4.2.2. Un problema de ejemplo: cálculo de sumatorios

Ahora que ya hemos presentado lo fundamental de los bucles, vamos a resolver algunos problemas concretos. Empezaremos por un programa que calcula la suma de los 1000 primeros números, es decir, un programa que calcula el sumatorio

$$\sum_{i=1}^{1000} i,$$

o, lo que es lo mismo, el resultado de $1 + 2 + 3 + \cdots + 999 + 1000$.

Vamos paso a paso. La primera idea que suele venir a quienes aprenden a programar es reproducir la fórmula con una sola expresión Python, es decir:

sumatorio.py

```
1 sumatorio = 1 + 2 + 3 + ... + 999 + 1000
2 print sumatorio
```

Pero, obviamente, no funciona: los puntos suspensivos no significan nada para Python. Aunque una persona puede aplicar su intuición para deducir qué significan los puntos suspensivos en ese contexto, Python carece de intuición alguna: exige que todo se describa de forma precisa y rigurosa. Esa es la mayor dificultad de la programación: el nivel de detalle y precisión con el que hay que describir qué se quiere hacer.

Bien. Abordémoslo de otro modo. Vamos a intentar calcular el valor del sumatorio «acumulando» el valor de cada número en una variable. Analiza este otro programa (incompleto):

```
1 sumatorio = 0
2 sumatorio += 1
3 sumatorio += 2
4 sumatorio += 3
  ⋮
1000 sumatorio += 999
1001 sumatorio += 1000
1002 print sumatorio
```

Como programa no es el colmo de la elegancia. Fíjate en que, además, presenta una estructura casi repetitiva: las líneas de la 2 a la 1001 son todas de la forma

sumatorio += *número*

donde *número* va tomando todos los valores entre 1 y 1000. Ya que esa sentencia, con ligeras variaciones, se repite una y otra vez, vamos a tratar de utilizar un bucle. Empecemos construyendo un borrador incompleto que iremos refinando progresivamente:

sumatorio.py

```
1 sumatorio = 0
2 while condición:
3     sumatorio += número
4 print sumatorio
```

Hemos dicho que *número* ha de tomar todos los valores crecientes desde 1 hasta 1000. Podemos usar una variable que, una vez inicializada, vaya tomando valores sucesivos con cada iteración del bucle:

sumatorio.py

```
1 sumatorio = 0
2 i = 1
3 while condición:
4     sumatorio += i
5     i += 1
6 print sumatorio
```

Sólo resta indicar la condición con la que se decide si hemos de iterar de nuevo o, por el contrario, hemos de finalizar el bucle:

sumatorio_4.py — sumatorio.py

```
sumatorio = 0
i = 1
while i <= 1000:
    sumatorio += i
    i += 1
print sumatorio
```

EJERCICIOS

▶ 106 Estudia las diferencias entre el siguiente programa y el último que hemos estudiado. ¿Producen ambos el mismo resultado?

sumatorio_5.py — sumatorio.py

```
sumatorio = 0
i = 0
while i < 1000:
    i += 1
    sumatorio += i
print sumatorio
```

▶ 107 Diseña un programa que calcule

$$\sum_{i=n}^{m} i,$$

donde n y m son números enteros que deberá introducir el usuario por teclado.

▶ 108 Modifica el programa anterior para que si $n > m$, el programa no efectúe ningún cálculo y muestre por pantalla un mensaje que diga que n debe ser menor o igual que m.

▶ 109 Queremos hacer un programa que calcule el factorial de un número entero positivo. El factorial de n se denota con $n!$, pero no existe ningún operador Python que permita efectuar este cálculo directamente. Sabiendo que

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-1) \cdot n$$

y que $0! = 1$, haz un programa que pida el valor de n y muestre por pantalla el resultado de calcular $n!$.

▶ 110 El número de combinaciones que podemos formar tomando m elementos de un conjunto con n elementos es:

$$C_n^m = \binom{n}{m} = \frac{n!}{(n-m)!\,m!}.$$

Diseña un programa que pida el valor de n y m y calcule C_n^m. (Ten en cuenta que n ha de ser mayor o igual que m.)

(Puedes comprobar la validez de tu programa introduciendo los valores $n = 15$ y $m = 10$: el resultado es 3003.)