Programación

Clase 03

U2

Elementos básicos de la programación: Ciclos

Objetivos

Comprender cómo funcionan los ciclos for y while

Aplicar los ciclos en el desarrollo de problemas de programación

Aplicar los ciclos para generar control de errores

Comentarios

Antes de seguir, necesitamos una forma de describir lo que estamos programando, para que luego no te olvides de qué hacía cada cosa.

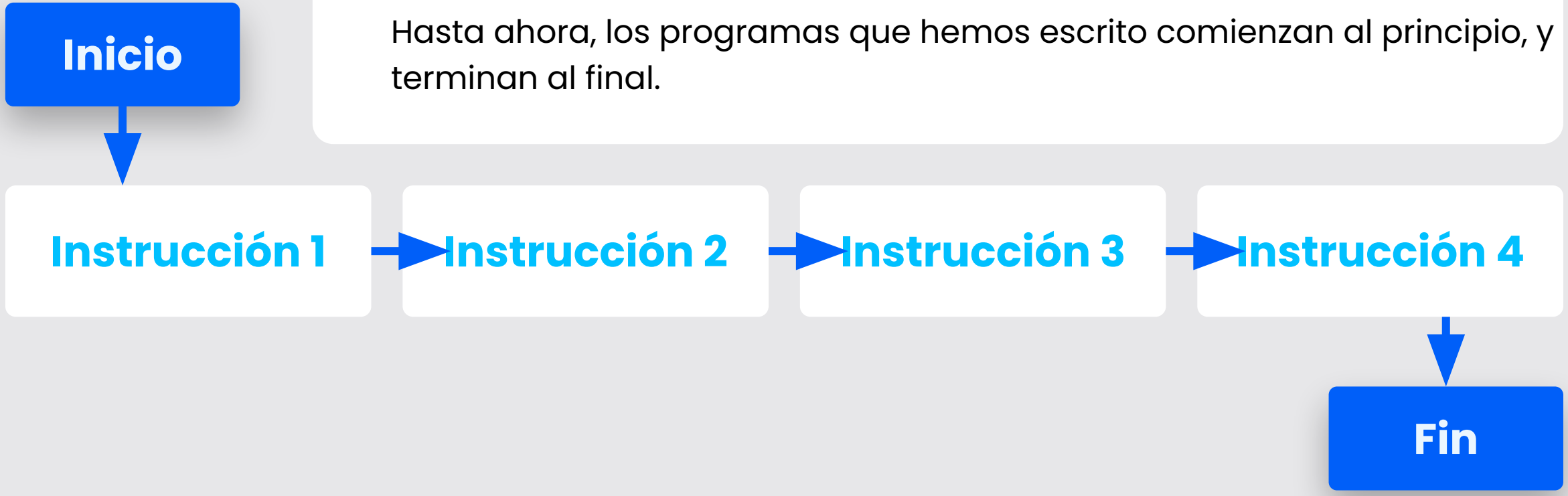
También ayuda a tu equipo de trabajo a entender mejor que hizo cada uno.

Estamos hablando de los cometarios, los que se escriben de la siguiente forma.

```
#esto es un comentario
a = 25 #esta variable guarda mi edad
```

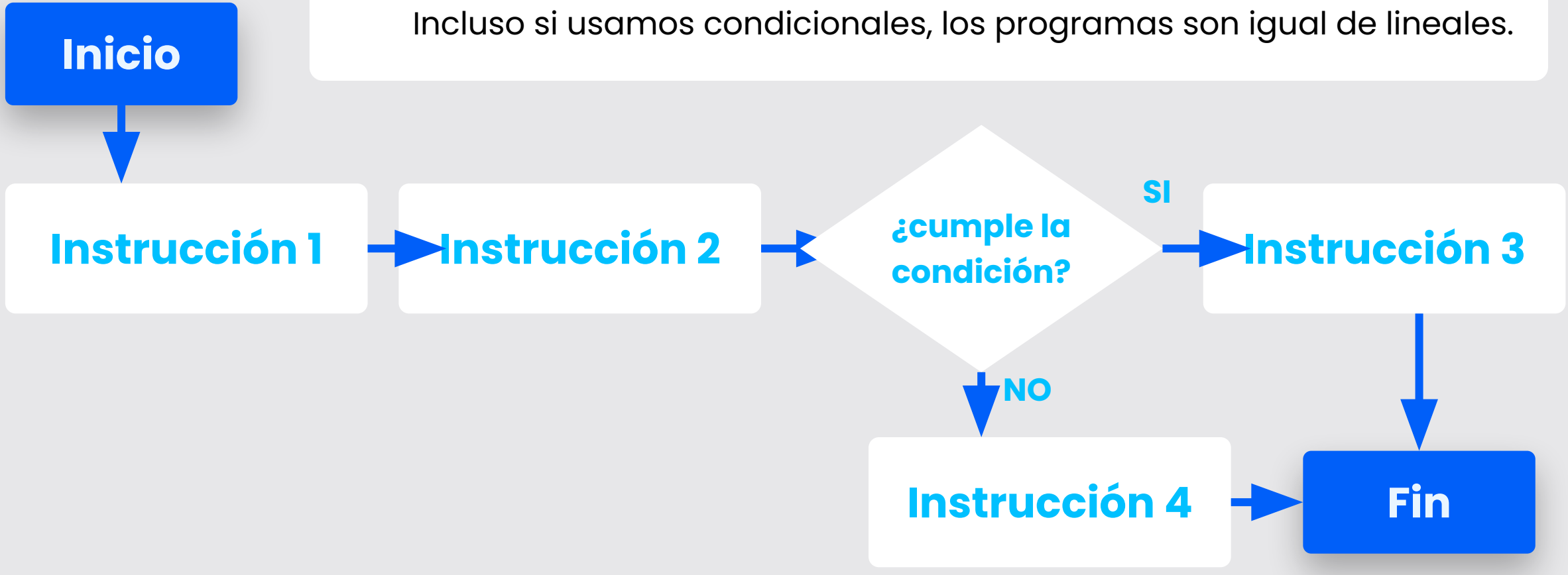
Programas lineales

Hasta ahora, los programas que hemos escrito comienzan al principio, y terminan al final.



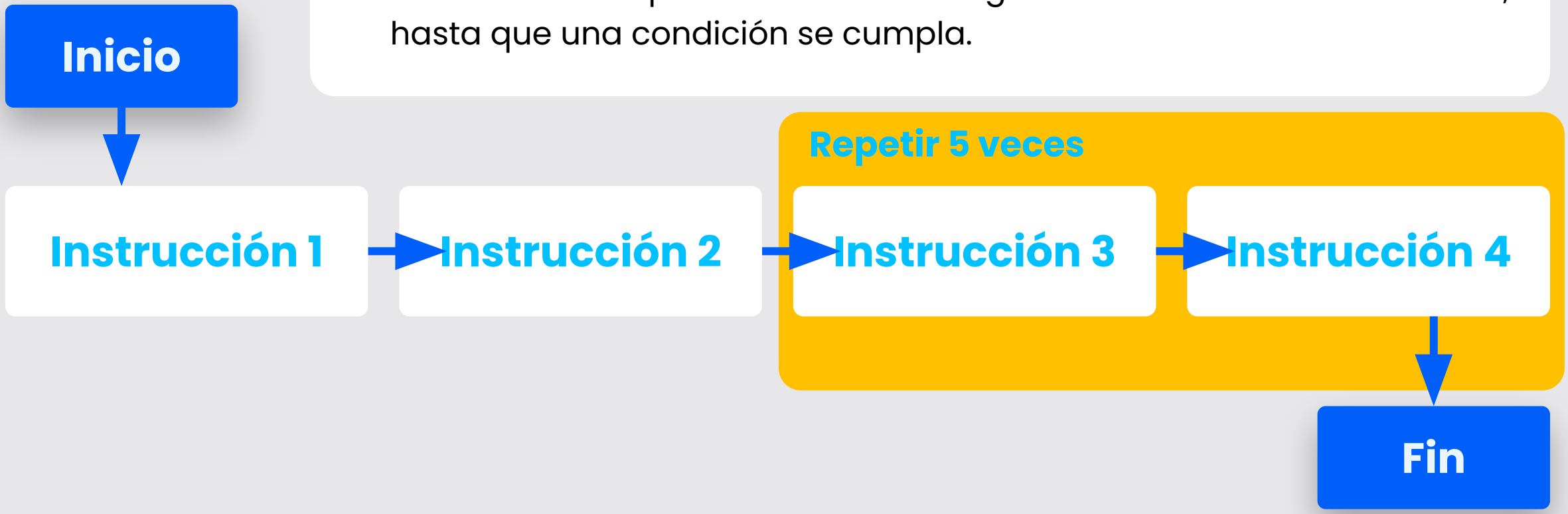
Programas lineales

Incluso si usamos condicionales, los programas son igual de lineales.



Repeticiones

Para solucionar muchos de los problemas del mundo real, se necesita una forma de repetir un trozo de código una cierta cantidad de veces, o hasta que una condición se cumpla.



