Curso de Python

2 – Tipos básicos de datos

Ramón Invarato Menéndez

Ricardo Moya García

String

Tipo de dato para trabajar con el texto

- Los valores String en Python 3 se almacenan en formato **Unicode**
- Los **String** en Python se **delimitan con 'comillas simples'** o **"dobles comillas"**

Código

```
comillas_simples = 'Un texto'
comillas_dobles = "un texto"
```

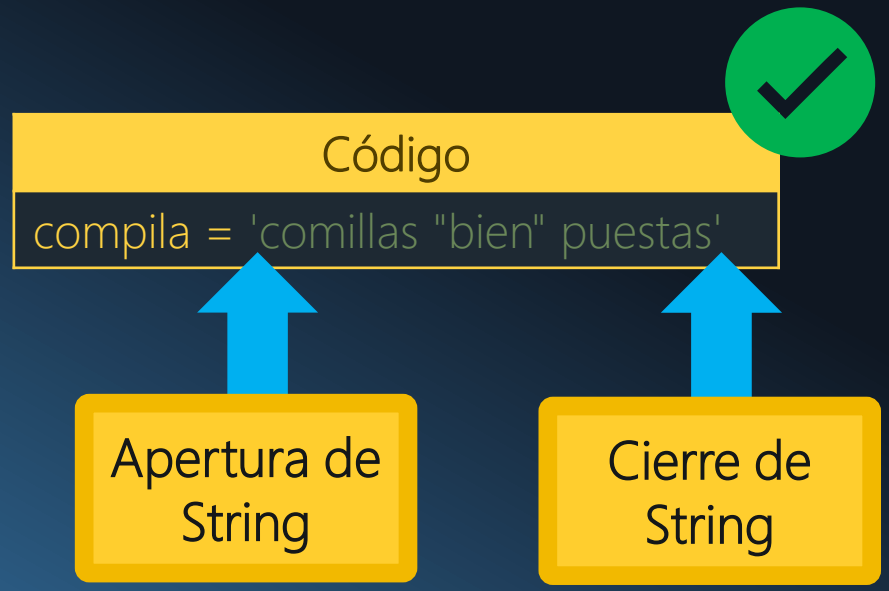
- Se pueden **convertir otros tipos a String con** el método **str**(objeto):

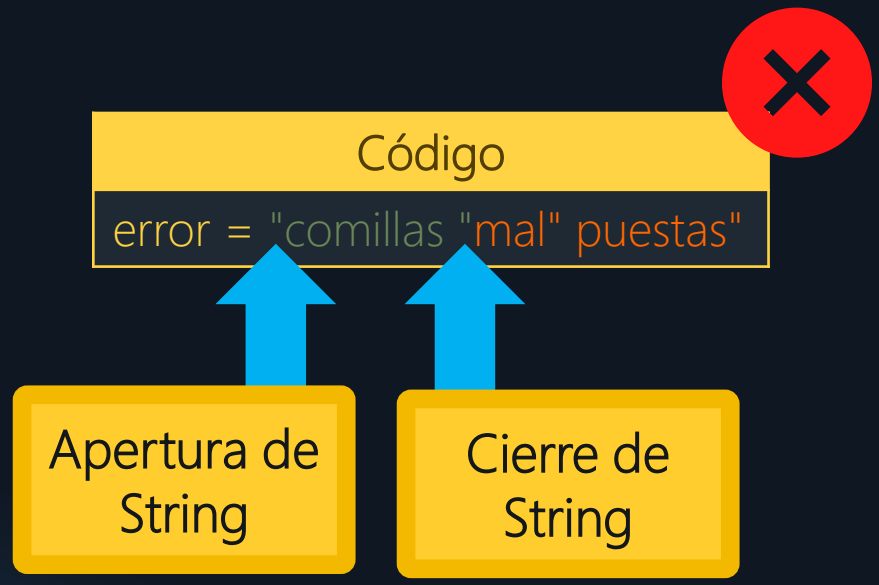
Código

```
numero_como_string = str(123)
```

