Actividades iniciadas el 01sept2021

**Python If ... Else**

**Condiciones de Python y declaraciones If**

Python admite las condiciones lógicas habituales de las matemáticas:

* Es igual a: a == b
* No es igual a: a! = b
* Menor que: a <b
* Menor o igual a: a <= b
* Mayor que: a> b
* Mayor o igual a: a> = b

Estas condiciones se pueden utilizar de varias formas, más comúnmente en "sentencias if" y bucles.

Una "instrucción if" se escribe utilizando la palabra clave if.

Example

If statement:

a = 33  
b = 200  
if b > a:  
  print("b is greater than a")

En este ejemplo usamos dos variables, a y b, que se usan como parte de la instrucción if para probar si b es mayor que a. Como a es 33 y b es 200, sabemos que 200 es mayor que 33, por lo que imprimimos en la pantalla que "b es mayor que a".

**Sangría**

Python se basa en la sangría (espacio en blanco al comienzo de una línea) para definir el alcance en el código. Otros lenguajes de programación a menudo usan corchetes para este propósito.

### Example

If statement, without indentation (will raise an error):

a = 33  
b = 200  
if b > a:  
print("b is greater than a") # you will get an error

**Elif**

La palabra clave elif es la forma de python de decir "si las condiciones anteriores no eran verdaderas, entonces pruebe esta condición".

### Example

a = 33  
b = 33  
if b > a:  
  print("b is greater than a")  
elif a == b:  
  print("a and b are equal")

En este ejemplo, a es igual a b, por lo que la primera condición no es verdadera, pero la condición elif es verdadera, por lo que imprimimos en la pantalla que "a y b son iguales".

**else**

La palabra clave else captura cualquier cosa que no sea detectada por las condiciones anteriores.

### Example

a = 200  
b = 33  
if b > a:  
  print("b is greater than a")  
elif a == b:  
  print("a and b are equal")  
else:  
  print("a is greater than b")

En este ejemplo, a es mayor que b, por lo que la primera condición no es verdadera, también la condición elif no es verdadera, así que vamos a la condición else e imprimimos en la pantalla que "a es mayor que b".

También puede tener un else sin elif:

### Example

a = 200  
b = 33  
if b > a:  
  print("b is greater than a")  
else:  
  print("b is not greater than a")

## **Short Hand If**

Si solo tiene una instrucción para ejecutar, puede ponerla en la misma línea que la instrucción if.

### Example

One line if statement:

if a > b: print("a is greater than b")

## **Short Hand If ... Else**

Si solo tiene una declaración para ejecutar, una para if y otra para else, puede ponerlas todas en la misma línea:

### Example

One line if else statement:

a = 2  
b = 330  
print("A") if a > b else print("B")

Esta técnica se conoce como **Operadores ternarios** o **Expresiones condicionales**.

También puede tener varias declaraciones else en la misma línea:

### Example

One line if else statement, with 3 conditions:

a = 330  
b = 330  
print("A") if a > b else print("=") if a == b else print("B")

**And**

La palabra clave and es un operador lógico y se utiliza para combinar declaraciones condicionales:

### Example

Test if a is greater than b, AND if c is greater than a:

a = 200  
b = 33  
c = 500  
if a > b and c > a:  
  print("Both conditions are True")

**Or**

La palabra clave or es un operador lógico y se utiliza para combinar declaraciones condicionales:

### Example

Test if a is greater than b, OR if a is greater than c:

a = 200  
b = 33  
c = 500  
if a > b or a > c:  
  print("At least one of the conditions is True")

**If Anidado**

Puede tener sentencias if dentro de sentencias if, esto se llama sentencias if anidadas.

### Example

x = 41  
  
if x > 10:  
  print("Above ten,")  
  if x > 20:  
    print("and also above 20!")  
  else:  
    print("but not above 20.")

**La declaración pass**

Las declaraciones if no pueden estar vacías, pero si por alguna razón tiene una declaración if sin contenido, coloque la declaración pass para evitar errores.

Example

a = 33  
b = 200  
  
if b > a:  
  pass

# having an empty if statement like this, would raise an error without the pass statement

**Bucles while de Python**

**Bucles de Python**

Python tiene dos comandos de bucle primitivos:

* Bucle while
* Bucle for

**El bucle while**

Con el ciclo while podemos ejecutar un conjunto de declaraciones siempre que una condición sea verdadera.

