Estructuras repetitivas

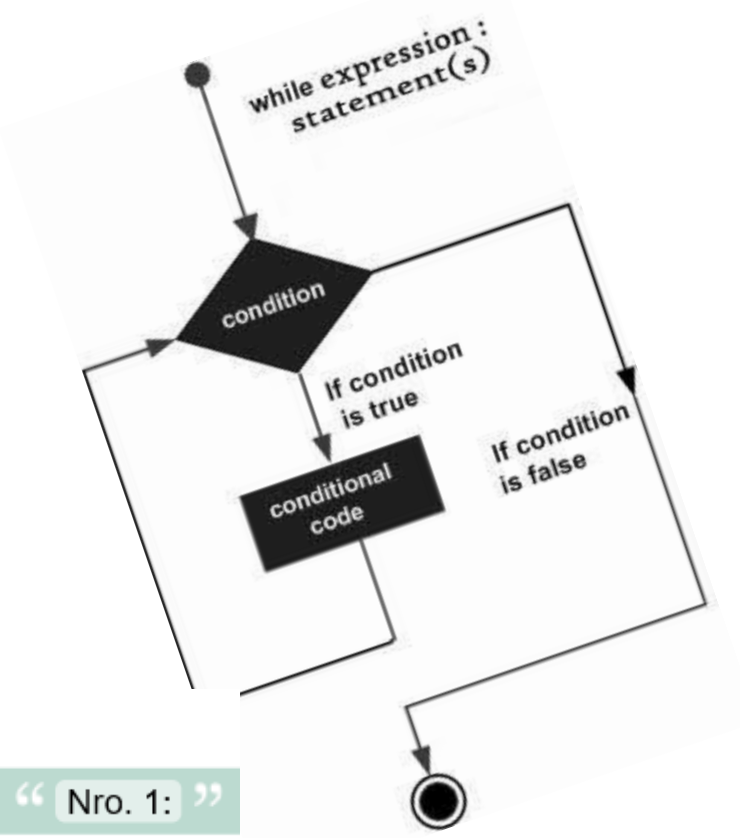
Juan Zamora Osorio

IWI-131 / 2021

Repeticiones

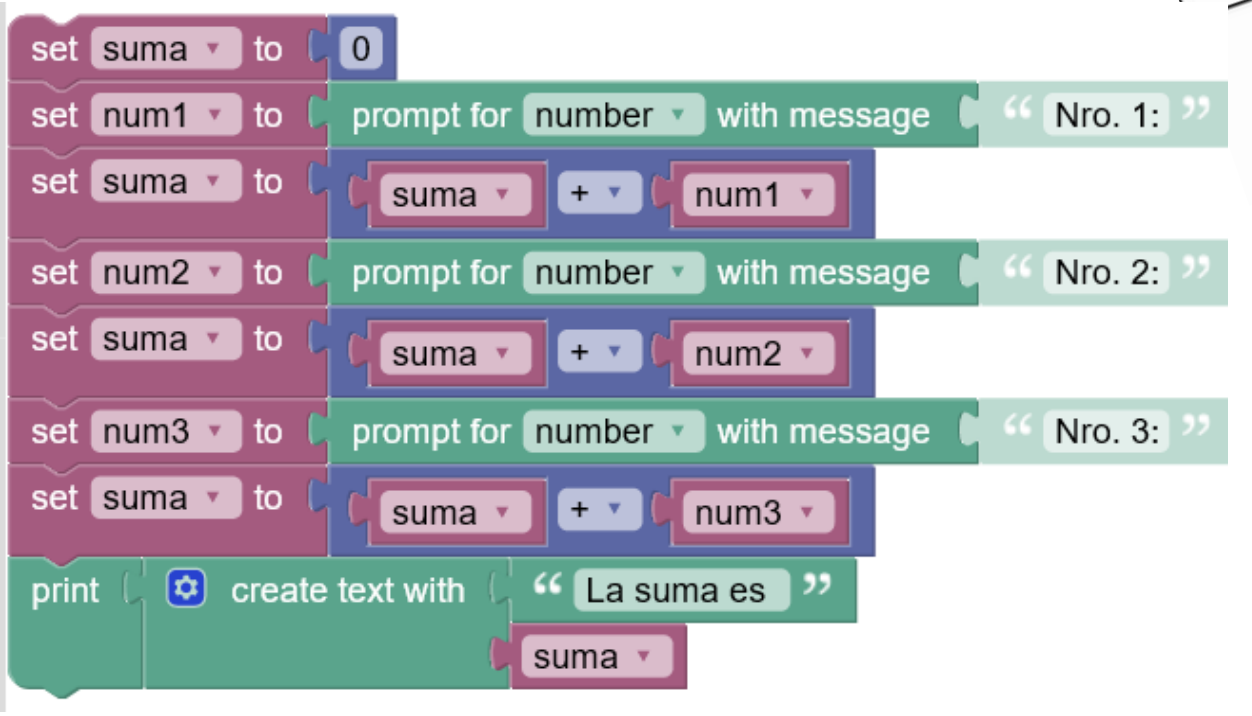
- En ocasiones es útil poder ejecutar un **bloque de instrucciones** tantas *veces como queramos*
- Una alternativa es repetir el mismo bloque una y otra vez en nuestro programa
- La **otra** es usar una estructura que tome nuestro bloque y de manera interna lo ejecute tantas veces como queramos
- Además, existen casos en que la cantidad de repeticiones solo puede ser conocida al ejecutar el programa