String

Delimitar correctamente un valor String con las comillas

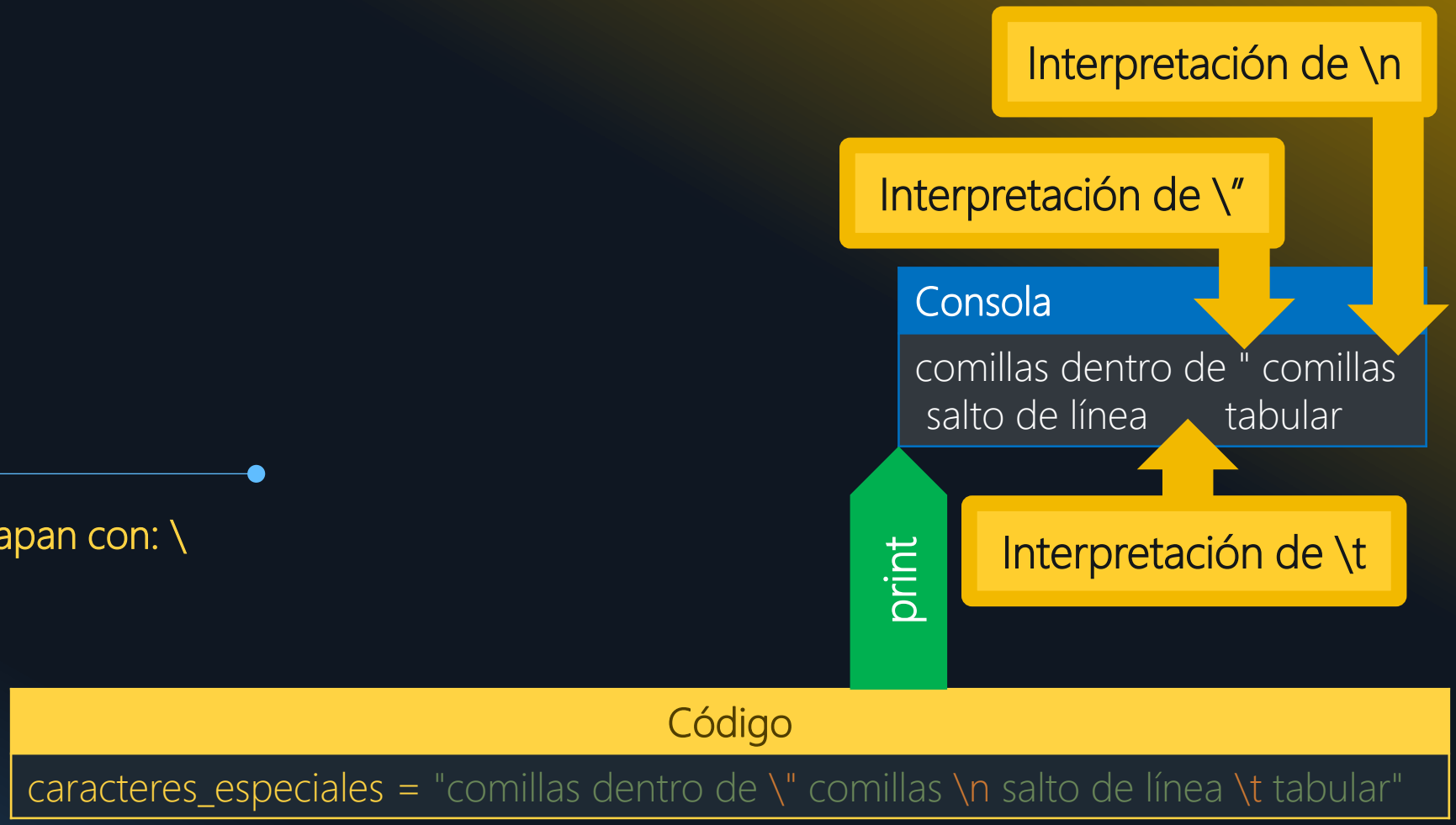




ASCII

Los caracteres ASCII se escapan con: \

- \n: Salto de línea (LF)
- \t: Tabular (TAB)
- \': Comilla simple
- \": Comilla doble
- \\: Barra invertida



Truco para cumplir el PEP8

Código en varias líneas

- En Python se puede **separar un código muy largo en varias líneas si se usa la \ al final de cada una de las líneas del código**.
- En un **String NO hay que confundir esta separación de líneas** que se escribe fuera del valor (fuera de las comillas), **con** el carácter de salto de línea (\n) que se escribe dentro del valor (dentro de las comillas)

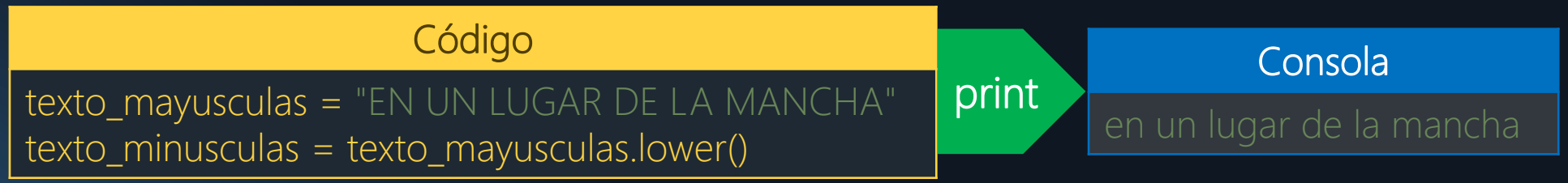
Código

```
string_en_varias_lineas = "este es un texto largo " \
                          "que he escrito en varias líneas " \
                          "para que sea más legible, " \
                          "pero no se imprime en varias líneas " \
                          "sino se usa>\n<otra línea"
```

Lower

Letras a minúsculas

```
letras_minuculas = "LETRAS MAYÚSCULAS".lower()
```



Upper

Letras a mayúsculas

```
letras_mayusculas = "letras minúsculas".upper()
```

Código

```
texto_minusculas = "en un lugar de la mancha"
texto_mayusculas = texto_minusculas.upper()
```

print

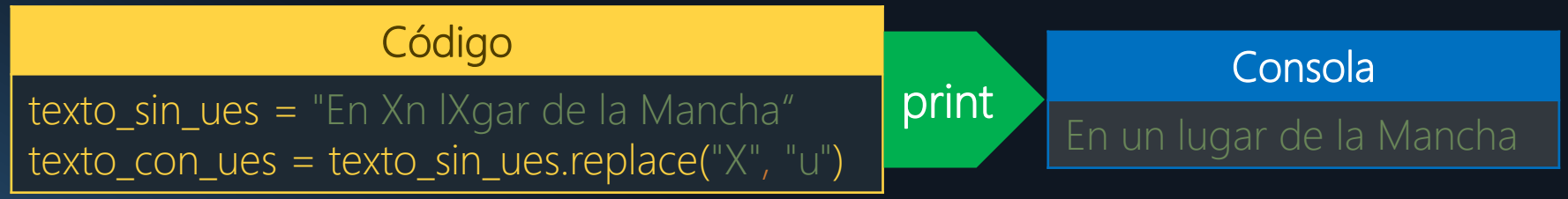
Consola

```
EN UN LUGAR DE LA MANCHA
```

