**¿Qué son las estructuras condicionales en Python?**

Las **estructuras condicionales** en todos los lenguajes de programación, nos permite realizar distintas acciones según una condición. En **python**, tiene una pequeña variante que cuando uno empieza en **python** suelen equivocarse y en la “indentación”.

**La indentación** es una característica que diferencia Python de otros lenguajes de programación, dónde el código a ejecutar de cumplirse la condición se encierra entre llaves. Esta característica tiene el propósito de mejorar la legibilidad de los programas.

## Sentencia if

La sentencia condicional más básica en Python es la sentencia if, la cual se expresa de la siguiente forma:

**if** condicion:

# ejecutar un código

En la expresión previa:

* La condición es una expresión booleana que se evalúa como verdadera (True) o falsa (False).
* Se requiere el uso de dos puntos (:) al final de la condición.
* Todas las líneas de código a ejecutar si se cumple la condición tienen que estar indentadas respecto la sentencia if.

Veamos el uso de la sentencia if con un ejemplo:

x = 15

if x > 10:

    print('x es mayor que 10')

**Sentencia *else***

A la sentencia if se le puede añadir opcionalmente una sentencia else. Esta sentencia contiene el código a ejecutar en caso de que no se cumpla la condición de la sentencia if. Esta estructura se expresa del siguiente modo:

**if** condicion:

# ejecutar un código

**else**:

# ejecutar un código distinto

Veamos un ejemplo parecido al anterior al que se le ha añadido una sentencia else:

if x > 10:

    print('x es mayor que 10')

else:

    print('x es menor o igual que 10')

**Sentencia *elif***

A una sentencia if else se pueden añadir un número indefinido de condiciones adicionales a verificar. Estas condiciones se definen mediante la sentencia elif , la cual es un abreviación de else if. Ésta se define así:

**if** condicion:

# ejecutar un código

**elif** otra\_condicion:

# ejecutar otro código

**else**:

# ejecutar un código distinto

Siguiendo con el ejemplo de los números, podemos añadir una sentencia elif del siguiente modo:

x = 7

if x > 10:

    print('x es mayor que 10')

elif x < 10:

    print('x es menor que 10')

else:

    print('x es 10')

...

x es menor que 10

En este caso como x tiene asignado el valor 7, la condición de la sentencia elif se evalúa como verdadera. Por este motivo el programa ejecuta la instrucción print() asociada a esta sentencia.

Un punto importante de las sentencias if elif es que una cuando una condición es evaluada como verdadera se ignoran el resto de condiciones. Este hecho se ilustra en el siguiente ejemplo:

x = 15

if x > 5:

    print('x es mayor que 5')

elif x > 10:

    print('x es mayor que 10')

else:

    print('x es menor que 5')

...

x es mayor que 5

En el ejemplo nunca va a evaluarse la condición de la sentencia elif, ya que los valores mayores que 10 también cumplen la condición de la sentencia if (ser mayores que 5). Por consiguiente la instrucciónprint asociada al elif es una línea de código inaccesible.

EJERCICIO

1.- Escribir un programa que pregunte al usuario su edad y muestre por pantalla si es mayor de edad o no.

EDAD = int(input("¿Cuál es tu edad? "))

if EDAD < 18:

    print ("Eres menor de edad.")

else:

    print("Eres mayor de edad.")

2.- Escribir un programa que almacene la cadena de caracteres `contraseña` en una variable, pregunte al usuario por la contraseña e imprima por pantalla si la contraseña introducida por el usuario coincide con la guardada en la variable sin tener en cuenta mayúsculas y minúsculas.

key = "contraseña"

password = input("Introduce la contraseña: ")

if key == "contraseña":

    print("La contaseña coincide")

else:

    print("La contraseña no coincide")

3.- Crear un programa que al ingresar tres notas entregue el promedio, y si el promedio es mayor o igual a 4 de un mensaje alumnos a aprobado

print ("Escriba la nota de las 5 clases")

nota1 = float(input("Nota1....: "))

nota2 = float(input("Nota2....: "))

nota3 = float(input("Nota3....: "))

promedio=(nota1+nota2+nota3)/3

print ("El total de sus 3 calificaciones es: ",promedio)

if(promedio>=3.95):

    print("Su promedio es..; " ,promedio, "Felicitaciones, Aprobo la Asignatura")

else:

    print("Su promedio es..; " ,promedio, "Lamento informar que reprobo la Asignatura")

