BUCLES

Fundamentos de Informática

Grado en Ingeniería Química

Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación

UC | ETS de Ingenieros Industriales y de Telecomunicación

ESTRUCTURAS DE REPETICIÓN

Una estructura de repetición, también llamada lazo o bucle, hace posible la ejecución repetida de una sección específica de código.

Hay dos tipos básicos de estructuras de repetición, cuya diferencia principal radica en cómo se controlan las mismas:

- Repetición controlada por contador o bucle **for-in**:

  Un bloque de sentencias se ejecuta una vez para cada uno de los valores que va tomando un contador. Se ejecuta un número **específico** de veces, siendo el número de repeticiones conocido antes de que empiece la ejecución de tal bucle.

- Repetición controlada por expresión lógica o bucle **while**:

  Un bloque de sentencias se ejecuta un número **indefinido** de veces, mientras se satisfaga alguna condición establecida por el usuario, lo cual equivale a que una expresión lógica tome el valor True.

bucle for-in

Ejecuta un bloque de sentencias un número específico de veces.
Sintaxis:

for *índice* **in range(start, stop [, step]):**
    sentencia_1
    [sentencia_2]
    ...
} *Cuerpo del bucle*

índice es una variable que toma los valores generados por range. Para cada valor de la serie, se realiza una pasada, iteración o ciclo del bucle.
range es un **generador de valores**, desde **start** hasta **stop**, incluido el primero, pero **no** el último, con pasos de **step**.
start, stop y **step** pueden ser constantes, variables o expresiones enteras.

Posibilidades y valores por defecto**:**
range(start, stop, step) donde**:**

start es el valor inicial de la serie generada.

stop es el valor final de la serie*, no incluído*.

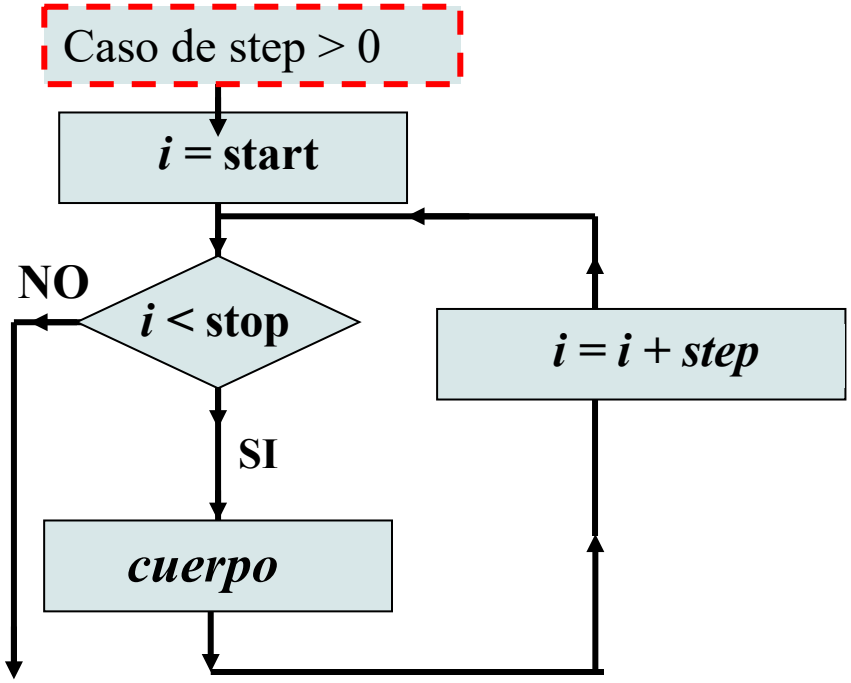
step indica el incremento/decremento en la serie. Puede ser positivo o negativo, pero no cero.

Nº iteraciones =$len(range(\text{start,stop,step}))$

range(start, stop) step es 1 por defecto.
range(stop) start es 0 y step 1 por defecto.

