

Table of Contents

- 1 Comparaciones
- 2 ¿Qué es el control flow o control de flujo?
- 3 Estructuras de selección:
 - 3.1 La función `if`
 - 3.2 If ... else ...
 - 3.3 If ... elif ... else
 - 3.4 If anidados
 - 3.5 Condiciones múltiples
 - 3.6 Repaso: Condicionales if ... elif ... else
- 4 Estructuras de iteración
 - 4.1 `while`
- 5 Ejercicios

Comparaciones

Nuestra primera clase aprendimos que eran los operadores condicionales, los cuales incluían `==`, `!=`, `<=`, `>=`, etc. Estos operadores son importantes en los bucles `if` de Python porque permiten que el programa tome decisiones basadas en una condición. En otras palabras, los operadores condicionales permiten que el programa execute diferentes secciones de código según se cumpla o no una condición dada.

En Python, el operador condicional más comúnmente utilizado en los bucles `if` es el operador de comparación "`==`", que se utiliza para comparar dos valores. Por ejemplo, si queremos comprobar si una variable "x" es igual a 10. Recordemos algunos de ellos:

In [59]: `# Lo primero que hacemos es definir unas variables`

```
a1 = 0
a2 = 1
a3 = 2
```

In [60]: `# es a1 menor que a2?`

```
print("Es 'a1' menor que 'a2'?:", a1 < a2)
print("Es 'a3' mayor o igual que 'a1'?:", a3 >= a1)
print("Es 'a3' mayor que 'a1' y menor que 'a2'?:", a1 < a3 < a2)
print("Es 'a1' igual que 'a2':", a1 == a2)
```

Es 'a1' menor que 'a2'? True
Es 'a3' mayor o igual que 'a1'? True
Es 'a3' mayor que 'a1' y menor que 'a2'? False
Es 'a1' igual que 'a2'? False

En resumen, para estas comparaciones (condiciones) teníamos los siguientes operadores:

- `A > B`, True si A es mayor que B

- A < B , True si A es menor que B
- A == B , True si A es igual a B
- A >= B , True si A es mayor que B o igual a B
- A <= B , True si A es menor que B o igual a B
- A != B , True si A es distinto a B

¿Qué es el control flow o control de flujo?

En Python, el "control flow" o control de flujo se refiere al orden en que se ejecutan las instrucciones en un programa o código. El control flow determina qué instrucciones se ejecutan y en qué orden, y esto es controlado por estructuras como los bucles, las condiciones y las funciones.

Existen tres tipos principales de estructuras de *control flow* en Python:

- **Estructuras de selección:** permiten al programa tomar decisiones y ejecutar diferentes secciones de código en función de una condición. La estructura de selección más común en Python es el bucle `if`, que ejecuta una sección de código si se cumple una condición y otra sección de código si no se cumple la condición.
 - `if`
 - `if ... else ...`
 - `if ... elif ... else`
- **Estructuras de iteración:** permiten al programa repetir una sección de código varias veces. El bucle "for" es una estructura de iteración común en Python, que ejecuta una sección de código una vez para cada elemento de una lista o un rango de valores.
 - `for`
 - `while`
- **Estructuras de control de funciones:** permiten al programa llamar a una función y controlar el flujo de ejecución del programa dentro y fuera de la función. Por ejemplo, la estructura "try/except" permite al programa manejar errores dentro de una función y continuar ejecutando el código.
 - `break`
 - `continue`
 - `pass`

En resumen, el control flow en Python se refiere al orden en que se ejecutan las instrucciones en un programa y está controlado por estructuras como los bucles, las condiciones y las funciones. Estas estructuras permiten al programa tomar decisiones, repetir secciones de código y controlar el flujo de ejecución.

En las próximas lecciones iremos viendo cada una de estas estructuras. Pero antes de ponernos manos a la obra, pongamos un ejemplo sencillo, digamos que queremos medir la temperatura del aire en el exterior. ¿Qué haríamos como "humanas" para ejecutar esta acción?

- 1 Cogemos el termómetro,
- 2 Abrimos la puerta,
- 3 Salimos al exterior,
- 4 Cerramos la puerta,
- 5 Buscamos un lugar a la sombra,
- 6 Medimos la temperatura y
- 7 La anotamos.

Estructuras de selección:

La función `if`

La función `if` en Python es una estructura de control de flujo que se utiliza para tomar decisiones basadas en una condición. La sintaxis básica de la función `if` es la siguiente:

```
if condición:
    # Sección de código que se ejecuta si la condición es verdadera
```

- Si la condición `if` se evalúa como True, entonces se ejecuta el código indentado que sigue a la sentencia.
- Si la expresión se evalúa como Falso, entonces el código indentado que sigue a la sentencia `if` se salta y el programa ejecuta la siguiente línea de código que está indentada al mismo nivel que la sentencia `if`.

Veamos algunos ejemplos:

```
In [61]: # definimos una variable usando la función input donde os preguntaremos cuántas
## 🌟 Nota: si ejecutais este celda, poned un número que sea igual a 6.

numero_peras = int(input("¿Cuántas peras tienes en la nevera?"))
print("El número de peras que tenemos es:", numero_peras)
```

El número de peras que tenemos es: 6

```
In [62]: # vamos a empezar a hacer algunas operaciones básicas, empezaremos por chequear
if numero_peras > 4: # si el número de peras es mayor que 4
    # en caso de que se cumpla la condición queremos sacar un mensaje, para lo q
    print('Tenemos muchas peras, ¿no te apetece comerte una?')
```

Tenemos muchas peras, ¿no te apetece comerte una?

```
In [63]: # Pero... ¿Qué pasaría si no se cumple la condición que especificamos en el `if`?
# vamos a chequear si tenemos más de 10 peras en la nevera. Es decir, en este ca
if numero_peras > 10:
    print('Tenemos muchas peras, ¿no te apetece comerte una?')
```

Upss... no ha pasado nada!!! Y es que si nos fijamos en nuestro código especificamos lo que queríamos que pasara si la condición se cumple, pero no dijimos que debía ocurrir si la condición no se cumple. Pero no nos preocupemos, un poco más adelante en esta lección aprenderemos como hacerlo!

