Selección de instrucciones

Juan Álvarez Rubio

Hasta ahora las instrucciones de los programas y de las funciones se han ejecutado secuencialmente, es decir, una tras otra en forma incondicional. Sin embargo, se sabe que el computador tiene la capacidad de elegir o seleccionar entre alternativas. Por ejemplo, el siguiente programa simula el lanzamiento de una moneda al aire escribiendo la palabra “cara” o la palabra “sello”:

```
#cara o sello
from random import *
n=randint(1,2)
if n==1:
    moneda='cara'
else:
    moneda='sello'
print(moneda)
```

Inicialmente, el programa genera un número entero al azar entre 1 y 2 y lo asigna a la variable n. A continuación, **si (if)** n es igual a 1, entonces se asigna el string ‘cara’ a la variable moneda. **Si no (else)**, es decir, si n es distinto de 1, entonces la variable moneda toma el valor ‘sello’. Finalmente, y después de ejecutar una de las dos instrucciones de asignación, el programa escribe el valor de la variable moneda.

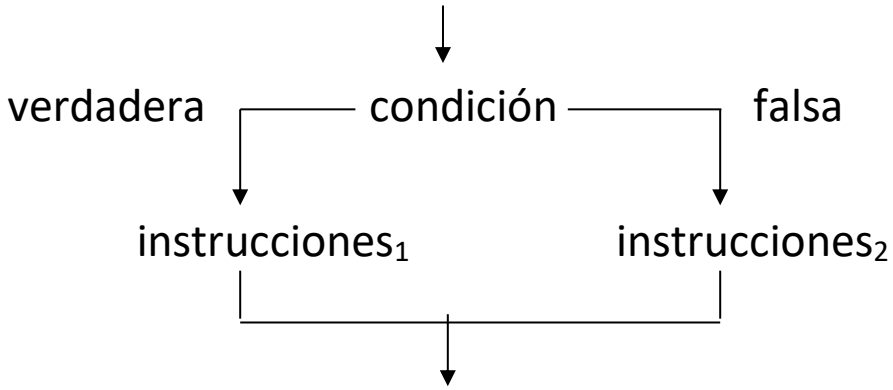
Sintaxis y semántica de la instrucción if

El ejemplo anterior permite deducir que la forma general de la instrucción if es la siguiente:

```
if condición:
  instrucciones1
else:
  instrucciones2
```

La sintaxis de la instrucción exige que las palabras if y else estén en el mismo margen y que las instrucciones$_1$ e instrucciones$_2$ estén indentadas y alineadas, de lo contrario se producirá un error: “IndentationError: expected an indented block”. Al respecto, después de ”if condición:” y de “else:” puede haber una o más instrucciones. En general, en el lenguaje Python, después de dos puntos (:) las instrucciones deben escribirse en la siguiente línea e indentadas.

La semántica de la instrucción se deduce del significado de las palabras if y else. Es decir, si (if) la condición se cumple, es decir si es verdadera, entonces se ejecutarán las instrucciones$_1$. De lo contrario (else), si la condición no se cumple, es decir si es falsa, entonces se ejecutarán las instrucciones$_2$. Gráficamente, el significado de la instrucción if se puede visualizar de la siguiente manera:



En otras palabras, cuando el computador se encuentra con la instrucción if, se evalúa la condición para comprobar si se cumple o no y elije ejecutar las instrucciones$_1$ o las instrucciones$_2$ respectivamente. Después de ejecutar cualquiera de ellas, el flujo continúa con la instrucción que está a continuación.

La condición (o expresión condicional) también tiene reglas de sintaxis. Por ejemplo, las condiciones n==1 y b**2>=4*a*c permiten inferir que una condición consiste de dos expresiones separadas por un operador de comparación. Los operadores de relación o de comparación son los siguientes: ==, <, >, <=, >=, !=. Para comparar por igualdad se utiliza == para no confundirlo con el = de la instrucción de asignación. Por otra parte, <= (menor o igual), >= (mayor o igual) y != (distinto) se escriben hacia el lado y en dos posiciones consecutivas (sin espacios entre ellas) y no en una posición como los símbolos matemáticos ≤, ≥ y ≠.

