

## ESTRUCTURAS DE DATOS



**Enlace de github:** <https://github.com/bereduro/Actividad-01-Estructuras-De-Datos.git>

**Docente:** Ing. Gustavo Tantani Mamani

**Estudiante:** Brody Deiby Velarde Hurtado

SANTA CRUZ-BOLIVIA

<https://github.com/bereduro/Actividad-01-Estructuras-De-Datos.git>

"""

1. Algoritmo recursivo para determinar si n es primo.

Ej.: 7 es primo, 8 no es primo

"""

```
def es_primo(n, divisor=2):
```

```
    if n < 2:
```

```
        return False
```

```
    if divisor * divisor > n:
```

```
        return True
```

```
    if n % divisor == 0:
```

```
        return False
```

```
    return es_primo(n, divisor + 1)
```

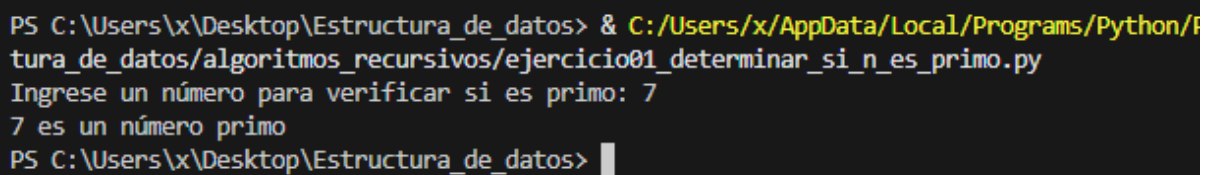
```
n = int(input("Ingrese un número para verificar si es primo: "))
```

```
if es_primo(n):
```

```
    print(f"{n} es un número primo")
```

```
else:
```

```
    print(f"{n} no es un número primo")
```



```
PS C:\Users\x\Desktop\Estructura_de_datos> & C:/Users/x/AppData/Local/Programs/Python/Python39-64/Scripts/python.exe C:/Users/x/Desktop/Estructura_de_datos/algoritmos_recursivos/ejercicio01_determinar_si_n_es_primo.py
Ingrese un número para verificar si es primo: 7
7 es un número primo
PS C:\Users\x\Desktop\Estructura_de_datos> |
```

"""

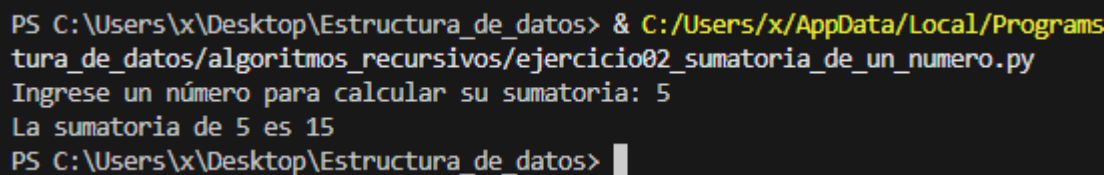
<https://github.com/bereduro/Actividad-01-Estructuras-De-Datos.git>

2. Algoritmo recursivo para calcular la sumatoria de un número.

Ej.: n= 5, suma = 1+2+3+4+5 = 15

"""

```
def sumatoria(n):
    if n <= 1:
        return n
    else:
        return n + sumatoria(n - 1)
n = int(input("Ingrese un número para calcular su sumatoria: "))
print(f"La sumatoria de {n} es {sumatoria(n)}")
```



```
PS C:\Users\x\Desktop\Estructura_de_datos> & C:/Users/x/AppData/Local/Programs/Python/Python311/Python.exe C:\Users\x\Desktop\Estructura_de_datos/algoritmos_rekursivos/ejercicio02_sumatoria_de_un_numero.py
Ingrese un número para calcular su sumatoria: 5
La sumatoria de 5 es 15
PS C:\Users\x\Desktop\Estructura_de_datos> █
```

