



I.E.S Jacarandá

Unidad 1

Programación Estructurada

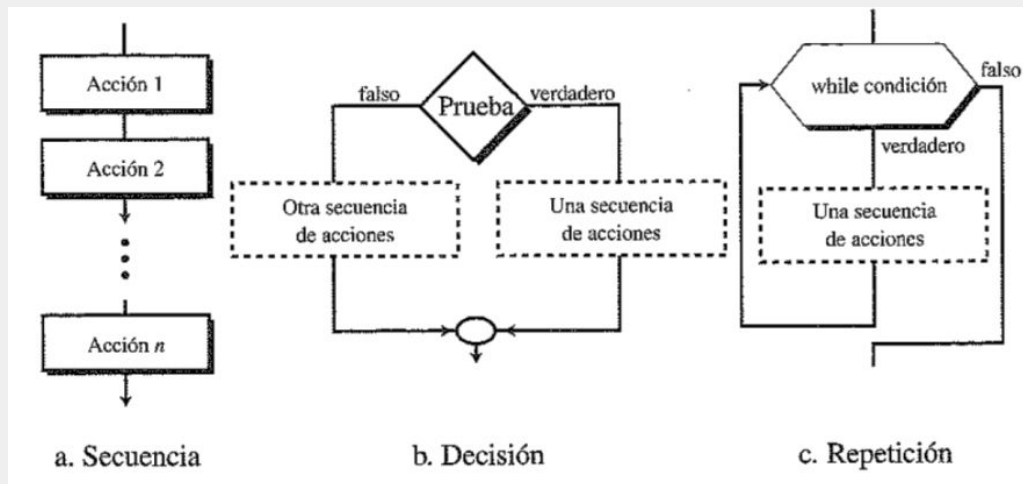
1º DAW

Definición Programación Estructurada

La programación estructurada es aquel **paradigma de programación** que dice que todo programa puede escribirse únicamente utilizando 3 estructuras básicas de control:

- Secuencial
- Selección o alternativa
- Iteración o Repetitiva

De manera que (tal y como hemos visto con los diagramas de flujo) las instrucciones pueden no ejecutar una tras otras según se escriben en el programa.

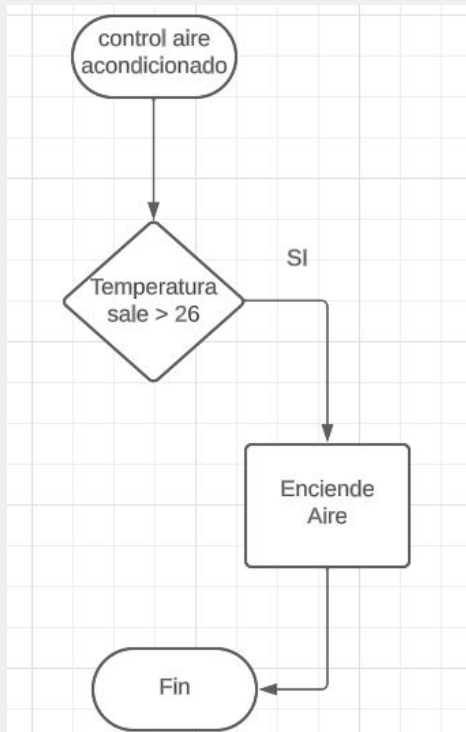


Estructura Alternativa o Condicionales

Según una condición, se ejecutará o no un grupo de instrucciones u otro.

Condicional Simple

Usaremos estas estructuras cuando deseemos que se ejecute un bloque de instrucciones u otros en función de si se cumple o no una condición.



Por ejemplo, imagina el control de temperatura que controla el encendido o el apagado de un aire acondicionado (cierto o falso). Deseamos que el aire se encienda si la temperatura está por encima de 26 grados

Estructura Alternativa o Condicionales

Condicional Simple

Sintaxis en Python:

if condicionLogica :

#condicionLogica devuelve True o False

 instruccionSiSeCumple1

 instruccionSiSeCumple2

...