Replace

Remplazar un texto por otro

```
texto_remplazado = "texto completo original".replace("texto a remplazar", "texto por el que remplazar")
```



Strip

Limpiar espacios y saltos de líneapor delante y por detrás del texto

texto_sin_espacios_de_mas = " texto con espacios de más ".strip()

Código de la plantilla

```
plantilla = "Estimado {cliente}:\n{empresa} se complace en presentarle el nuevo {producto}.\nAtentamente,\n\n{jefe}."
```

Format

Aplicar un formato al texto

```
texto_formateado = "Plantilla de texto {variable_de_plantilla}".format(variable_de_plantilla="Valor para la variable")
```

Código

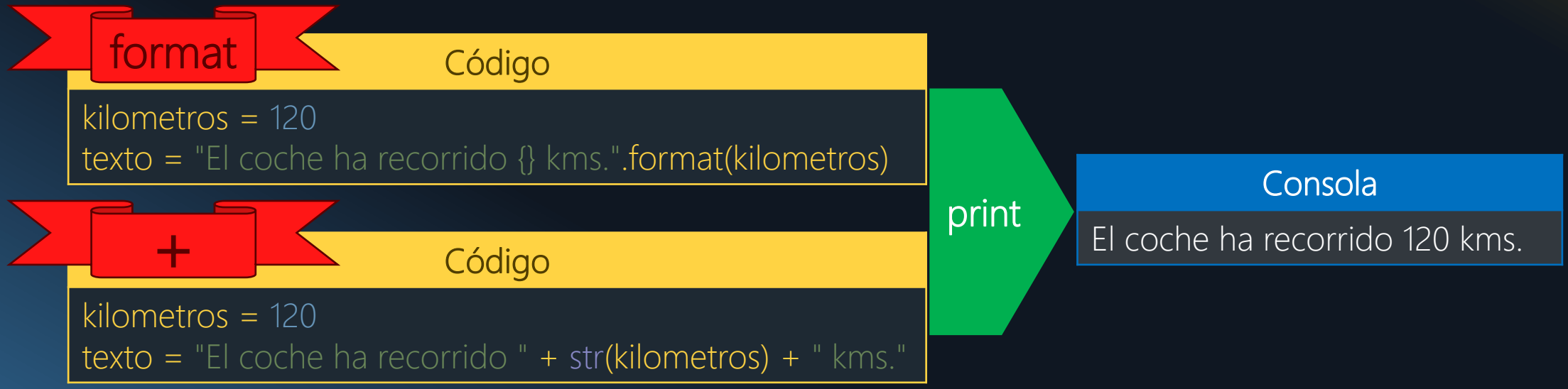
```
texto_limpio= plantilla.format(cliente="Juan Sánchez",
                               empresa="Google",
                               producto="buscador",
                               jefe="Larry Page")
```

print

Consola

```
Estimado Juan Sánchez:
Google se complace en presentarle el nuevo buscador.
Atentamente,

Larry Page.
```

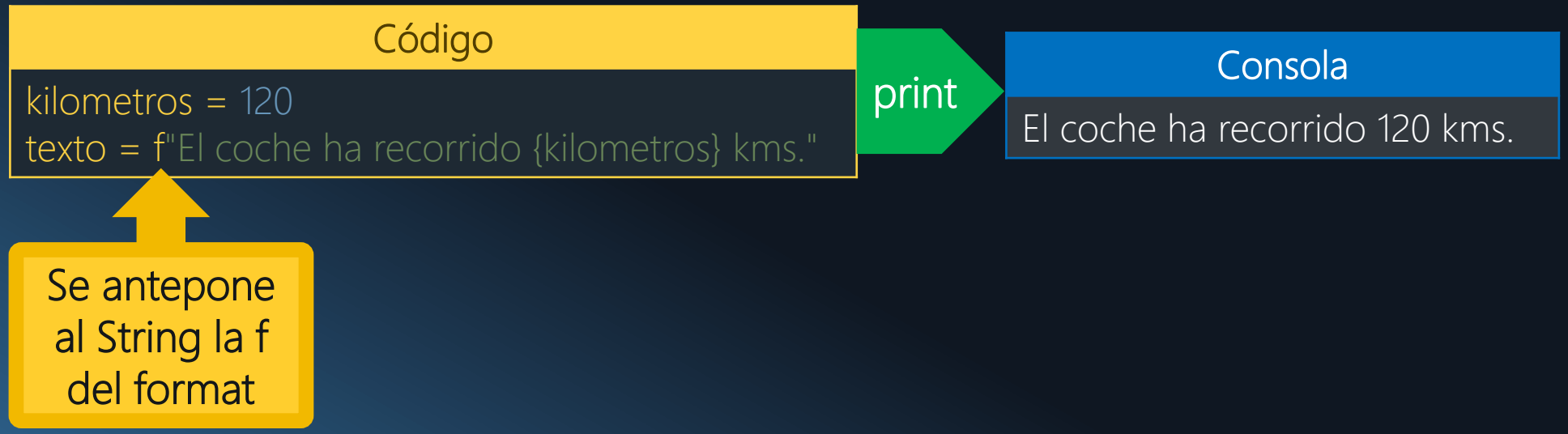
Format vs concatenación de Strings (+)



f-String (Interpolación de literales en un String)

