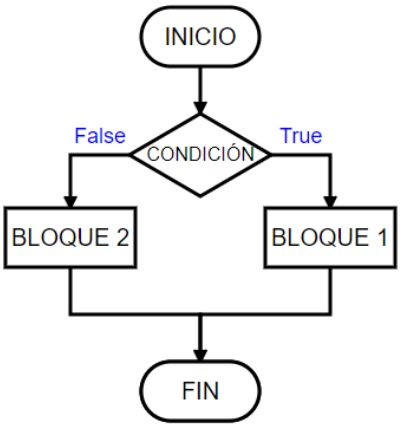
- ¿Qué es un condicional?

Se trata de una situación a cumplir para que se ejecute una parte (true) u otra (false) del código posteriormente detallado. Suele trabajarse en la estructura de control if, la cual permite que un programa ejecute unas instrucciones cuando se cumplan una o más condiciones. Para poder trabajar con múltiples situaciones, tenemos como soporte elif (para otra situación específica posterior al if inicial) y else (para el resto de las situaciones no detalladas previamente).



Como ejemplo teórico tenemos la imagen de la derecha, y como ejemplo práctico la de abajo:

```python
a = 10
b = 20

if a < b:
    print(f"{a} es menor que {b}")
elif b < a:
    print(f"{a} es mayor que {b}")
else:
    print(f"{a} y {b} son iguales")
```

- ¿Cuáles son los diferentes tipos de bucles en Python? ¿Por qué son útiles?

En la gran mayoría de lenguajes nos solemos encontrar con 3 tipos de bucles: for, while, do while. Sin embargo en Python sólo tenemos disponibles los 2 primeros.

El bucle for, se aplica cuando el bloque a iterar tiene un número final definido.

```python
for i in range(1, 11):
    print(i)
```

Por otro lado, el bucle while, se aplica cuando la iteración no tiene un número definido. Esto puede llevar a bucles infinitos (Ctrl+C para detener la ejecución) como el siguiente:

```python
a = 1
while a < 10:
    print(a)
```

Para evitar que se produzcan estos bucles infinitos, es necesario añadir una operación de suma al condicionante del bucle, en este caso la variable a:

```
a = 1
while a < 10:
    print(a)
    a += 1
```

- ¿Qué es una lista por comprensión en Python?

La comprensión de listas es una construcción en Python que nos permite crear listas a partir de otras listas, tuplas y cualquier iterable. Nos permite escribir en forma más concisa un algoritmo. Es una funcionalidad que nos permite crear listas avanzadas en una misma línea de código. Esto se ve mucho mejor en la práctica, por lo que ahí va el ejemplo:

```
# Método tradicional
lista = []
for letra in 'casa':
    lista.append(letra)
print(lista)

# ['c', 'a', 's', 'a']

# Con comprensión de listas
lista = [letra for letra in 'casa']
print(lista)

# ['c', 'a', 's', 'a']
```

- ¿Qué es un argumento en Python?

Los argumentos en funciones se refieren a los valores que se pasan a una función para que realice una tarea específica. Como desarrolladores, es común encontrarnos con situaciones en las que necesitamos que una función tome distintos valores para realizar una tarea en particular.

A veces se le confunde con el parámetro. Sin embargo, la diferencia está en que se le llama parámetro a la variable que se declara al definir la función (en donde se recibe el valor). En cambio, se le llama argumento al valor que es enviado a la función cuando se invoca.

```
def my_function(parameter):#El parámetro
  print(parameter)

my_function("argument") #El argumento
```

- ¿Qué es una función Lambda en Python?

En Python, una función Lambda se refiere a una pequeña función anónima. Las llamamos “funciones anónimas” porque técnicamente carecen de nombre. Al contrario que una función normal, no la definimos con la palabra clave estándar def que utilizamos en Python.

El controlador de la función de Lambda es el método del código de la función que procesa eventos. Cuando se invoca una función, Lambda ejecuta el método del controlador. La función se ejecuta hasta que el controlador devuelve una respuesta, se cierra o se agota el tiempo de espera.

Las expresiones lambda se usan idealmente cuando necesitamos hacer algo simple y estamos más interesados en hacer el trabajo rápidamente en lugar de nombrar formalmente la función.

```
# Función Lambda para calcular el cuadrado de un número
square = lambda x: x ** 2
print(square(3)) # Resultado: 9

# Funcion tradicional para calcular el cuadrado de un numero
def square1(num):
  return num ** 2
print(square(5)) # Resultado: 25
```

- ¿Qué es un paquete pip?

pip es el sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python. Los paquetes de software son conjuntos de módulos y bibliotecas que pueden ser reutilizados en diferentes proyectos. Los paquetes pueden proporcionar funcionalidades que van desde operaciones matemáticas hasta acceso a bases de datos y análisis de datos, entre otros.

La ventaja de usar un gestor de paquetes como **pip** es que simplifica el proceso de instalación y actualización de paquetes, así como la gestión de dependencias entre ellos. Además, **pip** permite a los desarrolladores compartir sus propios paquetes con la comunidad y descargar paquetes creados por otros desarrolladores para usarlos en sus proyectos.

Algunas características clave de pip incluyen: instalación (**pip install nombre_del_paquete**), actualización (**pip install –upgrade nombre_del_paquete**), desinstalación (**pip uninstall nombre_del_paquete**) y listado de paquetes (**pip list**).