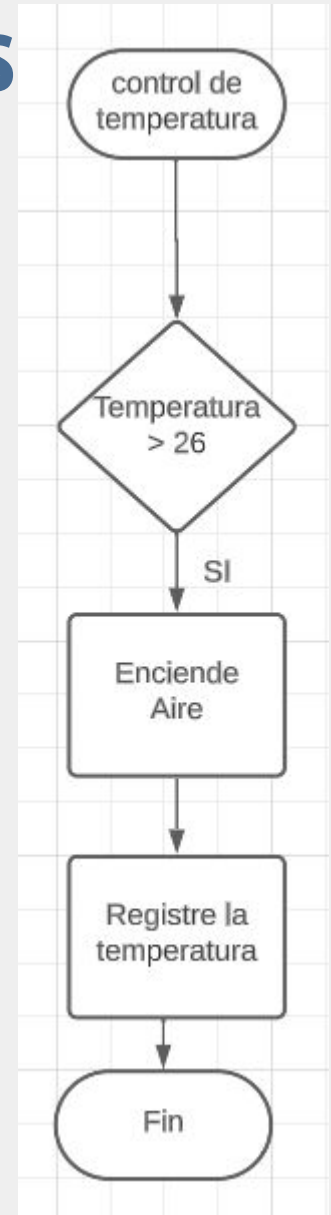
 instruccionSiSeCumplen

instruccionFueraDelSI

IMPORTANTE:

Las instrucciones **dentro** del if son **tabuladas**.

Sin tabulador → fin de bloque if



Estructura Alternativa o Condicionales

Condicional Simple

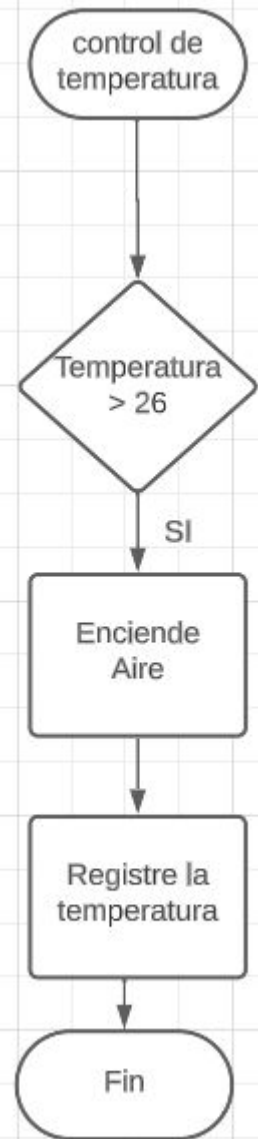
Sintaxis en Python:

```
temperatura = float (input("La temperatura  
de la sala es: "))
```

```
if temperatura > 26 : #devuelve True
```

```
    print("Encendiendo aire acondicionado")
```

```
print("Registrada:" + str(temperatura))
```



Estructura Alternativa o Condicionales

Condicional Simple

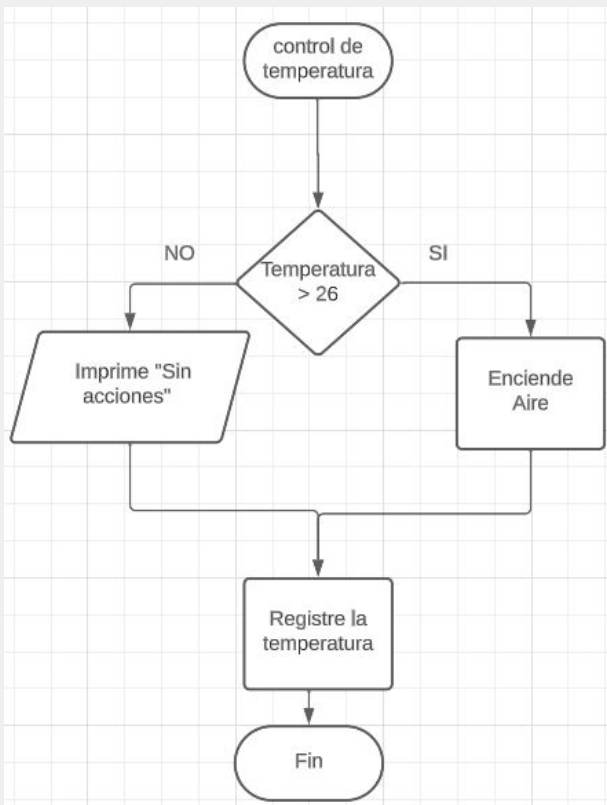
Es tu turno...

