Clase 2 – Operaciones Lógicas

IIC1103 Sección 9 – 2019-2

Profesor: Felipe López

Fecha: 22 de agosto del 2019

Resumen clase pasada

- **Operaciones básicas:**
  - Vimos los operadores `+, - , * , / , ** , // , %`
- **Variables:** Objeto que almacena información
  - Se define escribiendo su nombre. Se le puede asignar inmediatamente un valor o bien puede ser el resultado de una operación
  - Se puede operar con variables
- **Tipos de datos:** Las variables y datos pueden tener distintos valores. También se pueden hacer conversiones entre ellos.
  - `int`: dato de tipo entero
  - `float:` dato de tipo decimal
  - `string`: dato de tipo texto
- Para imprimir en consola se usa `print(…)`. Para pedirle información al usuario se usa `input(…)`.

Contenidos

1. Otros comandos de Python y paquetes
2. Variables bool
3. Operadores lógicos
4. Operadores binarios
5. Ejemplos

Otros comandos

Es importante notar, que además de los comandos que ya hemos visto (como `print()`, `input()`, `int()`, `float()`, `str()`) existen muchos más.

No se verán todos los comandos de Python en el curso, pero sí habrán varios que se verán a medida que avance el semestre.

Otros comandos

Todos los comandos que tiene Python son:

Built-in Functions				
abs()	delattr()	hash()	memoryview()	set()
all()	dict()	help()	min()	setattr()
any()	dir()	hex()	next()	slice()
ascii()	divmod()	id()	object()	sorted()
bin()	enumerate()	input()	oct()	staticmethod()
bool()	eval()	int()	open()	str()
breakpoint()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	

Comandos que ya hemos visto

Comandos que veremos durante el semestre

Paquetes

Además de las funcionalidades que implican los comandos vistos anteriormente, también se pueden añadir funcionalidades extras.

Estas se hacen mediante **paquetes**.

Hay muchos paquetes en internet que cumplen distintos objetivos.

Paquete Random

- Digamos que tenemos el siguiente problema: Queremos simular el lanzamiento de un dado, y el resultado lo queremos multiplicar por cierto número ¿Cómo podríamos hacerlo?

- El paquete Random permite generar un número aleatorio entre dos números.

Paquete Random

- Veamos un ejemplo:

```
#ejemplo paquete random
import random
dado=random.randint(1,6)
numero_mult=int(input("¿Por qué número quieres multiplicar el resultado del dado?"))
print(dado*numero_mult)
```

```
¿Por qué número quieres multiplicar el resultado del dado? 1200
6000
```

