4.1.6. En caso contrario (else)

Fíjate en que tanto en el ejemplo que estamos desarrollando ahora como en el anterior hemos recurrido a sentencias condicionales que conducen a ejecutar una acción si se cumple una condición y a ejecutar otra si esa misma condición no se cumple:

```
if condición:
    acciones
if condición contraria:
    otras acciones
```

Este tipo de combinación es muy frecuente, hasta el punto de que se ha incorporado al lenguaje de programación una forma abreviada que significa lo mismo:

```
if condición:
    acciones
else:
    otras acciones
```

La palabra «else» significa, en inglés, «si no» o «en caso contrario». Es muy importante que respetes la indentación: las acciones siempre un poco a la derecha, y el **if** y el **else**, alineados en la misma columna.

segundo_grado_13.py — segundo_grado.py

```
from math import sqrt

a = float(raw_input('Valor de a: '))
b = float(raw_input('Valor de b: '))
c = float(raw_input('Valor de c: '))

if a != 0:
    x1 = (-b + sqrt(b**2 - 4*a*c)) / (2 * a)
    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 else:
12 |   print 'No␣es␣una␣ecuación␣de␣segundo␣grado.'
```

El programa no acaba de estar bien. Es verdad que cuando a vale 0, la ecuación es de primer grado, pero, aunque sabemos resolverla, no lo estamos haciendo. Sería mucho mejor si, en ese caso, el programa nos ofreciera la solución:

segundo_grado_14.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 else:
12 |   x = -c / b
13 |   print 'Solución␣de␣la␣ecuación:␣x=%4.3f' % x
```

Mmmm... aún hay un problema: ¿Qué pasa si a vale 0 y b también vale 0? La secuencia de líneas que se ejecutarán será: 1, 2, 3, 4, 5, 6, 7, 11 y 12. De la línea 12 no pasará porque se producirá una división por cero.

```
Valor de a: 0
Valor de b: 0
Valor de c: 2
Traceback (innermost last):
  File 'segundo_grado.py', line 12, in ?
    x = -c / b
ZeroDivisionError: float division
```

¿Cómo evitar este nuevo error? Muy sencillo, añadiendo nuevos controles con la sentencia **if**, tal y como hicimos para resolver correctamente una ecuación de primer grado:

segundo_grado_15.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 else:
12 |   if b != 0:
13 |   |   x = -c / b
14 |   |   print 'Solución␣de␣la␣ecuación:␣x=%4.3f' % x
15 |   else:
16 |   |   if c!= 0:
17 |   |   |   print 'La␣ecuación␣no␣tiene␣solución.'
18 |   |   else:
19 |   |   |   print 'La␣ecuación␣tiene␣infinitas␣soluciones.'
```

Es muy importante que te fijes en que las líneas 12–19 presentan una indentación tal que *todas ellas* dependen del **else** de la línea 11. Las líneas 13 y 14 dependen del **if** de la línea 12, y las líneas 16–19 dependen del **else** de la línea 15. Estudia bien el programa: aparecen sentencias condicionales anidadas en otras sentencias condicionales que, a su vez, están anidadas. ¿Complicado? No tanto. Los principios que aplicamos son siempre los mismos. Si analizas el programa y lo estudias por partes, verás que no es *tan* difícil de entender. Pero quizá lo verdaderamente difícil no sea entender programas con bastantes niveles de anidamiento, sino diseñarlos.

EJERCICIOS

▶ 69 ¿Hay alguna diferencia entre el programa anterior y este otro cuando los ejecutamos?

segundo_grado_16.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      if b == 0:
9          if c == 0:
10             print 'La ecuación tiene infinitas soluciones.'
11         else:
12             print 'La ecuación no tiene solución.'
13     else:
14         x = -c / b
15         print 'Solución de la ecuación: x=%4.3f' % x
16 else:
17     x1 = (-b + sqrt(b**2 - 4*a*c)) / (2 * a)
18     x2 = (-b - sqrt(b**2 - 4*a*c)) / (2 * a)
19     print 'Soluciones de la ecuación: x1=%4.3f y x2=%4.3f' % (x1, x2)
```

▶ 70 ¿Hay alguna diferencia entre el programa anterior y este otro cuando los ejecutamos?

segundo_grado_17.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 and b == 0 and c == 0:
8      print 'La ecuación tiene infinitas soluciones.'
9  else:
10     if a == 0 and b == 0:
11         print 'La ecuación no tiene solución.'
12     else:
13         if a == 0:
14             x = -c / b
15             print 'Solución de la ecuación: x=%4.3f' % x
16         else:
17             x1 = (-b + sqrt(b**2 - 4*a*c)) / (2 * a)
18             x2 = (-b - sqrt(b**2 - 4*a*c)) / (2 * a)
19             print 'Soluciones de la ecuación: x1=%4.3f y x2=%4.3f' % (x1, x2)
```

▶ 71 Ejecuta paso a paso, con ayuda del entorno de depuración de PythonG, el programa del ejercicio anterior.

▶ 72 Diseña un programa Python que lea un carácter cualquiera desde el teclado, y muestre el mensaje «`Es una MAYÚSCULA`» cuando el carácter sea una letra mayúscula y el mensaje «`Es una MINÚSCULA`» cuando sea una minúscula. En cualquier otro caso, no mostrará mensaje alguno. (Considera únicamente letras del alfabeto inglés.) Pista: aunque parezca una obviedad, recuerda que una letra es minúscula si está entre la `'a'` y la `'z'`, y mayúscula si está entre la `'A'` y la `'Z'`.

▶ 73 Amplía la solución al ejercicio anterior para que cuando el carácter introducido no sea una letra muestre el mensaje «`No es una letra`». (Nota: no te preocupes por las letras eñe, ce cedilla, vocales acentuadas, etc.)

▶ 74 Amplía el programa del ejercicio anterior para que pueda identificar las letras eñe minúscula y mayúscula.

▶ 75 Modifica el programa que propusiste como solución al ejercicio 66 sustituyendo todas las condiciones que sea posible por cláusulas **else** de condiciones anteriores.