### Example

Imprima i siempre que i sea menor que 6:

i = 1  
while i < 6:  
  print(i)  
  i += 1

**Nota**: recuerde incrementar i, de lo contrario el ciclo continuará para siempre.

El ciclo while requiere que las variables relevantes estén listas, en este ejemplo necesitamos definir una variable de indexación, i, que establecemos en 1.

**La declaración break**

Con la declaración break podemos detener el ciclo incluso si la condición while es verdadera:

Example

Salir del ciclo cuando i es 3:

i = 1  
while i < 6:  
  print(i)  
  if i == 3:  
    break  
  i += 1

**La declaración continue**

Con la declaración continue podemos detener la iteración actual y continuar con la siguiente:

### Example

Continúe con la siguiente iteración si i es 3:

i = 0  
while i < 6:  
  i += 1  
  if i == 3:  
    continue  
  print(i)

**La declaración else**

Con la declaración else podemos ejecutar un bloque de código una vez cuando la condición ya no es verdadera:

Example

Imprime un mensaje una vez que la condición sea falsa:

i = 1  
while i < 6:  
  print(i)  
  i += 1  
else:  
  print("i is no longer less than 6")

**Bucle For de Python**

Un bucle for se usa para iterar sobre una secuencia (que es una lista, una tupla, un diccionario, un conjunto o una cadena).

Esto es menos parecido a la palabra clave for en otros lenguajes de programación y funciona más como un método iterador como se encuentra en otros lenguajes de programación orientados a objetos.

Con el bucle for podemos ejecutar un conjunto de declaraciones, una vez para cada elemento de una lista, tupla, conjunto, etc.

Example

Print each fruit in a fruit list:

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
  print(x)

El bucle for no requiere que se establezca una variable de indexación de antemano.

**Bucle a través de una cadena**

Incluso las cadenas son objetos iterables, contienen una secuencia de caracteres:

### Example

Loop through the letters in the word "banana":

for x in "banana":  
  print(x)

**La declaración break**

Con la instrucción break podemos detener el ciclo antes de que haya pasado por todos los elementos:

Example

Exit the loop when x is "banana":

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
  print(x)  
  if x == "banana":  
    break

Example

Exit the loop when x is "banana", but this time the break comes before the print:

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
  if x == "banana":  
    break  
  print(x)

**La declaración continue**

Con la instrucción continue podemos detener la iteración actual del ciclo y continuar con la siguiente:

Example

Do not print banana:

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
  if x == "banana":  
    continue  
  print(x)

**La función range()**

Para recorrer un conjunto de código un número específico de veces, podemos usar la función range(),

La función range() devuelve una secuencia de números, comenzando desde 0 de forma predeterminada, se incrementa en 1 (de forma predeterminada) y termina en un número especificado.

### Example

Using the range() function:

for x in range(6):  
  print(x)

Tenga en cuenta que el rango (6) no son los valores de 0 a 6, sino los valores de 0 a 5.

La función range () tiene por defecto 0 como valor inicial, sin embargo, es posible especificar el valor inicial agregando un parámetro: range (2, 6), que significa valores de 2 a 6 (pero sin incluir 6):

Example

Using the start parameter:

for x in range(2, 6):  
  print(x)

La función range () tiene como valor predeterminado incrementar la secuencia en 1, sin embargo, es posible especificar el valor de incremento agregando un tercer parámetro: range (2, 30, 3):

### Example

Increment the sequence with 3 (default is 1):

for x in range(2, 30, 3):  
  print(x)

**Else en el bucle For**

La palabra clave else en un bucle for especifica un bloque de código que se ejecutará cuando finalice el bucle:

### Example

Imprima todos los números del 0 al 5 e imprima un mensaje cuando finalice el ciclo:

for x in range(6):  
  print(x)  
else:  
  print("Finally finished!")

**Nota**: El bloque else NO se ejecutará si el bucle se detiene mediante una instrucción break.

Example

Break the loop when x is 3, and see what happens with the else block:

for x in range(6):  
  if x == 3: break  
  print(x)  
else:  
  print("Finally finished!")

**Bucles anidados**

Un bucle anidado es un bucle dentro de un bucle.

El "bucle interno" se ejecutará una vez por cada iteración del "bucle externo":

### Example

Print each adjective for every fruit:

adj = ["red", "big", "tasty"]  
fruits = ["apple", "banana", "cherry"]  
  
for x in adj:  
  for y in fruits:  
    print(x, y)

**La declaración pass**

Los bucles for no pueden estar vacíos, pero si por alguna razón tiene un bucle for sin contenido, ingrese la instrucción pass para evitar errores.

### Example

for x in [0, 1, 2]:  
  pass