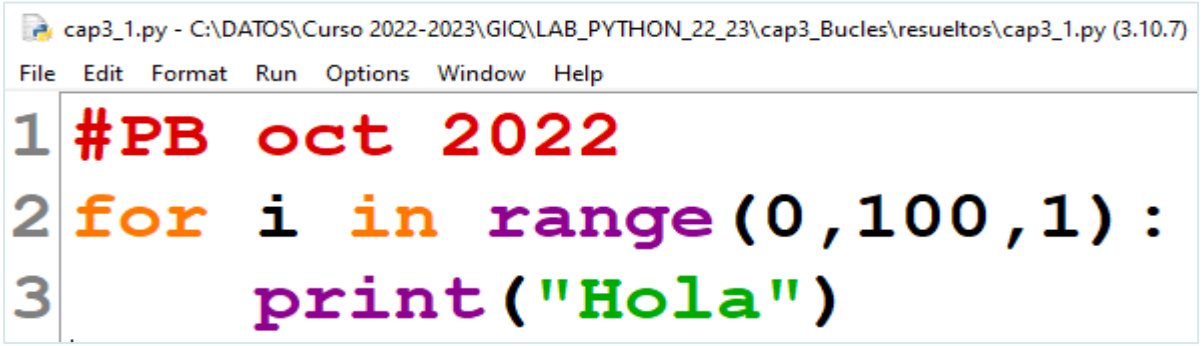
bucle for-in, con step positivo

for *i* **in range (start, stop [, step]):**
cuerpo del bucle



bucle for-in con step positivo. Ejercicio 1

Mostrar un mensaje por pantalla, por ejemplo, Hola, 100 veces.



Primero range genera la serie: 0, 1, 2, …, 99, y *a continuación*, se realizará una iteración para cada uno de los valores de esa serie.

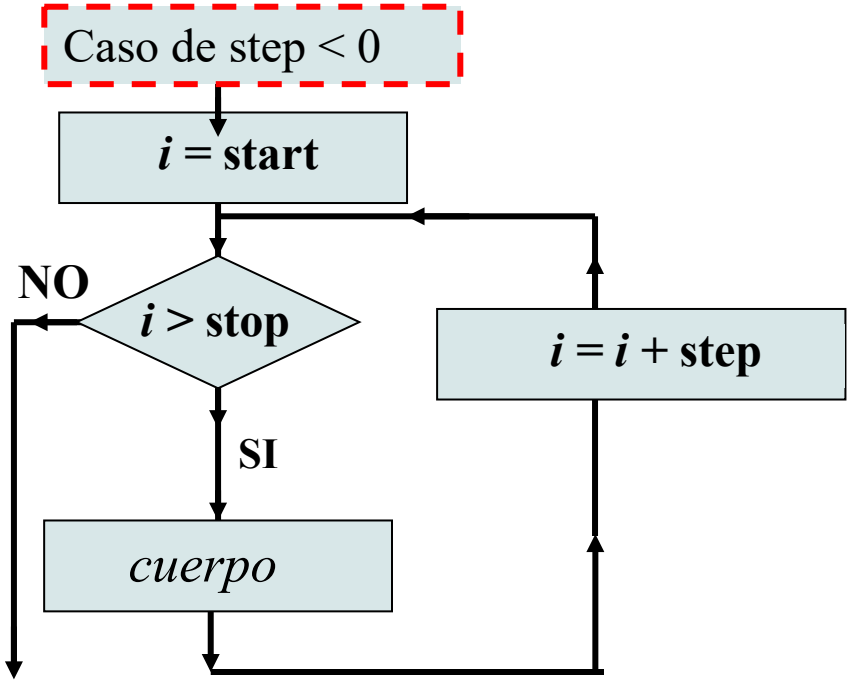
Es decir, i toma el valor 0 y se ejecuta una iteración, 1 y otra iteración,…, hasta el 99 que ejecuta la última iteración.

El índice i del for controla **cuántas** veces se ejecuta el cuerpo del bucle.