Format simplificado

```
texto_formateado = f"Plantilla de texto {variable}"
```



Métodos útiles para trabajar con String

Tipo de dato de texto

- Realizando la **operación suma con Strings se pueden concatenar** fácilmente

Código

```
textos_concatenados = "super" + "heroe"
```

- El método "**format**" facilita la creación de **Strings complejos** por el método de la sustitución de variables

Código

```
texto_formateado = "{} vive en una {}".format("Juan", "casa")
texto_formateado = "{nombre} vive en una {objeto}".format(nombre="Juan", objeto="casa")
```

- Se puede convertir un **String a bytes con "encode" y viceversa con "decode"** (y se puede cambiar el formato)

Código

```
texto_unicode = "Avión"
texto_a_bytes = texto_unicode.encode("utf-8")
texto_windows1252 = texto_a_bytes.decode("windows-1252")
```

- Se puede convertir **desde otro tipo a String** (siempre que implemente __str__): **str(variable)**

Código

```
numero_como_string = str(123)
```

int

Número entero

- Los números enteros son todos los positivos, negativos y el cero, sin decimales (sin comas)
- Los números enteros en Python 3 pueden ser tan largos como queramos (son int y long)
- Utilizaremos **int(valor)** si queremos **convertir cualquier otro valor a entero** (siempre que sea permitido)

Código

```
texto = "100"
numero = 100
```

Código

```
numero_entero = 123
numero_entero = 0
numero_entero = -5
```

Código

```
numero_texto = "2020"
numero_entero = int(numero_texto)
```

Código

```
numero_float = 2.37478
numero_float = 100.0
numero_float = -5.3
```

Código

```
numero_texto = "13.14"
numero_float = float(numero_texto)
```

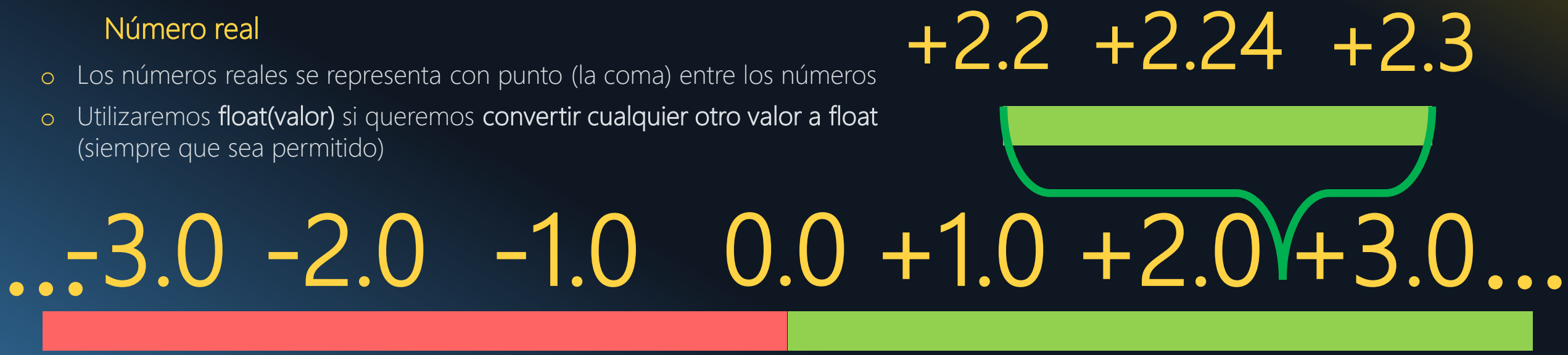
Código

```
numero_entero = 123
numero_float = float(numero_entero)
```

float

Número real

- Los números reales se representa con punto (la coma) entre los números
- Utilizaremos **float(valor)** si queremos **convertir cualquier otro valor a float** (siempre que sea permitido)



Operadores aritméticos

Operadores aritméticos: Suma

resultado = sumando_1 + sumando_2

Código

```
billetes_a = 3
billetes_b = 2

total_billetes = billetes_a + billetes_b
total_billetes
```

print

Consola

```
5
```

Operadores aritméticos: Resta

resultado = minuendo - sustraendo

Código

```
billetes = 2
factura = 3

total_billetes = billetes – factura
total_billetes
```

print

Consola

```
-1
```

Operadores aritméticos: Multiplicación

resultado = multiplicando * multiplicador

```
Código
billetes = 3
sacos = 2

total_billetes = billetes * sacos
total_billetes
```

print

```
Consola
6
```

Operadores aritméticos: Exponente

resultado = base ** exponente

Código

```
billetes_a = 3
billetes_b = 2

total_billetes = billetes_a ** billetes_b
total_billetes
```

print

Consola

```
9
```

- El **exponente** es el "**elevado a**". Se representa con ** (no con ^)

