

Elaborado: Mtro. Humberto Acevedo hac106@hotmail.com





## Módulo ②

**Tipos de datos, variables, operaciones básicas de entrada y salida, operadores básicos**

**En esta sección aprenderemos:**

- En este módulo, aprenderás:
- Cómo escribir y ejecutar programas simples en Python.
- Qué son los literales, operadores y expresiones en Python.
- Qué son las variables y cuáles son las reglas que las gobiernan.
- Cómo realizar operaciones básicas de entrada y salida.

c106@hotmail.com



# PYTHON

---

## *ELEMENTOS DEL LENGUAJE*

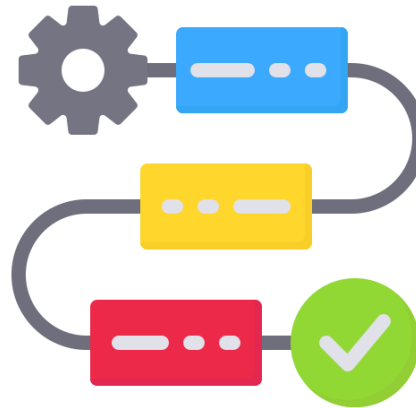
Elaborado: Mtro. Humberto Acevedo hac106@hotmail.com



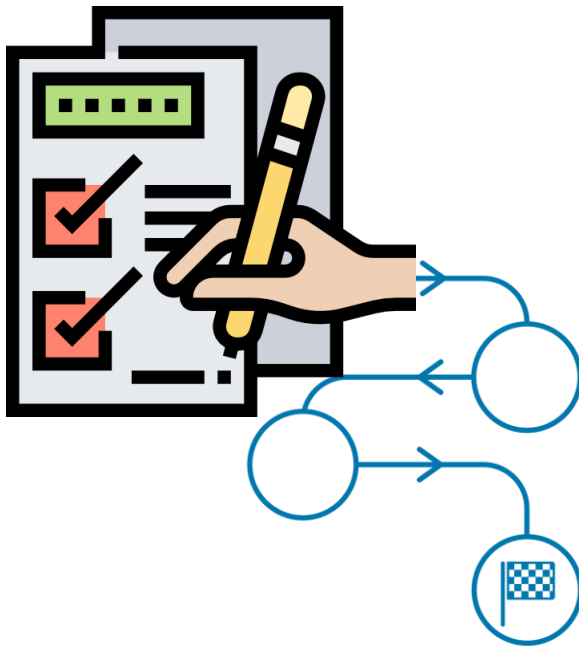
# Flujo de datos

Es el **camino** que siguen los datos a través de un sistema o proceso, desde su **origen** hasta su **destino**.

Puede involucrar operaciones de **entrada**, **procesamiento** y **salida**, así como la comunicación entre diferentes componentes de un sistema.



# Comandos

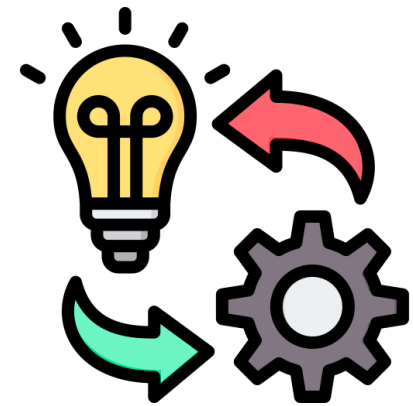


Una **instrucción** que el usuario proporciona a un sistema informático.

Desde la línea de órdenes (como una shell) o desde una llamada de Programación.

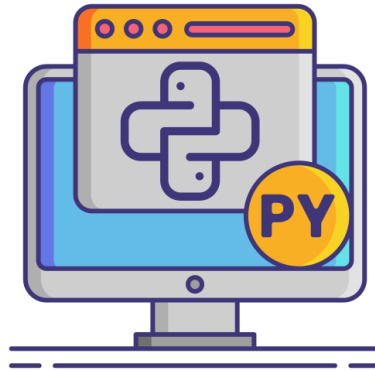
# Funciones

- En programación, una función es un bloque de código que realiza una **tarea específica** y puede ser invocado o llamado desde otras partes del programa.
- Las funciones se utilizan para modularizar el código, mejorar la legibilidad, reutilizar código y **simplificar** el desarrollo.



# Funciones Integradas

El intérprete Python tiene un número de **funciones integradas** (built-in) las cuales están siempre disponibles.



