3.4. Legibilidad de los programas

Los programas que estamos diseñando son bastante sencillos. No ocupan más allá de tres o cuatro líneas y siempre presentan una misma estructura:

- Piden el valor de una serie de datos (mediante *raw_input*).
- Efectúan unos cálculos con ellos.
- Muestran el resultado de los cálculos (con **print**).

Estos programas son fáciles de leer y, en cierto modo, autoexplicativos.

Fíjate en este programa y trata de descifrar qué hace:

ilegible.py

```
1 h = float(raw_input('Dame h: '))
2 v = float(raw_input('y v: '))
3 z = h * v
4 print 'Resultado 1 %6.2f' % z
5 v = 2 * h + v + v
6 print 'Resultado 2 %6.2f' % v
```

Mmmm... no está muy claro, ¿verdad? Podemos entender qué hace el programa línea a línea, pero es difícil captar su propósito.

Ahora trata de leer este otro.

legible.py

```
1 print 'Programa para el cálculo del perímetro y el área de un rectángulo.'
2
3 altura = float(raw_input('Dame la altura (en metros): '))
4 anchura = float(raw_input('Dame la anchura (en metros): '))
5
6 area = altura * anchura
7 perimetro = 2 * altura + 2 * anchura
8
9 print 'El perímetro es de %6.2f metros.' % perimetro
10 print 'El área es de %6.2f metros cuadrados.' % area
```

Sencillo, ¿verdad? Hemos separado visualmente cuatro zonas con la ayuda de líneas en blanco. En la primera línea se anuncia el cometido del programa. En las dos siguientes líneas no blancas se pide el valor de dos datos y el nombre de las variables en los que los almacenamos ya sugiere qué son esos datos. A continuación, se efectúan unos cálculos. También en este caso el nombre de las variables ayuda a entender qué significan los resultados obtenidos. Finalmente, en las dos últimas líneas del programa se muestran los resultados por pantalla. *Evidentemente*, el programa pide la altura y la anchura de un rectángulo y calcula su perímetro y área, valores que muestra a continuación.

EJERCICIOS

▶ 45 Diseña un programa que solicite el radio de una circunferencia y muestre su área y perímetro con sólo 2 decimales.

3.4.1. Algunas claves para aumentar la legibilidad

¿Por qué uno de los programas ha resultado más sencillo de leer que el otro?

- `ilegible.py` usa nombres arbitrarios y breves para las variables, mientras que `legible.py` *utiliza identificadores representativos y tan largos como sea necesario*. El programador de `ilegible.py` pensaba más en teclear poco que en hacer comprensible el programa.

- `ilegible.py` no tiene una estructura clara: mezcla cálculos con impresión de resultados. En su lugar, `legible.py` *diferencia claramente zonas distintas del programa* (lectura de datos, realización de cálculos y visualización de resultados) y llega a usar marcas visuales como las líneas en blanco para separarlas. Probablemente el programador de `ilegible.py` escribía el programa conforme se le iban ocurriendo cosas. El programador de `legible.py` tenía claro qué iba a hacer desde el principio: planificó la estructura del programa.

- `ilegible.py` utiliza fórmulas poco frecuentes para realizar algunos de los cálculos: la forma en que calcula el perímetro es válida, pero poco ortodoxa. Por contra, `legible.py` *utiliza formas de expresión de los cálculos que son estándar*. El programador de `ilegible.py` debería haber pensado en los convenios a la hora de utilizar fórmulas.

- Los mensajes de `ilegible.py`, tanto al pedir datos como al mostrar resultados, son de pésima calidad. Un usuario que se enfrenta al programa por primera vez tendrá serios problemas para entender qué se le pide y qué se le muestra como resultado. El programa `legible.py` *emplea mensajes de entrada/salida muy informativos*. Seguro que el programador de `ilegible.py` pensaba que él sería el único usuario de su programa.

La legibilidad de los programas es clave para hacerlos prácticos. ¿Y por qué querría un programador leer programas ya escritos? Por varias razones. He aquí algunas:

- Es posible que el programa se escribiera hace unas semanas o meses (o incluso años) y ahora se desee modificar para extender su funcionalidad. Un programa legible nos permitirá ponernos mano a la obra rápidamente.

- Puede que el programa contenga errores de programación y deseemos detectarlos y corregirlos. Cuanto más legible sea el programa, más fácil y rápido será depurarlo.

- O puede que el programa lo haya escrito un programador de la empresa que ya no está trabajando en nuestro equipo. Si nos encargan trabajar sobre ese programa, nos gustaría que el mismo estuviera bien organizado y fuera fácilmente legible.[3]

Atenerte a la reglas usadas en `legible.py` será fundamental para hacer legibles tus programas.

[3] Si éste es tu libro de texto, míralo desde un lado «académicamente pragmático»: si tus programas han de ser evaluados por un profesor, ¿qué calificación obtendrás si le dificultas enormemente la lectura? `;-)`