1. Modifica el código anterior para que además de imprimir que se enciende, muestre otro mensaje diciendo. “sistema monitorizado automáticamente”
2. Modifica ahora el diagrama y el código para que, **una vez encendido** el aire, se controle que el aire se apague si la temperatura de la sala es menor que 23.
3. A continuación de estos dos bloques if, añade otro bloque si la temperatura está entre 23 y 24 grados para que suba la temperatura

Estructura Alternativa o Condicionales

Condicional Compuesta

En este caso, se establecerán múltiples ramas de ejecución en función de los diferentes valores de la condición. Por ejemplo, se establece un flujo si True y otro si False.



Por ejemplo, ahora queremos que si la temperatura NO es mayor que 26, registre que no hay que hacer nada.

En cualquiera de los dos casos, se desea registrar la temperatura

Estructura Alternativa o Condicionales

Condicional Compuesta

Sintaxis en Python:

if condicionLogica : #si True

instruccionSiSeCumple1

...

instruccionSiSeCumpleN

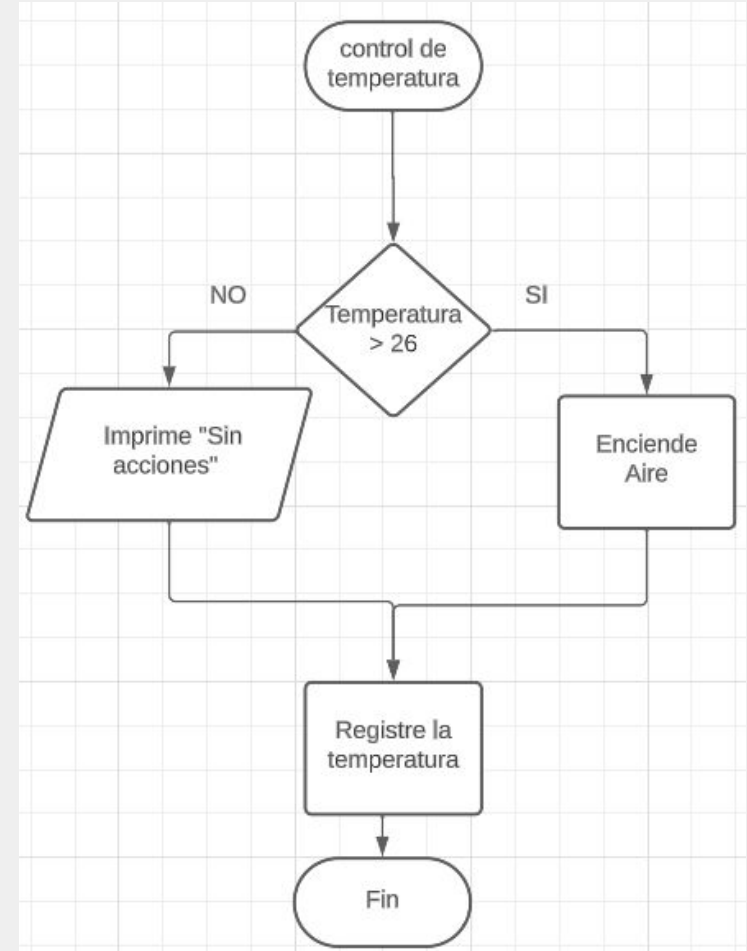
else: #si No (False)

instruccionNOSeCumple1

...