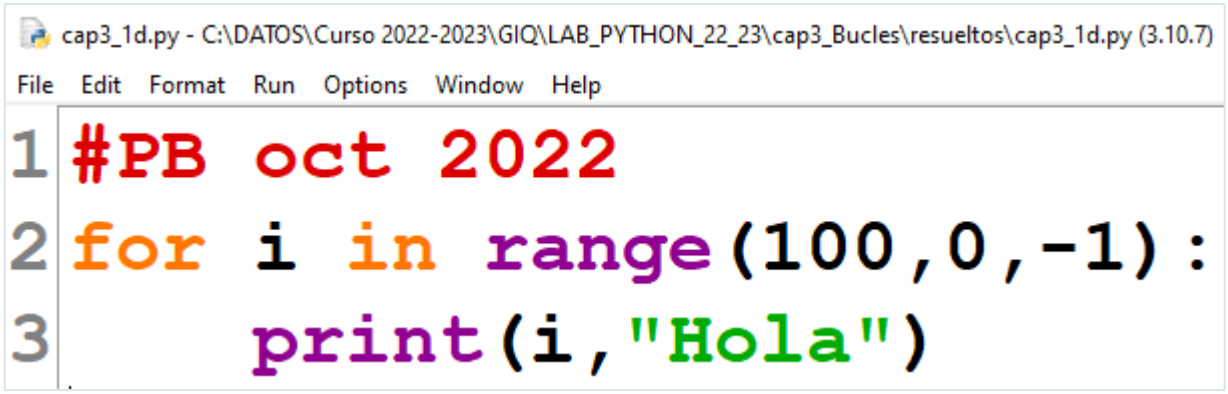
¿Cuántas posibilidades hay de cambiar los valores de los parámetros de **range** sin alterar el resultado de la ejecución del programa?

bucle for-in con step negativo

for *i* **in range (start, stop [, step]):**
cuerpo del bucle



bucle for-in con step negativo. Ejercicio 1d



range genera la serie: 100, 99, 98, …, 1
A continuación, i toma cada uno de los valores de esa serie, ejecutando el cuerpo para cada valor.

¿Qué ocurre si step es cero?

bucle for-in. Ejercicio 2

Sumar los números naturales desde el 1 hasta el 100, ambos incluidos.

cap3_2.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap3_Bucles\resueltos\cap3_2.py (3.10.7)

File Edit Format Run Options Window Help

```
1 #PB oct 2022
2 suma=0
3 for i in range(1,101):
4     suma=suma+i    #acumulador
5     print('i,suma ',i,suma)
6 print("Suma del 1 al 100: ", suma)
```

El índice del bucle **interviene** en el cuerpo del mismo.

¿Para qué sirve el print indentado?

Utiliza http://pythontutor.com/visualize.html#mode=edit para comprender mejor el camino de ejecución del bucle.

¿Qué resultado se obtiene si se cambia el cabecero del bucle por la instrucción: for i in range (100,0,-1):

Forma compacta de asignaciones

Sintaxis:
variable = *variable* **OPERADOR ARITMÉTICO** *expresión_aritmética*
variable **OPERADOR ARITMÉTICO** = *expresión_aritmética*

Operador aritmético	Descripción
+	Suma
-	Resta
*	Multiplicación
/	División
//	División Entera
%	Resto
**	Potencia

Generalmente, se usan en los bucles para redefinir o actualizar la *variable* en cada iteración. Ejemplo: las variables acumulador y contador.

Variables especiales en estructuras repetitivas I

- **Acumuladores**. Se utilizan para "acumular" resultados parciales en cada iteración.

Siempre se usan igual. Se inicializan antes de entrar en el bucle y aparecen dentro del mismo a la izquierda y derecha de una sentencia de asignación.

Para sumar:

```
acum = 0 #inicializar a cero
for-in o while…:
        acum+= expresión_numérica
```

Para multiplicar:

```
acum = 1 #inicializar a uno
for-in o while…:
        acum*= expresión_numérica
```

bucle for-in. Ejercicio 3

Sumar números en un intervalo dado, cuyas características (extremos y paso) se piden por teclado.

cap3_3.py - C:\DATOS\Curso 2024-2025\GIQ\LAB_PYTHON_24_25\cap3_Bucles\resueltos\cap3_3.py (3.12.6)

File Edit Format Run Options Window Help

```
1  #PB sep 2024
2  inf=int(input("extremo inferior: "))
3  sup=int(input("extremo superior: "))
4  paso=int(input("paso: "))
5  acum=0
6  for i in range(inf,sup+1, paso):
7      acum+=i
8  print("Suma desde",inf,'hasta',sup,'de',paso,'en',paso,'=',acum)
9  cadena="Suma desde {0} hasta {1} de {2} en {2} = {3}"
10 print(cadena.format(inf,sup,paso,acum))
```

Se muestran dos formas de escribir en pantalla los resultados.

