# PROGRAMACIÓN 2

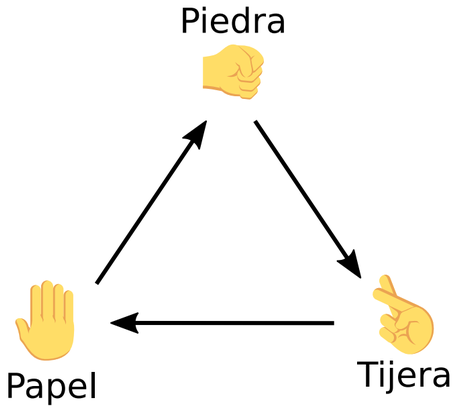
Guía 4

**DAVID VALENCIA SANDOVAL**
INGENIERO EN SISTEMAS
david.valencia@ciaf.edu.co

# Actividad de Calentamiento

Escriba un programa que simule el juego Piedra, papel, tijera para dos jugadores (Inés y Juan). Las reglas del juego son las siguientes:

- Los dos jugadores muestran una mano en tres posibles posiciones:
- Piedra: se muestra el puño cerrado.
- Papel: se muestra la palma de la mano.
- Tijera: se muestra la palma de la mano con los dedos separados en dos grupos.
- El jugador que ha sacado Piedra gana al jugador que ha sacado Tijera.
- El jugador que ha sacado Tijera gana al jugador que ha sacado Papel.
- El jugador que ha sacado Papel gana al jugador que ha sacado Piedra.
- Si sacan lo mismo nadie gana.




```
PIEDRA, PAPEL, ... ¡TIJERA!
Inés ha sacado piedra.
Juan ha sacado papel.
Ha ganado Juan.
```

```
PIEDRA, PAPEL, ... ¡TIJERA!
Inés ha sacado tijera.
Juan ha sacado tijera.
Han empatado.
```

# Ciclos

Los ciclos, también conocidos como bucles o loops, son estructuras que permiten repetir una secuencia de instrucciones múltiples veces. Se usan cuando necesitamos realizar una tarea repetitiva hasta que se cumpla una condición específica.

## Ciclo while:

Un ciclo **while** permite repetir la ejecución de un grupo de instrucciones mientras se cumpla una condición (es decir, mientras la condición tenga el valor True).

La sintaxis del bucle **while** es la siguiente:

```
while condicion:
    cuerpo del bucle
```

**La ejecución de esta estructura de control while es la siguiente:**

Python evalúa la condición:

- si el resultado es True se ejecuta el cuerpo del bucle. Una vez ejecutado el cuerpo del bucle, se repite el proceso (se evalúa de nuevo la condición y, si es cierta, se ejecuta de nuevo el cuerpo del bucle) una y otra vez mientras la condición sea cierta.
- si el resultado es False, el cuerpo del bucle no se ejecuta y continúa la ejecución del resto del programa.

# Ciclo While

```
i = 1
while i <= 3:
    print(i)
    i += 1
print("Programa terminado")
```

La variable o las variables que aparezcan en la condición se suelen llamar variables de control. Las variables de control deben definirse antes del bucle while y modificarse en el bucle while.

Por ejemplo, el programa anterior escribe los números del 1 al 3:

```
1
2
3
Programa terminado
```

Tendremos este resultado

# Ciclo While

```
i = 1
while i <= 3:
    print(i)
    i += 1
print("Programa terminado")
```

Declaramos la variable de control en el numero que necesitamos que empiece

Creamos la sentencia **While** con la condición de control, en este caso hasta que sea menor o igual a 3

