

IWI 131-  
PROGRAMACIÓN

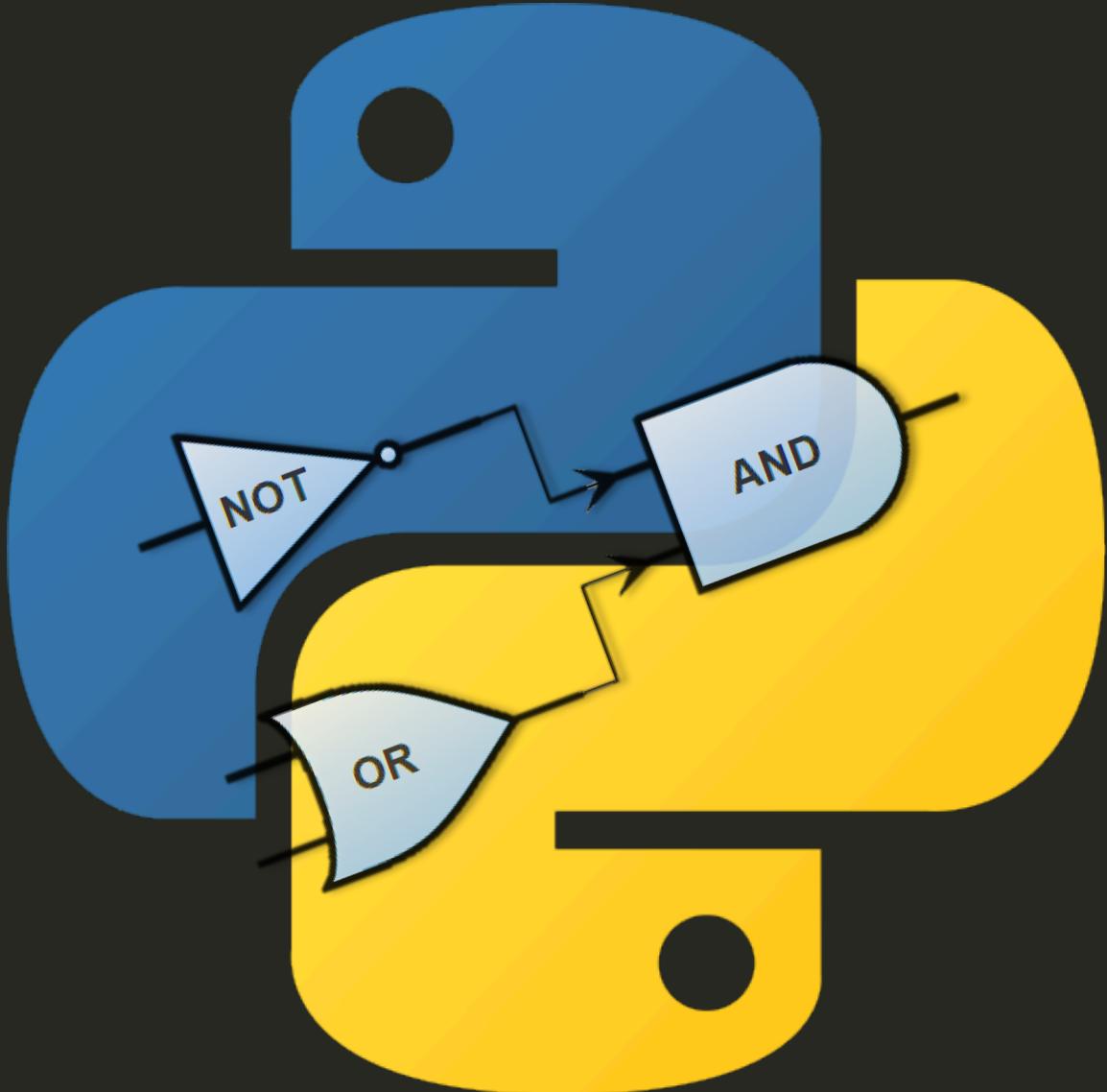
---

Ayudantía 3

Ayudante: Anastasiia Fedorova

Paralelo: 212

Fecha: 22.04.2020



# RECORDATORIO: TIPO BOOL

Booleanos: True / False (1 / 0)