Repeticiones



Por ejemplo

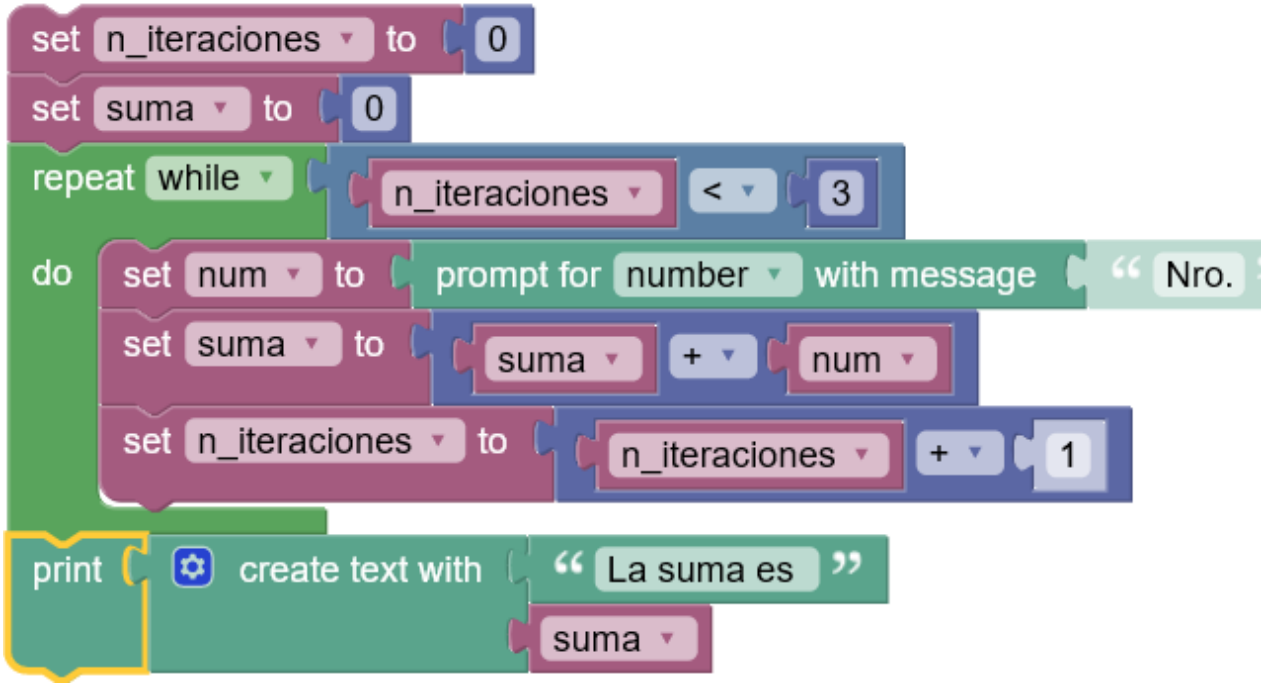
- Pedir 3 números a un usuario
- Se reporta la suma de los números ingresados



Repeticiones

Por ejemplo (cont.)

