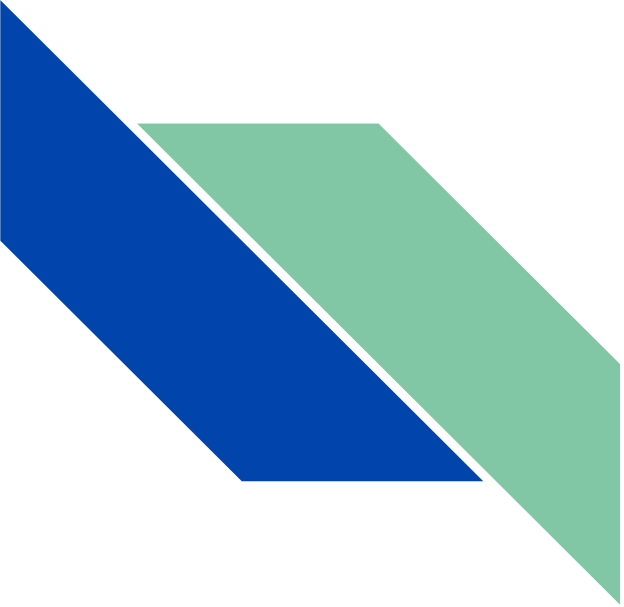

Curso Intensivo de Python - Tema 4

Agustín Arenas

Curso Intensivo de Python - Tema 4

1) Estructuras de repetición
   a) Loops
   b) Bucle while
   c) Bucle for
   d) Control de bucle y anidamiento

Loops

- Los desarrolladores generalmente escriben código que lleva a cabo una acción un número limitado (o ilimitado) de veces.

- Para evitar el código spaguetti los lenguajes (en su mayoría) incorporan **estructuras de repetición**, las cuales hacen que una sentencia o conjunto de sentencias se ejecuten de forma repetida.

- Dichas estructuras vienen en dos sabores en Python: **controladas por condición** (Condition controlled) y **controladas por contador** (Counter-controlled).

Bucle while

- Un bucle controlado por condición ejecuta una sentencia o conjunto de sentencias mientras que la condición sea **True**. En Python, este bucle se utiliza con la palabra reservada **while**.

- El ciclo while consiste de dos partes:
  - La condición a evaluar
  - El conjunto de sentencias a ejecutar

- Es un **pre-tested loop**, lo cual quiere decir, que evalúa la condición antes de ejecutar la/s sentencia/s.

Bucle while

- Puede darse el caso que la condición sea siempre **True**, en tal caso se tiene un bucle **infinito ∞**
- Es una situación a evitar excepto que se tenga control y conocimiento de la misma
- Veamos un ejemplo

Bucle while ∞

```
condicion = 0
while (condicion < 10): # Aquí también debemos usar :
    print('Cuenta: {0}'.format(condicion))
    #¿Hay error aquí?
print('Fin while')
```

Bucle while

```
while (condicion < 10): # Aquí también debemos usar :
    print('Cuenta: {0:.4f}'.format(condicion))
    condicion += 1 #Forma corta de:
    #condicion = condicion + 1
print('Fin while')
```

Bucle for

- El bucle controlado por contador en Python es el **for**
- Su ejecución está dada por un número limitado de veces, sin lugar a generar loops infinitos
- En su ejecución, el bucle for asigna a una variable temporal los valores (Comenzado por el primero) de una secuencia de datos
- Veamos un ejemplo

Bucle for

```
datos = [1,2,3,4,5]
for dato in datos: #dato es la variable temporal. IN
siempre se usa
    print(dato)
print('Fin de for')
```

Bucle for

```
rango_v1 = range(5) #0,1,2,3,4
rango_v2 = range(-5,5) #-5,-4,-3,-2,-1,0,1,2,3,4
rango_v3 = range(-5,5,2) #-5,-3,-1,1,3
#range(inicio,final,salto) es una función definida en Python
#Viene de 3 sabores:
#    -> range(final):
#    genera una secuencia de datos desde 0 hasta final - 1
#    -> range(inicio,final):
#    genera una secuencia de datos desde inicio hasta final - 1
#    -> range(inicio,final,salto):
#    genera una secuencia de datos desde inicio hasta final - 1, cada salto valores
for valor in rango_v1:
    print(valor,end='\t')
print('\nFin for v1')
for valor in rango_v2:
    print(valor,end='\t')
print('\nFin for v2')
for valor in rango_v3:
    print(valor,end='\t')
print('\nFin for v3')
```

Control de bucle y anidamiento

- Los bucles son imprescindibles cuando se tienen que hacer:
  - Acumuladores (Sumadores)
  - Centinelas: Variables que controlan la ejecución hasta que se da una condición
  - Validadores de entrada: Solo se avanza cuando el usuario introduce un dato válido (Para el programador)

- Además para los menues de consola interactivos son muy útiles.

- Existen dos palabras reservadas para controlar la ejecución de un bucle: **break** la cual provoca que se termine la ejecución de forma inmediata y **continúe** la cual provoca que se comience una iteración nueva de forma inmediata (iteración == nueva ejecución desde el comienzo del bucle)

- Como los condicionales, los bucles se pueden anidar para tener un bucle dentro de otro bucle, independiente su tipo.Veamos algunos ejemplos

Control de bucle y anidamiento

```
import random as rd

aleatorio = 0
intentos  = 0

ACIERTO   = 7 #Declaracion de valor constante (Mayúsculas)

while(True): #bucle infinito...
    aleatorio = rd.randint(0,10)
    if(aleatorio == ACIERTO):
        break #Termina ejecución de bucle y las
    #sentencias posteriores NO se ejecutan
    else:
        intentos += 1
print('Se necesitaron {0} intentos'.format(intentos))
```

Control de bucle y anidamiento

```
import random as rd

aleatorio = 0
intentos  = 0

ACIERTO   = 7 #Declaracion de valor constante (Mayúsculas)

while(True): #bucle infinito...
    aleatorio = rd.randint(0,10)
    if(aleatorio == ACIERTO):
        break #Termina ejecución de bucle y las
    #sentencias posteriores NO se ejecutan
    else:
        continue #Se vuelve al comienzo del loop
        #¿Qué produce esto?
        intentos += 1
print('Se necesitaron {0} intentos'.format(intentos))
```

Control de bucle y anidamiento

```
import random as rd
MAX_CANTIDAD_ALUMNOS = 9
MAX_CANTIDAD_EXAMENES = 3
cantidadAlumnos = 0
total = 0
promedio = 0
while (cantidadAlumnos < MAX_CANTIDAD_ALUMNOS):
    print('Alumno N°{0}'.format(cantidadAlumnos + 1))
    print('------------')
    for nota in [0] * 3:  # [0,1,2]
        total += rd.randint(2, 11)  #2 - 10 valores
    promedio = total / MAX_CANTIDAD_EXAMENES
    print('Promedio obtenido
{0:.2f}'.format(promedio))
    print('-----------------------------------')
    total = 0
    promedio = 0
    cantidadAlumnos += 1
print('Fin programa')
```

¿Preguntas?