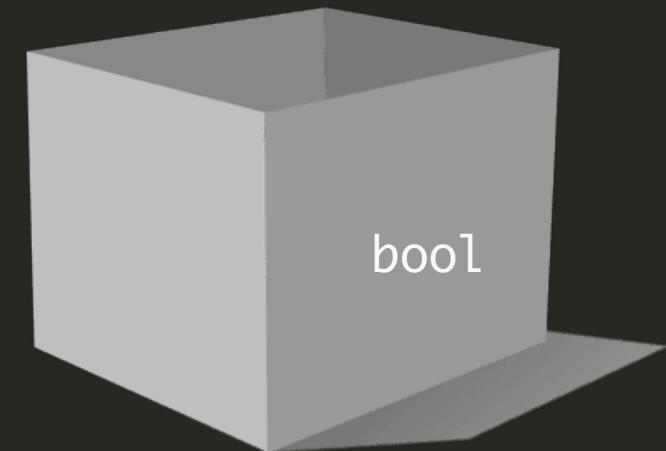
En general los elementos nulos o vacíos se consideran False y el resto se consideran True. De lo que ya conocemos como objeto de Python (str, int, float):

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

PERO

```
>>> bool("")  
False
```

```
>>> bool(1)  
True  
>>> bool(12.1)  
True  
>>> bool("Hola!")  
True
```



Recuerden que >>> se usa solo para marcar que la linea se evalua directamente en terminal, despues de escribir python3.

# OPERADORES DE COMPARACIÓN

Python trae consigo una serie de operadores, que permiten comparar dos elementos.

Igual a



```
var_1 = 1  
var_2 = 2
```

Si usamos el operador ==,  
var1 == var2  
se evalua como False

Distinto a



Pero si usamos !=.  
var1 != var2  
se evalua como True

# OPERADORES DE COMPARACIÓN

Para comparar la magnitud de los elementos, se usan `>` y `<`

Mayor que



```
var_1 = 1  
var_2 = 2
```

Si usamos el operador `>`,  
`var1 > var2`  
se evalua como `False`

Menor que

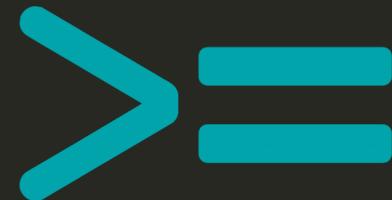


Pero si usamos `<`.  
`var1 < var2`  
se evalua como `True`

# OPERADORES DE COMPARACIÓN

No mezclar con mate – el igual se escribe aparte pero siempre despues de > o <.

Mayor o igual



```
var_1 = 1  
var_2 = 2
```

Si usamos el operador `>=`,  
`var1 >= var2`  
se evalua como `False`

Menor o igual



Pero si usamos `<=`.  
`var1 <= var2`  
se evalua como `True`

# OPERADORES LÓGICOS

and

Condición 1	Condición 2	Resultado
False	False	False
False	True	False
True	False	False
True	True	True

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

or

Condición 1	Condición 2	Resultado
False	False	False
False	True	True
True	False	True
True	True	True

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

not

Condición 1	Resultado
False	True
True	False

Negación, invierte el input.

## EJERCICIO 1

Por ejemplo, quiero filtrar números que sean pares y, además, no divisibles entre 8.

