Universidad Andrés Bello®

Unab Online

Lógica

León, R. (2021). *Lógica* [apunte].

Chile. UNAB

LÓGICA

1. Introducción

Anteriormente, habíamos mencionado que un operador es un elemento del lenguaje que representa una acción sobre los datos, como una suma o una concatenación de palabras. Además de los operadores mencionados anteriormente, existen los operadores lógicos, los cuales nos permiten comparar valores lógicos (booleanos) los cuales son el resultado de ciertas expresiones, denominadas *expresiones lógicas.*

2. Operadores lógicos

Los operadores lógicos se aplican a valores lógicos y son los siguientes:

Tabla 1
Operadores lógicos

Comparación lógica	Operador
Y	and
O	or
Negación	not

La regla para cada uno de los operadores aplica de forma similar que la lógica clásica. Por ejemplo, si queremos saber si un número entero x se encuentra en un intervalo determinado [a,b], debo comprobar dos situaciones: (i) si x es mayor o igual al valor de a, **y** (ii) si x es menor o igual al valor de b. Con los operadores anteriores solamente podemos escribir cada situación por separado: (i) x >= a y (ii) x <= b. La única posibilidad de que el valor de x esté en el intervalo es que la situación (i) sea verdadera y la situación (ii) también. Es decir, True y True, lo que en lógica resulta ser True (verdadero). Y eso es lo que realiza el operador *and* evalúa el valor lógico de cada expresión y devuelve un valor lógico dependiendo del resultado, en este ejemplo, devolverá True solamente si ambas expresiones son verdaderas. En la Figura 1, se puede observar distintos casos para el ejemplo.

Figura 1
Uso de operadores matemáticos.

Programa

`a,b = -4,3`
 	
`x = 1`
`esta = (x >= a) and (x <= b)`
`print(esta)`
 	
`x = -6`
`esta = (x >= a) and (x <= b)`
`print(esta)`
 	
`x = 5`
`esta = (x >= a) and (x <= b)`
`print(esta)`

Salida del programa
True False False

A continuación, se muestra una tabla con los valores lógicos resultantes al aplicar los operadores *and, or* y *not*.

Tabla 2
Valores lógicos

and	**or**	**Not**
True and True = True	True or True = True	not True = False
True and False = False	True or False = True	not False = True
False and True = False	False or True = True	
False and False = False	False or False = False	

En la Figura 2, se observa el uso de los distintos operadores lógicos.

Figura 2
Uso de los distintos operadores lógicos.

Programa

```
exp_1 = ("auto" == "moto")
exp_2 = ("casa" != "hogar")
exp_3 = (3 < 99)
exp_4 = ("z" > "g")

#Utilizando un operador logico
print(exp_1 and exp_2)
print(exp_1 or exp_2)
print(not exp_3)

#Utilizando mas de un operador logico
print(exp_1 and exp2_ and exp_3)
print(exp_1 and (exp_2 or exp_3))
print((exp_1 or exp_2) or (exp_2 and exp_3) or not exp_4)
```

Salida del programa

```
False
True
False
False
False
True
```


3. Precedencia de operadores

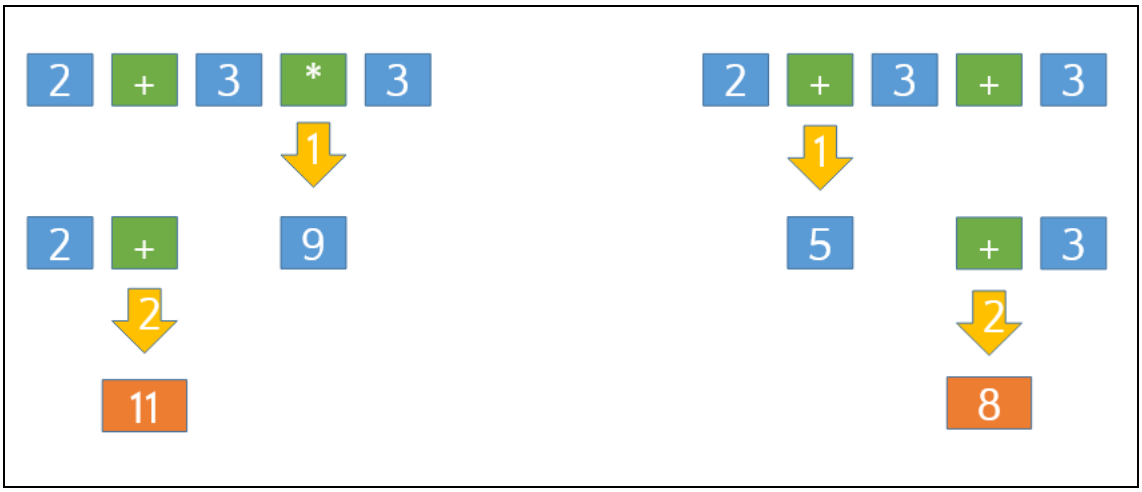
Todos los operadores (matemáticos, de comparación y lógicos) pueden combinarse entre sí. Cuando comenzamos a combinar operadores, cobra relevancia el concepto de *precedencia*.

La precedencia es el orden en el que los operadores se ejecutan. La precedencia es una característica propia del operador que le permite al computador decidir cuál operación ejecutar primero. Cuando tenemos una expresión que incluye varios operadores, cada uno de ellos se va ejecutando de acuerdo con su precedencia,

reemplazándose el operador y sus operandos por el resultado de la operación. Este reemplazo es lo que permite combinar operadores. Cuando los operadores tienen la misma precedencia se ejecutan en el orden en el que aparecen en la expresión desde izquierda a derecha.