El ciclo for

En Python, el ciclo “for” nos permitirá repetir una secuencia de instrucciones una cantidad dada de veces. Su estructura básica es la siguiente:

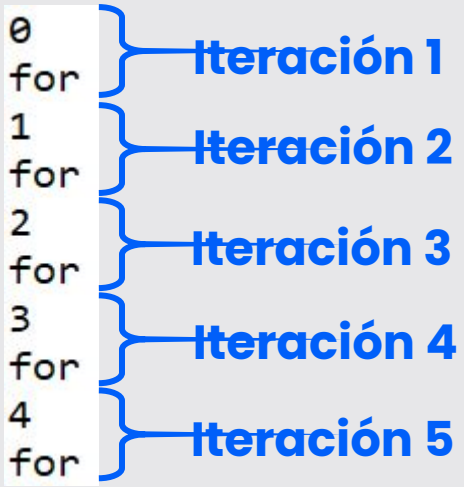
```
for i in range(numero_de_veces):
    instrucción que se repetirá
    instrucción que se repetirá
```

Las instrucciones se ejecutarán el número de veces correspondiente al valor dentro del **range**. Y la variable **i** iterará sobre ese valor, comenzando en **0** y terminando en **numero_de_veces – 1**.

El ciclo for

Veamos un ejemplo:

```
for i in range(5):
    print(i)
    print('for')
```



El código dentro del bloque se ejecutará 5 veces.

La variable i tomará valores entre 0 y 4.

El ciclo for

También puedes modificar el range de esta manera:

```
for i in range(2, 6):
    print(i)
```

```
2
3
4
5
```

```
for i in range(2, 9, 3):
    print(i)
```

```
2
5
8
```

Por lo tanto, estas tres notaciones tienen el mismo resultado.

```
for i in range(5):
    print(i)
```

```
for i in range(0, 5):
    print(i)
```

```
for i in range(0, 5, 1):
    print(i)
```

El ciclo for

Existe otra manera de escribir el **for**.

```
lista = [2, 5, 6, 1]

for i in lista:
    print(i)
```

```
2
5
6
1
```

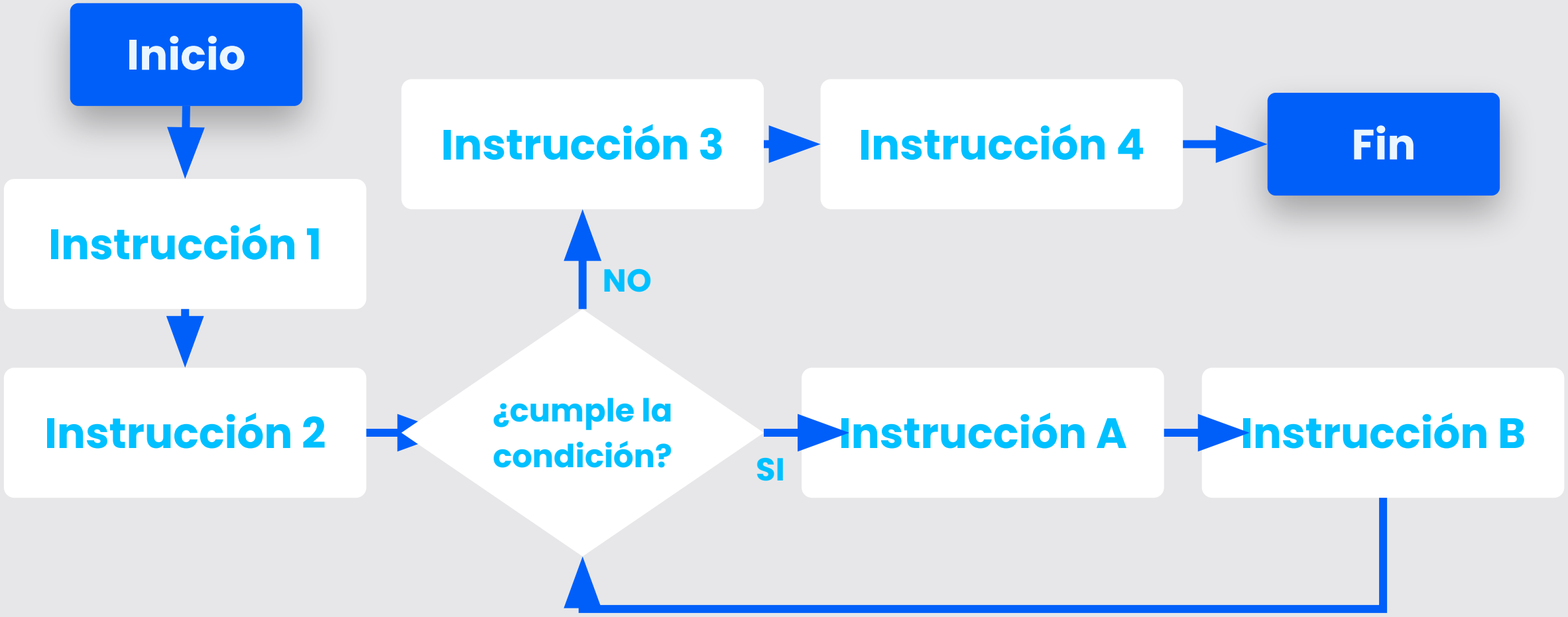
Cuando quieras recorrer los valores de una lista el iterador **i** tomará el valor de cada elemento de la lista. Esta forma de escribir el **for** la veremos mas adelante cuando sepas utilizar listas.

El ciclo while

El ciclo “while” nos permitirá repetir una secuencia de código MIENTRAS una condición se mantenga siendo verdadera:

```
instrucción1
instrucción2

while condición:
    instrucción A
    instrucción B

instrucción3
instrucción4
```

El ciclo while



El concepto de contador

¿Qué pasa si ejecuto el siguiente código?

```
y = 14
y = y + 1
print(y)
```

El ciclo while

¿Qué pasa si ejecuto el siguiente código?

```
x = 1

while x < 5:
    x = x + 1
    print(x)

print(x)
```

```
2
3
4
5
5
```

A esto le llamaremos un contador, ya que "cuenta" la cantidad de veces que se repite el código.

El ciclo while

¿Qué pasa si ejecuto el siguiente código?

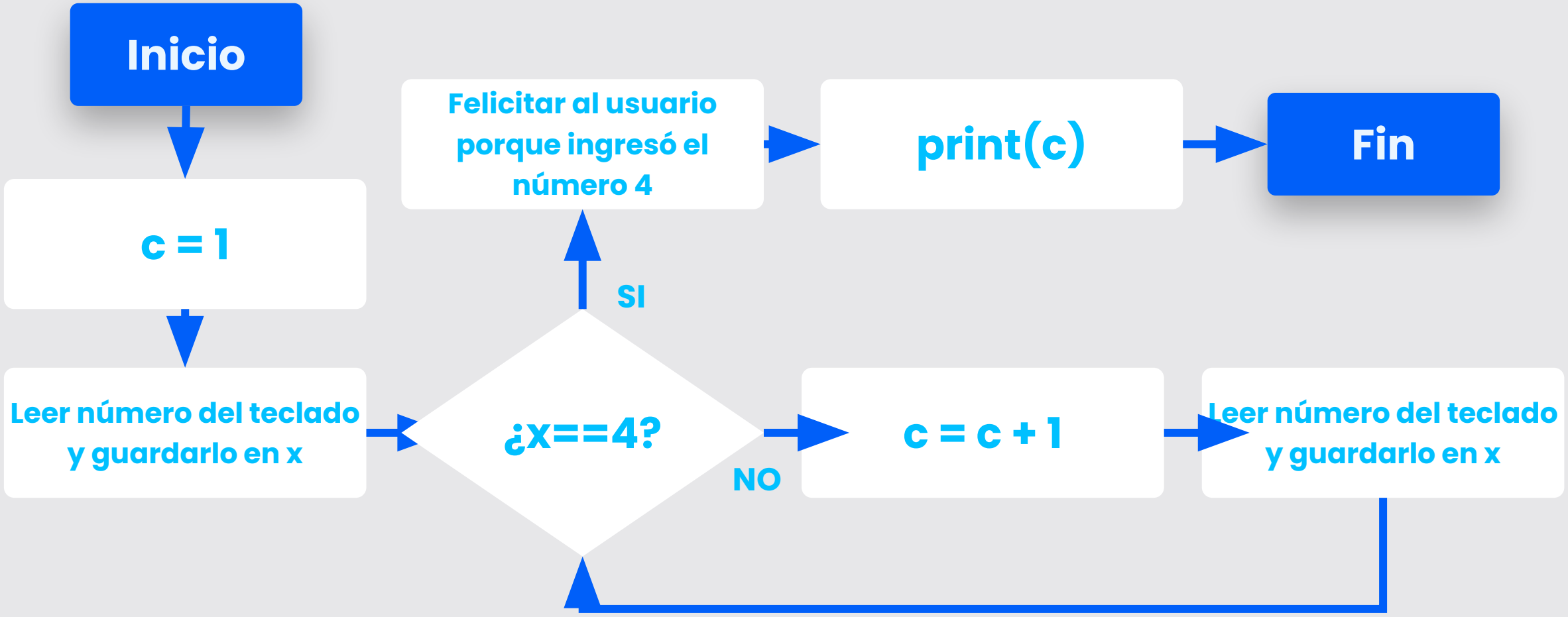
```
x = 0
s = 0
while x < 5:
    x = x + 1
    s = s + 3

print(x)
print(s)
```

5

15

“x” es un contador y “s” es un acumulador, ya que va acumulando algo que queremos sumar.

