

Guía de Ejercicios 3: Condicionales y tipo bool

Objetivos:

- Familiarizarse con el tipo de datos bool y la lógica proposicional.
- Ejercitar el uso de condicionales (if/elif/else) como herramienta para ejecutar selectivamente bloques de código, según el estado actual del programa.

Ejercicio 1. Evaluación de expresiones.

- (a) Para cada una de las siguientes expresiones, determinar su tipo y evaluarla a mano:

- | | |
|---------------------|-----------------------------------------------------|
| (I) True | (V) $1 > 0$ or not ('a'=='b') |
| (II) not False | (VI) $5.6 > 2.0$ and len('hola') < 2 |
| (III) $1 > 0$ | (VII) $5.6 > 2.0$ or len('hola') < 2 |
| (IV) not ('a'=='b') | (VIII) $\text{int}('3') - 1 == \text{len}('x' * 2)$ |

- (b) Revisar las respuestas del punto anterior, evaluando ahora las expresiones en una consola ipython y averiguando su tipo con la operación type().

Ejercicio 2. Condicionales.

- (a) Sean los siguientes programas, donde p y q son variables de tipo bool:

```

1 # programa I
2 if p:
3     if q:
4         print('A')
5     else:
6         print('B')
```

```

1 # programa II
2 if p:
3     if q:
4         print('A')
5     else:
6         print('B')
```

¿Qué diferencia hay entre los programas I y II? Para las 4 combinaciones de valores que pueden tener p y q, ¿qué imprime cada programa por pantalla?

- (b) Completar los espacios en blanco, de modo que los programas III y IV se comporten igual a los programas I y II, respectivamente, para todos los posibles valores de p y q.

```

1 # programa III
2 if ____:
3     print('A')
4 elif ____:
5     print('B')
```

```

1 # programa IV
2 if ____:
3     print('A')
4 elif ____:
5     print('B')
```

Evaluación de cortocircuito de expresiones lógicas

En lenguajes como Python, los operadores `and` y `or` no son conmutativos. Las expresiones lógicas se evalúan de izquierda a derecha, y a veces se interrumpe la evaluación cuando ya se tiene la información suficiente para saber el resultado:

- `False and EXP` → `False`
- `True or EXP` → `True`

En estos dos casos, la expresión lógica `EXP` no se llega a evaluar, porque con lo visto hasta ese momento (`False and ...`, o bien `True or ...`) ya alcanza para saber el resultado de la evaluación. A este comportamiento se lo conoce como **evaluación de cortocircuito**, que puede ser útil para escribir código más simple y claro.

Ejercicio 3. Para cada una de las siguientes expresiones, determinar su tipo y evaluarla (primero a mano, luego en la consola `ipython`):

- | | |
|------------------------------------|------------------------------------|
| (a) <code>True and (1 / 0)</code> | (e) <code>(1 / 0) and True</code> |
| (b) <code>True or (1 / 0)</code> | (f) <code>(1 / 0) or True</code> |
| (c) <code>False and (1 / 0)</code> | (g) <code>(1 / 0) and False</code> |
| (d) <code>False or (1 / 0)</code> | (h) <code>(1 / 0) or False</code> |

Ejercicio 4. Sea la función `f` definida de la siguiente manera:

```
1 def f(s:str) -> str:
2     if len(s)>0 and s[0]=='A':
3         return 'Eureka'
4     else:
5         return 'Me aburro'
```

(a) ¿Qué devuelven estas invocaciones? (Pensar las respuestas.)

- (I) `f('Algoritmos')`
- (II) `f('zzz')`
- (III) `f('')`

(b) Repetir el punto anterior, pero considerando esta definición de `f`:

```
1 def f(s:str) -> str:
2     if s[0]=='A' and len(s)>0:
3         return 'Eureka'
4     else:
5         return 'Me aburro'
```

Lógica proposicional

Ejercicio 5. Representar los siguientes enunciados en lógica proposicional, usando las variables indicadas:

- (a) Si p : “María aprobó Plástica”, q : “María aprobó Química”:
- (I) “María aprobó Plástica y Química”.
 - (II) “María aprobó Plástica, pero no aprobó Química”.
 - (III) “María aprobó exactamente una de las dos materias”.
- (b) Si p : “Pedro juega al paddle”, t : “Pedro juega al tenis”, r : “Pedro juega al rugby”:
- (I) “Pedro juega al paddle, al tenis y al rugby”.
 - (II) “Pedro juega al paddle, pero no al tenis ni al rugby”.
 - (III) “Pedro no juega al paddle, pero sí juega al tenis y al rugby”.
 - (IV) “Pedro no juega al paddle, ni al tenis, ni al rugby”.
 - (V) “Pedro juega a exactamente uno de esos tres deportes (juega solamente al paddle, o bien juega solamente al tenis, o bien juega solamente al rugby).”
 - (VI) “Pedro juega a exactamente dos de esos tres deportes.”

