fidélitas
INGENIERÍA EN SISTEMAS DE COMPUTACIÓN

APRENDAMOS A PROGRAMAR

Programación Básica.

Clase 05

Python

Contenido

Introducción a las estructuras repetitivas

Es muy común encontrar en la práctica algunos programas cuyas operaciones se deben ejecutar un número repetido de veces, quizás esa cantidad de veces está muy bien definida o quizás se realiza de acuerdo con una situación específica.

De hecho, las estructuras repetitivas se presentan en nuestra vida a diario y en cada momento.

Algunos eventos de la vida diaria se repiten una determinada cantidad de veces, por ejemplo, **la cantidad de veces que asistimos a la Universidad**.

Otros dependen de una condición para ejecutarse o detener su ejecución, por ejemplo, mientras tengamos un virus que pueda contagiarse a otras personas, no podremos permanecer en lugares conglomerados. No conocemos cuántos días tardaremos en recuperarnos, por lo que **mientras estemos enfermos, debemos permanecer en casa.**

Así como en la vida cotidiana existen acciones que se repiten, en la programación podemos representar esas repeticiones a través de estructuras de control conocidas como **CICLOS.**

Según el lenguaje de programación que se utilice, los ciclos pueden ser **finitos y condicionados**, es decir que actúan bajo una cantidad de veces o una determinada condición. Sin embargo, es necesario aclarar que, por ejemplo, en la programación de dispositivos como tarjetas de Arduino o Raspberry PI, los ciclos no finalizarán hasta que se detengan abruptamente (por medio de un sensor).

Tipos de ciclos

Cuando analizamos un requerimiento que implica la programación de un ciclo podemos identificar que este se repetirá una cantidad de veces o hasta que se cumpla una condición.

Finaliza después de N repeticiones.

Finaliza hasta que algo suceda, una condición.

FOR

Esta estructura cíclica se utiliza cuando conocemos la cantidad de veces se repetirá un conjunto de instrucciones. Para este fin, Python provee la estructura repetitiva llamada **FOR.** A través del ciclo for podemos resolver escenarios como los siguientes:

- **La cantidad de veces que asistimos a este curso**
- **Pedir 10 salarios al usuario**
- **Mostrar los nombres de los estudiantes de esta clase**
- **Mostrar al usuario los números pares del 1 al 100**

Estructura del ciclo FOR

```
for i in range(20):
    print(i)
```

i=variable de control
20=cantidad de veces que se repetirá el ciclo

Nota: se asume un cero como valor inicial.

```
for i in range(5,10):
    print(i)
```

i=variable de control

5= valor inicial

10= valor final

Nota: Se asume como valor final 9.

```
for i in range(3,15,5):
    print(i)
```

i=variable de control 3=valor inicial 15=valor final 5=incremento

Nota: Se debe recordar que la ejecución finalizará un valor antes del valor indicado como valor final. (Por ejemplo, si se indicó 15 como valor final, se realizará 14 veces).

Algunos otros ejemplos son:

```
for i in range(1,100):
    if (100-i) % 5 == 0:
        print(100-i)
```

```
for i in range(1,13):
    for j in range(1,13):
        print(i, "x", j, "=", i*j)
```

```
tabla = int(input("Tabla de multiplicar: "))
for i in range(1,10):
    print(tabla, "x", i, "=", i*tabla)
```

WHILE

Esta estructura repetitiva se utiliza cuando no conocemos la cantidad de veces que se ejecutará una o varias sentencias, sino que sabemos que se repetirá mientras que se cumpla una condición. En esos casos, en Python, utilizamos el ciclo **WHILE** o "mientras". A continuación, se presentan algunos ejemplos:

- **Solicitar una contraseña hasta que ingrese la correcta**
- **Que el usuario adivine un número entre 1 y 100**
- **Asistir a la universidad mientras no hayamos finalizado la cantidad de cursos**
- **Mostrar al usuario los primeros 10 números primos**

Estructura del ciclo WHILE

```
while(condición):
    sentencias
```

La condición es una expresión booleana que devuelve un valor verdadero o falso y determina si la condición se cumple o no.

```
#Este programa busca el factorial de un
#número, consulte a su profesor qué se
#obtiene con el factorial
resultado = 1
valor = int(input("Ingrese un valor: "))
while valor > 0:
    resultado *= valor
    valor -= 1
print(resultado)
```

VALOR INICIAL

Algunos otros ejemplos son:

```
#Este programa busca el factorial de un
#número, consulte a su profesor qué se
#obtiene con el factorial
resultado = 1
valor = int(input("Ingrese un valor: "))
while valor > 0:
    resultado *= valor
    valor -= 1
print(resultado)
```

```
i = 1
while i <= 12:
    j = 1
    while j <= 12:
        print(i, "x", j, "=", i*j)
        j += 1
    i += 1
```

El mismo ejemplo de las tablas de multiplicar utilizando el ciclo while.

```
while True:
    print("No puedo salir")
```

```
usuario = ""
clave = ""
while usuario != "admin" or clave != "123":
    usuario = input("Usuario: ")
    clave = input("Clave: ")
    if usuario == "admin" and clave == "123":
        print("Bienvenido")
    else:
        print("Error, intente otra vez")
```

Ejercicios

- Una fábrica de galletas nos ha solicitado que desarrollemos un programa que calcule la suma total de los salarios de sus 10 colaboradores. Tome en cuenta que en cada salario se deberá deducir el 9% de cargas sociales. Al final muestre el total pagado por la empresa por concepto de salarios.

Tiempo aproximado: 15 minutos

- Desarrolle un programa que calcule y muestre la suma de los números pares contenidos en 10 valores enteros ingresados por el usuario.

Tiempo aproximado: 15 minutos

- Desarrolle un programa que convierta a binario un número ingresado por el usuario. Mejore el ejercicio mostrándolo en octal y hexadecimal.

Tiempo aproximado: 15 minutos

Los ciclos son estructuras muy utilizadas en la programación, nos permiten repetir operaciones con pequeños programas.