

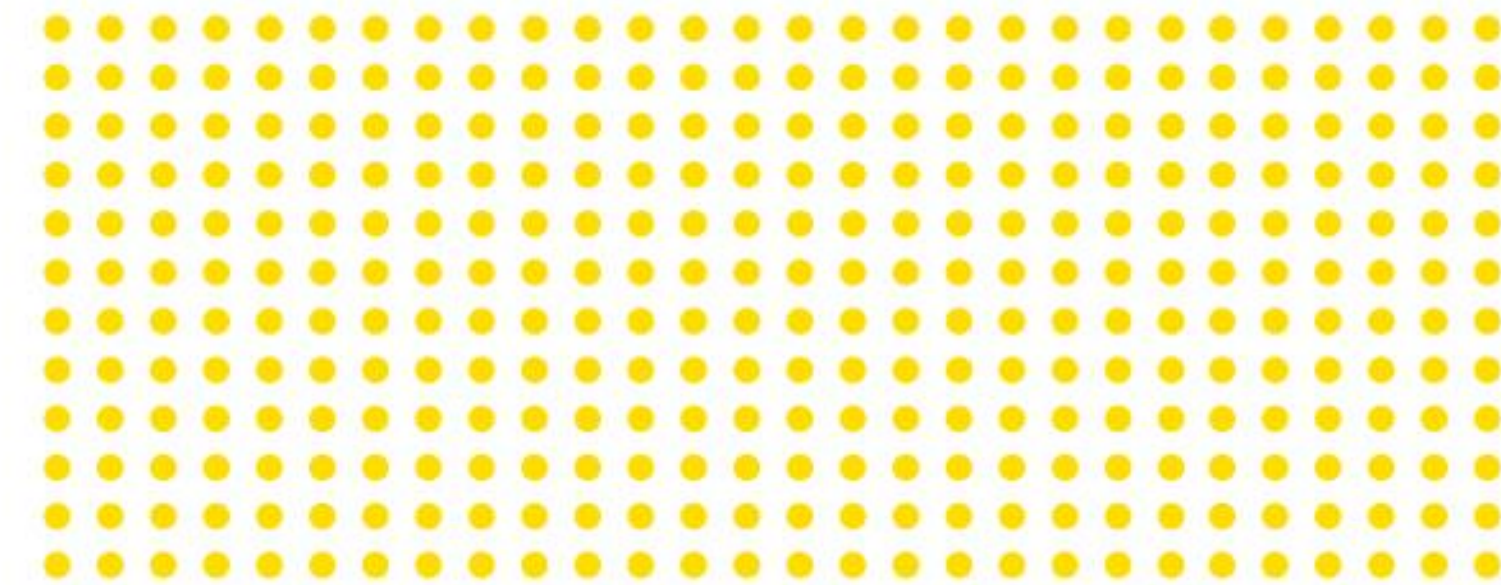
Pensamiento algorítmico: Estructuras lógicas

