Operadores Lógicos

Juan Felipe Acevedo Pérez

Monitor Junior

UIFCE
UNIDAD DE INFORMÁTICA

Curso Libre

Correo: uniic_bog@unal.edu.co
Tel: 3165000 ***Ext:*** 12301

Facultad de Ciencias Económicas
Unidad de Informática
Sede Bogotá

UNIVERSIDAD NACIONAL DE COLOMBIA

Contenido

Operadores Lógicos

Sentencias Condicionales

1

Operadores Lógicos

Correo: uniic_bog@unal.edu.co
Teléfono: 3165000 **ext** 12301

UIFCE
UNIDAD DE INFORMÁTICA

Valores booleanos

- Una variable de tipo booleano sólo puede tener dos valores: True (cierto) y False (falso). Estos valores son especialmente importantes para las expresiones condicionales y los bucles.

Operadores lógicos

- Estos son los distintos tipos de operadores con los que podemos trabajar con valores booleanos, los llamados operadores lógicos o condicionales:

Operador	Nombre	Ejemplo
and	Devuelve True si ambos elementos son True	True and True = True
or	Devuelve True si al menos un elemento es True	True or False = True
not	Devuelve el contrario, True si es Falso y viceversa	not True = False

AND	True	False
True	True	False
False	False	False

OR	True	False
True	True	True
False	True	False

Correo: uniic_bog@unal.edu.co
Teléfono: 3165000 **ext** 12301

Operadores en Python

OPERADOR	DESCRIPCIÓN	USO
and	Devuelve True si ambos operandos son True	a and b
or	Devuelve True si alguno de los operandos es True	a or b
not	Devuelve True si alguno de los operandos False	not a

Operadores en R

Operador	Comparación	Ejemplo	Resultado
`x | y`	x Ó y es verdadero	`TRUE | FALSE`	`TRUE`
`x & y`	x Y y son verdaderos	`TRUE & FALSE`	`FALSE`
`!x`	x no es verdadero (negación)	`!TRUE`	`FALSE`
`isTRUE(x)`	x es verdadero (afirmación)	`isTRUE(TRUE)`	`TRUE`

2

Sentencias Condicionales

Correo: uniic_bog@unal.edu.co
Teléfono: 3165000 **ext** 12301

UIFCE
UNIDAD DE INFORMÁTICA

Sentencias condicionales

- Si un programa no fuera más que una lista de órdenes a ejecutar de forma secuencial, una por una, no tendría mucha utilidad. Los condicionales nos permiten comprobar condiciones y hacer que nuestro programa se comporte de una forma u otra, que ejecute un fragmento de código u otro, dependiendo de esta condición.

IF

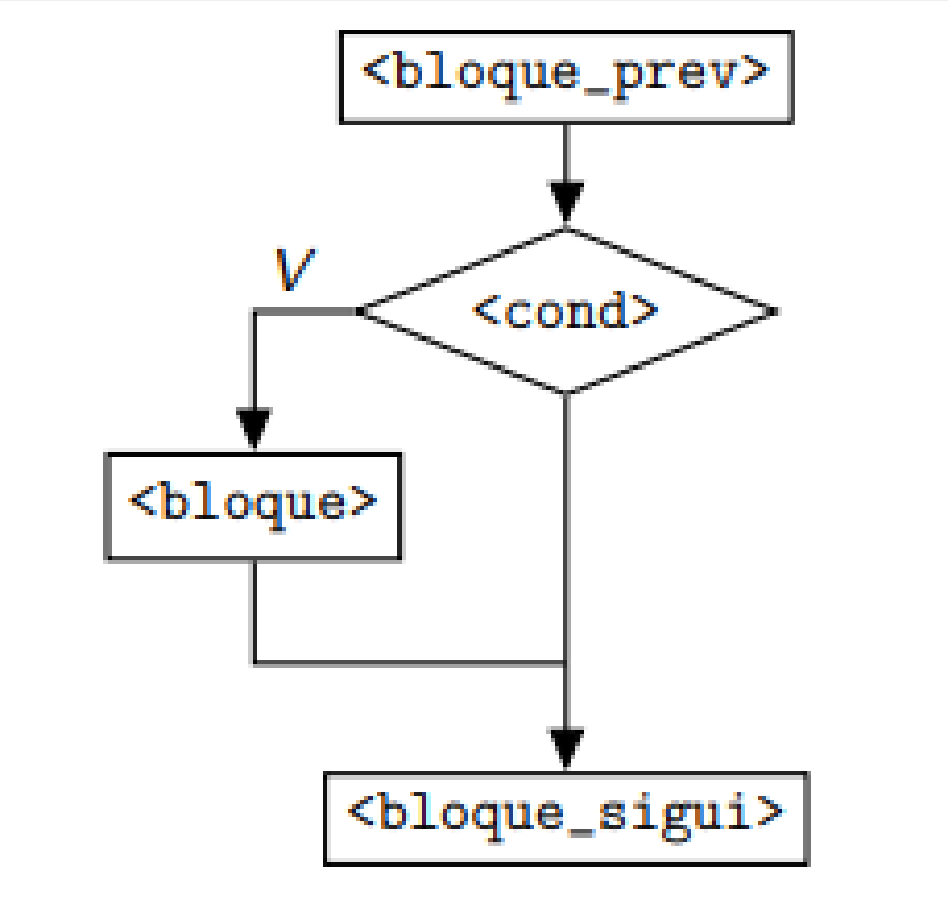
- La forma más simple de un estamento condicional es un if (del inglés si) seguido de la condición a evaluar, dos puntos (:) y en la siguiente línea e indentado, el código a ejecutar en caso de que se cumpla dicha condición.

