



universidad
cenfotec_
La U de la informática

Principios de Programación.

Estructuras de construcción.
Expresiones lógicas complejas.

Objetivos

- Mostrar el funcionamiento los operadores lógicos para la construcción de expresiones lógicas complejas.
- Identificar la estructura de una expresión lógica simple y compleja.
- Presentar los operadores lógicos.
- Construir expresiones lógicas complejas para plantear condiciones.

Repaso: Operadores relacionales

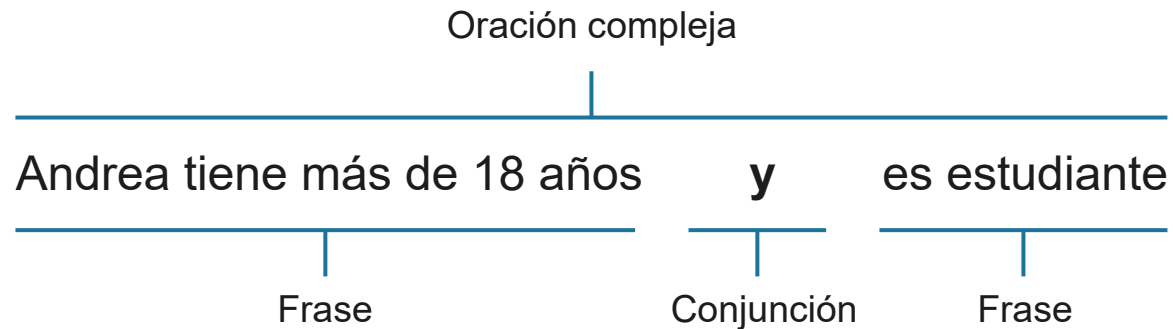
Los **operadores relacionales** son símbolos que permiten comparar dos valores dados. El resultado será siempre *verdadero* o *falso*, dependiendo del resultado de la comparación. Los operadores relacionales son los siguientes:

Operadores relacionales		
Símbolo	Lectura	Ejemplo
>	Mayor que	numero1 > numero2
<	Menor que	numero1 < numero2
>=	Mayor o igual que	numero1 >= numero2
<=	Menor o igual que	numero1 <= numero2
==	Igual a	numero1 == numero2
!=	Diferente a	numero1 != numero2

Operadores lógicos y expresiones lógicas complejas

Operadores lógicos

A la hora de **comunicarse** es muy común que para transmitir el mensaje de forma completa y entendible sea necesario usar más de una idea. Para esto se unen dos o más frases simples utilizando **conjunciones**, lo que genera un nexo entre ambas frases dando como resultado una oración más compleja. Por ejemplo:



Al leer la oración completa, se puede determinar si esta es **verdadera** o **falsa**. Ya que es posible que Andrea realmente no tenga más de 18 años o que ya no sea estudiante.

Operadores lógicos

Al igual que es posible unir frases en un lenguaje natural utilizando conjunciones, es posible unir dos **expresiones lógicas** en los lenguajes de programación utilizando **operadores lógicos**.

Los **operadores lógicos** son símbolos que permiten unir dos expresiones simples. Al hacerlo el operador evalúa ambas expresiones y retorna un valor booleano (*Verdadero o Falso*). Esto permite evaluar más de una condición a la hora de decidir el flujo a seguir dentro de un programa.

Operadores lógicos	
Símbolo	Lectura
&&	AND
	OR
!	NOT

El operador AND

Este se puede leer también como “Y”. Al unir dos expresiones mediante el operador **AND** si ambas son verdaderas, se obtiene como resultado *Verdadero*. En caso de que cualquiera de las dos expresiones sea falsa, se obtiene *Falso* como resultado.

Expresión A	Expresión B	A and B
verdadero	verdadero	verdadero
verdadero	falso	falso
falso	verdadero	falso
falso	falso	falso

El operador OR

Este se puede leer también como “O”. Al unir dos expresiones mediante el operador **OR** si cualquiera de las dos expresiones es verdadera, se obtiene como resultado *Verdadero*. Para obtener un *Falso* como resultado, ambas expresiones deben ser falsas.

