# Universidad Politecnica Salesiana

## Nombre: Jessica Ñauta.

## Asigantura: Simulación.

# Método Cuadrado Medio

In [1]:
```
from collections import Counter
from collections import defaultdict
import random
import psutil
import numpy as np
import pandas as pd
import math
import collections
import matplotlib.pyplot as plt
```

In [2]:
```
valores =[2317, 9823, 1639, 4820, 3792]
arreglorn=[]
def get_pos(digs):
    val1 =0
    val2 =0
    if digs%2 !=0:
        val1 = int(digs/2)
        val2 = int(digs/2)+1
    else:
        val1 = int(digs/2)
        val2 = int(digs/2)
    return val1,val2

def calcular_num(iters, val, digs):
    x0_semilla = int(val)
    aum = get_pos(digs)
    print("ITERACIÓN", "Xn", "Xn*Xn", "Longitud","Ui","Rn")
    for i in range(iters):
        xn2= x0_semilla**2
        lon = len(str(xn2))
        ui = str(xn2)[int(lon/2)-aum[0]:int(lon/2)+aum[1]]
        rn = int(ui)/10**digs
        arreglorn.append(rn)
        #df=pd.DataFrame({"Xn":x0_semilla, "Xn*Xn":xn2 ,"Longitud":lon, "UI ":ui, "R
        print(i, " ", x0_semilla," ",xn2, " ", lon, " ",ui, " ", rn)
        x0_semilla=int(ui)
    print("    ")

iters = int(input("Iteraciones: "))
digs = int(input("Ingrese el digito: "))
for i in valores:
    print("*****************************************************************")
    print("i: ", i)
    calcular_num(iters, i, digs)
```

```
Iteraciones: 30
Ingrese el digito: 4
*****************************************************************
i:  2317
ITERACIÓN Xn Xn*Xn Longitud Ui Rn
0   2317   5368489   7   3684   0.3684
1   3684   13571856   8   5718   0.5718
```

```
2   5718   32695524   8   6955   0.6955
3   6955   48372025   8   3720   0.372
4   3720   13838400   8   8384   0.8384
5   8384   70291456   8   2914   0.2914
6   2914   8491396   7   4913   0.4913
7   4913   24137569   8   1375   0.1375
8   1375   1890625   7   8906   0.8906
9   8906   79316836   8   3168   0.3168
10   3168   10036224   8   0362   0.0362
11   362   131044   6   3104   0.3104
12   3104   9634816   7   6348   0.6348
13   6348   40297104   8   2971   0.2971
14   2971   8826841   7   8268   0.8268
15   8268   68359824   8   3598   0.3598
16   3598   12945604   8   9456   0.9456
17   9456   89415936   8   4159   0.4159
18   4159   17297281   8   2972   0.2972
19   2972   8832784   7   8327   0.8327
20   8327   69338929   8   3389   0.3389
21   3389   11485321   8   4853   0.4853
22   4853   23551609   8   5516   0.5516
23   5516   30426256   8   4262   0.4262
24   4262   18164644   8   1646   0.1646
25   1646   2709316   7   7093   0.7093
26   7093   50310649   8   3106   0.3106
27   3106   9647236   7   6472   0.6472
28   6472   41886784   8   8867   0.8867
29   8867   78623689   8   6236   0.6236

************************************************************
i:  9823
ITERACIÓN Xn Xn*Xn Longitud Ui Rn
0   9823   96491329   8   4913   0.4913
1   4913   24137569   8   1375   0.1375
2   1375   1890625   7   8906   0.8906
3   8906   79316836   8   3168   0.3168
4   3168   10036224   8   0362   0.0362
5   362   131044   6   3104   0.3104
6   3104   9634816   7   6348   0.6348
7   6348   40297104   8   2971   0.2971
8   2971   8826841   7   8268   0.8268
9   8268   68359824   8   3598   0.3598
10   3598   12945604   8   9456   0.9456
11   9456   89415936   8   4159   0.4159
12   4159   17297281   8   2972   0.2972
13   2972   8832784   7   8327   0.8327
14   8327   69338929   8   3389   0.3389
15   3389   11485321   8   4853   0.4853
16   4853   23551609   8   5516   0.5516
17   5516   30426256   8   4262   0.4262
18   4262   18164644   8   1646   0.1646
19   1646   2709316   7   7093   0.7093
20   7093   50310649   8   3106   0.3106
21   3106   9647236   7   6472   0.6472
22   6472   41886784   8   8867   0.8867
23   8867   78623689   8   6236   0.6236
24   6236   38887696   8   8876   0.8876
25   8876   78783376   8   7833   0.7833
26   7833   61355889   8   3558   0.3558
27   3558   12659364   8   6593   0.6593
28   6593   43467649   8   4676   0.4676
29   4676   21864976   8   8649   0.8649

************************************************************
i:  1639
ITERACIÓN Xn Xn*Xn Longitud Ui Rn
0   1639   2686321   7   6863   0.6863