Python

```
<bloque_prev>
if <cond>:
    <bloque>
<bloque_sigui>
```

R

```
if(<condicion>) {
  ## bloque de código
}
```



Correo: uniic_bog@unal.edu.co
Teléfono: 3165000 **ext** 12301

IF … Else

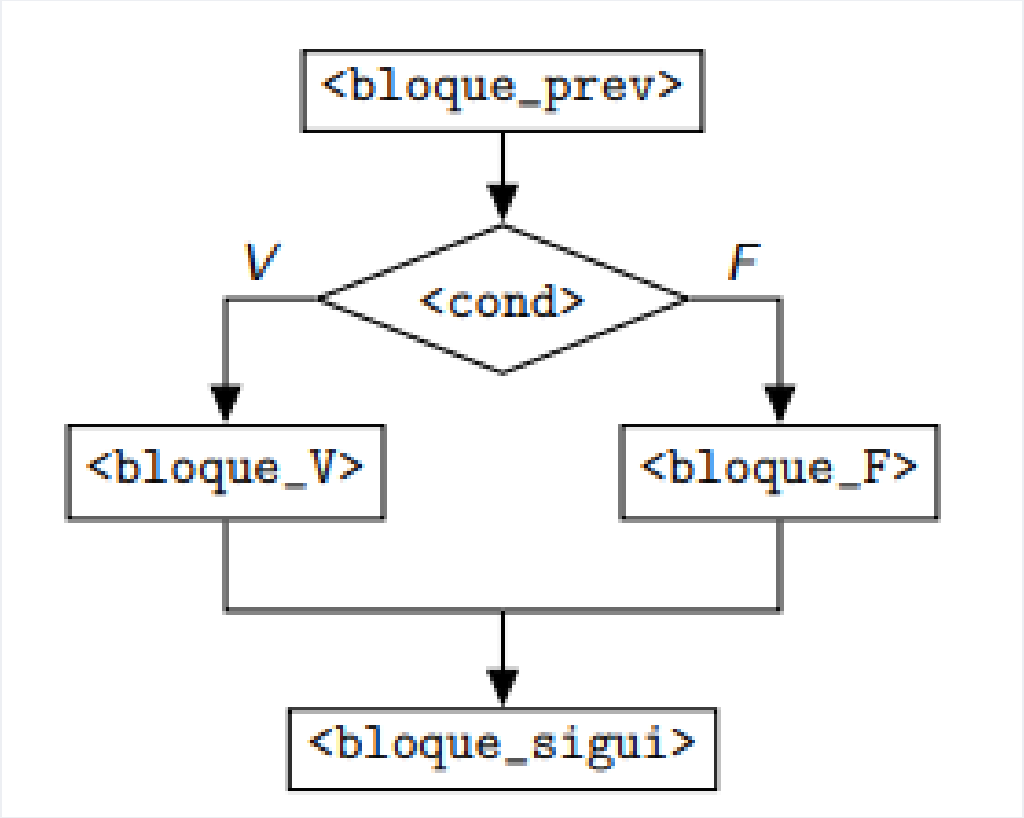
- ¿Qué haríamos si quisiéramos que se ejecutaran unas ciertas órdenes en el caso de que la condición no se cumpliera? Sin duda podríamos añadir otro if que tuviera como condición la negación del primero, pero el condicional tiene una segunda construcción mucho más útil:

Python

```
<bloque_prev>
if <cond>:
    <bloque_V>
else:
    <bloque_F>
<bloque_sigui>
```

R

```
## Continue with rest of code
if(<condicion>) {
  ## bloque de código
} else {
  ## otro bloque de código
}
```



Correo: uniic_bog@unal.edu.co
Teléfono: 3165000 **ext** 12301

IF … Elif … Else

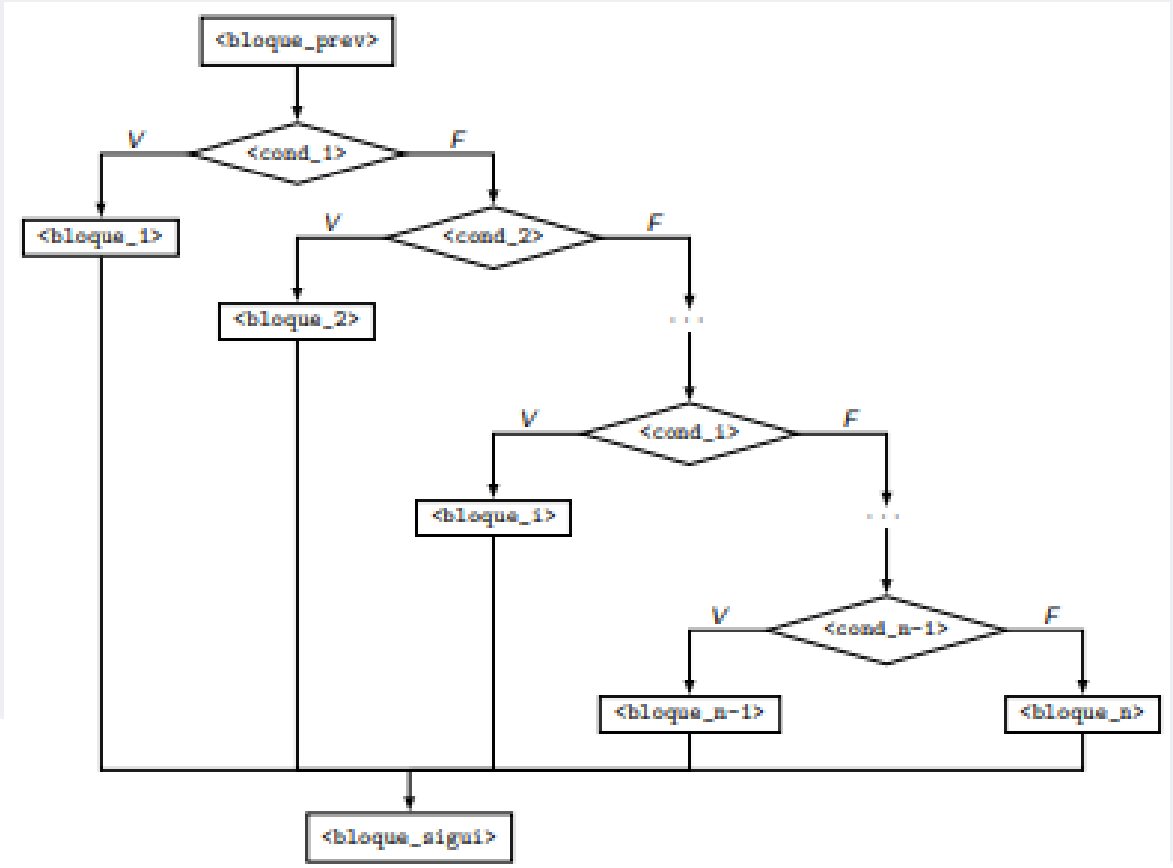
- elif es una contracción de else if, por lo tanto elif numero > 0 puede leerse como “si no, si numero es mayor que 0”. Es decir, primero se evalúa la condición del if. Si es cierta, se ejecuta su código y se continúa ejecutando el código posterior al condicional; si no se cumple, se evalúa la condición del elif.

Python

```
<bloque_prev>
if <cond_1>:
   <bloque_1>
elif <cond_2>:
   <bloque_2>
  ...
elif <cond_i>:
   <bloque_i>
  ...
elif <cond_n-1>:
   <bloque_n-1>
else:
   <bloque_n>
<bloque_sigui>
```

R

```
if(<condition1>) {
  ## bloque de código
} else if(<condicion2>) {
  ## otro bloque de código
} else {
  ## otro bloque de código
}
```



Correo: uniic_bog@unal.edu.co
Teléfono: 3165000 **ext** 12301