Imprimimos la variable **i** para mostrar los números que van

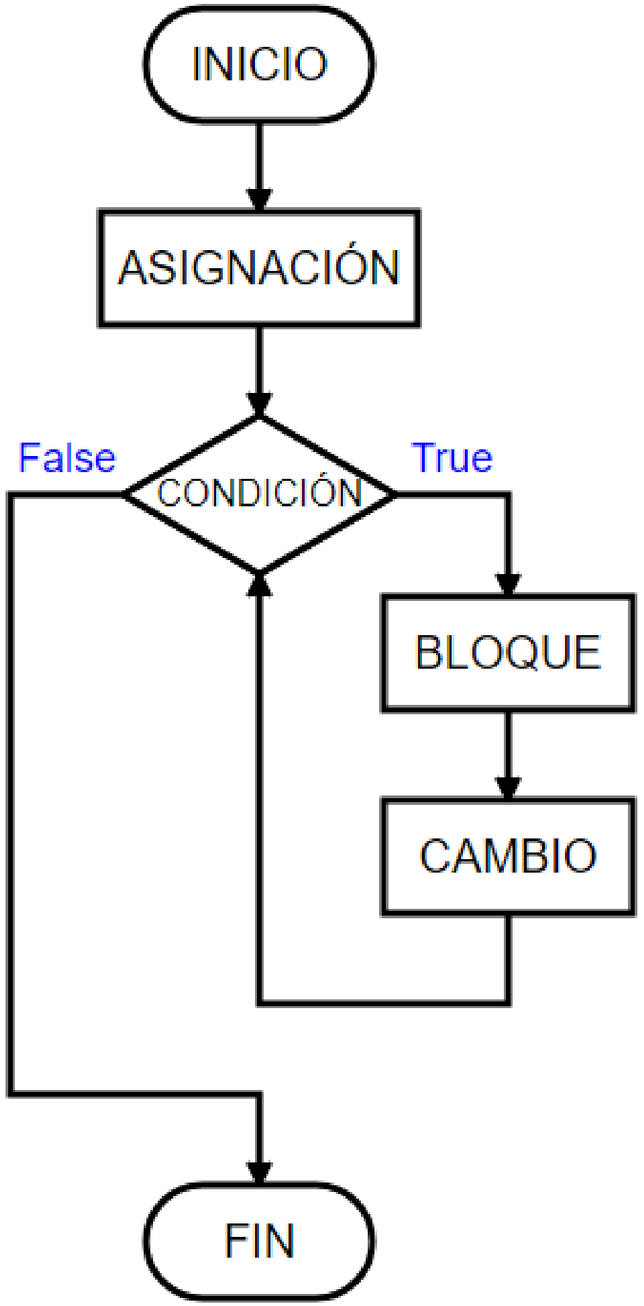
Cada vez que entra a el **While** le sumamos un 1 a la variable **i**, esto se hace para que el **While** tenga fin y salida.

Por ejemplo, el programa anterior escribe los números del 1 al 3:

```
1
2
3
Programa terminado
```

Tendremos este resultado

# Ciclo While



# Ciclo While

La ventaja de un bucle **while** es que la variable de control se puede modificar con mayor flexibilidad, como en el ejemplo siguiente:

```
i = 1
while i <= 50:
    print(i)
    i = 3 * i + 1
print("Programa terminado")
```

En este caso, da el valor 1 a la variable i. La variable se utilizará como variable de control en el bucle while posterior.

La condición es que la variable de control sea inferior o igual 50. Como **i** vale 1, la condición se cumple, así que se pasa a ejecutar las instrucciones del bucle.

A continuación se ejecutan las instrucciones del bloque.

A continuación se modifica la variable de control **i**, multiplicando su valor por 3 y sumando 1.

# Ciclo While

Otra ventaja del bucle **while** es que el número de iteraciones no está definida antes de empezar el bucle, por ejemplo porque los datos los proporciona el usuario. El siguiente ejemplo pide un número positivo al usuario una y otra vez hasta que el usuario lo haga correctamente:

```
numero = int(input("Escriba un número positivo: "))
while numero < 0:
    print("¡Ha escrito un número negativo! Inténtelo de nuevo")
    numero = int(input("Escriba un número positivo: "))
print("Gracias por su colaboración")
```

# Ciclo While

## Bucles infinitos

Si la condición del bucle se cumple siempre, el bucle no terminará nunca de ejecutarse y tendremos lo que se denomina un bucle infinito. Aunque a veces es necesario utilizar bucles infinitos en un programa, normalmente se deben a errores que se deben corregir.

