Programación II

Más iteraciones

Lic. Federico Severino Guimpel
LCC - FCEIA - UNR

While

Hasta aquí el modelo iterativo que presentamos estaba limitado a la cantidad de elementos que teníamos en una lista; sin embargo, hay casos donde esto no nos permite escribir una solución al problema.
Vamos a ver una forma de iteración que utiliza una condición, es decir, mientras la condición se cumpla, se realizan las sentencias que queremos.
La sintaxis de esta definición es:

```
while condición:
    sentencias
```

While

Veamos unos ejemplos, sencillos, de su uso:

```
>>> def felicitaciones():
        edad = 0
        while edad < 5:
                edad = edad + 1
                print("Felicidades,  tienes ",edad)


>>> felicitaciones()
Felicidades,  tienes  1
Felicidades,  tienes  2
Felicidades,  tienes  3
Felicidades,  tienes  4
Felicidades,  tienes  5
```

While

Veamos unos ejemplos, sencillos, de su uso:

```
>>> def eco():
        salir = False
        while not salir:
                entrada = input("Ingrese un mensaje: ")
                if entrada == 'adios':
                        salir = True
                else:
                        print(entrada)


>>> eco()
Ingrese un mensaje: hola
hola
Ingrese un mensaje: cómo
cómo
Ingrese un mensaje: estás
estás
Ingrese un mensaje: adios
```

While

Vamos a reescribir algunos de los programas que hicimos utilizando al **while**.

Comencemos con el Factorial:

```
>>> def factorialWhile(n):
        if n == 0:
                return 1
        else:
                x = 1
                factorial = 1
                while x <= n:
                        factorial = factorial * x
                        x = x + 1
                return factorial


>>> factorialWhile(1)
1
>>> factorialWhile(5)
120
```

While

Si analizamos el código del Factorial:

```
>>> def factorialWhile(n):
        if n == 0:
                return 1
        else:
                x = 1
                factorial = 1
                while x <= n:
                        factorial = factorial * x
                        x = x + 1
                return factorial
```

¿el if inicial, es preciso que esté?

While

Podemos ver que la condición del **while** hace que se itere mientras x sea menor o igual que n. Pensemos entonces, si **n** fuera negativo:

- ¿La función termina?
- Si es así, ¿qué resultado nos muestra?
- ¿Es correcto eso?

Escriba una función alternativa que resuelva este problema.

En la presentación 1 (slide 45), escribimos un programa recursiva que permita el ingreso de números enteros hasta que se ingrese un 0. Cuando se detecte el ingreso de un 0, la función debe mostrar la suma de los números que se ingresaron y la cantidad (sin incluir al último 0).

¿Se podría escribir ese programa usando **for**?

While

Veremos una alternativa usando **while**:

```
>>> def sumaNumeros():
        suma = 0
        contador = 0
        n = int(input("Ingrese un número o 0 para finalizar: "))
        while n!=0:
                suma = suma + n
                contador = contador + 1
                n = int(input("Ingrese un número o 0 para finalizar: "))
        print("La suma de los valores ingresados fue", suma, "y se ingresaron", contador, "numeros")


>>> sumaNumeros()
Ingrese un número o 0 para finalizar: 1
Ingrese un número o 0 para finalizar: 2
Ingrese un número o 0 para finalizar: 5
Ingrese un número o 0 para finalizar: 10
Ingrese un número o 0 para finalizar: 0
La suma de los valores ingresados fue 18 y se ingresaron 4 numeros
```

While

Escribir una función que permita ingresar un número mientras sea distinto de -1. Cuando finalice de ingresar se debe mostrar el mayor y el menor número ingresado (sin tener en cuenta el -1 final).

Vamos a escribir una versión recursiva del programa. ¿De qué manera podría escribirlo usado iteraciones?

While

```
>>> def encuentraMenorYMayor():
        ingreso = int(input("Ingrese un número o -1 para finalizar: "))
        if ingreso == -1:
                print("No se han ingresado números")
                return
        else:
                maximo = ingreso
                minimo = ingreso
        while (ingreso != -1):
                if minimo > ingreso:
                        minimo = ingreso
                if maximo < ingreso:
                        maximo = ingreso
                ingreso = int(input("Ingrese un número o -1 para finalizar: "))
        print("El máximo ingresado fue", maximo, "y el mínimo", minimo)
```

Analizando el código podemos ver que estamos realizando un primer ingreso de datos y validamos el valor ingresado.

Esto es porque, no podemos inicializar arbitrariamente a las variables que van a almacenar el máximo y mínimo ingresados, sin tener un valor ingresado, válido, por el usuario.

La salida, con este código sería:

```
>>> encuentraMenorYMayor()
Ingrese un número o -1 para finalizar: 7
Ingrese un número o -1 para finalizar: 1
Ingrese un número o -1 para finalizar: 18
Ingrese un número o -1 para finalizar: 9
Ingrese un número o -1 para finalizar: -1
El máximo ingresado fue 18 y el mínimo 1
```