<https://github.com/bereduro/Actividad-01-Estructuras-De-Datos.git>

"""

3. Algoritmo recursivo para determinar si un número n esta ordenado con respecto a sus dígitos.

Ej.: n = 1258, res = true

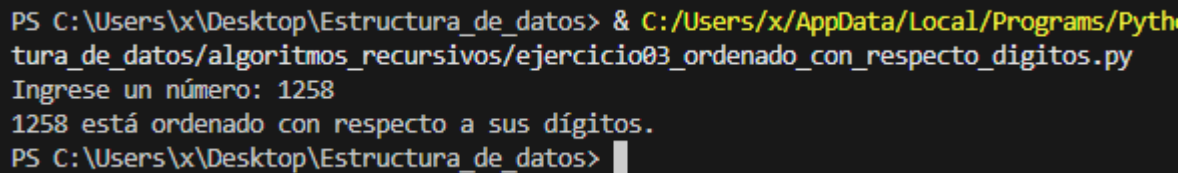
"""

```
def esta_ordenado(n):
    n = abs(n)
    if n < 10:
        return True
    ultimo = n % 10
    penultimo = (n // 10) % 10
    if penultimo > ultimo:
        return False
    return esta_ordenado(n // 10)
```

```

n = int(input("Ingrese un número: "))
if esta_ordenado(n):
    print(f"{n} está ordenado con respecto a sus dígitos.")
else:
    print(f"{n} NO está ordenado con respecto a sus dígitos.")

```



```

PS C:\Users\x\Desktop\Estructura_de_datos> & C:/Users/x/AppData/Local/Programs/Python/Python39-64/Scripts/python.exe C:\Users\x\Desktop\Estructura_de_datos\algoritmos_recursivos\ejercicio03_ordenado_con_respecto_digitos.py
Ingrese un número: 1258
1258 está ordenado con respecto a sus dígitos.
PS C:\Users\x\Desktop\Estructura_de_datos>

```

"""

<https://github.com/bereduro/Actividad-01-Estructuras-De-Datos.git>

4. Algoritmo recursivo para calcular la suma de los divisores de N.

Ej.: N = 6 , res = 1+2+3+6 = 12

"""

```

def suma_divisores(n, divisor=1):
    if divisor > n:
        return 0
    if n % divisor == 0:
        return divisor + suma_divisores(n, divisor + 1)
    else:
        return suma_divisores(n, divisor + 1)

```

N = 6

res = suma\_divisores(N)

print(f"La suma de los divisores de {N} es {res}")

```
PS C:\Users\x\Desktop\Estructura_de_datos> & C:/Users/x/AppData/Local/Programs/Python/Python313/Scripts/python C:\Users\x\Desktop\Estructura_de_datos/algoritmos_recurativos/ejercicio04_calcular_suma_de_divisores_de_N.py"
Introduce un número entero positivo N: 6
La suma de los divisores de 6 es 12
PS C:\Users\x\Desktop\Estructura_de_datos> █
```

```
"""
```

5. Algoritmo recursivo para calcular la suma de los factores primos de N.

Ej.:  $N=12$ ,  $res=2+2+3=7$

```
"""
```

```
def suma_factores_primos(n, divisor=2):
```

```
    if n == 1:
```

```
        return 0
```

```
    if n % divisor == 0:
```

```
        return divisor + suma_factores_primos(n // divisor, divisor)
```

```
    else:
```

```
        return suma_factores_primos(n, divisor + 1)
```

```
N = int(input("Ingrese un número para calcular la suma de sus factores primos: "))
```

```
res = suma_factores_primos(N)
```

```
print(f"La suma de los factores primos de {N} es: {res}")
```

```
PS C:\Users\x\Desktop\Estructura_de_datos> & C:/Users/x/AppData/Local/Programs/Python/Python313/Scripts/python C:\Users\x\Desktop\Estructura_de_datos/algoritmos_recurativos/ejercicio05_calcular_la_suma_de_factores_primos_de_N.py"
Ingrese un número para calcular la suma de sus factores primos: 12
La suma de los factores primos de 12 es: 7
PS C:\Users\x\Desktop\Estructura_de_datos> █
```

```
"""
```

