Instrucciones condicionales

Las instrucciones condicionales permiten elegir una entre dos vías para resolver un problema, según sea el resultado de evaluar un predicado.

```
Si predicado_se_cumple:
    hacer esto
si no:
    hacer lo otro
```

Por ejemplo, para saber si alguien aprueba o no, basta con ver si la nota es un cinco, al menos:

```
Si "la nota es un cinco al menos":
    la respuesta es que sí, que el estudiante aprueba
si no:
    la respuesta es negativa, pues el estudiante no aprueba
```

La respuesta a la pregunta de si "la nota es un cinco al menos" a un sí o un no, esto es, True o False, que son los valores booleanos. En la instrucción condicional, siempre hay una expresión booleana que gobierna cuál es la instrucción que se realiza.

Instrucción condicional completa

Lo anterior se puede expresar en Python así:

In [1]:
```
def aprobado(nota):
    if nota >= 5.0:
        return True
    else:
        return False

aprobado(3.0), aprobado(7.5), aprobado(5.0), aprobado(10.0), aprobado(4.
```

Out[1]:
```
(False, True, True, True, False)
```

La función anterior no está documentada, salvo la elección clara de los identificadores. Queremos progresar tan rápido que a menudo omitimos la documentación. He aquí la misma función de nuevo con una documentación más completa:

In [2]:
```
def aprobado(nota):
    """
    Averigua si una nota numérica es suficiente para aprobar o no

    Parameters
    ----------
    nota : float
        La nota numérica

    Precondition
    ------------
    nota >= 0 and nota <= 10

    Returns
    -------
    bool
        True, si la nota es suficiente para aprobar
        False, si la nota no lo es
    """
    if nota >= 5.0:
        return True
    else:
        return False

aprobado(3.0), aprobado(7.5), aprobado(5.0), aprobado(10.0), aprobado(4.
```

Out[2]:
```
(False, True, True, True, False)
```

Podemos añadir anotaciones de tipos, y entonces no deben repetirse en el docstring:

In [3]:
```
def aprobado(nota: float) -> bool:
    """
    Averigua si una nota numérica es suficiente para aprobar o no

    Parameters
    ----------
    nota : la nota numérica

    Precondition
    ------------
    nota >= 0 and nota <= 10

    Returns
    -------
        True, si la nota es suficiente para aprobar
        False, si la nota no lo es
    """
    if nota >= 5.0:
        return True
    else:
        return False

aprobado(3.0), aprobado(7.5), aprobado(5.0), aprobado(10.0), aprobado(4.
```

Out[3]:
```
(False, True, True, True, False)
```

Observa las dos versiones siguientes de la función. Las presento sin repetir la documentación por brevedad. Valora cuál de ellas es más clara.

In [4]:
```
def aprobado(nota):
    return True if nota >= 5.0 else False

aprobado(3.0), aprobado(7.5), aprobado(5.0), aprobado(10.0), aprobado(4.
```

Out[4]:
```
(False, True, True, True, False)
```

In [5]:
```
def aprobado(nota):
    return nota >= 5.0

aprobado(3.0), aprobado(7.5), aprobado(5.0), aprobado(10.0), aprobado(4.
```

Out[5]:
```
(False, True, True, True, False)
```

Instrucción condicional incompleta

La rama "else" no siempre es necesaria:

In [6]:
```
def valor_absoluto(x):
    if x < 0:
      # cambio de signo del dato de entrada
      x = -x
    return x

valor_absoluto(10), valor_absoluto(-10)
```

Out[6]:
```
(10, 10)
```

Seguro que puedes proponer tú una documentación más completa de la función anterior. En este caso, podríamos haber usado también la instrucción condicional completa:

In [7]:
```
def valor_absoluto(x):
    if x < 0:
        return -x
    else:
        return x

print(valor_absoluto(10), valor_absoluto(-10))

# La misma función con su documentación completa:

def valor_absoluto(x):
    """
    Devuelve el valor absoluto de un número, real

    Parameters
    ----------
    x : float

    Returns
    -------
    float
        El valor absoluto de x
    """
    if x < 0:
        return -x
    else:
        return x

print(valor_absoluto(7), valor_absoluto(-7))
```

```
10 10
7 7
```

Otro ejemplo:

In [8]:
```
def max_min(x, y):
    if x < y :
        max = y
        min = x
    else:
        max = x
        min = y
    return max, min

print(max_min(2, 3), max_min(3, 2))
```

```
(3, 2) (3, 2)
```

El ejemplo anterior ha sido útil como ejemplo de if... else ... Pero me extraña mucho que Python no tenga ya funciones predefinidas para calcular el máximo y el mínimo. Me arriesgo y pruebo a ver:

In [9]:
```
print(max(10, 13))
print(min(10, 13))
```

```
13
10
```

Por tanto, la función anterior puero redefinirla usando los recursos que me ofrece Python:

In [10]:
```
def max_min(x, y):
    return max(x, y), min(x, y)

print(max_min(2, 3), max_min(3, 2))
```

```
(3, 2) (3, 2)
```

Recomendación: antes de diseñar funciones, puede ser una buena idea ver si Python tiene ya una solución creada para ello:

```
Google: python maximum, minimum
```

Podemos anidar condicionales

In [11]:
```
def grado_poli(a, b, c):
    """
    Devuelve el grado del polinomio a*x^2 + b*x + c

    Parameters
    ----------
    a, b, c : float
        Coeficientes del polinomio

    Returns
    -------
    int
        Grado del polinomio
    """
    if a == 0:
        if b == 0:
            grado = 0
        else:
            grado = 1
    else:
        grado = 2
    return grado

grado_poli(2, 3, 4), grado_poli(0, 1, 2), grado_poli(1, 0, 0), grado_pol
```

Out[11]:
```
(2, 1, 2, 0, 0)
```

Elecciones múltiples

In [12]:
```
def calificacion(num_nota):
    """
    Converierte una calificación numérica en su denominación

    Parameters
    ----------
    num_nota : float
        la nota numérica

    Precondition
    ------------
    num_nota >= 0 and num_nota <= 10

    Returns
    -------
    str
        la denominación de la calificación
    """
    if num_nota < 5:
        return "Suspenso"
    elif num_nota < 7:
        return "Aprobado"
    elif num_nota < 9:
        return "Notable"
    else:
        return "Sobresaliente"

calificacion(10), calificacion(1.23), calificacion(3.45), calificacion(4
```

Out[12]:
```
('Sobresaliente', 'Suspenso', 'Suspenso', 'Suspenso', 'Notable', 'Aprob
ado')
```

Por tanto, es preferible, para las elecciones múltiples, usar "if .... elif ... elif ... else"

Expresión condicional

Observa la diferencia entre una *instrucción* condicional y una *expresión* condicional:

In [13]:
```python
def valor_abs_1(x):
    if x >= 0:
        return x
    else:
        return -x

def valor_abs_2(x):
    return x if x >= 0 else -x

print(valor_abs_1(7))
print(valor_abs_1(-7))
print(valor_abs_2(7))
print(valor_abs_2(-7))
```

```
7
7
7
7
```

Errores comunes

No cubrir todos los casos posibles.

In [14]:
```python
def val_absoluto_mal(x):
    if x < 0:
        return -x
    if x > 0:
        return x

print(val_absoluto_mal(0) == 0)
print(val_absoluto_mal(0))
```

```
False
None
```

Referencias

He aquí nuestra referencia, tomada de w3shhols:

`https://www.w3schools.com/python/python_conditions.asp`