Entonces, tenemos 2 condiciones: el número tiene que dar resto 0 al dividir este entre 2 y la division entre 8 no debe dar 0. Eso mismo, pero en código:

`n % 2 == 0`

`n % 8 != 0`

Entonces, para que se cumplan ambas condiciones, tenemos que unirlas con el "y" lógico:

`(n % 2 == 0) and (n % 8 != 0)`

# EJERCICIO 1

$(n \% 2 == 0) \text{ and } (n \% 8 != 0)$

¿A qué se evalua?

n = 10

n = 8

n = 4

n = 1

# EJERCICIO 1

$(n \% 2 == 0) \text{ and } (n \% 8 != 0)$

¿A qué se evalua?

$n = 10$	True and True	True
$n = 8$	True and False	False
$n = 4$	True and True	True
$n = 1$	False and True	False

¿Es posible escribir el código usando solo operador  $\equiv ?$  ¿Cómo?

# EJERCICIO 1

$(n \% 2 == 0) \text{ and } (\text{not}(n \% 8 == 0))$

¿A qué se evalua?

$n = 10$	$\text{True and not False}$	True
$n = 8$	$\text{True and not True}$	False
$n = 4$	$\text{True and not False}$	True
$n = 1$	$\text{False and not False}$	False

Respuesta: usando el operador de la negación. Expresiones con muchos operadores lógicos se conocen como expresiones de lógica compuesta.

# IF-ELIF-ELSE

La estructura de control `if()...` permite que un programa ejecute unas instrucciones cuando la condición dentro de los paréntesis se evalua como True.

La sentencia `else` se conoce como sentencia por defecto – se ejecuta si el `if` se evalua como falso. ( o el `if` y todos los `elif` anidados)

Con los bloques:

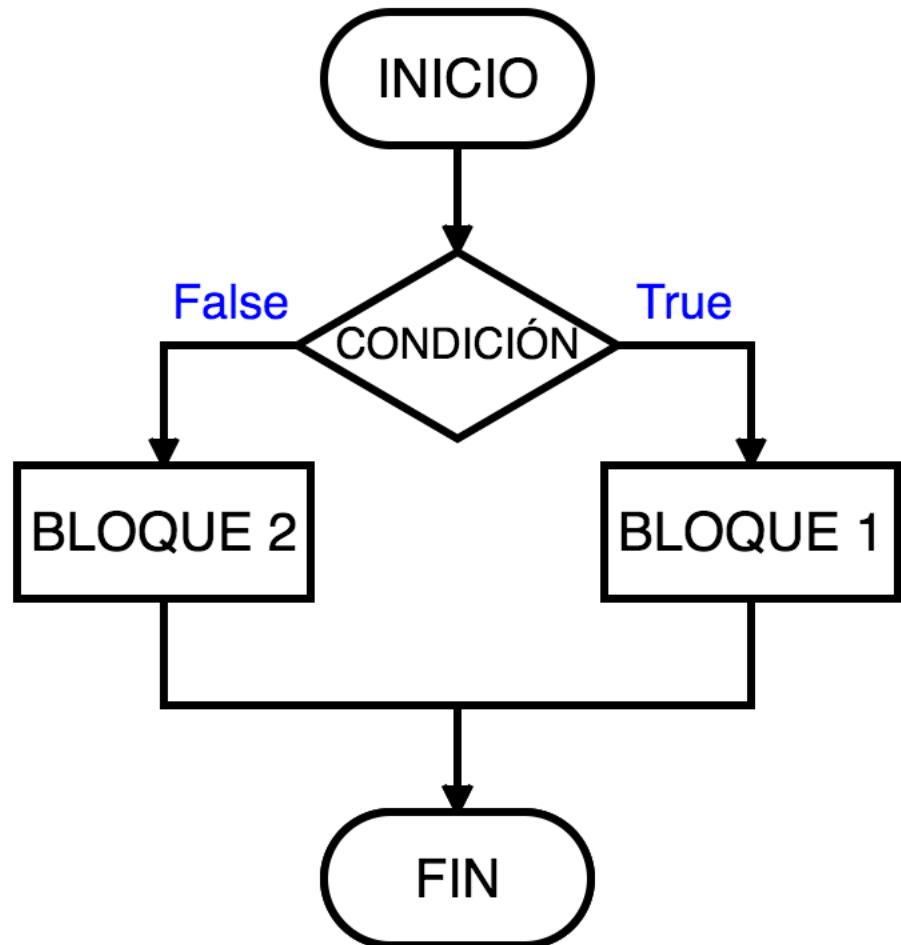
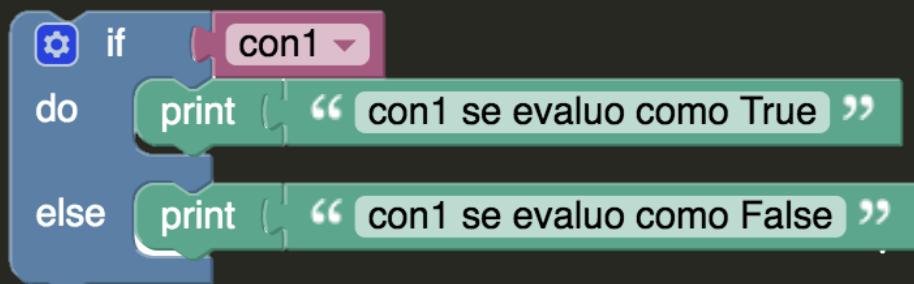


