## Ejercicio #1

Construya una función recursiva en Python llamada **invertirNum(N)** que reciba un número entero cualquiera y retorna una hilera que ofrece el resultado del mecanismo de cálculo de la raíz de un número de acuerdo a lo indicado anteriormente.

|  |  |
| --- | --- |
| Número Original | Número Invertido |
| 12345 | 54321 |
| 10025 | 52001 |
| -45986 | -68954 |

## Ejercicio #2

Construya una función recursiva en Python llamada **divisoresDe(N)** que reciba un número entero positivo cualquiera y retorna sus divisores en una hilera.

|  |  |
| --- | --- |
| Número Original | Reporte de sus divisores |
| 42 | “1-2-3-6-7-21-42” |
| 81 | “1-3-9-17-81” |
| 256 | “1-2-4-8-16-32-64-128-256” |

## Ejercicio #3

Construya una función recursiva en Python llamada **sumaDivisores(N)** que reciba un número entero positivo cualquiera y retorna un valor numérico que representa la suma de todos los divisores de N.

|  |  |
| --- | --- |
| Número Original | Suma de sus divisores |
| 42 | 82 |
| 81 | 111 |
| 256 | 511 |

## Ejercicio #4

Un número A se define como amigo de otro número B si y sólo sí la suma de divisores de A excepto él es igual a B y la suma de divisores de B excepto él es igual a A.

Construya una función recursiva en Python llamada **sonAmigos (a,b)** que reciba dos números enteros positivos cualquiera y retorna si ambos números son amigos o no.

|  |  |
| --- | --- |
| Números A y B | ¿Son amigos? |
| A: 220  B:284 | Divisores de 220:  1+ 2+ 4+ 5+ 10+ 11+ 20+ 22+ 44+ 55+ 110= 284  Divisores de 284= 1+2+4+71+142=220  220 y 284 son amigos |
| A:170  B:215 | Divisores de 170: 1+2+5+34+85 = 131  Divisores de 215: 1+5+43= 49  170 y 215 NO son amigos |

Construya una función recursiva en Python llamada **sonAmigos (a,b)** que reciba dos números enteros positivos cualquiera y retorna un valor booleano si ambos números a y b son amigos.

## Ejercicio #5

Un número perfecto es un [número natural](http://es.wikipedia.org/wiki/N%C3%BAmero_natural) que es igual a la suma de sus [divisores propios positivos](http://es.wikipedia.org/wiki/Divisor_propio), sin incluirse él mismo. Dicho de otra forma, un número perfecto es aquel que es [amigo](http://es.wikipedia.org/wiki/N%C3%BAmeros_amigos) de sí mismo.

|  |  |
| --- | --- |
| Números | ¿Es perfecto? |
| A=14 | Divisores de A: 1+2+7 = 10  El número 14 no es perfecto, retorna False |
| A=28 | Divisores de A: 1+2+4+7+14 = 28  El número 28 es perfecto, retorna True |

Construya una función recursiva en Python llamada **esPerfecto (a)** que reciba un número entero positivo mayor que cero y retorna un valor booleano si el número a es perfecto.

## Ejercicio #6

La raíz de un número N se determina por medio de la obtención de sus factores primos, haciendo grupos de divisores de acuerdo al índice de la raíz

|  |  |
| --- | --- |
| Ejemplo: Se desea obtener la raíz cuadrada de 16.  Se determinan los factores primos | 16 2  8 2  4 2  2 2  1 |
| Como la raíz es cuadrada se hacen grupos de dos factores iguales | 2 \* 2 \* 2 \* 2 |
| Se saca un valor por grupo y se dejan los que no se agruparon dentro de la raíz | 2 \* 2 |
| Resultado de la raíz cuadrada de 16 | 4 |

De este modo:

|  |  |  |
| --- | --- | --- |
|  | Descomposición de primos y agrupación por índice | Resultado |
| Raíz cuadrada de 40 | 2 \* 2 \* 2 \* 5 | 2 \* √10 |
| Raíz cúbica de 40 | 2 \* 2 \* 2 \* 5 | 2 \* √5 |
| Raíz cuadrada de 81 | 3 \* 3 \* 3 \* 3 | 9 |

Construya una función recursiva en Python llamada **calculaRaiz(N, indice)** que reciba un número entero mayor que cero y retorna una hilera que ofrece el resultado del mecanismo de cálculo de la raíz de un número de acuerdo a lo indicado anteriormente.

