Durante estos temas estuvimos programando algunos algoritmos que están relacionados a las matemáticas uno de ellos es saber si un número es primo o no, este algoritmo me pareció interesante ya que en la asignatura de Teoría de números vimos una introducción a este tema, y así mismo con los números Fibonacci, que antes de este tema no tenía muy en claro que eran estos números, pero gracias a este algoritmo les pude entender.

Numero primo: Empezare este reporte con el numero primo que pues sabemos que para que sea numero primo el número se debe de dividir entre el 1 y sí mismo, a esto se le conoce numero primo.

Algoritmo: Creamos una variable global contador para saber el número de operaciones que se le deben de realzar por cada número y con esto determinamos que los números menores de la raíz cuadrada de n son divisibles entre ellos quedando como resto 0, entonces no es primo

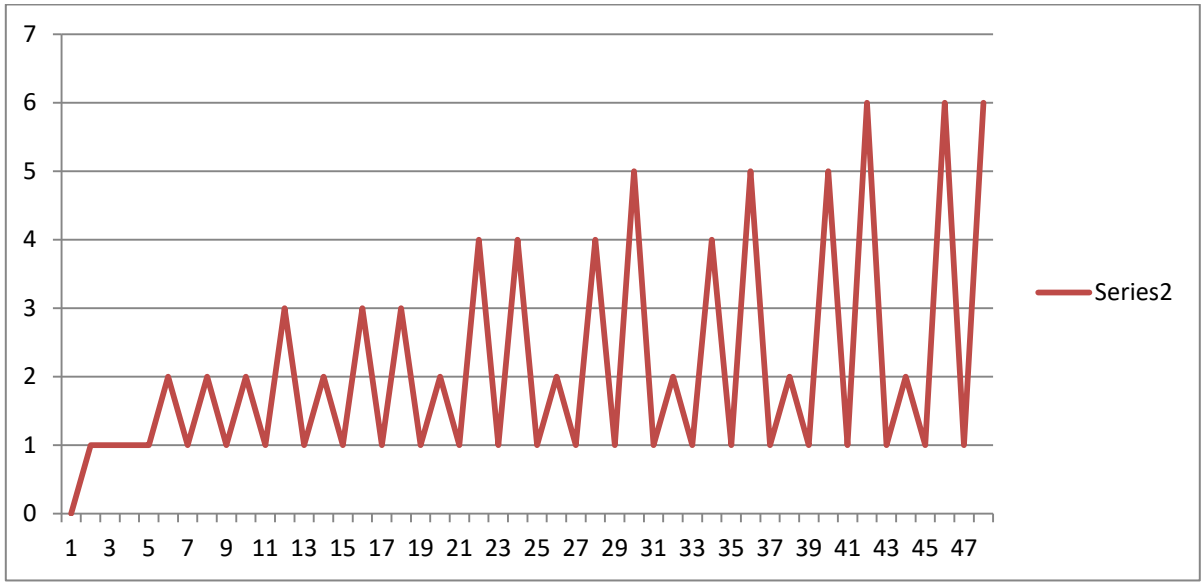
```
def primo(n):
    global cnt

    for i in range (2,round(n**(1/2)+1)):
        cnt=cnt+1
        if ((n%i)==0):
            return ("No es primo")

    return ("Es primo")



for i in range(2,50):
    cnt=0
    primo(i)
    print(i,cnt)
```

Tabla de números primos

	0
3	1
4	1
5	1
6	1
7	2
8	1
9	2
10	1
11	2
12	1
13	3
14	1
15	2
16	1
17	3
18	1
19	3
20	1
21	2
22	1
23	4
24	1
25	4
26	1
27	2
28	1
29	4
30	1
31	5
32	1
33	2
34	1
35	4
36	1
37	5
38	1
39	2
40	1
41	5
42	1

43	6
44	1
45	2
46	1
47	6
48	1
49	6

GRAFICA



Ahora estaré usando los algoritmos de números Fibonacci que es una sucesión infinita de números enteros donde un número es resultado de la suma de sus dos anteriores.

Hemos realizado varios algoritmos para ver su eficacia y que alguno es más rápido que los otros

Finobacci 1

En este Fibonacci es necesario primero es necesario definir una función para darle instrucciones acerca de cómo obtener el resultado deseado.