Creando un while más complejo

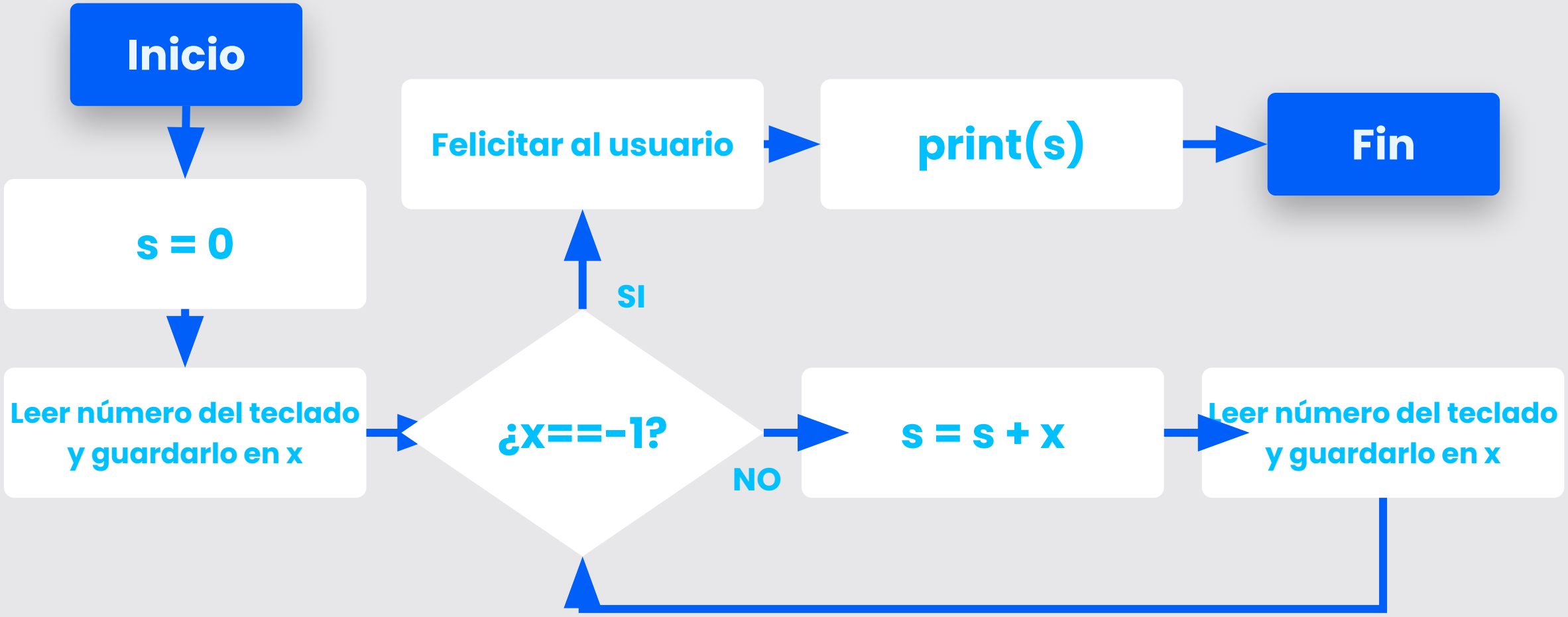


Leyendo números del teclado

Cómo puedo escribir algo que realice lo siguiente:

- Quiero sumar un conjunto de números, y saber el total de dicha suma
- El programa me tiene que preguntar los números uno a uno
- Yo no sé cuántos números se van a ingresar
- Pero si el usuario ingresa un -1 significa que ya no hay más números

Leyendo números del teclado

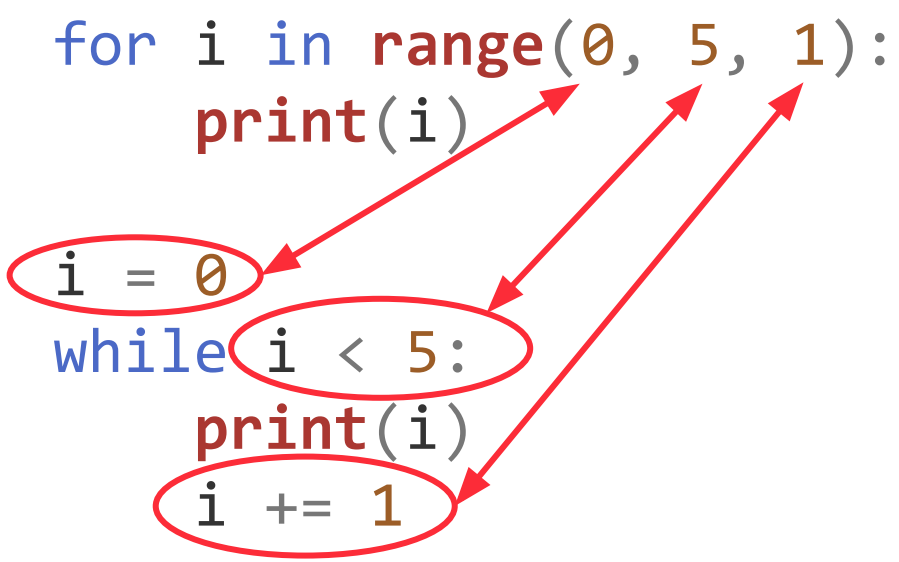


Leyendo números del teclado

Cómo puedo escribir algo que realice lo siguiente:

- Queremos contar cuantos estudiantes varones son mayores de 18 años
- También queremos desplegar la suma de las edades de los alumnos menores a 18 años
- Hay 15 estudiantes en el curso

Comparando ambos ciclos



Como puedes observar, puedes transformar un ciclo for en un ciclo while y viceversa.

Dependerá de la situación cuando usar uno u otro.

Control de errores

A veces necesitas que el usuario ingrese un tipo especifico de información, y que en caso de un ingreso erróneo se vuelva a preguntar por el valor requerido.

Esto se puede hacer fácilmente con el uso de ciclos while.

Solo necesitas pensar bien en las condiciones de término de cada ciclo.

Probemos con un ejemplo simple. Necesitamos que el usuario nos diga si quiere su orden para llevar en un sistema de ventas. La respuesta solo puede ser **Si** o **No**.

Control de errores

```
¿Pedido para llevar? (Si,

Error, ¿Pedido para lleva
gdfsh

Error, ¿Pedido para lleva
```

```
opcion = input('¿Pedido para llevar? (Si, No): ').lower()

while opcion != 'si' and opcion != 'no':
    opcion = input('Error, ¿Pedido para llevar? (Si, No): ').lower()
```

Mientras las entradas de texto no sean ni si, ni no, el ciclo seguirá preguntando por una opción.

Con el comando lower() o upper() transformas los textos en minúsculas o mayúsculas respectivamente. Con esto te evitas crear opciones para todas las combinaciones.

Programación

Clase 03.1

Ejercicios

Ejercicio 1

Comencemos con algo sencillo. Crea un ciclo for que se repita 5 veces y que imprima la variable iteradora.

Luego haz lo mismo con un ciclo while.

Crea un programa que imprima los números impares hasta el 50.

Luego intenta imprimir la suma de los primeros 100 enteros.

Ejercicio 2

Ahora pregunta por un numero inicial, uno final e imprime la suma de los números que se encuentren entre este rango, excluyendo los extremos.

Ahora pide un tercer numero. Y luego imprime la sucesión de números en el intervalo del tercer numero.

Por ejemplo, si el inicial es 5, el final es 20 y el intervalo es 4, tendrás que imprimir el rango de números de 4 en 4.

5, 9, 13, 17.

Ejercicio 3

Escribe un programa que muestre en pantalla los números del 1 al 100, sustituyendo los múltiplos de 3 por la palabra Fizz y, a su vez, los múltiplos de 5 por Buzz. Para los números que, al mismo tiempo, son múltiplos de 3 y 5, utiliza el combinado FizzBuzz.

Por ejemplo:

1
2
Fizz
4
Buzz
Fizz
7
8
Fizz
Buzz
11
Fizz
13
14
FizzBuzz
16
...

Trabajo autónomo mínimo

Revisar capítulo 4 libro guía.

Resolver ejercicios 1, 2, 4, 13, 14 y 20 del capitulo 4.