instruccionNOSeCumpleN

instruccionFueraBloqueSI

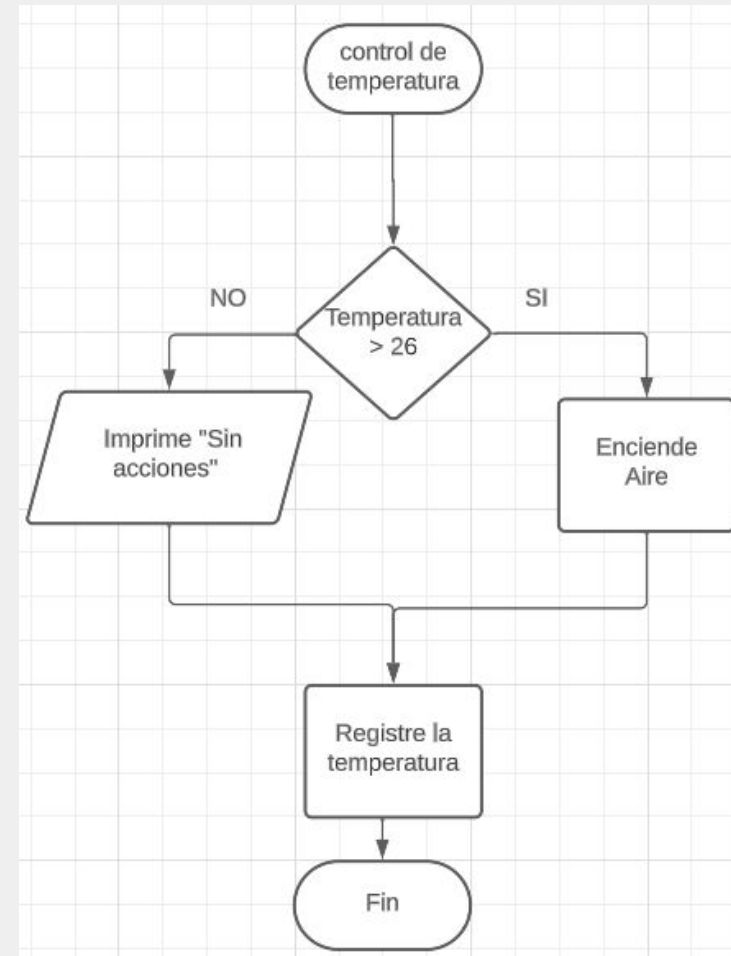


Estructura Alternativa o Condicionales

Condicional Compuesta

Sintaxis en Python:

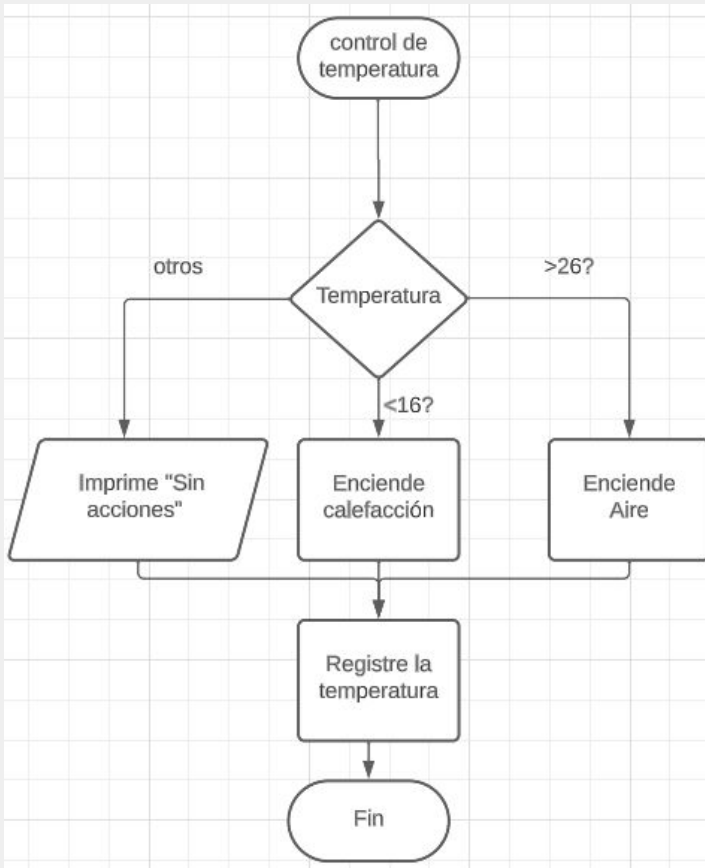
```
temperatura = float (input(" La
temperatura de la sala es: "))
if temperatura > 26 : #si True
    print("Encendiendo aire acondic")
else: #Otros → False
    print("Sin acciones")
print("Temperatura registrada:" +
str(temperatura))
```



Estructura Alternativa o Condicionales

Condicional Compuesta Multivalor

Ahora el bloque condicional **no devolverá un lógico, si no un valor y en función de eso se establecerán las condiciones lógicas** que harán que se ejecuten un flujo u otro.



