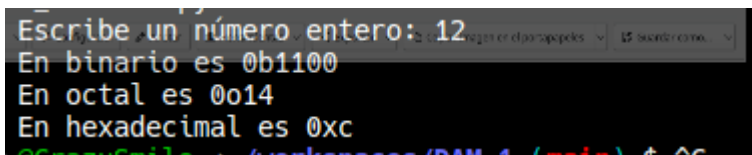


Practica 1.1

Sistema de codificación y aritmética binaria

1. Crea un fichero llamado "codifica_decimal.py" con el siguiente contenido. Una vez guardado pon permisos de ejecución con "chmod +x codifica_decimal.py" y obtén una captura del resultado tras ejecutar "./codifica_decimal.py". ¿Con qué prefijo indica python que está mostrando un número en el sistema de codificación binario, octal y hexadecimal?

```
#!/usr/bin/env python3
# Obtiene una cadena de caracteres escrita por teclado
numero_cadena = input("Escribe un número entero: ")
# Transforma la cadena en un número entero
numero_decimal = int(numero_cadena)
# Muestra la codificación binaria, octal y hexadecimal
print("En binario es", bin(numero_decimal))
print("En octal es", oct(numero_decimal))
print("En hexadecimal es", hex(numero_decimal))
```



```
Escribe un número entero: 12
En binario es 0b1100
En octal es 0o14
En hexadecimal es 0xc
```

El prefijo que indica que esta mostrando:

- **Binario:** El prefijo es el **0b**, que la **b** significa binario.
- **Octal:** El prefijo es el **0o**, donde la **o** es octal.
- **Hexadecimal:** El prefijo es **0x**, que **x** significa hexadecimal.

2. El método "int(string, base)" en python permite convertir una cadena de caracteres (string) a un número entero codificado con la base indicada. Si no se indica la base, por defecto utiliza el sistema decimal (base 10). Crea un programa en python que solicite un número binario y muestre su codificación en octal, hexadecimal y decimal.

```
#!/usr/bin/env python

numero_cadena = input("Escribe un número binario: ")

numero_binario = int(numero_cadena, 2)

print("Número Decimal: ", int(numero_binario))

print("Número hexadecimal: ", hex(numero_binario))

print("Número octal: ", oct(numero_binario))
```

3. Crea un programa en python que solicite un número hexadecimal y muestre su codificación en binario, octal y decimal.

```
#!/usr/bin/env python

numero_cadena = input("Escribe un numero hexadecimal: ")

numero_hexa=int(numero_cadena, 16)

print("Numero decimal: ", int(numero_hexa))

print("Numero binario: ", bin(numero_hexa))

print("Numero octal: ", oct(numero_hexa))
```

4. Codifica a mano de binario a decimal los siguientes números, comprueba con las aplicaciones realizadas anteriormente si la solución es correcta:

a. $10111_2 = 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 23$

b. $110111_2 = 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 55$

5. Codifica a mano de decimal a binario los siguientes números, comprueba con las aplicaciones realizadas anteriormente si la solución es correcta.

a. $101_{10} \Rightarrow 1100101$

128	64	32	16	8	4	2	1
0	1	1	0	0	1	0	1

b. $26_{10} \Rightarrow 11010$

128	64	32	16	8	4	2	1
0	0	0	1	1	0	1	0

6. Codifica a mano de binario a octal y a hexadecimal los siguientes números, comprueba con las aplicaciones realizadas anteriormente si la solución es correcta.

a. $111101110_2 = 1DD_{16} = 756_8$

0001	1110	1110
1	D	D
111	101	110
7	5	6

$$b. 1010111_2 = 57_{16} = 127_8$$

0101	0111
5	7

001	010	111
1	2	7

7. Codifica a mano de hexadecimal a binario los siguientes números, comprueba con las aplicaciones realizadas anteriormente si la solución es correcta.

$$a. 1F1_{16} = 111110001_2$$

1	F	1
0001	1111	0001

$$b. A2_{16} = 10100010_2$$

A	2
1010	0010

8. Realiza a mano las siguientes operaciones en binario:

$$a. 100110111 + 1001111 = 1110000110$$

1100110111
+ 1001111
1110000110

$$b. 10110000 - 101111 = 10000001$$

10110000
-101111
10000001

$$c. 11101110 \times 1010 = 100101001100$$

0	0	0	0	0	0	0	0

			0	0	0	0	0	0	0	0		
			1	1	1	0	1	1	1	0		
			0	0	0	0	0	0	0	0		
			1	1	1	0	1	1	1	0		
			-	-	-	-	-	-	-	-		
			1	0	0	1	0	0	1	1	0	0

d. 101111011 : 101 = 1001011

1	0	1	1	1	1	0	1	1		/	1	0	1
1	0	1									1	0	0
0	0	0	1	1	1								
			1	0	1								
			0	1	0	0	1						
				1	0	1							
				0	0	0	0	1					