Paquete Random

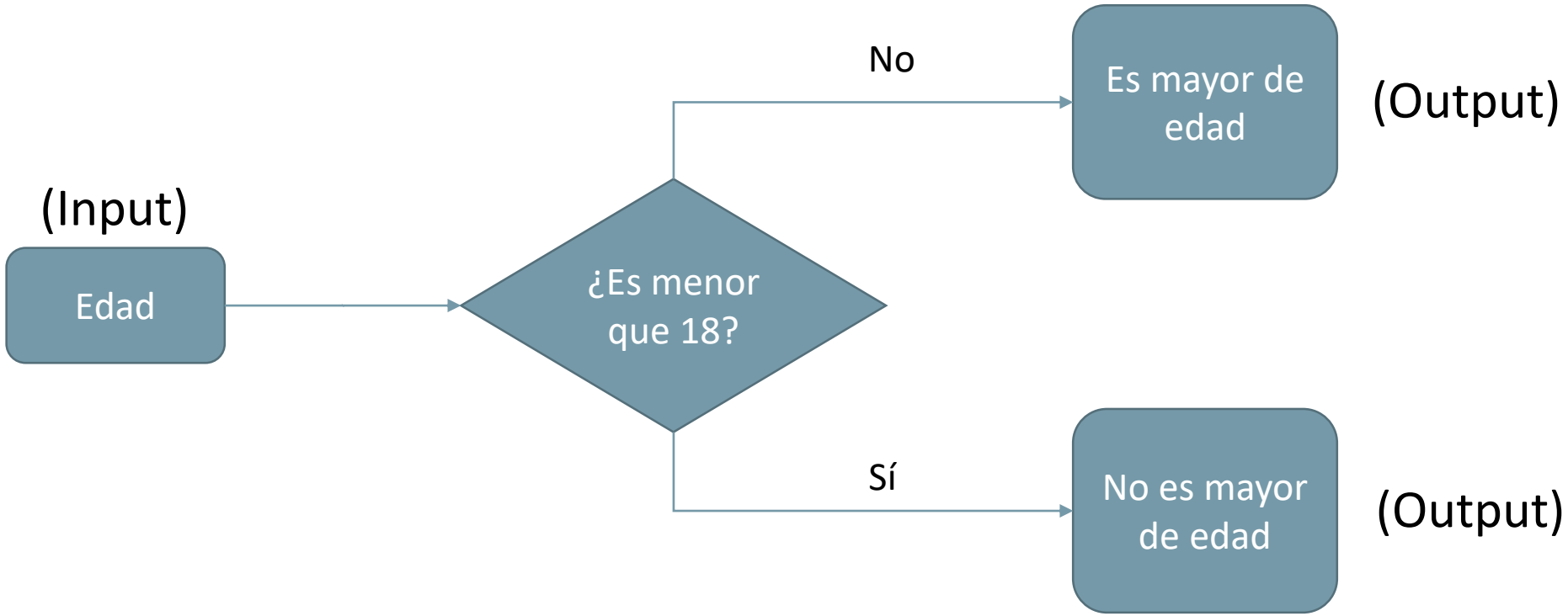
- El comando `import` nos permite usar paquetes que están online. En este caso queremos usar el paquete “Random”