6. Algoritmo recursivo para calcular el factorial de un número.

Ej.:  $5 = 5 * 4 * 3 * 2 * 1 = 120$

```
"""
```

```
def factorial(n):
```

```
    if n == 0 or n == 1:
```

```
        return 1
```

else:

return n \* factorial(n - 1)

numero = int(input("Ingrese un número: "))

print(f"El factorial de {numero} es {factorial(numero)}")

```
PS C:\Users\x\Desktop\Estructura_de_datos> & C:/Users/x/AppData/Local/Programs/Python/Python39-64/Scripts/python.exe C:\Users\x\Desktop\Estructura_de_datos/algoritmos_rekursivos/ejercicio06_calcular_el_factorial_de_un_numero.py
Ingrese un número: 5
El factorial de 5 es 120
PS C:\Users\x\Desktop\Estructura_de_datos> █
```

"""

7. Algoritmo recursivo para calcular la suma de dígitos de n.

Ej.: 1234 -> 1 + 2 + 3 + 4 = 10

"""

def suma\_digitos(n):

if n < 10:

return n

else:

return n % 10 + suma\_digitos(n // 10)

n = int(input("Ingrese un número para calcular la suma de sus dígitos: "))

resultado = suma\_digitos(n)

print(f"La suma de los dígitos de {n} es: {resultado}")

```
PS C:\Users\x\Desktop\Estructura_de_datos> & C:/Users/x/AppData/Local/Programs/Python/Python39-64/Scripts/python.exe C:\Users\x\Desktop\Estructura_de_datos/algoritmos_rekursivos/ejercicio07_calcular_la_suma_de_digitos_de_n.py
Ingrese un número para calcular la suma de sus dígitos: 1234
La suma de los dígitos de 1234 es: 10
PS C:\Users\x\Desktop\Estructura_de_datos> █
```

"""

8. Algoritmo recursivo para calcular la cantidad de bit unos que

tiene un número al convertir de base 10 a base 2.

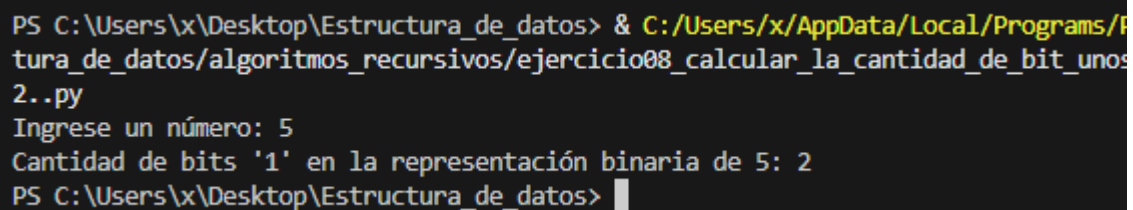
Ej.: N = 5 -> B = 101 -> Res = 2.

"""

```
def contar_unos(n):
    if n == 0:
        return 0
    else:
        return (n % 2) + contar_unos(n // 2)

numero = int(input("Ingrese un número: "))

print(f"Cantidad de bits '1' en la representación binaria de {numero}: {contar_unos(numero)}")
```



```
PS C:\Users\x\Desktop\Estructura_de_datos> & C:/Users/x/AppData/Local/Programs/Python/Python38-64/Scripts/python.exe C:/Users/x/Desktop/Estructura_de_datos/
algoritmos_recursivos/ejercicio08_calcular_la_cantidad_de_bit_unos2.py
Ingrese un número: 5
Cantidad de bits '1' en la representación binaria de 5: 2
PS C:\Users\x\Desktop\Estructura_de_datos> █
```

"""