El método **format** recibe una cadena y 4 argumentos cuyos valores se escriben en pantalla en las posiciones indicadas por las llaves en la cadena, en el orden indicado. Es decir, escribe el valor de inf en la posición {0} de la cadena, sup en la posición {1} de la cadena,…

bucle for-in. Ejercicio 4

Calcular el valor de π aplicando la fórmula:

π = 4* (1 – 1/3 + 1/5 – 1/7 ...) incluyendo hasta el término 1 / 99.

Construir las 4 versiones siguientes:

Ej4_a. Construir dos bucles for-in, uno para sumar los términos positivos de la serie y otro para los negativos. Restar ambos resultados y multiplicar por 4.

Ej4_b. Repetir el ejercicio usando un solo bucle for-in. Por ejemplo, el índice del bucle tomará los valores del denominador de la serie alternada y una variable almacenará el signo de cada término.

Ej4_c. Idem que b, pero usando el término general de la serie alternada.

Ej4_d. Suponer pi de math el valor exacto, ¿cuántos términos de la serie se necesitan para tener una precisión en el cálculo de 0.001?

bucle for-in. Ejercicios 4a, 4b, 4c

cap3_4a.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap3_Bucles\resueltos\cap3_4a.py (3.10.7)

File Edit Format Run Options Window Help

```
1 #PB oct 2022
2 pip=0
3 for n in range(1,100,4):
4     pip=pip+1/n
5 pin=0
6 for n in range(3,100,4):
7     pin=pin+1/n
8 pic=4*(pip-pin)
9 print("pi calculado",pic)
```

cap3_4b.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap3_Bucles\resueltos\cap3_4b.py (3.10.7)

File Edit Format Run Options Window Help

```
1 #PB oct 2022
2 pic=0
3 signo=1
4 for n in range(1,100,2):
5     pic=pic+1/n*signo
6     signo=-signo
7 pic=4*pic
8 print("pi calculado",pic)
```

cap3_4c.py - C:\DATOS\Curso 2024-2025\GIQ\LAB_PYTHON_24_25\ca

File Edit Format Run Options Window Help

```
1 #PB sep 2024
2 pic=0
3 for n in range(50):
4     pic=pic+1/(2*n+1)*(-1)**n
5 pic=4*pic
6 print("pi calculado",pic)
```

¿Cuál es más fácil de entender? ¿Cuál es más largo?

bucle for-in. Ejercicio 4d

cap3_4d.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap3_Bucles\resueltos\cap3_4d.py (3.10.7)

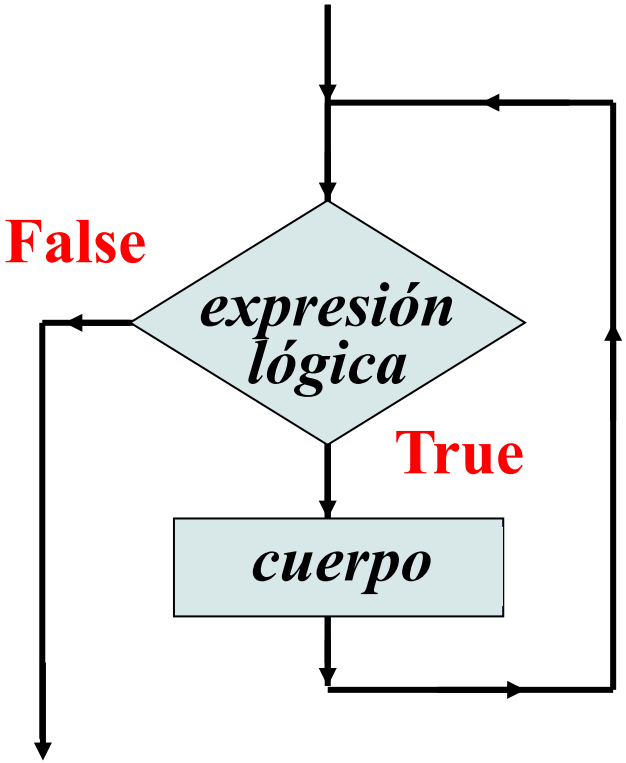
File Edit Format Run Options Window Help

```
1  #PB oct 2022
2  from math import pi,fabs
3  pic=0
4  n=0
5  while fabs(pi-4*pic)>1e-3:
6      pic+=1/(2*n+1)*(-1)**n
7      n+=1
8  pic=4*pic
9  print("pi calculado",pic)
10 print("Se necesitan ",n,'términos')
```

Bucle while I

En este tipo de bucles, el número de iteraciones es desconocido a priori. Mientras una condición sea cierta se ejecuta el cuerpo del bucle.