Fundamentos de programación



Universidad de
los Andes

Educación
Continua
Vicerrectoría Académica

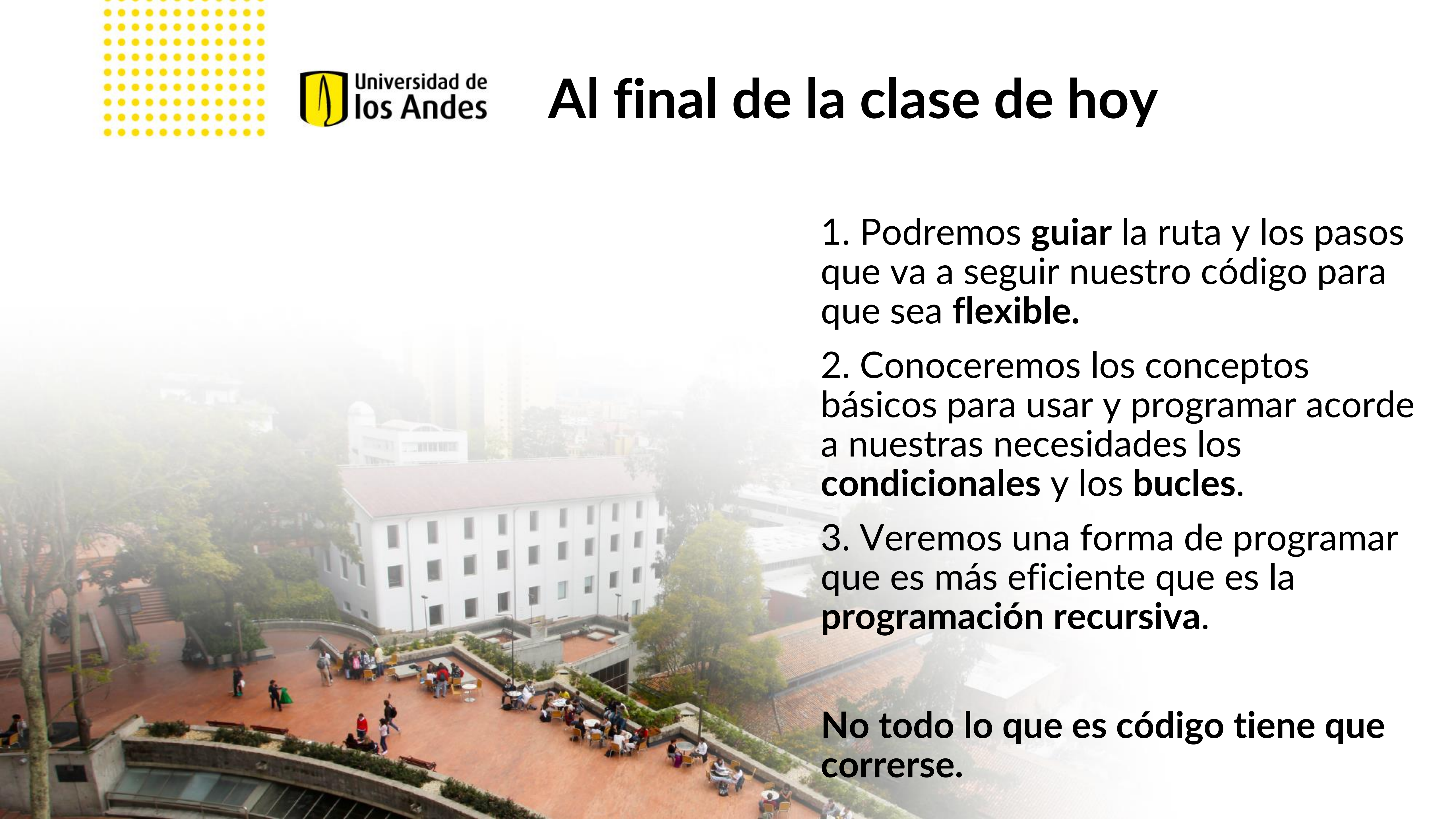




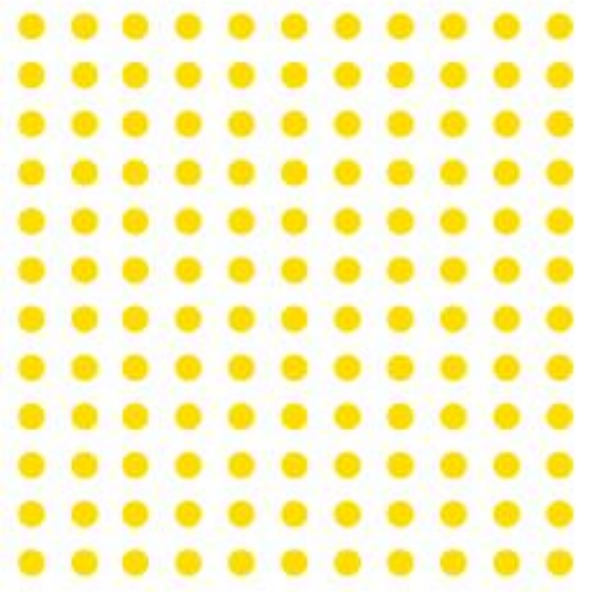
Al final de la clase de hoy

1. Podremos **guiar** la ruta y los pasos que va a seguir nuestro código para que sea **flexible**.
2. Conoceremos los conceptos básicos para usar y programar acorde a nuestras necesidades los **condicionales** y los **bucles**.
3. Veremos una forma de programar que es más eficiente que es la **programación recursiva**.

No todo lo que es código tiene que correrse.



Contenido

- 
- Condicionales
 - If, else, elif
 - Bucles
 - Qué es
 - While loop
 - For loop
 - Programación recursiva



Ahora vamos a reutilizar tareas

Condicionales (if, else, elif)