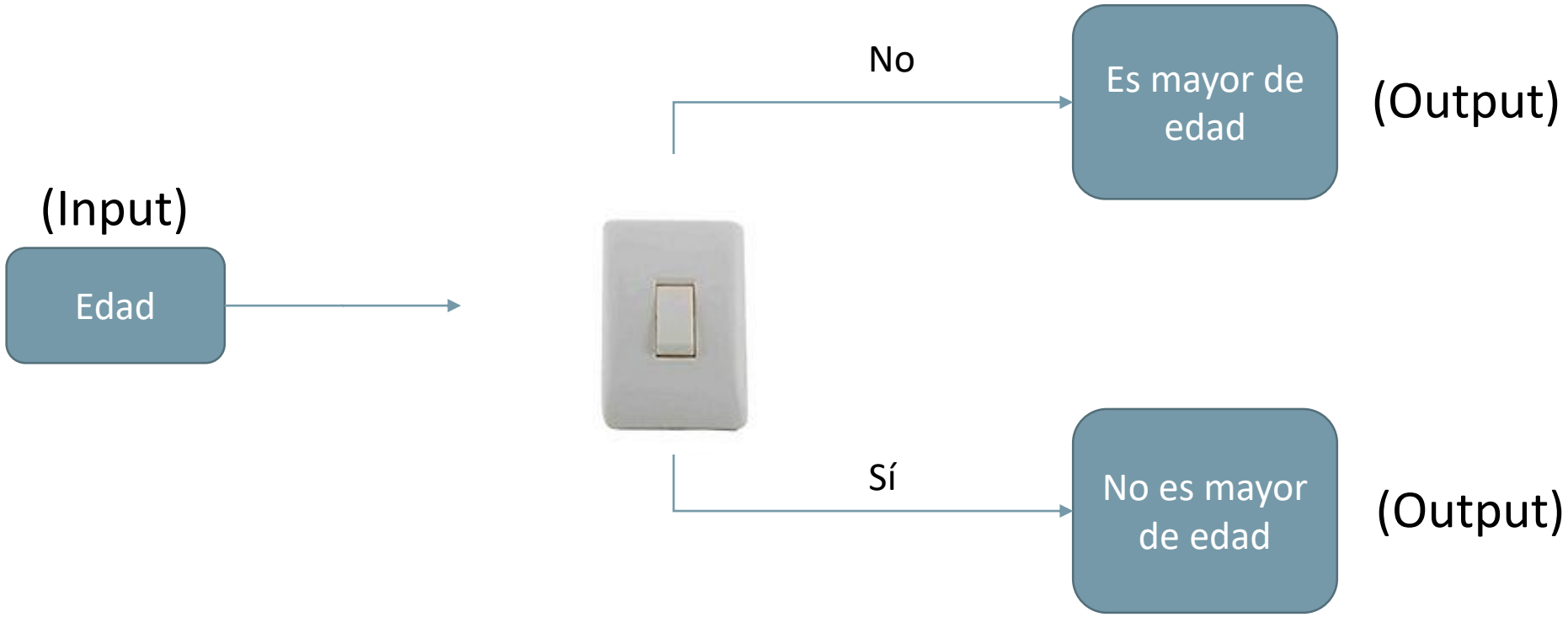
```
import random
```

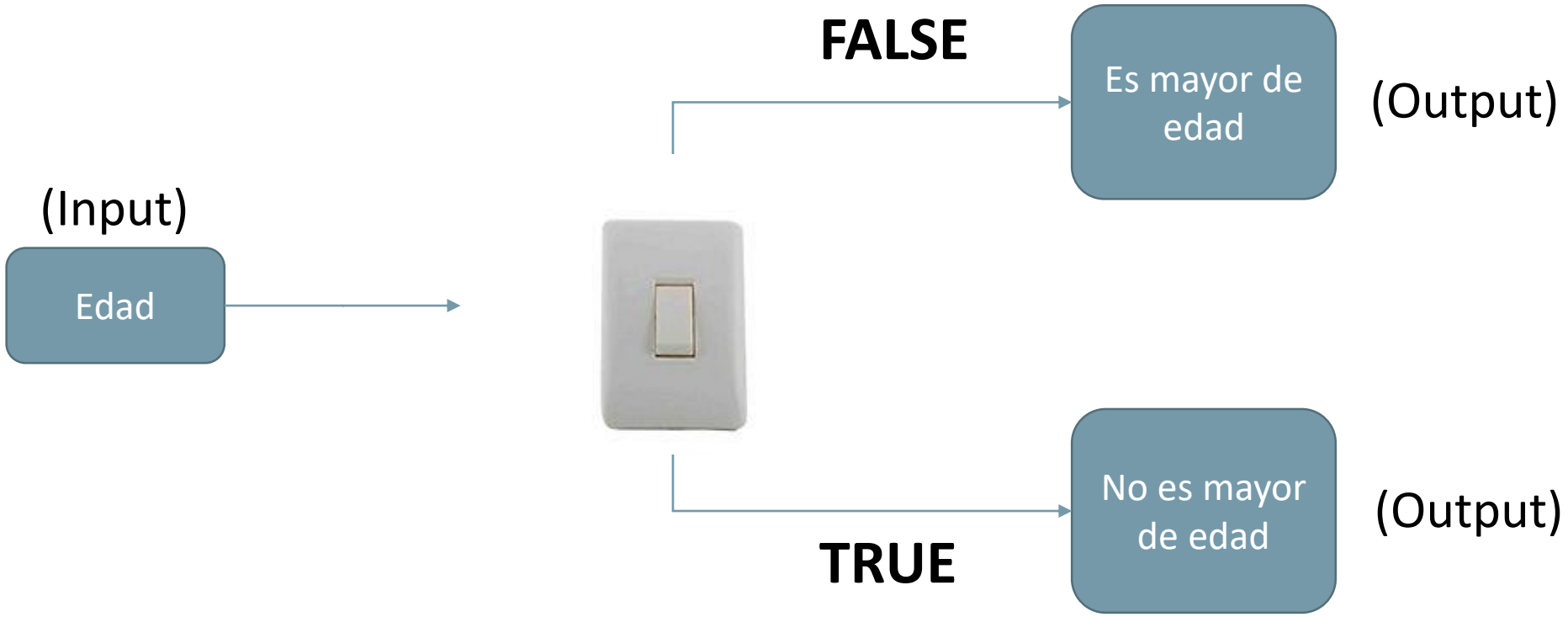
- `random.randint(1,6)` nos indica que del paquete `random`, ocuparemos la función (concepto que veremos más adelante) `randint`. Esta función permite generar un número aleatorio entre el primer número y el segundo número (ambos incluídos) del intervalo que definimos entre paréntesis.

```
dado=random.randint(1,6)
```