9. Algoritmo recursivo para convertir a base binario un número en base 10.

Ej.:  $N = 12 \rightarrow \text{Res} = 1100$

"""

```
def decimal_a_binario(n):
    if n < 2:
        return str(n)
    else:
        return decimal_a_binario(n // 2) + str(n % 2)
```

```
n = int(input("Ingrese un número decimal para convertir a binario: "))
resultado = decimal_a_binario(n)
print(f"El número {n} en binario es: {resultado}")
```

```
PS C:\Users\x\Desktop\Estructura_de_datos> & C:/Users/x/AppData/Local/Programs/Python/Python311/Scripts/python C:\Users\x\Desktop\Estructura_de_datos/algoritmos_recursivos/ejercicio09_convertir_a_base_binario_un_numero_base_10.py
Ingrese un número decimal para convertir a binario: 12
El número 12 en binario es: 1100
PS C:\Users\x\Desktop\Estructura_de_datos>
```

"""

10. Algoritmo recursivo para determinar si un número es perfecto.

Ej.:  $N = 28 \rightarrow \text{res} = \text{true}$

"""

```
def es_perfecto(n, divisor=1, suma=0):
```

```
    if divisor == n:
```

```
        return suma == n
```

```
    if n % divisor == 0:
```

```
        suma += divisor
```

```
    return es_perfecto(n, divisor + 1, suma)
```

```
n = int(input("Ingrese un número para verificar si es perfecto: "))
```

```
if es_perfecto(n):
```

```
    print(f"{n} es un número perfecto")
```

```
else:
```

```
    print(f"{n} no es un número perfecto")
```

```
# un número perfecto es aquel que es igual a la suma de sus divisores propios
```

```
# ejemplo, 28 es perfecto porque  $1 + 2 + 4 + 7 + 14 = 28$ 
```

```
PS C:\Users\x\Desktop\Estructura_de_datos> & C:/Users/x/AppData/Local/Programs/Python/Python311/Scripts/python C:\Users\x\Desktop\Estructura_de_datos/algoritmos_recursivos/ejercicio10_determinar_si_un_numero_es_perfecto.py
Ingrese un número para verificar si es perfecto: 28
28 es un número perfecto
PS C:\Users\x\Desktop\Estructura_de_datos> & C:/Users/x/AppData/Local/Programs/Python/Python311/Scripts/python C:\Users\x\Desktop\Estructura_de_datos/algoritmos_recursivos/ejercicio10_determinar_si_un_numero_es_perfecto.py
Ingrese un número para verificar si es perfecto: 33
33 no es un número perfecto
PS C:\Users\x\Desktop\Estructura_de_datos>
```

"""

11. Algoritmo recursivo para determinar si dos números son amigos.



Ej.: 220 y 284 son números amigos

"""

```
def suma_divisores(n, i=1):
```

```
    if i == n:
```

```
        return 0
```

```
    if n % i == 0:
```

```
        return i + suma_divisores(n, i + 1)
```

```
    else:
```

```
        return suma_divisores(n, i + 1)
```

```
def son_amigos(a, b):
```

```
    return suma_divisores(a) == b and suma_divisores(b) == a
```

```
a = int(input("Ingrese el primer número: "))
```