Vamos a repetir el ejemplo, pero con más consecuencias. Queremos que si hay muchas peras en la nevera el ordenador me lo diga no una sino dos veces. En cambio, si no son tantas, no quiero que me alerte. Ejecuta las celdas de abajo. ¿Hace lo que te esperabas?

```
In [64]: print("El número de peras que tenemos es:", numero_peras)
print("-----")
# si el número de peras en la nevera es mayor que 4 queremos que:
if numero_peras > 4:
    # nos printees primero que hay muchas peras
    print('Tenemos muchas peras, ¿no te apetece comerte una?')
    # nos printees despues un mesaje repitiendo que tenemos muchas peras
    print('Repite: Tenemos muchas peras, ¿no te apetece comerte una?')
```

El número de peras que tenemos es: 6

Tenemos muchas peras, ¿no te apetece comerte una?

Repite: Tenemos muchas peras, ¿no te apetece comerte una?

No necesariamente podemos poner **prints**. Podemos hacer operaciones dentro del **if**. Imaginemos que queremos hacer la lista de la compra. Nuestro criterio para añadir elementos a la lista de la compra es que si tenemos menos de 2 peras, queremos comprar más. Es decir, si tenemos dos o menos de dos peras en la nevera lo queremos añadir a la lista de la compra.

¿Cómo haríamos esto con código? 🤔

Para esto tendremos que usar métodos que hemos ido aprendiendo en lecciones anteriores:

```
In [65]: # Lo primero que hacemos es crearnos una lista, que será nuestra lista de la compra

lista_compra = []
print("La lista de la compra contiene:", lista_compra)
print("-----")

print("El número de peras que tenemos en la nevera es:", numero_peras)
print("-----")

# imaginemos que nos hemos comido 5 peras. Tendremos que restar 5 a nuestra variable

numero_peras -= 5 # esto es lo mismo que poner numero_peras = numero_peras - 5
print("El número de peras que tenemos en la nevera es:", numero_peras)
print("-----")

# por último definimos nuestro if. Lo que chequearemos es si tenemos menos de dos peras

if numero_peras < 2:
    # si esta condición se cumple queremos añadir peras a nuestra lista. Para eso usaremos el método append()
    lista_compra.append("peras")
print("La lista de la compra contiene:", lista_compra)
```

```
La lista de la compra contiene: []
-----
El número de peras que tenemos en la nevera es: 6
-----
El número de peras que tenemos en la nevera es: 1
-----
La lista de la compra contiene: ['peras']
```

Y voilá!!! ya tenemos las peras en nuestra lista 🍎. Otras operaciones que podremos hacer dentro de un `if` son sumas, restas, multiplicaciones, etc. Veamos un ejemplo. Imaginemos ahora que queremos actualizar el número de peras si he ido a la compra.

```
In [66]: print("El número de peras que tenemos en la nevera es:", numero_peras)
print("-----")

# imaginemos que fuimos a La compra y hemos vuelto a casa y queremos actualizar
peras_compradas = 10
print("Hemos comprado:", peras_compradas, "peras")
print("-----")

# ahora tendremos que chequear si en nuestra lista de la compra teníamos peras.
# chequeamos si peras está en nuestra lista
if "peras" in lista_compra:

    # si está en nuestra lista, lo que haremos será sumar al número de peras que
    # tenemos en la nevera
    numero_peras += peras_compradas
print("El número de peras que tenemos en la nevera es:", numero_peras)
print("-----")
```

```
El número de peras que tenemos en la nevera es: 1
-----
Hemos comprado: 10 peras
-----
El número de peras que tenemos en la nevera es: 11
-----
```

Si os fijais, dentro del `if` hemos usando una sintaxis que nos puede resultar nueva. Es la forma abreviada de:

`numero_peras = numero_peras + peras_compradas`
Es decir:

```
# esta Línea de código
numero_peras = numero_peras + peras_compradas

# es exactamente lo mismo que esta
numero_peras += peras_compradas
```

If ... else ...

Hemos visto que pasaba si la condición que le pasabamos al `if` se cumplía, pero hasta ahora no hemos visto que pasaba si la condición no se cumplía. En Python, el `if ... else` es una estructura condicional que se utiliza para ejecutar un bloque de código si se cumple una condición y otro bloque de código si no se cumple.

La sintaxis básica del `if ... else` en Python es la siguiente:

```
if condicion_1:
    # código a ejecutar si se cumple la condición 1
else:
    # código a ejecutar si no se cumple ninguna de las condiciones
    # anteriores
```

En el mismo contexto de la lista de compra, ahora querremos saber si tenemos peras suficientes o si nos tiene que avisar para añadirla a la lista de la compra. Veamos un ejemplo:

```
In [67]: # Lo primero que haremos será recordar cuántas peras tenemos en la nevera

print("El número de peras que tenemos en la nevera es:", numero_peras)
print("-----")

# primero chequearemos si tenemos menos de 2 peras. Recordamos era nuestra condición
if numero_peras < 2:
    print('Tenemos pocas peras, hay que meterlas en la lista de la compra')
    lista_compra.append("peras")
# si esta condición no se cumple, usaremos el else para especificar que queremos
else:
    print('Tenemos muchas peras, deberíamos comernoslas...')
```

El número de peras que tenemos en la nevera es: 11

Tenemos muchas peras, deberíamos comernoslas...