La semántica de una condición establece que primero hay que evaluar las dos expresiones, por ejemplo, b**2 y 4*a*c. En seguida se comparan los resultados de las expresiones para determinar si la comparación se cumple o no. Si la condición se cumple, por ejemplo, si b**2 es mayor o igual que 4*a*c, entonces la condición devuelve un resultado verdadero (valor True). Y si no se cumple entrega un resultado falso (valor False). Y los valores True o False determinan si la instrucción if ejecuta las instrucciones$_1$ o las instrucciones$_2$.

Instrucción if en una función

Una función puede contener una instrucción if, lo que significa que el resultado que entrega depende de una condición. Por ejemplo, una función que recibe dos números como parámetros y entrega el mayor de ellos se puede escribir de la siguiente manera siguiendo estrictamente la receta de diseño:

```
#mayor: num num -> num
#mayor valor entre x e y
#ejs: mayor(3,2)->3, mayor(2,3)->3, mayor(3,3)->3
def mayor(x,y):
```

```
  if x > y :
    m = x
  else:
    m = y
  return m
assert mayor(3,2)==3 #mayor es el primero
assert mayor(2,3)==3 #mayor es el segundo
assert mayor(3,3)==3 #dos valores iguales
```

La función debe probarse al menos en los dos casos posibles: que el primer parámetro sea mayor que el segundo parámetro, o que no sea mayor, es decir que el primer parámetro sea menor o igual que el segundo parámetro. Con el propósito de probar exhaustivamente la función, se incluyeron instrucciones assert para probar los tres casos posibles.

Otra forma de escribir la función es la siguiente:

```
def mayor(x,y):
  if x > y :
    return x
  else:
    return y
```

En este caso cada rama de la instrucción if es una instrucción return, lo que significa que la función termina devolviendo el valor indicado (x o y) al lugar donde se invocó a la función. De hecho, en la función mayor no hay ninguna instrucción después de la instrucción if.

La función mayor también podría invocarse con strings. Por ejemplo, mayor("pi","paz") y mayor("paz","pi") entregan como resultado "pi". Los strings se comparan "alfabéticamente", por lo tanto "pi" es mayor que "paz" puesto que, si bien los dos comienzan con la letra "p", sin embargo, la segunda letra ("i") es mayor que la "a" y por lo tanto "pi" es mayor que "paz" aunque tenga menos letras. Curiosamente, mayor("Pi","pi") entrega "pi" porque, por una razón histórica, todas las letras minúsculas son mayores que las mayúsculas según la convención para la representación interna de los caracteres.

Casos especiales

La sintaxis de la instrucción if permite que en cada rama puede haber otra instrucción if. Por ejemplo, una función que entregue el mayor valor entre sus tres parámetros podría escribirse de la siguiente manera:

```
#mayor: num num num -> num
#mayor valor entre x, y, z
#ej: mayor(4,2,3), mayor(3,4,2), mayor(3,2,4)->4
def mayor(x,y,z):
  if x > y :
```

```
    if x > z :
      return x
    else:
      return z
  else:
    if y > z :
      return y
    else:
      return z
assert mayor(4,2,3)==4 #primero es el mayor
assert mayor(3,4,2)==4 #segundo es el mayor
assert mayor(3,2,4)==4 #tercero es el mayor
```

La función debe probarse al menos para los tres casos posibles: que el mayor sea el primero, el segundo o el tercero de los parámetros. Respecto de la indentación, se han usado márgenes de dos espacios. Al ingresar el programa usando un editor para Python se dejan automáticamente cuatro espacios.