Sintaxis:



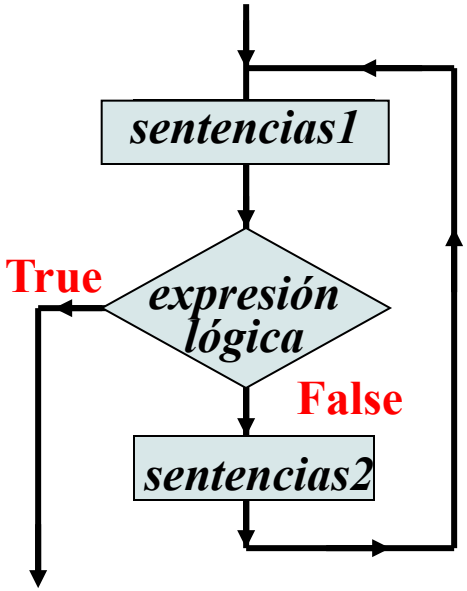
while *expresión lógica*:
sentencia_1
[sentencia_2]
...
} *cuerpo*

Para evitar bucles infinitos, la variable de la expresión lógica debe aparecer a la izquierda de una sentencia de asignación en el cuerpo. Es decir, se modifica en cada iteración. Este hecho permite que en alguna iteración, la expresión lógica sea falsa y se salga del bucle.

Bucle while II

En esta estructura ocurre lo contrario de la anterior.

Cuando una condición es cierta, se sale del bucle.



Sintaxis:

```
while True:
    [sentencias_1]
    ...
    if expresión_lógica: break
    [sentencias_2]
    ...
```

El bloque de sentencias con indentación se ejecuta indefinidamente mientras *expresión_lógica* sea falsa.

Cuando *expresión_lógica* es cierta se ejecuta la cláusula **break** que transfiere el control fuera del bucle **while**, a la primera sentencia no indentada por debajo.

break significa salir del bucle.

Si la expresión lógica nunca se hace cierta, el bucle es infinito!

break puede usarse tanto en bucles **while** como en **for-in**.

Ejemplos comparativos, for-in versus while

- Escribir 10 veces el mensaje Hola en pantalla:

```
for i in range(10):
    print("Hola")

i=0   #start en range
while i<10:  #stop
    print("Hola")
    i+=1 #step
```

- Sumar los números pares de 10 a 100 sin leer ningún dato por teclado.

```
suma=0
for i in range(10,101,2):
    suma+=i
print(suma)

suma=0
i=10
while i<101:
    suma+=i
    i+=2
print(suma)
```

Variables especiales en estructuras repetitivas II

- **Contadores**. Se utilizan para "contar" iteraciones.

Siempre se usan igual. Se inicializan antes de entrar en el bucle y aparecen dentro del mismo a la izquierda y derecha de una sentencia de asignación incrementándose una unidad.

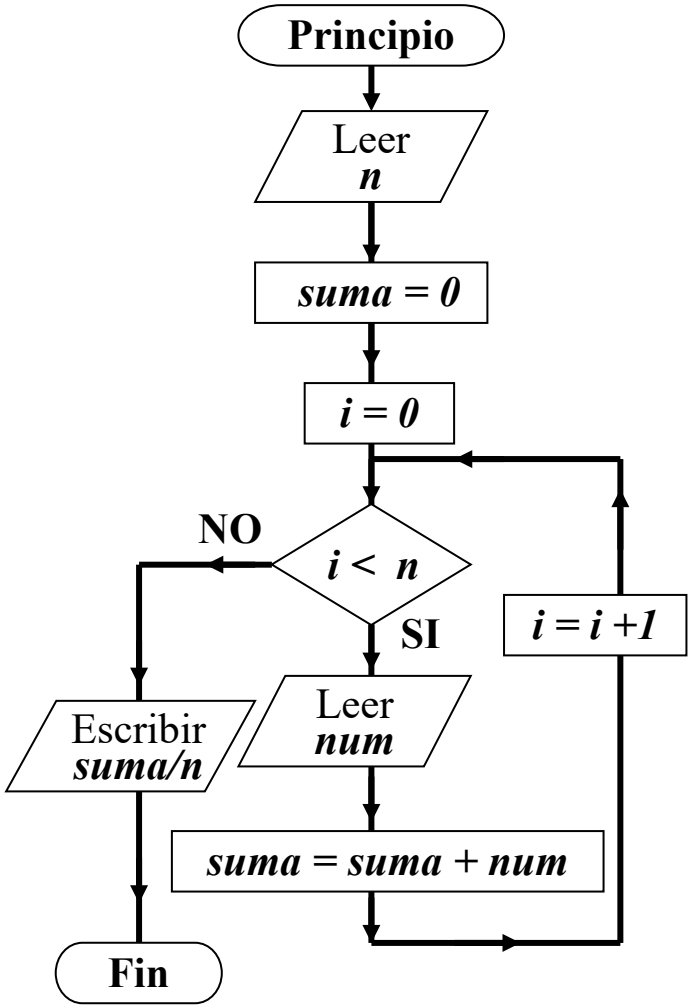
```
contador = 0 #inicializar a cero

for-in o while…:

        contador+= 1
```

