



# Python: Estructuras de Control

Centro de Servicios y Gestión Empresarial  
SENA Regional Antioquia

# Conceptualización

# Estructuras de Control

Las estructuras de control permiten modificar el flujo de ejecución de un programa.

En Python existen tres tipos principales:

- **Condicionales** (if, elif, else)
  - **Bucles** (while, for)
  - **Control de flujo** (break, continue, pass)
- 
- **Sentencias de casos** (match-case).



# Condicionales

# Condicionales

Los **Condicionales** son (*if, elif, else*) y permiten ejecutar un bloque de código si se cumple una condición es True o False.

## Sintaxis

```
if condicion:  
    # Código si la condición es verdadera  
else:  
    # Código si la condición es falsa
```

# Condicionales

## Ejemplo:

Determinar si un usuario es mayor de edad o menor de edad.

## Solución:

```
edad = int(input("Ingrese su edad: "))

if edad >= 18:
    print("Eres mayor de edad.")
else:
    print("Eres menor de edad.")
```

# Condicionales

Si una condición **if** es **False**, se evalúan las condiciones **elif**.  
Si ninguna es **True**, se ejecuta el **else**.

## Sintaxis

```
if condicion1:  
    # Código si condicion1 es verdadera  
elif condicion2:  
    # Código si condicion2 es verdadera  
elif condicion3:  
    # Código si condicion3 es verdadera  
else:  
    # Código si ninguna de las condiciones anteriores es verdadera
```

# Condicionales

## Ejemplo:

Solicitar al usuario ingresar una nota numérica (entero) y, con base en el valor ingresado, muestre un mensaje indicando su desempeño según la siguiente clasificación:

- **Excelente:** si la nota es 90 o superior.
- **Muy bien:** si la nota está entre 80 y 89.
- **Bien:** si la nota está entre 70 y 79.
- **Necesitas mejorar:** si la nota es menor a 70.

# Condicionales

## Solución:

```
nota = int(input("Ingrese su nota: "))

if nota >= 90:
    print("Excelente")
elif nota >= 80:
    print("Muy bien")
elif nota >= 70:
    print("Bien")
else:
    print("Necesitas mejorar")
```

# Operadores Lógicos

# Operadores Comparación

Se usan para comparar valores y devuelven True o False.

Operador	Descripción	Ejemplo
<code>==</code>	Igual a	$5 == 5 \rightarrow \text{True}$
<code>!=</code>	Diferente de	$5 != 3 \rightarrow \text{True}$
<code>&gt;</code>	Mayor que	$10 > 5 \rightarrow \text{True}$
<code>&lt;</code>	Menor que	$5 < 10 \rightarrow \text{True}$
<code>&gt;=</code>	Mayor o igual que	$5 >= 5 \rightarrow \text{True}$
<code>&lt;=</code>	Menor o igual que	$4 <= 3 \rightarrow \text{False}$

# Operadores Lógicos

Se usan para combinar expresiones lógicas.

Operador	Descripción	Ejemplo
<b>and</b>	True si ambas condiciones son True	$(5 > 3) \text{ and } (10 > 5) \rightarrow \text{True}$
<b>or</b>	True si al menos una condición es True	$(5 > 3) \text{ or } (10 < 5) \rightarrow \text{True}$
<b>not</b>	Niega la condición	$\text{not}(5 > 3) \rightarrow \text{False}$

# Ejemplo

## Ejemplo – Comparación

```
a = 10
b = 5
print(a == b) # False
print(a > b) # True
```

## Ejemplo – Lógicos

```
edad = 25
es_mayor = edad > 18 and edad < 30
print(es_mayor) # True
```

# Ejemplo



## Ejemplo – Lógicos

```
llueve = False

if not llueve:
    print("Puedes salir sin paraguas.")
else:
    print("Lleva un paraguas.")
```

# Ejemplo Integrador



## Ejemplo:

Determinar si un usuario cumple con los requisitos para solicitar productos bancarios.