1   6863   47100769   8   1007   0.1007
2   1007   1014049   7   0140   0.014
```

```
3   140   19600   5   1960   0.196
4   1960   3841600   7   8416   0.8416
5   8416   70829056   8   8290   0.829
6   8290   68724100   8   7241   0.7241
7   7241   52432081   8   4320   0.432
8   4320   18662400   8   6624   0.6624
9   6624   43877376   8   8773   0.8773
10   8773   76965529   8   9655   0.9655
11   9655   93219025   8   2190   0.219
12   2190   4796100   7   7961   0.7961
13   7961   63377521   8   3775   0.3775
14   3775   14250625   8   2506   0.2506
15   2506   6280036   7   2800   0.28
16   2800   7840000   7   8400   0.84
17   8400   70560000   8   5600   0.56
18   5600   31360000   8   3600   0.36
19   3600   12960000   8   9600   0.96
20   9600   92160000   8   1600   0.16
21   1600   2560000   7   5600   0.56
22   5600   31360000   8   3600   0.36
23   3600   12960000   8   9600   0.96
24   9600   92160000   8   1600   0.16
25   1600   2560000   7   5600   0.56
26   5600   31360000   8   3600   0.36
27   3600   12960000   8   9600   0.96
28   9600   92160000   8   1600   0.16
29   1600   2560000   7   5600   0.56

************************************************************
i:  4820
ITERACIÓN Xn Xn*Xn Longitud Ui Rn
0   4820   23232400   8   2324   0.2324
1   2324   5400976   7   4009   0.4009
2   4009   16072081   8   0720   0.072
3   720   518400   6   1840   0.184
4   1840   3385600   7   3856   0.3856
5   3856   14868736   8   8687   0.8687
6   8687   75463969   8   4639   0.4639
7   4639   21520321   8   5203   0.5203
8   5203   27071209   8   0712   0.0712
9   712   506944   6   0694   0.0694
10   694   481636   6   8163   0.8163
11   8163   66634569   8   6345   0.6345
12   6345   40259025   8   2590   0.259
13   2590   6708100   7   7081   0.7081
14   7081   50140561   8   1405   0.1405
15   1405   1974025   7   9740   0.974
16   9740   94867600   8   8676   0.8676
17   8676   75272976   8   2729   0.2729
18   2729   7447441   7   4474   0.4474
19   4474   20016676   8   0166   0.0166
20   166   27556   5   2755   0.2755
21   2755   7590025   7   5900   0.59
22   5900   34810000   8   8100   0.81
23   8100   65610000   8   6100   0.61
24   6100   37210000   8   2100   0.21
25   2100   4410000   7   4100   0.41
26   4100   16810000   8   8100   0.81
27   8100   65610000   8   6100   0.61
28   6100   37210000   8   2100   0.21
29   2100   4410000   7   4100   0.41

************************************************************
i:  3792
ITERACIÓN Xn Xn*Xn Longitud Ui Rn
0   3792   14379264   8   3792   0.3792
1   3792   14379264   8   3792   0.3792
2   3792   14379264   8   3792   0.3792
3   3792   14379264   8   3792   0.3792
```

| | | | | | |
|---|---|---|---|---|---|
| 4 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 5 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 6 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 7 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 8 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 9 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 10 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 11 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 12 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 13 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 14 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 15 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 16 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 17 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 18 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 19 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 20 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 21 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 22 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 23 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 24 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 25 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 26 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 27 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 28 | 3792 | 14379264 | 8 | 3792 | 0.3792 |
| 29 | 3792 | 14379264 | 8 | 3792 | 0.3792 |