# Funciones Integradas

Funciones Built-in			
<b>A</b> abs() aiter() all() any() anext() ascii()	<b>E</b> enumerate() eval() exec()	<b>L</b> len() list() locals()	<b>R</b> range() repr() reversed() round()
<b>B</b> bin() bool() breakpoint() bytearray() bytes()	<b>F</b> filter() float() format() frozenset()	<b>M</b> map() max() memoryview() min()	<b>S</b> set() setattr() slice() sorted() staticmethod() str() sum() super()
<b>C</b> callable() chr() classmethod() compile() complex()	<b>G</b> getattr() globals()	<b>N</b> next()	<b>T</b> tuple() type()
<b>D</b> delattr() dict() dir() divmod()	<b>H</b> hasattr() hash() help() hex()	<b>O</b> object() oct() open() ord()	<b>V</b> vars()
	<b>I</b> id() input() int() instance() issubclass() iter()	<b>P</b> pow() print() property()	<b>Z</b> zip()
			__import__()

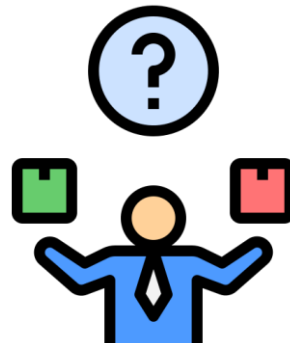


# Parámetros vs Argumentos

La diferencia es muy sutil:

**Parámetro:** los usamos para definir la función.

**Argumento:** a los valores que se utilizan cuando usamos a la función.



# Parámetros vs Argumentos

## Argumentos, Parámetros, y Resultados

```
>>> grande = max('Hola mundo')  
>>> print(grande)  
w
```

'Hola mundo' →  
Argumento

```
def max(inp):  
    blah  
    blah  
    for x in inp:  
        blah  
        blah  
    return 'w'
```

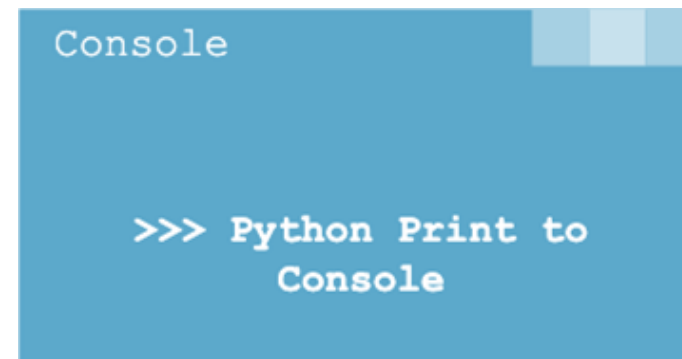
Parámetro

→ 'w'  
↑  
Resultado

# La función integrada `print()`

Esta función permite, **mostrar** o «imprimir», mensajes o texto por pantalla (Consola de comandos).

```
print("Curso de Python")
```