## Solución:

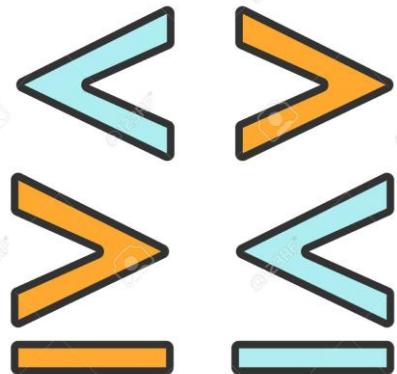
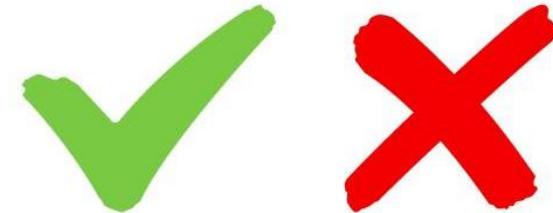
```
edad = 25
ingresos = 3000
genero = "M"

if edad >= 18 and ingresos >= 2500 and genero == "F" or genero == "M":
    print("Puedes solicitar una tarjeta de crédito.")
else:
    print("No cumples los requisitos.")
```

# Ejercicios de Aplicación

# Ejercicios de Aplicación

- Realizar una aplicación que permita determinar si un numero que digite el usuario es positivo o negativo.



- Realizar una aplicación que permita determinar el numero mayor a partir de tres números enteros que ingresa el usuario por teclado.

# Ejercicios de Aplicación

- Realizar una aplicación que permita determinar el tipo de Chile utilizado de acuerdo con las unidades de Scoville que ingrese el usuario por teclado.



## Unidades de Scoville

Habanero: 200 mil a 445 mil

Chiltepín: 100 mil a 200 mil

Piquín: 30 mil a 100 mil

Árbol: 23 mil a 30 mil

Serrano: 5 mil a 23 mil

Jalapeño: 1mil a 5 mil

Bell: 0 mil a 1 mil

# Condicional en una sola línea

Si el bloque de código tiene solo una instrucción, se puede escribir en una línea.

## Sintaxis:

```
valor_si_verdadero if condicion else valor_si_falso
```

## Ejemplo:

```
edad = 25
if edad >= 18:
    print("Eres mayor de edad.")
else:
    print("Eres menor de edad.")
```

```
print("Eres mayor de edad.") if edad >= 18 else print("Eres menor de edad.")
```



# Bucles / Ciclos

# Bucles / Ciclos

Se usan para repetir bloques de código varias veces.

for

Recorre una secuencia de elementos

while

Repite mientras una condición sea verdadera



# Ciclo For

El bucle for se usa para recorrer elementos en una secuencia (listas, tuplas, cadenas, rangos, etc.).

## Sintaxis

```
for variables in range(inicio, fin, incremento):  
    # Código que se ejecuta en cada iteración
```

# Ciclo For

## Ejemplos:

```
# Range con Valor Final  
for valor in range(10):  
    print(valor)
```

```
# Range con Valor Inicial y  
Final  
for valor in range(1, 10):  
    print(valor)
```

