DECISIONES (CONDICIONALES)

Introducción a la Programación (11071)
Departamento de Ciencias Básicas
Universidad Nacional de Luján

ESTRUCTURA DE LA CLASE

- Introducción y contexto.
- El tipo de dato “booleano”.
- Operadores relacionales.
- Operadores lógicos.
- Estructura condicional simple.

CONTEXTO

DECISIONES

La mayoría de los programas necesitan tomar alguna *decisión*.

SECUENCIALIDAD

En el paradigma procedural, los programas son, por naturaleza, secuenciales. Necesitamos poder “romper” la secuencialidad para “tomar decisiones” (ejecutar o no cierta porción del código).

EL TIPO “BOOLEANO”

TIPO DE DATO PRIMITIVO

“boolean” es un tipo primitivo (ya implementado), presente en la mayoría de los lenguajes de programación.

VALORES ADMITIDOS

Las variables de tipo *boolean* admiten sólo dos valores: “verdadero” (*true*), o “falso” (*false*).

OPERACIONES

Las operaciones que se pueden hacer con los booleanos se derivan de la *lógica proposicional*. Ejemplos: *conjunción, disyunción, negación*, etcétera.

EN PYTHON

TIPO BOOL

Python admite los valores **True** y **False** para las variables de tipo *bool* (prestar atención al case-sensitive!).

EJEMPLO

```
predicado = True
print(predicado)
```

Al ejecutarlo, muestra:

```
True
```

True y False no llevan comillas, porque no son literales.
True ≠ "True"

UTILIDAD DE LOS BOOL

LÓGICA PROPOSICIONAL

Con una variable booleana, podemos expresar si un predicado (condición) es verdadero o falso.

¿DECISIONES?

Si tenemos la posibilidad de expresar, dentro de nuestro código, si algo es “verdadero” o “falso”, estamos a un paso de poder tomar una decisión dependiendo de ello.

UTILIDAD DE LOS BOOL

LÓGICA PROPOSICIONAL

Con una variable booleana, podemos expresar si un predicado (condición) es verdadero o falso.

¿DECISIONES?

Si tenemos la posibilidad de expresar, dentro de nuestro código, si algo es “verdadero” o “falso”, estamos a un paso de poder tomar una decisión dependiendo de ello.

Pero antes...

OPERADORES RELACIONALES

RESULTADOS BOOLEANOS

Ciertas operaciones dan como resultado un valor booleano. Ejemplos: comparar si un número es mayor a otro, verificar si un string coincide con “hola”, etcétera.

EJEMPLOS

```
10 > 20
```

```
n1 <= n2
```

```
cadena == "Hola"
```

```
len(nombre1) != len(nombre2)
```

EN PYTHON

OPERADORES RELACIONALES

Sintaxis	Operación
==	Igual que...
!=	Distinto de...
<	Menor estricto que...
>	Mayor estricto que...
<=	Menor o igual que...
>=	Mayor o igual que...

En Python, todos los operadores relacionales tienen el mismo orden de precedencia, por lo que se resuelven de izquierda a derecha.

OPERADORES LÓGICOS

OPERACIONES ENTRE BOOLEANOS

A diferencia de los *operadores relacionales*, los *operadores lógicos* se aplican exclusivamente entre booleanos, y dan como resultado un nuevo valor booleano.

OPERACIONES COMUNES

- Conjunción (**y** lógico, **and**).
- Disyunción (**o** lógico, **or**).
- Negación (**no** lógico, **not**).

CONJUNCIÓN

DESCRIPCIÓN DE LA OPERACIÓN

Retorna *verdadero* si y sólo si ambos operandos son *verdaderos*.

SINTAXIS PYTHON

```
operando1 and operando2
```

EJEMPLOS

```
(20 > 10) and (50 < 100)
```

```
True and False
```

```
condicion1 and condicion2 and (100 > numero)
```

DISYUNCIÓN

DESCRIPCIÓN DE LA OPERACIÓN

Retorna *verdadero* si al menos uno de los operandos es *verdadero*.