Operadores aritméticos: División

cociente = divisor / dividendo


```
Código
billetes = 7
personas = 2

total_billetes = billetes / personas
total_billetes
```

print

Consola
3.5

Operadores aritméticos: División Entera

cociente = divisor // dividendo

```
Código
billetes = 7
personas = 2

total_billetes = billetes / personas
total_billetes
```

print

```
Consola
3
```

Operadores aritméticos: Módulo

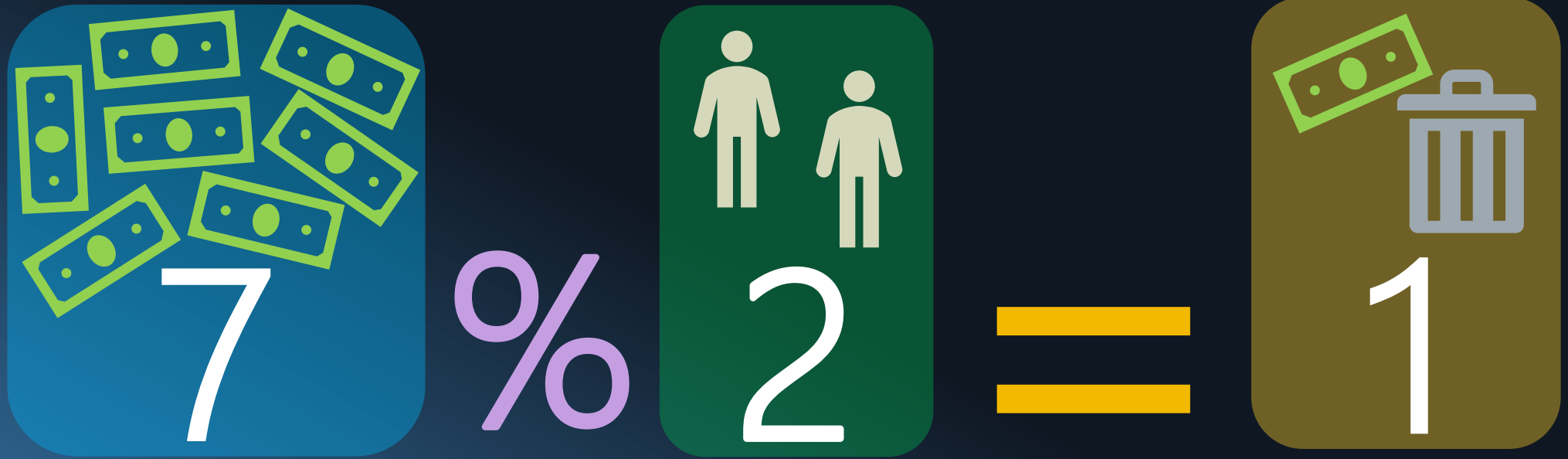
resto = divisor % dividendo

```
Código
billetes = 7
personas = 2

total_billetes = billetes % personas
total_billetes
```

print

```
Consola
1
```

- El módulo es el resto de la división entera



Operadores aritméticos: Módulo

resto = divisor % dividendo

Tabla módulo 3

Operación	Resto	Deducción
1 % 3	1	Rojo
2 % 3	2	Verde
3 % 3	0	Azul
4 % 3	1	Rojo
5 % 3	2	Verde
6 % 3	0	Azul
7 % 3	1	Rojo

Tabla módulo 2

Operación	Resto	Deducción
1 % 2	1	Impar
2 % 2	0	Par
3 % 2	1	Impar
4 % 2	0	Par
5 % 2	1	Impar
6 % 2	0	Par
7 % 2	1	Impar
…	…	…

Resumen de los operadores aritméticos

Operador	Descripción	Ejemplo	Resultado
A + B	Suma	2 + 3	5
A - B	Resta	2 - 3	-1
-A	Negación	-6	-6
A * B	Multiplicación	2 * 3	6
A ** B	Potencia	2 ** 3	8
A / B	División	5.0 / 2.0	2.5
A // B	División Entera	5.0 // 2.0	2
A % B	Módulo	5 % 2	1

Orden de precedencia de operadores

- Ejemplo de precedencia:

Código

```
resultado = 4 * 3 + 2
resultado = 2 + 3 * 4
```

print

Consola

```
14
```

- Mismo nivel de precedencia se calcula de izquierda a derecha:

Código

```
resultado = 5 + 3 - 2
```

print

Consola

```
6
```

- Los paréntesis siempre se resuelven de dentro hacia fuera:

Código

```
resultado = (4 * (3 + 2))
```

print

Consola

```
20
```

Booleanos

Booleano o Tipo de dato lógico

- **Booleano** o Tipo de dato lógico:
  - **True**: se considera como verdad/encendido
  - **False**: se considera como mentira/apagado
- Se puede inicializar un booleano con el método bool() cuyo valor pasado se evaluará como un contexto de veracidad resultando en True o False.