Si nos fijamos en este ejemplo, ¿Qué es lo que está pasando?

- Python empezará leyendo el código del `if` y chequeará si esa condición es verdadera o Falsa.
- En este caso, la condición es Falsa, por lo que no se ejecutará el código que está en el interior del `if`.
- Después seguirá leyendo código. El `else` lo que indica es lo que queremos que ocurra si no se cumple la condición del `if`.
- En este caso, al ser False la condición del `if` se ejecutarán las líneas de código que están dentro del `else`, es decir, tenemos que comer peras.

💡 Nota :

- Fijaos como la indentación de la `else` es igual que la de la `if`.
- En el `else` no hay que pasar ninguna condición, ya que el `else` va a englobar todas las condiciones que no se cumplen en el `if`.

¿Qué pasaría si la condición del `if` se cumple?

```
In [68]: # creamos otra variable de numero de peras distinta
numero_peras2 = 1
print("El número de peras que tenemos en la nevera es:", numero_peras2)
print("-----")
```

```
# primero chequearemos si tenemos menos de 2 peras. Recordamos era nuestra condición
if numero_peras2 < 2:
    print('Tenemos pocas peras, hay que meterlas en la lista de la compra')
    lista_compra.append("peras")
# si esta condición no se cumple, usaremos el else para especificar que queremos
else:
    print('Tenemos muchas peras, deberíamos comernoslas...')

print("La lista de la compra contiene:", lista_compra)
print("-----")
```

El número de peras que tenemos en la nevera es: 1

```
-----
Tenemos pocas peras, hay que meterlas en la lista de la compra
La lista de la compra contiene: ['peras', 'peras']
-----
```

Fijaos que tenemos dos veces peras porque ya la añadimos en el apartado anterior cuando estabamos aprendiendo la condición `if`.

Repaso:

- Con la palabra `else` ampliamos la función `if` para el caso que la condición sea falsa.
- `Else` tiene la misma indentación que la `if` al cual pertenece, y su consecuencia la misma indentación que la consecuencia de la `if`.

If ... elif ... else

Muchas veces, nos interesa chequear más de una condición. Como vimos, la condición que ponemos en la línea de la `if` tiene un `True` o un `False` como resultado, y con el `else` incluimos todas las situaciones que no se cumplen en el `if`. La sentencia `elif` de Python permite realizar comprobaciones continuas después de una sentencia `if` inicial. Una sentencia `elif` se diferencia de la sentencia `else` porque se proporciona otra expresión/condición para ser comprobada, al igual que con la sentencia `if` inicial.

Ya hemos visto el `if ... "else"`, que como hemos dicho es una estructura condicional que se utiliza para ejecutar un bloque de código si se cumple una condición y otro bloque de código si no se cumple, que seguía la siguiente sintaxis:

```
if condicion:
    # código a ejecutar si se cumple la condición
else:
    # código a ejecutar si no se cumple la condición
```

En este ejemplo, "condicion" es una expresión booleana que se evalúa como verdadera o falsa. Si "condicion" es verdadera, se ejecutará el código dentro del primer bloque `if`, y si es falsa, se ejecutará el código dentro del bloque "`else`".

Además, también se puede utilizar la estructura `if else` anidada para evaluar

múltiples condiciones. En este caso, la sintaxis sería la siguiente:

```
if condicion_1:  
    # código a ejecutar si se cumple la condición 1  
elif condicion_2:  
    # código a ejecutar si no se cumple la condición 1 pero se cumple  
    la condición 2  
else:  
    # código a ejecutar si no se cumple ninguna de las condiciones  
    anteriores
```

Imaginemos ahora que queremos incluir varias condiciones, estas serían:

- Si tenemos más de 50 peras, devolveremos un mensaje de que tenemos demasiadas peras y deberíamos donarlas.
- Si tenemos más de 4 peras, devolveremos un mensaje diciendo que si no te apetece comer peras
- Si tenemos más de 0 peras y menos de 4, las apuntaremos en la lista de compra
- Si no se cumple ninguna de las condiciones, devolveremos un mensaje de que nos hemos quedado sin peras.

```
In [69]: # recordemos cuántas peras teníamos  
print("El número de peras que tenemos en la nevera es:", numero_peras)  
print("-----")  
  
# indicamos nuestra primera condición. Es decir, tener mas de 50 peras  
if numero_peras > 50:  
    print('Tenemos demasiadas peras, dónalas al banco de alimentos.')  
  
# en el elif pondremos las nuevas condiciones que queremos chequear. En este caso  
elif numero_peras > 4:  
    print('Tenemos muchas peras, ¿no te apetece comerte una?')  
  
# podemos añadir todas las condiciones que queramos, es decir, podemos añadir tanto como queramos  
elif numero_peras > 0:  
    print("Añadiendo elemento a la lista de la compra")  
    lista_compra.append("peras")  
  
# por último añadimos el else especificamos lo que queremos que ocurra si no se cumplen las condiciones  
else:  
    print('No tenemos peras.')
```

El número de peras que tenemos en la nevera es: 11

Tenemos muchas peras, ¿no te apetece comerte una?