Ejercicio 6. Demostrar las siguientes propiedades usando tablas de verdad:

- (a) $(p \wedge q) \wedge r = p \wedge (q \wedge r)$ (asociatividad)
- (b) $p \wedge (q \vee r) = (p \wedge q) \vee (p \wedge r)$ (distributividad)
- (c) $p \vee (p \wedge q) = p$ (absorción)
- (d) $p \wedge (p \vee q) = p$ (absorción)
- (e) $\neg(p \vee q) = \neg p \wedge \neg q$ (De Morgan)

Ejercicio 7. Demostrar que los siguientes pares de expresiones son equivalentes, usando las propiedades conocidas (conmutatividad, asociatividad, distributividad, absorción, De Morgan, etc.):

- (a) p $p \wedge (p \vee q \vee r)$
- (b) $p \vee q \vee r$ $\neg(\neg p \wedge \neg q \wedge \neg r)$
- (c) $p \vee \neg q$ $\neg((p \vee q) \wedge \neg(\neg q \wedge p) \wedge \neg(p \wedge q))$
- (d) $p \wedge q$ $\neg(\neg p \vee \neg q) \vee (p \wedge q \wedge r)$
- (e) $p \wedge q$ $(p \vee q) \wedge \neg(\neg p \vee \neg q)$
- (f) $\neg p \wedge \neg q \wedge r$ $(\neg p \vee q \vee r) \wedge \neg(p \vee q \vee \neg r)$

Ejercicio 8.

- (a) Encontrar una expresión equivalente a $p \vee q$ que no use el operador \vee (o sea, una expresión que use solamente los operadores \neg y \wedge).
- (b) Encontrar una expresión equivalente a $p \wedge q$ que no use el operador \wedge .

Observación: En consecuencia, alcanza con dos operadores (\neg y \wedge , o bien \neg y \vee) para escribir cualquier expresión lógica.

Volviendo a Python...

Ejercicio 9. Un año es bisiesto si es múltiplo de 4 y no es múltiplo de 100, o bien si es múltiplo de 400. Ejemplos:

- 2020 es bisiesto, porque es múltiplo de 4 pero no de 100.
- 2021 no es bisiesto, porque no es múltiplo de 4.
- 1900 no es bisiesto, porque es múltiplo de 100 pero no de 400.
- 2000 es bisiesto, porque es múltiplo de 400.

Escribir una única expresión booleana en el lugar indicado en la siguiente función, de manera de cumplir con su especificación:

```
1 def bisiesto(a:int) -> bool:
2     ''' Requiere: a>=0
3         Devuelve: True si a es un año bisiesto; False si no.
4     '''
5     b:bool = _____
6     return b
```

Ejercicio 10. El sistema informático de una empresa ejecuta rutinariamente una serie de funciones para controlar que los valores de ciertas variables sean seguros. Nos piden auditar el código y encontramos algunas cosas que se podrían mejorar para lograr una mayor claridad.

(Sugerencia: Tener en cuenta las propiedades demostradas en el Ejercicio 6 y en el Ejercicio 7.)

- (a) En la función `controlar_m`, demostrar que las líneas 8 y 9 podrían eliminarse y obtener el mismo resultado en todos los casos.

```
1 def controlar_m(m1:int, m2:int) -> bool:
2     ''' Requiere: nada.
3         Devuelve: True si los valores de m1 y m2 son seguros; False si no.
4     '''
5     ok:bool = False
6     if m1<70:
7         ok = True
8     if m1<70 and m2<990:
9         ok = True
10    return ok
```

- (b) Mostrar que las líneas 6 y 7 de la función `controlar_x` podrían reemplazarse por una única condición más simple:

```
1 def controlar_x(x1:int, x2:int) -> bool:
2     ''' Requiere: nada.
3         Devuelve: True si los valores de x1 y x2 son seguros; False si no.
4     '''
5     ok:bool = False
6     if x1>100 or x2<50:
7         if not(x1<=100 or x2>=50):
8             ok = True
9     return ok
```