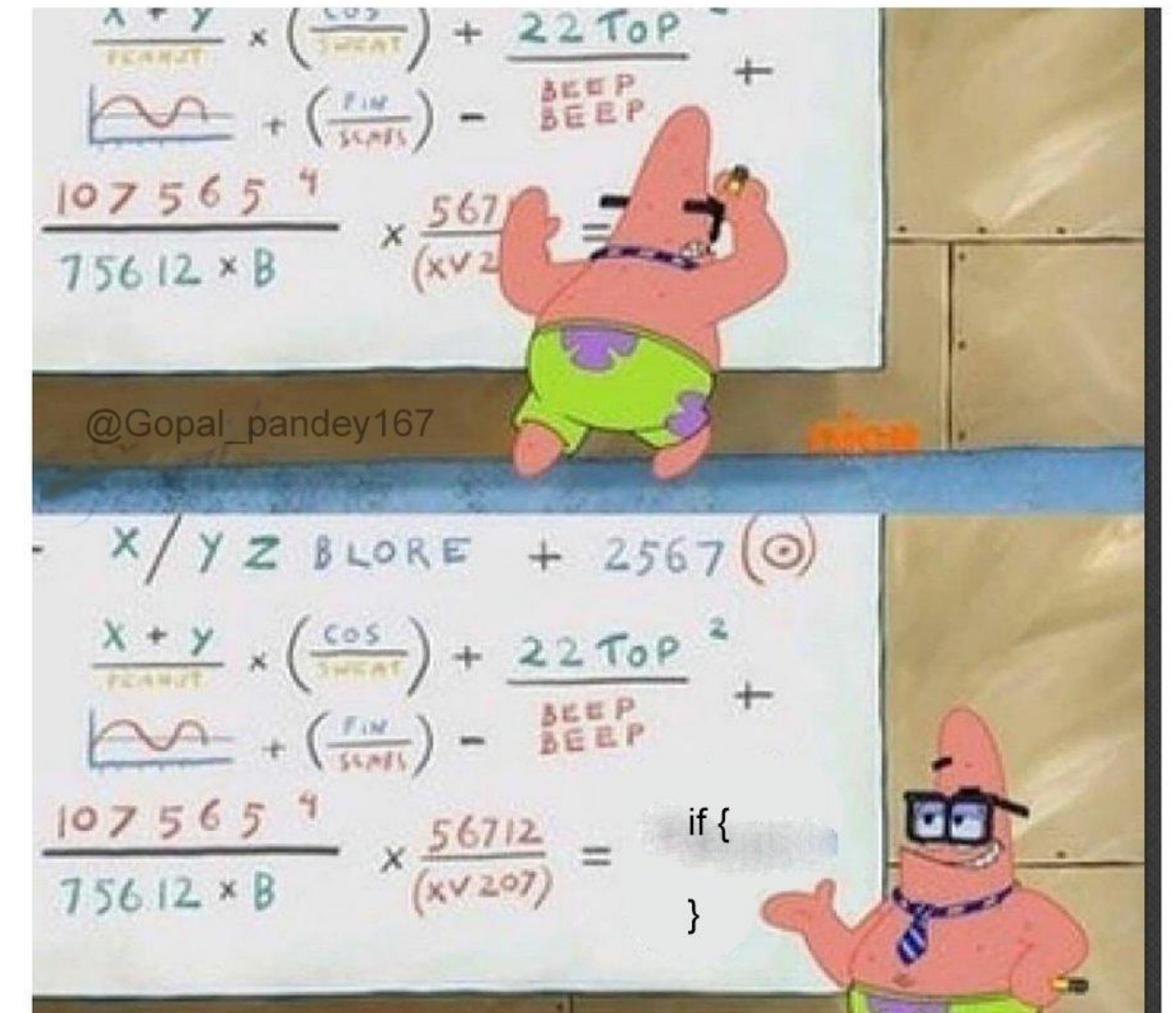
Ejemplo

Vamos a salir a comer, si hace sol vamos a un lugar campestre, llevamos gafas de sol, etc.

De lo contrario, vamos a un restaurante cerrado, nos vestimos formal y volvemos temprano.

Estas son condiciones establecidas que cambian la forma de actuar de nosotros y los algoritmos.

When you are making AI



Condicionales



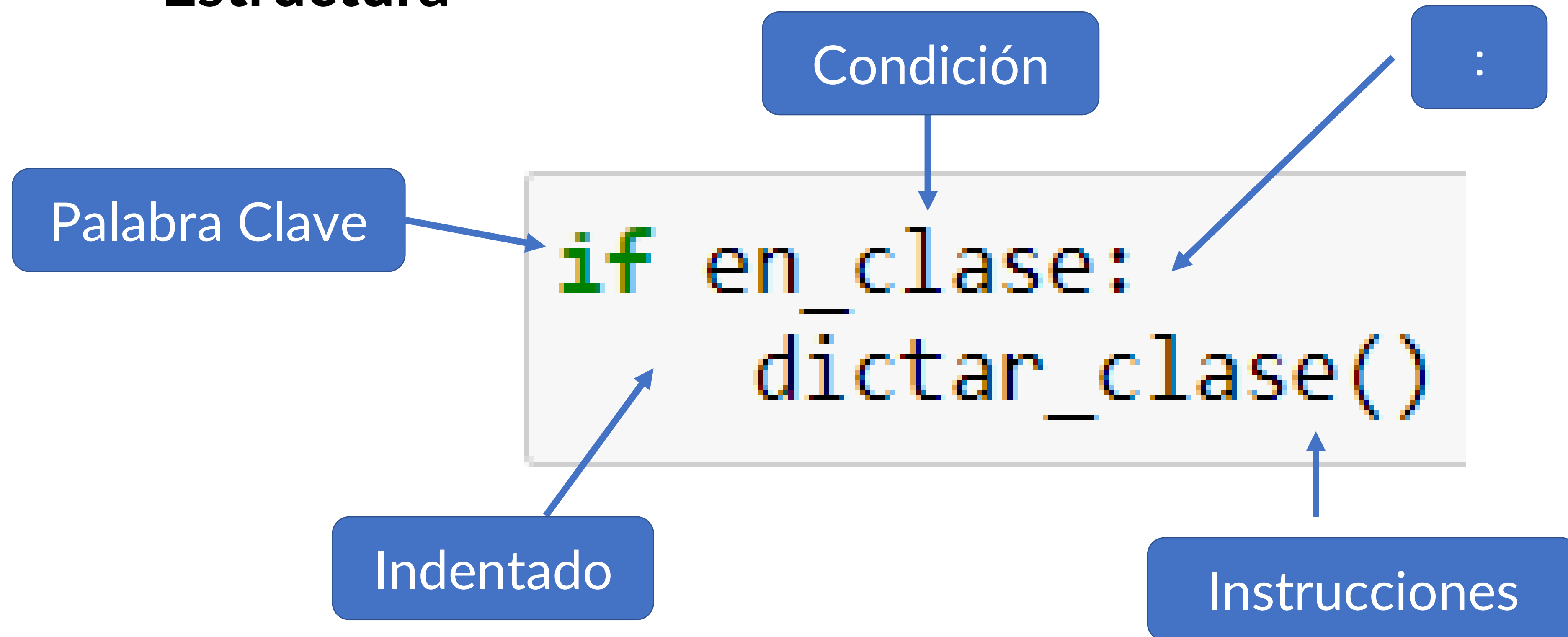
En Python tenemos dos familias básicas de condicionales