- Pedir 3 números a un usuario
- Se reporta la suma de los números ingresados



Repeticiones

Por ejemplo

- Pedir un número a un usuario hasta que ingrese un valor negativo
- Al ingresar el valor negativo, se reporta la suma de los números ingresados

Repeticiones

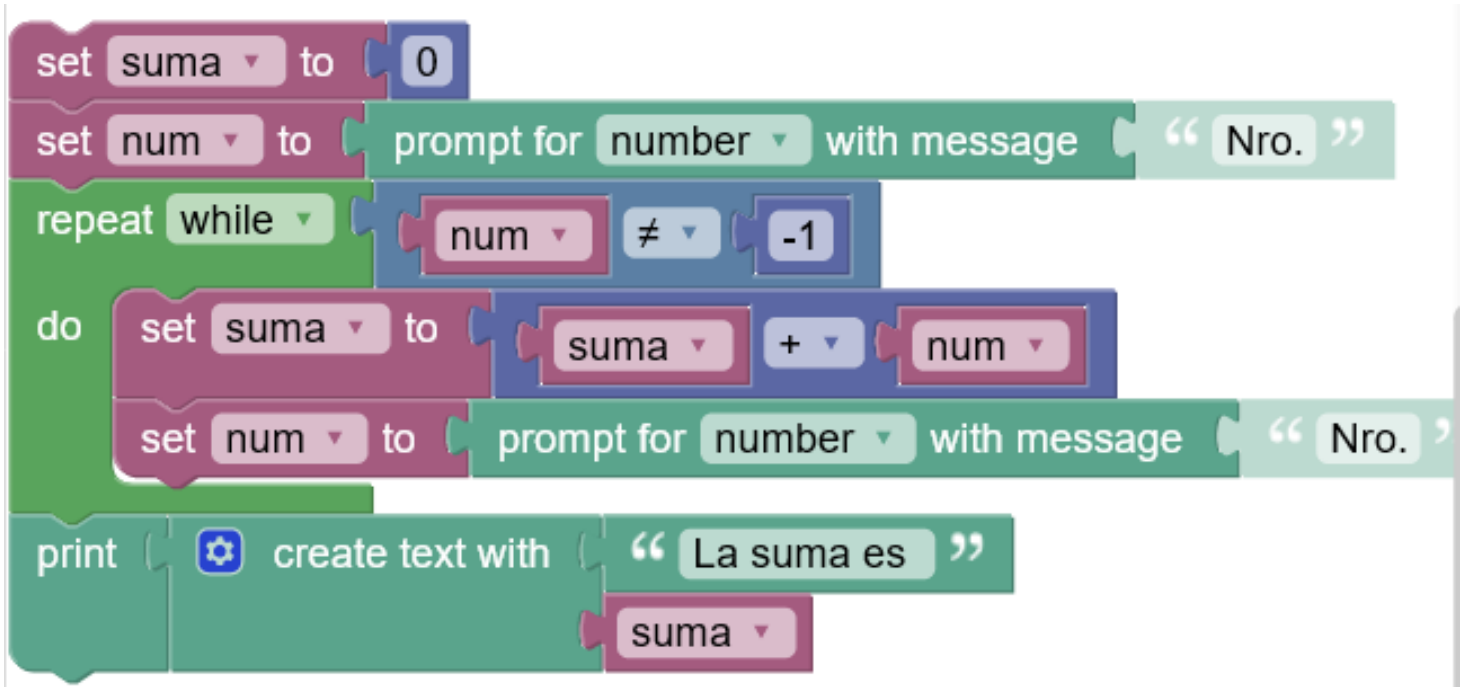
Por ejemplo

- Pedir un número a un usuario hasta que ingrese un valor negativo
- Al ingresar el valor negativo, se reporta la suma de los números ingresados

No sabemos qué valor debe tener la condición, ya que desconocemos la cantidad de repeticiones

¿Qué hacer cuando no se sabe la cantidad de iteraciones?

