

## MODULO 2

### ESTRUCTURAS REPETITIVAS

#### LOGRO DE LA SESION

Al concluir la sesión, los participantes serán capaces de comprender cómo funcionan las estructuras repetitivas y aplicarlas eficazmente en la resolución de requerimientos que involucren tareas repetitivas.

# Contenido



- **Introducción**
- **Conceptos previos**
- **Bucle: while**
- **Bucle: for**
- **Control de bucles**
- **Números aleatorios**
- **Ejercicios**
- **Conclusiones**
- **Evaluación Continua 3**

# Introducción



Un ciclo es una estructura que facilita la representación de un conjunto de instrucciones que se repiten un número finito de veces, típicamente determinado por una condición o un número específico de repeticiones o iteraciones.

Los ciclos permiten ejecutar un proceso varias veces según lo determine el programador o el usuario.



# Conceptos previos: contador

$$\text{CONTADOR} = \text{CONTADOR} + \text{CONSTANTE}$$

Es una variable que se utiliza para realizar un seguimiento del número de veces que ocurre un evento específico dentro de un bucle o en cualquier otra situación donde se necesite contar repeticiones o iteraciones.

El propósito principal de un contador es mantener un registro de cuántas veces se ha ejecutado un bloque de código, cuántas veces se ha cumplido una condición o cuántas iteraciones ha realizado un bucle.



# Conceptos previos: acumulador

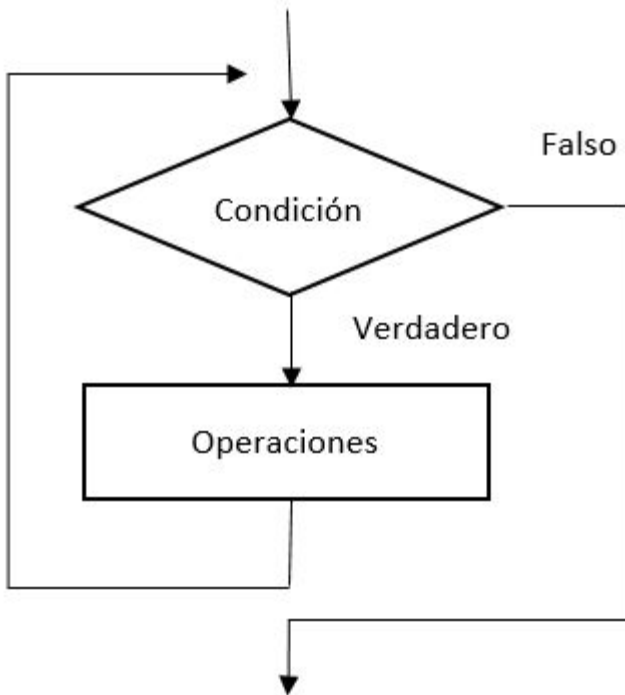
$$\text{ACUMULADOR} = \text{ACUMULADOR} + \text{NUEVO\_VALOR}$$

Un acumulador es una variable que se utiliza para **almacenar** y **acumular** valores a medida que se recorren datos o se realizan cálculos en un programa.

La función principal de un acumulador es mantener la suma o acumulación de valores a lo largo de la ejecución del programa.

El acumulador comienza típicamente con un valor inicial y se actualiza en cada iteración del ciclo, cada vez que se procesa un dato o cada vez que se realiza una operación que requiere agregar un nuevo valor al acumulado.

# Bucle: **while**



## Sintaxis:

```
while condición:  
    bloque de instrucciones
```

El bloque de instrucciones se ejecuta mientras la condición es verdadera.



# Bucle: **while**

## Ejemplo 1:

Desarrollar un programa que permita calcular la suma de los **N** primeros números.

**# Datos**

N = 10

**# Proceso**

suma = 0

cont = 1

while (cont <= N):

    suma += cont

    cont = cont + 1

**# Reporte**

print(f"La suma es {suma}")



# Bucle: **while**

## Ejemplo 2:

Desarrollar un programa que muestre los números impares entre **m** y **n**.

## Reto

Hacer que el programa funcione de la misma manera independientemente si **m** es el número mayor o menor.

```
# Datos
m = 5
n = 10
# Proceso
i = m
reporte = ""
while(i <= n):
    if (i%2 == 1):
        reporte += str(i) + " "
    i = i + 1
# Reporte
print("Reporte:", reporte)
```





# Bucle: for

## Sintaxis:

```
for elemento in iterable:  
    bloque de instrucciones
```

**elemento:** Es una variable que tomará el valor de cada elemento en el iterable en cada iteración del bucle.

**iterable:** Es una secuencia de elementos sobre la que se iterará, como una lista, una tupla, un diccionario, un conjunto, o cualquier otro objeto iterable en Python.