- Familia (If)
- Familia (Try)

if

Para hacer uso del “if” es muy simple.

Estructura



If else

¿ Y si no se cumple el if entonces se pierde todo? No

Estructura

Palabra Clave

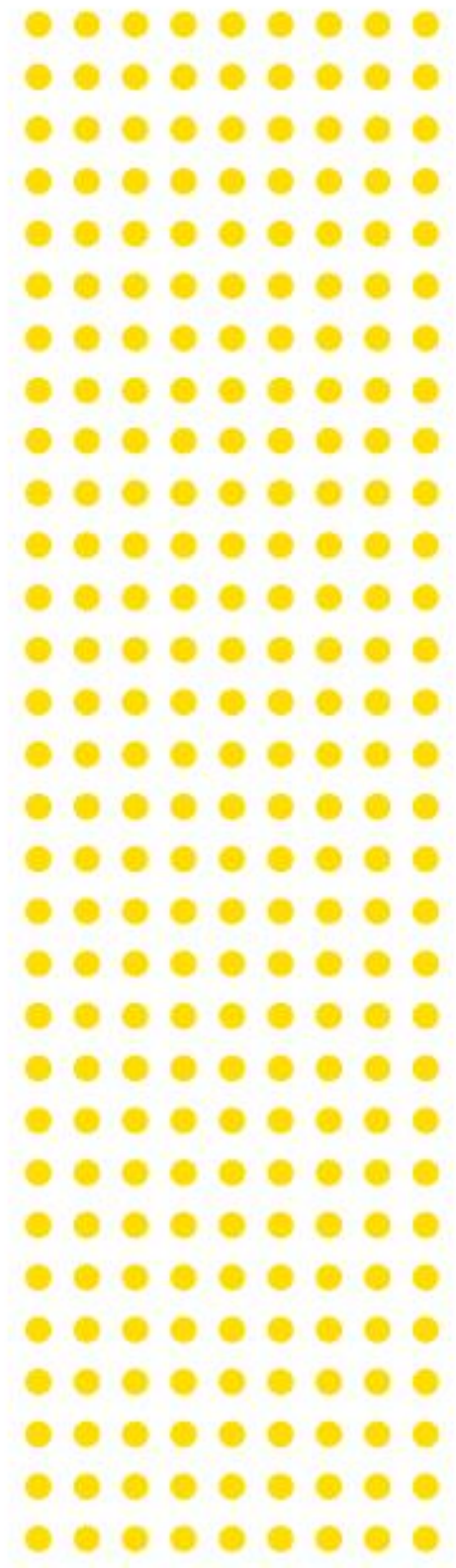


```
if en_clase:  
    dictar_clase()  
else:  
    dormir()
```

Palabra Clave 2



No hay condicional, solo :



elif

¿ Y si hago más de una condición? Entra a jugar el if else

Estructura

```
if en_clase:  
    dictar_clase()  
elif trabajando:  
    finjir_trabajar()  
elif comiendo:  
    comer()  
    comer()  
else:  
    dormir()
```

¿Cuántos **elif** puedo usar?

elif

Historia patria:

En algunos lenguajes se concatenaba el **else** con el **if**, de esto nace el **elif**



Ejercicio

Usted es responsable de desarrollar un simulador, para que las personas consulten si son elegibles para un subsidio. Para esto usted debe validar las siguientes condiciones: Genero_Femenino, menor de 25 años, con hijos.

Desarrolle el pseudocódigo para una aplicación, que pregunta estos tres datos y da como respuesta si puede recibir el subsidio.



Ahora vamos a reutilizar tareas

Bucles- Loops (while, for)



¿Qué es un bucle (Loop)?

Un bucle es repetir una serie de pasos un número x de veces

En programación existen 3 tipos de bucles que son los más conocidos y básicos

- While
- Do While
- For



¿Qué es un bucle (Loop)?

Un bucle es repetir una serie de pasos un número x de veces

En programación existen 3 tipos de bucles que son los más conocidos y básicos

- While
- ~~Do While~~
- For

Python no tiene Do
While



Ejemplo

Imagine que es un profesor de un colegio y mientras está sacando las notas de los estudiantes decide duplicar la nota más alta de cada uno.