Los bucles infinitos no intencionados deben evitarse pues significan perder el control del programa. Para interrumpir un bucle infinito, hay que pulsar la combinación de teclas Ctrl+C. Al interrumpir un programa se mostrará un mensaje de error similar a éste:

```
Traceback (most recent call last):
  File "ejemplo.py", line 3, in <module>
    print(i)
KeyboardInterrupt
```

# Actividad Ciclo While

Escriba un programa que pregunte una y otra vez si desea continuar con el programa, siempre que se conteste exactamente **SI**

# Actividad Ciclo While

```
print("DIGA SI PARA CONTINUAR")
respuesta = input("¿Desea continuar el programa?: ")

while respuesta == "SI":
    respuesta = input("¿Desea continuar el programa?: ")

print("¡Hasta la vista!")
```

# Actividad Ciclo While

Escriba un programa que pregunte una y otra vez si desea continuar con el programa, siempre que se conteste **S** o **s** (en mayúsculas o en minúsculas).

# Actividad Ciclo While

```
print("DIGA S O s PARA CONTINUAR")
respuesta = input("¿Desea continuar el programa?: ")

while respuesta == "S" or respuesta == "s":
    respuesta = input("¿Desea continuar el programa?: ")

print("¡Hasta la vista!")
```

# Actividad Ciclo While

Escriba un programa que solicite una contraseña y su confirmación, si las dos contraseñas no coinciden tiene que volverlo a solicitar hasta que coincidan.

# Actividad Ciclo While

```
print("CONFIRME SU CONTRASEÑA")
password_1 = input("Escriba su contraseña: ")
password_2 = input("Escriba de nuevo su contraseña: ")

while password_1 != password_2:
    print("Las contraseñas no coinciden. Inténtelo de nuevo.")
    password_1 = input("Escriba su contraseña: ")
    password_2 = input("Escriba de nuevo su contraseña: ")

print("Contraseña confirmada. ¡Hasta la vista!")
```

## Ciclo for:

Un ciclo **for** es un bucle que repite el bloque de instrucciones un número predeterminado de veces. El bloque de instrucciones que se repite se suele llamar cuerpo del bucle y cada repetición se suele llamar iteración.

La sintaxis de un bucle for es la siguiente:

**for variable in elemento iterable (lista, cadena, range, etc.):**

    cuerpo del bucle

- No es necesario definir la variable de control antes del bucle, aunque se puede utilizar como variable de control una variable ya definida en el programa.
- El cuerpo del bucle se ejecuta tantas veces como elementos tenga el elemento recorrible (elementos de una lista o de un range(), caracteres de una cadena, etc.). Por ejemplo:

```
print("Comienzo")
for i in [0, 1, 2]:
    print("Hola ", end="")
print()
print("Final")
```

```
Comienzo
Hola Hola Hola
Final
```

## Ciclo for:

```
print("Comienzo")
for i in [0, 1, 2]:
    print("Hola ", end="")
print()
print("Final")
```

Se ejecuta la primera instrucción del programa. En este caso, imprime el párrafo de comienzo.

A continuación se ejecuta el bucle. La variable de control toma el primer valor de la lista. La variable de control es **i** y toma el valor 0.

A continuación se ejecutan las instrucciones del bloque. El bloque consta de una sola instrucción que imprime el texto "Hola", la función **end** es para que imprima en la misma línea.

Una vez terminado el bucle, se ejecuta la instrucción que sigue al bucle. Imprime un salto de línea, para que el siguiente **print()** imprima en la línea siguiente.

La última instrucción del programa imprime el párrafo final.

## Ciclo for:

Si la lista está vacía, el ciclo no se ejecuta ninguna vez. Por ejemplo:

```
print("Comienzo")
for i in []:
    print("Hola ", end="")
print()
print("Final")
```

```
Comienzo

Final
```

## Ciclo for:

En el primer ejemplo, los valores que toma la variable no son importantes, lo que importa es que la lista tiene tres elementos y por tanto el bucle se ejecuta tres veces. El siguiente programa produciría el mismo resultado que el anterior:

```
print("Comienzo")
for i in [1, 1, 1]:
    print("Hola ", end="")
print()
print("Final")
```

```
Comienzo
Hola Hola Hola
Final
```

