4.1.4. Sentencias condicionales anidadas

Vamos a realizar un último refinamiento del programa. De momento, cuando a es 0 el programa muestra un mensaje que indica que la ecuación no tiene solución. Bueno, nosotros sabemos que esto no es cierto: si, además, b vale 0, entonces la ecuación tiene infinitas soluciones. Para que el programa dé una información correcta vamos a modificarlo de modo que, cuando a sea 0, muestre un mensaje u otro en función del valor de b:

primer_grado_12.py — primer_grado.py

```
1  a = float(raw_input('Valor de a: '))
2  b = float(raw_input('Valor de b: '))
3
4  if a != 0:
5      x = -b/a
6      print 'Solución: ', x
7  if a == 0:
8      if b != 0:
9          print 'La ecuación no tiene solución.'
10     if b == 0:
11         print 'La ecuación tiene infinitas soluciones.'
```

Fíjate en la indentación de las líneas. Las líneas 8–11 están más a la derecha que la línea 7. Ninguna de ellas se ejecutará a menos que la condición de la línea 7 se satisfaga. Más aún, la línea 9 está más a la derecha que la línea 8, por lo que su ejecución depende del resultado de la condición de dicha línea; y la ejecución de la línea 11 depende de la satisfacción de la condición de la línea 10. Recuerda que *en los programas Python la indentación determina de qué sentencia depende cada* bloque *de sentencias*.

Pues bien, acabamos de presentar una nueva idea muy potente: las estructuras de control pueden *anidarse*, es decir, aparecer unas «dentro» de otras. Esto no ha hecho más que empezar.

EJERCICIOS

▶ 57 Indica qué líneas del último programa (y en qué orden) se ejecutarán para cada uno de los siguientes casos:

a) $a = 2$ y $b = 6$. b) $a = 0$ y $b = 3$. c) $a = 0$ y $b = -3$. d) $a = 0$ y $b = 0$.

▶ 58 Diseña un programa que lea un número flotante por teclado y muestre por pantalla el mensaje «`El número es negativo.`» sólo si el número es menor que cero.

▶ 59 Diseña un programa que lea un número flotante por teclado y muestre por pantalla el mensaje «`El número es positivo.`» sólo si el número es mayor o *igual* que cero.

▶ 60 Diseña un programa que lea la edad de dos personas y diga quién es más joven, la primera o la segunda. Ten en cuenta que ambas pueden tener la misma edad. En tal caso, hazlo saber con un mensaje adecuado.

▶ 61 Diseña un programa que lea un carácter de teclado y muestre por pantalla el mensaje «Es paréntesis» sólo si el carácter leído es un paréntesis abierto o cerrado.

▶ 62 Indica en cada uno de los siguientes programas qué valores en las respectivas entradas provocan la aparición de los distintos mensajes. Piensa primero la solución y comprueba luego que es correcta ayudándote con el ordenador.

a)

misterio_3.py — misterio.py

```
letra = raw_input('Dame una letra minúscula: ')

if letra <= 'k':
    print 'Es de las primeras del alfabeto'
if letra >= 'l':
    print 'Es de las últimas del alfabeto'
```

b)

misterio_4.py — misterio.py

```
from math import ceil # ceil redondea al alza.

grados = float(raw_input('Dame un ángulo (en grados): '))

cuadrante = int(ceil(grados) % 360) / 90
if cuadrante == 0:
    print 'primer cuadrante'
if cuadrante == 1:
    print 'segundo cuadrante'
if cuadrante == 2:
    print 'tercer cuadrante'
if cuadrante == 3:
    print 'cuarto cuadrante'
```

▶ 63 ¿Qué mostrará por pantalla el siguiente programa?

comparaciones.py — comparaciones.py

```
if 14 < 120:
    print 'Primer saludo'
if '14' < '120':
    print 'Segundo saludo'
```

Por lo visto hasta el momento podemos comparar valores numéricos con valores numéricos y cadenas con cadenas. Tanto los valores numéricos como las cadenas pueden ser el resultado de una expresión que aparezca explícitamente en la propia comparación. Por ejemplo, para saber si el producto de dos números enteros es igual a 100, podemos utilizar este programa:

compara_expresiones.py — compara_expresiones.py

```
n = int(raw_input('Dame un número: '))
m = int(raw_input('Dame otro número: '))

if n * m == 100:
    print 'El producto %d * %d es igual a 100' % (n, m)
if n * m != 100:
    print 'El producto %d * %d es distinto de 100' % (n, m)
```

EJERCICIOS

▶ 64 Diseña un programa que, dado un número entero, muestre por pantalla el mensaje «`El número es par.`» cuando el número sea par y el mensaje «`El número es impar.`» cuando sea impar.