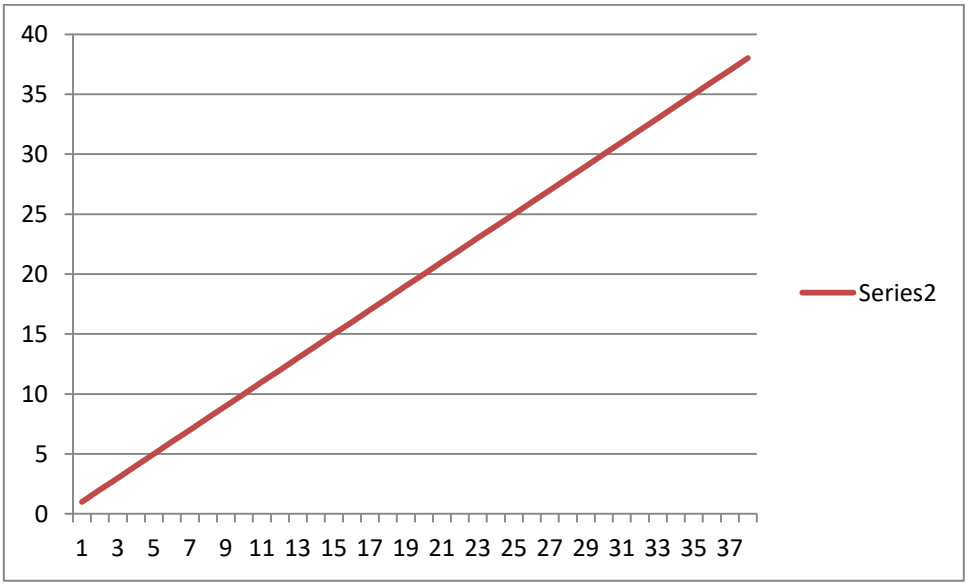
```
def fibo(n):
    global cnt
    cnt=0
    if n==0 or n==1:
        return (1)
    r,r1,r2=0,1,1
    for i in range (2,n+1):
        cnt+=1
        r=r1+r2
        r2=r1
        r1=r
    return r, cnt
for i in range (2, 40):
            cnt=0
            print(i, fibo(i), cnt )
```

Tablas con los valores hasta el 39

2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9
11	10
12	11

13	12
14	13
15	14
16	15
17	16
18	17
19	18
20	19
21	20
22	21
23	22
24	23
25	24
26	25
27	26
28	27
29	28
30	29
31	30
32	31
33	32
34	33
35	34
36	35
37	36
38	37
39	38

Grafica:



Fibonacci 2

En primer lugar declaramos una variable global, que será la encargada de contar el número de interacciones realizadas por el algoritmo para poder obtener dicho elemento que nosotros le solicitamos.

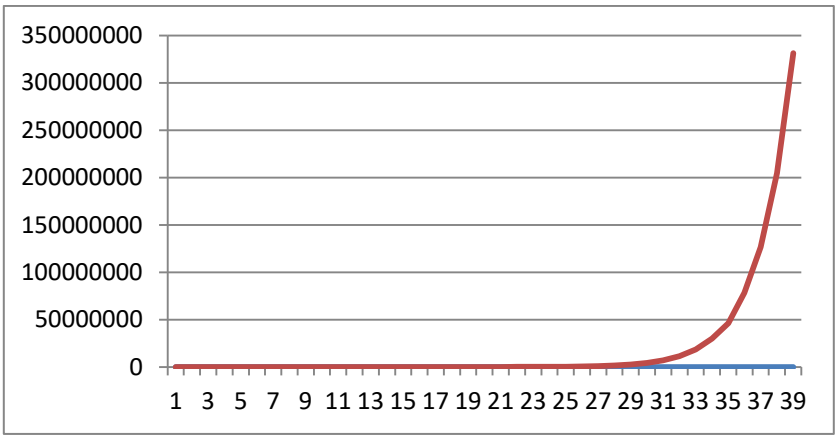
```
cnt=0

def fibonacci (n):

   global cnt

   cnt+=1

   if n==0 or n==1:

        return(1)



   else:

      return fibonacci(n-2)+fibonacci(n-1)



for i in range(2,40):

   cnt=0

   print(i, fibonacci(i), cnt)
```