# La función integrada `print()`

## Uso interactivo de HELP

```
✓ 0 s help (print)

Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep:  string inserted between values, default a space.
    end:  string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

# La salida estandar print()



1. Los programas de computadora, están compuestos por instrucciones, también llamadas comandos, estas especifican una tarea o instrucción específica
2. La función **print()**, Es una función integrada del Lenguaje Python, esta, genera una salida estándar a la consola del sistema
3. Es parte de las 69 Funciones Integradas del Sistema y por tanto, no requiere la inclusión de Librerías Externas
4. Para invocar una función dentro de nuestro programa, escribiremos su nombre, seguido de paréntesis y los argumentos que requiera, serán proporcionados de manera posicional o nombrada
5. Las cadenas de texto en Python se escriben acotadas por comillas dobles "" o sencillas ''

# La salida estandar print()

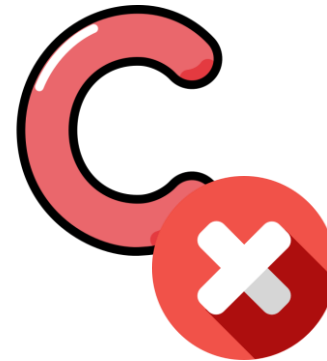


1. En Python, el caracter **backslash (\)** tiene un tratamiento especial, dependiente del caracter inmediato posterior. e.g., **\n** es el comando utilizado para solicitar un salto de línea
2. Los argumentos posicionales son aquellos que deben ser proporcionados a la función, en un orden específico y no requieren ser nombrados de manera explícita
3. Los argumentos nombrados son aquellos que son proporcionados a la función por su nombre, y siempre deberán ser posteriores a la lista de argumentos posicionales
4. Los parámetros **end** y **sep** son utilizados para formatear la salida de la función **print()**, son de tipo nombrado
  1. El parámetro **sep**, determina la separación entre los argumentos de salida
  2. El parámetro **end**, especifica la salida al final de la salida de impresión



# Literales

Un literal se refiere a datos cuyos valores están **determinados** por **el literal** mismo.



Respuesta: **123 es un literal**, y c no lo es.

Se utilizan literales para **codificar datos** y ponerlos dentro del código.

# Literales

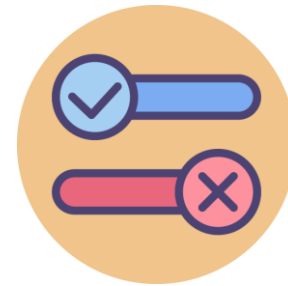
Son aquellos datos que son introducidos en el código del programa **directamente** en forma de valores, o argumentos de una función.





# Literales

Son aquellos datos que son introducidos en el código del programa **directamente** en forma de valores, o argumentos de una función.



# Literales numéricos

Entero `<int>` `<integer>`

Es una representación numérica en base 10 y sin punto decimal, pueden ser positivos o negativos.



# Literales numéricos

Flotante <float>

Es una representación numérica en base 10 **con punto decimal**, pueden ser positivos o negativos, son la representación de un numero **fraccional**



# Legibilidad

Ejemplo: el número once millones ciento once mil ciento once.

Si tomaras ahorita un lápiz en tu mano,  
escribirías el siguiente número:



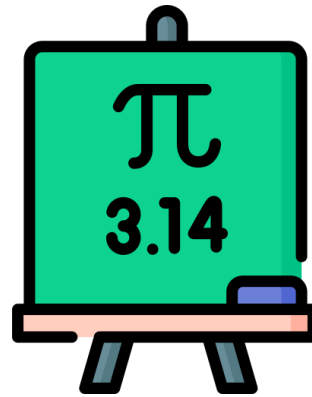
**11,111,111**



**11\_111\_111**

# Constantes

Que no se interrumpe y **persiste** en el estado en que se encuentra, sin variar su intensidad.



Es un número específico o un símbolo al que se le asigna un **valor fijo**.

# Literales tipo Cadena

Cadena `<string>` `<str>`

En general, una cadena de caracteres es una sucesión de **caracteres**.

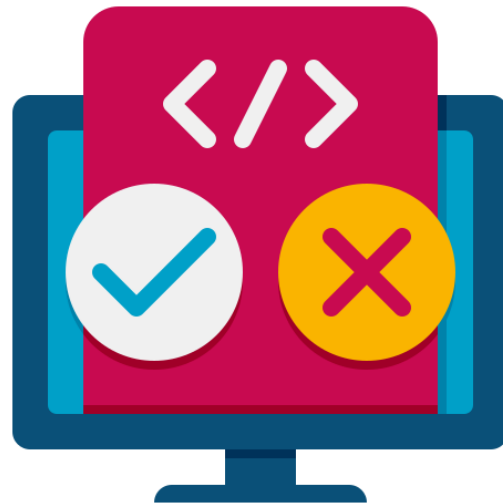


**Las cadenas requieren comillas**, así como los flotantes necesitan punto decimal.

# Booleanos

Booleano **<bool>**

Solo pueden adoptar dos valores **True** o **False** y regularmente son utilizados para evaluar la veracidad de una condición)



# Los operadores básicos

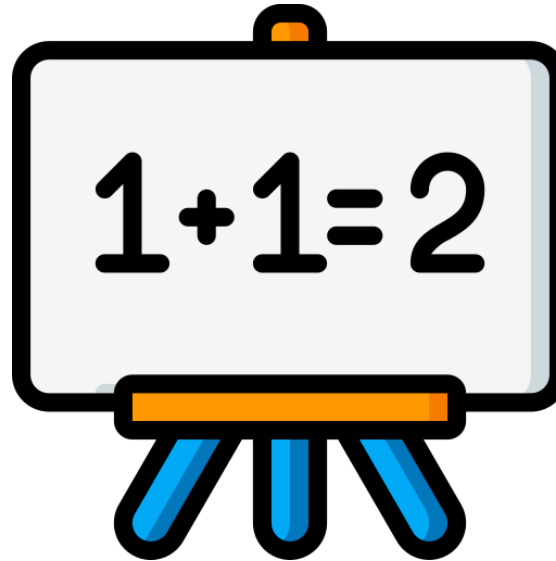