Ejercicio 5a

Calcular la media de un conjunto de números. La cantidad de números se pide por teclado.



Ejercicio 5a

cap3_5a.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap3_Bucles\resueltos\cap3_5a.py (3.10.7)

File Edit Format Run Options Window Help

```
1 #PB oct 2022
2 suma=0
3 n=int(input("cuantos numeros son "))
4 for i in range(n):
5     num=float(input("Dame un numero "))
6     suma=suma+num
7 print("La media es",suma/n)
```

Ejercicio 5b

Calcular la media de un conjunto de números. El programa recoge tantos datos como el usuario quiera.

cap3_5b.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap3_Bucles\resueltos\cap3_5b.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB oct 2022
2  suma=0   #acumulador
3  otro_numero="si"
4  i=0   #contador
5  while otro_numero=="si" or otro_numero=="SI":
6      num=float(input("Dame un numero "))
7      suma+=num
8      i+=1
9      otro_numero=input("¿Otro número? si o no ")
10 print("La media es ",suma/i)
```

Bucles anidados

Un bucle puede estar contenido completamente dentro de otro bucle. En este caso, se dice que ambos bucles están *anidados*.

Cuando dos bucles están anidados, el bucle interno se ejecuta completamente para cada iteración del bucle externo. Si son bucles del tipo for-in, el índice del bucle interno toma todos los valores posibles para cada uno de los valores posibles del índice del bucle externo.

Los índices de bucles anidados deben tener identificadores distintos.

```
k=0
for i in range(4):
    for j in range(3):
        k=k+1
print(k)
```

Ejecuta este programa en pythontutor para entenderlo mejor.

¿Cuánto vale k en la sentencia print?

i	j	k
0	0	1
0	1	2
0	2	3
1	0	4
1	1	5
1	2	6
⋮	⋮	⋮
3	2	12

Ejercicio 6a

Calcular varias medias. La cantidad de medias **m** y de números para cada media **n** se piden por teclado.

cap3_6a.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap3_Bucles\resueltos\cap3_6a.py (3.10.7)

File Edit Format Run Options Window Help

```
1 #PB oct 2022
2 m=int(input("Cuantas medias "))
3 for j in range(m):
4     suma=0
5     n=int(input("cuantos numeros son "))
6     for i in range(n):
7         num=float(input("Dame un numero "))
8         suma=suma+num
9     print("La media es ",suma/n)
```

Ejercicios 5a versus 6a

cap3_5a.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap3_Bucles\resueltos\cap3_5a.py (3.10.7)

File Edit Format Run Options Window Help

```
#PB oct 2022
suma=0
n=int(input("cuantos numeros son "))
for i in range(n):
    num=float(input("Dame un numero "))
    suma=suma+num
print("La media es",suma/n)
```

cap3_6a.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap3_Bucles\resueltos\cap3_6a.py (3.10.7)

File Edit Format Run Options Window Help