## Ejercicio #7

Un número primo de Wilson o número de Wilson, llamado así en honor al [matemático](http://es.wikipedia.org/wiki/Matem%C3%A1tico) [John Wilson](http://es.wikipedia.org/w/index.php?title=John_Wilson_(matem%C3%A1tico)&action=edit&redlink=1), es un tipo de [primo](http://es.wikipedia.org/wiki/N%C3%BAmero_primo) p tal que p² divide a (p − 1)! + 1, donde «!» denota la [función factorial](http://es.wikipedia.org/wiki/Factorial).

Tiene cierta similitud con el [teorema de Wilson](http://es.wikipedia.org/wiki/Teorema_de_Wilson), el cual cita que cada número primo p divide a (p − 1)! + 1.

Se ha conjeturizado que existe un cantidad infinta de números de esta categoría, no obstante los números Wilson que se conocen entre 1 y 1000 son el 5, 13 y 563.

La función «!» o función factorial se define a continuación:

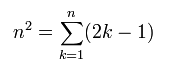

   n! =
   \begin{cases}
      1              & \text{si, } n = 0 \\
      (n-1)!\times n & \text{si, } n > 0
   \end{cases}


Construya una función en Python llamada numerosWilson(desde, hasta) para que pueda comprobar que si se ejecuta la función utilizando los parámetros desde = 1 y hasta=1000 se reporte como números Wilson en ese rango a los números 5, 13 y 563.

Elabore todas las funciones que se requieran para contestar esta pregunta. Considere que esta característica es propia de los números primos.

## Ejercicio #8

Elevar un número positivo n al cuadrado (n2) se puede describir por medio la suma de los primeros n números impares o bien por medio de la fórmula:

  
Ejemplo 1:

42= (2\*1-1)+(2\*2-1)+(2\*3-1)+(2\*4-1)

42 = 1 + 3+ 5 + 7

42 = 16

Ejemplo 2:

102= ( 2\*1-1)+(2\*2-1)+(2\*3-1)+(2\*4-1)+ ( 2\*5-1)+(2\*6-1)+(2\*7-1)+(2\*8-1)+(2\*9-1)+(2\*10-1)

102= 1+3+5+7+9+11+13+15+17+19

102=100

Escriba una función recursiva en Pyhton llamada **cuadradoDeN(N)** que reciba un valor numérico N y retorne su cuadrado mediante la fórmula descrita en el proceso anterior.

## Ejercicio #9

Un cuadrado perfecto se define como aquel número que se puede escribir como un número X tal que X2 = N.

Ejemplo

8 no es un cuadrado perfecto pues no hay un número X tal que X2 = 8

9 es un cuadrado perfecto pues sí hay un número X tal que X2 = 9 y ese X=3

Escriba una función recursiva en Pyhton llamada **esCuadradoPerfecto(N)** que reciba un valor numérico N que determine si este número N es cuadrado perfecto, en cuyo caso retornará el valor del número del que es cuadrado, y eb caso de que no lo sea retorne un mensaje indicando esta situación.

## Ejercicio #10

**El teorema de los cuatro cuadrados de Lagrange.**

Este teorema dice que cada [número entero](http://es.wikipedia.org/wiki/N%C3%BAmero_entero) positivo puede expresarse como la suma de cuatro cuadrados de enteros. Por ejemplo,

31 = 52 + 22 + 12 + 12

310 = 172 + 42 + 22 + 12

Escriba una función recursiva en Python llamada **cuatroCuadrados(N)** que reciba un valor numérico N y retorne una hilera con los cuatro cuadrados que forman al número N recibido