Expresión A	Expresión B	A or B
verdadero	verdadero	verdadero
verdadero	falso	verdadero
falso	verdadero	verdadero
falso	falso	falso

El operador NOT

Este se puede leer también como “NO”. Este operador se utiliza para obtener el valor opuesto de una expresión. Por ejemplo, si la Expresión A tiene como valor *Verdadero*, se obtiene como resultado *Falso*, si la expresión tiene como valor *Falso* se obtiene como resultado *Verdadero*.

Expresión A	not A
verdadero	falso
falso	verdadero

Expresiones lógicas complejas

Una **expresión lógica compleja** es aquella que mediante el uso de un operador lógico une dos expresiones lógicas simples, para obtener un único resultado.

En la siguiente tabla se puede ver como una **expresión lógica simple** toma dos valores, los cuales los compara utilizando un operador relacional. En este caso, el resultado será *Verdadero* o *Falso* dependiendo del valor de la variable edad.

Expresiones lógicas		
Simple	Sintaxis	<code>valor1 operador_relacional valor2</code>
	Ejemplo 1	<code>edad >= 18</code>
	Sintaxis	<code>valor3 operador_relacional valor4</code>
	Ejemplo 2	<code>es_estudiante == True</code>

Expresiones lógicas complejas

En una **expresión lógica compleja**, se toman dos expresiones simples y se evalúa el resultado de ambas utilizando un operador lógico. El resultado de la expresión compleja depende del operador utilizado y de los valores obtenidos de las expresiones simples.

Expresiones lógicas		
Compleja	Sintaxis	<code>expresiónSimpleA operador_lógico expresiónSimpleB</code>
	Ejemplo	<code>edad >= 18 and es_estudiante == True</code>

Ejemplo 1 de expresiones lógicas complejas

En un restaurante se crea un sistema de recompensas para **clientes frecuentes**, en el cual, si se han hecho 6 o más compras anteriores y la nueva compra es igual o mayor a 10.000 colones, se le aplica un descuento de 35% sobre el total de la factura.

Por lo que, para recibir el descuento se deben cumplir dos condiciones:

1. Ha realizado 6 o más compras anteriormente en el mismo restaurante.
2. La nueva compra es mayor o igual a 10.000 colones.



Ejemplo 1 de expresiones lógicas complejas

Una vez se realiza el análisis del problema, se deben generar las variables necesarias para dar solución al algoritmo:

ENTRADAS			
Descripción	Notación		Ejemplo
	Nombre	Tipo de dato	
Variable que recibe la cantidad de compras realizadas por el cliente.	compras_realizadas	Entero	7
Variable que recibe el costo total de la factura a pagar.	costo_total	Float	15800

Ejemplo 1 de expresiones lógicas complejas

Las expresiones simples a tomar en cuenta a la hora de aplicar el descuento son las siguientes:

1. Las compras realizadas son mayores o iguales a 6. 
2. La compra es mayor o igual a 10000 colones. 

```
compras_realizadas >= 6  
costo_total >= 10000
```


Para aplicar el descuento es necesario que se cumplan **ambas** condiciones, por lo que para este caso se debe utilizar el operador **AND**. El resultado de unir ambas expresiones mediante el operador sería la siguiente:

```
( compras_realizadas >= 6 and costo_total >= 10000 )
```

Ejemplo 1 de expresiones lógicas complejas

En el siguiente código puede ver cómo el valor de la variable **costo_final** se le aplicará el descuento únicamente si se cumplen las dos condiciones planteadas anteriormente.

compras.py

```
1     compras_realizadas = 0
2     costo_total = 0
3     costo_final = 0
4     compras_realizadas = int(input('Digite la cantidad de compras realizadas por el cliente: '))
5     costo_total = float(input('Digite el costo total de la compra: '))
6     if (compras_realizadas >= 6 and costo_total >= 10000) : 
7         #Se aplica el descuento del 35% sobre el costo total de la compra.
8         costo_final = costo_total - (costo_total * 0.35)
9     else:
10         costo_final = costo_total
11     print('El costo final de la compra es: ' , costo_final)
```

Ejemplo 2 de expresiones lógicas complejas

Un **programador** desea crear un algoritmo para decidir si va o no al cine de acuerdo a ciertas condiciones. Las cuales son que la fecha de la función debe ser después del día 15 del mes o que el cine tiene promoción en las entradas. Además, la valoración de la película debe ser mayor a 4 estrellas o que no sea para mayores de edad.