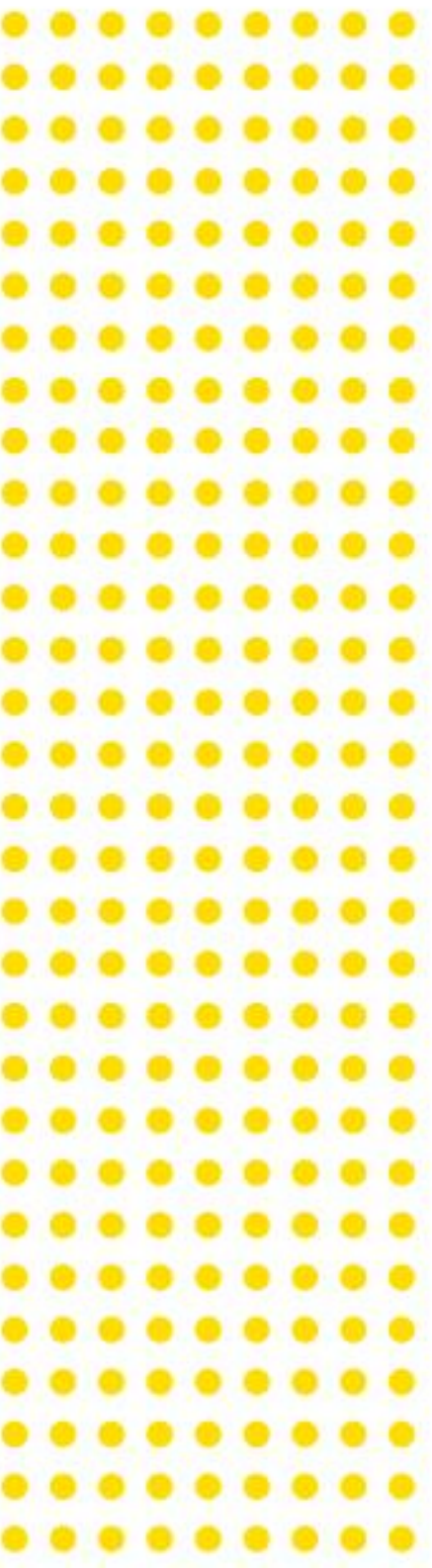
Para esto, debe mirar las notas de cada estudiante y sacar la nota mayor, luego la vuelve a copiar.

Esto sería un loop, una tarea que va a repetir por cada estudiante.



Diferencia entre un loop y una función

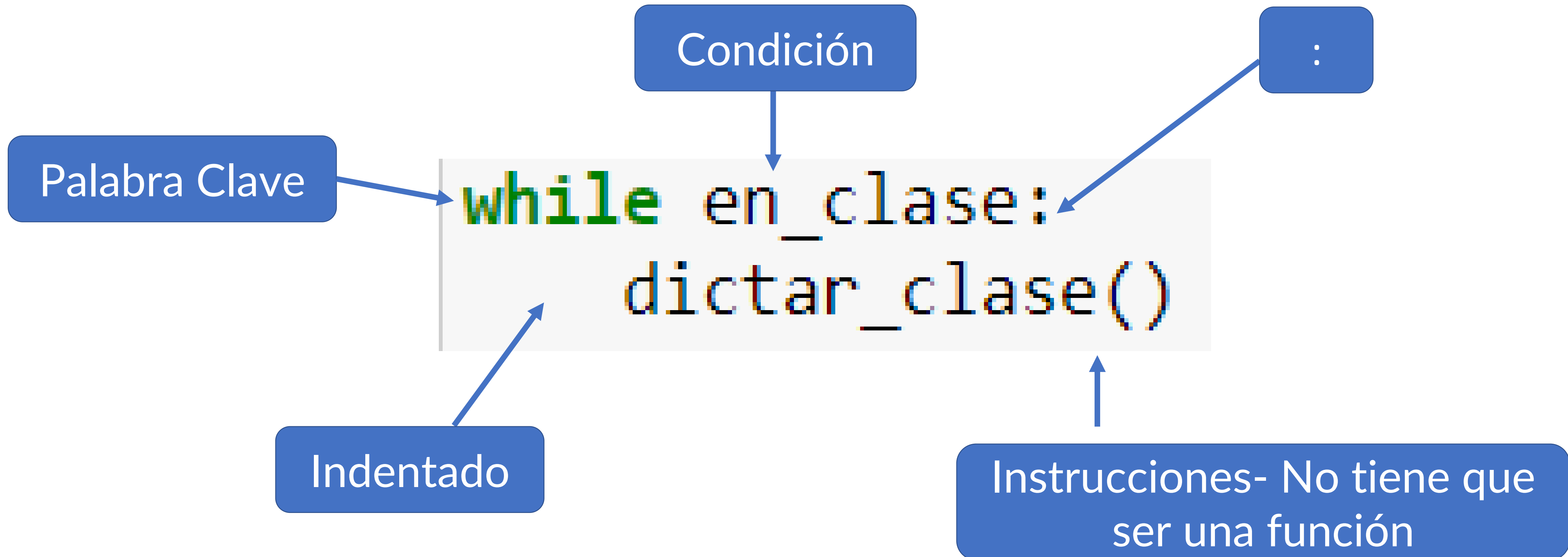
- La principal diferencia es que el loop se va a repetir de una forma seguida, mientras que una función se repite solo cuando yo la llamo.
- Un loop suele ser un paso o una instrucción simple que se tiene que realizar para cada elemento de un grupo.
- El loop es usualmente usado para crear, revisar o modificar valores de una estructura como una lista o un diccionario.



