "UI ":ui, "RN":rn})
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)
print(df)

```

Semilla: 1190209435  
 Ingrese # digitos: 4  
 digito: 4  
 Ingrese # de iteraciones: 35  
 iteraciones: 35

	Semilla Xn	Xn x Xn	UI	RN
0	1190209435	01416598499163019225	4991	0.4991
1	4991	24910081	9100	0.9100
2	9100	82810000	8100	0.8100
3	8100	65610000	6100	0.6100
4	6100	37210000	2100	0.2100
5	2100	04410000	4100	0.4100
6	4100	16810000	8100	0.8100
7	8100	65610000	6100	0.6100
8	6100	37210000	2100	0.2100

9	2100	04410000	4100	0.4100
10	4100	16810000	8100	0.8100
11	8100	65610000	6100	0.6100
12	6100	37210000	2100	0.2100
13	2100	04410000	4100	0.4100
14	4100	16810000	8100	0.8100
15	8100	65610000	6100	0.6100
16	6100	37210000	2100	0.2100
17	2100	04410000	4100	0.4100
18	4100	16810000	8100	0.8100
19	8100	65610000	6100	0.6100
20	6100	37210000	2100	0.2100
21	2100	04410000	4100	0.4100
22	4100	16810000	8100	0.8100
23	8100	65610000	6100	0.6100
24	6100	37210000	2100	0.2100
25	2100	04410000	4100	0.4100
26	4100	16810000	8100	0.8100
27	8100	65610000	6100	0.6100
28	6100	37210000	2100	0.2100
29	2100	04410000	4100	0.4100
30	4100	16810000	8100	0.8100
31	8100	65610000	6100	0.6100
32	6100	37210000	2100	0.2100
33	2100	04410000	4100	0.4100
34	4100	16810000	8100	0.8100

```
[16]: counts_por_elem = Counter(rn)

indices_por_elem = defaultdict(list)
indices = []

for indice, elem in enumerate(rn):
    if counts_por_elem[elem] > 1:
        indices.append(indice)
        indices_por_elem[elem].append(indice)

print("-----")
print("RESULTADOS: ")
print(indices_por_elem)
print("-----")
print("Frecuencia de iteracciones para que se repita la semilla")
print("-----")
print("Frecuencia de iteracciones: ", indices[0])
```

```
-----
RESULTADOS:
defaultdict(<class 'list'>, {0.81: [2, 6, 10, 14, 18, 22, 26, 30, 34], 0.61: [3,
7, 11, 15, 19, 23, 27, 31], 0.21: [4, 8, 12, 16, 20, 24, 28, 32], 0.41: [5, 9,
```

```
13, 17, 21, 25, 29, 33]})
```

```
-----  
Frecuencia de iteraciones para que se repita la semilla
```

```
-----  
Frecuencia de iteraciones: 2
```

## 10. Numero de paquetes recibidos

```
[2]: packets_rec = psutil.net_io_counters()  
     packets_rec= packets_rec.bytes_sent  
     packets_rec
```

```
[2]: 69868347
```

```
[3]: from collections import Counter  
     from collections import defaultdict  
     import random  
     import psutil  
     import numpy as np  
     import pandas as pd  
     import math  
  
     numero = packets_rec  
     print("Semilla:", numero)  
  
     digito=int(input("Ingrese # digitos: "))  
     print("digito: ", digito)  
  
     iteraciones = int(input("Ingrese # de iteraciones: "))  
     print("iteraciones:", iteraciones)  
  
     xn=[]  
     ui=[]  
     multiplicacion=[]  
     rn=[]  
     def centros(mul):  
         cortarI=int(digito/2)  
         cortarD=digito-cortarI  
         mitad=math.floor(len(mul)/2)  
         unir=''  
         for i in range(mitad-cortarI, mitad+cortarD, 1):  
             unir=unir+mul[i]  
         ui.append(unir)  
         return unir  
  
     def cuadrado(num):  
  
         multi=(num*num)
```

```

m=str(multi)
lon=len(m)
if(len(m)%2!=0):
    if (lon < len(m)+1):
        m=str(m).zfill(len(m)+1)
multiplicacion.append(m)
return m

def dividido(n):
    ceros=[int(str(num).ljust(digito+1, "0")) for num in [1]]
    res=n/ceros[0]
    rn.append(res)
    return res

for i in range(iteraciones):
    m=str(cuadrado(int(numero)))
    if(len(m)-1>digito and int(numero)>0):
        xn.append(numero)
        dividido(int(centros(m)))
        numero=ui[-1]
    else:
        print('-Datos Erroneos')
        break

df=pd.DataFrame({"Semilla Xn":xn, "Xn x Xn":multiplicacion, "UI ":ui, "RN":rn})
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
pd.set_option('display.width', None)
pd.set_option('display.max_colwidth', None)
print(df)

```

Semilla: 69868347

Ingrese # digitos: 4

digito: 4

Ingrese # de iteraciones: 35

iteraciones: 35

	Semilla Xn	Xn x Xn	UI	RN
0	69868347	4881585912512409	5912	0.5912
1	5912	34951744	9517	0.9517
2	9517	90573289	5732	0.5732
3	5732	32855824	8558	0.8558
4	8558	73239364	2393	0.2393
5	2393	05726449	7264	0.7264
6	7264	52765696	7656	0.7656
7	7656	58614336	6143	0.6143
8	6143	37736449	7364	0.7364
9	7364	54228496	2284	0.2284
10	2284	05216656	2166	0.2166

11	2166	04691556	6915	0.6915
12	6915	47817225	8172	0.8172
13	8172	66781584	7815	0.7815
14	7815	61074225	0742	0.0742
15	0742	550564	5056	0.5056
16	5056	25563136	5631	0.5631
17	5631	31708161	7081	0.7081
18	7081	50140561	1405	0.1405
19	1405	01974025	9740	0.9740
20	9740	94867600	8676	0.8676
21	8676	75272976	2729	0.2729
22	2729	07447441	4474	0.4474
23	4474	20016676	0166	0.0166
24	0166	027556	2755	0.2755
25	2755	07590025	5900	0.5900
26	5900	34810000	8100	0.8100
27	8100	65610000	6100	0.6100
28	6100	37210000	2100	0.2100
29	2100	04410000	4100	0.4100
30	4100	16810000	8100	0.8100
31	8100	65610000	6100	0.6100
32	6100	37210000	2100	0.2100
33	2100	04410000	4100	0.4100
34	4100	16810000	8100	0.8100

```
[4]: counts_por_elem = Counter(rn)

indices_por_elem = defaultdict(list)
indices = []

for indice, elem in enumerate(rn):
    if counts_por_elem[elem] > 1:
        indices.append(indice)
        indices_por_elem[elem].append(indice)

print("-----")
print("RESULTADOS: ")
print(indices_por_elem)
print("-----")
print("Frecuencia de iteracciones para que se repita la semilla")
print("-----")
print("Frecuencia de iteracciones: ", indices[0])
```

-----  
RESULTADOS:

defaultdict(<class 'list'>, {0.81: [26, 30, 34], 0.61: [27, 31], 0.21: [28, 32], 0.41: [29, 33]})

-----  
Frecuencia de iteracciones para que se repita la semilla

-----  
Frecuencia de iteraciones: 26