- Identificar cuál es la condición de término de la iteración
  - Se alcanza un valor de una variable superior/inferior a un determinado umbral
  - Se recibe un determinado ingreso por parte del usuario
- Se debe iterar mientras se cumpla esa condición (sea True)



Repeticiones

¿Qué hacer cuando no se sabe la cantidad de iteraciones?

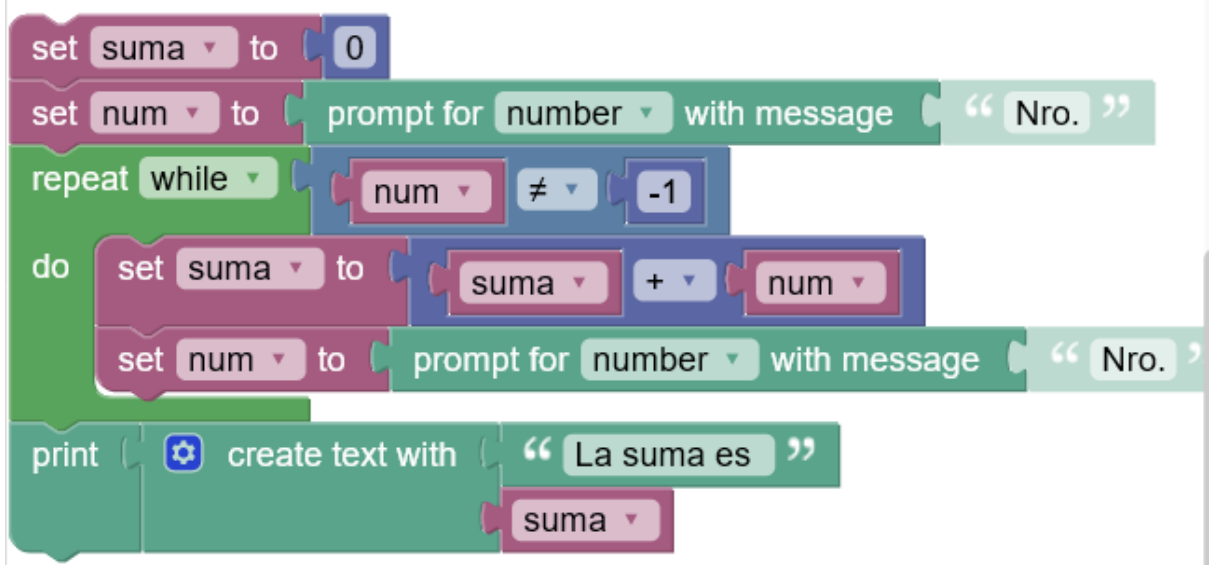
- Identificar cuál es la condición de término de la iteración
  - Se alcanza un valor de una variable superior/inferior a un determinado umbral
  - Se recibe un determinado ingreso por parte del usuario
- Se debe iterar mientras se cumpla esa condición (sea True)

Sintáxis

- Palabra clave: **while**
- **Condición** es una expresión Booleana

```
while CONDICION :
    instrucción_1
    instrucción_2
    ...
```

Sintáxis




```
suma = 0
num = float(input('Nro.'))
while num > 0:
    suma = suma + num
    num = float(input('Nro.'))
#print('La suma es ' + str(suma))
print('La suma es ', suma)
```

