ESCUELA POLITÉCNICA NACIONAL

cec·epn CENTRO DE EDUCACIÓN CONTINUA

## UNIDAD 3

# Valores booleanos, ejecución condicional, bucles, operaciones lógicas

# Datos lógicos - booleanos.

Una variable booleana es aquella que sólo puede tomar dos posibles valores: True (verdadero) o False (falso).

En Python cualquier variable (en general, cualquier objeto) puede considerarse como una variable booleana. En general los elementos nulos o vacíos se consideran False y el resto se consideran True.

```
>>> bool(0)
False
>>> bool(0.0)
False
>>> bool("")
False
```

```
>>> bool(25)
True
>>> bool(-9.5)
True
>>> bool("abc")
True
```

## Operadores lógicos

and: "y" lógico. Este operador da como resultado True si y sólo si sus dos operandos son True:

```
>>> True and True
True
>>> True and False
False
>>> False and True
False
>>> False and False
False
```

or: "o" lógico. Este operador da como resultado True si algún operando es True:

```
>>> True or True
True
>>> True or False
True
>>> False or True
True
>>> False or False
False
```

not: negación. Este operador da como resultado True si y sólo si su argumento es False:

```
>>> not True
False
>>> not False
True
```

## Asignación múltiple

Es posible asignar en una sola instrucción, múltiples variables:

```
a, b, c = 'string', 15, True
```

## Estructuras de control de flujo condicionales

Aquellas que permiten evaluar si una o más condiciones se cumplen, para decir qué acción se va a ejecutar en consecuencia. La evaluación de condiciones, solo puede arrojar 1 de 2 resultados: verdadero o falso (True o False).

Para describir la evaluación a realizar sobre una condición, se utilizan operadores relacionales (o de comparación)

| Símbolo | Significado | Ejemplo | Resultado |
|---|---|---|---|
| `==` | Igual que | `5 == 7` | `False` |
| `!=` | Distinto que | `rojo != verde` | `True` |
| `<` | Menor que | `8 < 12` | `True` |
| `>` | Mayor que | `12 > 7` | `True` |
| `<=` | Menor o igual que | `12 <= 12` | `True` |
| `>=` | Mayor o igual que | `4 >= 5` | `False` |

Para evaluar más de una condición simultáneamente, se utilizan operadores lógicos:

| Operador | Ejemplo | Explicación | Resultado |
|---|---|---|---|
| `and` | `5 == 7 and 7 < 12` | `False and False` | `False` |
| `and` | `9 < 12 and 12 > 7` | `True and True` | `True` |
| `and` | `9 < 12 and 12 > 15` | `True and False` | `False` |
| `or` | `12 == 12 or 15 < 7` | `True or False` | `True` |
| `or` | `7 > 5 or 9 < 12` | `True or True` | `True` |

Las estructuras de control de flujo condicionales, se definen mediante el uso de tres palabras claves reservadas, del lenguaje: if (si), elif (sino, si) y else (sino).

```
if compra <= 100:
    print "Pago en efectivo"
elif compra > 100 and compra < 300:
    print "Pago con tarjeta de débito"
else:
    print "Pago con tarjeta de crédito"
```

## Listas

Es una estructura de datos y un tipo de dato en python con características especiales. Lo especial de las listas en Python es que nos permiten almacenar cualquier tipo de valor como enteros, strings y hasta otras funciones; por ejemplo:

```
lista = [1, 2.5, 'DevCode', [5,6] ,4]
```

Para acceder a estos datos podemos hacer mediante su índice.

```
print lista[0] # 1
print lista[1] # 2.5
print lista[2] # DevCode
print lista[3] # [5,6]
print lista[3][0] # 5
print lista[3][1] # 6
print lista[1:3] # [2.5, 'DevCode']
print lista[1:6] # [2.5, 'DevCode', [5, 6], 4]
print lista[1:6:2] # [2.5, [5, 6]]
```

**Heterogéneas**: pueden estar conformadas por elementos de distintos tipo, incluidos otras listas.
**Mutables**: sus elementos pueden modificarse.

A través de los índices es posible cambiar el valor de esa posición.

```
>>> factura[1] = "carne"
```

## Operadores de Pertenencia

Se emplea para identificar pertenencia en alguna secuencia (listas, strings).

*in* y *not in* son operadores de pertenencia.

in devuelve True si el valor especificado se encuentra en la secuencia. En caso contrario devuelve False.

not in devuelve True si el valor especificado no se encuentra en la secuencia. En caso contrario devuelve False.

```
>>> a = [1,2,3,4,5]
>>> print (3 in a)
True
>>> print (12 not in a)
True
>>> print (1 not in a)
False
```

## Operadores de Identidad

Se emplea para comprobar si dos variables emplean la misma ubicación en memoria.

*is* e *is not* son operadores de identidad.

*is* devuelve True si los operandos se refieren al mismo objeto. En caso contrario devuelve False. *is not* devuelve True si los operandos no se refieren al mismo objeto. En caso contrario devuelve False.

Se debe tener en cuenta que dos valores, cuando son iguales, no implica necesariamente que sean idénticos.

```
a = 3
b = 3
c = 4
print (a is b) # muestra True
print (a is not b) # muestra False
print (a is not c) # muestra True

x = 1
y = x
z = y
print (z is 1) # muestra True
print (z is x) # muestra True

str1 = "FreeCodeCamp"
str2 = "FreeCodeCamp"

print (str1 is str2) # muestra True
print ("Code" is str2) # muestra False

a = [10,20,30]
b = [10,20,30]

print (a is b) # muestra False (ya que las listas son objetos mutables en Python)
```

# Estructuras de control iterativas

También llamadas cíclicas o bucles, permiten ejecutar un mismo código, de manera repetida, mientras se cumpla una condición.

## Bucle while

Este bucle, se encarga de ejecutar una misma acción "mientras que" una determinada condición se cumpla.

```
# -*- coding: utf-8 -*-
anio = 2001
while anio <= 2012:
    print "Informes del Año", str(anio)
    anio += 1
```

## Bucle for

El bucle for, en Python, es aquel que nos permitirá iterar sobre una variable compleja, del tipo lista o tupla

```
mi_lista = ['Juan', 'Antonio', 'Pedro', 'Herminio']
for nombre in mi_lista:
    print nombre
```


ESCUELA
POLITÉCNICA
NACIONAL


cec·epn
CENTRO DE EDUCACIÓN CONTINUA


python™