# Bucle: for

## Ejemplo 3:

Desarrollar un programa para encontrar el valor de la siguiente sumatoria:

$$S = \sum_{i=0}^n (2i + 1)$$

# Datos

n = 1

# Proceso

s = 0

for i in range(n+1):

    s += (2\*i + 1)

# Reporte

print(f"La suma es {s}")



# Bucle: for

## Ejemplo 4:

Desarrollar un programa para encontrar el factorial de un numero.

# Datos

n = 4

# Proceso

f = 1

for i in range(1,n+1):

f \*= i

# Reporte

print("Factorial:",f)



# Bucle: for

## Ejemplo 5:

Desarrollar un programa para encontrar la suma de los números pares entre 1 y n.

# Datos

n = 10

# Proceso

s = 0

for i in range(2,n+1,2):

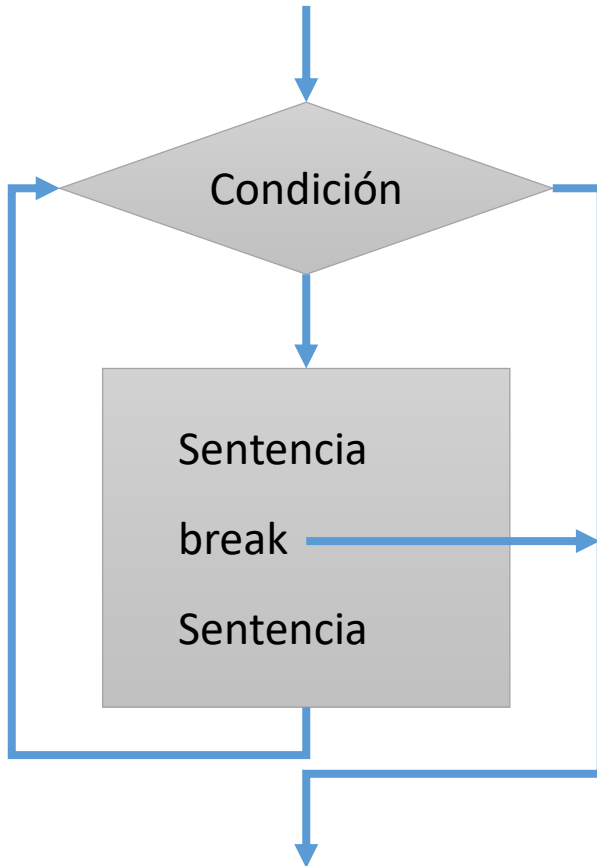
    s += i

# Reporte

print("Suma:",s)



# Control de bucles: **break**



- La instrucción **break** se utiliza para salir de un bucle de manera prematura.
- Cuando se encuentra la instrucción **break** dentro de un bucle (como un bucle **for** o **while**), el bucle se detiene inmediatamente y la ejecución del programa continúa con la siguiente instrucción después del bucle.
- Es útil cuando se desea terminar un bucle antes de que se complete su ciclo normal.



# Control de bucles: **break**

## Ejemplo 6:

Desarrollar un programa para encontrar la suma de los primeros 3 números múltiplos de 3 que se encuentren entre m y n inclusive.

Los valores de m y n los ingresa el usuario.

El valor de m debe ser menor que el valor de n.

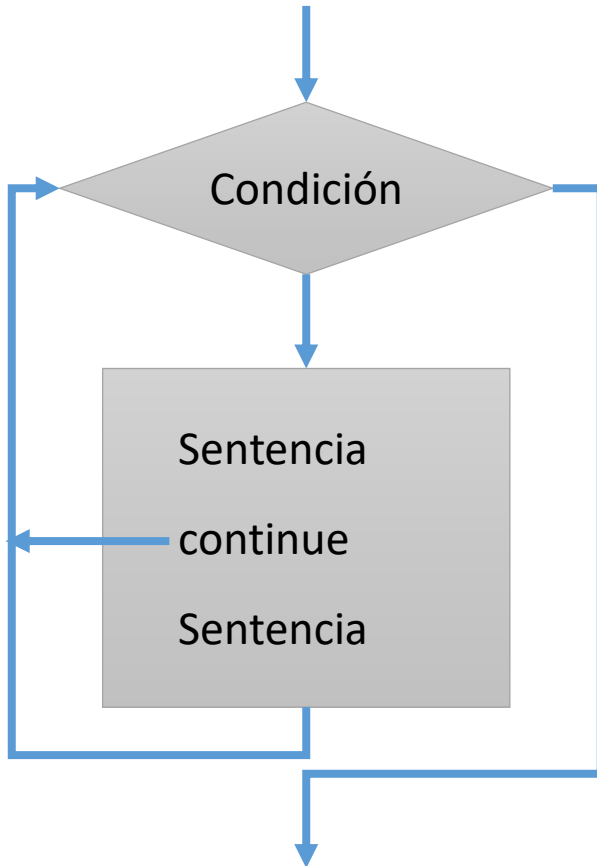
```
# Datos
m = int(input("Valor de m: "))
n = int(input("Valor de n: "))

# Proceso
s = 0
cont = 0
for i in range(m,n+1):
    if i%3 == 0:
        s += i
        cont += 1
    if cont == 3: break

# Reporte
print("Suma:",s)
```