- En este caso estamos asignando este número aleatorio a la variable dado.


(Input)
Edad
¿Es menor que 18?
No
Es mayor de edad
(Output)
Sí
No es mayor de edad
(Output)


(Input)
Edad
No
Es mayor de edad
(Output)
Sí
No es mayor de edad
(Output)


(Input)
Edad
FALSE
Es mayor de edad
(Output)
TRUE
No es mayor de edad
(Output)

¿Qué pasa cuando hago una operación y no la guardo en ninguna variable?

¿Qué pasa cuando hago una operación y no la guardo en ninguna variable?

Se pierde

¿Qué pasa cuando hago una operación y no la guardo en ninguna variable?

Se pierde

¿Pueden el TRUE o FALSE existir sin una variable?

¿Qué pasa cuando hago una operación y no la guardo en ninguna variable?

Se pierde

¿Pueden el TRUE o FALSE existir sin una variable?

¡NO!

¿Qué pasa cuando hago una operación y no la guardo en ninguna variable?

Se pierde

¿Pueden el TRUE o FALSE existir sin una variable?

¡NO! Por lo tanto, los almacenamos en una variable.

Esta variable será de tipo **bool**.

Operadores Lógicos

¿Cómo podemos llegar a una variable booleana (es decir, que tenga como valor TRUE o FALSE) de forma no explícita?

¡Por medio de operadores lógicos!

Operadores Lógicos

Los primeros operadores lógicos que veremos son los de comparación. Se llaman así porque sirven para comparar dos valores. El primer operador lógico de comparación que veremos es:

- ==: Compara el valor de la derecha y la izquierda. Si son iguales retorna TRUE y si son distintos retorna FALSE

Ejemplo:

```
#ejemplo operador lógico igualdad
a=1
b=2
c=1
print(a==a)
print(a==b)
print(b==a)
print(a==c)
```

```
True
False
False
True
```

Operadores Lógicos

- !=: Compara el valor de la derecha y la izquierda. Si son distintos retorna TRUE y si son distintos entrega FALSE.

Veamos un ejemplo (equivalente al anterior):

```
#ejemplo operador lógico desigualdad
a=1
b=2
c=1
print(a!=a)
print(a!=b)
print(b!=a)
print(a!=c)
```

```
False
True
True
False
```

Operadores Lógicos

- Es muy importante notar que los operadores == y != sirven también para comparar `strings`. Por ejemplo:

```
#ejemplo operador lógico textos
a="texto1"
b="texto2"
c="texto1"
print(a==a)
print(a!=b)
print(b!=a)
print(a==c)
```

```
True
True
True
True
```

Operadores Lógicos

- <: Compara el valor de la derecha y la izquierda. Si el de la izquierda es menor que la derecha, entonces retorna TRUE. Si el valor de la izquierda es mayor o igual al de la derecha, entonces retorna FALSE. Veamos un ejemplo:

```
#ejemplo operador lógico <
print(4<7)
print(7<7)
print(11<7)
```

```
True
False
False
```

Operadores Lógicos

- >: Compara el valor de la derecha y la izquierda. Si el de la izquierda es mayor que la derecha, entonces retorna TRUE. Si el valor de la izquierda es menor o igual al de la derecha, entonces retorna FALSE. Veamos un ejemplo:

```
#ejemplo operador lógico >
print(4>7)
print(7>7)
print(11>7)
```

```
False
False
True
```