Diagrama de flujo para if-else

# IF-ELIF-ELSE

La estructura de control `if ... elif ... else` permite encadenar varias condiciones. `elif` es una contracción de `else if`.

Con los bloques:

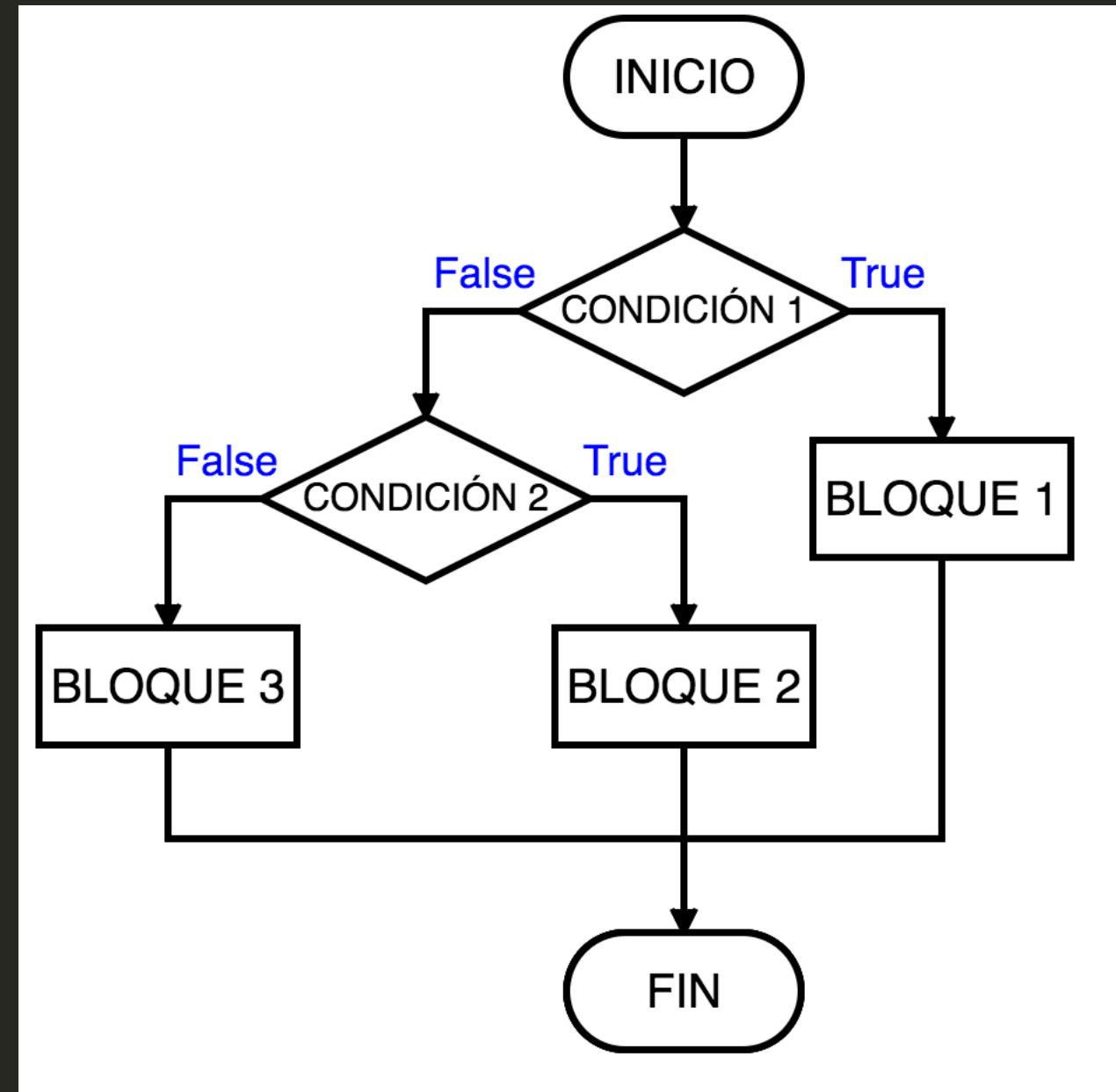
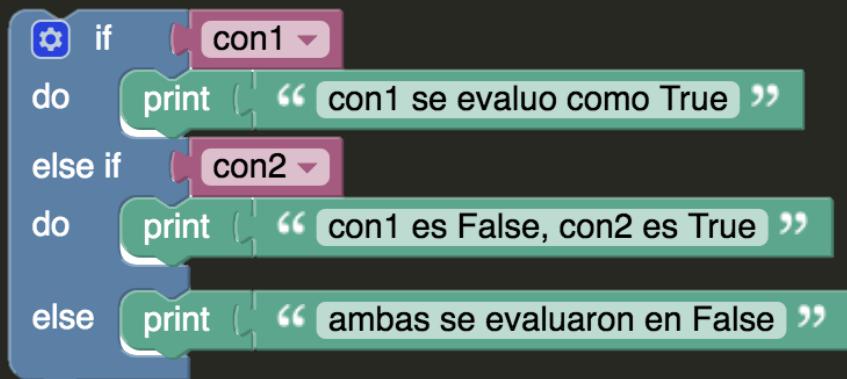


Diagrama de flujo para `if-elif-else`

# IF-ELIF-ELSE

Su sintaxis correcta es:

```
if(condicion1):
    #bloque 1, se ejecuta si condicion1 es True
elif(condicion2):
    #bloque 2, se ejecuta si condicion1 es False y condicion2 es True
elif(condicion3):
    #bloque 3, condicion3 es True y condicion1 y condicion2 False
else:
    #bloque por defecto, todas las condiciones se evaluaron como False
```

## EJERCICIO 2 - CONSONANTE Y VOCAL

La lengua materna de su ayudante de hecho no es español (gracias, Capitán Obvio), por lo que un día esta tuvo que aprender su lenguaje para poder transmitir los conocimientos de progra a los mechones (y de paso, estudiar en una U, detalles). Pero, como esto implicaba aprender un alfabeto nuevo, se le surgieron dudas sobre las vocales y las consonantes.



Para ayudarle, se les pide escribir un programa que imprima que letra fue ingresada por el usuario – vocal o consonante. Asuma que este puede ingresar tanto mayúsculas como minúsculas.