Programación

Clase 03.2

U2

Elementos básicos de la programación: Algoritmos

Objetivos

Comprender el concepto de algoritmos para la solución de problemas

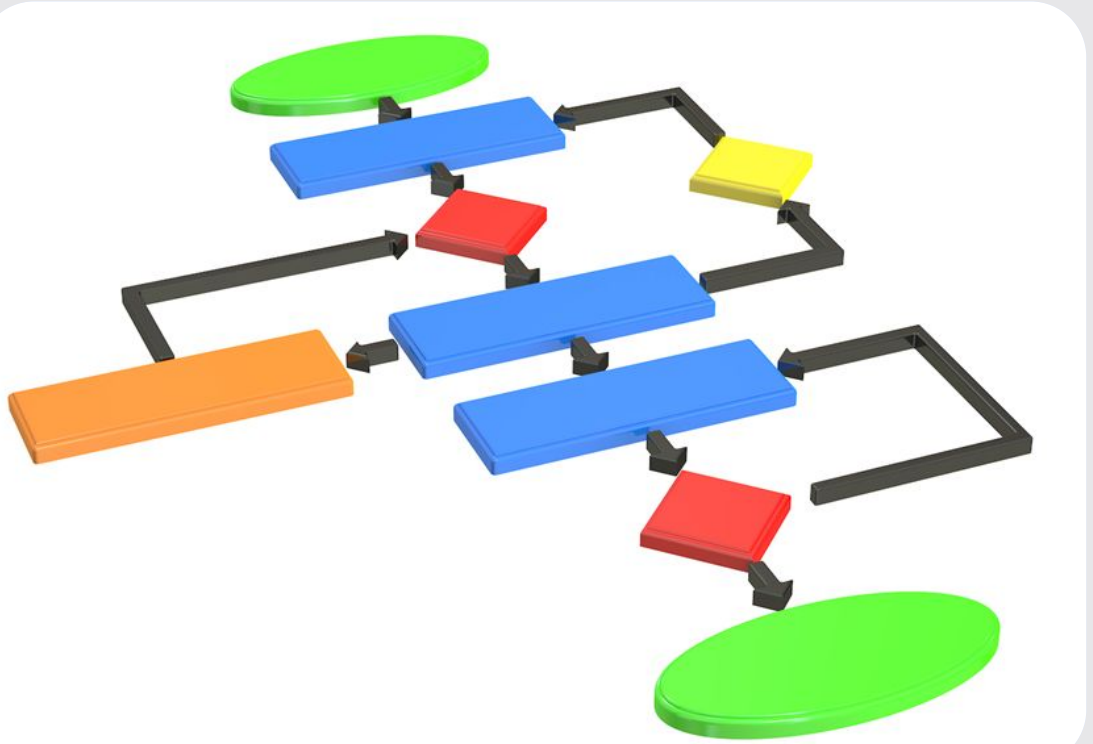
Aplicar algoritmos para resolver problemas de programación

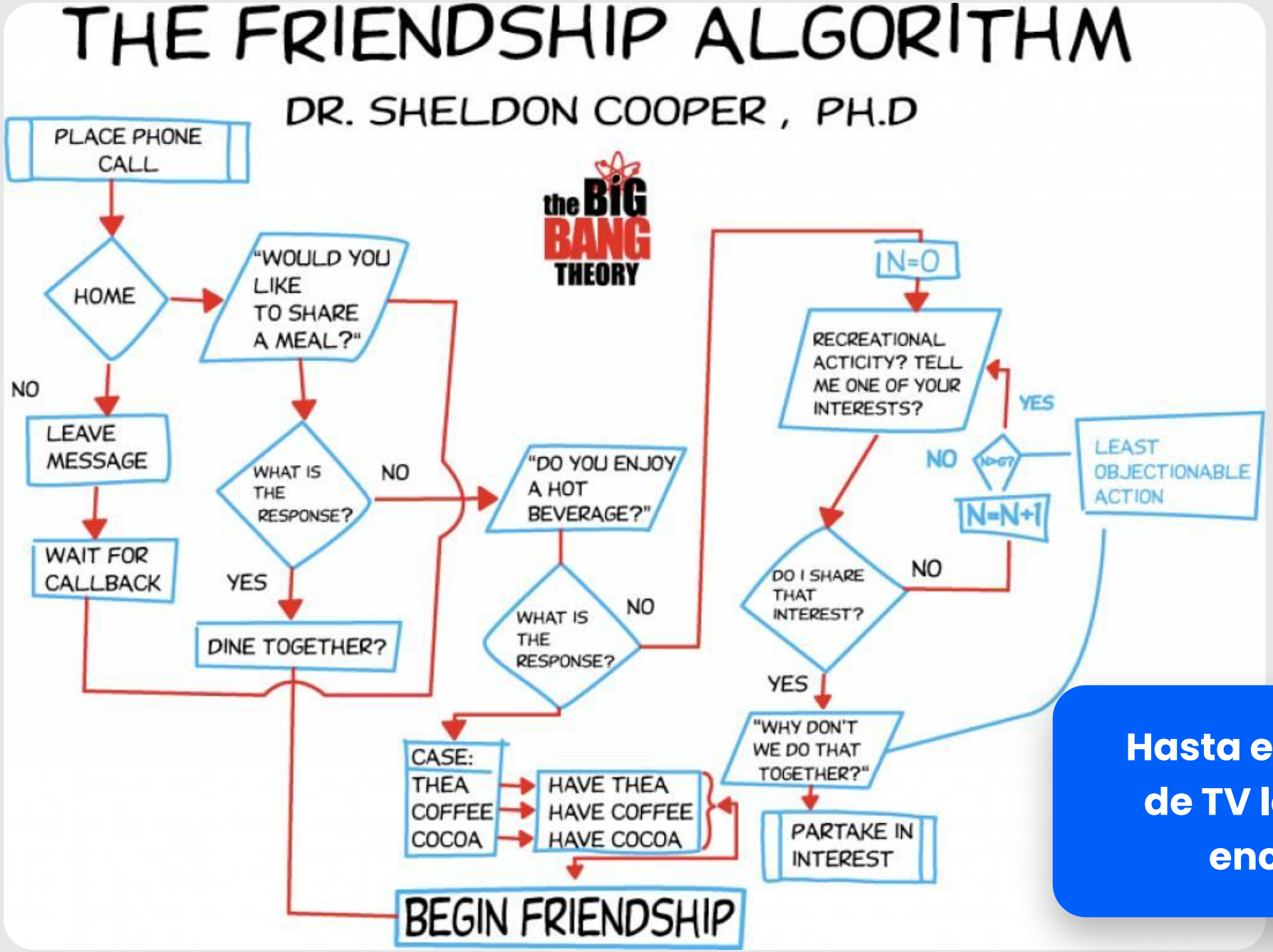
Diseñar algoritmos propios para la resolución de problemas de programación

¿Qué es un algoritmo?

Un algoritmo es un conjunto de instrucciones realizadas en un orden especifico, para solucionar algún problema.

Encontrarás algoritmos en diferentes ámbitos cotidianos.





Hasta en una serie de TV los puedes encontrar

Antes de comenzar a programar...

Antes de ponernos manos a la obra, debemos darnos una idea general de los pasos que seguiremos para lograr resolver nuestro problema.

Hay dos cosas que deberías hacer en papel antes de usar el teclado:

- Refinar
- Diagramar

Refino

Cuando pienses en una solución en tu mente y visualices el paso a paso requerido, debes refinar cada idea para que se te haga mas fácil programarlo luego.

El refino sucesivo es la búsqueda de la solución de un problema, detallando nivel a nivel la solución, es decir, las tareas necesarias de realizar para llevar a cabo una función.

Un problema determinado es separado funcionalmente en distintas tareas que permiten su solución.

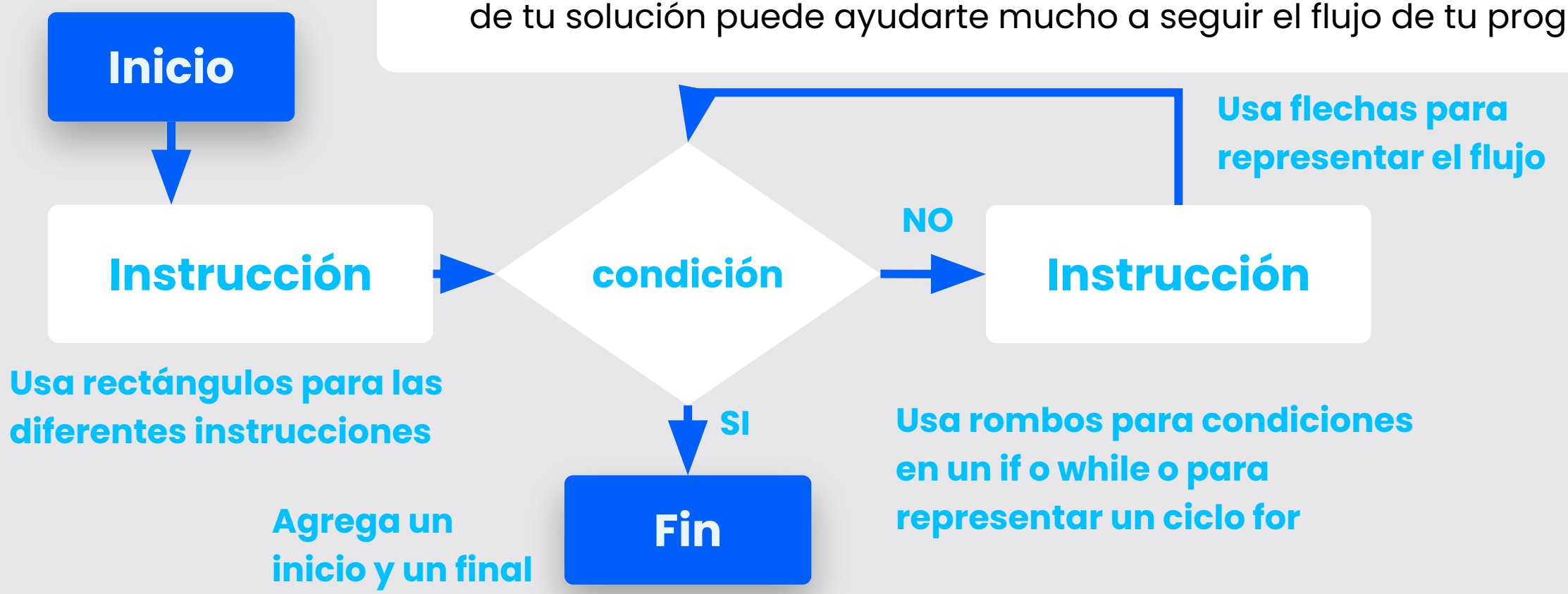
Las tareas definidas, por su parte, pueden nuevamente ser detalladas en subniveles, que determinan un tercer nivel de solución.