Operadores Lógicos

- >=: Compara el valor de la derecha y la izquierda. Si el de la izquierda es mayor o igual que la derecha, entonces retorna TRUE. Si el valor de la izquierda es menor al de la derecha, entonces retorna FALSE. Veamos un ejemplo:

```
#ejemplo operador lógico >=
print(4>=7)
print(7>=7)
print(11>=7)
```

```
False
True
True
```

Operadores Lógicos

- <=: Compara el valor de la derecha y la izquierda. Si el de la izquierda es menor o igual que la derecha, entonces retorna TRUE. Si el valor de la izquierda es mayor al de la derecha, entonces retorna FALSE. Veamos un ejemplo:

```
#ejemplo operador lógico <=
print(4<=7)
print(7<=7)
print(11<=7)
```

```
True
True
False
```

Operadores Lógicos

Al igual que con los operadores matemáticos que vimos anteriormente, puedes ocupar los operadores lógicos con variables. Mira el siguiente ejemplo:

```
#varias operaciones lógicas
valor_a = 1
valor_b = 2

res = valor_a == valor_a
print(res)
res = valor_b != valor_b
print(res)
res = valor_a < valor_b
print(res)
res = valor_a > valor_b
print(res)
res = valor_a <= valor_a
print(res)
res = valor_a >= valor_b
print(res)
```

Operadores Lógicos

Al igual que con los operadores matemáticos que vimos anteriormente, puedes ocupar los operadores lógicos con variables. Mira el siguiente ejemplo:

```
#varias operaciones lógicas
valor_a = 1
valor_b = 2

res = valor_a == valor_a
print(res)
res = valor_b != valor_b
print(res)
res = valor_a < valor_b
print(res)
res = valor_a > valor_b
print(res)
res = valor_a <= valor_a
print(res)
res = valor_a >= valor_b
print(res)
```

```
True
False
True
False
True
False
```

Operador Lógico NOT

El operador lógico `not` sirve para poder "cambiar" el valor de una operación lógica. Es decir, si uno le aplica este operador a una operación lógica, cambia el valor de TRUE a FALSE o viceversa. Por ejemplo:

```
#ejemplo operador lógico not
print(3>=2)
print(not 3>=2)

print(1!=1)
print(not 1!=1)
```

```
True
False
False
True
```

Operadores Lógicos

Volvamos al operador lógico `<=`

Este operador se llama literalmente “menor o igual que”. Esto significa que al hacer las siguientes operaciones:

```
1<=1
```

```
1<=2
```

¡Ambas son verdad!

Operadores Lógicos

Volvamos al operador lógico `<=`

Este operador se llama literalmente “menor o igual que”. Esto significa que al hacer las siguientes operaciones:

```
1<=1
1<=2
```

¡Ambas son verdad!

La razón de esto es porque tal como dice el nombre, para que se cumpla la operación que dice el operador se debe cumplir que sea menor **ó** que sea igual. Si se cumple cualquiera de las dos condiciones entonces la operación completa es verdadera.

Pausa

5 min.

Operadores Lógicos Binarios

Muchas veces una sola operación lógica no es suficiente. Para esto existen los operadores lógicos binarios.

Estos operadores toman dos operaciones lógicas y entregan un resultado en base a sus valores.

Partiremos introduciendo el operador lógico binario OR.

OR

Compara dos valores y entrega un resultado. Para esto, ocupa el operador or. Si al menos uno de los valores es TRUE, entonces el resultado total también lo es. En esta tabla se resume lo anterior:

A	B	A or B
TRUE	TRUE	TRUE
TRUE	FALSE	TRUE
FALSE	TRUE	TRUE
FALSE	FALSE	FALSE

OR

Veamos un ejemplo:

```
#ejemplo operador binario or
a=True
b=False

print("a or b")
print("True or False")
print(a or b)

print("a or a")
print("True or True")
print(a or a)

print("b or b")
print("False or False")
print(b or b)
```

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

OR

El verdadero uso práctico de esta función es que las variable con las que ocupamos OR sean operaciones lógicas. Por ejemplo, digamos que queremos saber si un número no pertenece al intervalo (3,10) ¿Cómo podríamos escribir un programa que resolviera este problema?

