

EJERCITACIÓN Nº 4: CICLOS INDEFINIDOS

Cátedra Redictado Programación II

Abril 2024


Ciclos: Definido (for), While, Interactivos, y Con Centinelas

A continuación mostraremos el código de la función **clasificación** la cual resuelve el problema inicialmente planteado clasificar un número dado según este sea Positivo, Negativo, o Cero.



```
1
2 def clasificacion (num):
3     """
4     numero: Integer
5     mensaje: String
6     clasificacion: Integer -> String
7     Clasifica un número dado en Positivo, Cero o Negativo.
8     Devuelve un mensaje con la categoría asignada al valor dado.
9     clasificacion (10) = "Es Positivo"
10    clasificacion (-10) = "Es Negativo"
11    clasificacion (0) = "Es Cero"
12    """
13    respuesta = ""
14    if (num > 0):
15        respuesta = "Es Positivo"
16    elif (num == 0):
17        respuesta = "Es Cero"
18    else:
19        respuesta = "Es Negativo"
20
21    return respuesta
22
23 def test_clasificacion():
24     assert clasificacion (0) == "Es Cero"
25     assert clasificacion (-10) == "Es Negativo"
26     assert clasificacion (10) == "Es Positivo"
```


1) **Ciclo Definido – for**: contamos con la cantidad de iteraciones de antemano mediante un valor que es conocido o pedido previamente por teclado.



```
1 def ciclodefinido ():
2     """ciclodefinido:None -> None
3     Procesa una cantidad específica de números dados por el usuario ↔
4     categorizándolos como positivos, cero, o negativo. Imprime un ↔
5     mensaje por pantalla con la clasificación.
6     """
7     cant = int (input ("Cuántos números quiere procesar?:"))
8     for j in range (0, cant):
9         num = int (input ("Ingrese un número: "))
10        respuesta = clasificacion(num)
11        print (respuesta)
```


Construir la tabla de iteraciones para el ciclo for de la función ciclodefinido() con: cant =3, num=2, num =0, num =-1

2) Ciclo con Condición: while




```
1 def ciclowhile ():
2     cant = int (input ("Cuántos números quiere procesar?:"))
3     j = 0
4     while j < cant
5         num = int (input ("Ingrese un número: "))
6         respuesta = clasificacion(num)
7         print (respuesta)
8         j = j + 1
```

Construir la tabla de iteraciones para el ciclo while de la función ciclowhile() con: cant =3, num=2, num =0, num =-1

a) Ciclo while interactivo:

```
1
2 def ciclointeractivo():
3     hay_mas_datos = "Si"
4     while hay_mas_datos == "Si":
5         num = int(input("Ingrese un numero: "))
6         respuesta = clasificacion(num)
7         print (respuesta)
8         hay_mas_datos = input("Quiere seguir? <Si-No>: ")
```


Construir la tabla de iteraciones para el ciclo while de la función ciclointeractivo() con la siguiente secuencia de valores, asociando las variables num, y hay_mas_datos, a los siguientes pares ordenados: (2, "Si"), (0, "Si"), (-1, "No")

b) Ciclo while con centinela:

```
1
2 def leer_centinela():
3     return input("Ingrese un numero (* para terminar): ")
4
5 def ciclocentinela():
6     centinela = leer_centinela()
7     while centinela != "*":
8         num = int(centinela)
9         respuesta = clasificacion(num)
10        print (respuesta)
11        centinela = leer_centinela()
```

Construir la tabla de iteraciones para el ciclo while de la función ciclocentinela() con la siguiente secuencia de valores, asociado a la variable centinela: centinela = 2, centinela = 0, centinela = -1, centinela = "*"


d) Ciclo Infinito:



```
1 def muchos_pcn():
2     while True:
3         centinela = input("Ingrese un numero ('*' para terminar): ")
4         if centinela == '*':
5             break
6         num = int(centinela)
7         respuesta = clasificacion(num)
8         print (respuesta)
```

Para usar ciclos infinitos cuando los necesitemos definir como tales, se seguirá este formato presentado. Pensar ejemplos de ciclos infinitos, algunos procesos que nunca deben dejar de funcionar salvo que se cumpla una condición específica que les indique que deben darse de baja.

Tener en cuenta que en general los ciclos mal diseñados se tornan infinitos porque su condición de salida nunca se cumple. Como por ejemplo este caso:



```
1 num = 5
2 while num <= 5:
3     print("El Espacio la Última Frontera ... ")
```

¿Cuál es el problema?

Ejercitación

Desarrollar cada uno de los ejercicios de forma completa, detallando cada una de las etapas involucradas en la construcción de un programa, siguiendo las indicaciones de la receta para programas Python. Para la etapa de prueba de nuestro programa se deberá emplear el módulo `pytest`. Tener en cuenta que deberá crear funciones que permitan el pasaje de argumentos para posteriormente testear el programa contra los datos de testing.

Bucles For

EJERCICIO 1. Escribir un ciclo definido para imprimir por pantalla todos los números entre 10 y 20.

EJERCICIO 2. Escribir una función que tome una cantidad m de valores que son ingresados por el usuario y, en la medida que lo ingresa, se muestra el factorial de ese número. El valor de m es ingresado inicialmente por el usuario.

EJERCICIO 3. Usando la función, dada como ejemplo en la presentación de *La Receta en Python*, para convertir una temperatura en Fahrenheit a Celsius se pide que genere una tabla de conversión de temperaturas, desde 0°F hasta 120°F, de 10 en 10.