Tabla de numero de operaciones que se realizan

2	3
3	5
4	9

5	15
6	25
7	41
8	67
9	109
10	177
11	287
12	465
13	753
14	1219
15	1973
16	3193
17	5167
18	8361
19	13529
20	21891
21	35421
22	57313
23	92735
24	150049
25	242785
26	392835
27	635621
28	1028457
29	1664079
30	2692537
31	4356617
32	7049155
33	11405773
34	18454929
35	29860703
36	46315633
37	78176337
38	126491971
39	204668309
40	331160281

Grafica:



Como una observación este algoritmo de Fibonacci lo había hecho los primero 50 pero debido a que se tarda mucho lo decidí dejarlo hasta los primeros 40 que de echo aun así es tardado

Fibonacci 3

Este fibonacci es con memoria y es necesario definir un arreglo vacio, después una función para darle instrucciones acerca de como obtener el elemento deseado.

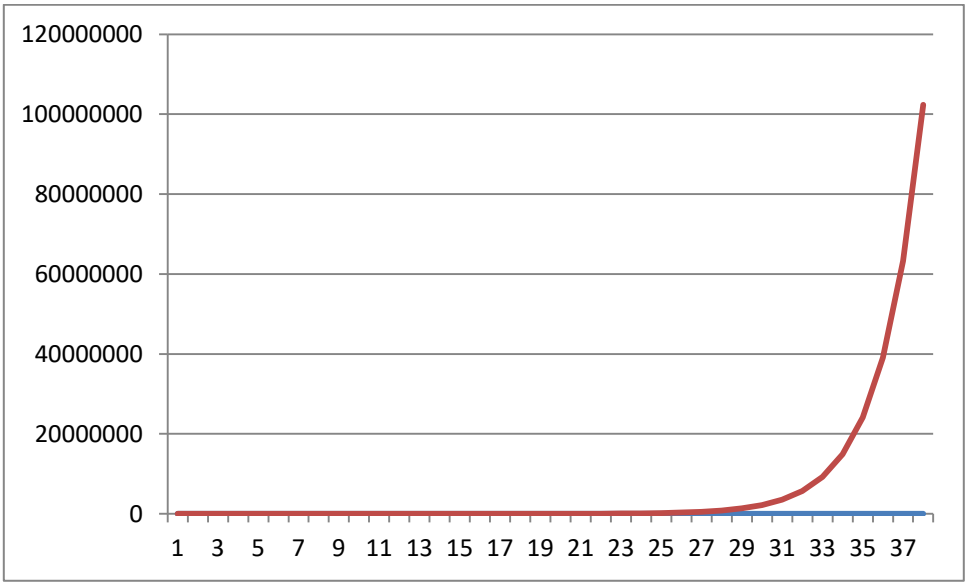
```
arr={}

cnt=0

def fibonacci(n):

    global arr, cnt

    cnt+=1

    if n==0 or n==1:

        return(1)

    if n in arr:

        return arr[n]

    else:

        val=fibonacci(n-2)+fibonacci(n-1)
```

```
        arr[n]=val
        return val

for i in range(2,40):
    cnt=0
    print(i,fibonacci(i),cnt)
```

Tablas de valores:

2	2
3	3
4	5
5	8
6	13
7	21
8	34
9	55
10	89
11	144
12	233
13	377
14	610
15	987
16	1587
17	2584
18	4181
19	6765
20	10946
21	17711
22	28657
23	43368
24	75025
25	121393
26	196418
27	317811
28	514229
29	832040
30	1346269

31	2178309
32	3524578
33	5702887
34	9227465
35	14930352
36	24157817
37	39088169
38	63245986
39	102334155



Conclusión:

Esta etapa me pareció más interesante y creo que le he entendido un poco más a Python y además me he logrado dar cuenta de porque se llama la materia Matemáticas computacionales ya que cosas en las que algún momento las vamos a ver en nuestra carrera o ya las hemos visto aquí las vamos a ver, claro no todas de ellas porque pues nunca acabaríamos pero hemos hecho un algoritmo de numero primo que eso está muy padre como lo logramos, además entendí el concepto de numero Fibonacci además esto se puede seguir utilizando en los otros temas que veamos mas adelantes.