Por ejemplo, en este caso, en función del valor de la temperatura controlaremos qué hacer:

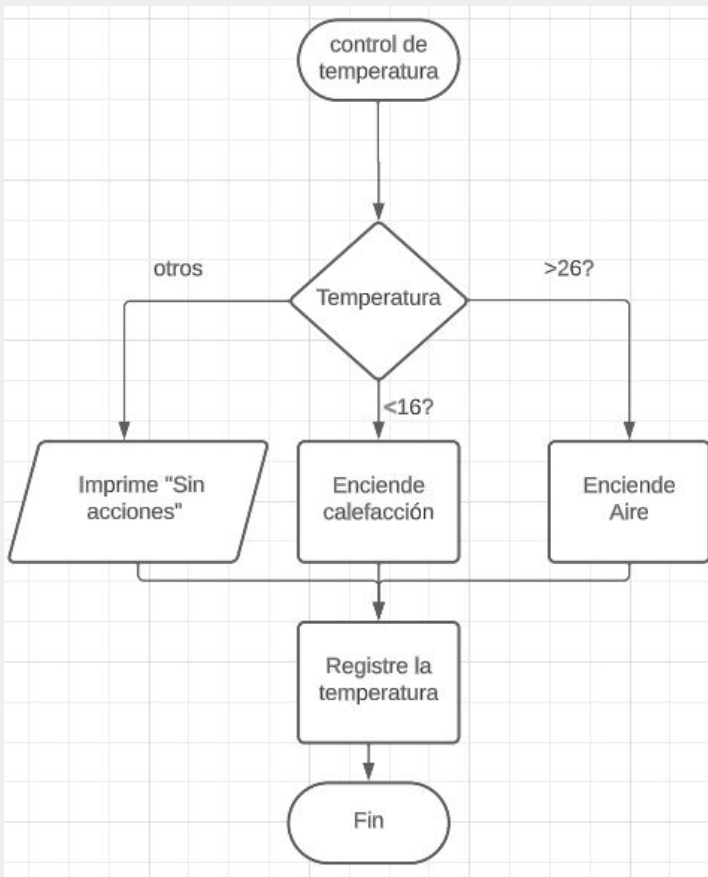
- si la temperatura está por debajo de 16 grados → el sistema debe encender la calefacción
- si está por encima de 26 → el sistema deberá encender el aire
- y si no ocurre ninguna de las anteriores → imprime Sin acciones

En cualquier caso, debe registrar la temperatura.

Estructura Alternativa o Condicionales

Condicional Compuesta Multivalor

Imagina que deseamos añadir otra condición dentro de la misma estructura. De forma que la respuesta no sea sí o no, si no que en función del valor de algo se ejecute una rama u otra



Por ejemplo, en este caso, en **función de la temperatura** controlaremos qué hacer:

- si la temperatura está **por debajo de 16 grados** → el sistema debe encender la calefacción
- si está **por encima de 26** → el sistema deberá encender el aire
- y si no ocurre ninguna de las anteriores (otras) → imprime Sin acciones

En cualquier caso, debe registrar la temperatura.