(Una pista: un número es par si el resto de dividirlo por 2 es 0, e impar en caso contrario.)

▶ 65 Diseña un programa que, dado un número entero, determine si éste es el doble de un número impar. (Ejemplo: 14 es el doble de 7, que es impar.)

▶ 66 Diseña un programa que, dados dos números enteros, muestre por pantalla uno de estos mensajes: «`El segundo es el cuadrado exacto del primero.`», «`El segundo es menor que el cuadrado del primero.`» o «`El segundo es mayor que el cuadrado del primero.`», dependiendo de la verificación de la condición correspondiente al significado de cada mensaje.

▶ 67 Un capital de C euros a un interés del x por cien anual durante n años se convierte en $C \cdot (1 + x/100)^n$ euros. Diseña un programa Python que solicite la cantidad C y el interés x y calcule el capital final *sólo si x es una cantidad positiva*.

▶ 68 Realiza un programa que calcule el desglose en billetes y monedas de una cantidad exacta de euros. Hay billetes de 500, 200, 100, 50, 20, 10 y 5 € y monedas de 2 y 1 €.

Por ejemplo, si deseamos conocer el desglose de 434 €, el programa mostrará por pantalla el siguiente resultado:

```
2 billetes de 200 euros.
1 billete de 20 euros.
1 billete de 10 euros.
2 monedas de 2 euros.
```

(¿Que cómo se efectúa el desglose? Muy fácil. Empieza por calcular la división entera entre la cantidad y 500 (el valor de la mayor moneda): 434 entre 500 da 0, así que no hay billetes de 500 € en el desglose; divide a continuación la cantidad 434 entre 200, cabe a 2 y sobran 34, así que en el desglose hay 2 billetes de 200 €; dividimos a continuación 34 entre 100 y vemos que no hay ningún billete de 100 € en el desglose (cabe a 0); como el resto de la última división es 34, pasamos a dividir 34 entre 20 y vemos que el desglose incluye un billete de 20 € y aún nos faltan 14 € por desglosar...)

4.1.5. Otro ejemplo: resolución de ecuaciones de segundo grado

Para afianzar los conceptos presentados (y aprender alguno nuevo), vamos a presentar otro ejemplo. En esta ocasión vamos a resolver ecuaciones de segundo grado, que son de la forma

$$ax^2 + bx + c = 0.$$

¿Cuáles son los datos del problema? Los coeficientes a, b y c. ¿Qué deseamos calcular? Los valores de x que hacen cierta la ecuación. Dichos valores son:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad \text{y} \qquad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}.$$

Un programa directo para este cálculo es:

segundo_grado_11.py — segundo_grado.py

```
1  from math import sqrt  # sqrt calcula la raíz cuadrada.
2
3  a = float(raw_input('Valor de a: '))
4  b = float(raw_input('Valor de b: '))
5  c = float(raw_input('Valor de c: '))
6
7  x1 = (-b + sqrt(b**2 - 4*a*c)) / (2 * a)
8  x2 = (-b - sqrt(b**2 - 4*a*c)) / (2 * a)
9
10 print 'Soluciones de la ecuación: x1=%4.3f y x2=%4.3f' % (x1, x2)
```

Ejecutemos el programa:

```
Valor de a: 2
Valor de b: 7
Valor de c: 2
Soluciones de la ecuación: x1=-0.314 y x2=-3.186
```

Un problema evidente de nuestro programa es la división por cero que tiene lugar cuando a vale 0 (pues entonces el denominador, $2a$, es nulo). Tratemos de evitar el problema de la división por cero del mismo modo que antes, pero mostrando un mensaje distinto, pues cuando a vale 0 la ecuación no es de segundo grado, sino de primer grado.

segundo_grado_12.py **segundo_grado.py**

```
1  from math import sqrt
2
3  a = float(raw_input('Valor de a: '))
4  b = float(raw_input('Valor de b: '))
5  c = float(raw_input('Valor de c: '))
6
7  if a != 0:
8      x1 = (-b + sqrt(b**2 - 4*a*c)) / (2 * a)
9      x2 = (-b - sqrt(b**2 - 4*a*c)) / (2 * a)
10     print 'Soluciones de la ecuación: x1=%4.3f y x2=%4.3f' % (x1, x2)
11 if a == 0:
12     print 'No es una ecuación de segundo grado.'
```