Ejercicios if else

Ejercicio 1

Tienes que clasificar los exámenes de un montón de alumnos. Dichos exámenes tienen una nota numérica del 0 al 10. El mínimo para aprobar está en un 7.

Crea una función `passes_test` que recibe una nota numérica y devuelve `True` si es igual o superior a 7 y `False` en caso contrario.

Ejercicio 2

Aunque parezca mentira, se puede usar el sonido de los grillos para saber la temperatura. La frecuencia del grillo (ruídos por minuto) depende de su metabolismo, que a su vez depende de la temperatura.

La fórmula para obetner la temperatura a partir de la frecuencia (en minto) es la siguiente:

$$temperatura = 10 + \frac{frecuencia - 40}{7}$$

Crea la función `temp` que reibe la frecuencia y chirridos por minuto y te devulve la temperatura.

¡OJO¡ Esto sólo es válido par t > 0

Nuestra función no es total, es parcial y deberíamos tener en cuenta la condición excepcional (t<=0) y gestionar dicho error. Aun no sabemso cómo hacerlo, pero lo veremos luego.

Ejercicio 3

En una empresa quieren subir el sueldo a todos los empleados cuyo sueldo sea inferior a un cierto nivel.

Crea la función `new_salary` que recibe el sueldo actual. Si dicho sueldo es inferior o igual a €1000, le aplica un incremento del 8%. Si no, se queda como está.

Ejercicio 3.1

Si has hecho algo así:

```
def new_salary(old_salary):
    if old_salary <= 1000:
        return old_salary * 1.08
    else:
        return old_salary
```

No está mal, pero el fantasma de Edsger Dijkstra (averigua quién fue) te perseguirá toda tu vida:

> Una función sólo puede tener un punto de salida, es decir, un sólo return

Ejercicio 3.2

Ya tienes la función como le gustaría al programador más gruñón de la historia:

```
def new_salary(old_salary):
    new_salary = old_salary
    if old_salary <= 1000:
        new_salary =  old_salary * 1.08
    return new_salary
```

Al mes siguiente te dicen que se equivocaron y que quieren subirle el sueldo a todos los que tengan un sueldo inferior a €1200, un 9%.

¿Cómo usarias la misma función, haciendo las menores modificaciones posibles y sabiendo que es muy probable que lo vuelvan a cambiar en el futuro?

> Parametriza todo y usa valores por defecto para los valores comunes.

In [ ]:

```
def new_salary(old_salary, threshold = 1200, rise = 0.09 ):
    new_salary = old_salary
    if old_salary <= threshold:
        new_salary = old_salary * (1 + rise)
    return new_salary
```

Ejercicio 4

Modifica todos los ejercicios anterior para que tengan un solo return

Ejercicio 5

¡Primavera en El Corte Inglés! Si en tu carta de compra tienes un total superior a €1400, se te aplicará un descuento del 15%.

Crea la función `discount` que recibe el total y devuelve el descuento en euros. Si el total es €1400 o más, se devuelve el nuevo precio con el descuento aplicado.

Si la compra es de menos de €1400, el descuento no se aplica y la función devuelve el total original.

Ejercicio 6

Se ha descubierto que el consumo de cerveza (en litros) por persona y al mes puede ser estimado según la temperatura promedio de dicho mes (en Celsius).

La fórmula encontrada es la siguiente

$$litros = 1.2 * temp - 22$$

1. Crea la función `beer_per_month` que estima esa fórmula: recibe la temperatura y devuelve los litros por barba (estimados)
   A. ¿Es una función total o parcial?
2. Crea la función `is_too_much_beer` que recibe la temperatura, los litros que una persona ha bebido y devuelve `True` si la persona se está pasando (bebiendo más del promedio) y `False` en caso contrario.
3. Ahora crea la función `is_too_little_beer` que recibe lo mimso que la anterior, pero devuelve `True` si está bebiendo de menos y `False` si bebe de más.
   A. ¿Seguro que no hay una forma más sencilla de hacerlo?

Ejercicio 7

Hay un operador aritmético llamado *módulo* que no hemso visto hasta ahora. Funciona como la división, pero el resultado que te dá es el *resto de la división entera*.

Por ejemplo, `3 / 2` devuelve `1.5`, mientras que `3 % 2` devuelve `0.5`.

Esto nos permite saber si un nomero es divisible por otro: si el resultado de la operación es cero.

Crea dos funciones `is_even y is_odd` que reciben un número y determinar si éste es par o impar, respectivamente.

Ejercicio 8

Crea la función `has_suffix(string, prefix)` que recibe dos cadenas y devuelve `True` la segunda es un prefijo de la primera.

Es decir `has_suffix("hola mundo", "hola")` devolvería `True`.

Pista Si `prejijo` es más largo que `string`, ¿hace falta comparar algo?

- Crea la función `is_suffix`, que hace lo opuesto: comprueba si la segunda cadena es un sufijo de la primera

Predicados

Hemos visto algunso ejemplos de funciones que reciben algún tipo, y devuelven un `bool`.

A esas funciones se les llama *predicados* (del Latín *predicare*, afrimar, hacer saber) y son muy comunes e importantes.

Sirven normalmente como un test, para saber si el valor de entrada supera o no alguna condición.

Veremos muchos *predicados* por el camino.

Nomenclatura

Suelen tener nombres que empiezan por *is* (es), *has* (tiene) etc.

Por ejemplo:

- `is_even`, `is_male`, etc
- `has_discount`, `has_cookies`, etc