trabajo encargado numero 01

ANDRÉS LADISLAO CORNEJO PINTO cod:190555

octubre 2022

1 Trabajo Practico - N° 001

Implementar segun el ejemplo de clase una funci´on numPar que reciba un parametro n y calcule el n´umeros pares entre 1 y n, as´ı mismo realizar el c´alculo de tiempo para la evaluaci´on de la implementaci´on, para hacer el llamado a la funci´on se deber´a hacer a traves de uno o m´as threads. Las pruebas unitarias deberan ser con los siguientes parametros a la funcion numPar: n=10,n=30,n=200,n=1000

2 Codigo fuente de la solucion

In [ ]:
```
codigo #1
import time #pip install time
import math
from threading import Thread

def npar(n):
    time_ini = time.time()
    i=0
    while(i!=0):
        print(i)
        i= i+2
    print("total" , math.floor(n/2))
    time_end = time.time()
    total= time_end - time_ini
    print("tiempo",total)

npar(1000)
```

codigo1

In [3]:
```
#codigo #2
import time #pip install time
import math
from threading import Thread

def npar2(n):
    time_ini = time.time()

    #calculo
    for i in  range(n):
        if i % 2 ==0:
            print(i)
    print("total", math.floor(n/2))
    time_end = time.time()
    total = time_end - time_ini
    print("tiempo", total)


npar2(10)
```

```
0
2
4
6
8
total 5
tiempo 0.0
```

In [ ]:

codigo 2

3 capturas de los resultados

In [5]:
```
#codigo #2
import time #pip install time
import math
from threading import Thread

def npar2(n):
    time_ini = time.time()

    #calculo
    for i in  range(n):
        if i % 2 ==0:
            print(i)
    print("total", math.floor(n/2))
    time_end = time.time()
    total = time_end - time_ini
    print("tiempo", total)


npar2(30)
```

```
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
total 15
tiempo 0.0
```

Figure 1: resultado numero 1

In [8]:
```python
#codigo #2
import time #pip install time
import math
from threading import Thread

def npar2(n):
    time_ini = time.time()
    print("inicio",time_ini)

    #calculo
    for i in  range(n):
        if i % 2 ==0:
            print(i)
    print("total", math.floor(n/2))
    time_end = time.time()
    print("final",time_end)
    total = time_end - time_ini

    print("tiempo", total)


npar2(10)
```

```
inicio 1666002439.3138266
0
2
4
6
8
total 5
final 1666002439.3138266
tiempo 0.0
```

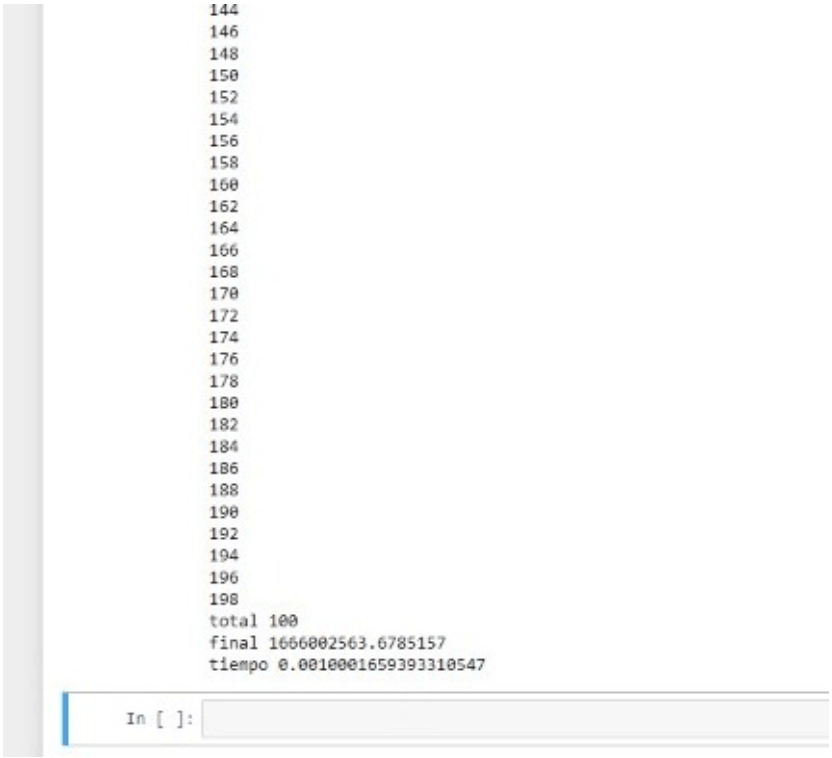
resultado numero 2

```
def npar2(n):
    time_ini = time.time()
    print("inicio",time_ini)

    #calculo
    for i in  range(n):
        if i % 2 ==0:
            print(i)
    print("total", math.floor(n/2))
    time_end = time.time()
    print("final",time_end)
    total = time_end - time_ini

    print("tiempo", total)


npar2(5030)
```

```
4998
5000
5002
5004
5006
5008
5010
5012
5014
5016
5018
5020
5022
5024
5026
5028
total 2515
final 1666002499.8515205
tiempo 0.006000041961669922
```

```
In [ ]:
```

resultado numero 3



resultado numero 4

4 interpretacion

pordemos deducir que en los 4 resultados , los tiempos de respuesta son diferentes mientras usamos el metodo convencional el tiempo de respuesta es mayor al de cuando usamos los threads llegando a la conclusion de que se reduce significativamente el tiempo.