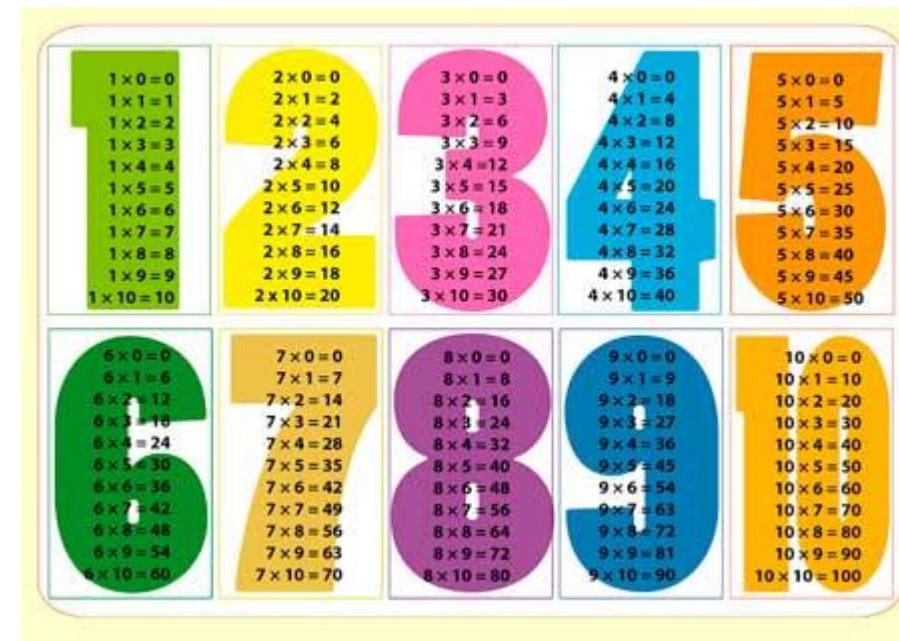
```
# Range con Valor Inicial,  
Final e Incremento  
for valor in range(1, 10, 3):  
    print(valor)
```

```
# Range con Valor Inicial,  
Final e Incremento Negativo /  
Decremento  
for valor in range(10, 1, -1):  
    print(valor)
```

# Ciclo For

## Ejercicio:

A partir de un numero que digite el usuario, mostrar la tabla de multiplicar del numero.



# Ciclo For

## Ejercicio:

A partir de un numero que digite el usuario, mostrar la tabla de multiplicar del numero.

## Solución:

```
numero = int(input("Ingrese un número: "))

for i in range(1, 11):
    print(f"{numero} x {i} = {numero * i}")
```

# Ciclo For (*Secuencia*)

## Sintaxis:

```
for variable in secuencia:  
    # Código que se ejecuta en cada iteración
```

## Ejemplo:

```
palabra = "Python"  
for letra in palabra:  
    print(letra)
```

```
numeros = 123456789  
for numero in str(numeros):  
    print(numero)
```

# Ciclo While

El bucle while se ejecuta mientras una condición sea True.

## Sintaxis:

```
condicion = valor_inicio  
  
while condicion:  
    # Código que se ejecuta mientras la condición sea verdadera  
    condicion = valor_actualizacion
```

# Operadores Asignación

Se usan para asignar valores a variables.

Operador	Equivalente a	Ejemplo
=	$a = b$	$a = 5$
$+=$	$a = a + b$	$a += 3$
$-=$	$a = a - b$	$a -= 2$
$*=$	$a = a * b$	$a *= 4$
$/=$	$a = a / b$	$a /= 3$
$//=$	$a = a // b$	$a // 2$
$%=$	$a = a \% b$	$a \% 3$
$**=$	$a = a ** b$	$a **= 2$

# Ciclo While

El bucle while se ejecuta mientras una condición sea True.

## Ejemplo:

```
condicion = 5

while condicion <= 20:
    print(condicion)
    condicion += 5
```

```
contador = 20

while contador >= 0:
    print(contador)
    contador -= 2
```

# Contador / Acumulador

Los acumuladores y contadores son variables utilizadas en los bucles para almacenar valores o contar iteraciones.

- Un **contador** es una variable que se incrementa en una cantidad fija (generalmente +1) en cada iteración de un bucle.
- Un **acumulador** es una variable que se incrementa en valores variables (por ejemplo, sumando los números ingresados por el usuario).

Concepto	Función	Ejemplo
Contador	Cuenta la cantidad de iteraciones.	contador += 1
Acumulador	Suma valores durante el bucle.	acumulador += valor

# Contador / Acumulador

## Ejemplo:

```
contador = 1 # Para contar las iteraciones
acumulador = 0 # Para almacenar la suma

while contador <= 5:
    num = int(input(f"{contador}. Ingrese el número: "))
    acumulador += num # Suma los valores ingresados
    contador += 1 # Incrementa el contador

print(f"La suma total de los {contador-1} números ingresados es:
{acumulador}")
```

# Ejercicio While

Trabajas en una empresa y debes calcular el sueldo semanal de los empleados basado en las horas trabajadas. Un sistema solicita la cantidad de horas trabajadas por cada empleado hasta que se introduzca un número negativo para finalizar el proceso.