Un **operador** es un símbolo del lenguaje de programación, el cual es capaz de realizar operaciones con los valores.





# Los operadores básicos

Por ejemplo, como en la aritmética, el signo de **+** (más) es un operador el cual es capaz de **sumar** dos números, dando el resultado de la suma.



# Los operadores básicos

Sin embargo, no todos los operadores de Python son tan simples como el signo de más.

**+, -, \*, /, //, %, \*\***

**Recuerda:** Cuando los datos y operadores se **unen**, forman juntos **expresiones**.  
La expresión más sencilla es el literal.

# Operadores Aritméticos: exponenciación

Un signo de \*\* (**doble asterisco**) es un operador de exponenciación (**potencia**).

$$2^{**}3$$

El argumento a la izquierda es la base, el de la derecha, el exponente.

# Operadores Aritméticos: multiplicación

Un símbolo de \* (**asterisco**) es un operador de **multiplicación**.

$$2 * 4$$

# Operadores Aritméticos: división

Un signo de / (**diagonal**) es un operador de **división**.

$$10 / 5$$

El valor **después** de la diagonal es el **dividendo**,  
el valor **antes** de la diagonal es el **divisor**.

## Operadores Aritméticos: división entera

Un signo de // (**doble diagonal**) es un operador de **división**.

$$6 // 4$$

El resultado carece de la parte fraccionaria.

- Ausente (**enteros**)
- Siempre es igual a **cero** (**flotantes**)

Los resultados siempre son **redondeados**.

# Operadores: residuo (módulo)

Un signo de % (**porcentaje**) es un operador de (**residuo**).

**14 % 4**

El resultado de la operación es el **residuo** que queda de la división entera.

•

Elaborado: Mtro. Humberto Acevedo hac106@hotmail.com



# Jerarquía de prioridades

Considera la siguiente expresión:

$$2 + 3 * 5$$

.



# Operadores y sus enlaces

**Enlace de un operador:** determina el orden en que se computan las operaciones de los operadores con la misma prioridad.

**`print(9 % 6 % 2)`**

El resultado de la operación es el **residuo** que queda de la división **entera**.

# Puntos clave

1. Una **expresión** es una combinación de valores: las cuales son evaluadas y dan como resultado un valor, por ejemplo,  $1 + 2$ .
2. Una **expresión** es una combinación de valores (o variables, operadores, llamadas a funciones, aprenderás de ello pronto) las cuales son evaluadas y dan como resultado un valor, por ejemplo,  $1 +$

# Variables



Python ofrece "cajas" (contenedores), estas cajas son llamadas **variables** - el nombre mismo sugiere que el contenido de estos contenedores puede **variar** en casi cualquier forma.

# Variables

¿Cuáles son los componentes o **elementos** de una **variable** en Python?

- Un **nombre**.
- Un **valor** (el contenido del contenedor).



Elaborado por: [fernando.fernandez@unam.mx](mailto:fernando.fernandez@unam.mx)

# Reglas para NOMBRAR

Si se desea **nombrar una variable**:

- El nombre de la variable debe de estar compuesto por MAYÚSCULAS, minúsculas, dígitos, y el carácter \_ (guion bajo).
- El nombre de la variable debe comenzar con una letra.
- El carácter guion bajo es considerado una letra.



# Reglas para NOMBRAR

- Las mayúsculas y minúsculas se tratan de forma distinta (un poco diferente que en el mundo real - *Alicia* y *ALICIA* son el mismo nombre, pero en Python son dos nombres de variable distintos, subsecuentemente, son dos variables diferentes).
- El nombre de las variables no puede ser igual a alguna de las palabras reservadas de Python (se explicará más de esto pronto).



# VARIABLES

## OBTENER EL TIPO DE UNA VARIABLE

Es posible obtener el tipo de una variable a través de la función `type()`

Ejemplo