Estructura Alternativa o Condicionales

Condicional Compuesta Multivalor

Sintaxis en Python:

```
if condicionLog1 : #condicionLog1 == true
```

```
    instruccionSiSeCumple1
```

```
    ...
```

```
    instruccionSiSeCumpleN
```

```
elif condicionLog2: #condicionLog2 == true
```

```
    instruccionSiSeCumple2
```

```
    ...
```

```
    instruccionSiSeCumpleN
```

```
else: # Otras
```

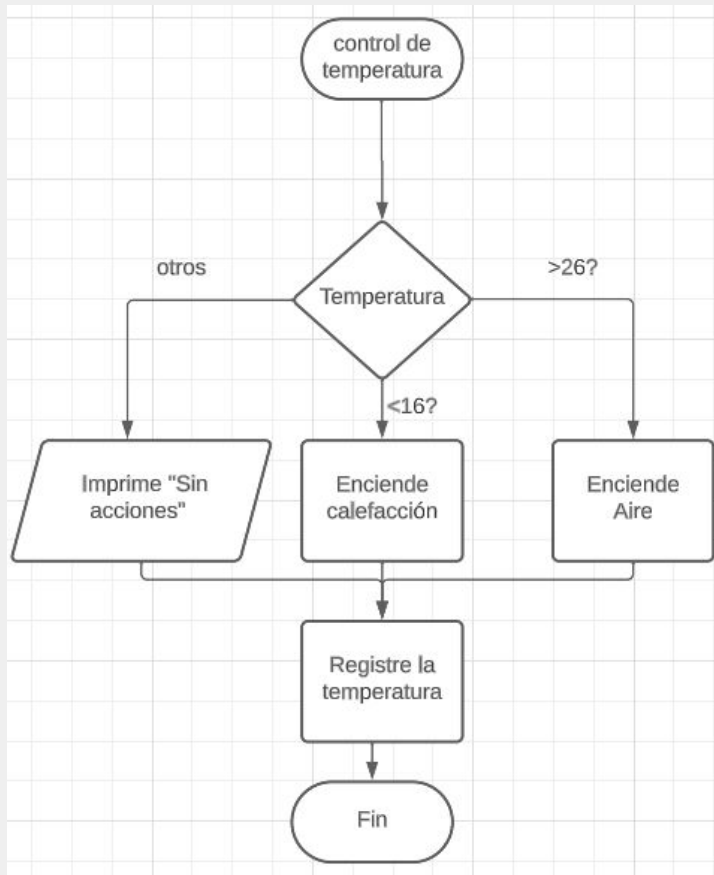
```
# NO se cumple condicionLog1 ni condicionLog2
```

```
    instruccionOTRAS1
```

```
    ...
```

```
    instruccionOTRASN
```