# Memoria

In [3]:
```
mem = psutil.virtual_memory()
memoria=mem.total
memoria
```

Out[3]:
```
12776136704
```

In [4]:
```
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 el digito:"))
iteraciones = int(input("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('Error')
        break

df=pd.DataFrame({"Iteracion Xn":xn, "Xn*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: 12776136704
Ingrese el digito:4
Iteraciones: 30
   Iteracion Xn                   Xn*Xn   Ui       Rn
0   12776136704  0163229669079295983616  9079  0.9079
1          9079                82428241  4282  0.4282
2          4282                18335524  3355  0.3355
3          3355                11256025  2560  0.2560
4          2560                06553600  5536  0.5536
5          5536                30647296  6472  0.6472
6          6472                41886784  8867  0.8867
7          8867                78623689  6236  0.6236
8          6236                38887696  8876  0.8876
9          8876                78783376  7833  0.7833
10         7833                61355889  3558  0.3558
11         3558                12659364  6593  0.6593
12         6593                43467649  4676  0.4676
13         4676                21864976  8649  0.8649
14         8649                74805201  8052  0.8052
15         8052                64834704  8347  0.8347
16         8347                69672409  6724  0.6724
17         6724                45212176  2121  0.2121
18         2121                04498641  4986  0.4986
19         4986                24860196  8601  0.8601
20         8601                73977201  9772  0.9772
21         9772                95491984  4919  0.4919
22         4919                24196561  1965  0.1965
23         1965                03861225  8612  0.8612
24         8612                74166544  1665  0.1665
25         1665                02772225  7722  0.7722
26         7722                59629284  6292  0.6292
27         6292                39589264  5892  0.5892
28         5892                34715664  7156  0.7156
29         7156                51208336  2083  0.2083
```

# Frecuencia

In [5]:
```
frecuencia =psutil.cpu_freq()
frecuencia = int(frecuencia.current)
frecuencia
```

Out[5]: 1801

In [15]:
```
numero = frecuencia
print("Semilla:",numero)

digito=int(input("Ingrese el digito:"))
iteraciones = int(input("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('Error')
        break

df=pd.DataFrame({"Iteracion  Xn":xn, "Xn*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: 1801
Ingrese el digito:4
```

```
Iteraciones: 20
   Iteracion  Xn     Xn*Xn    Ui      Rn
0           1801  03243601  2436  0.2436
1           2436  05934096  9340  0.9340
2           9340  87235600  2356  0.2356
3           2356  05550736  5507  0.5507
4           5507  30327049  3270  0.3270
5           3270  10692900  6929  0.6929
6           6929  48011041  0110  0.0110
7           0110    012100  1210  0.1210
8           1210  01464100  4641  0.4641
9           4641  21538881  5388  0.5388
10          5388  29030544  0305  0.0305
11          0305    093025  9302  0.9302
12          9302  86527204  5272  0.5272
13          5272  27793984  7939  0.7939
14          7939  63027721  0277  0.0277
15          0277    076729  7672  0.7672
16          7672  58859584  8595  0.8595
17          8595  73874025  8740  0.8740
18          8740  76387600  3876  0.3876
19          3876  15023376  0233  0.0233
```

# Disco

In [7]:
```
disco = psutil.disk_usage('/')
disco = int(disco.total/10000000)
disco
```