# Contador / Acumulador

## Ejercicio:

Desarrolla un programa en Python que simule el funcionamiento de un cajero automático.

El usuario tendrá un saldo inicial de \$1000 y podrá realizar retiros de dinero hasta que su saldo llegue a \$0 o decida salir ingresando 0.



# Contador / Acumulador

## Solución:

```
saldo = 10
cantidad = -1
while cantidad != 0 and saldo > 0:
    cantidad = float(input("Ingrese la cantidad a retirar (o 0 para salir): "))
    if cantidad > saldo:
        print("No tienes suficiente saldo.")
    elif cantidad < 0:
        print("No puedes retirar cantidades negativas.")
    else:
        saldo -= cantidad
    print(f"Tu saldo actual es de ${saldo}")
```

# Control de Flujo

# Declaraciones

Python proporciona tres instrucciones para modificar el flujo de ejecución dentro de los bucles (for y while):

## Break

Permite detener el ciclo incluso si la condición es verdadera

## Continue

Permite detener la iteración actual y continuar con la siguiente

## Pass

No hace nada; se usa como marcador de posición.

# Declaración Break



**WHILE** ←

```
# Break en una estructura While
i = 1
while i < 6:
    #print (i)
    if (i == 4):
        break
    i += 1
print ("Finalizo en la Iteracion: ",i)
```

```
# Break en una estructura For
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "banana":
        break
    print(x)
```

→ **FOR**

# Declaración Continue

**WHILE** ←

```
# Break en una estructura While
i = 0
while i < 6:
    i += 1
    if i == 3:
        continue
    print(i)
```

```
# Break en una estructura While
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "banana":
        continue
    print(x)
```

→ **FOR**

# Ejercicio Break, continue, pass

Se requiere una aplicación para validar la contraseña y permitir el acceso al sistema:

- El usuario tiene 3 intentos para ingresarla correctamente. Si se agotan los intentos, se bloquea el acceso.
- Si la contraseña es incorrecta, se le pide que la ingrese nuevamente.
- Si la contraseña es correcta, se otorga acceso y se finaliza el programa.



# Estructura de Casos Switch

# Match - case

El **match - case** es una estructura de control introducida en Python 3.10 que permite manejar múltiples condiciones de manera más clara y eficiente, similar a switch en otros lenguajes como Java o C.

## Sintaxis

```
match variable:  
    case valor1:  
        # Código a ejecutar si variable == valor1  
    case valor2:  
        # Código a ejecutar si variable == valor2  
    case _:  
        # Código a ejecutar si no coincide con ningún caso (default)
```

# Match - case

## Menú de opciones

```
opcion = int(input("Ingrese una opción (1-3): "))

match opcion:
    case 1:
        print("Has seleccionado la opción 1.")
    case 2:
        print("Has seleccionado la opción 2.")
    case 3:
        print("Has seleccionado la opción 3.")
    case _:
        print("Opción no válida.")
```

# Match - case

## Ejercicio:

💡 El usuario seleccionará una opción de conversión de moneda:

- Dólares a Euros
- Dólares a Pesos Colombianos
- Dólares a Yen Japonés

El programa pedirá la cantidad en dólares y mostrará el monto convertido.





# G R A C I A S

Presentó: Alvaro Pérez Niño

Instructor Técnico

Correo: aperezn@sena.edu.co

<http://centrodesserviciosygestionempresarial.blogspot.com/>

Línea de atención al ciudadano: 01 8000 910270

Línea de atención al empresario: 01 8000 910682



[www.sena.edu.co](http://www.sena.edu.co)