```
#PB oct 2022
m=int(input("Cuantas medias "))
for j in range(m):
    suma=0
    n=int(input("cuantos numeros son "))
    for i in range(n):
        num=float(input("Dame un numero "))
        suma=suma+num
    print("La media es ",suma/n)
```

Ejercicio 6b

¿Cómo cambia el programa 6a si **n** y **m** son desconocidos, a priori?

cap3_6b.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap3_Bucles\resueltos\cap3_6b.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB oct 2022
2  otra_media='si'
3  while otra_media=="si" or otra_media=="SI":
4      suma=0  #acumulador
5      otro_numero="si"
6      i=0  #contador
7      while otro_numero=="si" or otro_numero=="SI":
8          num=float(input("Dame un numero "))
9          suma+=num
10         i+=1
11         otro_numero=input("Otro numero? si o no ")
12     print("La media es ",suma/i)
13     otra_media=input("Otra media? si o no ")
```

Ejercicios 5b versus 6b

cap3_5b.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap3_Bucles\resueltos\cap3_5b.py (3.10.7)

File Edit Format Run Options Window Help

```
#PB oct 2022
suma=0  #acumulador
otro_numero="si"
i=0  #contador
while otro_numero=="si" or otro_numero=="SI":
    num=float(input("Dame un numero "))
    suma+=num
    i+=1
    otro_numero=input("¿Otro número? si o no ")
print("La media es ",suma/i)
```

cap3_6b.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap3_Bucles\resueltos\cap3_6b.py (3.10.7)

File Edit Format Run Options Window Help

```
#PB oct 2022
otra_media='si'
while otra_media=="si" or otra_media=="SI":
    suma=0  #acumulador
    otro_numero="si"
    i=0  #contador
    while otro_numero=="si" or otro_numero=="SI":
        num=float(input("Dame un numero "))
        suma+=num
        i+=1
        otro_numero=input("Otro numero? si o no ")
    print("La media es ",suma/i)
    otra_media=input("Otra media? si o no ")
```

continue en un bucle. Ejercicio 7

continue es una palabra reservada de Python que sólo tiene sentido dentro de un bucle (for-in o while). Su ejecución significa ignorar las sentencias que haya por debajo de continue, pasando a la siguiente iteración.

Calcula y escribe en pantalla la cantidad de números positivos que hay en una lista dada de N elementos. El proceso se repite todas las veces que el usuario quiera.

cap3_7.py - C:\DATOS\Curso 2023-2024\GIQ\LAB_PYTHON_23_24\cap3_Bucles\resueltos\cap3_7.py (3.11.5)

File Edit Format Run Options Window Help

```
1  #PB oct 2023
2  continuar=True
3  while continuar:
4      n=int(input("cuantos números: "))
5      contaposi=0
6      for i in range(n):
7          num=float(input("Dame un número "))
8          if num<=0:
9              continue
10         contaposi=contaposi+1
11     cad="{0} positivos de los {1} leídos"
12     print(cad.format(contaposi,n))
13     continuar=int(input("¿Otra lista? \n1 sí, 0 no "))
```

¿Cómo evitar continue? Escribe otra versión del programa sin usarlo.

Evolución de un bucle a dos bucles anidados

Ejemplo:

Calcular el factorial de los números 3 al 6 y mostrar los resultados por pantalla, sin usar la librería **math**.

- Calcular el factorial de 3 con un for-in.
- ¿Cómo se calcula el factorial de 4? ¿y de 5? ¿y de 6?
- Crear un nuevo for-in que incluya al anterior, cuyo índice tome los valores 3, 4, 5, 6.
- ¿Cómo calcular los factoriales en un intervalo de n a m?

```
#Factorial de 3
factor=1
for i in range(3,1,-1):
    factor*=i
print(3,'!= ',factor)
```

```
#Factoriales de 3,4,5,6
for j in range(3,7):
    factor=1
    for i in range(j,1,-1):
        factor*=i
    print(j,'!= ',factor)
```

```
#Factoriales de n a m
n=int(input("n: "))
m=int(input("m: "))
for j in range(n,m+1):
    factor=1
    for i in range(j,1,-1):
        factor*=i
    print(j,'!= ',factor)
```

Bucles dentro de estructuras if y viceversa. Ejercicio 8

Es posible tener bucles dentro de estructuras **if** o tener estructuras **if** dentro de bucles.

Si un bucle se incluye dentro de una estructura **if**, el bucle debe estar completamente dentro de una sola *rama*.

Una estructura **if** dentro de un bucle debe acabar antes de acabar el bucle que la contiene.