```
x = 5
```

```
y = "John"
```

```
print(type(x))
```

```
print(type(y))
```

```
<class 'int'>
```

```
<class 'str'>
```

# VARIABLES

## MÚLTIPLES VALORES A MÚLTIPLES VARIABLES

Python permite asignar múltiples valores a múltiples variables en una sola línea de código.

Ejemplo:

```
x, y, z = "Orange", "Banana", "Cherry"
```

```
print(x)
```

```
print(y)
```

```
print(z)
```

Orange

Banana

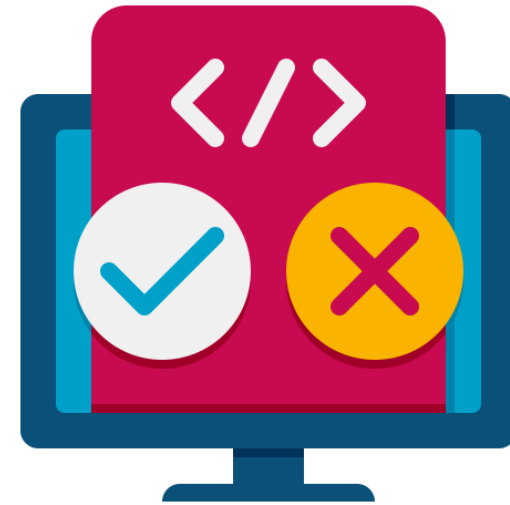
Cherry



# Literales de tipo Boleano

Booleano <bool>

Solo pueden adoptar dos valores **True** o **False** y regularmente son utilizados para evaluar la veracidad de una condición)



# Operadores Lógicos

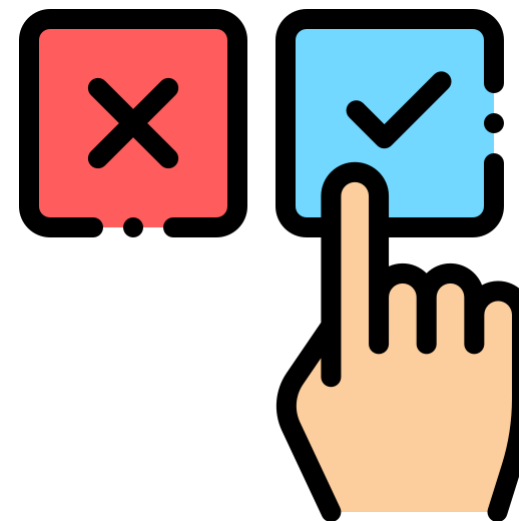
## Comparación

$>$  ,  $>=$  ,  $<$  ,  $<=$  ,  $==$

Las comparaciones también dan como resultado valores booleanos.

# Álgebra Booleana

Rama del álgebra matemática que se enfoca en el estudio y manipulación de expresiones lógicas y proposiciones utilizando operadores booleanos, como AND, OR y NOT.



# Tablas de verdad

**Tabla de verdad**

**Variable o proposición** → **conectivo lógico**

**Estados de la variable**

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

**Fila**

**Columna**

# Operador Lógico AND

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

Indica que es necesario que en todas sus entradas se tenga un estado binario 1 para que la salida otorgue un 1 binario.

# Operador Lógico OR

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

Esta compuerta permite que con cualquiera de sus entradas que este en estado binario 1, su salida pasara a un estado 1 también.

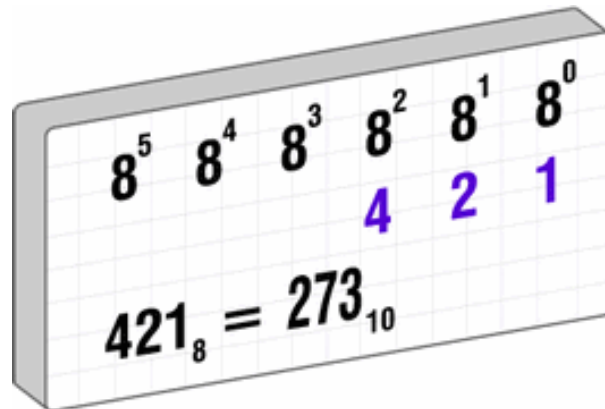
## Función de Entrada con argumento

# input (argumento)

Solicita al usuario que **inserte** algún dato desde la consola.

# Sistema Posicional Decimal, Binario, Octal y Hexadecimal

...	$10^5$	$10^4$	$10^3$	$10^2$	$10^1$	$10^0$		$10^{-1}$	$10^{-2}$	$10^{-3}$	$10^{-4}$	...
...	100000	10000	1000	100	10	1	.	1/10	1/100	1/1000	1/10000	...
...	Centenas de millar	Decenas de millar	Unidades de millar	Centenas	Decenas	Unidades	Punto decimal	Décimas	Centésimas	Milésimas	Diez milésimas	...

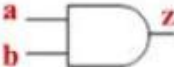
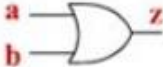
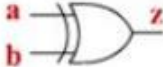
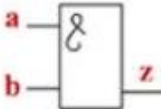
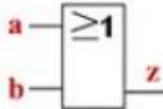
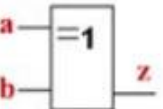
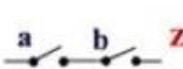
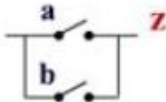
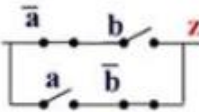


$$16^n \leftarrow \begin{matrix} 16^4 & 16^3 & 16^2 & 16^1 & 16^0 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 65536 & 4096 & 256 & 16 & 1 \end{matrix}$$



# Funciones Lógicas Booleanas

En matemática, electrónica digital e informática, el álgebra de Boole, también llamada álgebra booleana, es una estructura algebraica que esquematiza las operaciones lógicas.

NOMRE	AND - Y	OR - O	XOR O-exclusiva																																													
SÍMBOLO																																																
SÍMBOLO																																																
TABLA DE VERDAD	<table><tr><th>a</th><th>b</th><th>z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	a	b	z	0	0	0	0	1	0	1	0	0	1	1	1	<table><tr><th>a</th><th>b</th><th>z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	a	b	z	0	0	0	0	1	1	1	0	1	1	1	1	<table><tr><th>a</th><th>b</th><th>z</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	a	b	z	0	0	0	0	1	1	1	0	1	1	1	0