## Ciclo for:

En los ejemplos anteriores, la variable de control "i" no se utilizaba en el bloque de instrucciones, pero en muchos casos sí que se utiliza. Cuando se utiliza, hay que tener en cuenta que la variable de control va tomando los valores del elemento recorrible. Por ejemplo:

```
print("Comienzo")
for i in [3, 4, 5]:
    print(f"Hola. Ahora i vale {i} y su cuadrado {i ** 2}")
print("Final")
```

```
Comienzo
Hola. Ahora i vale 3 y su cuadrado 9
Hola. Ahora i vale 4 y su cuadrado 16
Hola. Ahora i vale 5 y su cuadrado 25
Final
```

## Ciclo for:

La lista puede contener cualquier tipo de elementos, no sólo números. El bucle se repetirá siempre tantas veces como elementos tenga la lista y la variable irá tomando los valores de uno en uno. Por ejemplo:

```
print("Comienzo")
for i in ["Alba", "Benito", 27]:
    print(f"Hola. Ahora i vale {i}")
print("Final")
```

```
Comienzo
Hola. Ahora i vale Alba
Hola. Ahora i vale Benito
Hola. Ahora i vale 27
Final
```

## Ciclo for:

La costumbre más extendida es utilizar la letra i como nombre de la variable de control, pero se puede utilizar cualquier otro nombre válido. Por ejemplo:

```
print("Comienzo")
for numero in  [0, 1, 2, 3]:
    print(f"{numero} * {numero} = {numero * numero}")
print("Final")
```

```
Comienzo
0 * 0 = 0
1 * 1 = 1
2 * 2 = 4
3 * 3 = 9
Final
```

## Ciclo for:

En los ejemplos anteriores se ha utilizado una lista para facilitar la comprensión del funcionamiento de los bucles pero, si es posible hacerlo, se recomienda utilizar tipos **range()**, entre otros motivos porque durante la ejecución del programa ocupan menos memoria en el ordenador.

El siguiente programa es equivalente al programa del ejemplo anterior:

```
print("Comienzo")
for i in range(3):
    print("Hola ", end="")
print()
print("Final")
```

```
Comienzo
Hola Hola Hola
Final
```

## Ciclo for:

Esto permite que el número de iteraciones dependa del desarrollo del programa. En el ejemplo siguiente es el usuario quien decide cuántas veces se ejecuta el bucle:

```
veces = int(input("¿Cuántas veces quiere que le salude? "))
for i in range(veces):
    print("Hola ", end="")
print()
print("Adiós")
```

```
¿Cuántas veces quiere que le salude? 9
Hola Hola Hola Hola Hola Hola Hola Hola Hola
Adiós
```

# Actividad Ciclo for

Escriba un programa que pregunte cuántos números se van a introducir, pida esos números, y muestre un mensaje cada vez que un número no sea mayor que el primero.

# Actividad Ciclo for

```
print("MAYORES QUE EL PRIMERO")
valores = int(input("¿Cuántos valores va a introducir? "))
if valores < 1:
    print("¡Imposible!")
else:
    primero = int(input("Escriba un número: "))
    for i in range(valores - 1):
        numero = int(input(f"Escriba un número más grande que {primero}: "))
        if numero <= primero:
            print(f"¡{numero} no es mayor que {primero}!")
    print("Gracias por su colaboración.")
```

```
MAYORES QUE EL PRIMERO
¿Cuántos valores va a introducir? 5
Escriba un número: 2
Escriba un número más grande que 2: 1
¡1 no es mayor que 2!
```

# Actividad Ciclo for

Escriba un programa que pregunte cuántos números se van a introducir, pida esos números, y muestre un mensaje cada vez que un número no sea mayor que el anterior.

# Actividad Ciclo for

```python
print("MAYORES QUE EL ANTERIOR")
valores = int(input("¿Cuántos valores va a introducir? "))
if valores < 1:
    print("¡Imposible!")
else:
    anterior = int(input("Escriba un número: "))
    for i in range(valores - 1):
        numero = int(input(f"Escriba un número más grande que {anterior}: "))
        if numero <= anterior:
            print(f"¡{numero} no es mayor que {anterior}!")
        anterior = numero
    print("Gracias por su colaboración.")
```

```
MAYORES QUE EL ANTERIOR
¿Cuántos valores va a introducir? 6
Escriba un número: 5
Escriba un número más grande que 5: 1
¡1 no es mayor que 5!
```