Fijaos que el orden de las condiciones importa, ya que va a ir chequeando condición a condición y en el momento en el que la condición sea cierta el programa se parará y no se ejecutará el resto del código:

Imaginemos que ponemos primero la condición de mayor que 0 que la de mayor que 4. ¿Qué creéis que pasará?

```
In [70]: # recordemos cuántas peras teníamos
print("El número de peras que tenemos en la nevera es:", numero_peras)
print("-----")

# indicamos nuestra primera condición. Es decir, tener mas de 50 peras
if numero_peras > 50:
    print('Tenemos demasiadas peras, dónalas al banco de alimentos.')

elif numero_peras > 0:
    print("Añadiendo elemento a la lista de la compra")
    lista_compra.append("peras")

elif numero_peras > 4:
    print('Tenemos muchas peras, ¿no te apetece comer una?')

else:
    print('No tenemos peras.')
```

El número de peras que tenemos en la nevera es: 11

Añadiendo elemento a la lista de la compra

¿Qué ha pasado aquí?

Python a empezado a leer nuestro programa:

- ¿Es el `numero_peras` mayor que 50? `False`, por lo tanto no se ejecuta la línea que tenemos "dentro" del `if`, así que sigue leyendo código.
- ¿Es el `numero_peras` mayor que 0? `True`, por lo tanto se ejecuta la línea de código que tenemos dentro del primer `elif`, es decir, nos añade "peras" a la lista de la compra y nos devuelve el mensaje de que la está añadiendo.
- Como la condición se ha cumplido, el programa se para y no sigue leyendo código. Por esto no nos sale el `print` del siguiente `elif` aunque se cumpla la condición.

Lo mismo pasaría si el número de peras fuera 60. Veamoslo con un ejemplo:

```
In [71]: # definimos una nueva variable donde el número de peras sea 60
numero_peras3 = 60
print("El número de peras que tenemos en la nevera es:", numero_peras3)
print("-----")

# si ejecutamos el código que creamos al inicio del apartado
if numero_peras3 > 0:
    print("Añadiendo elemento a la lista de la compra")
    lista_compra.append("peras")

elif numero_peras3 > 50:
    print('Tenemos demasiadas peras, dónalas al banco de alimentos.')

elif numero_peras3 > 4:
    print('Tenemos muchas peras, ¿no te apetece comer una?')

else:
    print('No tenemos peras.')
```

El número de peras que tenemos en la nevera es: 60

Añadiendo elemento a la lista de la compra

A pesar de que todas las condiciones, tanto del `if` como las de los `elif` son ciertas, lo que va a ocurrir es que el programa se parará en el momento en el que Python encuentre una condición verdadera. En este caso, la primera.

Algunas cosas importantes cuando hacemos este tipo de condicionales incluyendo `if ... elif ... else :`

- Hay múltiples `elif`, pero un solo `if` y `else`.
- El programa se parará cuando encuentre una condición que sea `True`. Ignorando el código que viene a continuación.

If anidados

Básicamente es una sentencia `if` dentro de otra sentencia `if`. Los `if` anidados pueden anidarse tanto como sea necesario, lo que permite construir programas muy complejos que responden a múltiples condiciones. Sin embargo, es importante tener en cuenta que el uso excesivo de `if` anidados puede hacer que el código sea difícil de leer y mantener. Por lo tanto, es recomendable utilizarlos con moderación y, en su lugar, utilizar técnicas de programación más avanzadas, como las estructuras de datos y los algoritmos.

En este caso vamos a dividir nuestro código en varios bloques:

- 1 Chequearemos si tenemos más de 4 peras. En caso de que tengamos más de 4 peras, definiremos nuevas condiciones:
 - 1.1 La primera condición que chequearemos dentro de este primer bloque será si tenemos más de 50 peras. En este caso devolveremos un mensaje sugiriendo donarlas.
 - 1.2 Si esta condición no se cumple devolveremos un mensaje ofreciendo comer una pera.
- 2 Estableceremos el resto de las posibilidades, es decir, que tengamos 4 o menos peras
 - 2.1 En caso de que tengamos menos de 4 peras, si el número de peras es mayor que 0, añadiremos peras a nuestra lista de la compra
 - 2.2 En caso de que no se cumpla la condición anterior devolveremos un mensaje de que estamos sin peras.

```
In [72]: print("El número de peras que tenemos en la nevera es:", numero_peras)
print("-----")
```

```
# 1 primera condición. Es el número de peras mayor que 4?  
if numero_peras > 4:  
  
    # 1.1 en caso de que sea si, ¿Es el número de peras mayor que 50?  
    if numero_peras > 50 :  
        print('Tenemos demasiadas peras, dónalas al banco de alimentos.')  
  
    # si se cumple la condición 1 pero no la 1.1 se ejecutará esta línea de  
    else:  
        print('Tenemos muchas peras, ¿no te apetece comerte una?')  
  
# si no se cumple la condición 1, pasaremos a esta parte del código, es decir,  
else:  
    # 2.1 Si el número de peras es menor que 4, pero mayor que 0 se añadirá p  
    if numero_peras > 0 :  
        lista_compra.append("peras")  
        print("añadiendo peras a la lista de la compra")  
  
    # 2.2 en caso de que no cumplan ninguna de las condiciones se ejecutará e  
    else:  
        print('No tenemos peras.')
```

El número de peras que tenemos en la nevera es: 11

Tenemos muchas peras, ¿no te apetece comerte una?

En este caso teníamos 11 peras, por lo tanto se cumple la condición **1** y la condición **1.2**, por lo tanto nos devuelve el mensaje de sugerencia de comernos una pera.