Ejemplo

```
Paso 1: leer las tres notas
        Paso 1.1 leer nota 1
        Paso 1.2 leer nota 2
        Paso 1.3 leer nota 3

Paso 2: calcular promedio
        Paso 2.1 sumar nota 1
        Paso 2.2 sumar nota 2
        Paso 2.3 sumar nota 3
        Paso 2.4 dividir el resultado de la suma entre 3

Paso 3: desplegar el resultado
        Paso 2.1 si el promedio es menor o igual a 3.9 desplegar "reprobado"
        Paso 2.2 si el promedio es mayor o igual a 4.0 desplegar "aprobado"
```

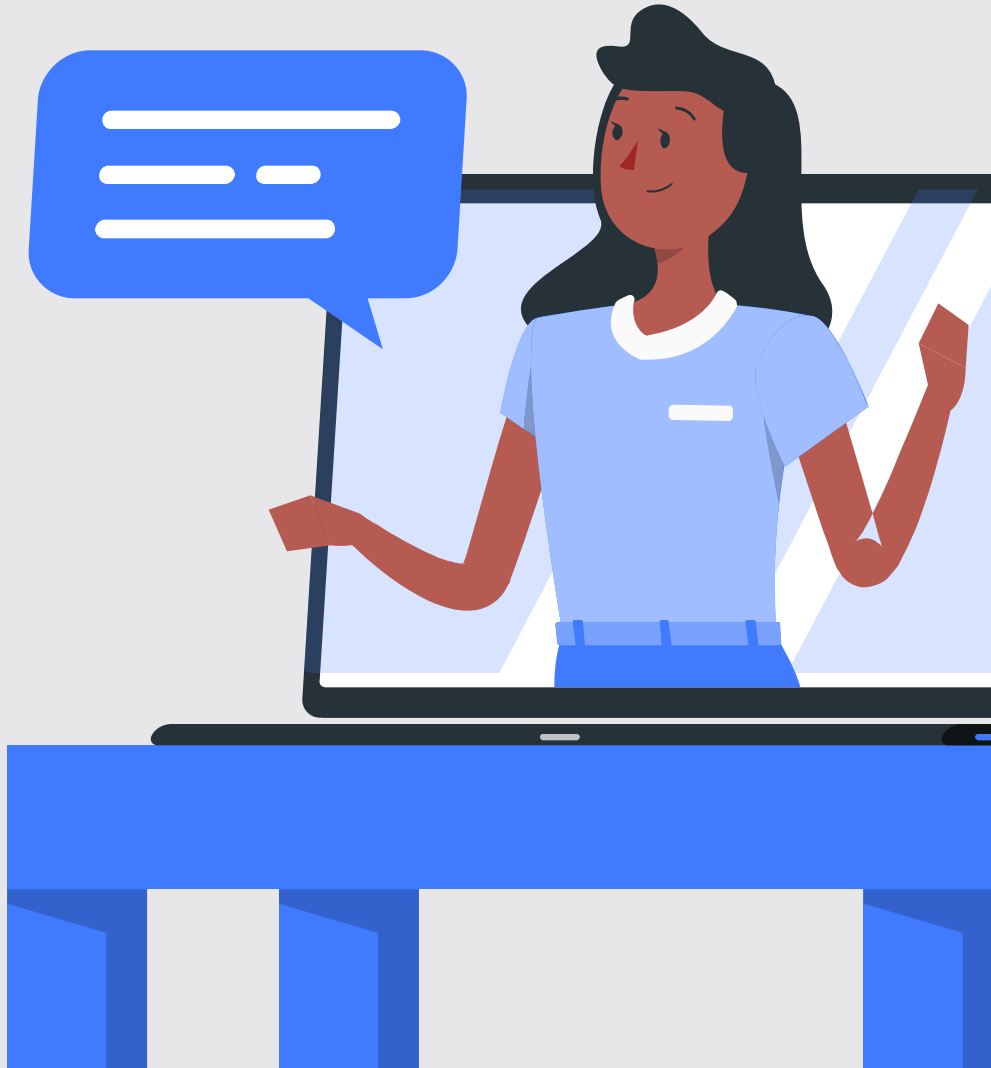
Diagramar

También es bueno crear un diagrama. Tener una representación visual de tu solución puede ayudarte mucho a seguir el flujo de tu programa.



Clase 04: Algoritmos

Vamos con algunos algoritmos



El mayor

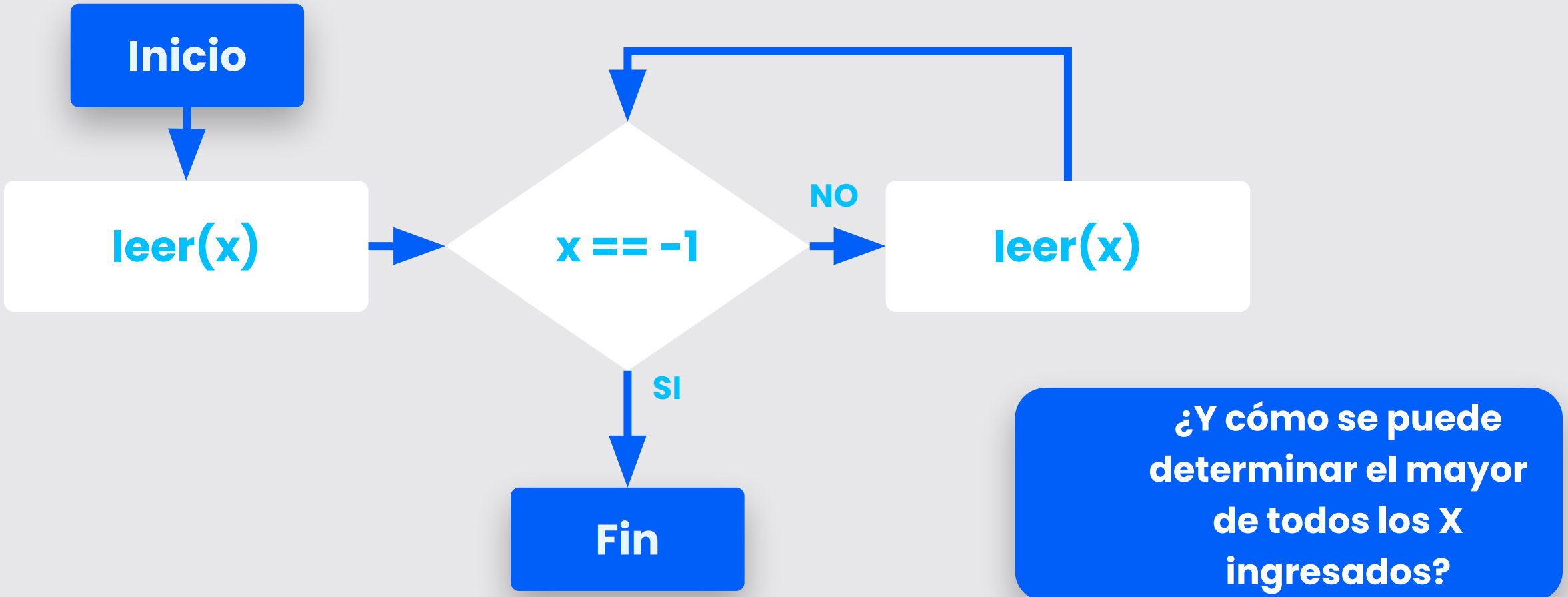
Algo que muchas veces vamos a necesitar es encontrar el número mayor entre una secuencia de valores.

Por ejemplo, si queremos encontrar a la persona con la mayor edad entre un conjunto de personas.