# Control de bucles: **continue**



- La instrucción **continue** se utiliza para omitir el resto del código dentro de un bucle en una iteración específica y pasar a la siguiente iteración.
- Cuando se encuentra la instrucción **continue**, el resto del código dentro del bucle no se ejecuta para esa iteración en particular, pero el bucle continúa con la siguiente iteración.
- Es útil cuando se desea saltar ciertas iteraciones en un bucle sin salir completamente de él.



# Control de bucles: **continue**

## Ejemplo 7:

Desarrollar un programa para encontrar la suma de los números múltiplos de 5 que se encuentren entre m y n inclusive.

Los valores de m y n los ingresa el usuario.

El valor de m debe ser menor que el valor de n.

### # Datos

```
m = int(input("Valor de m: "))
```

```
n = int(input("Valor de n: "))
```

### # Proceso

```
s = 0
```

```
cont = 0
```

```
for i in range(m,n+1):
```

```
    if i%5 != 0: continue
```

```
    s += i
```

### # Reporte

```
print("Suma:",s)
```





# Números aleatorios enteros



## Paso 1: Importar librería

```
import random
```

## Paso 2: Generar números

```
random.randint(valor_inicial, valor_final)
```



# Números aleatorios enteros

## Ejemplo 8:

Desarrollar un programa para encontrar el promedio de 5 notas generados de manera aleatoria.

Las notas deben estar en el rango de 0 a 20 inclusive.

```
import random
# Datos
n = 5
# Proceso
notas = ""
suma = 0
for i in range(1,n+1):
    nota = random.randint(1,20)
    suma += nota
    notas += f"{nota} "
pr = suma / n
# Reporte
print(f"Notas: {notas}")
print(f"Suma: {suma}")
print(f"Promedio: {pr}")
```

# Ejercicios

Debes aplicar los conceptos que se bien utilizando: Entrada de datos  $\Rightarrow$  Proceso  $\Rightarrow$  Reporte



# Ejercicio 1



Elaborar un programa que permita calcular la suma de los cuadrados de los  $N$  primeros números naturales.

# Ejercicio 2



Elaborar un programa que permita encontrar los divisores de un numero entero positivo mayor que 10.

# Ejercicio 3



Desarrollar un programa para encontrar el valor de la siguiente sumatoria:

$$S = \sum_{i=3}^n \frac{k - 1}{k(k + 1)}$$

# Ejercicio 4

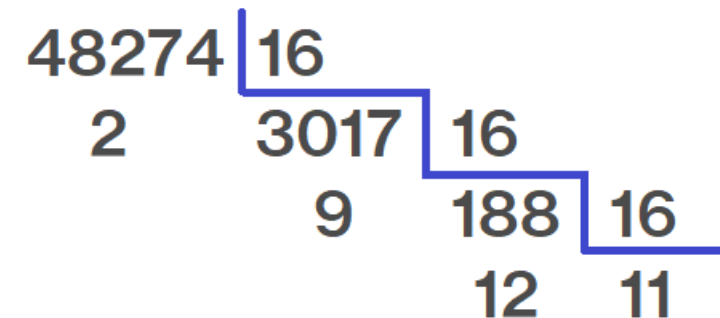


## DE BASE 10 A BASE 16

■ Ejemplo:  $48274_{10} \rightarrow$  base 16

■ Resultado:  $11 - 12 - 9 - 2 = \text{BC92}_{16}$

Desarrollar un programa para convertir un número de base 10 a base 16.



# Ejercicio 5



Un número es primo cuando es divisible por 1 y por si mismo.

Desarrollar un programa que permita averiguar si un número es primo.





# Ejercicio 6



**Desarrollar un programa que genere N números enteros entre 10 y 99.**

**Luego el programa debe determinar cuantos y cuales son números múltiplos de 3 y 5 respectivamente.**

# Conclusiones

- En esta sesión has aprendido los aspectos más importantes de los bucle while y for.
- Estas estructuras de control es claves para trabajar algoritmos mas complejos en la solución de problemas computacionales.
- Por último, se han desarrollado diversos tipos de problemas que te permiten tener una base solida en estos tipos de estructuras.

**GRACIAS  
TOTALES**