Lista por comprensión:

La lista por comprensión ayuda a crear una nueva lista a partir de objetos iterables existentes (listas, cadenas y tuplas) aplicando una expresión/condición a cada elemento para determinar qué elementos se incluirán en la lista resultante

La sintaxis (para una lista) es la siguiente:

```
<new_list> = [<expression> for <item> in <iterable>]
```

Esta sintaxis permite un código más legible, eficiente y conciso que los bucles *for* en modelos sencillos, es decir, donde la expresión no es compleja. Por ejemplo, tenemos una lista y queremos crear una nueva donde los elementos sean el doble de los elementos iniciales.

Usando el bucle *for …. in* lo haríamos así:

Ejemplo 1

```
nums = [1, 2, 3, 4, 5]
numeros_dobles = []

for num in nums:
    numeros_dobles.append(num*2)

print(numeros_dobles) #Output: [1, 4, 6, 8,10]
```

Utilizando el método de lista por comprensión el código quedaría así:

Ejemplo 2

```
nums = [1, 2, 3, 4, 5]

numeros_dobles = [num*2 for num in nums] # indicamos que cada elemento de la lista se multiplique por dos.

print(numeros_dobles) #Output: [1, 4, 6, 8,10]
```

También se puede utilizar este método con condicionales. En el siguiente ejemplo se busca extraer sólo los números impares que hay en el rango de valores del 0 al 9.

El código con el bucle *for* sería:

Ejemplo 3

```
numeros_impares = []
for num in range(10):
    if num % 2 != 0: # si el resto de la división del número entre 2 es distinto a cero entonces el número es impar
    numeros_impares.append(num)

print(numeros_impares) # Output: [1, 3, 5, 7, 9]
```

Con el método de la lista por comprehensión, el código sería:

Ejemplo 4

```
numeros_impares = [num for num in range(10) if num % 2 != 0]

print(numeros_impares) # Output: [1, 3, 5, 7, 9]
```