Condiciones y ciclos

Condiciones

- Se ejecuta o no al cumplirse una condición
- Dependen de una operación lógica

Condiciones lógicas

- Igual ==
- Diferencia ≠

Nos permiten comparar si dos objetos tienen el mismo valor, longitud o son del mismo tipo.

Para variables numéricas utilizamos:

- Menor que <
- Mayor que >
- Menor o igual que ≤
- Mayor o igual que ≥

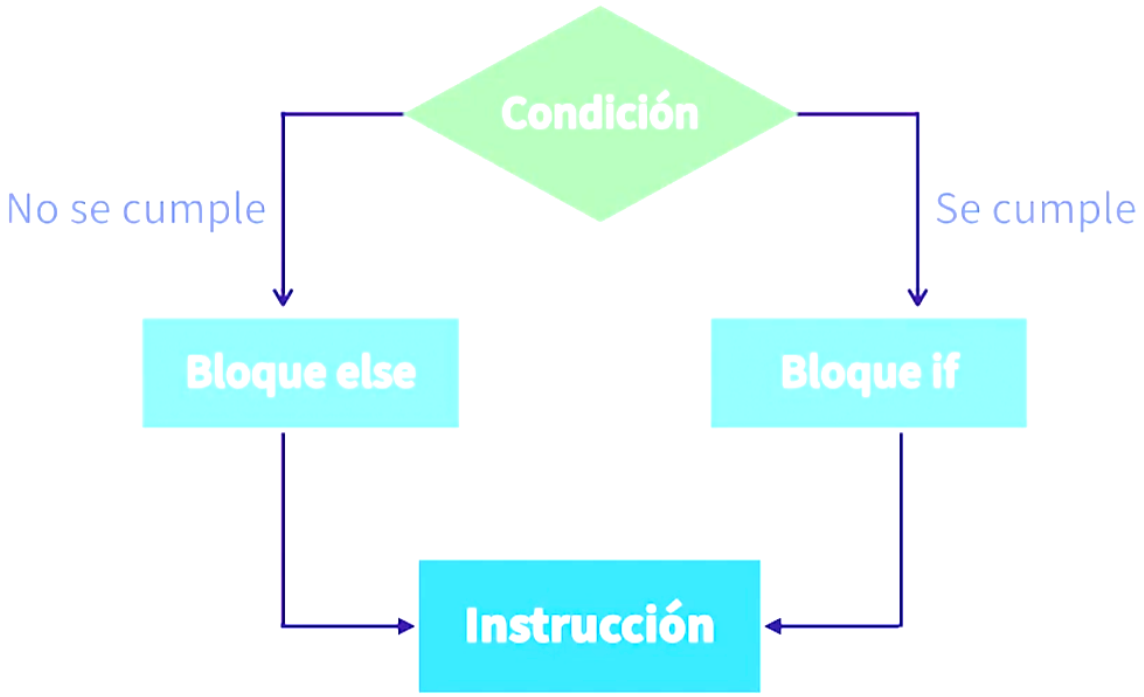
Retornan una variable *bool* al evaluar la condición. Por ejemplo:

```
2 < 3 # Da False
```

Expresiones condicionales

- is (si dos variables se refieren al mismo objeto)
- and (se utiliza para unir dos o más condiciones, si todas se cumplen devuelve true, sino devuelve false)
- or (se utiliza para unir dos o más condiciones, si al menos una se cumple devuelve true)
- not (retorna true si es valor que comprueba no es verdadero, o saber si un elemento está o no contenido en una lista o estructura de datos)

Esquema de condiciones:



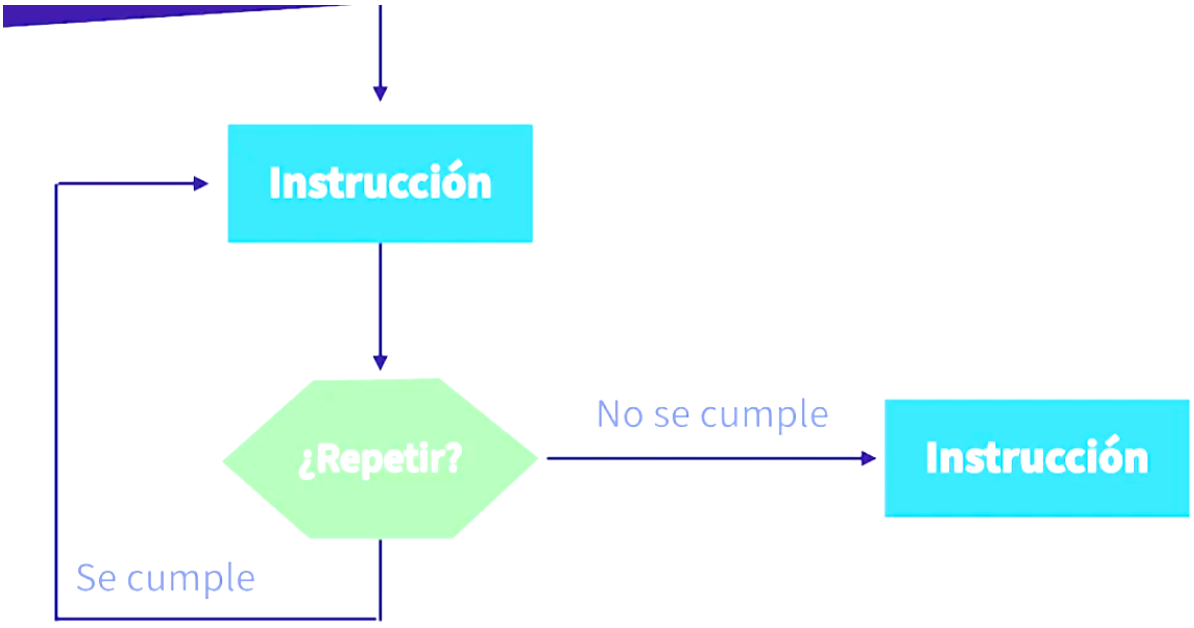
Sintaxis "*if*" - "*elif*":

```
if <condición lógica>:
    print("si se cumple la condición)
```

```
elif <condición lógica>:
    print("si se cumple otra condición distinta") # Pueden ha
else:
    print("si ninguna condición de las anteriores se cumple")
```