Puede que esto nos resulte algo confuso, pero lo podemos entender como un camino en el que nos vamos encontrando bifurcaciones que nos van a ir definiendo nuestro camino en función de que vayan cumpliendo las condiciones pasadas.

```
In [73]: # creamos otra variable con un número diferente de peras
numero_peras4 = 60

print("El número de peras que tenemos en la nevera es:", numero_peras4)
print("-----")

# 1 primera condición. Es el número de peras mayor que 4?
if numero_peras4 > 4:

    # 1.1 en caso de que sea si, ¿Es el número de peras mayor que 50?
    if numero_peras4 > 50:
        print('Tenemos demasiadas peras, dónalas al banco de alimentos.')

    # si se cumple la condición 1 pero no la 1.1 se ejecutará esta línea de
    else:
        print('Tenemos muchas peras, ¿no te apetece comerte una?')

# si no se cumple la condición 1, pasaremos a esta parte del código, es decir,
else:
    # 2.1 Si el número de peras es menor que 4, pero mayor que 0 se añadirá p
    if numero_peras4 > 0:
        lista_compra.append("peras")
        print("añadiendo peras a la lista de la compra")

    # 2.2 en caso de que no cumplan ninguna de las condiciones se ejecutará e
    else:
        print('No tenemos peras.')
```

El número de peras que tenemos en la nevera es: 60

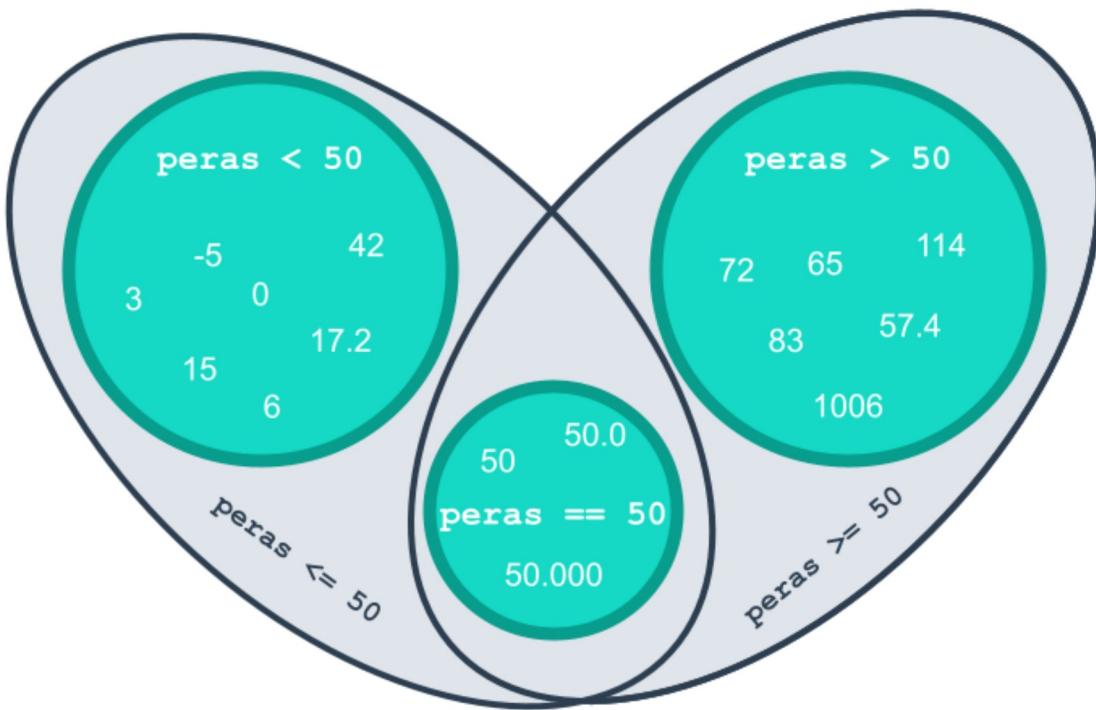
Tenemos demasiadas peras, dónalas al banco de alimentos.

En este caso se cumplen las condiciones 1 y 1.1, por lo tanto nos devuelve el mensaje de donar.

Condiciones múltiples

Usando `if` especificamos hasta ahora una sola condición. Si no se cumple la condición, seguimos con `elif` y `else`. Incluso hemos visto ejemplos incluyendo una función `if ... elif else` dentro de otra función `if`.

Si nos fijamos en la imagen de abajo, los números en blanco pueden ser agrupados de distintas maneras. Nota como los tres círculos turquesas no se solapan entre sí, mientras cada uno de los grises envuelve dos círculos turquesas. Si solo nos interesa la situación en que hay menos que 50 peras, es decir el valor de la variable `peras` cumple la condición `< 50`, podríamos especificar tanto `peras < 50` o bien `peras >= 50` seguido por un `else`.



```
In [74]: print("El número de peras que tenemos en la nevera es:", numero_peras)
print("-----")
# recordemos también el código que habíamos creado un poco más arriba incluyendo
# Recordad, siempre es importante Leer el código con detenimiento
if numero_peras > 50:
    print('Tenemos demasiadas peras, dónalas al banco de alimentos.')

elif numero_peras > 4:
    print('Tenemos muchas peras, ¿no te apetece comerte una?')

elif numero_peras > 0:
    print("Añadiendo elemento a la lista de la compra")
    lista_compra.append("peras")

else:
    print('No tenemos peras.')
```

El número de peras que tenemos en la nevera es: 11

Tenemos muchas peras, ¿no te apetece comerte una?