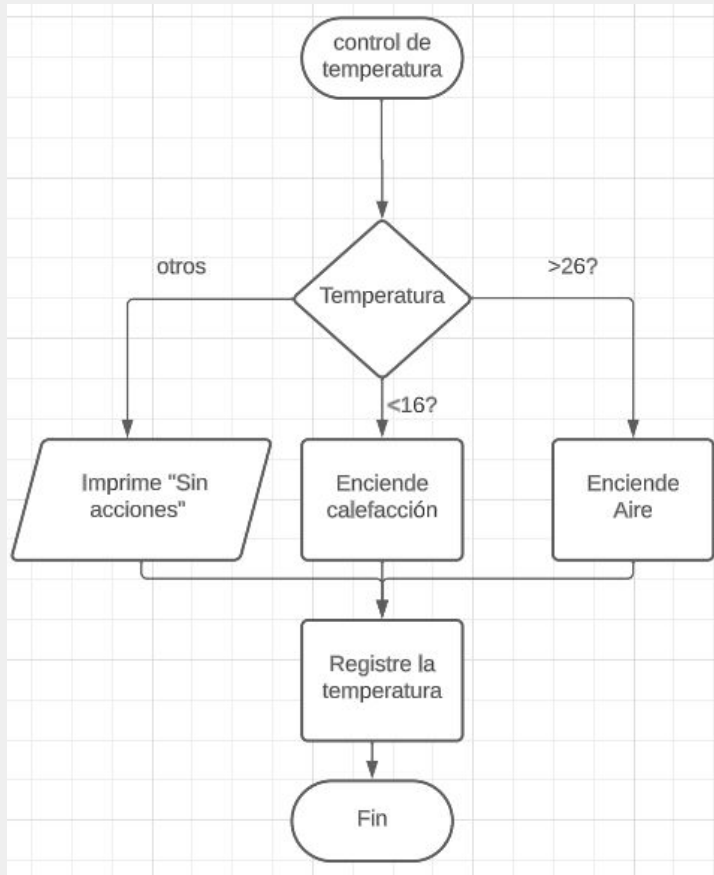
```
instruccionFueraBloqueSI
```



Estructura Alternativa o Condicionales

Condicional Compuesta Multivalor

Sintaxis en Python:



```
temperatura = 23
```

```
if temperatura > 26 :#cond1 == true
```

```
    print("Encendiendo aire acondic")
```

```
elif temperatura < 16 : #cond2 == true
```

```
    print("Encendiendo calefacción")
```

```
else: #NO cumple ni cond1 NI cond2
```

```
    print("Sin acciones")
```

```
#Lo siguiente está fuera del bloque IF
```

```
print("Temperatura registrada:" +
```

```
str(temperatura))
```

Estructura Alternativa o Condicionales

Condicional Compuesta Compleja

Se denomina así porque permite establecer diferentes caminos en **función del valor de una variable**. A diferencia del caso anterior, en el que teníamos varias condiciones diferentes, en este caso se compara siempre con respecto a la misma variable pero con diferentes valores. Volvamos a nuestro ejemplo de climatización, imagina que nuestro enunciado es el siguiente:

- Si la **temperatura** == 26 → Enciende aire
- Si la **temperatura** == 27 → Enciende y baja la temperatura
- Si la **temperatura** == 14 → Enciende calefacción

En este caso, siempre estoy comparando si es igual a un valor, para este tipo de condiciones será más adecuado utilizar el operador **match** de Python.

Estructura Alternativa o Condicionales

Condicional Compuesta Compleja

Sintaxis en Python:

```
match variable :  
    case 400 :  
        # valor 1  
    case 404 :  
        # valor 2  
    case _ :  
        # lo que se hace por defecto Es opcional
```

Consideraciones:

- Disponible a partir de Python 3.10
- Se pueden combinar varios literales en un solo patrón usando | («o»):.
Por ejemplo, `case 401 | 403 :`

Estructura Alternativa o Condicionales

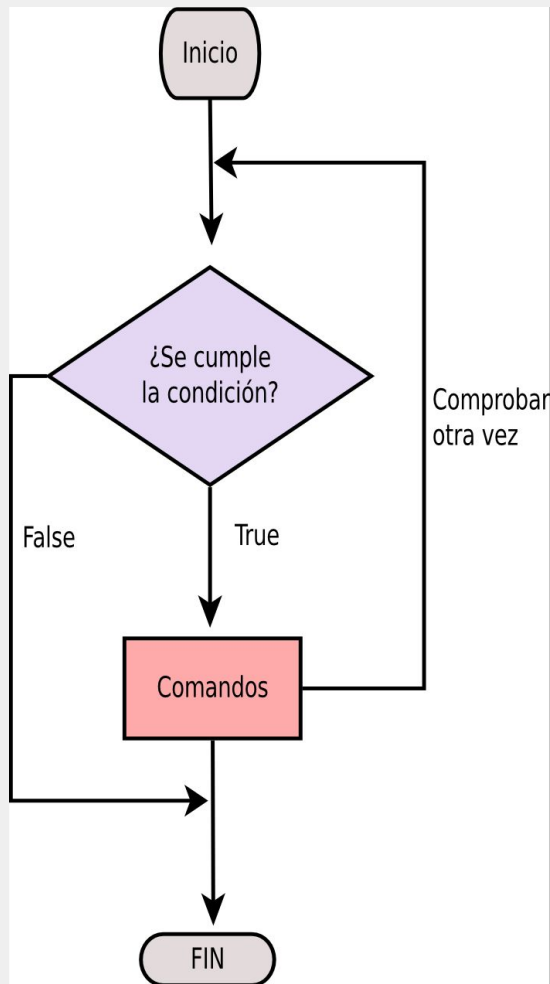
Condicional Compuesta Compleja

Sintaxis en Python Ejemplo

```
match codigoHttp :  
    case 400 :  
        mensajeError = "Bad Request"  
    case 404 :  
        mensajeError = "Nof found"  
    case 401 | 403 | 407 :  
        mensajeError = "Not allowed"  
    case _ :  
        mensajeError = "Something wrong"
```


Estructura Iterativa

Gracias a este tipo de estructura, una instrucción puede ejecutarse varias veces.