EJERCICIO 4. Escriba una función que reciba una cantidad de pesos, una tasa de interés y un número de años para pagarlo, y retorne, como resultado, el monto final a obtener. La fórmula a utilizar es:

$$C_n = C \times \left(1 + \frac{x}{100}\right)^n$$

Donde C es el capital inicial, x es la tasa de interés y n es el número de años a calcular. En el programa se podrá visualizar los intereses por año, el monto original pedido, el monto final de dinero abonado. Los intereses por año, mostrarlo de forma tabular, encabezado por el monto original y al final el monto total pagado por el crédito.

EJERCICIO 5. Escribir una función que reciba un número n por parámetro e imprima los primeros n números triangulares, junto con su índice. Los números triangulares se obtienen mediante la suma de los números naturales desde 1 hasta n . Es decir, si se piden los primeros 5 números triangulares, el programa debe imprimir:

```
1 1 - 1
2 2 - 3
3 3 - 6
4 4 - 10
5 5 - 15
```

Nota: hacerlo usando y sin usar la ecuación:

$$\sum_{i=1}^n i = \frac{n \times (n + 1)}{2}.$$

¿Cuál realiza más operaciones?

Bucles While

EJERCICIO 6. Nos piden que escribamos una función que le pida al usuario que ingrese un número positivo. Si el usuario ingresa cualquier número que no sea lo pedido, se le debe informar de su error mediante un mensaje y volverle a pedir el número.

Resolver este problema usando:

- (a) Un ciclo interactivo.
- (b) Un ciclo con centinela.

¿Tendría sentido hacerlo con ciclo denido? Justificar.

EJERCICIO 7. Escribir un programa que permita al usuario ingresar un conjunto de notas, preguntando a cada paso si desea ingresar más notas, e imprimiendo el promedio correspondiente, al finalizar la toma de datos.

EJERCICIO 8. Escribir un programa que le pida al usuario que ingrese una sucesión de números naturales (primero uno, luego otro, y así hasta que el usuario ingrese '-1' como condición de salida). Al final, el programa debe imprimir cuántos números fueron ingresados, la suma total de los valores y el promedio.

EJERCICIO 9. Escribir una función que reciba dos números como parámetros, y devuelva cuántos múltiplos del primero hay, que sean menores que el segundo.

- a) Implementarla utilizando un ciclo `for`, desde el primer número hasta el segundo.
- b) Implementarla utilizando un ciclo `while`, que multiplique el primer número hasta que sea mayor que el segundo.
- c) Comparar ambas implementaciones: ¿Cuáles más clara? ¿Cuál realiza menos operaciones?

EJERCICIO 10. Escriba una función que reciba un número natural e imprima todos los números primos que hay hasta ese número. Para esto se pide que:

- a) Defina una función `es_primo` que toma un número natural y verifique si es un número primo.
- b) Resuelva el problema usando la función definida en el punto anterior.

EJERCICIO 11. Manejo de contraseñas

- a) Escribir un programa que contenga una contraseña inventada, que le pregunte al usuario la contraseña, y no le permita continuar hasta que la haya ingresado correctamente.
- b) Modificar el programa anterior para que solamente permita una cantidad fija de intentos.
- c) Modificar el programa anterior para que sea una función que devuelva si el usuario ingresó o no la contraseña correctamente, mediante un valor booleano (`True` o `False`).


EJERCICIO 12. Potencias de dos

- a) Escribir una función `es_potencia_de_dos` que reciba como parámetro un número natural, y devuelva `True` si el número es una potencia de 2, y `False` en caso contrario.
- b) Escribir una función que, dados dos números naturales pasados como parámetros, devuelva la suma de todas las potencias de 2 que hay en el rango formado por esos números (0 si no hay ninguna potencia de 2 entre los dos). Utilizar la función `es_potencia_de_dos`, descrita en el punto anterior.

Ciclos Infinitos

Desde hace mucho tiempo los ciclos infinitos vienen provocando dolores de cabeza a los programadores. Cuando un programa deja de responder y se utiliza todos los recursos de la computadora, suele deberse a que entró en un ciclo del que no puede salir. Estos bucles pueden aparecer por una gran variedad de causas. A continuación algunos ejemplos de ciclos de los que no se puede salir, siempre o para ciertos parámetros. Queda como ejercicio encontrar el error en cada uno.

a) Ciclos Infinitos con problemas




```
1 def buscar_impar(x):
2     """Divide el número recibido por 2 hasta que sea impar."""
3
4     while x % 2 == 0:
5         x = x / 2
6
7     return x
```

Construir la tabla de iteraciones para el ciclo `while` de la función `buscar_impar()` llamada con los siguientes argumentos:

- `buscar_impar(5)`
- `buscar_impar(2)`
- `buscar_impar(6)`
- `buscar_impar(0)`

¿Pudo encontrar algún problema? Explique con sus palabras que sucedió.

b) Ciclos Infinitos con problemas



```
1 def menor_factor_primo(x):
2     """Devuelve el menor factor primo del número x."""
3     n = 2
4     while n <= x:
5         if x % n == 0:
6             return n
```

Referencias

- [1] Think Python: How to Think Like a Computer Scientist, Allen B. Downey, 2nd Edition, Version 2.2.18.
- [2] Algoritmos y Programación I, Aprendiendo a programar usando Phyton como herramienta, Rosita Wachenchauzer et.al., 2016, (sin publicar).