CEPSUNI

Dirección de Responsabilidad Social y Desarrollo Sostenible

PYTHON BÁSICO - NIVEL I

TANIA ALDORADIN QUINTANILLA

Clase 3: Estructuras de repetición

Objetivos

Al finalizar con éxito este curso, podrá diseñar, crear, modificar programas usando el lenguaje de programación Python vas a:

- Comenzar con la instalación de Python.
- Conocer los diferentes IDE de Python.
- Conocer las sentencias de control.
- Conocer las sentencias de repetición.
- Conocer los arreglos lineales (Listas, Tuplas y Diccionarios).
- Conocer las operaciones con conjuntos (Unión e intercepción).

Contenido

- Estructuras de control repetitivas
- Uso de la sentencia de repetición while
- Uso de acumuladores y totalizadores
- Uso de break, continue y pass.
- Ejercicios de aplicación

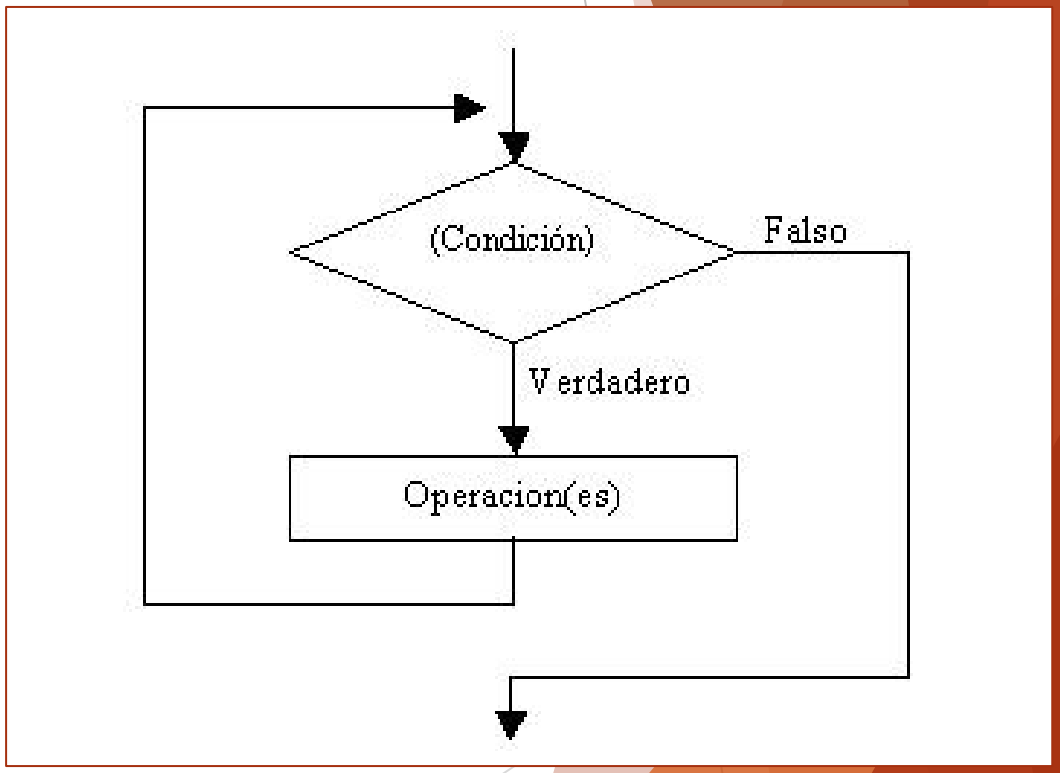
Estructuras de control repetitivas

Ahora que ya hemos podido ver las estructuras condicionales, nos queda por ver otro tipo de estructuras tan importantes como las anteriores. Estas se denominan estructuras repetitivas.

Las estructuras o sentencias repetitivas, también llamados bucles permiten ejecutar una o varias acciones(procesos) repetidas veces, determinado por un rango.

Una estructura repetitiva se caracteriza por:

- La sentencia o las sentencias que se repiten.
- El test o prueba de condición antes de cada repetición, que motivará que se repitan o no las instrucciones.



Uso de la sentencia de repetición 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.

```
while True:
    print("Bucle es infinito")
```

```
while False:
    print("Bucle no se ejecuta")
```

Uso de la sentencia de repetición while

```
# Imprime: 0,1,2,3
i = 0
while i <4:
    i +=1
    print(i)
```

```
# Imprime: 1,2,3,4
i = 0
while i <4:
    print(i)
    i +=1
```

Uso de la sentencia de repetición while

```
# Salida: 3,2,1,0
i = 4
while i > 0:
    i -=1
    print(i)

print("------------------")
```

```
x = 4
while x > 0: x-=1; print(x)
```

Else y While

El uso de la cláusula else al final del while. La sección de código que se encuentra dentro del else, se ejecutará cuando el bucle termine, pero solo si lo hace “por razones naturales”. Es decir, si el bucle termina porque la condición se deja de cumplir, y no porque se ha hecho uso del break.

```
m = 1
while m <= 5:
    print("El número es:",m)
    m +=1
else:
    print("El bucle ha terminado")
```

```
El número es: 1
El número es: 2
El número es: 3
El número es: 4
El número es: 5
El bucle ha terminado
```

Else y While

```
p = 7
while True:
    p -= 1
    print("Número ",p,"es: ",p) #4, 3, 2, 1, 0
    if p == 0:
        break
else:
    print("Este bloque no se ejecuta porque termina con un break")
```

```
Número  6 es:  6
Número  5 es:  5
Número  4 es:  4
Número  3 es:  3
Número  2 es:  2
Número  1 es:  1
Número  0 es:  0
```

Bucles anidados

```
i = 0
j = 0
while i < 3:
    while j < 3:
        print(i,j)
        j += 1
    i += 1
    j = 0
```

```
1 2
2 0
2 1
2 2
```

Bucle for

El for es un tipo de bucle, parecido al while pero con ciertas diferencias. La principal es que el número de iteraciones de un for esta definido de antemano, mientras que en un while no.

```
for i in range(8):
    print(i)
```

```
0
1
2
3
4
5
6
7
```

Ejemplos

```
#range(inicio, fin, salto)
```

```
for i in range(0, 8):
    print(i)
```

```
for i in range(3,8):
    print(i)
```

```
for i in range(1,20,2):
    print(i)
```

Contadores

Un contador es una variable entera que la utilizamos para contar cuando ocurre un suceso. Un contador:

- Se inicializa a un valor inicial.

  contador=0;

- Se incrementa, cuando ocurre el suceso que estamos contado se le suma 1.

  contador= contador+1;

- Otra forma de incrementar el contador:

  contador+=1;

```
a=5
a=a+1
print("El valor de a es: ",a)
a=a+1
print("El valor nuevo de a es: ",a)
a+=1
print("El valor final de a es: ",a)
```

Acumulador

Un acumulador es una variable numérica que permite ir acumulando operaciones. Me permite ir haciendo operaciones parciales.

Un acumulador:

Se inicializa a un valor inicial según la operación que se va a acumular: a 0 si es una suma o a 1 si es un producto.

Se acumula un valor intermedio.

acumulador=acumulador+num;

```
acumulador=0
incremento=100
acumulador  =  acumulador + incremento
print(acumulador)
incremento=120
acumulador  =  acumulador + incremento
print(acumulador)
incremento=75
acumulador  =  acumulador + incremento
print(acumulador)
```

Gracias