código

```
bool("algo") == True
bool("") == False
```

código

```
variable = False
```

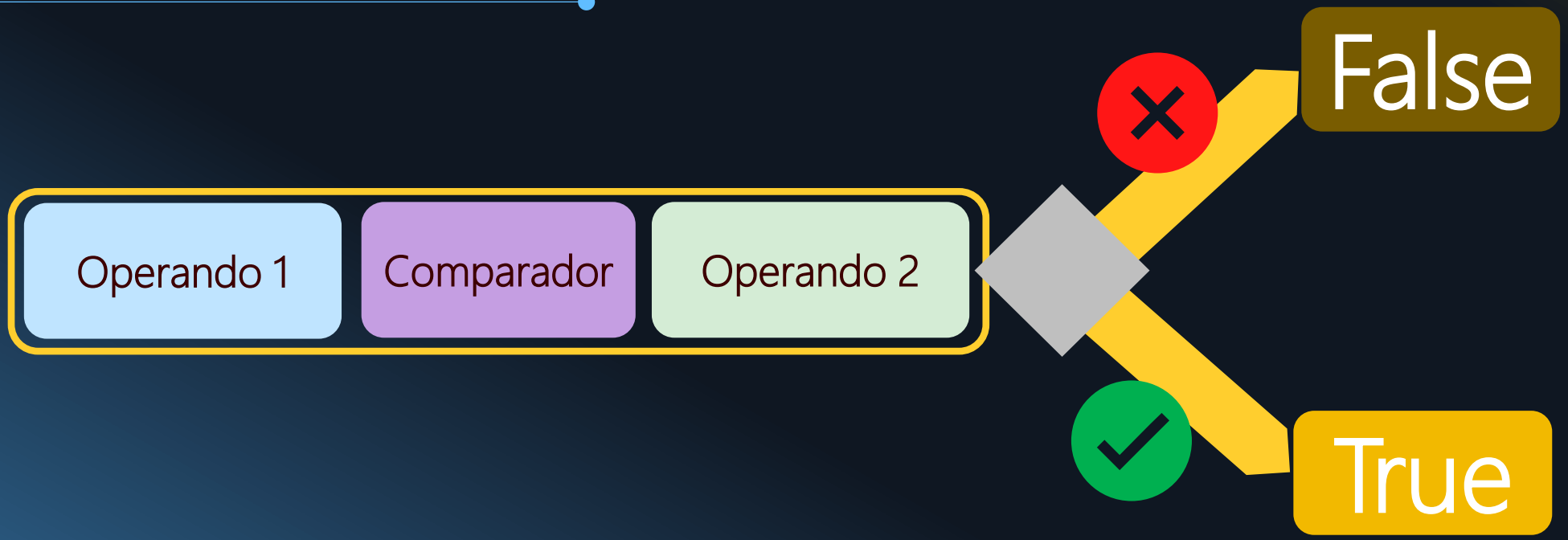
False

código

```
variable = True
```

True

Comparadores



Comparadores: Igualdad

verdad = a == b

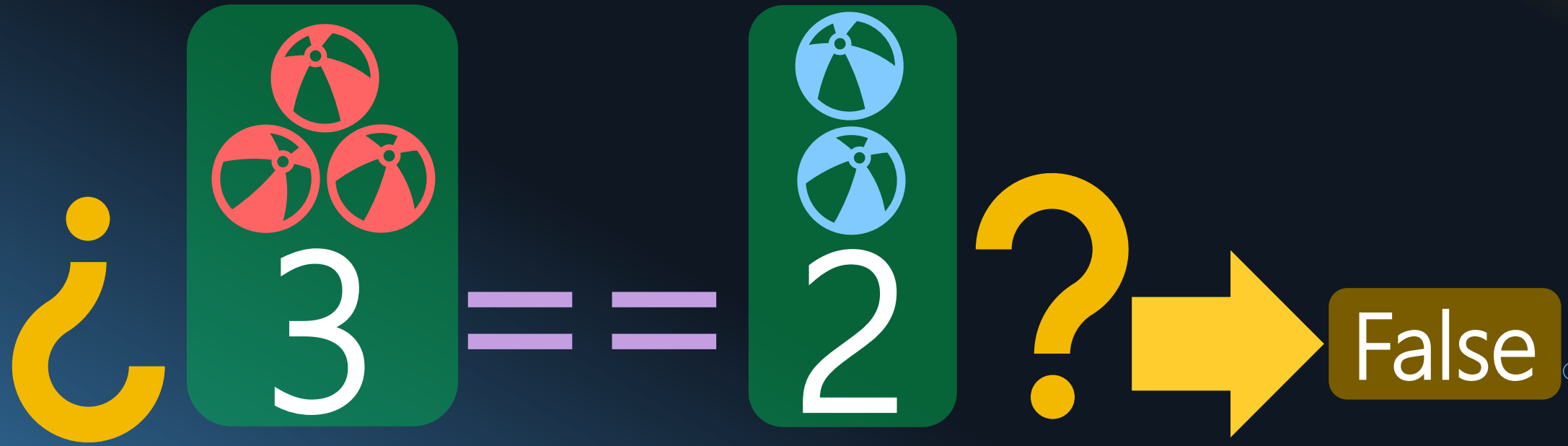
Código

```
pelotas_a = 3
pelotas_b = 2

es_verdad = pelotas_a == pelotas_b
es_verdad
```

print

Consola

```
False
```



Comparadores: Distinción

verdad = a != b

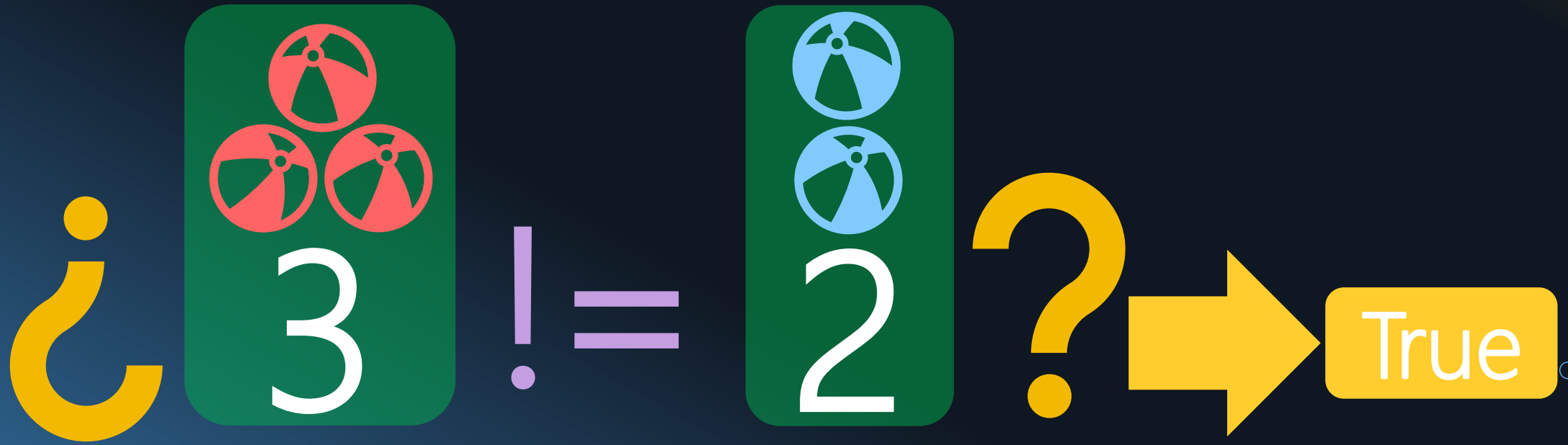
Código

```
pelotas_a = 3
pelotas_b = 2

es_verdad = pelotas_a != pelotas_b
es_verdad
```