Input:	Output:
a	Vocal
O	Vocal
B	Consonante

# EJERCICIO 2 - SOLUCIÓN

```
letra = input("Ingrese una letra: ")
if(letra == 'a' or letra == 'e' or letra == 'i' or letra == 'o' or letra == 'u'):
    print("Vocal")
elif(letra == 'A' or letra == 'E' or letra == 'I' or letra == 'O' or letra == 'U'):
    print("Vocal")
else:
    print("Consonante")
```

Tip para los interesados:

Investiguen sobre `isupper()`, `islower()`, `upper()` y `lower()`  
y reescriban el programa con estas funciones.

## EJERCICIO 3 - ESTOY TEMBLANDO

Su ayudante de progra (again) estaba viendo la clase de FIS140 en paz, cuando de repente su profesor dijo "Ahora usaremos la representación fasorial de esta solución de la ecuación de la onda electromagnética" y su mundo temblo (o solo Chile).

Su ayudante les pide escribir un programa, que reciba la magnitud de temblor y la clasifique según la escala de Richter.

Input:

5.1

0.1

Input:

Moderado

Micro

Magnitudes Richter	Descripción
Menos de 2,0	Micro
2,0-2,9	
3,0-3,9	Menor
4,0-4,9	Ligero
5,0-5,9	Moderado
6,0-6,9	Fuerte
7,0-7,9	Mayor
8,0-8,9	
9,0-9,9	Gran
10,0+	Épico

# EJERCICIO 3 - SOLUCIÓN

```
magnitud = float(input("Ingrese la magnitud de temblor: "))
if(magnitud < 2):
    print("Micro")
elif((magnitud >= 2 and magnitud <= 3.9)):
    print("Menor")
elif(magnitud >= 4.0 and magnitud <= 4.9):
    print("Ligero")
elif(magnitud >= 5.0 and magnitud <= 5.9):
    print("Moderado")
elif(magnitud >= 6.0 and magnitud <= 6.9):
    print("Fuerte")
elif(magnitud >= 7.0 and magnitud <= 7.9):
    print("Mayor")
elif(magnitud >= 8.0 and magnitud <= 9.9):
    print("Gran")
else:
    print("Epico")
```

# EJERCICIO 4 - RUTEO

```
a = 10
c = True
x = 0
y = x + 1
z = 0 and c
if(x or z):
    print("1")
elif(x and y):
    print("2")
elif((not z or c) and a):
    print("3")
else:
    print("4")
```

# EJERCICIO 4 - SOLUCIÓN

```
a = 10
c = True
x = 0
y = x + 1
z = 0 and c
if(x or z):
    print("1")
elif(x and y):
    print("2")
elif((not z or c) and a):
    print("3")
else:
    print("4")
```

a	c	x	y	z
10				
	True			
		0		
			1	
				False

Pantalla: 3

# EJERCICIO 5 – BHASKARA (DESAFÍO)

Para pasar MAT21, es indispensable conocer cómo resolver las ecuaciones cuadráticas. Para esto, existe la formula de Bhaskara:

$$x = \frac{-b \pm \sqrt{b^2 - 4.a.c}}{2.a}$$

Pedir al usuario los coeficientes  $a$ ,  $b$  y  $c$ , y calcular las 2 raíces de la ecuación cuadrática. En caso de que no sea posible encontrar las raíces debido a la división entre 0 o determinante negativo, imprimir “No es posible calcular”.

Input:

10.0

20.1

5.1

Input:

R1 = -0.29788

R2 = -1.71212

Input:

0.0

20.0

5.0

Output:

No es posible calcular

# EJERCICIO 5 - SOLUCIÓN

```
a = float(input("Ingrese a: "))
b = float(input("Ingrese b: "))
c = float(input("Ingrese c: "))
if (a == 0):
    print("No es posible calcular")
else:
    D = (b ** 2 - 4 * a * c)
    if (D < 0):
        print("No es posible calcular")
    else:
        D = D ** 0.5 #la forma mas corta es D **= 0.5
        R1 = (-b + D) / (2 * a)
        R2 = (-b - D) / (2 * a)
        print("R1 =", R1)
        print("R2 =", R2)
```