Considere el siguiente procedimiento para aproximar la raíz cuadrada de un número S:

```
S = int(input(":"))

x_0 = S/2 # estimación inicial
x_1 = (x_0 + S/x_0)/2 # primera actualización
print(x_1)

x_0 = x_1
x_1 = (x_0 + S/x_0)/2 # actualización
print(x_1)

x_0 = x_1
x_1 = (x_0 + S/x_0)/2 # actualización
print(x_1)
```

Reemplacemos repeticiones manuales por un CICLO:

```
S = int(input(":"))

x_0 = S/2 # estimación inicial
x_1 = (x_0 + S/x_0)/2 # primera actualización
print(x_1)

n_iteraciones = 10

while n_iteraciones >= 0:
  x_0 = x_1
  x_1 = (x_0 + S/x_0)/2 # actualización
  print(x_1)
  n_iteraciones = n_iteraciones - 1
```

```
S = int(input(":"))

x_0 = S/2 # estimación inicial
x_1 = (x_0 + S/x_0)/2 # primera actualización
print(x_1)

x_0 = x_1
x_1 = (x_0 + S/x_0)/2 # actualización
print(x_1)

x_0 = x_1
x_1 = (x_0 + S/x_0)/2 # actualización
print(x_1)
```

Reemplacemos CICLO FIJO por uno que dependa del nivel del progreso de la aproimación:

```
S = int(input(":"))

x_0 = S/2 # estimación inicial
x_1 = (x_0 + S/x_0)/2 # primera actualización
print(x_1)
diferencia = x_1

while diferencia > 1e-4:
  x_0 = x_1
  x_1 = (x_0 + S/x_0)/2 # actualización
  print(x_1)
  diferencia = abs(x_1 - x_0)
```

```
S = int(input(":"))

x_0 = S/2 # estimación inicial
x_1 = (x_0 + S/x_0)/2 # primera actualización
print(x_1)

x_0 = x_1
x_1 = (x_0 + S/x_0)/2 # actualización
print(x_1)

x_0 = x_1
x_1 = (x_0 + S/x_0)/2 # actualización
print(x_1)
```

Ejercicios

- Pedir 10 nombres y por cada uno, mostrar un mensaje de saludo.
- Pedir 10 números y al finalizar, se debe entregar la suma de los números ingresados.
- Pedir números hasta que se ingrese un 0. Al finalizar, se debe entregar el producto de los números ingresados.
- Pedir números entre 0 y 100. Si se ingresa un número mayor que 100 o el programa termina
- **Pedir números entre 0 y 100. Si se ingresa un número mayor que 100 o menor que 0 el programa termina**
- Pedir números hasta que se ingrese un 0. Al finalizar, se debe entregar el producto de los números ingresados.
- Pedir números hasta que se ingrese un 0. Al finalizar se debe mostrar el mayor valor ingresado.

Promedios de varias personas

- Calcular promedio de 3 notas para 4 personas. Reportar finalmente el promedio de c/u.

Hint: Comience con el programa que calcula el promedio de 3 notas para una sola personas. Luego, repita.

Ejercicios: Patrones con ciclos anidados

```
Patron 1

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

```
Patron 2

5
5 4
5 4 3
5 4 3 2
5 4 3 2 1
```

```
Patron 3

5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

```
Patron 4

5 4 3 2 1
5 4 3 2
5 4 3
5 4
5
```

```
Patron 5

        1
      1 2
    1 2 3
  1 2 3 4
1 2 3 4 5
```

```
Patron 6

    1
  1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

Parámetro: cantidad de filas (ejemplos usan 5 filas)

+ Ejercicios

Escriba un programa que muestre la tabla de multiplicar del 1 al 10 del número ingresado por el usuario:

```
Ingrese un numero:9
9 x 1 = 9
9 x 2 = 18
9 x 3 = 27
9 x 4 = 36
9 x 5 = 45
9 x 6 = 54
9 x 7 = 63
9 x 8 = 72
9 x 9 = 81
9 x 10 = 90
```

+ Ejercicios

- *Ídem* al anterior, pero en lugar de ingresar un número, simplemente encontrar todos los números cuyo producto resulte en un número par (use las tablas de 1 al 10).
  - Sugerencia: Utilice el operador mod %

+ Ejercicios

Escriba un programa que lea un número y muestre sus divisores.