print

Consola

True



Comparadores: mayor que

verdad = A > b

Código

```
pelotas_a = 3
pelotas_b = 2

es_verdad = pelotas_a > pelotas_b
es_verdad
```

print

Consola

```
True
```

Comparadores: menor que

verdad = a < B

Código

```
pelotas_a = 3
pelotas_b = 2

es_verdad = pelotas_a < pelotas_b
es_verdad
```

print

Consola

```
False
```

Comparadores: mayor o igual que

verdad = A >= b

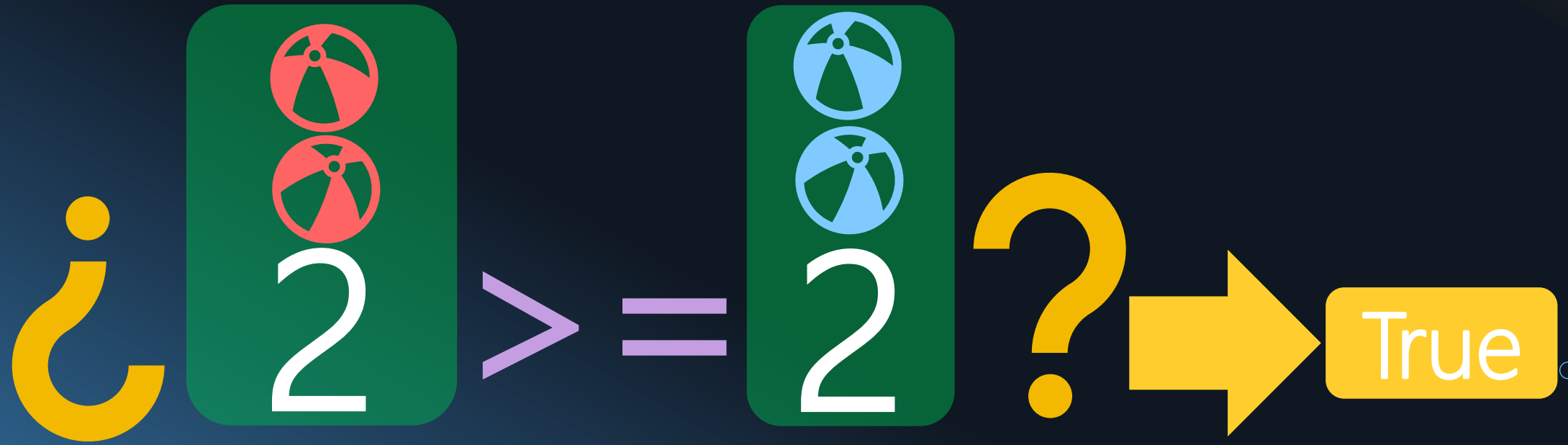
Código

```
pelotas_a = 2
pelotas_b = 2

es_verdad = pelotas_a >= pelotas_b
es_verdad
```