En este apartado de la lección estamos aprendiendo condiciones múltiples, es decir, dentro de un `if` o `elif` podemos incluir varias condiciones. Vamos a intentar reducir el código anterior a un menor número de líneas. Para esto tendremos que usar operadores como `and` u `or` que aprendimos en lecciones anteriores.

```
In [75]: print("El número de peras que tenemos en la nevera es:", numero_peras)
print("-----")

if numero_peras > 4 and numero_peras < 50:
    print('Tenemos demasiadas peras, dónalas al banco de alimentos.')

elif 4 < numero_peras and numero_peras > 0:
    print('Añadiendo elemento a la lista de la compra')
```

El número de peras que tenemos en la nevera es: 11

Tenemos demasiadas peras, dónalas al banco de alimentos.

Mucho más corto, ¿verdad? De este ejemplo podemos sacar varias conclusiones:

- Se pueden especificar rangos de valores para una variable en una sola línea. Para ello recuerda que tenemos < , <= , >= , y > .
- Podemos usar los operadores or y and para establecer varias condiciones.
- El else es opcional.

Hasta ahora, todas las condiciones que evaluamos eran valores de la misma variable.

Vamos a añadir otra variable, de tipo bool: nevera_en_marcha . En este caso estableceremos las siguientes condiciones:

- 1 Si la nevera está en marcha
 - 1.1 Y el número de peras está entre 0 y 4 añadiremos las peras a nuestra lista de la compra
 - 1.2 Y el numero de peras está entre 4 y 50 printaremos un mensaje de sugerencia para comer peras
- 2 En caso de que la nevera esté apagada printaremos un mensaje diciendo que la nevera está rota.

```
In [76]: # establecemos que la nevera está en marcha
nevera_en_marcha = True
print("¿La nevera esta encendida?", nevera_en_marcha)
print("-----")

print("El número de peras que tenemos en la nevera es:", numero_peras)
print("-----")

# chequeamos el valor de la nevera. En caso de que esté encendida
if nevera_en_marcha: # esto es lo mismo que poner if nevera == True

    # en caso de que este encendida y el número de peras este entre 0 y 4
    if 0 < numero_peras < 4:
        print('Añadimos peras a la lista')
        lista_compra.append('peras')

    # en caso de que el número de peras este entre 4 y 50 sugerimos comer peras
    elif 4 < numero_peras < 50:
        print('Tenemos muchas peras, ¿no te apetece comerte una?')

# en caso de que la nevera esté apagada, es decir que el valor de nevera_en_marcha
else:
    # sacaremos por pantalla el mensaje de que la nevera esta rota
    print('La nevera está rota.')
```

```
¿La nevera esta encendida? True
-----
El número de peras que tenemos en la nevera es: 11
-----
Tenemos muchas peras, ¿no te apetece comerte una?
```

En este caso:

- Se cumple la primera condición, es decir, la variable `nevera_en_marcha == True`.
- Dentro de esta condición tenemos otras dos condiciones:
 - Que el número de peras este entre 0 - 4, el `if` dentro del primer `if`.
 - Que el número de peras este entre 4 - 50, el `else` dentro del primer `if`.

En este caso se cumple la segunda condición, por lo tanto, el programa nos devolverá el mensaje de que tenemos muchas peras.

```
In [77]: # volvemos a ejecutar el código de antes, pero en este caso para la variable nevera_en_marcha2 = False

print("¿La nevera esta encendida?", nevera_en_marcha2)
print("-----")

print("El número de peras que tenemos en la nevera es:", numero_peras)
print("-----")

# chequeamos el valor de la nevera. En caso de que esté encendida
if nevera_en_marcha2: # esto es lo mismo que poner if nevera == True

    # en caso de que este encendida y el número de peras este entre 0 y 4
    if 0 < numero_peras < 4:
        print('Añadimos peras a la lista')
        lista_compra.append("peras")

    # en caso de que el número de peras este entre 4 y 50 sugerimos comer peras
    elif 4 < numero_peras < 50:
        print('Tenemos muchas peras, ¿no te apetece comerte una?')

# en caso de que la nevera esté apagada, es decir que el valor de nevera_en_marcha2 es False
else:
    # sacaremos por pantalla el mensaje de que la nevera esta rota
    print('La nevera está rota.')
```

```
¿La nevera esta encendida? False
-----
El número de peras que tenemos en la nevera es: 11
-----
La nevera está rota.
```

En este caso, como el valor de `nevera_en_marcha2` es igual a `False`, el primer `if` no se cumplirá por lo que el programa pasará automáticamente al primer `else` y se nos printeará el mensaje de que la nevera está rota.

👉 Notad como el orden de las condiciones es importante! Y como antes de ponernos a definir nuestras condiciones debemos pensar en que orden queremos ejecutarlas. De nuevo chicas, siempre es recomendable si tenemos dudas escribir en un papel lo que

queremos y luego "traducirlo" a Python.