4.- Escribir un programa que pida al usuario un número entero y muestre por pantalla si es par o impar.

n = int(input("Introduce un número entero: "))

if n % 2 == 0:

    print("El número " + str(n) + " es par")

else:

    print("El número " + str(n) + " es impar")

5.- Escribir un programa que pida al usuario dos números y devuelva su división. Si el usuario no introduce números debe devolver un aviso de error y si el divisor es cero también.

n = float(input("Introduce el dividendo: "))

m = float(input("Introduce el divisior: "))

if m == 0:

    print("¡Error! No se puede dividir por 0.")

else:

    print(n/m)

6.- Escribir un programa para una empresa que tiene salas de juegos para todas las edades y quiere calcular de forma automática el precio que debe cobrar a sus clientes por entrar. El programa debe preguntar al usuario la edad del cliente y mostrar el precio de la entrada. Si el cliente es menor de 4 años puede entrar gratis, si tiene entre 4 y 18 años debe pagar $ 4.000 y si es mayor de 18 años, $10.000

edad = int(input("Introduce tu edad: "))

# Decisión del precio en función de la edad

if edad < 4:

    precio = 0

elif edad <= 18:

    precio = 4000

else:

    precio = 10000

# Mostrar precio

print("El precio de la entrada es...$ ", precio)

**¿Cuáles son los ciclos en Python?**

Un **ciclo en Python** o bucle en **Python** (como prefieras llamarlos) te permite repetir una o varias instrucciones cuantas veces lo necesitemos, por ejemplo, si quisiéramos escribir los números del uno al cien no tendría sentido escribir cien líneas de código mostrando un número en cada una, para eso y para varias cosas más ..

En Python se pueden utilizar dos tipos de ciclos: for y while. Recordemos que los ciclos se utilizan cuando es necesario ejecutar un bloque de instrucciones varias veces, hasta que, o mientras que, se cumpla una condición dada.

# For

Los ciclos for permiten ejecutar una o varias instrucciones de forma iterativa, una vez por cada elemento en la colección.

Las colecciones pueden ser de varios tipos, el for puede recibir una colección predefinida o directamente de la salida de una función.

Ejemplo

1.-Escribir un programa que pida al usuario una palabra y la muestre 10 veces por pantalla.

Palabra = input("Introduce una palabra: ")

for i in range(10):

    print(Palabra)

2.-El siguiente fragmento de código es utilizado para iterar a través de un rango del 1 al 10.

for contador in range(1,10):

      print (contador)

3.- Escribir un programa que pregunte al usuario su edad y muestre por pantalla todos los años que ha cumplido (desde 1 hasta su edad).

cumplido = int(input("¿Cuántos años tienes? "))

for i in range(cumplido):

    print("Has cumplido " + str(i+1) + " años")

4.- Para repetir o iterar sobre una lista.

numeros = [0, 1, 2, 3, 4, 5]

for numero in numeros:

    print (numero)

5.- Si se quiere iterar sobre un diccionario.

frutas = {'Fresa':'roja', 'Limon':'verde', 'Papaya':'naranja', 'Manzana':'amarilla', 'Guayaba':'rosa'}

for nombre, color in frutas.items():

      print (nombre, "es de color", color)

Anteriormente hemos visto el uso del [if](https://ellibrodepython.com/if-python) y el [for](https://ellibrodepython.com/for-python) para modificar el flujo de ejecución del código. A continuación vemos otra forma de hacerlo con el while.

**While**

El uso del while nos permite **ejecutar una sección de código repetidas veces**, de ahí su nombre. El código se ejecutará **mientras** una condición determinada se cumpla. Cuando se deje de cumplir, se saldrá del bucle y se continuará la ejecución normal. Llamaremos **iteración** a una ejecución completa del bloque de código.

Cabe destacar que existe dos tipos de bucles, los que tienen un número de iteraciones **no definidas**, y los que tienen un número de iteraciones **definidas**. El while estaría dentro del primer tipo. Mas adelante veremos los for, que se engloban en el segundo.

x = 5

while x > 0:

x -=1

print(x)

# Salida: 4,3,2,1,0

1.- Escribir un programa que pida al usuario una palabra y la muestre 10 veces por pantalla.

word = input("Introduce una palabra: ")

for i in range(10):

    print(word)