OR

```python
#ejemplo operador binario or
numero = int(input("Ingrese el número que desea comprobar que no pertenezca al intervalo (3,10)\n"))
menor_que_3 = numero <= 3
mayor_que_10 = numero >= 10
print(menor_que_3 or mayor_que_10)
```

```
Ingrese el número que desea comprobar que no pertenezca al intervalo (3,10)
 5
False
```

```
Ingrese el número que desea comprobar que no pertenezca al intervalo (3,10)
 10
True
```

```
Ingrese el número que desea comprobar que no pertenezca al intervalo (3,10)
 -15
True
```

AND

Compara dos valores y entrega un resultado. Para esto, ocupa el operador and. Si ambos valores son TRUE, entonces el resultado total también lo es. En esta tabla se resume lo anterior:

A	B	A and B
TRUE	TRUE	TRUE
TRUE	FALSE	FALSE
FALSE	TRUE	FALSE
FALSE	FALSE	FALSE

AND

Veamos otro ejemplo:

```
#ejemplo operador binario and
a=True
b=False

print("a and b")
print("True and False")
print(a and b)

print("a and a")
print("True and True")
print(a and a)

print("b and b")
print("False and False")
print(b and b)
```

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

AND

Volvamos al ejercicio anterior. Por ejemplo, queremos saber si un número pertenece al intervalo [3,10).

AND

```
#ejemplo 2 operador binario and
numero = int(input("Ingrese el número que desea comprobar si pertenece al intervalo [3,10)\n"))
mayor_que_3 = numero >= 3
menor_que_10 = numero < 10
print(menor_que_10 and mayor_que_3)
```

```
Ingrese el número que desea comprobar si pertenece al intervalo [3,10)
 3
True
```

```
Ingrese el número que desea comprobar si pertenece al intervalo [3,10)
 10
False
```

```
Ingrese el número que desea comprobar si pertenece al intervalo [3,10)
 -15
False
```

Ejercicio: Cálculo de promedio final

La nota final (NF) del curso se calcula en base a la siguiente fórmula:

$$\boldsymbol{NF} = 15\% \cdot \boldsymbol{I1} + 15\% \cdot \boldsymbol{I2} + 30\% \cdot \boldsymbol{E} + 10\% \cdot \boldsymbol{T1} + 10\% \cdot \boldsymbol{T2} + 10\% \cdot \boldsymbol{T3} + 10\% \cdot \boldsymbol{NP}$$

- Promedio de evaluaciones:

$$\boldsymbol{NE} = \frac{(15\% \cdot \boldsymbol{I1} + 15\% \cdot \boldsymbol{I2} + 30\% \cdot \boldsymbol{E})}{60\%} \geq 4.0$$

- Promedio de tareas:

$$\boldsymbol{NT} = \frac{(10\% \cdot \boldsymbol{T1} + 10\% \cdot \boldsymbol{T2} + 10\% \cdot \boldsymbol{T3})}{30\%} \geq 4.0$$

Y obviamente, NF>=4.0

Ejercicio: Cálculo de promedio final

La nota final (NF) del curso se calcula en base a la siguiente fórmula:

$$\boldsymbol{NF} = 15\% \cdot \boldsymbol{I1} + 15\% \cdot \boldsymbol{I2} + 30\% \cdot \boldsymbol{E} + 10\% \cdot \boldsymbol{T1} + 10\% \cdot \boldsymbol{T2} + 10\% \cdot \boldsymbol{T3} + 10\% \cdot \boldsymbol{NP}$$

- Promedio de evaluaciones:

$$\boldsymbol{NE} = \frac{(15\% \cdot \boldsymbol{I1} + 15\% \cdot \boldsymbol{I2} + 30\% \cdot \boldsymbol{E})}{60\%} \geq 4.0$$