Las **condiciones lógicas simples** en este caso son las siguientes:

1. La función es después del día 15 del mes.
2. El cine tiene promoción de entradas.
3. La valoración de la película es mayor a 4.
4. La película no es para mayores de edad.

Ejemplo 2 de expresiones lógicas complejas

Una vez se realiza el análisis del problema, se deben generar las variables necesarias para dar solución al algoritmo:

ENTRADAS			
Descripción	Notación		Ejemplo
	Nombre	Tipo de dato	
Variable que almacena el día del mes.	dia_del_mes	Entero	14
Variable que almacena un valor booleano si el cine tiene promoción un día concreto o no.	hay_promocion	Booleano	False
Variable que almacena la valoración que tiene la película.	puntuacion_pelicula	Float	4.5
Variable que almacena un valor booleano si película es para mayores de edad	para_mayores_de_edad	Booleano	True

Ejemplo 2 de expresiones lógicas complejas

Las expresiones simples a tomar en cuenta a la hora de decidir si el usuario va al cine son:

1. La función es después del día 15 del mes.	→	<code>dia_del_mes > 15</code>
2. El cine tiene promoción de entradas	→	<code>hay_promocion == True</code>
2. La valoración de la película es mayor a 4.	→	<code>puntuacion_pelicula > 4</code>
2. La película no es para mayores de edad.	→	<code>not para_mayores_de_edad</code>

Para decidir si se va al cine, se debe cumplir ya sea que la función sea después del día 15 del mes **o** el cine tiene promoción en las entradas. **Y** además, se debe cumplir que la valoración es mayor a 4 **o** que la película no sea para mayores de edad.

```
(dia_del_mes > 15 or hay_promocion) and (puntuacion_pelicula > 4 or not para_mayores_de_edad)
```

Ejemplo 2 de expresiones lógicas complejas

A continuación, se muestra la solución paso a paso de la expresión compleja, tomando como ejemplo los valores de las variables de entrada, las cuales se encuentran en la tabla inferior. Se toma el primer paréntesis, en donde se da el valor de **14** a la variable **dia_del_mes** y el valor **False** a la variable **hay_promocion**.

El valor 14 de **dia_del_mes** no es mayor a 15, por lo que da como resultado **False** y se da el valor **False** para **hay_promocion**. Y tomando como referencia el resultado de **False OR False**, se concluye que la primera expresión tiene como resultado **False**.

$(\text{dia_del_mes} > 15 \text{ or } \text{hay_promocion}) \longrightarrow \text{False}$

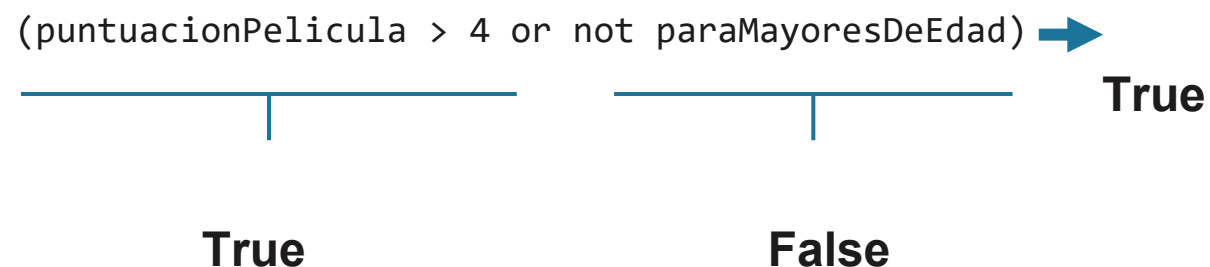
False **False**

VARIABLES	
Nombre	Ejemplo
dia_del_mes	14
hay_promocion	False
puntuacion_pelicula	4.5
para_mayores_de_edad	True

Ejemplo 2 de expresiones lógicas complejas

En el segundo paréntesis, se le da el valor de **4.5** a la variable **puntuacion_pelicula** y a la variable **para_mayores_de_edad** el valor de **True**.