Out[7]:
```
23886
```

In [9]:
```
from collections import Counter
from collections import defaultdict
import random
import psutil
import numpy as np
import pandas as pd
import math

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

digito=int(input("Ingrese el digito:"))
iteraciones = int(input("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('Error')
        break

df=pd.DataFrame({"Iteracion Xn":xn, "Xn*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: 23886
Ingrese el digito:4
Iteraciones:30
   Iteracion Xn       Xn*Xn    Ui      Rn
0         23886  0570540996  0540  0.0540
1          0540      291600  9160  0.9160
2          9160    83905600  9056  0.9056
3          9056    82011136  0111  0.0111
4          0111      012321  1232  0.1232
5          1232    01517824  5178  0.5178
6          5178    26811684  8116  0.8116
7          8116    65869456  8694  0.8694
8          8694    75585636  5856  0.5856
9          5856    34292736  2927  0.2927
10         2927    08567329  5673  0.5673
11         5673    32182929  1829  0.1829
12         1829    03345241  3452  0.3452
13         3452    11916304  9163  0.9163
14         9163    83960569  9605  0.9605
15         9605    92256025  2560  0.2560
16         2560    06553600  5536  0.5536
17         5536    30647296  6472  0.6472
18         6472    41886784  8867  0.8867
19         8867    78623689  6236  0.6236
20         6236    38887696  8876  0.8876
21         8876    78783376  7833  0.7833
22         7833    61355889  3558  0.3558
23         3558    12659364  6593  0.6593
24         6593    43467649  4676  0.4676
25         4676    21864976  8649  0.8649
26         8649    74805201  8052  0.8052
27         8052    64834704  8347  0.8347
28         8347    69672409  6724  0.6724
29         6724    45212176  2121  0.2121
```

# Numero de Lectura de Disco

In [10]:
```
numero_write=psutil.disk_io_counters()
```

```
numero_write=numero_write.write_count
numero_write
```

Out[10]: 525365

In [11]:
```
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 el digito:"))
iteraciones = int(input("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('Error')
        break

df=pd.DataFrame({"Iteracion  Xn":xn, "Xn*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: 525365
Ingrese el digito:4
Iteraciones:30
   Iteracion  Xn         Xn*Xn   Ui       Rn
0         525365  276008383225  0838  0.0838
1           0838        702244  0224  0.0224
2           0224        050176  5017  0.5017
3           5017      25170289  1702  0.1702
4           1702      02896804  8968  0.8968
5           8968      80425024  4250  0.4250
6           4250      18062500  0625  0.0625
7           0625        390625  9062  0.9062
8           9062      82119844  1198  0.1198
9           1198      01435204  4352  0.4352
10          4352      18939904  9399  0.9399
11          9399      88341201  3412  0.3412
12          3412      11641744  6417  0.6417
13          6417      41177889  1778  0.1778
14          1778      03161284  1612  0.1612
15          1612      02598544  5985  0.5985
16          5985      35820225  8202  0.8202
17          8202      67272804  2728  0.2728
18          2728      07441984  4419  0.4419
19          4419      19527561  5275  0.5275
20          5275      27825625  8256  0.8256
21          8256      68161536  1615  0.1615
22          1615      02608225  6082  0.6082
23          6082      36990724  9907  0.9907
24          9907      98148649  1486  0.1486
25          1486      02208196  2081  0.2081
26          2081      04330561  3305  0.3305
27          3305      10923025  9230  0.9230
28          9230      85192900  1929  0.1929
29          1929      03721041  7210  0.7210
```

# Número de Bytes recibidos

In [12]:
```
bytes_rec = psutil.net_io_counters()
bytes_rec= bytes_rec.bytes_recv
bytes_rec
```

Out[12]:
```
17030534
```

In [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 el digito:"))
iteraciones = int(input("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('Error')
        break

df=pd.DataFrame({"Iteracion Xn":xn, "Xn*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: 17030534
Ingrese el digito:4
Iteraciones:20
   Iteracion Xn             Xn*Xn    Ui      Rn
0      17030534  0290039088325156  9088  0.9088
1          9088          82591744  5917  0.5917
2          5917          35010889  0108  0.0108
3          0108            011664  1166  0.1166
4          1166          01359556  3595  0.3595
5          3595          12924025  9240  0.9240
6          9240          85377600  3776  0.3776
7          3776          14258176  2581  0.2581
8          2581          06661561  6615  0.6615
9          6615          43758225  7582  0.7582
10         7582          57486724  4867  0.4867
11         4867          23687689  6876  0.6876
12         6876          47279376  2793  0.2793
13         2793          07800849  8008  0.8008
14         8008          64128064  1280  0.1280
15         1280          01638400  6384  0.6384
16         6384          40755456  7554  0.7554
17         7554          57062916  0629  0.0629
18         0629            395641  9564  0.9564
19         9564          91470096  4700  0.4700
```