Veamos un último ejemplo: todos los viernes nos comemos un helado. Si mi congelador está roto, salgo a la heladería para cumplir con nuestro ritual. Cuando es viernes, mi *primera opción* obviamente será buscar los helados en mi propio congelador.

```
# definimos nuestra primera condición, que sea viernes o no
if hoy_es_viernes:
    if congelador_en_marcha:
        print('Coge un helado, ¡te lo mereces!')
    else:
        print('Sal a la heladería, ¡te lo mereces!')

# definimos nuestra segunda condición, que el congelador funcione o no
if congelador_en_marcha:
    if hoy_es_viernes:
        print('Coge un helado, ¡te lo mereces!')
else:
    if hoy_es_viernes:
        print('Sal a la heladería, ¡te lo mereces!')
```

Pero este código de arriba puede resultar algo redundante, ya que estamos aprendiendo que podemos definir múltiples condiciones en un `if` o un `elif` usando operadores del tipo `and` u `or`. Y este ejemplo que tenemos ahora es un buen ejemplo! Veamos como lo podemos simplificar:

```
In [78]: # como siempre, definimos nuestras variables

hoy_es_viernes = True
print("¿Es hoy viernes?", hoy_es_viernes)
print("-----")

congelador_en_marcha = False
print("¿Está en congelador en marcha?", congelador_en_marcha)
print("-----")

# es el momento de definir todas nuestras condiciones

# si es viernes (hoy_es_viernes == True) Y el congelador funciona (congelador_en_marcha == True)
if hoy_es_viernes and congelador_en_marcha:
    print('Coge un helado, ¡te lo mereces!')

# en caso de que sea viernes (hoy_es_viernes == True), Y el congelador no funcione (congelador_en_marcha == False)
elif hoy_es_viernes and not congelador_en_marcha:
    print('Sal a la heladería, ¡te lo mereces!')

else:
    print("Todavía no toca helado, pero ya queda menos!")

¿Es hoy viernes? True
-----
¿Está en congelador en marcha? False
-----
Sal a la heladería, ¡te lo mereces!
```

```
In [79]: # ¿qué pasaría si no fuera viernes?

hoy_es_viernes2 = False
print("¿Es hoy viernes?", hoy_es_viernes2)
print("-----")

congelador_en_marcha2 = False
print("¿Está en congelador en marcha?", congelador_en_marcha2)
print("-----")

if hoy_es_viernes2 and congelador_en_marcha2:
    print('Coge un helado, ¡te lo mereces!')

elif hoy_es_viernes2 and not congelador_en_marcha2:
    print('Sal a la heladería, ¡te lo mereces!')
else:
    print("Todavía no toca helado, pero ya queda menos!")
```

```
¿Es hoy viernes? False
-----
¿Está en congelador en marcha? False
-----
Todavía no toca helado, pero ya queda menos!
```

Pero... nos podríamos preguntar que diferencia hay entre el `and` y el `or`

- El `and` lo usaremos cuando queramos que se cumplan **TODAS** las condiciones que especificamos
- El `or` lo usaremos cuando queramos que se cumplan **ALGUNA** de las condiciones que especifiquemos

En los ejemplos que hemos visto hasta ahora estamos usando el `and`, pero veamos que pasaría si usaramos el `or`

```
In [80]: print("¿Es hoy viernes?", hoy_es_viernes)
print("-----")

print("¿Está en congelador en marcha?", congelador_en_marcha)
print("-----")

# en este caso chequearemos si es viernes o el congelador está en marcha
if hoy_es_viernes or congelador_en_marcha:
    print('Coge un helado, ¡te lo mereces!')

# en esta condición chequearemos si es viernes o si el congelador no funciona
elif hoy_es_viernes or not congelador_en_marcha:
    print('Sal a la heladería, ¡te lo mereces!')

else:
    print("Todavía no toca helado, pero ya queda menos!")
```

¿Es hoy viernes? True

¿Está en congelador en marcha? False

Coge un helado, ¡te lo mereces!

En este ejemplo, **¿Qué es lo que está pasando?**

Python va a empezar a leer el programa y seguirá los siguientes pasos:

- El `if` :
 - Es la variable `hoy_es_viernes == True` **O** la variable `congelador_en_marcha == True`. En caso de que se cumpla alguna de las condiciones el programa se parará y nos devolverá el mensaje de `Coge un helado`. Este es nuestro caso ya que aunque el congelador esté apagado, es viernes! como esta primera condición se cumple, el programa se para y no sigue ejecutando las siguientes líneas de código.
- En el `elif` :
 - Es la variable `hoy_es_viernes == True` **O** la variable `congelador_en_marcha == False`
- En el `else` :
 - En caso de que no se cumplan ninguna de las condiciones anteriores, se ejecutará la línea de código que está dentro.

Llegados a este punto, os animamos a que vayáis cambiando los valores de las variables `hoy_es_viernes` y `congelador_en_marcha` para ver como van cambiando los resultados.

Repaso: Condicionales if ... elif ... else

- La sintaxis básica de estos condicionales es:

if condicion1:
pasan cosas # puede ser un print, operaciones aritméticas, operaciones con listas, diccionarios, etc.

elif condición2: # si queremos incluir más de una condición. Podremos poner tantos elif como queramos
pasas cosas

else: # no tendremos que poner condiciones nunca. En el else se incluyen todas las condiciones que no se cumplan en los pasos anteriores
pasan cosas

🚨 todas las acciones después de los if, elif, else deben ir indentadas hacia la derecha

- Una condición es una comparación que da como resultado un True o un False . Si la comparación sólo menciona una variable, sin operador, se entiende como == True . De la misma manera not más una variable se entiende como que la variable == False .
- Las condiciones son evaluadas de arriba hacia abajo.
- En el momento que una condición se cumple el programa depara y no evalúa el resto de las condiciones. Por lo tanto, el orden en el que establezcamos nuestras condiciones es importante!!

Para establecer las comparaciones podemos usar los operadores > , < , == , != o las palabras in o not in .

- Podemos incluir varias condiciones en una línea usando las palabras and y/u or .
 - Con el and se tienen que cumplir **TODAS** las condiciones se tienen que cumplir
 - Con el operador or se tiene que cumplir **ALGUNA** de las condiciones.