While

El While loop repite las instrucciones hasta que la condición no se cumpla.

Estructura



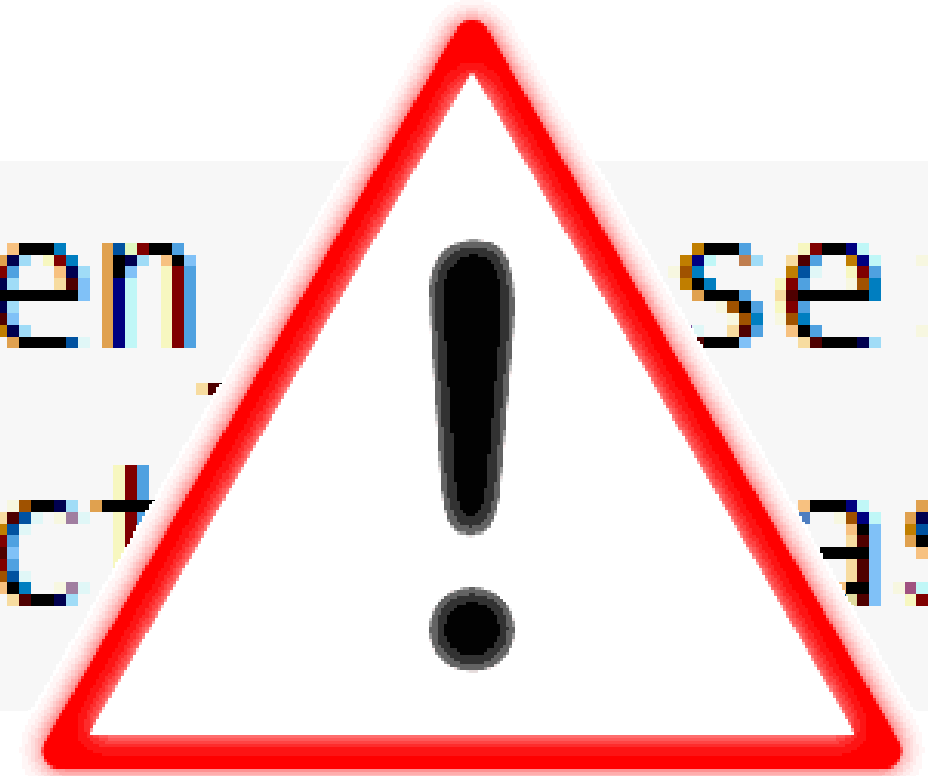
While

```
while en_clase:  
    dictar_clase()
```



While

```
while en...se:  
    dict...ase()
```



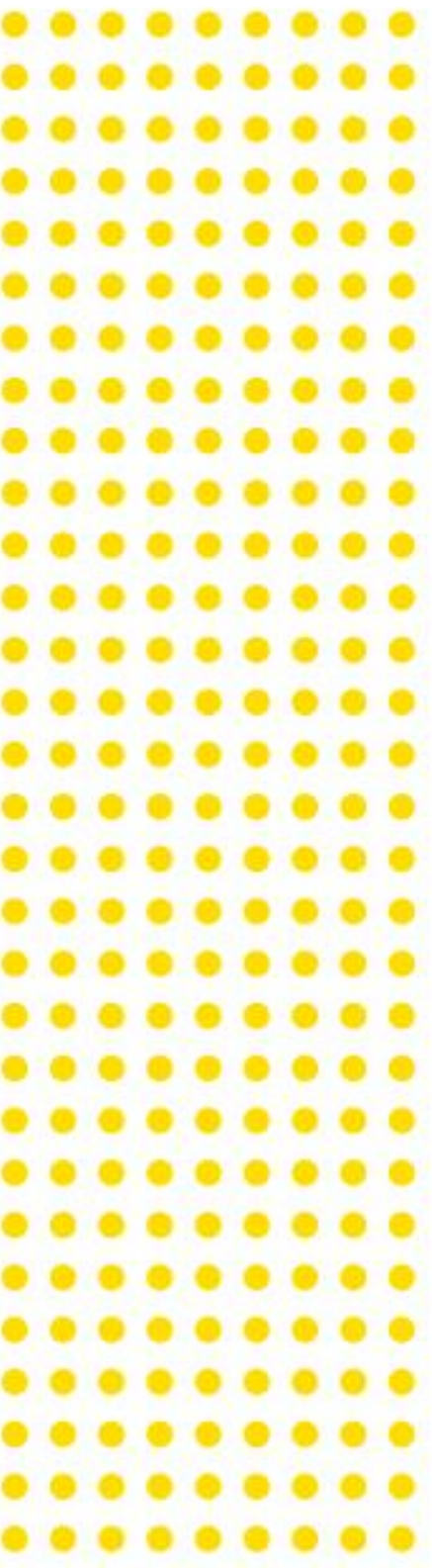
DANGER

while

Tenemos que tener cuidado con los **Loops infinitos**.

En este ejemplo no se especifica ninguna condición que cambie la condición inicial.

