

Nivel 2 Introduccion a Programacion

Tomas Rodriguez - 202212868

February 2022

1 Booleanos y sus Operadores

Los tipos de dato booleanos se caracterizan por ser representados como un *True* y *False* o en el sistema binario como 1, 0. Este es solo otro tipo de dato mas, por ende, este tambien posee expresiones, se puede almacenar en variables y posee funciones.

1.1 Operadores relacionales

Son aquellos operadores que relacionan valores, variables, parametros, llamados de funciones u otras expresiones entre si.

OPERADORES RELACIONALES		
Es igual que	<code>==</code>	<code>x == y</code> es True si x es igual a y
Es diferente de	<code>!=</code>	<code>x != y</code> es True si x es diferente de y
Es menor que	<code><</code>	<code>x < y</code> es True si x es menor que y
Es mayor que	<code>></code>	<code>x > y</code> es True si x es mayor que y
Es menor o igual que	<code><=</code>	<code>x <= y</code> es True si x es menor o igual que y
Es mayor o igual que	<code>>=</code>	<code>x >= y</code> es True si x es mayor o igual que y


Figure 1: Operadores Relacionales

Existen en python tambien otros operadores, los **operadores de identidad**.

- **is** y **is not**: sirven para saber si son iguales o son el mismo, ademas en OOP tienen muchos usos.

1.2 Operadores logicos

Permiten describir situaciones mas complejas que los racionales y se dan con valores, variables, parametros, llamados de funciones de retorno booleano.

OPERADORES LÓGICOS		
Operador	Se lee como	
and (conjunción)	y	La semántica (significado) de estos operadores es similar a su significado en español. Por ejemplo: $x > 0$ and $x < 10$ es verdadero (True) solo si x es mayor que 0 y al mismo tiempo, x es menor que 10
or (disyunción)	o	
not (negación)	no	

operando1 and operando2	Es cierto, si ambos operandos son verdaderos
operando1 or operando2	Es cierto, si cualquiera de los dos operandos es verdadero
not operando1	Es cierto, si el operando es falso

Figure 2: Operadores Logicos

2 Tablas de Verdad y Algebra Booleana

Las tablas de verdad son aquellas que nos permiten conocer el resultado entre dos operandos entre los operadores logicos posibles.

a	b	a and b
True	True	True
True	False	False
False	False	False
False	True	False

Table 1: Tabla Verdad AND

a	b	a or b
True	True	True
True	False	True
False	False	False
False	True	True

Table 2: Tabla Verdad OR

a	Not a
True	False
False	True

Table 3: Caption

Ahora bien, el algebra booleana es un conjunto de reglas que se usa para simplificar las expresiones.

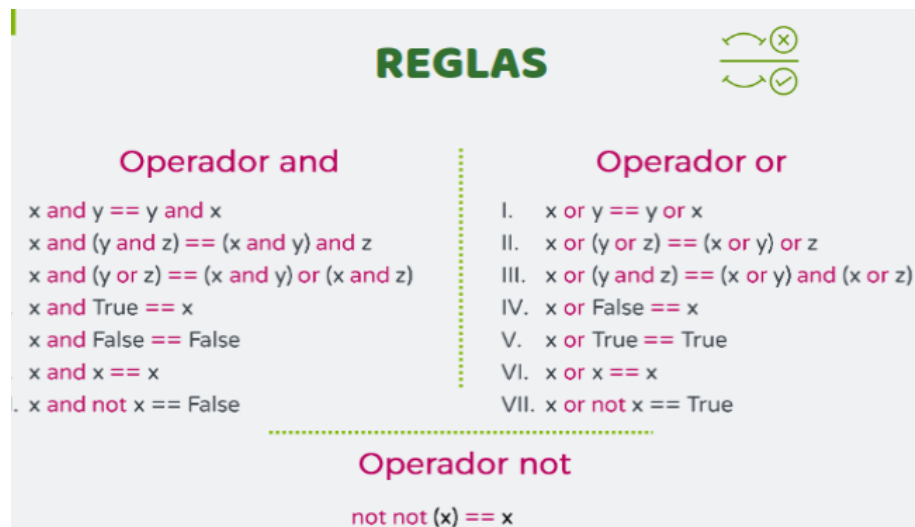


Figure 3: Reglas Algebra Booleana

3 Condicionales

Se usan cuando se debe tomar una desicion teniendo en cuenta diferentes casos y de acuerdo a estos la desicion es diferente.

```
1 if ExpresionBooleana:
2     bloque de codigo
3 else:
4     bloque de codigo
```

Listing 1: Operaciones If-else

```
1 if ExpresionBooleana:
2     bloque de codigo
```

Listing 2: Operaciones If

```
1 if ExpresionBooleana:
2     bloque de codigo
```

```

3 elif ExpresionBooleana2:
4     bloque de codigo
5 else:
6     bloque de codigo

```

Listing 3: Operaciones If en casacada

```

1 if ExpresionBooleana:
2     bloque de codigo
3 if ExpresionBooleana2:
4     bloque de codigo
5 if ExpresionBooleana3:
6     bloque de codigo

```

Listing 4: Operaciones If en consecutivo

```

1 if ExpresionBooleana:
2     bloque de codigo
3     if ExpresionBooleana2:
4         bloque de codigo
5         if ExpresionBooleana3:
6             bloque de codigo

```

Listing 5: Operaciones If en anidadas

En estas operaciones es bueno tener en cuenta los opuestos logicos de las operaciones logicas.