Estructuras de iteración

while

En Python, los bucles while son una estructura de control que permite ejecutar un bloque de código varias veces mientras se cumpla una determinada condición. El bucle while se repite continuamente mientras la condición especificada sea verdadera. Si la condición es falsa, el bucle while termina y la ejecución continúa con el código que sigue al bucle.

La sintaxis básica del bucle while en Python es la siguiente:

while condición:

bloque de código a ejecutar mientras se cumpla la condición

La condición en el bucle "while" puede ser cualquier expresión booleana, es decir, cualquier expresión que se evalúe como verdadera o falsa. Por ejemplo, podemos utilizar una variable, una comparación, o cualquier otro operador o función que devuelva un valor booleano.

NORMALMENTE USAMOS UN `while` cuando no sabemos cuántas veces vamos a tener que ejecutar el bucle.

Es importante tener en cuenta que, si la condición del bucle "while" nunca se vuelve falsa, el bucle se ejecutará infinitamente, lo que puede llevar a errores y problemas en el programa. Por lo tanto, es importante asegurarse de que la condición se vuelve falsa en algún momento.

Sigamos con el ejemplo de las peras que hemos ido usando en este jupyter. Hasta ahora teníamos una variable donde teníamos el número de peras que teníamos en la nevera. Pero seamos sinceras, eso no era del todo real. Lo ideal sería que cada vez que nos comieramos una pera, restaramos esa pera a la cantidad que teníamos. Y que cuando me quedarán menos de dos peras nos lo añadiera a la lista de la compra.

Veamos como podríamos hacerlo con código de Python:

```
In [81]: # Los primero que vamos a hacer es definir una variable, que será nuestro Límite
limite_peras = 2

print("El límite de peras para ir al supermercado es:", limite_peras)
print("-----")

print("El número de peras que tenemos en la nevera es:", numero_peras)
print("-----")

# Lo que estamos haciendo aquí es establecer la condición usando el while, donde
# este código parará cuando el número de peras sea igual o menor a el Límite de
while numero_peras > limite_peras:
    print(f'El número de peras que tenemos en la nevera es {numero_peras}')

    # cada vez que me como una pera La quito de la cantidad de peras que tenemos e
    numero_peras -= 1    # recordemos que sinónimo de i = i + 1
```

El límite de peras para ir al supermercado es: 2

El número de peras que tenemos en la nevera es: 11

El número de peras que tenemos en la nevera es 11
El número de peras que tenemos en la nevera es 10
El número de peras que tenemos en la nevera es 9
El número de peras que tenemos en la nevera es 8
El número de peras que tenemos en la nevera es 7
El número de peras que tenemos en la nevera es 6
El número de peras que tenemos en la nevera es 5
El número de peras que tenemos en la nevera es 4
El número de peras que tenemos en la nevera es 3

¿Qué es lo que está pasando?

- Empezamos el bucle `while` y se va a evaluar si el `numero_peras` es mayor que el `limite_peras`, es decir, ¿es el `numero_peras` (11) mayor que `limite_peras(2)`?
 - En caso de que si, imprimimos el número de peras que tenemos y restamos una pera a la cantidad total. Es decir, después de este primer while y la resta, el número de peras será 10.
- Cómo la condición se ha cumplido se sigue ejecutando el código. Volvemos a hacer la pregunta ¿es el `numero_peras` (en este caso 10, por que en el paso anterior ya restamos una) mayor que `limite_peras(2)`?
 - En caso de que si, imprimimos el número de peras que tenemos y restamos una pera a la cantidad total. Es decir, después de este primer while y la resta, el número de peras será 9.
- Esto va a ocurrir hasta que la pregunta ¿es el `numero_peras` mayor que `limite_peras`? sea `False`. En ese caso el bucle `while` se parará y terminará el programa.

Veamos otro ejemplo, pero vamos a complicarlo un poquito. En el ejemplo anterior solo hemos visto un `while` sin incluir condiciones dentro. Pero lo podemos hacer! Dentro de un `while` podremos incluir `if ... elif ... else ...`

En este caso vamos a crear un programa que mientras que nuestra condición sea `True`, le preguntaremos al usuario que seleccione una opción de las que le pasemos. En caso de que el usuario pase una opción no válida, cambiaremos el valor a `False` y el programa se parará.

```
In [86]: # definimos una variable con el valor de True.
opcion = True

print("El valor de la variable 'opcion' es:", opcion)
print("-----")
# especificamos que si la variable opcion es True, entonces seguiremos ejecutando
while opcion == True:

    # pedimos al usuario que introduzca una de las opciones que tenemos
    selecciona_opcion = input("Por favor selecciona una letra de las siguientes:

    # empezamos poniendo condicionales if. Si la opción del usuario es "a", print
    if selecciona_opcion == "a":
        print("Has seleccionado 'a'!")

    # hacemos lo mismo para el resto de las opciones que le pasamos al usuario
    elif selecciona_opcion == "b":
        print("Has seleccionado 'b'!")
    elif selecciona_opcion == "c":
        print("Has seleccionado 'c'!")

    # en caso de que el usuario no haya seleccionado ninguna de las opciones que
    else:
        print("Has seleccionado", selecciona_opcion, "! la cual no es una opción

    # en caso de que ocurra esta condición, nuestra variable opcion cambiará
    print()
```

```
opcion = False
```

```
El valor de la variable 'opcion' es: True
-----
Has seleccionado 'a'!
Has seleccionado 'b'!
Has seleccionado 'c'!
Has seleccionado t ! la cual no es una opción inválida 😊
```

Os animamos a que ejecuteis esta celda y probéis vosotras para terminar de interiorizar este ejercicio.