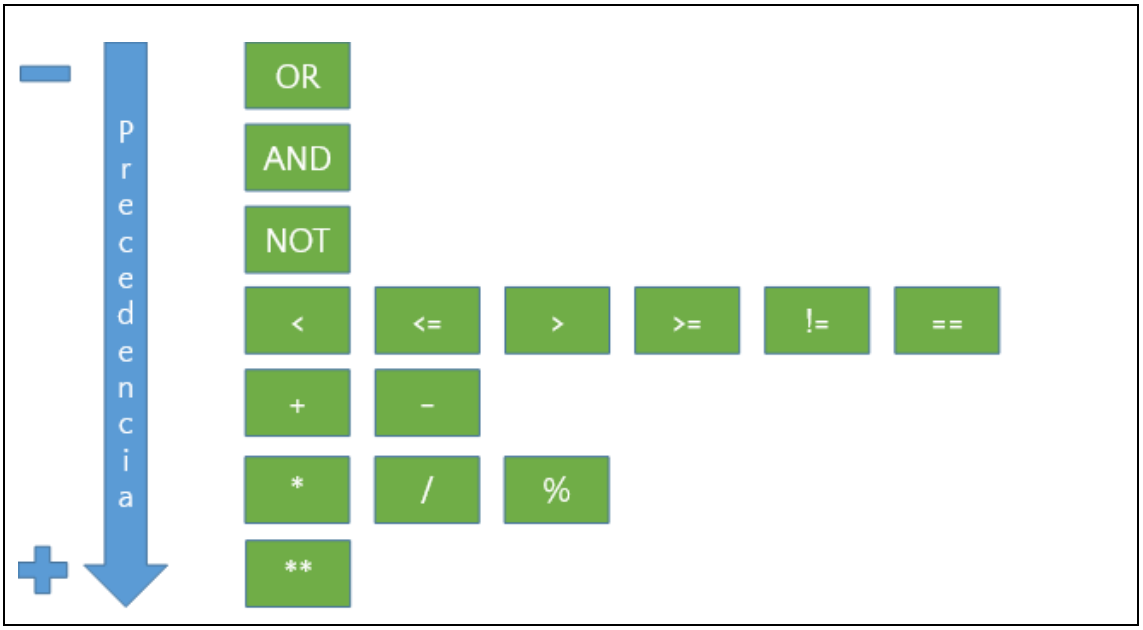
Se puede observar un ejemplo en la Figura 3.

Figura 3
Ejemplo de precedencia.



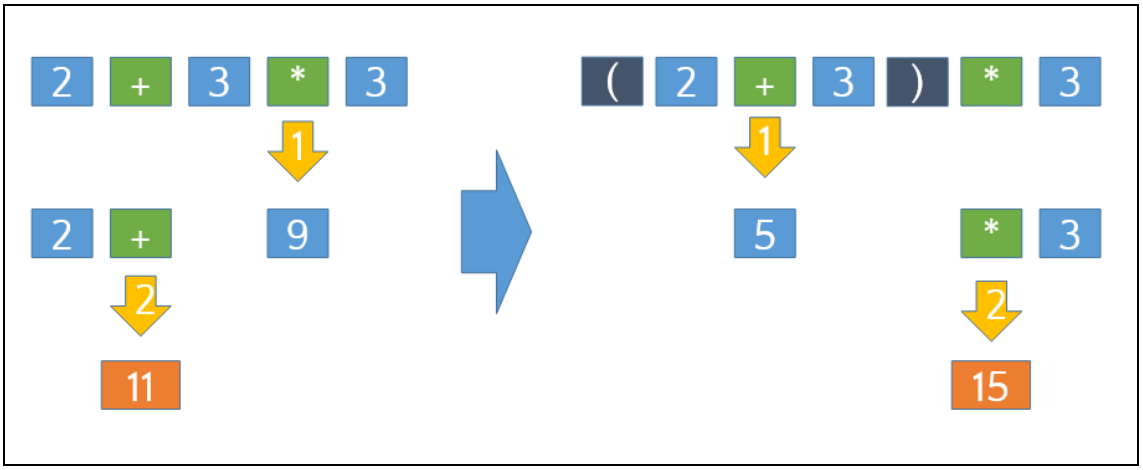
En la Figura 4 se puede observar el orden de precedencia, es decir, mientras el operando tenga mayor precedencia se ejecuta primero.

Figura 4
Orden de precedencia.



En la Figura 5 se puede observar un ejemplo donde se aplica el orden de precedencia. Es importante tener claro que, si se quiere alterar la precedencia, se pueden utilizar los paréntesis para tal efecto.

Figura 5
Ejemplo de orden de precedencia.



"Para complementar la información y contenidos presentados, puedes dirigirte a":

Bibliografía

1. **Downey, A. (2013). Think Python. Green Tea Press.** Accesado el 2 de diciembre de 2019 en https://greenteapress.com/wp/think-python-2e/

2. **Python software foundation, Python v3 Documentation.** Accesado el 2 de diciembre de 2019 en https://docs.python.org/3/

3. Marzal Varó, A., Gracia Luengo, I., & García Sevilla, P. (2014). Introducción a la programación con Python 3. Universitat Jaume I. https://doi.org/10.6035/sapientia93 (Capítulo 1, secciones 1.1 y 1.2, páginas: 11 - 16)