OPUESTOS LÓGICOS

Cada uno de los 6 operadores relacionales tienen un opuesto lógico. Son útiles para expresar condiciones que requieren una negación ya que el operador de negación no siempre es evidente de usar

Operador	Opuesto lógico
==	!=
!=	==
<	>=
<=	>
>	<=
>=	<

Figure 4: Opuestos Logicos

4 Cadenas de caracteres o Strings

Existen diversos estandares de caracteres, sin embargo, los mas relevantes son.

- ASCII (127 caracteres)

- ISO-8859-1 (Incluye caracteres europeos como los del frances)
- UNICODE (Incluye todos los idiomas y emojis)

Ahora bien, dentro de los **strings** hay ciertos caracteres los cuales cumplen funciones de control y longitud.

4.1 Caracteres de control

Son caracteres especiales que no tienen representaciones triviales.

- `\n`: Cambio de linea
- `\t`: Tabulador
- `\\`: Barra invertida

4.2 Longitud de cadena

Es en pocas palabras el numero de caracteres o letras que posee una palabra. Para esta se usa la funcion `len()` en python. Hay que tener en cuenta que esta funcion tambien cuenta los espacion entre palabras.

4.3 Operaciones

Asi como en los booleanos y numeros, las cadenas de caracteres tambien pueden ser comparables, por ende, los operadores de relacion y logicos tambien pueden ser aplicados en strings. Sin embargo, en las cadenas hay un nuevo operador.

- **in** y **not in**: este comprueba si una cadena hace parte de otra cadena.

```
1 "x" in "manzana" #Va a dar False
2 "x" not in "manzana" #Va a dar True
```

Listing 6: Operacion in

4.4 Metodos

Ahora bien, las cadenas tienen sus propios metodos. Sin embargo, hay que tener en cuenta que una vez definida una cadena esta no puede cambiar su contenido, si se quiere cambiar, hay que crear una nueva cadena(**Inmutabilidad**). Estos son algunos metodos.

- `lower()`: devuelve una nueva cadena con todos los caracteres en minusculas.
- `upper()`: devuelve una nueva cadena con todos los caracteres en mayusculas.

- `title()`: devuelve una nueva cadena con la primera letra de cada palabra en mayusculas.
- `swapcase()`: devuelve una nueva cadena con los caracteres en minuscula convertidos en mayusculas y viceversa.
- `replace()`: Recibe dos cadenas, un patron y un reemplazo, el metodo busca el patron en la cadena y lo cambia por el reemplazo en cada una de sus apariciones.
- `find()`: Se le da una cadena con el fin de que la busque en la cadena principal, esta devuelve el indice en la que se encuentra la cadena en la otra, y si no lo encuentra devuelve `-1`.
- `count()`: Dada una cadena, cuenta cuantas veces esta esa cadena en la cadena principal.
- `format()`: se colocan `{}` en la cadena con los indices dentro de la forma `{0}` con el fin de que sea remplazado con los parametros del metodo `format`.

Todos estos metodos y su informacion, puede ser consultada con el metodo `help()` en python.

5 Diccionarios

Son **estructuras de datos**, es decir, tipos de datos compuestos que nos permiten manejar otros datos determinados en `{llave:valor}`. Su tipo de dato en python se determina como **dict**.

Estos se pueden iniciar como `variable = {}` y se pueden crear ya con datos. Se pueden agregar datos de la siguiente manera:

```

1 agenda_dia = {}
2
3 agenda_dia[6] = "Empieza el dia"
4 agenda_dia[8] = "Desayuno con Clara"
5 agenda_dia[9.5] = "Clase de IP"
6 agenda_dia[11] = "Clase de calculo"
7 agenda_dia[13] = "Almuerzo"
8 agenda_dia[15] = "Natacion"
9 agenda_dia[16] = "Repaso de quimica"
10 agenda_dia[19.5] = "Visita a los abuelos"

```

Listing 7: Operacion adicion diccionario

Ahora bien, para acceder, modificar se usa la forma:

```

1 #Modificacion
2 agenda_dia[6] = "Empieza el noche"
3 #Consulta
4 agenda_dia[6]
5 agenda_dia.get(6, "No hay esa hora en la agenda") # Se le da la
    llave y un valor si no existe

```

```
6 agenda_dia.get(6, None) # Se da el valor de None tambien ya que este  
significa ausencia de valor
```

Listing 8: Operacion modificacion y consulta diccionario

Por otra parte, en los diccionarios tambien se hace uso de los operadores **in** y **not in** con el fin de conocer si la llave existe en el diccionario y tambien se puede usar la funcion **len()**.

Hay una componente a tener en cuenta en los diccionarios y es que son **mutables**, esto quiere decir que si hay dos variables apuntando al mismo diccionario, y se modifica alguna de estas, la otra tambien se modifica. Entonces, si queremos conservar el diccionario original, debemos hacer una copia con el metodo **copy()**