La instrucción if permite omitir la rama else, es decir si la condición es falsa no se ejecutan instrucciones y se sigue con la instrucción siguiente. Por ejemplo, la función para entregar el mayor valor entre tres parámetros podría escribirse en la siguiente forma:

```
def mayor(x,y,z):
  m=x        #suposición: 1° es el mayor
  if y > m:  #si 2° es mas grande
    m=y      #  entonces 2° es el mayor
  if z > m:  #si 3° es mas grande
    m=z      #  entonces 3° es el mayor
  return m
```

La función es mucho más breve que la anterior: 6 líneas en lugar de 10 líneas de la solución anterior. En caso que después del if exista una sola instrucción se puede usar otra indentación que acorta aún más el tamaño de la función:

```
def mayor(x,y,z):
  m = x
  if y > m: m=y
  if z > m: m=z
  return m
```

Una ventaja de esta última solución es que al agregar un cuarto parámetro, para determinar el mayor entre 4 valores, entonces bastaría añadir una sola línea con una instrucción if. Cabe recordar que existe la función predefinida max que acepta dos o más argumentos y entrega el de mayor valor. Análogamente existe la función min que entrega el menor valor.

Selección entre múltiples alternativas

La instrucción if permite seleccionar entre más de dos alternativas. Por ejemplo, para simular una jugada de "cachipún" se puede escribir:

```
#piedra, papel o tijeras
from random import *
n=randint(1,3)
if n==1:
    jugada="piedra"
elif n==2:
    jugada="papel"
else:
    jugada="tijeras"
print(jugada)
```

En este caso el programa eligió entre tres alternativas, pero pueden ser más. Por ejemplo, la siguiente es una función que lee una nota entre 1.0 y 7.0 y la expresa en forma cualitativa:

```
#nota cualitativa
nota=float(input("nota?"))
if nota>7:
    print("máximo 7")
elif nota>=5.5:
    print("bueno")
elif nota>=4:
    print("regular")
elif nota>=1:
    print("malo")
else:
    print("mínimo
```

En resumen, la sintaxis y la semántica de la instrucción if es la siguiente:

```
  if condicion1:   #una vez
    instrucciones1
  elif condicion2: #cero o más veces
    instrucciones2
  …
  else:            #opcional
    instrucciones
```

La sintaxis exige que las palabras if, elif y else deben estar alineadas y las instrucciones asociadas a cada caso deben estar indentadas y alineadas. La opción `elif condición`$_i$`:instrucciones`$_i$ puede aparecer cero o más veces. La opción `else:instrucciones` puede aparecer o no.

La semántica establece que se evalúan las condiciones en orden de aparición: si la condición$_i$ es verdadera, entonces se ejecutan las instrucciones$_i$ y después se continúa con la instrucción siguiente a if (a menos que instrucciones$_i$ termine con una instrucción return). Si ninguna de las condiciones resulta verdadera se ejecutan las instrucciones después de else. Posteriormente, el programa continúa con la instrucción siguiente a la instrucción if.

Problema: resolver una ecuación de segundo grado

Un uso de la instrucción if se puede apreciar al resolver una ecuación de segundo grado $ax^2 + bx + c = 0$. El programa debe leer los coeficientes a, b y c y escribir los valores de x que satisfacen la ecuación. En caso que la ecuación tenga dos soluciones reales, estas se deben calcular con la fórmula $x = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$

```
from math import *
print("resolver ax^2 + bx + c = 0")
a=float(input("a?"))
b=float(input("b?"))
c=float(input("c?"))
if a==0:
    if b==0:
        print("ecuacion incorrecta")
    else:
        print("x=",-c/b)
else:
    d=b**2-4*a*c #discriminante
    if d==0:
        print("x=",-b/(2*a))
    elif d>0:
        r=sqrt(d)  #raiz de discrimimante
        print("x=",(-b+r)/(2*a),(-b-r)/(2*a))
    else:
        print("dos raices complejas")
```

El programa tiene una instrucción if principal que distingue entre una ecuación de primer o segundo grado. Si a es cero, entonces una instrucción if encajonada resuelve la ecuación bx+c=0. Por otra parte, si a es distinto de cero, otra instrucción if encajonada distingue los tres casos posibles de acuerdo al valor del discriminante: una raíz real, dos raíces reales o dos raíces complejas.