Módulos

Python es un lenguaje que se puede (y muchas veces debe) modularizar, es decir, dividir su código en distintos módulos reutilizables. De hecho, el propio núcleo de Python que instalamos en el sistema ya está modularizado, y podemos incorporar algunos de esos módulos a nuestros programas con la instrucción `import`.

1. Importando módulos de Python

Como decíamos, la instrucción `import` nos permite incorporar a nuestro programa módulos del núcleo de Python para poderlos utilizar. En el siguiente ejemplo importamos el módulo `sys` para acceder a los parámetros que se pasan al programa principal (`sys.argv`).

```
import sys

for i in range(1, len(sys.argv)):
    # Recorremos los parámetros quitando el 0 (ejecutable)
    print(sys.argv[i])
```

2. Descomponiendo nuestro código en módulos

A medida que el código de nuestro programa crece, puede ser necesario dividirlo en distintos ficheros fuentes. Al hacer esto, podemos emplear la instrucción `import` para "cargar" o utilizar unos módulos dentro de otros. Por ejemplo, supongamos que incluimos en un archivo llamado *modulo.py* el siguiente contenido:

```
pi = 3.141592

def sumar(a, b):
    return a + b
```

Si queremos utilizar estos elementos desde otro fichero, basta con importarlo:

```
import modulo

print(modulo.sumar(3, 4))
print(modulo.pi * 8)
```

Además, podemos utilizar una sintaxis alternativa para importar directamente unos elementos seleccionados, y así evitar tener que anteponer el nombre del módulo delante de esos elementos:

```
from modulo import pi, sumar

print(sumar(3, 4))
print(pi * 8)
```

> **Ejercicio 1:**
>
> Descompón el ejercicio 3 de esta sesión anterior en módulos, de forma que cada clase esté en un módulo, y mediante instrucciones `import` se incluyan donde se necesiten.