SINTAXIS PYTHON

```
operando1 or operando2
```

EJEMPLOS

```
(20 > 10) or (50 < 100)
```

```
True or False
```

```
condicion1 or condicion2 or (100 > numero)
```

NEGACIÓN

DESCRIPCIÓN DE LA OPERACIÓN

Niega (invierte) el valor del operando.

SINTAXIS PYTHON

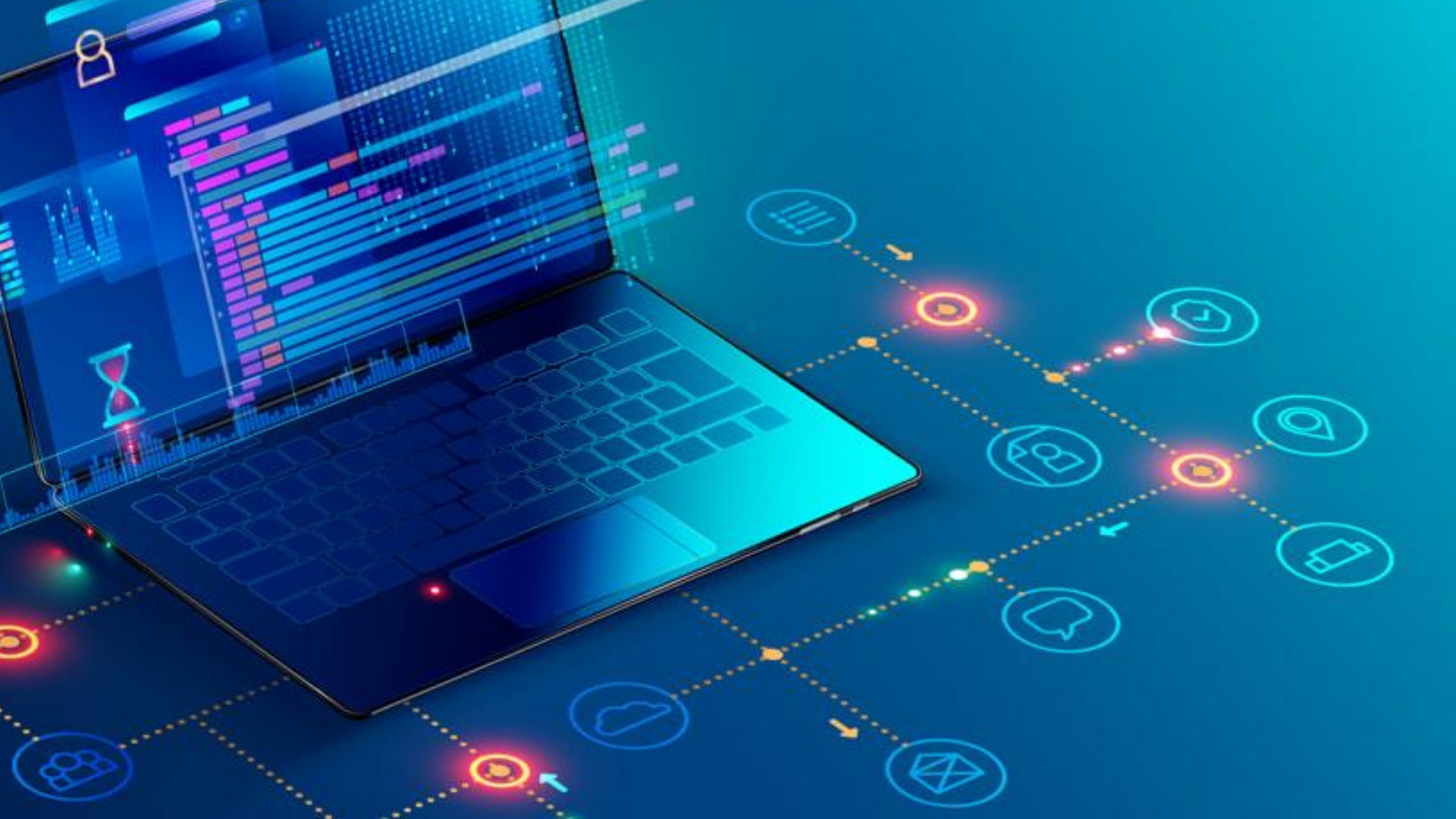
```
not operando
```

EJEMPLOS

```
not (20 > 10)
```

```
not (True or False)
```

```
not (condicion1 or condicion2 or (100 > numero))
```