```
Ingrese un numero:12
1 2 3 4 6 12
```

+ Ejercicios

- Utilice el programa anterior. Luego, dado un número, determinar si es primo o no.

  - Recuerde que un número primo es aquel mayor que 1 y que solo es divisible por 1 y por sí mismo. Ej. 1,2,3,5,7,…
  - Sugerencia: Utilice operador modulo %

+ Ejercicios

Escriba un programa que lea números hasta que se ingrese un número menor que el anterior. Al finalizar se debe imprimir cuántos números se ingresaron, sin contar el último.

```
Ingrese un numero:5
Ingrese un numero:5.2
Ingrese un numero:11
Ingrese un numero:11
Ingrese un numero:9
Números: 4
```

+ Ejercicios

Escriba un programa que lea dos números enteros y muestre la multiplicación entre ellos sin usar el operador *

```
Ingrese un numero:4
Ingrese un numero:7
28
```

+ Ejercicios

La famosa serie **FizzBuzz** para un número natural N es una sucesión desde 1 hasta N donde:

Los números que sean múltiplos de 3 se cambian por *Fizz*.
Los números que sean múltiplos de 5 se cambian por *Buzz*.
Los números que sean múltiplos de 3 y 5 se cambian por *FizzBuzz*.

Escriba un programa que lea N e imprima la serie **FizzBuzz**.

```
Ingrese n:15
1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz
```

[40 %] Uno de los grupos de música más importantes de Pythonia es Deuman. Sus integrantes han producido un nuevo disco corto de cuatro canciones, las que llamaremos: A, B, C y D. El grupo quiere saber cuál de las canciones tiene mejor recepción por parte del público y para ello realizará una encuesta a 1000 personas. Cada persona debe ingresar un dígito entre 1 y 7 para indicar cuánto le gustó cada canción: 1 es lo peor y 7 es lo máximo. Por ejemplo, si una persona ingresa 3675, siginifica que le asignó nota 3 a la canción A, un 6 a la canción B, 7 a la canción C y 5 a la canción D. Para hacer el proceso más automático, Deuman le solicita a Usted implementar lo siguiente:

a) **[20 %]** Escriba la función `mejor(votos)`, donde `votos` es un entero de 4 dígitos que almacena los votos de las canciones, en la forma descrita. Los votos contendrán siempre 4 dígitos entre 1 y 7. La función debe retornar la canción mejor evaluada, es decir, el string `'A'`, `'B'`, `'C'` o `'D'` dependiendo de cuál obtuvo la nota más alta. En caso de haber empate en la nota máxima, la función debe retornar cualquiera de las canciones que empataron en el máximo. Sin embargo, hay una excepción: si en el voto ninguna canción obtiene nota ≥ 4, entonces la función debe retornar el string vacío: `''`, pues suponemos que a la persona encuestada en realidad no le gustó mucho ninguna canción. Observe que la función debe retornar su resultado, pero no imprimir nada.

Ejemplo:

```
>>> print(mejor(1621))
'B'
>>> print(mejor(1166))
'C'
>>> print(mejor(1223))
''
```

b) **[20 %]** Utilizando la función anterior, se le pide ahora que escriba un programa que pregunte su voto a las 1000 personas encuestadas. El programa debe responder cuál fue la canción que resultó favorita en la mayor cantidad de votos. En caso de empate, puede escribir cualquiera de las que empataron en el máximo. Suponga que los votos contendrán siempre 4 dígitos entre 1 y 7.

Ejemplo:

```
Voto? 6727
Voto? 1166
Voto? 7117
...
Voto? 7777
La favorita es D con 510 votos.
```

Nota: Usted puede contestar a esta pregunta aunque no haya respondido la parte (a), o aunque la tenga incorrecta. Lo importante es que utilice adecuadamente la función tal y como se describió, haciendo los llamados de manera adecuada, en el momento oportuno y utilizando correctamente lo que la función retorna.