El valor de 4.5 de la puntuación es mayor a 4, por lo que da como resultado **True** y al utilizar el operador **NOT** sobre el valor **True** de la variable **para_mayores_de_edad** se obtiene el valor **False**. Por lo que solucionar la expresión **True OR False** se obtiene como resultado **True**.

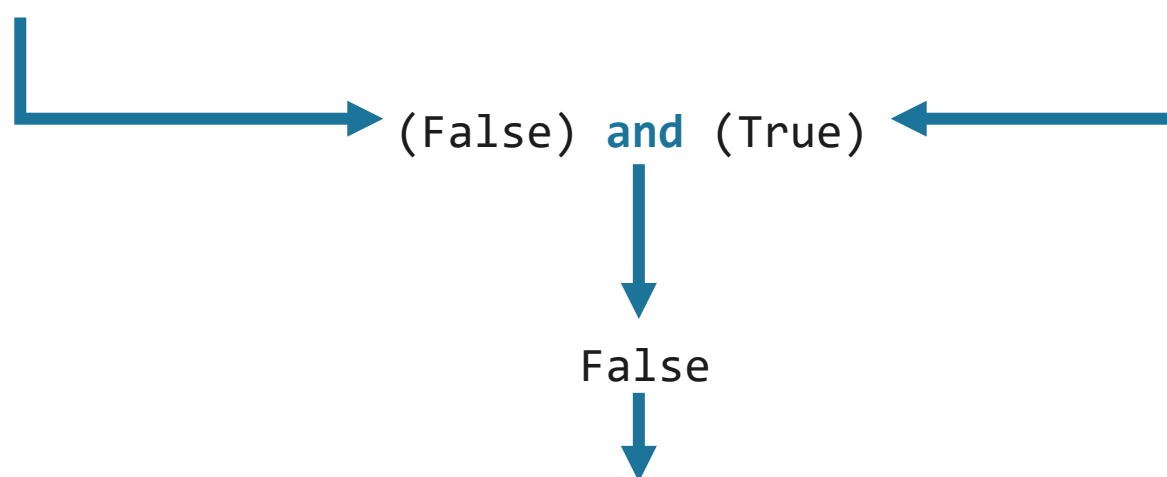


VARIABLES	
Nombre	Ejemplo
dia_del_mes	14
hay_promocion	False
puntuacion_pelicula	4.5
para_mayores_de_edad	True

Ejemplo 2 de expresiones lógicas complejas

Finalmente, una vez resueltos ambos paréntesis, se debe resolver el resultado de los dos utilizando el operador **AND**. Al ejecutar **False AND True** obtiene el resultado de **False**, como se muestra a continuación:

```
(dia_del_mes > 15 or hay_promocion) and (puntuacion_pelicula > 4 or not para_mayores_de_edad)
```



Se concluye que el programador no irá al cine en esta ocasión.

Ejemplo 2 de expresiones lógicas complejas

En el siguiente código puede ver cómo el usuario irá al cine únicamente si el resultado final de la expresión es **True**. Con los valores dados anteriormente, se obtuvo como resultado **false**, por lo que se mostrará el mensaje: *“Hoy no voy al cine”*.

cine.py

```
1     dia_del_mes = 14
2     hay_promocion = False
3     puntuacion_pelicula = 4.5
4     para_mayores_de_edad = True
5
6     if ((dia_del_mes > 15 or hay_promocion) and (puntuacion_pelicula > 4 or not para_mayores_de_edad)):
7         print('Hoy voy al cine.')
8     else:
9         print('Hoy no voy al cine.')
```

Resumen

Como se pudo observar en los ejemplos anteriores, utilizando las expresiones lógicas complejas se puedan definir **múltiples condiciones** para determinar si se llevan a cabo un conjunto de instrucciones o no. Es importante que recuerde en qué momentos se utiliza cada operador y cuál es el resultado que se obtiene con los valores que se proveen.

La siguiente tabla le muestra los resultados obtenidos con todas las posibles combinaciones de valores:

Expresión A	Expresión B	A B	A && B
verdadero	verdadero	verdadero	verdadero
verdadero	falso	verdadero	falso
falso	verdadero	verdadero	falso
falso	falso	falso	falso

Expresión A	!A
verdadero	falso
falso	verdadero



universidad
cenfotec_
La U de la informática