En este tipo de estructuras podemos:

1. Ejecutar la instrucción mientras se cumple una condición.
2. Ejecutar la instrucción un número de veces

Consideraciones:

Con las estructuras repetitivas es muy recomendable hacer trazas para saber el valor de las variables en cada iteración

Estructura Iterativa

Estructura iterativa: For

```
for variable_indice in range(tamaño):  
    instruccion_1  
    instruccion_2
```

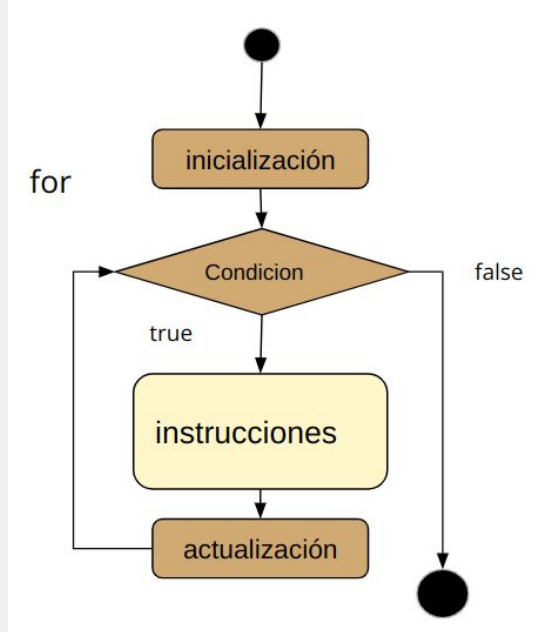
Consideraciones:

- Variable_indice será la variable con el que contemos el número de veces que hemos repetido la instrucción. Inicialmente, tendrá un valor de 0.
- Esta variable sólo existe dentro del contexto del bucle for, no pudiéndose utilizar fuera.

Estructura Iterativa

Estructura iterativa: For

Usaremos la estructura for cuando queramos repetir una instrucción un número de veces.



```
for variable_indice in range(tamaño):  
    instruccion_1  
    instruccion_2  
    ...
```

Por ejemplo, vamos a 4 veces la cadena
Hola

```
for i in range(4):  
    print("Hola")
```

Ejercicio:

Modifica el código anterior para que muestre los números del 0 al 3

Estructura Iterativa

Estructura iterativa: For

```
•   for variable_indice in range(tamaño):  
       instruccion_1  
       instruccion_2
```

Función range:

- Permite crear diferentes tipos de rango sobre los que iterar.
- Su sintaxis es `range(limite_inferior, limite_superior, incremento)`, pero también nos permite llamarla o ejecutarla sin pasar esos tres valores.
 - `range(3)` inferior= 0, superior= 3, incremento 1 → valores: 0, 1, 2
 - `range(1, 3)` inferior=1, superior=3, incremento 1 → valores: 1,2
 - `range(1,3, 2)` inferior=1, superior=3, incremento 2 → valores: 1

Estructura Iterativa

Estructura iterativa: For

Función range:

`range(limite_inferior, limite_superior, incremento)`

- Se comienza por el límite inferior
- Y se termina en el valor anterior al límite superior. (superior NO incluido)
- El incremento puede ser:
 - Incrementando de uno en uno (valor por defecto)
 - De dos en dos `range(1,3, 2)`
 - O un valor negativo, si lo que deseamos es hacer una cuenta atrás. Ejemplo de `range(10, 0, -1)`

Estructura Iterativa

Estructura iterativa: For

Es tu turno:

- Construye un programa que me imprima todos los números pares del 0 al 20
- Modifica el programa anterior para que ahora comience desde el 1
- Por último modifica el programa para que vaya incrementando de 2 en 2 siendo 0 el valor de inicio ¿Qué ocurre?

Estructura Iterativa

Estructura iterativa: While

```
while expresion_logica :
```

```
    sentencia_1
```

```
    sentencia_2
```

```
    ...
```

TODO