print

Consola

True



Comparadores: menor o igual que

verdad = a <= B

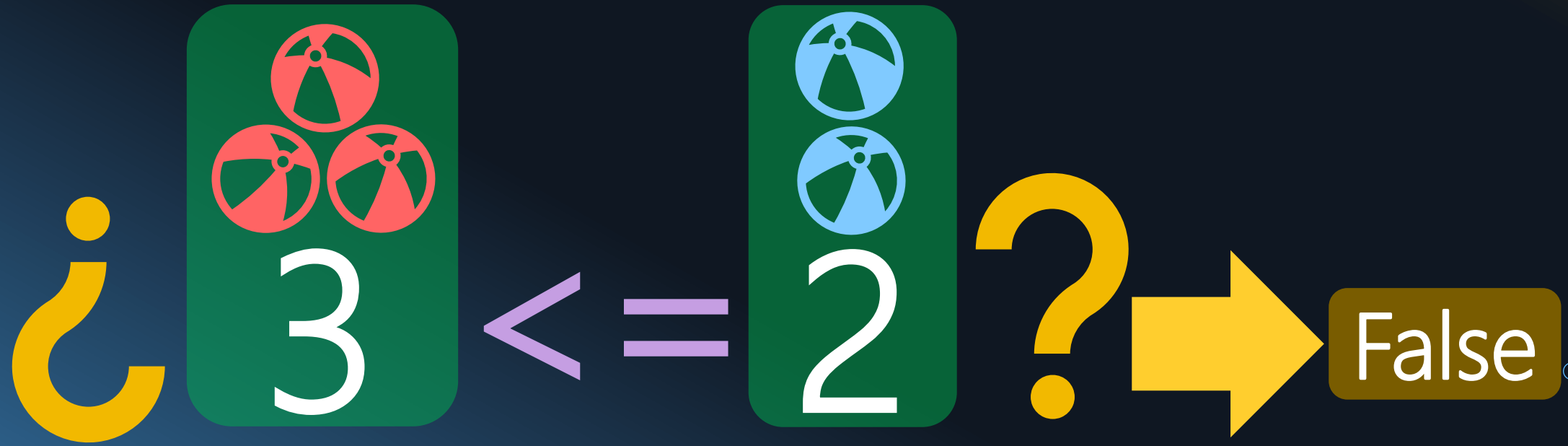
Código

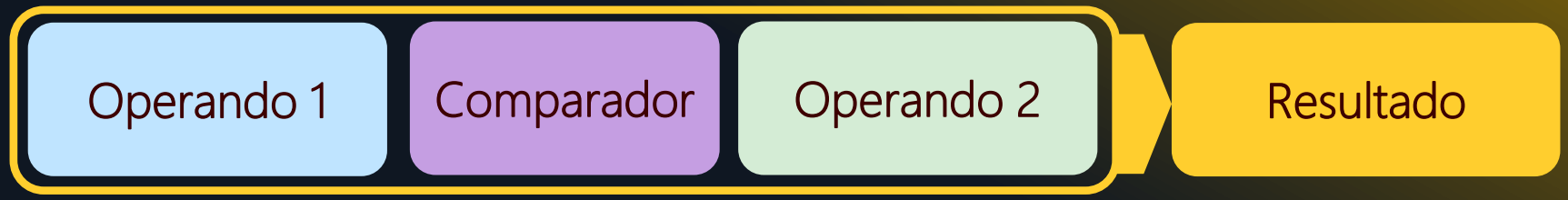
```
pelotas_a = 3
pelotas_b = 2

es_verdad = pelotas_a <= pelotas_b
es_verdad
```

print

Consola

```
False
```





Resumen de Comparadores

Operador	Descripción	Ejemplo	Resultado
A == B	¿iguales A y B?	3 == 2	False
A != B	¿distintos A y B?	3 != 2	True
A < B	¿A es menor que B?	3 < 2	False
A > B	¿A es mayor que B?	3 > 2	True
A <= B	¿A menor o igual que B?	3 <= 3	True
A >= B	¿A mayor o igual a B?	5 >= 3	True

Comparar otros tipos: String

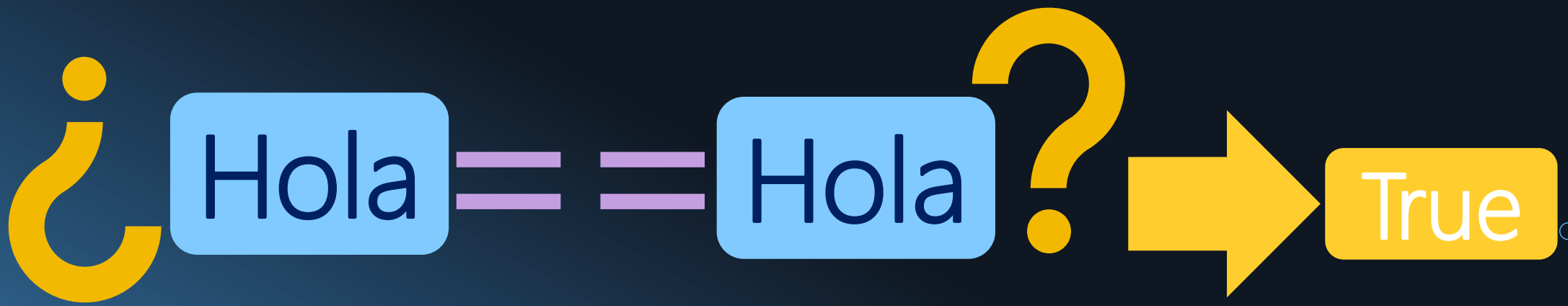
Código

```
texto_a = "Hola"
texto_b = "Hola"

es_verdad = texto_a == texto_b
es_verdad
```

print

Consola

```
True
```



Comparar otros tipos: String

Código

```
texto_a = "A"
texto_b = "B"

en_orden_arfabetico = texto_a < texto_b
en_orden_arfabetico
```

print

Consola

True

Binario	Dec	Hex	Representación
0100 0000	64	40	@
0100 0001	65	41	A
0100 0010	66	42	B
0100 0011	67	43	C

https://es.wikipedia.org/wiki/ASCII

Tablas de verdad

Variable

A	A
True	True
False	False

Negación, NOT (no)

A	not A
True	False
False	True

Conjunción, AND (y)

A	B	A and B
True	True	True
True	False	False
False	True	False
False	False	False

Disyunción, OR (o)

A	B	A or B
True	True	True
True	False	True
False	True	True
False	False	False

(Proposición B) Operador Proposición A → Valor de verdad

Resumen de las tablas de verdad

- La negación es el valor contrario
- **And** solo es **True** cuando las dos proposiciones son **True**
- **Or** solo es **False** cuando las dos proposiciones son **False**

Operador	Descripción	Ejemplo	Resultado
A **and** B	¿se cumple A y B?	True and False	False
A **or** B	¿se cumple A o B?	True or False	True
not A	No A	not True	False

¡GRACIAS!

Web: **https://jarroba.com/**

Ramón Invarato Menéndez

Linked-in
https://www.linkedin.com/in/rinvarato/

Github
https://github.com/Invarato

Ricardo Moya García

Linked-in
https://www.linkedin.com/in/phdricardomoya

Github
https://github.com/RicardoMoya