Bucles

- Instrucciones que se repiten hasta que se cumple una condición
- Hay 2 tipos:
  - For
  - While



Sintaxis ciclo *for*:

```
for <element> in <object>: # object puede ser una tupla, set,
    print("Elemento:" <element>) # código a ejecutar en cada
```

Sintaxis ciclo *while*:

```
while <condicion>:
    print("Código")
```

Es necesario tener una instrucción para detener este ciclo.

Condiciones if

```
n_one = 1
n_two = 2

if n_one < n_two:
    print("A es menor que B") # Se printeará ya que 1 es meno
```

```
n_one = 1
n_two = 2

if n_one > n_two:
    print("1 es mayor que 2")
else:
    print("2 es mayor que 1") # Se ejecutará la lógica del el
```

```
n_one = 1
n_two = 2
n_three = 3

if n_one > n_three:
    print("1 es mayor que 2")
elif n_three > n_one:
```

```
    print("3 es mayor que 1");
else:
    print("Las anteriores condiciones no se cumplieron") # Se
```

```
n_one = 1
n_two = 2
n_three = 3

if n_one < n_two and n_two < n_three: # Se cumplen ambas
    print("Yes!")
else:
    print("No!")
```

```
n_one = 1
n_two = 2
n_three = 3

if n_one > n_two or n_two < n_three: # Se cumple una, así que
    print("Yes!")
else:
    print("No!")
```

Bucle for

```
for char in "text":
    print(char) # Printeará t e x t o
```

```
languages = ["python", "java", "golang"]
```

```
for lang in languages:
    print(lang) # Printeará cada elemento de la lista
```

Modificando el flujo del ciclo con break / continue:

```
languages = ["python", "java", "golang"]

for lang in languages:
    print(lang) # Printeará cada elemento de la lista
    break # Se romperá con la primera iteración
```

```
languages = ["python", "java", "golang"]

for lang in languages:
    print(lang) # Printeará cada elemento de la lista
    if lang == "java":
        break # Se romperá cuando llegue al elemento java
```

```
languages = ["python", "java", "golang"]

for lang in languages:
    if lang == "java":
        continue # Saltará el elemento java, pasará directame
```

Función range:

```
for el in range(5):
    print(el) # Esto imprimirá: 0, 1, 2, 3, 4 -> la función r
```

```
for el in range(1, 5): # Ahora especificamos un rango de inic
    print(el) # Esto imprimirá: 1, 2, 3, 4 -> la función rang
```

While

- Ejecuta 1 o más instrucciones mientras se cumple 1 condición
- Necesita un contador para saber cuándo parar el ciclo

```
count = 1

while count <= 5:
    print(count)
    count += 1 # Ejecutará 1, 2, 3, 4, 5 ya que count empezab
```

Utilizando la función *break*:

```
count = 1

while count <= 5:
    print(count)
    count += 1
    if count == 3:
        break; # El while finaliará cuando el count valga 3
```

Iterar una lista

```
languages = ["python", "java", "golang"]

# Utilizando un bucle for
for lang in languages:
    print(lang)

# Utilizando índices
for index in range(len(languages)):
    print("index: ", index)
    print("element: ", languages[index])

# Utilizando un bucle while
count = 0
while count < len(languages):
    print(languages[count])
```

Iterar un diccionario

```
language = {
    "name": "python",
    "author": "Guido"
}

# Utilizando el bucle for
for key in language:
    print("Key: ", key)
    print("Value: ", language[key])

# Utilizando la función keys
for el in language.keys():
    print(el) # Imprime las llaves del diccionario

# Utilizando la función values
for el in language.values():
```

```
    print(el) # Imprime los valores

# Utilizando items -> Retorna una lista de tuplas
for key, value in language.items():
    print(key, value) # Imprime llave y valor

"""

Lista de tuplas que se crea en el último bucle.

De esta estructura:
_____________________
language = {
    "name": "python",
    "author": "Guido"
}

A esta estructura:
_____________________
list = [(key: value), (key. value)...]

"""
```

@Desiré Marrón