a	b	z																																														
0	0	0																																														
0	1	0																																														
1	0	0																																														
1	1	1																																														
a	b	z																																														
0	0	0																																														
0	1	1																																														
1	0	1																																														
1	1	1																																														
a	b	z																																														
0	0	0																																														
0	1	1																																														
1	0	1																																														
1	1	0																																														
EQUIVALENTE EN CONTACTOS																																																
AXIOMA	$z = a \cdot b$	$z = a + b$	$z = \bar{a} \cdot b + a \cdot \bar{b}$																																													

# Literales 1



1. Se denominan valores literales a aquellos datos que son introducidos en el código del programa directamente en forma de valores, o argumentos de una función
2. El sistema binario es un sistema de numeración que utiliza 2 símbolos 0 (cero) y 1 (uno), denominados dígitos binarios

El sistema de numeración octal es un sistema de numeración en base 8, Otro modo de manejar números binarios es con el uso del sistema de numeración hexadecimal. Este sistema es de base 16, y utiliza los dígitos 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F.

3. Es una representación numérica en base 10 sin punto decimal, pueden ser positivos o negativos
4. Es una representación numérica en base 10 con punto decimal, pueden ser positivos o negativos, son la representación de un numero fraccional
5. El uso de caracteres de escape \Backslash, así como el uso combinado de comillas, permiten generar literales o argumentos complejos, tales como: |
6. Las literales de tipo bool o booleano, solo pueden adoptar dos valores **True** o **False** y regularmente son utilizados para evaluar la veracidad de una condición)

# Laboratorio

## Objetivo:

- Familiarizarse con el concepto de variables y trabajar con ellas.
- Realizar operaciones básicas y conversiones.
- Experimentar con el código de Python

## Escenario:

Millas y kilómetros son unidades de longitud o distancia.

Teniendo en mente que 1 milla equivale aproximadamente a 1.61 kilómetros, complementa el programa en el editor para que convierta de:

Millas a kilómetros.

Kilómetros a millas.

```
kilometers = 12.25  
miles = 7.38
```

```
miles_to_kilometers = ###  
kilometers_to_miles = ###
```

```
print(miles, "millas son", round(miles_to_kilometers, 2), "kilómetros")  
print(kilometers, "kilómetros son", round(kilometers_to_miles, 2), "millas")
```

Elaborado: Mtro. Humberto Acevedo hac106@hotmail.com



## RETO DE TIEMPO

```
x = 1
```

```
y = 0
```

```
z = ((x == y) and (x == y)) or not (x == y)
```

```
print(not(z))
```

# RETO DE TIEMPO

```
x = 8
```

```
y = 23
```

```
a = x & y
```

```
b = x | y
```

```
c = ~x
```

```
d = x ^ 5
```

```
e = x >> 2
```

```
f = x << 2
```

```
print(a, b, c, d, e, f)
```