De igual forma es posible que un While loop no se ejecute nunca.



while

Acá se ejecutaría una sola vez

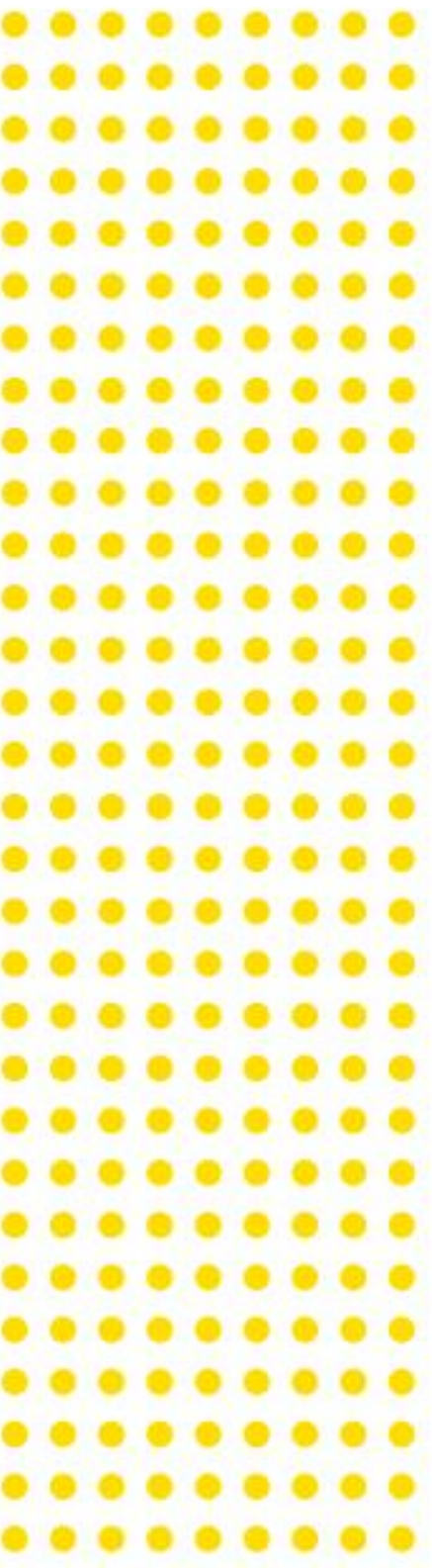
```
en_clase = True
while en_clase:
    dictar_clase()
    en_clase = False
```


for

Es el tipo de Loop más usado.

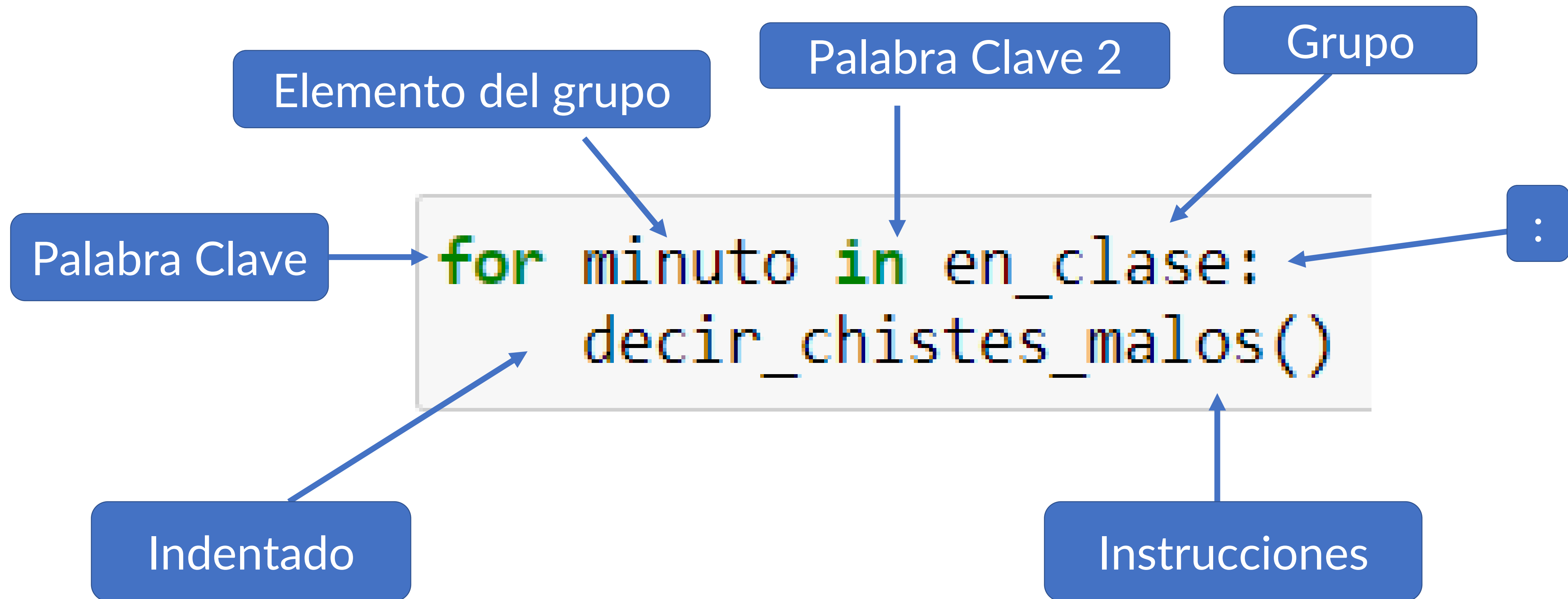
Este tipo de loop va a repetir las instrucciones un número específico de veces.