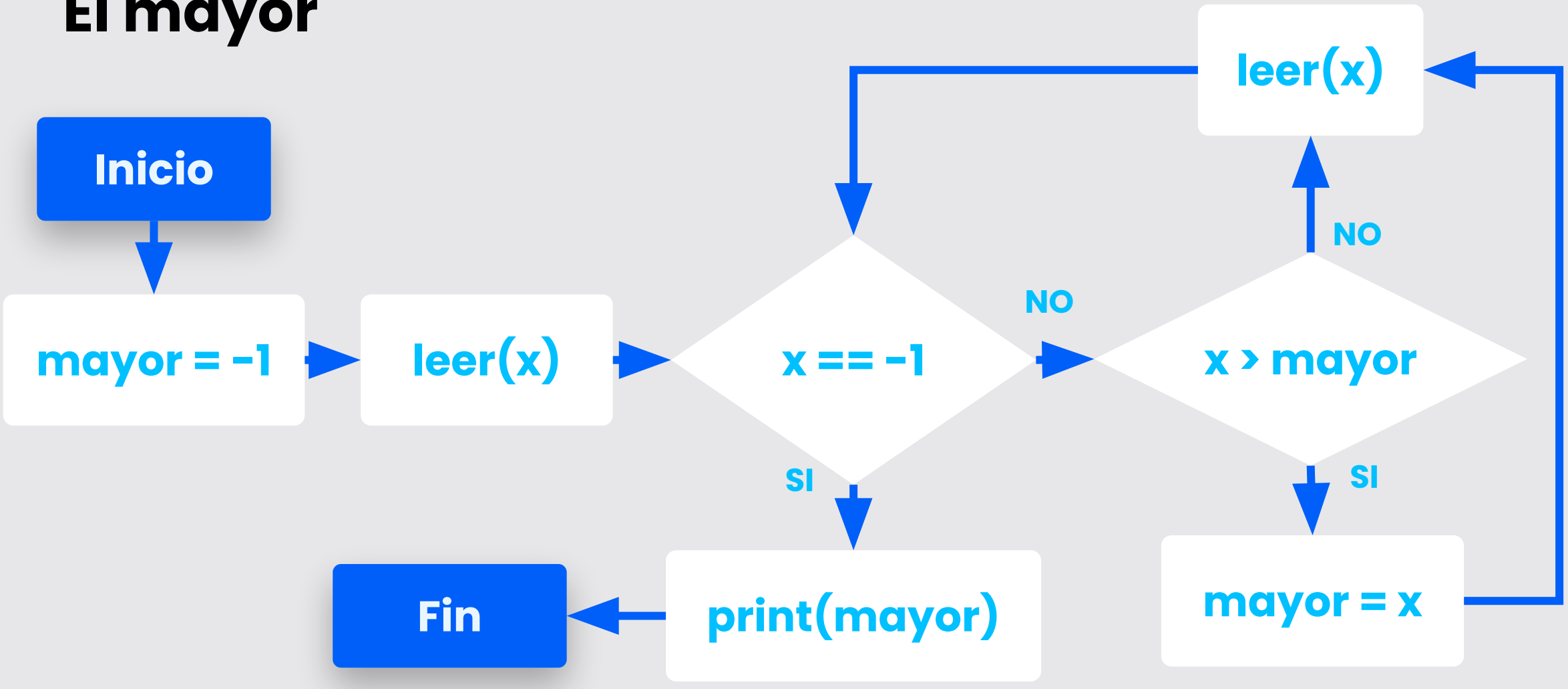
Vamos a simular eso preguntando por el teclado una secuencia de números.

Como el -1 no es una edad válida, vamos a usar ese número para terminar el ingreso de datos (OJO, que el -1 no es una edad, así que no hay que tomarlo en cuenta)

El mayor: Solo ciclo de lectura



El mayor


Inicio
mayor = -1
leer(x)
x == -1
NO
SI
x > mayor
NO
SI
leer(x)
mayor = x
print(mayor)
Fin

El menor

Fíjate que al buscar el mayor, usamos una variable inicializada con un valor “especial”

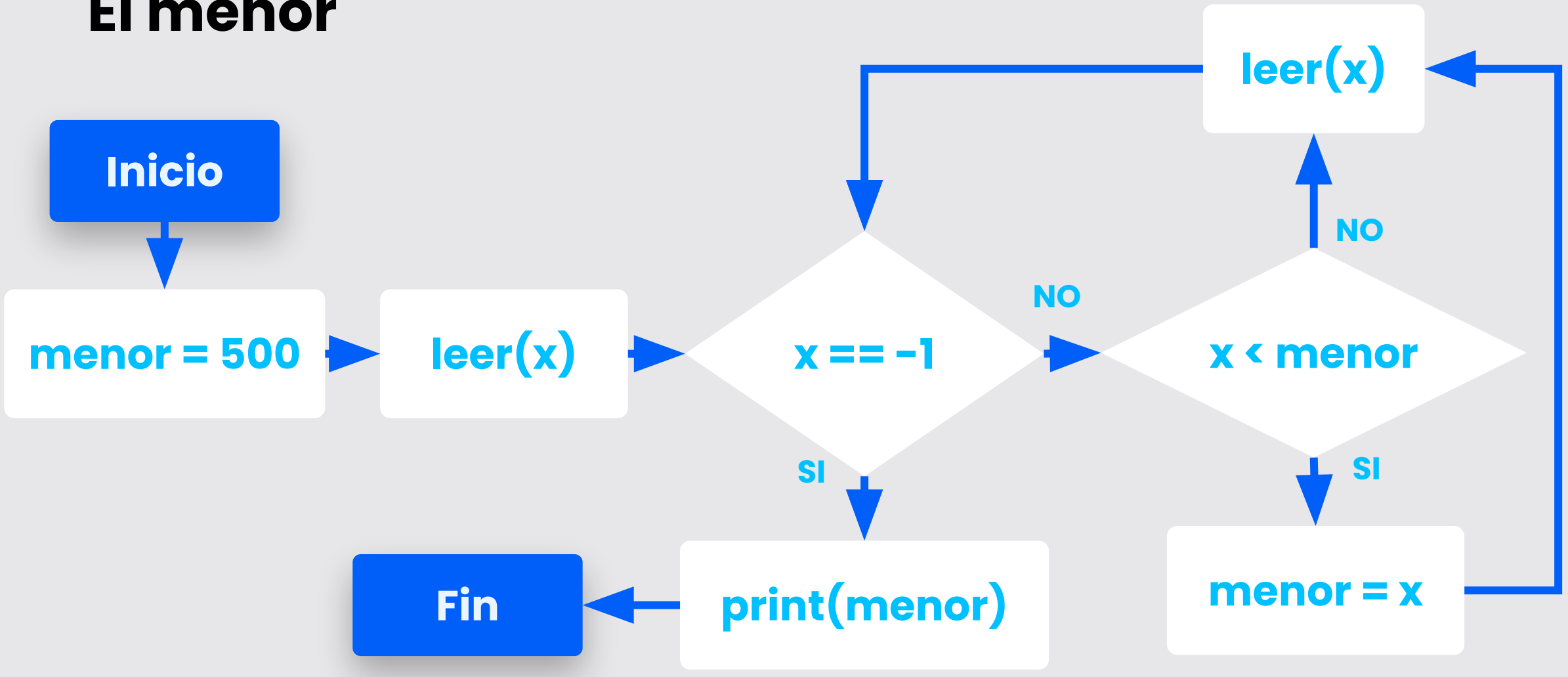
Este valor está fuera del rango válido

- **No hay edades negativas**

¿Qué hay que hacer para buscar la edad menor?

- **Usar un valor fuera del rango válido**

El menor


Inicio
menor = 500
leer(x)
x == -1
NO
SI
x < menor
NO
SI
leer(x)
menor = x
print(menor)
Fin

El promedio de todos los valores

Ahora queremos saber el valor **promedio** de todos los valores ingresados.

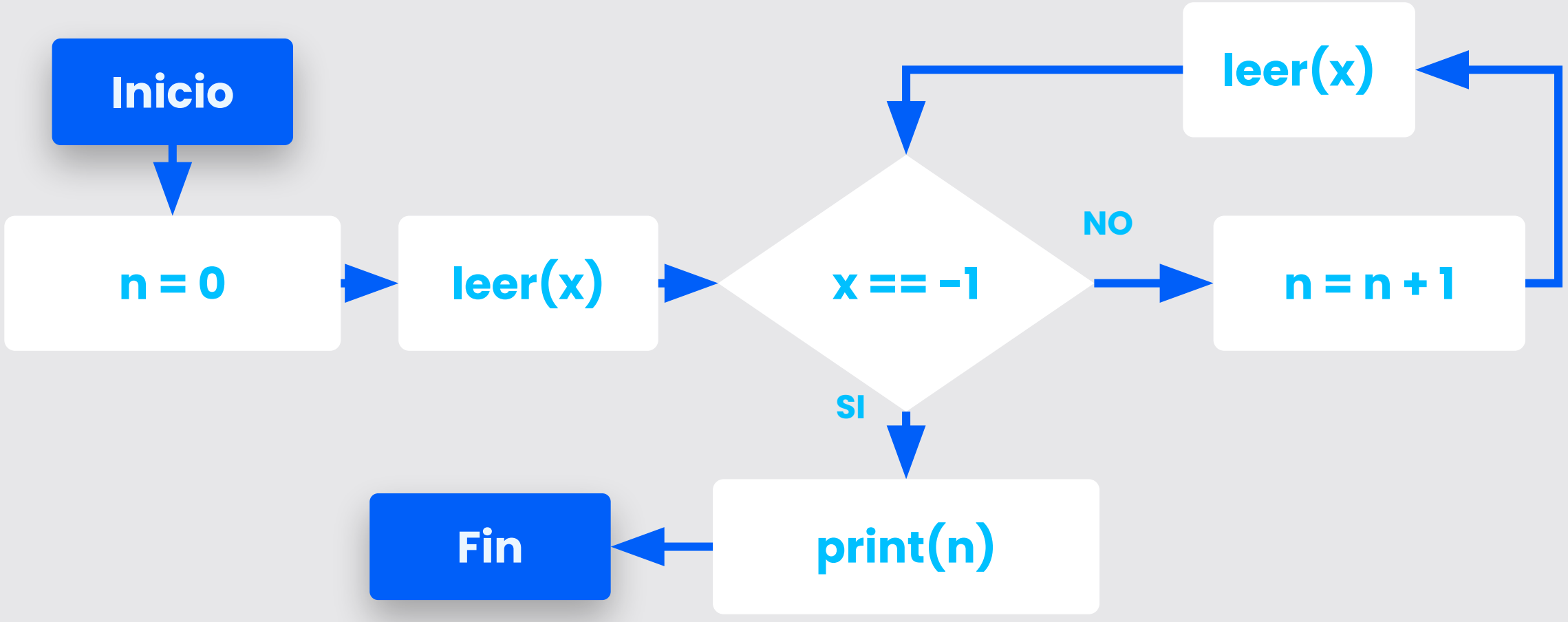
Obviamente esto es simplemente la **suma** de todos los valores, dividido por la **cantidad** de valores