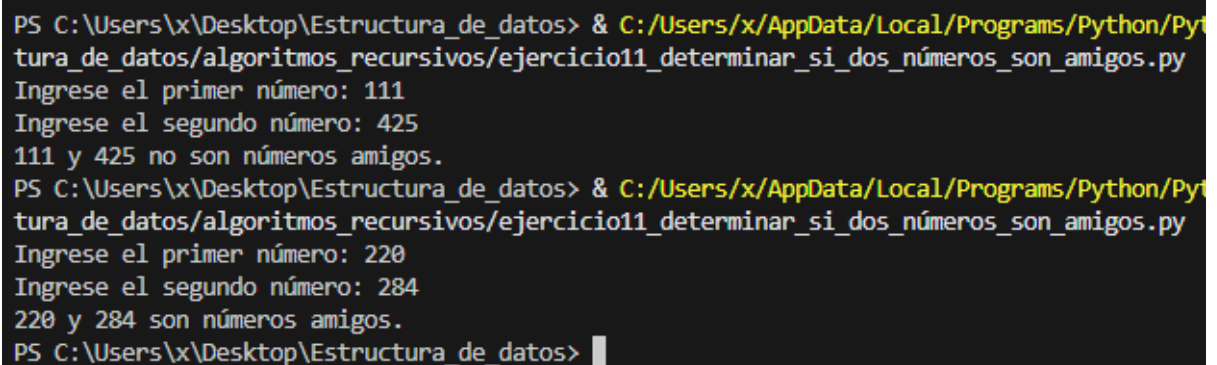
```
b = int(input("Ingrese el segundo número: "))
```

```
if son_amigos(a, b):
```

```
    print(f"{a} y {b} son números amigos.")
```

```
else:
```

```
    print(f"{a} y {b} no son números amigos.")
```



```
PS C:\Users\x\Desktop\Estructura_de_datos> & C:/Users/x/AppData/Local/Programs/Python/Python38-64/Scripts/python.exe C:\Users\x\Desktop\Estructura_de_datos/algoritmos_rekursivos/ejercicio11_determinar_si_dos_números_son_amigos.py
Ingrese el primer número: 111
Ingrese el segundo número: 425
111 y 425 no son números amigos.
PS C:\Users\x\Desktop\Estructura_de_datos> & C:/Users/x/AppData/Local/Programs/Python/Python38-64/Scripts/python.exe C:\Users\x\Desktop\Estructura_de_datos/algoritmos_rekursivos/ejercicio11_determinar_si_dos_números_son_amigos.py
Ingrese el primer número: 220
Ingrese el segundo número: 284
220 y 284 son números amigos.
PS C:\Users\x\Desktop\Estructura_de_datos> █
```

"""

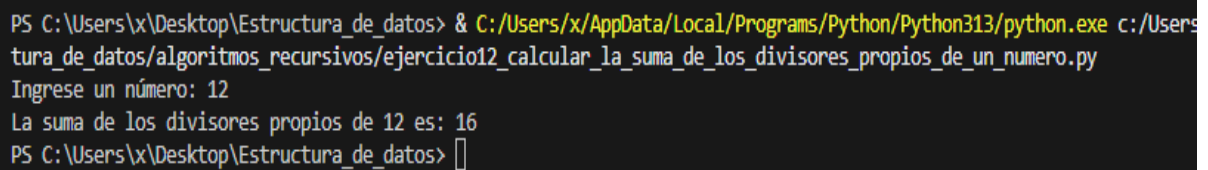
12. Algoritmo recursivo para calcular la suma de los divisores propios de un número.

Ej.:  $N = 12 \rightarrow \text{div} = 1 + 2 + 3 + 4 + 6 \Rightarrow \text{sdp} = 16$

(Los divisores propios son aquellos que dividen exactamente al número y son menores que él)

"""

```
def suma_divisores_propios(n, divisor=1):  
    if divisor >= n:  
        return 0  
    if n % divisor == 0:  
        return divisor + suma_divisores_propios(n, divisor + 1)  
    else:  
        return suma_divisores_propios(n, divisor + 1)  
num = int(input("Ingrese un número: "))  
resultado = suma_divisores_propios(num)  
print(f"La suma de los divisores propios de {num} es: {resultado}")
```



```
PS C:\Users\x\Desktop\Estructura_de_datos> & C:/Users/x/AppData/Local/Programs/Python/Python313/python.exe c:/Users/tura_de_datos/algoritmos_recurativos/ejercicio12_calcular_la_suma_de_los_divisores_propios_de_un_numero.py  
Ingrese un número: 12  
La suma de los divisores propios de 12 es: 16  
PS C:\Users\x\Desktop\Estructura_de_datos> 
```

<https://github.com/bereduro/Actividad-01-Estructuras-De-Datos.git>