¿Cuántas veces?



For- forma 1

Estructura



For- forma 2

Estructura

Cantidad de veces

```
for x in range(2):  
    print (x)
```

Empieza en 0

0

1

No llega a 2

Range es una estructura que marca un comienzo y un final.

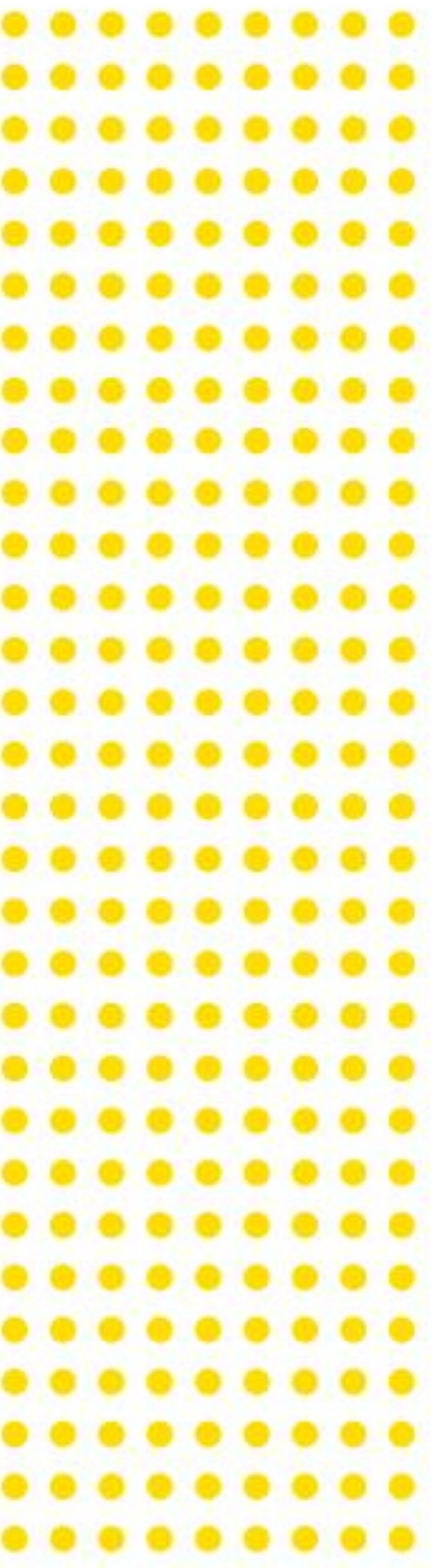
```
range(3)
```

```
range(0, 3)
```


for

Python es inteligente, permite usar el For Loop con varios tipos de variables de una forma “intuitiva”:

- Listas
- Tuplas
- Strings
- Diccionarios
- Conjuntos





Ejercicio

Usted tendrá la siguiente lista de nombres en Python: [Sergio, Alfredo, Alejandra, David]

Escriba el pseudocódigo para imprimir en Python:

“Buenos días, Sergio”

“Buenos días, Alfredo”

“Buenos días, Alejandra”

“Buenos días, David”

- Si se modifica la lista y se agrega a “Miguel”, el pseudocódigo no debe cambiar.

Ahora vamos a reutilizar tareas

Programación recursiva



Recursividad

Bueno, ya vimos funciones, bucles y condicionales.... Vamos a unir todo esto para crear la forma de programar que es más efectiva.



Recursividad

En pocas palabras una recursión es una función que se llama a sí misma.



Es una forma de descomponer una tarea para que realice una función y se vuelva a repetir hasta completar el programa.

Ejercicio

Diseñe el pseudocódigo para una función que reciba un número y calcule su factorial. Tenga en cuenta que:

$n! = 1$, para $n = \{0,1\}$

$n! = (n-1)! * n$ $n > 1$



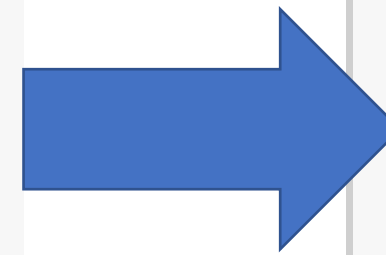


Recursividad



Ejemplo

```
def buscar (n, lista):  
    for i in lista:  
        if i == n:  
            return True  
    return False
```



```
def buscar (n, lista):  
    if len(lista) == 0:  
        return False  
    if lista[0] == n:  
        return True  
    return buscar(n, lista[1:])
```




Con todo esto...

1. Ya sabemos cómo controlar el flujo del código.
2. Sabemos diferentes formas de hacer que un código sea más flexible para diferentes tareas.
3. Tenemos en cuenta la eficiencia de un código, no solo que al final funcione.



¡Gracias!

Aprendiendo juntos a lo largo de la Vida

educacioncontinua.uniandes.edu.co

Síguenos en **EdcoUniandes**