ESTRUCTURA CONDICIONAL SIMPLE

OBJETIVO

Tomar decisiones en nuestro programa, utilizando condiciones booleanas como condición para ejecutar (o no) determinado código.

SINTAXIS PYTHON

```
if (condición):
    [ código a ejecutar si la condición es verdadera ]
```

Recordar que la indentación es parte de la sintaxis, y es importante para indicar qué está incluído dentro del bloque if, y cuándo termina.

CONDICIÓN

La condición del *if* puede ser cualquier expresión o referencia a variable, siempre que su valor final sea un booleano.

ESTRUCTURA CONDICIONAL SIMPLE

EJEMPLOS

```
if (10 > 4):
    print('10 es mayor que 4! Denme el Nobel!')
```

```
if (bandera):
    print('La bandera es verdadera.')
```

```
if (not bandera):
    print('La bandera NO es verdadera.')
```

```
if (numero1 == numero2) and (numero3 > numero4):
    print('Condiciones numéricas cumplidas!')
    # código espagueti
    # mucho más código espagueti
```

SI... SINO?

ELSE

Opcionalmente, se puede extender el *if* con un *else*, el cual ejecutará un bloque de código determinado cuando la condición sea *falsa*.

SINTAXIS PYTHON

```
if (condición):
    [ código a ejecutar si la condición es verdadera ]
else:
    [ código a ejecutar si la condición es falsa ]
```

SI... SINO?

EJEMPLOS

```
if (10 > 4):
    print('10 es mayor que 4! Denme el Nobel!')
else:
    print('Algo anda muy mal en Python...')
```

```
if (bandera):
    print('La bandera es verdadera.')
else:
    print('La bandera es falsa.')
```

```
if (numero1 == numero2) and (numero3 > numero4):
    print('Condiciones numéricas cumplidas!')
else:
    print('Condiciones insatisfechas.')
```

CONDICIONALES ANIDADOS

OBJETIVO

Se pueden evaluar varias condiciones de manera consecutiva, “anidando” (escribiendo secuencialmente) varios *if* y *else*. Python tiene una sintaxis especial para estos casos (***elif***).

SINTAXIS PYTHON

```
if (condición 1):
      [ código a ejecutar si la condición 1 es verdadera ]
elif (condición 2):
      [ código a ejecutar si la condición 1 es falsa, y la condición 2 es verdadera. ]
```

CONDICIONALES ANIDADOS

EJEMPLOS

```
if (10 > 20):
    print('Mmmmm...')
elif (50 > 25):
    print('En este caso anduvo.')
```

```
if (bandera):
    print('La bandera es verdadera.')
else:
    print('La bandera es falsa.')
```

```
if (numero1 == numero2) and (numero3 > numero4):
    print('Condiciones numéricas cumplidas!')
elif (numero1 == numero2):
    print('Al menos la primera parte de la condición es verdadera.')
```

CONDICIONALES ANIDADOS

Cualquier estructura condicional *if* se puede extender con una cantidad ilimitada de *elif*, e incluso agregar también el *else*, si queremos realizar algo si todas las demás condiciones resultan falsas.

EJEMPLO

```
if (numero1 == numero2) and (numero3 > numero4):
    print('Condiciones numéricas cumplidas!')
elif (numero1 == numero2):
    print('Al menos la primera parte de la condición es verdadera.')
elif (numero3 > numero4):
    print('Al menos la segunda parte de la condición es verdadera.')
else:
    print('Ambas partes de la condición son falsas.')
```