Entonces, necesitamos

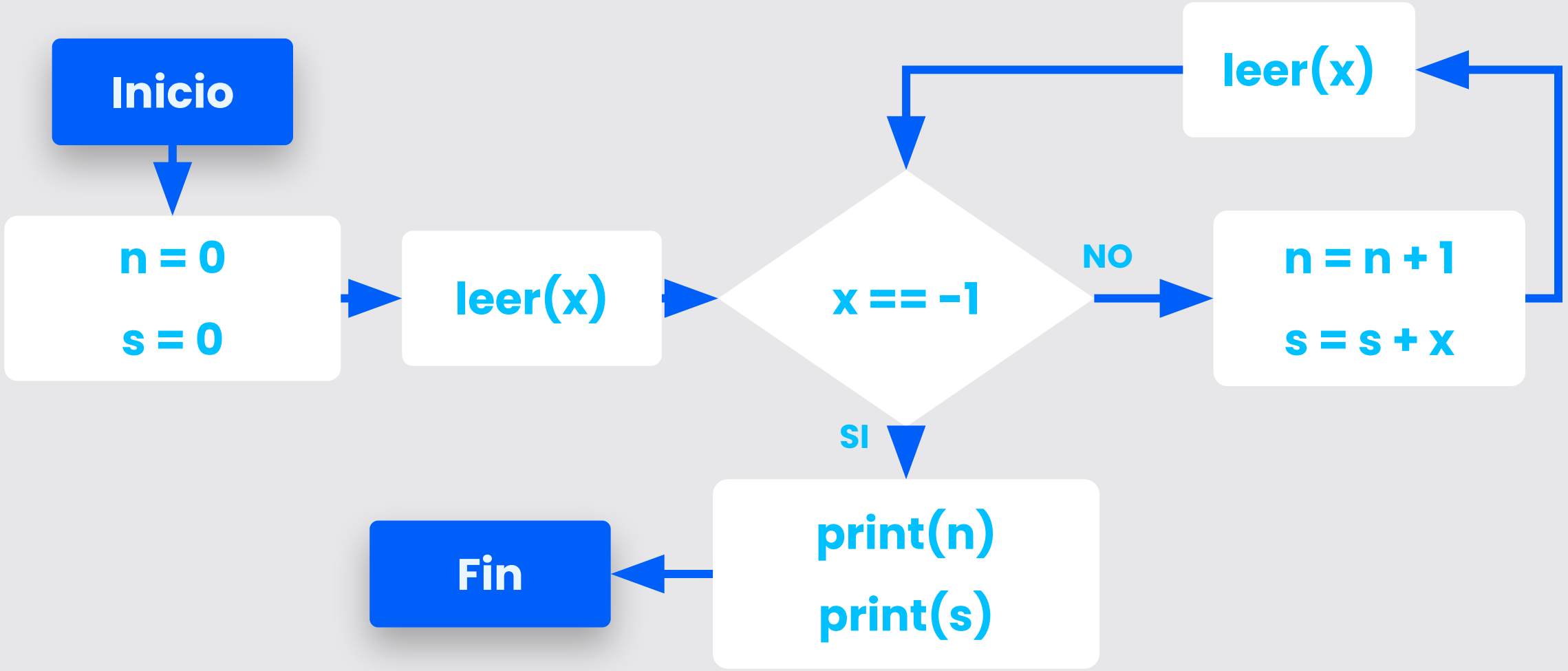
- Un **contador**, para saber cuántos valores hemos ingresados
- Un **acumulador**, para ir "acumulando" los valores a medida que los vamos ingresando

Mantendremos el **-1** como indicador de **fin de datos**

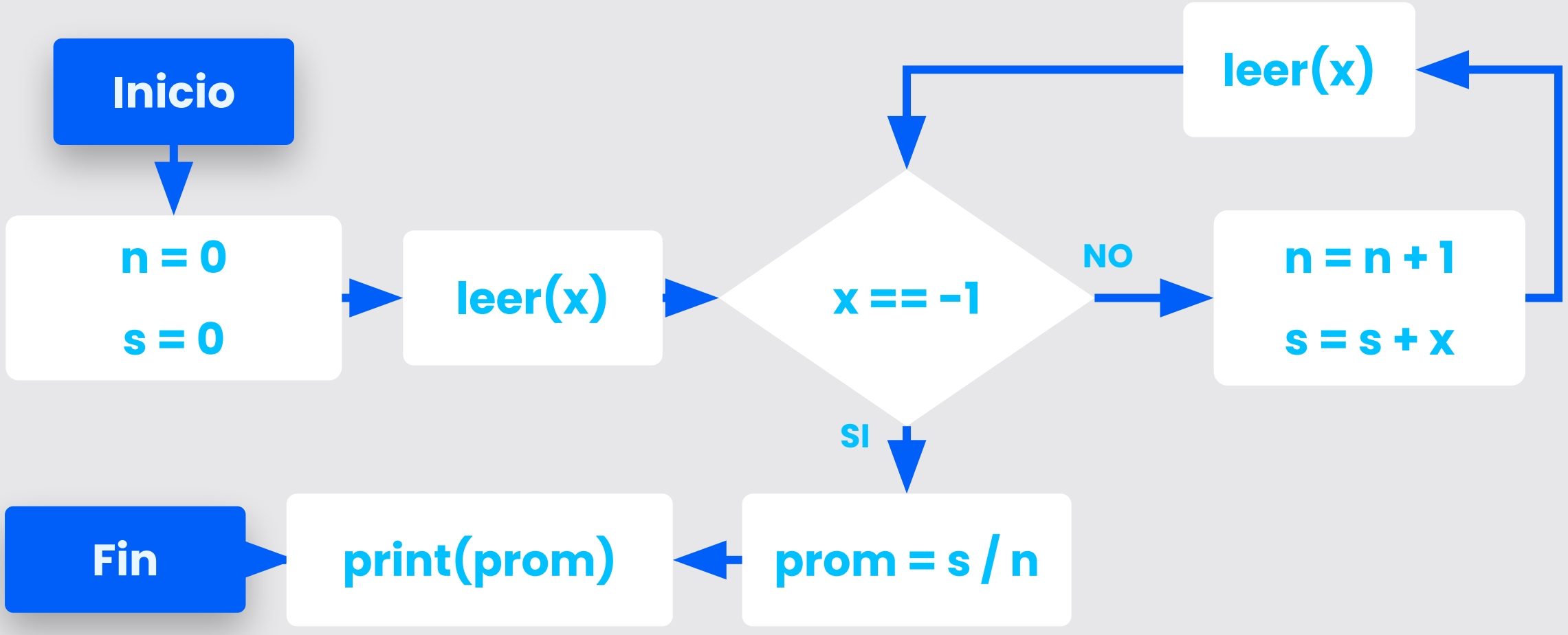
El promedio: contando los valores



El promedio: sumando los valores



El promedio


Inicio
n = 0
s = 0
leer(x)
x == -1
NO
n = n + 1
s = s + x
leer(x)
SI
prom = s / n
print(prom)
Fin

Porcentajes

Imagina que tienes todas tus notas, y quieres saber el % de tus notas que fue superior o igual a 4.0