Escribe un código que en base a las variables: i1, i2, ex calcule la nota de evaluación. Luego, y por medio de una operación lógica, pueda determinar si es que la nota de evaluación es mayor o igual que 4.0

Ejercicio: Cálculo de promedio final

- Promedio de evaluaciones:

```
#cálculo promedio de evaluaciones
ne = ((0.15*i1+0.15*i2+0.3*ex)/.6)
ne_cumple = ne >= 4.0
print("Tu nota de evaluación (NE) es",ne,"¿Apruebas el
requisito?",ne_cumple)
```

```
Tu nota de evaluación (NE) es 4.000000000000001 ¿Apruebas?
True
```

```
#ejemplo: inputs
i1=4.4
i2=3.6
ex=4
t1=1.0
t2=5
t3=7.0
np=1
```

Ejercicio: Cálculo de promedio final

- Promedio de tareas:

```
#cálculo promedio de tareas
nt = ((0.1*t1+0.1*t2+0.1*t3)/0.3)
nt_cumple = nt >= 4.0
print("Tu nota de tareas (nt) es",nt,"¿Apruebas el
requisito?",nt_cumple)
```

```
Tu nota de tareas (nt) es 4.333333333333334 ¿Apruebas? True
```

```
#ejemplo: inputs
i1=4.4
i2=3.6
ex=4
t1=1.0
t2=5
t3=7.0
np=1
```

Ejercicio: Cálculo de promedio final

- Promedio final:

```
#cálculo promedio final
pf = 0.6*ne+0.3*nt+0.1*np
pf_cumple = pf >= 4.0
print("Tu promedio final (pf) es",pf,"¿Apruebas el
requisito?",pf_cumple)

pasar_el_ramo = ne_cumple and nt_cumple and pf_cumple
print("¿Pasaste?",pasar_el_ramo)
```

```
Tu promedio final (pf) es 3.8000000000000003 ¿Apruebas?
False
¿Pasaste? False
```

```
#ejemplo: inputs
i1=4.4
i2=3.6
ex=4
t1=1.0
t2=5
t3=7.0
np=1
```

Ejercicio: Cálculo de promedio final

- Código completo:

```
#ejemplo: cálculo promedio final
i1=4.4
i2=3.6
ex=4
t1=1.0
t2=5
t3=7.0
np=1

ne = ((0.15*i1+0.15*i2+0.3*ex)/.6)
ne_cumple = ne >= 4.0
print("Tu nota de evaluación (NE) es",ne,"¿Apruebas el requisito?")

nt = ((0.1*t1+0.1*t2+0.1*t3)/0.3)
nt_cumple = nt >= 4.0
print("Tu nota de tareas (NT) es",nt,"¿Apruebas el requisito?",nt_cumple)

pf = 0.6*ne+0.3*nt+0.1*np
pf_cumple = pf >= 4.0
print("Tu promedio final (PF) es",pf,"¿Apruebas el requisito?",pf_cumple)

pasar_el_ramo = ne_cumple and nt_cumple and pf_cumple
print("¿Pasaste?",pasar_el_ramo)
```

Resumen de la clase

- **Variable booleana:** Puede tener solo dos valores (`True o False`)
- **Operadores lógicos:** Vimos los operadores ==,!=,`<`,`>`,`<=`,`>=`
- **Operador not:** Sirve para cambiar el valor de una variable u operación booleana
- **Operadores lógicos binarios:** Sirve para comparar dos variables booleanas
  - `or`: Basta con que uno de los valores sea `True`
  - `and`: Ambos valores deben ser `True`

Links

- https://repl.it/@FelipeLopez/IIC1103OperacionesLogicas que contiene todos los ejemplos cortos de la clase.

Bibliografía

- http://runest.ing.puc.cl/if.html#construccion-de-condiciones
- A. B. Downey. Think Python: How to think like a computer scientist. Green Tea Press, 2013 -> Capítulo5

Clase 3 – Operaciones Lógicas

IIC1103 Sección 9 – 2019-2

Profesor: Felipe López

Fecha: 22 de agosto del 2019