Calcular el factorial de un número natural (que se pedirá por teclado). Si el número introducido es negativo escribir un mensaje avisando de que no existe su factorial.

c. Usar la función factorial de math.

cap3_8c.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap3_Bucles\resueltos\cap3_8c.py (3.10.7)

File Edit Format Run Options Window Help

```
1 #PB oct 2022
2 from math import factorial
3 num=int(input("Dame un número:"))
4 if num<0:
5     print("No existe el factorial de un negativo")
6 else:
7     print("El factorial de",num," es: ",factorial(num))
```

Ejercicio 8a, 8b

Calcular el factorial de un número natural (que se pedirá por teclado) usando un bucle for-in. Escribir los resultados parciales en cada iteración. Si el número introducido es negativo escribir un mensaje avisando de que no existe su factorial. Dos formas de controlar los negativos:

a. Estructura if.

b. Bucle while de validación del número.

cap3_8a.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap3_Bucles\resueltos\cap3_8a.py (3.10.7)

File Edit Format Run Options Window Help

```
#PB oct 2022
num=int(input("Dame un número: "))
if num<0:
    print("No existe el factorial de un negativo")
else:
    fact=1
    for i in range(num,1,-1):
        fact=fact*i
        print('i,fact',i,fact)
    print(num,"!: ",fact)
```

cap3_8b.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap3_Bucles\resueltos\cap3_8b.py (3.10.7)

File Edit Format Run Options Window Help

```
#PB oct 2022
while True:
    num=int(input("Dame un número: "))
    if num>=0: break
    print("No puede ser negativo")
    print("Vuelve a intentarlo")

fact=1
for i in range(num,1,-1):
    fact=fact*i
    print('i,fact ',i,fact)
print(num,"! ",fact)
```

Variables especiales en estructuras repetitivas III

- **Interruptores o switches**. Se utilizan para averiguar si ha ocurrido algo o no en un bucle. Sólo toman dos valores distintos, generalmente 0 (apagado) ó 1 (encendido). Finalizado el bucle, en función de su valor, se actúa de una forma u otra.

Siempre se usan igual. Se inicializan antes de entrar en el bucle, por ejemplo a cero, y dentro del mismo, si una condición es cierta, se cambia su valor a uno. Finalizado el bucle, en función del valor del interruptor, se ejecutan unas instrucciones u otras.

```
sw= 0
for-in o while…:
        if condición:
                sw= 1
                …
if sw==0:
        …
else:
        …
```

Ejercicio 9a

Programa que pida un número por teclado y diga si es primo o no.

a. Mostrando todos sus divisores.

cap3_9a.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap3_Bucles\resueltos\cap3_9a.py (3.10.7)

File Edit Format Run Options Window Help

```
1  num=int(input("Dame un numero "))
2  switch=0  #varible interruptor
3  for divisor in range(2,num):
4      resto=num%divisor
5      if resto==0:
6          print(divisor," es un divisor de ",num)
7          switch=1
8  if switch==0:
9      print(num,' es primo')
10 else:
11     print(num,' no es primo')
```

Ejercicio 9b

¿Cómo modificarías la versión a si no se escriben los divisores del número?

cap3_9b.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap3_Bucles\resueltos\cap3_9b.py (3.10.7)

File Edit Format Run Options Window Help

```
1  num=int(input("Dame un numero "))
2  switch=0  #varible interruptor
3  for divisor in range(2,num):
4      resto=num%divisor
5      if resto==0:
6          switch=1
7          break #provoca la salida del bucle
8  if switch==0:
9      print(num,' es primo')
10 else:
11     print(num,' no es primo')
```

Ejercicio 9c

Sin mostrar divisores, pero con un algoritmo más eficiente, interesante para números grandes. Ejemplo: 593 es primo.

"Un número n es primo si no es divisible entre ninguno de los siguientes: 2, 3, 4, …, $int(\sqrt{n})$".

cap3_9c.py - C:\DATOS\Curso 2023-2024\GIQ\LAB_PYTHON_23_24\cap3_Bucles\resueltos\cap3_9c.py (3.11.5)

File Edit Format Run Options Window Help

```
1 #PB Oct 2023
2 num=int(input("Dame un numero "))
3 divisor=2
4 while num%divisor!=0 and divisor<=num//divisor:
5     divisor+=1
6 if num%divisor!=0:
7     print(num,'es primo')
8 else:
9     print(num,'no es primo')
```