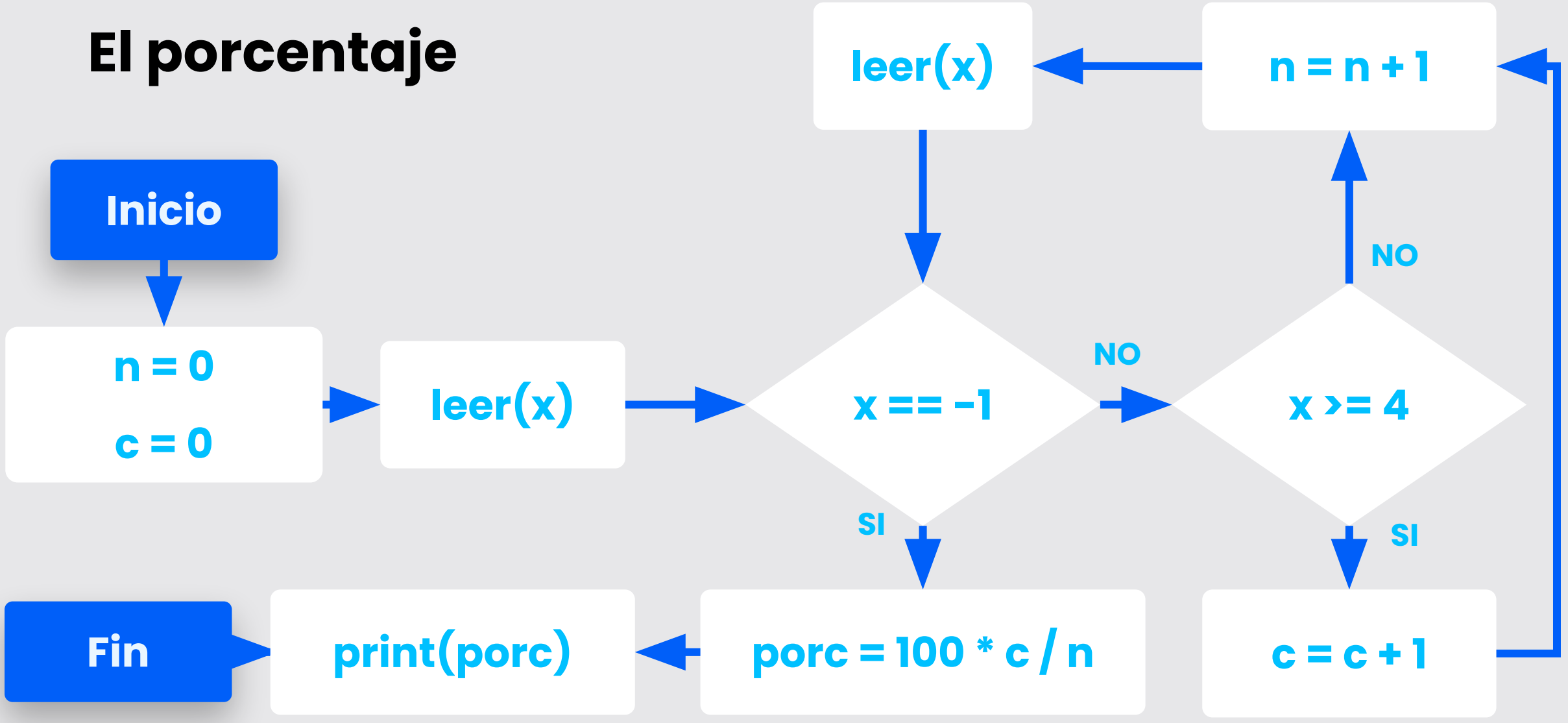
¿Cómo se calcula el % de algo?

Entonces, para poder hacer este cálculo, hay que saber dos cosas:

- La cantidad de notas que cumplen la condición (o sea, >= a 4.0)
- La cantidad de notas en total

Mantendremos el **-1** como indicador de **fin de datos**

El porcentaje



Mayor mejorado

Ya conociste este algoritmo, pero solamente leímos una lista de números.

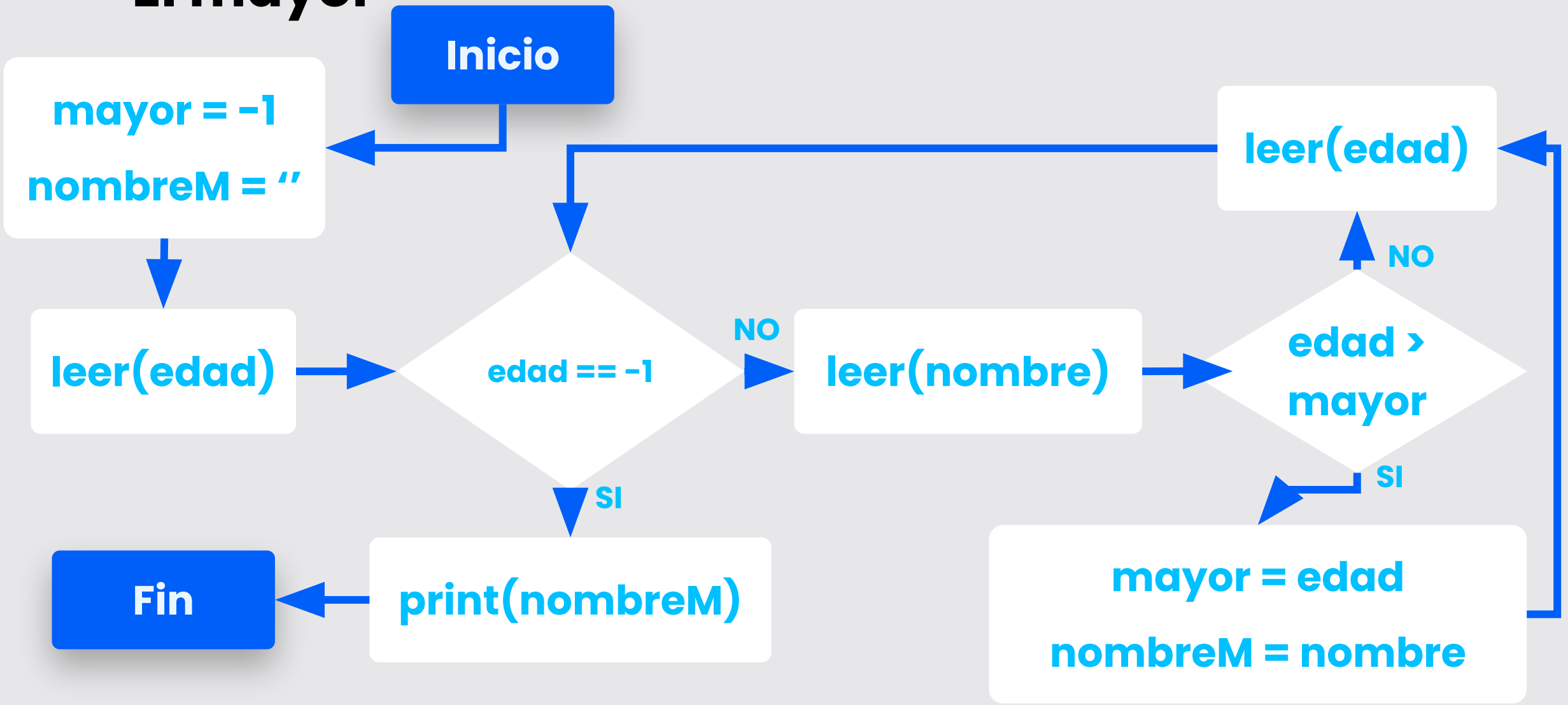
Esto no es tan útil.

Generalmente los números no vienen solos, sino que acompañan a algo más: por ejemplo, nombres.

Imagina que tenemos un conjunto de nombre de personas y sus edades, y queremos saber el nombre de la persona de más edad.

- Mantendremos el -1 como indicador de fin de datos
- **OJO**: Vamos a leer la edad **primero** y **después** el nombre. Si se ingresa un **-1**, no es necesario leer el nombre

El mayor


Inicio
mayor = -1
nombreM = ''
leer(edad)
edad == -1
NO
leer(nombre)
edad > mayor
NO
leer(edad)
SI
mayor = edad
nombreM = nombre
SI
print(nombreM)
Fin

Programación

Clase 03.2

Ejercicios

Ejercicio 1

En base al diagrama visto en clase implementa el mayor.

- Si ingresamos los números 4, 5, 1 y 0, ¿qué resultado te debería entregar? ¿qué resultado te entrega?
- Si los ingresamos en el orden inverso, ¿cambia el resultado?
- Si ingresamos un -1 como primer elemento, ¿qué pasa?
    - Corrige el código para que si esto sucede, diga "no se puede calcular el número mayor"

Ejercicio 2

En base al diagrama visto en clase implementa el menor.

- Si ingresamos los números 4, 5, 1 y 0, ¿qué resultado debería entregar? ¿qué resultado te entrega?
- Si los ingresamos en el orden inverso, ¿cambia el resultado?
- Si ingresamos un -1 como primer elemento, ¿qué pasa?
    - Corrige el código para que si esto sucede, diga "no se puede calcular el número menor"
- ¿Qué pasa si ingresamos 100, 1000, 700?

Ejercicio 3

En base al diagrama visto en clase implementa el promedio.

- Si ingresamos los números 4, 5, 1 y 0, ¿qué resultado te debería entregar? ¿qué resultado te entrega?
- Si los ingresamos en el orden inverso, ¿cambia el resultado?
- Si ingresamos un -1 como primer elemento, ¿qué pasa?
    - Corrige el código para que si esto sucede, diga "no se puede calcular el promedio"

Ejercicio 4

En base al diagrama visto en clase implementa el porcentaje.

- Si ingresamos las notas 1.7, 4.4, 5.5, 2.2, 3.9 y 7.0, ¿qué resultado te debería entregar? ¿qué resultado te entrega?
- Si los ingresamos en el orden inverso, ¿cambia el resultado?
- Si ingresamos un -1 como primer elemento, ¿qué pasa?
    - Corrige el código para que si esto sucede, diga "no se puede realizar el cálculo"

Ejercicio 5

En base al diagrama visto en clase implementa el mayor mejorado.

- Si ingresamos estos datos ¿qué resultado te entrega?
  - 40 juan
  - 30 pedro
  - 21 maria
  - 75 jose
- Si ingresamos un -1 como primer elemento, ¿qué pasa?
- Corrige el código para que si esto sucede, diga "no se puede calcular el número mayor"

Trabajo autónomo mínimo

Revisar capítulo 6 del libro guía.

Resolver ejercicios 9-13.